

$$\langle c | Q | c \rangle$$

A Course in Quantum Computing

(for the
Community College)

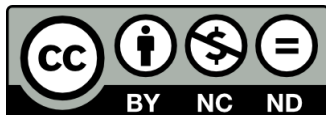
Volume 1

Michael Loceff
Foothill College
<mailto:LoceffMichael@fhda.edu>

© 2015 Michael Loceff
All Rights Reserved

This work is licensed under the Creative Commons
Attribution-NonCommercial-NoDerivatives 4.0 International License.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by-nc-nd/4.0/>.



Contents

0	Introduction	21
0.1	Welcome to Volume One	21
0.1.1	About this Volume	21
0.1.2	About this Introduction	21
0.2	Bits and Qubits	22
0.2.1	More Information than Zero and One	22
0.2.2	The Probabilistic Nature of Qubits	22
0.2.3	Quantum Mechanics – The Tool that Tames the Beast	24
0.2.4	Sneak Peek at the Coefficients α and β	24
0.3	The Promise of Quantum Computing	25
0.3.1	Early Results	25
0.3.2	The Role of Computer Scientists	26
0.4	The Two Sides of Quantum Computer Science	26
0.4.1	Circuit Design	27
0.4.2	Algorithms	28
0.5	Perspective	29
0.6	Navigating the Topics	29
1	Complex Arithmetic	31
1.1	Complex Numbers for Quantum Computing	31
1.2	The Field of Complex Numbers	32
1.2.1	The Real Numbers Just Don't Cut It	32
1.2.2	The Definition of \mathbb{C}	32
1.2.3	The Complex Plane	33
1.2.4	Operations on Complex Numbers	34
1.2.5	\mathbb{C} is a Field	36
1.3	Exploring the Complex Plane	36

1.3.1	Complex Numbers as Ordered Pairs of Real Numbers	36
1.3.2	Real and Imaginary Axes and Polar Representation	36
1.3.3	Complex Conjugate and Modulus	37
1.4	Transcendental Functions and Their Identities	39
1.4.1	The Complex Exponential Function Part 1: Pure Imaginary Case	39
1.4.2	Real $\sin()$ and $\cos()$ in Terms of the Complex Exponential . .	42
1.4.3	Complex Exponential Part 2: Any Complex Number	42
1.4.4	The Complex Trigonometric Functions	43
1.4.5	Polar Relations Expressed Using the Exponential	43
1.5	Roots of Unity	45
1.5.1	N Distinct Solutions to $z^N = 1$	45
1.5.2	Euler's Identity	47
1.5.3	Summation Notation	48
1.5.4	Summing Roots-of-Unity and the Kronecker Delta	49
2	Real Vector Spaces	52
2.1	Vector Spaces for Quantum Computing	52
2.2	Vectors and Vector Spaces	53
2.2.1	Standard Equipment: The Axioms	53
2.2.2	Optional Equipment: Inner Products, Modulus and Orthogonality	56
2.2.3	A Bigger Vector Space: \mathbb{R}^3	59
2.2.4	Preview of a Complex Vector Space: \mathbb{C}^2	60
2.3	Bases for a Vector Space	60
2.3.1	Linear Combination (or Superposition)	60
2.3.2	The Notion of Basis	61
2.3.3	Coordinates of Vectors	66
2.3.4	Independence of Basis (or <i>Not?</i>)	68
2.4	Subspaces	69
2.5	Higher Dimensional Vector Spaces	69
2.6	More Exercises	72
3	Matrices	73
3.1	Matrices in Quantum Computing	73
3.2	Definitions	74
3.2.1	Notation	74

3.3	Matrix Multiplication	74
3.3.1	Row \times Column	75
3.3.2	Definition of Matrix Multiplication	75
3.3.3	Product of a Vector by a Matrix	77
3.4	Matrix Transpose	78
3.5	Matrix Addition and Scalar Multiplication	79
3.6	Identity Matrix and Zero Matrix	79
3.7	Determinants	80
3.7.1	Determinant of a 2×2 Matrix	80
3.7.2	Determinant of a 3×3 Matrix	81
3.7.3	Determinant of an $n \times n$ Matrix	82
3.7.4	Determinants of Products	82
3.8	Matrix Inverses	83
3.8.1	Invertibility	83
3.8.2	Non-Singularity and the Big Inverse Theorem	84
3.9	Matrix Equations and Cramer's Rule	84
3.9.1	Systems of Linear Equations	84
3.9.2	Cramer's Rule	86
3.9.3	Completion of the Proof of the Big Inverse Theorem	89
4	Hilbert Space	92
4.1	Complex Vector Spaces for Quantum Computing	92
4.1.1	The Vector Nature of a Qubit	92
4.1.2	The Complex Nature of a Qubit	93
4.2	The Complex Vector Space, \mathbb{C}^n	93
4.3	The Complex Inner Product	94
4.3.1	Norm and Distance	96
4.3.2	Expansion Coefficients	97
4.4	Hilbert Space	100
4.4.1	Definitions	100
4.4.2	Old Friends and New Acquaintances	101
4.4.3	Some Useful Properties of Hilbert Spaces	102
4.5	Rays, Not Points	103
4.5.1	Modeling Quantum Systems	103
4.5.2	$\mathbf{0}$ is not a Quantum State	106

4.5.3	Why?	107
4.6	Almost There	108
5	Linear Transformations	109
5.1	Linear Transformations for Quantum Computing	109
5.1.1	A Concept More Fundamental Than the Matrix	109
5.1.2	The Role of Linear Transformations in Quantum Computing	110
5.2	Definitions and Examples	110
5.2.1	Actions as well as Name Changes	110
5.2.2	Formal Definition of Linear Transformation	111
5.3	The Special Role of Bases	114
5.3.1	Application: Rotations in Space	115
5.4	The Matrix of a Linear Transformation	116
5.4.1	From Matrix to Linear Transformation	116
5.4.2	From Linear Transformation to Matrix	117
5.4.3	Dependence of a Matrix on Basis	119
5.4.4	The Transformation in an Orthonormal Basis	122
5.5	Some Special Linear Transformations for Quantum Mechanics	127
5.5.1	The Adjoint of a Matrix	127
5.5.2	Unitary Operators	128
5.5.3	Hermitian Operators	133
5.6	Enter the Quantum World	134
6	The Experimental Basis of Quantum Computing	135
6.1	The Physical Underpinning for Spin 1/2 Quantum Mechanics	135
6.2	Physical Systems and Measurements	135
6.2.1	Quantum Mechanics as a Model of Reality	135
6.2.2	The Physical System, \mathcal{S}	136
6.2.3	Electron Spin as a Testbed for Quantum Mechanics	136
6.3	A Classical Attempt at Spin 1/2 Physics	137
6.3.1	An Imperfect Picture of Spin	137
6.3.2	A Naive Quantitative Definition of Electron Spin	138
6.3.3	Spherical Representation	139
6.4	Refining Our Model: Experiment #1	140
6.4.1	The Experiment	140
6.4.2	The Actual Results	142

6.4.3	The Quantum Reality	142
6.4.4	A Follow-Up to Experiment #1	143
6.4.5	Results of the Follow-Up	143
6.4.6	Quantum Mechanics Lesson #1	144
6.4.7	First Adjustment to the Model	145
6.5	Refining Our Model: Experiment #2	145
6.5.1	The Experiment	145
6.5.2	The Actual Results	147
6.5.3	The Quantum Reality	147
6.5.4	A Follow-Up to Experiment #2	147
6.5.5	Results of the Follow-Up	148
6.5.6	Quantum Mechanics Lesson #2	148
6.5.7	Second Adjustment to the Model	149
6.6	Refining Our Model: Experiment #3	150
6.6.1	What we Know	150
6.6.2	The Experiment	151
6.6.3	The Actual Results	153
6.6.4	The Quantum Reality	153
6.6.5	Quantum Mechanics Lesson #3	154
6.6.6	Third Adjustment to the Model	154
6.7	Onward to Formalism	155
7	Time Independent Quantum Mechanics	156
7.1	Quantum Mechanics for Quantum Computing	156
7.2	The Properties of Time Independent Quantum Mechanics	156
7.2.1	The State Space of a Physical System	157
7.3	The First Postulate of Quantum Mechanics	158
7.3.1	Trait #1 (The State Space)	158
7.3.2	The Fundamental State Space for Quantum Computing	159
7.3.3	Why Does a Projective \mathcal{H} model Spin-1/2?	160
7.4	The Second Postulate of Quantum Mechanics	161
7.4.1	Trait #2 (The Operator for an Observable)	161
7.4.2	The Observable S_z	162
7.5	The Third Postulate of Quantum Mechanics	163
7.5.1	Trait #3 (The Eigenvalues of an Observable)	163

7.5.2	Eigenvectors and Eigenvalues	164
7.5.3	The Eigenvalues and Eigenvectors of \mathbf{S}_z	165
7.6	Computing Eigenvectors and Eigenvalues	166
7.6.1	The Eigenvalues and Eigenvectors of \mathbf{S}_y	167
7.6.2	The Eigenvalues and Eigenvectors of \mathbf{S}_x	169
7.6.3	Summary of Eigenvectors and Eigenvalues for Spin-1/2 Observables	169
7.7	Observables and Orthonormal Bases	170
7.7.1	Trait #4 (Real Eigenvalues and Orthonormal Eigenvectors) . .	171
7.7.2	Using the \mathbf{x} - or \mathbf{y} -Basis	172
7.7.3	General States Expressed in Alternate Bases	173
7.8	The Completeness (or Closure) Relation	175
7.8.1	Orthonormal Bases in Higher Dimensions	175
7.8.2	Trait #5 (Closure Relation)	176
7.9	The Fourth Postulate of Quantum Mechanics	176
7.9.1	Trait #6 (Probability of Outcomes)	176
7.10	The Fifth Postulate of Quantum Mechanics	179
7.10.1	Trait #7 (Post-Measurement Collapse)	180
7.11	Summary of What Quantum Mechanics Tells Us About a System . .	181
7.12	Dirac's Bra-ket Notation	181
7.12.1	Kets and Bras	181
7.12.2	The Adjoint of an Operator	185
7.12.3	The Adjoint Conversion Rules	186
7.12.4	Trait #8 (Adjoint Conversion Rules)	187
7.13	Expectation Values	187
7.13.1	The Mean of the Experiments	188
7.13.2	Defining Expectation Value	188
7.13.3	Computing Expectation Value	190
7.13.4	Trait #9 (Computing an Expectation Value)	190
7.13.5	Expectation Values in Spin-1/2 Systems	191
7.14	It's About Time	192
8	Time Dependent Quantum Mechanics	194
8.1	Time Evolution in Quantum Computing	194
8.2	The Hamiltonian	194

8.2.1	Total Energy	194
8.2.2	From Classical Hamiltonian to Quantum Hamiltonian	195
8.3	The Hamiltonian for a Spin-1/2 System	195
8.3.1	A Classical Hamiltonian	195
8.3.2	A Quantum Hamiltonian	197
8.4	The Energy Eigenkets	197
8.4.1	Relationship Between H and S_z	197
8.4.2	Allowable Energies	198
8.5	Sixth Postulate of Quantum Mechanics	199
8.5.1	The Schrödinger Equation	199
8.5.2	The Evolution of Spin in a Constant Magnetic Field	200
8.5.3	Stationary States	203
8.5.4	General (Non-Stationary) States	203
8.5.5	General Technique for Computing Time-Evolved States	204
8.6	Larmor Precession	207
8.6.1	The Time-Evolved Spin State in a Uniform \mathbf{B} -Field	207
8.6.2	Evolution of the Spin Expectation Values	209
8.6.3	Summary of Larmor Precession	213
8.7	The End and the Beginning	214
9	The Qubit	215
9.1	Bits and Qubits as Vector Spaces	215
9.2	Classical Computation Models – Informal Approach	215
9.2.1	Informal Definition of Bits and Gates	215
9.3	Classical Computation Models – Formal Approach	218
9.3.1	A Miniature Vector Space	218
9.3.2	Formal Definition of a (Classical) Bit	220
9.3.3	Formal Definition of a (Classical) Logical Operator	221
9.4	The Qubit	225
9.4.1	Quantum Bits	225
9.4.2	Quantum Bit Values	227
9.4.3	Usual Definition of Qubit and its Value	227
9.4.4	Key Difference Between Bits and Qubits	228
9.5	Quantum Operators (Unary)	230
9.5.1	Definition and Notation	230

9.5.2	Case Study – Our First Quantum Gate, QNOT	230
9.5.3	The Phase Flip, Z	236
9.5.4	The Bit-and-Phase Flip Operator, Y	238
9.5.5	The Hadamard Gate, H	240
9.5.6	Phase-Shift Gates, S , T and R_θ	243
9.6	Putting Unary Gates to Use	244
9.6.1	Basis Conversion	244
9.6.2	Combining Gates	245
9.7	The Bloch Sphere	248
9.7.1	Introduction	248
9.7.2	Rewriting $ \psi\rangle$	248
9.7.3	The Expectation Vector for $ \psi\rangle$	249
9.7.4	Definition of the Bloch Sphere	250
10	Tensor Products	251
10.1	Tensor Product for Quantum Computing	251
10.2	The Tensor Product of Two Vector Spaces	252
10.2.1	Definitions	252
10.2.2	Tensor Coordinates from Component-Space Coordinates	260
10.3	Linear Operators on the Tensor Product Space	266
10.3.1	Separable Operators	266
10.3.2	The Matrix of a Separable Operator	268
10.3.3	The Matrix of a General Operator	270
10.3.4	Food for Thought	271
11	Two Qubits and Binary Quantum Gates	272
11.1	The Jump from One to Two	272
11.2	The State Space for Two Qubits	273
11.2.1	Definition of a Two Quantum Bit (“Bipartite”) System	273
11.2.2	The Preferred Bipartite CBS	274
11.2.3	Separable Bipartite States	275
11.2.4	Alternate Bipartite Bases	275
11.2.5	Non-Separable Bipartite Tensors	278
11.2.6	Usual Definition of Two Qubits and their Values	279
11.3	Fundamental Two-Qubit Logic Gates	279
11.3.1	Binary Quantum Operators	279

11.3.2	General Learning Example	280
11.3.3	Quantum Entanglement	285
11.3.4	The Controlled-NOT (CNOT) Gate	287
11.3.5	The Second-Order Hadamard Gate	297
11.4	Measuring Along Alternate Bases	305
11.4.1	Measuring Along the \mathbf{x} -Basis	306
11.4.2	Measuring Along <i>any</i> Separable Basis	306
11.4.3	The Entire Circuit Viewed in Terms of an Alternate Basis	308
11.4.4	Non-Separable Basis Conversion Gates	308
11.5	Variations on the Fundamental Binary Qubit Gates	309
11.5.1	Other Controlled Logic Gates	309
11.5.2	More Separable Logic Gates	315
11.6	The Born Rule and Partial Collapse	318
11.6.1	The Born Rule for a Two-Qubit System	319
11.7	Multi-Gate Circuits	322
11.7.1	A Circuit that Produces Bell States	322
11.7.2	A Circuit that Creates an Upside-Down CNOT	326
11.8	More than Two Qubits	328
11.8.1	Order-3 Tensor Products	328
11.8.2	Three Qubit Systems	329
11.8.3	Tripartite Bases and Separable States	329
11.8.4	End of Lesson	337
12	First Quantum Algorithms	338
12.1	Three Algorithms	338
12.2	Superdense Coding	338
12.2.1	Sending Information by Qubit	338
12.2.2	The Superdense Coding Algorithm	339
12.3	Quantum Teleportation	345
12.3.1	The Quantum Teleportation Algorithm	348
12.4	Introduction to Quantum Oracles	357
12.4.1	Boolean Functions and Reversibility	357
12.4.2	The Quantum Oracle of a Boolean Function	360
12.5	Deutsch's Problem	363
12.5.1	Definitions and Statement of the Problem	363

12.5.2	Deutsch's Algorithm	365
12.6	n Qubits and More Algorithms	371
13	Multi-Qubit Systems and Algorithms	372
13.1	Moving Up from 2 Qubits to n Qubits	372
13.2	General Tensor Products	372
13.2.1	Recap of Order-2 Tensor Products	372
13.2.2	Recap of Order-3 Tensor Products	373
13.2.3	Higher Order Tensor Products	374
13.3	n -Qubit Systems	377
13.3.1	Recap of Three Qubits	377
13.3.2	Three Qubit Logic Gates; the Toffoli Gate	378
13.3.3	n Qubits	381
13.3.4	n Qubit Logic Gates	383
13.3.5	Oracles for n Qubit Functions	389
13.4	Significant Deterministic Speed-Up: The Deutsch-Jozsa Problem	390
13.4.1	Deutsch-Jozsa Algorithm	391
13.4.2	Quantum vs. Classical Time Complexity	396
13.4.3	Alternate Proof of the Deutsch-Jozsa Algorithm	399
13.5	True Non-Deterministic Speed-Up: The Bernstein-Vazirani Problem	402
13.5.1	The Bernstein-Vazirani Algorithm	403
13.6	Generalized Born Rule	406
13.6.1	"Trait #15" (Generalized Born Rule for $(n + m)$ th order States)	406
13.7	Towards Advanced Quantum Algorithms	408
14	Probability Theory	409
14.1	Probability in Quantum Computing	409
14.1.1	Probability for Classical Algorithms	409
14.1.2	Probability for Quantum Algorithms	409
14.2	The Essential Vocabulary: Events vs. Probabilities	410
14.2.1	Events	410
14.2.2	Probabilities	411
14.3	The Quantum Coin Flip	411
14.4	Experimental Outcomes and the Sample Space	412
14.4.1	Outcomes	413
14.4.2	Requirements when Defining Outcomes	414

14.4.3	An Incorrect Attempt at Defining Outcomes	415
14.4.4	Events	415
14.4.5	The Sample Space	416
14.4.6	Set Operations	417
14.5	Alternate Views of the Ten Qubit Coin Flip	418
14.6	Mod-2 Vectors in Probability Computations for Quantum Algorithms	420
14.6.1	Ordinary Linear Independence and Spanning	421
14.6.2	Linear Independence and Spanning in the Mod-2 Sense	421
14.6.3	Probability Warm-Up: Counting	424
14.7	Fundamental Probability Theory	426
14.7.1	The Axioms	426
14.7.2	Assigning Probabilities in Finite, Equiprobable, Sample Spaces	427
14.8	Big Theorems and Consequences of the Probability Axioms	430
14.8.1	Unions	430
14.8.2	Conditional Probability and Bayes' Law	430
14.8.3	Statistical Independence	431
14.8.4	Other Forms and Examples	433
14.9	Wedge and Vee Notation and Lecture Recap	435
14.9.1	Disjoint Events	435
14.9.2	Partition of the Sample Space by Complements.	435
14.9.3	Bayes' Law.	435
14.9.4	Statistical Independence	436
14.10	Application to Deutsch-Jozsa	436
14.10.1	Sampling <i>with</i> Replacement	436
14.10.2	Analysis Given a Balanced f	437
14.10.3	Completion of Sampling with Replacement for Unknown f . .	438
14.10.4	Sampling <i>without</i> Replacement	439
14.11	A Condition for Constant Time Complexity in Non-Deterministic Algorithms	442
14.11.1	Non-Deterministic Algorithms	443
14.11.2	Preview of Time Complexity – An Algorithm's Dependence on Size	443
14.11.3	Looping Algorithms	443
14.11.4	Probabilistic Algorithms with Constant Time Complexity . . .	444
14.11.5	A Simple Test for for Constant Time Algorithms	444

15 Computational Complexity	447
15.1 Computational Complexity in Quantum Computing	447
15.2 An Algorithm's Sensitivity to Size	447
15.2.1 Some Examples of Time Complexity	447
15.2.2 Time Complexity vs. Space Complexity	449
15.2.3 Notation	449
15.3 <i>Big-O</i> Growth	450
15.3.1 Conflicting Ways to Measure an Algorithm	450
15.3.2 Definition of <i>Big-O</i>	451
15.3.3 Common Terminology for Certain <i>Big-O</i> Growth Rates	451
15.3.4 Factors and Terms We Can Ignore	452
15.4 Ω Growth	454
15.4.1 Definition of Ω	454
15.5 Θ Growth	454
15.6 <i>Little-o</i> Growth	455
15.7 Easy vs. Hard	455
15.8 Wrap-Up	455
16 Computational Basis States and Modular Arithmetic	456
16.1 Different Notations Used in Quantum Computing	456
16.2 Notation and Equivalence of Three Environments	456
16.2.1 First Environment – n -qubit Hilbert Space, $\mathcal{H}_{(n)}$	456
16.2.2 The Second Environment: The Finite Group $(\mathbb{Z}_2)^n$	458
16.2.3 The Third Environment: The Finite Group \mathbb{Z}_{2^n} with \oplus Arithmetic	462
16.2.4 Interchangeable Notation of $\mathcal{H}_{(n)}$, $(\mathbb{Z}_2)^n$ and $(\mathbb{Z}_{2^n}, \oplus)$	464
17 Quantum Oracles	467
17.1 Higher Dimensional Oracles and their Time Complexity	467
17.2 Simplest Oracle: a Boolean Function of One Bit	468
17.2.1 Circuit and Initial Remarks	468
17.2.2 A Two-Qubit Oracle's Action on the CBS	470
17.2.3 Case Study #1: $f(x) \equiv 1$	470
17.2.4 Case Study #2: $f(x) \equiv x$	472
17.2.5 Remaining Cases #3 and #4	472
17.3 Integers Mod-2 Review	473

17.3.1	The Classical f at the Heart of an Oracle	473
17.3.2	Mod-2 Notation for $\{0, 1\}^n$	473
17.3.3	\oplus Inside the Ket	474
17.4	Intermediate Oracle: f is a Boolean Function of a Multiple Input Bits	475
17.4.1	Circuit	475
17.4.2	An $(n + 1)$ -Qubit Oracle's Action on the CBS	475
17.4.3	Analyzing U_f for $x = 0$	476
17.4.4	Analyzing U_f for General x	477
17.5	Advanced Oracle: f is a Multi-Valued function of a Multiple Input Bits	480
17.5.1	Circuit and Vocabulary	480
17.5.2	Smallest Advanced Oracle: $m = 2$ (Range of $f \subseteq \mathbb{Z}_2^2$)	480
17.5.3	The 4×4 Sub-Matrix for a Fixed x	480
17.5.4	Easy Way to "See" Unitarity	486
17.5.5	The General Advanced Oracle: Any m (Range of $f \subseteq \mathbb{Z}_2^m$)	487
17.6	The Complexity of a Quantum Algorithm Relative to the Oracle	488
18	Simon's Algorithm for Period-Finding	490
18.1	The Importance of Simon's Algorithm	490
18.2	Periodicity	491
18.2.1	Ordinary Periodicity	491
18.2.2	$(\mathbb{Z}_2)^n$ Periodicity	491
18.2.3	\mathbb{Z}_2^n Periodicity	492
18.2.4	1-to-1, 2-to-1 and n -to-1	495
18.3	Simon's Problem	497
18.4	Simon's Quantum Circuit Overview and the Master Plan	498
18.4.1	The Circuit	498
18.4.2	The Plan	499
18.5	The Circuit Breakdown	500
18.6	Circuit Analysis Prior to Conceptual Measurement: Point B	500
18.6.1	Hadamard Preparation of the \mathbf{A} Register	500
18.6.2	The Quantum Oracle on CBS Inputs	501
18.6.3	The Quantum Oracle on Hadamard Superposition Inputs	503
18.6.4	Partitioning the Domain into Cosets	504
18.6.5	Rewriting the Output of the Oracle's \mathbf{B} Register	505
18.7	Analysis of the Remainder of the Circuit: Measurements	506

18.7.1	Hypothetical Measurement of the \mathbf{B} Register	506
18.7.2	Effect of a Final Hadamard on \mathbf{A} Register	507
18.7.3	The Orthogonality of \mathbf{A} Register Output Relative to the Unknown Period a	509
18.7.4	Foregoing the Conceptual Measurement	511
18.8	Circuit Analysis Conclusion	512
18.9	Simon's Algorithm	513
18.9.1	Producing $n - 1$ Linearly Independent Vectors	513
18.9.2	The Algorithm	514
18.9.3	Strange Behavior	516
18.10	Time Complexity of the Quantum Algorithm	518
18.10.1	Producing $n - 1$ Linearly-Independent \mathbf{w}_k in Polynomial Time – Argument 1	518
18.10.2	Proof of Theorem Used by Argument 1	518
18.10.3	Summary of Argument 1	523
18.10.4	Producing $n - 1$ Linearly-Independent \mathbf{w}_k in Polynomial Time – Argument 2	524
18.10.5	Proof of Theorem Used by Argument 2	524
18.10.6	Summary of Argument 2	527
18.10.7	Discussion of the Two Proofs' Complexity Estimates	527
18.11	The Hidden Classical Algorithms and Their Cost	528
18.11.1	Unaccounted for Steps	528
18.12	Solving Systems of Mod-2 Equations	529
18.12.1	Gaussian Elimination and Back Substitution	529
18.12.2	Gaussian Elimination	530
18.12.3	Back Substitution	534
18.12.4	The Total Cost of the Classical Techniques for Solving Mod-2 Systems	535
18.13	Applying GE and Back Substitution to Simon's Problem	535
18.13.1	Linear Independence	536
18.13.2	Completing the Basis with an n th Vector Not Orthogonal to a	537
18.13.3	Using Back-Substitution to Close the Deal	540
18.13.4	The Full Cost of the Hidden Classical Algorithms	540
18.14	Adjusted Algorithm	540
18.14.1	New Linear Independence Step	540
18.14.2	New Solution of System Step	543

18.14.3	Cost of Adjusted Implementation of Simon's Algorithm	543
18.15	Classical Complexity of Simon's Problem	544
18.15.1	Classical Deterministic Cost	544
18.15.2	Classical Probabilistic Cost	545
19	Real and Complex Fourier Series	548
19.1	The Classical Path to Quantum Fourier Transforms	548
19.2	Periodic Functions and Their Friends	549
19.2.1	Periodic Functions over \mathbb{R}	549
19.2.2	Functions with Bounded Domain or Compact Support	550
19.2.3	The Connection Between Periodicity and Bounded Domain . .	551
19.3	The Real Fourier Series of a Real Period Function	552
19.3.1	Definitions	552
19.3.2	Interpretation of the Fourier Series	553
19.3.3	Example of a Fourier Series	555
19.4	The Complex Fourier Series of a Periodic Function	560
19.4.1	Definitions	560
19.4.2	Computing The Complex Fourier Coefficients	561
19.5	Periods and Frequencies	563
19.5.1	The Frequency of any Periodic Function	563
19.5.2	Ordinary Frequency	563
19.5.3	Angular Frequency	564
20	The Continuous Fourier Transform	566
20.1	From Series Transform	566
20.2	Motivation and Definitions	566
20.2.1	Non-Periodic Functions	566
20.2.2	Main Result of Complex Fourier Series	567
20.2.3	Tweaking the Complex Fourier Series	567
20.2.4	Definition of the Fourier Transform	568
20.2.5	The Inverse Fourier Transform	568
20.2.6	Real vs. Complex, Even vs. Odd	569
20.2.7	Conditions for a function to Possess a Fourier Transform . . .	570
20.3	Learning to Compute	571
20.3.1	Example 1: Rectangular Pulse	572
20.3.2	Example 2: Gaussian	572

20.4	Interlude: The Delta Function	573
20.4.1	Characterization of the Delta Function	573
20.4.2	The Delta Function as a Limit of Rectangles	574
20.4.3	The Delta Function as a Limit of Exponentials	575
20.4.4	Sifting Property of the Delta Function	575
20.5	Fourier Transforms Involving $\delta(x)$	576
20.5.1	Example 3: A Constant	576
20.5.2	Example 4: A Cosine	578
20.5.3	Example 5: A Sine	578
20.6	Properties of the Fourier Transform	579
20.6.1	Translation Invariance	579
20.6.2	Plancherel's Theorem	580
20.6.3	Convolution	580
20.6.4	The Convolution Theorem	581
20.7	Period and Frequency in Fourier Transforms	581
20.8	Applications	582
20.8.1	What's the \mathcal{FT} Used For?	582
20.8.2	The Uncertainty Principle	584
20.9	Summary	584
21	The Discrete and Fast Fourier Transforms	586
21.1	From Continuous to Discrete	586
21.2	Motivation and Definitions	587
21.2.1	Functions Mapping $\mathbb{Z}_N \rightarrow \mathbb{C}$	587
21.2.2	Defining the \mathcal{DFT}	588
21.2.3	$\mathcal{DFT}^{(N)}$ Evaluated Outside \mathbb{Z}_N	592
21.3	Matrix Representation of the \mathcal{DFT}	593
21.4	Properties of \mathcal{DFT}	594
21.4.1	Convolution of Two Vectors	594
21.4.2	Translation Invariance (Shift Property) for Vectors	594
21.4.3	Computational Complexity of \mathcal{DFT}	595
21.5	Period and Frequency in Discrete Fourier Transforms	595
21.6	A Cost-Benefit Preview of the \mathcal{FFT}	598
21.6.1	Benefit	598
21.6.2	Cost	598

21.7	Recursive Equation for \mathcal{FFT}	598
21.7.1	Splitting f into f^{even} and f^{odd}	599
21.7.2	N th Order \mathcal{DFT} in Terms of Two $(N/2)$ th Order \mathcal{DFT} s	599
21.7.3	Danielson-Lanczos Recursion Relation	600
21.7.4	Code Samples: Recursive Algorithm	601
21.8	A Non-Recursive, $N \log N$ Solution that Defines the \mathcal{FFT}	604
21.8.1	The High-Level \mathcal{FFT} Method	604
21.8.2	Bit-Reversal	605
21.8.3	Rebuilding from the Bit-Reversed Array	610
21.8.4	Normalization	615
21.8.5	Overall Complexity	615
21.8.6	Software Testing	615
22	The Quantum Fourier Transform	616
22.1	From Classical Fourier Theory to the \mathcal{QFT}	616
22.2	Definitions	616
22.2.1	From \mathbb{C}^{2^n} to $\mathcal{H}_{(n)}$	616
22.2.2	Approaches to Operator Definition	617
22.2.3	Review of the Hadamard Operator	618
22.2.4	Defining the \mathcal{QFT}	619
22.3	Features of the \mathcal{QFT}	624
22.3.1	Shift Property	624
22.3.2	A Comparison between \mathcal{QFT} and H	624
22.3.3	The Quantum Fourier Basis	625
22.4	The \mathcal{QFT} Circuit	626
22.4.1	Notation	626
22.4.2	The Math that Leads to the Circuit: Factoring the \mathcal{QFT}	627
22.4.3	The \mathcal{QFT} Circuit from the Math: $n = 3$ Case Study	634
22.4.4	The \mathcal{QFT} Circuit from the Math: General Case	641
22.5	Computational Complexity of \mathcal{QFT}	641
23	Shor's Algorithm	644
23.1	The Role of Shor's Algorithms in Computing	644
23.1.1	Context for The Algorithms	644
23.1.2	Period Finding and Factoring	644
23.1.3	The Period Finding Problem and its Key Idea	645

23.2	Injective Periodicity	648
23.2.1	Functions of the Integers	648
23.2.2	Functions of the Group \mathbf{Z}_M	649
23.2.3	Discussion of Injective Periodicity	649
23.3	Shor's Periodicity Problem	650
23.3.1	Definitions and Recasting the Problem	652
23.3.2	The $\mathbf{Z}_N - (\mathbf{Z}_2)^n$ – CBS Connection	654
23.4	Shor's Quantum Circuit Overview and the Master Plan	655
23.4.1	The Circuit	655
23.4.2	The Plan	655
23.5	The Circuit Breakdown	657
23.6	Circuit Analysis Prior to Conceptual Measurement: Point B	657
23.6.1	The Hadamard Preparation of the A register	657
23.6.2	The Quantum Oracle	658
23.6.3	The Quantum Oracle on Hadamard Superposition Inputs	659
23.7	Fork-in-the Road: An Instructional Case Followed by the General Case	659
23.8	Intermezzo – Notation for GCD and Coprime	660
23.8.1	Greatest Common Divisor	660
23.8.2	Coprime (Relatively Prime)	660
23.9	First Fork: <i>Easy Case</i> ($a N$)	661
23.9.1	Partitioning the Domain into Cosets	661
23.9.2	Rewriting the Output of the Oracle	662
23.9.3	Implication of a Hypothetical Measurement of the B register Output	663
23.9.4	Effect of a Final QFT on the A Register	665
23.9.5	Computation of Final Measurement Probabilities (<i>Easy Case</i>)	667
23.9.6	STEP I: Identify a Special Set of a Elements, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ of Certain Measurement Likelihood	667
23.9.7	Step II: Observe that Each cm Will be Measured with Equal Likelihood	668
23.9.8	Step III: Prove that a Random Selection from $[0, a - 1]$ will be Coprime-to- a 50% of the Time	670
23.9.9	Step IV: Observe that a $y = cm$ Associated with c Coprime-to- a Will be Measured with Probability $1/2$	670
23.9.10	Step V: Observe that y_c Associated with c Coprime to a Will be Measured in Constant Time	672

23.9.11 Algorithm and Complexity Analysis (<i>Easy Case</i>)	673
23.10 Second Fork: General Case (We do not Assume $\mathbf{a} \mathbf{N}$)	675
23.10.1 Partitioning the Domain into Cosets	676
23.10.2 Rewriting the Output of the Oracle's output	678
23.10.3 Implication of a Hypothetical Measurement of the B register Output	679
23.10.4 Effect of a Final QFT on the A Register	681
23.10.5 Computation of Final Measurement Probabilities (General Case)	683
23.10.6 STEP I: Identify (<i>Without Proof</i>) a Special Set of a Elements, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ of High Measurement Likelihood	684
23.10.7 STEP II: <i>Prove</i> that the Values in, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ Have High Measurement Likelihood	689
23.10.8 STEP III: Associate $\{y_c\}_{c=0}^{a-1}$ with $\{c/a\}_{c=0}^{a-1}$	700
23.10.9 STEP IV: Describe an $O(\log^3 N)$ Algorithm that Will Produce c/a from y_c	703
23.10.10 Step V: Measure y_c Associated with c Coprime to a in Constant Time	705
23.10.1 Algorithm and Complexity Analysis (General Case)	714
23.10.1 Epilogue on Shor's Period-Finding Algorithm	716
24 Euclidean Algorithm and Continued Fractions	717
24.1 Ancient Algorithms for Quantum Computing	717
24.2 The Euclidean Algorithm	717
24.2.1 Greatest Common Divisor	717
24.2.2 Long Division	718
24.2.3 The Euclidean Algorithm, $EA(P, Q)$	718
24.2.4 The Algorithm	719
24.2.5 Time Complexity of EA	720
24.3 Convergents and the Continued Fraction Alorithm	721
24.3.1 Continued Fractions	721
24.3.2 Computing the CFA a_k s Using the EA if x is Rational	722
24.3.3 A Computer Science Method for Computing the a_k s of Contin- ued Fractions	723
24.3.4 Convergents of a Continued Fraction	724
24.3.5 An Algorithm for Computing the Convergents $\{n_k/d_k\}$	728
24.3.6 Easy Properties of the Convergents	728

24.3.7 CFA: Our Special Brand of Continued Fraction Algorithm . . .	729
25 From Period-Finding to Factoring	731
25.1 Period Finding to Factoring to RSA Encryption	731
25.2 The Problem and Two <i>Classically Easy</i> Cases	731
25.3 A Sufficient Condition	734
25.4 A Third Easy Case and Order-Finding in \mathbb{Z}_M	735
25.5 Using the Order of y to Find the x of our “Sufficient Condition” . . .	737
25.6 The Time Complexity of $f(x) = y^x \pmod{M}$	739
25.7 The Complexity Analysis	739
25.7.1 Complexity of Step 1	740
25.7.2 Complexity of Step 2	741
25.7.3 Absolute Complexity of Shor’s Factoring	741
 List of Figures	 742
 List of Tables	 747

Chapter 0

Introduction

0.1 Welcome to Volume One

0.1.1 About this Volume

This book accompanies **CS 83A**, the first of a three quarter quantum computing sequence offered to sophomores during their second or third years at *Foothill Community College* in Los Altos Hills, California. This first course focuses on quantum computing basics under the assumption that we have noise-free quantum components with which to build our circuits. The subsequent courses deal with advanced algorithms and quantum computing in the presence of noise, specifically, error correction and quantum encryption.

This is not a survey course; it skips many interesting aspects of quantum computing. On the other hand, it is in-depth. The focus is on *doing*. My hope is that some of you will apply the “hard” skills you learn here to discover quantum algorithms of your own. However, even if this is the only course you take on the subject, the computational tools you learn will be applicable far beyond the confines of quantum information theory.

0.1.2 About this Introduction

This short introduction contains samples of the math symbolism we will learn later in the course. These equations are intended only to serve as a taste of what’s ahead. You are not expected to know what the expressions and symbols mean yet, so don’t panic. All will be revealed in the next few weeks. Consider this introduction to be a no-obligation sneak preview of things to come.

0.2 Bits and Qubits

0.2.1 More Information than Zero and One

Classical computing is done with *classical bits*, each of which can take on a value of 0 or 1. Period.

Quantum computing happens in the world of quantum bits, a.k.a. *qubits*. Until a qubit, call it $|\psi\rangle$, is directly or indirectly measured, it is in a state that is neither 0 nor 1, but rather a *superposition* of the two, expressed formally as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle.$$

The symbol $|0\rangle$ corresponds to classical “0” value, while $|1\rangle$ is associated with a classical value of “1.” Meanwhile, the symbols α and β stand for numbers that express *how much* “0” and *how much* “1” are present in the qubit.

We’ll make this precise shortly, but the idea that you can have a teaspoon of “0” and a tablespoon of “1” contained in a single qubit immediately puts us on alert that we are no longer in the world of classical computing. This eerie concept becomes all the more magical when you consider that a qubit exists on a sub-atomic level (as photon or the spin state of an electron, for example), orders of magnitude smaller than the physical embodiment of a single classical bit which requires about a million atoms (or in research labs as few as 12).

That an infinitely small entity such as a qubit can store so much more information than a bulky classical bit comes at a price, however.

0.2.2 The Probabilistic Nature of Qubits

A Classical Experiment

If 100 classical *one-bit* memory locations are known to all hold the *same* value – call it x until we know what that value is – then they all hold $x = 0$ or all hold $x = 1$. If we measure the first location and find it to be 1, then we will have determined that *all 100* must hold a 1 (because of the assumption that all 100 locations are storing the exact same value). Likewise, if we measure a 0, we’d know that all 100 locations contain the value 0. Measuring the other 99 locations would confirm our conclusion. Everything is logical.

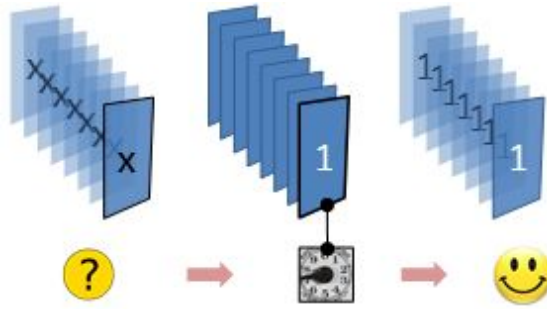


Figure 1: After measuring one location, we know them all

A Quantum Experiment

Qubits are a lot more slippery. Imagine a quantum computer capable of storing *qubits*. In this hypothetical we can inspect the contents of any memory location in our computer by attaching an *output meter* to that location and read a result off the meter.

Let's try that last experiment in our new quantum computer. We load 100 *qubit* memory locations with 100 *identically prepared qubits*. "Identically prepared" means that each qubit has the exact same value, call it $|\psi\rangle$. (Never mind that I haven't explained what the *value of a qubit* means; it has *some* meaning, and I'm asking you imagine that all 100 have the *same* value.)

Next, we use our meter to measure the first location. As in the classical case we discover that the output meter registers either a "0" or a "1." That's already a disappointment. We were hoping to get some science-fictiony-looking measurement from a *qubit*, especially one with a name like $|\psi\rangle$. Never mind; we carry on. Say the location gives us a measurement of "1."

Summary to this point. We loaded up all 100 locations with the same qubit, peered into the first location, and saw that it contained an ordinary "1."

What should we expect if we measure the other 99 locations? **Answer:** *We have no idea what to expect.*

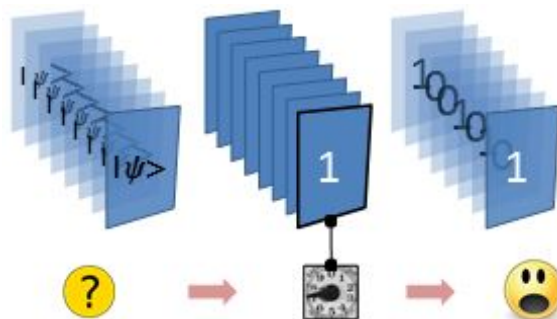


Figure 2: After measuring one location, we don't know much

Some of the remaining qubits may show us a "1," others a "0," all despite the *fact*

that the 100 locations initially stored identical qubit values.

This is disquieting. To our further annoyance,

- the measurement of each qubit always produces either a “0” or a “1” on our meter, despite our having prepared a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, falsely claiming to be some exciting combination of the two classical values,
- the measurement will have permanently destroyed the original state we prepared, leaving it in a classical condition of either “0” or “1,” no more magical superposition left in there,
- as already stated we know nothing (well, *almost* nothing, but that’s for another day) about the measurement outcomes of the other 99 supposedly identically prepared locations, and
- most bizarre of all, in certain situations, measuring the state of any one of these qubits will cause another qubit in a different computer, room, planet or galaxy to be modified without the benefit of wires, radio waves or time.

0.2.3 Quantum Mechanics – The Tool that Tames the Beast

Such wild behavior is actually well managed using *quantum mechanics*, the mathematical symbol-manipulation game that was invented in the early 20th century to help explain and predict the behavior of very, very small things. I cited the truculent nature of qubits in this introduction as a bit of a sensationalism to both scare and stimulate you. We can work with these things very easily despite their unusual nature.

The challenge for us is that quantum mechanics – and its application to information and algorithms – is not something one can learn in a week or two. But one *can* learn it in a few months, and that’s what we’re going to do in this course. I’ve prepared a sequence of lessons which will walk you through the fascinating mathematics and quantum mechanics needed to understand the new algorithms. Because it takes hard, analytical work, quantum computing isn’t for everyone. But my hope is that some among you will find this volume an accessible first step from which you can go on to further study and eventually invent quantum algorithms of your own.

0.2.4 Sneak Peek at the Coefficients α and β

So as not to appear too secretive, I’ll give you taste of what α and β *roughly* mean for the state $|\psi\rangle$. They tell us the respective *probabilities* that we would obtain a reading of either a “0” or a “1” were we to look into the memory location where $|\psi\rangle$ is stored. (In our quantum jargon, this is called *measuring the state $|\psi\rangle$* .) If, for

example, the values happened to be

$$\alpha = \frac{1}{2}, \quad \text{and} \\ \beta = \frac{\sqrt{3}}{2},$$

then the

$$\text{probability of measuring} \begin{cases} 0 & \text{would be} & (1/2)^2 = \frac{1}{4} = 25\% \\ 1 & \text{would be} & (\sqrt{3}/2)^2 = \frac{3}{4} = 75\% \end{cases}.$$

In other words, if a qubit with the precise and definitive value

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

is sitting in a one-(*qu*)bit memory location, and we look at that location, we will actually “see” a

$$\begin{aligned} 0 & \text{ with probability } |\alpha|^2 \text{ and} \\ 1 & \text{ with probability } |\beta|^2. \end{aligned}$$

This is far from the whole story as we’ll learn in our very first lecture, but it gives you a feel for how the probabilistic nature of the quantum world can be both slippery and quantitative at the same time. We don’t know what we’ll get when we query a quantum memory register, but we do know what the probabilities will be.

0.3 The Promise of Quantum Computing

0.3.1 Early Results

If we’re going to expend time studying math and quantum mechanics, we should expect something in return. The field is evolving rapidly, with new successes and failures being reported weekly. However, there are a few established results which are incontrovertible, and they are the reason so much effort is being brought to bear on quantum computer design.

Of the early results, perhaps the most dramatic is Shor’s period-finding algorithm, and it is this that I have selected as the endpoint of our first volume. It provides a basis for factoring extremely large numbers in a reasonable time when such feats would take classical computers billions of years. The applications, once implemented are profound. However the consequences may give one pause; network security as we know it would become obsolete.

Fortunately, or so we believe, there are different quantum techniques that offer alternatives to current network security and which could render it far more secure

than it is today. (These require additional theory, beyond the basics that we learn in volume 1 and will be covered in the sequel.)

There are also less sensational, but nevertheless real, improvements that have been discovered. Grover’s search algorithm for unsorted linear lists, while offering a modest speed-up over classical searches, is attractive merely by the ubiquity of search in computing. Related search techniques that look for items over a network are being discovered now and promise to replicate such results for graphs rather than list structures.

Quantum teleportation and super dense coding are among the simplest applications of quantum computing, and they provide a glimpse into possible new approaches to more efficient communication. We’ll get to these in this volume.

0.3.2 The Role of Computer Scientists

Quantum computers don’t exist yet. There is production grade hardware that appears to leverage quantum behavior but does not exhibit the simple qubit processing needs of the early – or indeed most of the current – quantum algorithms in computer science. On the other hand, many university rigs possess the “right stuff” for quantum algorithms, but they are years away from having the stability and/or size to appear in manufactured form.

The engineers and physicists are doing their part.

The wonderful news for us computer scientists is that we don’t have to wait. Regardless of what the hardware ultimately looks like, we already know what it will do. That’s because it is based on the most fundamental, firmly established and – despite my scary sounding lead-in – surprisingly simple quantum mechanics. We know what a qubit is, how a quantum logic gate will affect it, and what the consequences of reading qubit registers are. There is nothing preventing us from designing algorithms right now.

0.4 The Two Sides of Quantum Computer Science

Given that we can strap ourselves in and start work immediately, we should be clear on the tasks at hand. There are two.

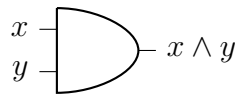
- **Circuit Design.** We know what the individual components will be, even if they don’t exist yet. So we must gain some understanding and proficiency in the assembly of these parts to produce full circuits.
- **Algorithm Design.** Because quantum mechanics is probabilistic by nature, we’ll have to get used to the idea that the circuits don’t always give us the answer right away. In some algorithms they do, but in others, we have to send the same inputs into the same circuits many times and let the laws of probability

play out. This requires us to analyze the math so we can know whether we have a fighting chance of our algorithm converging to an answer with adequate error tolerance.

0.4.1 Circuit Design

Classical

Classical logic gates are relatively easy to understand. An AND gate, for example, has a common symbol and straightforward truth table that defines it:



x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

You were introduced to logic like this in your first computer science class. After about 20 minutes of practice with various input combinations, you likely absorbed the full meaning of the AND gate without serious incident.

Quantum

A quantum logic gate requires significant vocabulary and symbolism to even define, never mind *apply*. If you promise not to panic, I'll give you a peek. Of course, you'll be trained in all the math and quantum mechanics in this course before we define such a circuit officially. By then, you'll be eating quantum logic for breakfast.

We'll take the example of something called a *second order Hadamard gate*. We would start by first considering the *second order qubit* on which the gate operates. Such a thing is symbolized using the mysterious notation and a column of numbers,

$$|\psi\rangle^2 = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}.$$

Next, we would send this qubit through the Hadamard gate using the symbolism

$$|\psi\rangle^2 \rightarrow \left\{ \begin{array}{c} \text{---} \text{---} \\ \text{---} \text{---} \end{array} \begin{array}{|c|} \hline H^{\otimes 2} \\ \hline \end{array} \begin{array}{c} \text{---} \text{---} \\ \text{---} \text{---} \end{array} \right\} H^{\otimes 2} |\psi\rangle^2$$

Although it means little to us at this stage, the diagram shows the qubit $|\psi\rangle^2$ entering the Hadamard gate, and another, $H^{\otimes 2}|\psi\rangle^2$, coming out.

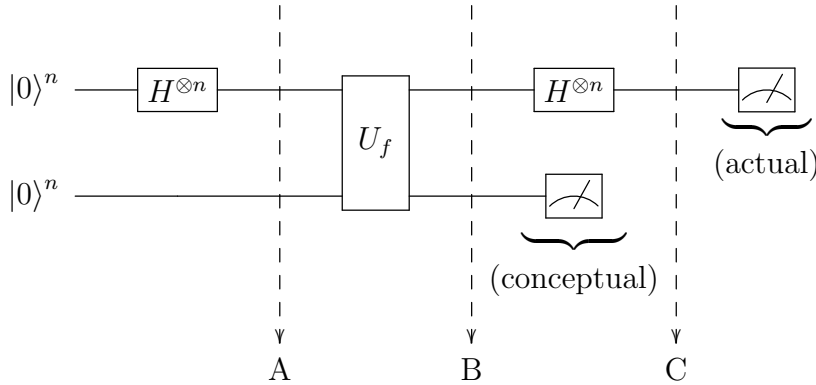
Finally, rather than a *truth table*, we will need a *matrix* to describe the behavior of the gate. Its action on our qubit would be the result of *matrix multiplication* (another topic we'll cover if you haven't had it),

$$H^{\otimes 2}|\psi\rangle^2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \alpha + \beta + \gamma + \delta \\ \alpha - \beta + \gamma - \delta \\ \alpha + \beta - \gamma - \delta \\ \alpha - \beta - \gamma + \delta \end{pmatrix}.$$

Again, we see that there is a lot of unlearned symbolism, and not the kind that can be explained in a few minutes or hours. We'll need weeks. But the weeks will be packed with exciting and useful information that you can apply to all areas of engineering and science, not just quantum computing.

0.4.2 Algorithms

In quantum computing, we first design a small circuit using the components that are (or will one day become) available to us. An example of such a circuit in diagram form (with no explanation offered today) is



There are access points, A, B and C, to assist with the analysis of the circuit. When we study these circuits in a few weeks, we'll be following the state of a qubit as it makes its way through each access point.

Deterministic

The algorithm may be *deterministic*, in which case we get an answer immediately. The final steps in the algorithm might read:

We run the circuit one time only and measure the output.

- *If we read a zero the function is constant.*
- *If we read any non-zero value, the function is balanced.*

This will differ from a corresponding classical algorithm that requires, typically, *many* evaluations of the circuit (or in computer language, many “loop passes”).

Probabilistic

Or perhaps our algorithm will be *probabilistic*, which means that once in a blue moon it will yield an incorrect answer. The final steps, then, might be:

- *If the above loop ended after $n + T$ full passes, we failed.*
- *Otherwise, we succeeded and have solved the problem with error probability $< 1/2^T$ and with a big- O time complexity of $O(n^3)$, i.e., in polynomial time.*

Once Again: I don’t expect you to know about *time complexity* or *probability* yet. You’ll learn it all here.

Whether deterministic or probabilistic, we will be designing circuits and their algorithms that can do things faster than their classical cousins.

0.5 Perspective

Quantum computing does not promise to do everything better than classical computing. In fact, the majority of our processing needs will almost certainly continue to be met more efficiently with today’s bit-based logic. We are designing new tools for currently unsolvable problems, not to fix things that are currently unbroken.

0.6 Navigating the Topics

Most students will find that a cover-to-cover reading of this book does not match their individual preparation or goals. One person may skip the chapters on *complex arithmetic* and *linear algebra*, while another may devote considerable – and I hope *pleasurable* – time luxuriating in those subjects. You will find your path in one of three ways:

1. **Self-Selection.** The titles of chapters and sections are visible in the click-able table of contents. You can use them to evaluate whether a set of topics is likely to be worth your time.

2. **Chapter Introductions.** The first sentence of some chapters or sections may qualify them as *optional*, intended for those who want more coverage of a specific topic. If any such optional component is needed in the later volumes accompanying **CS 83B** or **CS 83C**, the student will be referred back to it.
3. **Tips Found at the Course Site.** Students enrolled in **CS 83A** at Foothill College will have access to the course web site where weekly modules, discussion forums and private messages will contain individualized navigation advice.

Let's begin by learning some math.

Chapter 1

Complex Arithmetic

1.1 Complex Numbers for Quantum Computing

I presented the briefest of introductions to the quantum bit, or *qubit*, defining it as an expression that combines the usual binary values “0” and “1” (in the guise of the symbols $|0\rangle$ and $|1\rangle$). The qubit was actually a combination, or *superposition*, of those two binary values, each “weighted” by numbers α and β , respectively,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle .$$

But what are these numbers α and β ? If you are careful not to take it too seriously, you can imagine them to be numbers between 0 and 1, where small values mean less probable – or a small dose – and larger values mean more probable – or a high dose. So a particular qubit value, call it $|\psi_0\rangle$, defined to be

$$|\psi_0\rangle = (.9798) |0\rangle + (.200) |1\rangle ,$$

would mean a large “amount” of the classical bit 0 and a small “amount” of the classical bit 1. I’m intentionally using pedestrian terminology because it’s going to take us a few weeks to rigorously define all this. However, I can reveal something immediately: real numbers will not work for α or β . The vagaries of quantum mechanics require that these numbers be taken from the richer pool of *complex numbers*.

Our quest to learn quantum computing takes us through the field of quantum mechanics, and the first step in *that* effort must always be a mastery of complex arithmetic. Today we check off that box. And even if you’ve studied it in the past, our treatment today might include a few surprises – results we’ll be using repeatedly like Euler’s formula, how to sum complex roots-of-unity, the complex exponential function and polar forms. So without further ado, let’s get started.

1.2 The Field of Complex Numbers

1.2.1 The Real Numbers Just Don't Cut It

We can only go so far in studying quantum information theory if we bind ourselves to using only the real numbers, \mathbb{R} . The hint that \mathbb{R} is not adequate comes from simple algebra. We know the solution of

$$x^2 - 1 = 0$$

is $x = \pm 1$. But make the slightest revision,

$$x^2 + 1 = 0,$$

and we no longer have any solutions in the real numbers. Yet we need such solutions in physics, engineering and indeed, in every quantitative field from economics to neurobiology.

The problem is that the real numbers do not constitute a *complete field*, a term that expresses the fact that there are equations that have no solutions in that number system. We can force the last equation to have a solution by royal decree: we declare the number

$$i \equiv \sqrt{-1}$$

to be added to \mathbb{R} . It is called an *imaginary number*. Make sure you understand the meaning here. We are not computing a square root. We are defining a new number whose name is i and proclaiming that it have the property that

$$i^2 = -1.$$

i is merely shorthand for the black-and-white pattern of scribbles, " $\sqrt{-1}$." But those scribbles tells us the essential property of this new number, i .

This gives us a solution to the equation $x^2 + 1 = 0$, but there are infinitely many *other* equations that still don't have solutions. It seems like we would have to manufacture a new number for every equation that lacks a real solution (a.k.a. a *real zero* in math jargon).

1.2.2 The Definition of \mathbb{C}

However, we get lucky. The number i can't just be thrown in without also specifying how we will respond when someone wants to add or multiply it by a number like 3 or -71.6. And once we do that, we start proliferating new combinations – called *complex numbers*. Each such non-trivial combination (i.e., one with an i in it) will be a solution to *some* equation that doesn't have a real zero. Here are a few examples of the kinds of new numbers we will get:

$$2 + 3i, \quad 1 - i, \quad .05 + .002i, \quad \pi + \sqrt{2}i, \quad 52, \quad -100i$$

All of these numbers are of the form

$$\text{some real number} + \text{some real multiple of } i.$$

Terminology

When there is no *real* part (the first term in the above sum) to the complex number, it is called *purely imaginary*, or just *imaginary*. The last example in the list above is *purely imaginary*.

If we take all these combinations, we have a new supersized number system, the *complex numbers*, defined by

$$\mathbb{C} \equiv \{ a + bi \mid a \in \mathbb{R}, b \in \mathbb{R} \text{ and } i = \sqrt{-1} \}.$$

1.2.3 The Complex Plane

Since every complex number is defined by an ordered pair of real numbers, (a, b) , where it is understood that

$$(a, b) \leftrightarrow a + bi,$$

we have a natural way to represent each such number on a plane, whose x -axis is the real axis (which expresses the value a), and y -axis is the imaginary axis (which expresses the value b) (figure 1.1). This looks a lot like the real Cartesian plane, \mathbb{R}^2 ,

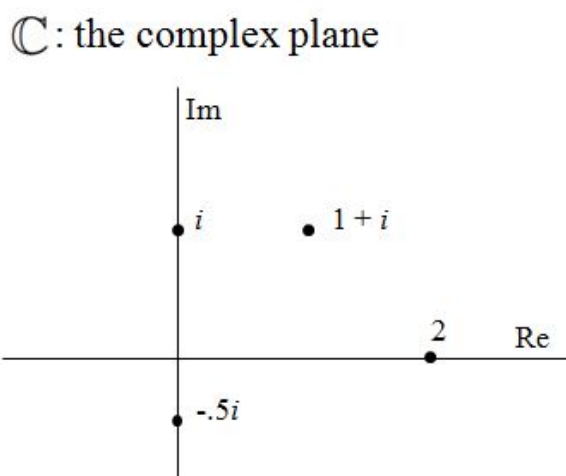


Figure 1.1: a few numbers plotted in the complex plane

but it isn't. Both pictures are a collection of ordered pairs of real numbers, but \mathbb{C} is richer than \mathbb{R}^2 when we dig deeper: it has *product* and *quotient* operations, both missing in \mathbb{R}^2 . (You'll see.) For now, just be careful to not confuse them.

One usually uses z , c or w to denote complex numbers. Often we write the i before the real coefficient:

$$z = x + iy, \quad c = a + ib, \quad w = u + iv$$

In quantum computing, our complex numbers are usually coefficients of the *computational basis states* – a new term for those special symbols $|0\rangle$ and $|1\rangle$ we have been toying with – in which case we may use Greek letters α or β , for the complex numbers,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle.$$

This notation emphasizes the fact that the complex numbers are *scalars* of the complex vector space under consideration.

[**Note.** If terms like *vector space* or *scalar* are new to you, fear not. I'm not officially defining them yet, and we'll have a full lecture on them. I just want to start exposing you to some vocabulary early.]

Equality of Two Complex Numbers

The criteria for two complex numbers to be equal follows the template set by two points in \mathbb{R}^2 being equal: both coordinates must be equal. If

$$\begin{aligned} z = x + iy \quad \text{and} \quad w = u + iv, \\ \text{then} \\ z = w \iff x = u \quad \text{and} \quad y = v. \end{aligned}$$

1.2.4 Operations on Complex Numbers

We define what we mean by addition, multiplication, etc. next. We already know from the definition of i , that

$$i^2 = -1,$$

which tells us the answer to $i \cdot i$. From there, we build-up by assuming that the operations \times and $+$ obey the same kinds of laws (commutative, associative, distributive) as they do for the reals. With these ground rules, we can quickly demonstrate that the only way to define addition (subtraction) and multiplication is

$$\begin{aligned} (a + ib) \pm (c + id) &\equiv (a \pm c) + i(b \pm d) \\ \text{and} \\ (a + ib)(c + id) &\equiv (ac - bd) + i(ad + bc). \end{aligned}$$

The rule for division can actually be derived from these,

$$\begin{aligned}\frac{a+ib}{c+id} &= \left(\frac{a+ib}{c+id}\right) \cdot \left(\frac{c-id}{c-id}\right) \\ &= \frac{(ac+bd) + i(bc-ad)}{c^2+d^2} \\ &= \frac{ac+bd}{c^2+d^2} + i \frac{bc-ad}{c^2+d^2}, \quad \text{where } c^2+d^2 \neq 0.\end{aligned}$$

A special consequence of this is the oft cited identity

$$\frac{1}{i} = -i.$$

[**Exercise.** Prove it.]

Addition (or subtraction) can be pictured as the vectorial sum of the two complex numbers (see figure 1.2). However, multiplication is more easily visualized when we

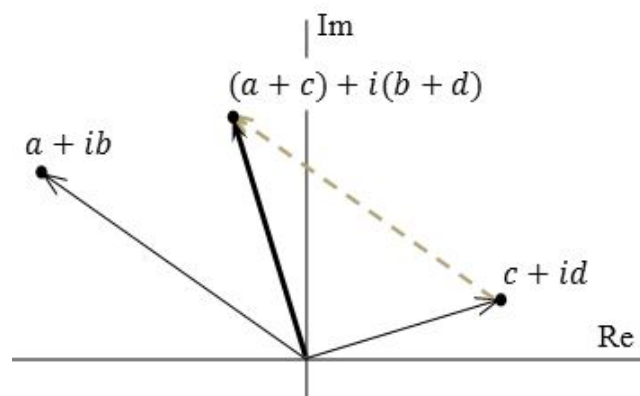


Figure 1.2: visualization of complex addition

get to polar coordinates, so we'll wait a few minutes before showing that picture.

[**Exercise.** For the following complex numbers,

$$2 + 3i, \quad 1 - i, \quad .05 + .002i, \quad \pi + \sqrt{2}i, \quad 52, \quad -100i,$$

- (a) square them,
- (b) subtract $1 - i$ from each of them,
- (c) multiply each by $1 - i$,
- (d) divide each by 5,
- (e) divide each by i , and
- (f) divide any three of them by $1 - i$.]

[**Exercise.** Explain why it is not necessarily true that the square of a complex number, z^2 , be positive or zero – or even real, for that matter. If z^2 is real, does that mean it must be non-negative?]

1.2.5 \mathbb{C} is a Field

A number system that has *addition* and *multiplication* replete with the usual properties is called a *field*. What we have outlined above is the fact that \mathbb{C} is, like \mathbb{R} , a *field*. When you have a field, you can then create a *vector space* over that field by taking n -tuples of numbers from that field. Just as we have real n -dimensional vector spaces, \mathbb{R}^n , we can as easily create n -dimensional vector spaces over \mathbb{C} which we call \mathbb{C}^n . (We have a whole lesson devoted to defining real and complex vector spaces.)

1.3 Exploring the Complex Plane

1.3.1 Complex Numbers as Ordered Pairs of Real Numbers

We already saw that each complex number has two aspects to it: the *real term* and the *term that has the i in it*. This creates a natural correspondence between \mathbb{C} and \mathbb{R}^2 ,

$$x + iy \longleftrightarrow (x, y).$$

As a consequence, a special name is given to Cartesian coordinates when applied to complex numbers: *the complex plane*.

[**Advanced Readers.** For those of you who already know about vector spaces, real and complex, I'll add a word of caution. This is not the same as a complex vector space consisting of ordered pairs of complex numbers (z, w) . The complex plane consists of one point for every complex number, not a point for every ordered pair of complex numbers.]

1.3.2 Real and Imaginary Axes and Polar Representation

The axes can still be referred to as the x -axis and y -axis, but they are more commonly called the *real*-axis and *imaginary* axis. The number i sits on the imaginary axis, one unit above the real axis. The number -3 is three units to the left of the imaginary axis on the real axis. The number $1 + i$ is in the “first quadrant.” [**Exercise.** Look up that term and describe what the *second*, *third* and *fourth quadrants* are.]

Besides using (x, y) to describe z , we can use the polar representation suggested by polar coordinates (r, θ) of the complex plane (figure 1.3).

$$x + iy \longleftrightarrow r(\cos \theta + i \sin \theta)$$

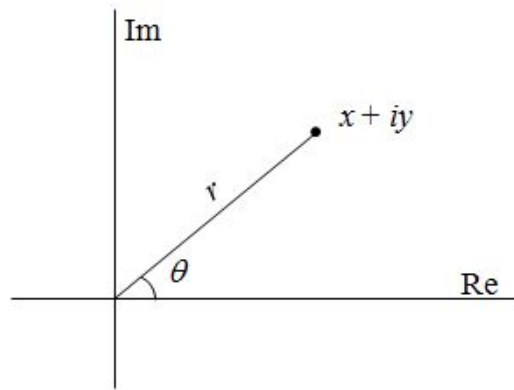


Figure 1.3: the connection between cartesian and polar coordinates of a complex number

Terminology

$x = \operatorname{Re}(z)$, the “real part” of z

$y = \operatorname{Im}(z)$, the “imaginary part” of z

$r = |z|$, the “modulus” (“magnitude”) of z

$\theta = \arg z$, the “argument” (“polar angle”) of z

$3i$, πi , $900i$, etc. “purely imaginary” numbers

Note. *Modulus* will be discussed more fully in a moment. For now, $|z| = r$ can be taken as a definition or just terminology with the definition to follow.

1.3.3 Complex Conjugate and Modulus

The Conjugate of a Complex Number

We can apply an operation, *complex conjugation*, to any complex number to produce another complex number by negating the imaginary part. If

$$z = x + iy$$

then its *complex conjugate* (or just *conjugate*) is designated and defined as

$$z^* \equiv x - iy, \quad \text{the “complex conjugate” of } z.$$

A common alternate notation places a horizontal bar above z (instead of an asterisk to the upper right),

$$\bar{z} \equiv z^*.$$

Geometrically, this is like reflecting z across the x (real) axis (figure 1.4).

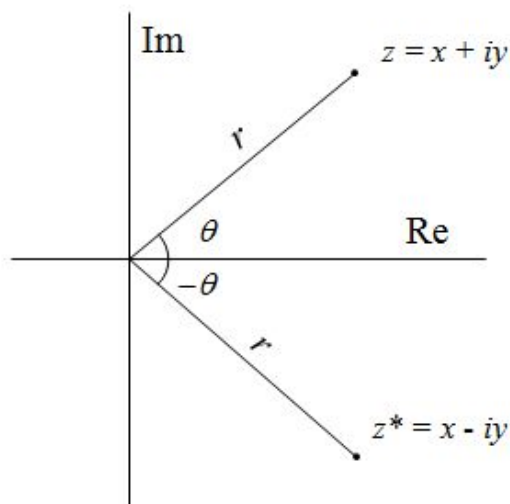


Figure 1.4: conjugation as reflection

Examples

$$\begin{aligned}
 (35 + 8i)^* &= 35 - 8i \\
 (2 - \sqrt{2}i)^* &= 2 + \sqrt{2}i \\
 (-3\sqrt{2}i)^* &= (3\sqrt{2}i) \\
 (1.5\pi)^* &= 1.5\pi
 \end{aligned}$$

It is easy to show that conjugation distributes across sums and products, i.e.,

$$\begin{aligned}
 (wz)^* &= w^* z^* \quad \text{and} \\
 (w + z)^* &= w^* + z^*.
 \end{aligned}$$

These little factoids will come in handy when we study *kets*, *bras* and *Hermitian conjugates* in a couple weeks.

[**Exercise.** Prove both assertions. What about quotients?]

The Modulus of a Complex Number

Just as in the case of \mathbb{R}^2 , the modulus of the complex z is the distance of the line segment (in the complex plane) $\overline{0z}$, that is,

$$\begin{aligned}
 |z| &\equiv \sqrt{\text{Re}(z)^2 + \text{Im}(z)^2} \\
 &= \sqrt{x^2 + y^2}.
 \end{aligned}$$

A short computation shows that multiplying z by its conjugate, z^* , results in a

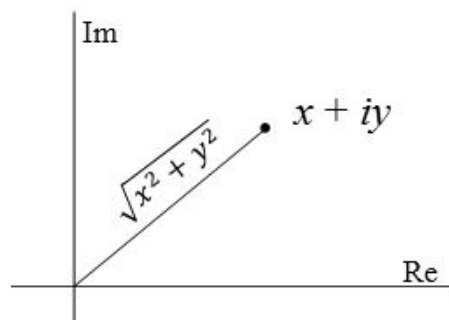


Figure 1.5: modulus of a complex number

non-negative real number which is the square of the modulus of z .

$$\begin{aligned} |z|^2 &= zz^* = z^*z = x^2 + y^2 \\ |z| &= \sqrt{z^*z} \end{aligned}$$

[**Exercise.** Fill in any details that are not immediately apparent.]

In these lectures you may have seen (and will continue to see) me square the *absolute value* of a number or function. You might have wondered why I bothered to use absolute value signs around something I was about to square. Now you know: the value of the number might be complex and simply squaring it would not necessarily result in a real – much less positive – number. The square of complex numbers are not normally real. (If you didn't do the complex arithmetic exercise earlier, do it now to get a few concrete examples of this phenomenon.)

1.4 Transcendental Functions and Their Identities



The term *transcendental function* has a formal definition, but for our purposes it means functions like $\sin x$, $\cos x$, e^x , $\sinh x$, etc. It's time to talk about how they are defined and relate to complex arithmetic.

1.4.1 The Complex Exponential Function Part 1: Pure Imaginary Case

From calculus, you may have learned that the real exponential function, $\exp(x) = e^x$, can be expressed – by some authors, *defined* – in terms of an infinite sum, the *Taylor*

expansion or Taylor series,

$$\exp(x) = e^x \equiv \sum_{n=0}^{\infty} \frac{x^n}{n!}.$$

This suggests that we can define a complex exponential function that has a similar expansion, only for a complex z rather than a real x .

Complex Exponential of a Pure Imaginary Number and Euler's Formula

We start by defining a new function of a purely imaginary number, $i\theta$, where θ is real,

$$\exp(i\theta) = e^{i\theta} \equiv \sum_{n=0}^{\infty} \frac{(i\theta)^n}{n!}.$$

(A detail that I am skipping is the proof that this series converges to a complex number for all real θ . But believe me, it does.) We've already defined the *arg* of a complex number z to be the angle, θ , z "makes" with the real axis in the complex plane. We'll see in a moment that if we take z to be the functional value we just defined in terms of θ , that is, $z = \exp(i\theta)$, then θ will turn out to be $\arg(z)$.

Let's expand the sum, but first, an observation about increasing powers of i .

$$\begin{aligned} i^0 &= 1 \\ i^1 &= i \\ i^2 &= -1 \\ i^3 &= -i \\ i^4 &= 1 \\ i^5 &= i \\ &\dots \\ i^{n+4} &= i^n \end{aligned}$$

Apply these powers of i to the infinite sum.

$$\begin{aligned} e^{i\theta} &= 1 + \frac{i\theta}{1!} - \frac{\theta^2}{2!} - \frac{i\theta^3}{3!} + \frac{\theta^4}{4!} \\ &\quad + \frac{i\theta^5}{5!} - \frac{\theta^6}{6!} - \frac{i\theta^7}{7!} + \frac{i\theta^8}{8!} \\ &\quad + \dots \end{aligned}$$

Rearrange the terms so that all the real terms are together and all the imaginary terms are together,

$$\begin{aligned} e^{i\theta} &= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \frac{\theta^8}{8!} + \dots \right) \\ &\quad + i \left(\frac{\theta}{1!} - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \dots \right). \end{aligned}$$

You may recognize the two parenthetical expressions as the Taylor series for $\cos \theta$ and $\sin \theta$, and we pause to summarize this result of profound and universal importance.

Euler's Formula

$$e^{i\theta} = \cos \theta + i \sin \theta.$$

Notice the implication,

$$|e^{i\theta}|^2 = \cos^2 \theta + \sin^2 \theta = 1,$$

which leads to one of the most necessary and widely used facts in all of physics and engineering,

$$|e^{i\theta}| = 1, \quad \text{for real } \theta.$$

[**Exercise.** Prove this last equality without recourse to Euler's formula, using exponential identities alone.]

[**Exercise.** How does Euler's formula justify my earlier claim that $\theta = \arg(e^{i\theta})$?]

Euler's formula tells us how to visualize the exponential of a pure imaginary. If we think of θ as *time*, then $e^{i\theta}$ is a “spec” (if you graph it in the complex plane) traveling around the unit-circle counter-clockwise at 1 radian-per-second (see Figure 1.6). I

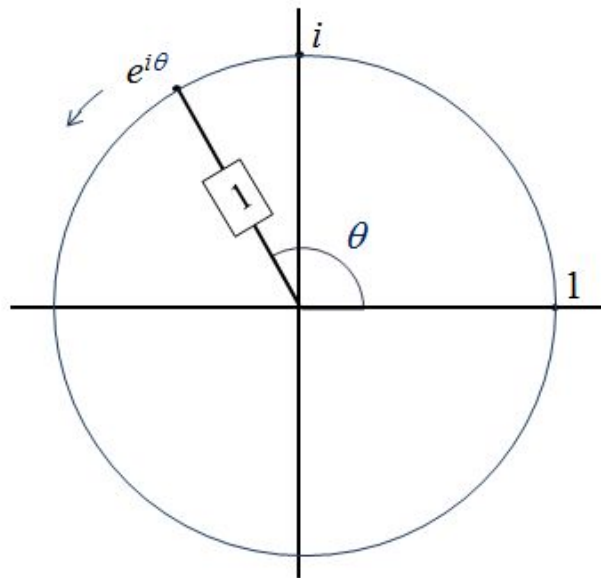


Figure 1.6: $e^{i\theta}$ after θ seconds, @ 1 rad/sec

cannot overstate the importance of Euler's formula and the accompanying picture. You'll need it in this course and almost in any other engineering study.

1.4.2 Real $\sin()$ and $\cos()$ in Terms of the Complex Exponential

Now try this: Plug $-\theta$ (minus theta) into Euler's formula, then add (or subtract) the resulting equation to the original formula. Because of the trigonometric identities

$$\begin{aligned}\sin(-\theta) &= -\sin(\theta), & \text{and} \\ \cos(-\theta) &= \cos(\theta),\end{aligned}$$

the so-called oddness and evenness of \sin and \cos , respectively, you would quickly discover the first (or second) equality

$$\begin{aligned}\cos \theta &= \frac{e^{i\theta} + e^{-i\theta}}{2}, \\ \sin \theta &= \frac{e^{i\theta} - e^{-i\theta}}{2i}.\end{aligned}$$

These appear often in physics and engineering, and we'll be relying on them later in the course.

1.4.3 Complex Exponential Part 2: Any Complex Number

We have defined $\exp()$ for pure imaginaries as an infinite sum, and we already knew that $\exp()$ for reals was an infinite sum, so we combine the two to define $\exp()$ for any complex number. If $z = x + iy$,

$$\exp(z) = e^z = e^{x+iy} \equiv e^x e^{iy}.$$

We can do this because each factor on the far right is a complex number (the first, of course, happens to also be real), so we can take their product.

For completeness, we should note (this requires proof, not supplied) that everything we have done leads to the promised Taylor expansion for e^z as a function of a complex z , namely,

$$\exp(z) = e^z \equiv \sum_{n=0}^{\infty} \frac{z^n}{n!}.$$

This definition implies, among other things, the correct behavior of $\exp(z)$ with regard to addition of "exponents," that is

$$\exp(z+w) = \exp(z) \exp(w),$$

or, more familiarly,

$$e^{z+w} = e^z e^w.$$

Easy Proof of Addition Law of Sines and Cosines

A consequence of all this hard work is an easy way to remember the trigonometric addition laws. I hope you're sitting down. Take A, B two real numbers – they can be angles. From Euler's formula, using $\theta = A + B$, we get:

$$e^{i(A+B)} = \cos(A+B) + i \sin(A+B)$$

On the other hand, we can apply Euler's formula to $\theta = A$ and $\theta = B$, individually, in this product:

$$\begin{aligned} e^{iA} e^{iB} &= (\cos A + i \sin A) (\cos B + i \sin B) \\ &= (\cos A \cos B - \sin A \sin B) \\ &\quad + i (\sin A \cos B + \cos A \sin B) \end{aligned}$$

Because of the law of exponents we know that the LHS of the last two equations are equal, so their RHSs must also be equal. Finally, equate the real and imaginary parts:

$$\begin{aligned} \cos(A+B) &= \cos A \cos B - \sin A \sin B \\ \sin(A+B) &= \sin A \cos B + \cos A \sin B \end{aligned}$$

QED

1.4.4 The Complex Trigonometric Functions

We won't really need these, but let's record them for posterity. For a complex z , we have,

$$\cos z = 1 - \frac{z^2}{2!} + \frac{z^4}{4!} - \frac{z^6}{6!} + \frac{z^8}{8!} + \dots$$

and

$$\sin z = \frac{z}{1!} - \frac{z^3}{3!} + \frac{z^5}{5!} - \frac{z^7}{7!} + \dots$$

We also get an Euler-like formula for general complex z , namely,

$$e^{iz} = \cos z + i \sin z.$$

1.4.5 Polar Relations Expressed Using the Exponential

We are comfortable expressing a complex number as the sum of its real and imaginary parts,

$$z = x + iy.$$

But from the equivalence of the Cartesian and polar coordinates of a complex number seen in figure 1.3 and expressed by

$$x + iy \longleftrightarrow r(\cos \theta + i \sin \theta),$$

we can use the Euler formula on the RHS to obtain the very useful

$$z = re^{i\theta}.$$

The latter version gives us a variety of important identities. If z and w are two complex numbers expressed in polar form,

$$\begin{aligned} z &= re^{i\theta} \quad \text{and} \\ w &= se^{i\phi}, \end{aligned}$$

then we have

$$\begin{aligned} zw &= rs e^{i(\theta+\phi)} \quad \text{and} \\ z/w &= \frac{r}{s} e^{i(\theta-\phi)}. \end{aligned}$$

Notice how the moduli multiply or divide and the args add or subtract (figure 1.7).

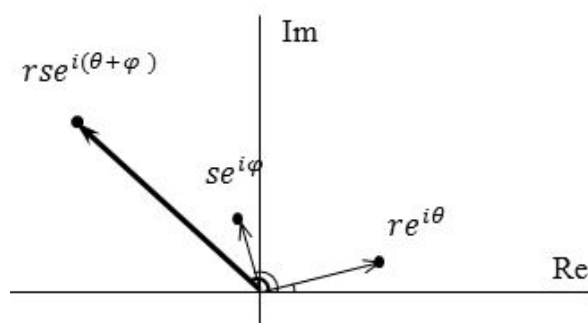


Figure 1.7: multiplication – moduli multiply and args add

Also, we can express reciprocals and conjugates nicely, using

$$\begin{aligned} 1/z &= \frac{1}{r} e^{-i\theta} \quad \text{and} \\ z^* &= re^{-i\theta}. \end{aligned}$$

In fact, that last equation is so useful, I'll restate it slightly. For any real number, θ ,

$$(re^{i\theta})^* = re^{-i\theta}.$$

[Exercise. Using polar notation, find a short proof that the conjugate of a product (quotient) is the product (quotient) of the conjugates. (This is an exercise you may have done above using more ink.)]

A common way to use these relationships is through equivalent identities that put the emphasis on the modulus and arg, separately,

$$\begin{aligned} |zw| &= |z| |w| , \\ \left| \frac{z}{w} \right| &= \frac{|z|}{|w|} , \\ |z^*| &= |z| , \\ \arg(zw) &= \arg z + \arg w , \\ \arg\left(\frac{z}{w}\right) &= \arg z - \arg w , \quad \text{and} \\ \arg(z^*) &= -\arg z . \end{aligned}$$

[**Exercise.** Verify all of the last dozen or so polar identities using the results of the earlier sections.]

1.5 Roots of Unity

An application of complex arithmetic that will be used extensively in Shor’s algorithm and the quantum Fourier transform involves the roots of unity.

1.5.1 N Distinct Solutions to $z^N = 1$

Here is something you don’t see much in the real numbers. The equation

$$z^N = 1 ,$$

with a positive integer, N , has either one or two real roots (“zeros”), depending on the “parity” of N . That is, if N is odd, the only zero is 1. If N is even, there are two zeros, -1 and 1.

In complex algebra, things are different. We’re going to see that there are N distinct solutions to this equation. But first, we take a step back.

Consider the complex number (in polar form),

$$\omega_N \equiv e^{i(2\pi/N)} .$$

ω_N is usually written

$$e^{2\pi i/N} ,$$

but I grouped the non- i factor in the exponent so you could clearly see that ω_N was of the form $e^{i\theta}$ that we just finished studying. Indeed, knowing that $\theta = 2\pi/N$ is a real number allows us to use Euler’s formula,

$$e^{i\theta} = \cos \theta + i \sin \theta ,$$

to immediately conclude the following:

- ω_N lies somewhere on the unit circle,
- $\omega_1 = e^{2\pi i} = 1$,
- $\omega_2 = e^{\pi i} = -1$,
- $\omega_3 = e^{2\pi i/3} = -\frac{1}{2} + \frac{\sqrt{3}}{2}i$,
- $\omega_4 = e^{\pi i/2} = i$, and
- for $N > 4$, as N increases (5, 6, 7, etc.), ω_N marches clockwise along the upper half of the unit circle approaching 1 (but never reaching it). For example ω_{1000} is almost indistinguishable from 1, lying just an “imperceptible fraction” above it.

For any $N > 0$, we can see that

$$\begin{aligned}(\omega_N)^N &= e^{iN(2\pi/N)} = e^{2\pi i} \\ &= \cos 2\pi + i \sin 2\pi = 1,\end{aligned}$$

earning it the name *Nth root of unity* (sometimes hyphenated *Nth root-of-unity*) as well as the symbolic formulation

$$\omega_N = \sqrt[N]{1}.$$

In fact, we should call this the *primitive* *Nth root-of-unity* to distinguish it from its siblings which we’ll meet in a moment. Finally, we can see that ω_N is a non-real (when $N > 2$) solution to the equation

$$z^N = 1.$$

Often, when we are using the same N and ω_N for many pages, we omit the subscript and use the simpler,

$$\omega \equiv e^{2\pi i/N},$$

with N understood by context, especially helpful when used in large formulas.

If we fix a *primitive* *Nth root-of-unity* for some N (think $N = 5$ or larger), and we take powers of ω_N , like ω_N , $(\omega_N)^2$, $(\omega_N)^3$, etc., each power is just the rotation on the unit circle of the previous power, counter-clockwise, by the angle $2\pi/N$. [**Exercise.** Explain why.] Eventually we’ll wrap around back to 1, producing N distinct complex numbers,

$$\sqrt[N]{1} = e^{i(2\pi/N)}, \quad e^{i2(2\pi/N)}, \quad e^{i3(2\pi/N)}, \quad \dots, \quad e^{i2\pi} = 1.$$

(See Figure 1.8.) These are also *Nth roots-of-unity*, generated by taking powers of the *primitive* *Nth root*.

[**Exercise.** Why are they called *Nth roots-of-unity*? **Hint:** Raise any one of them to the N th power.]

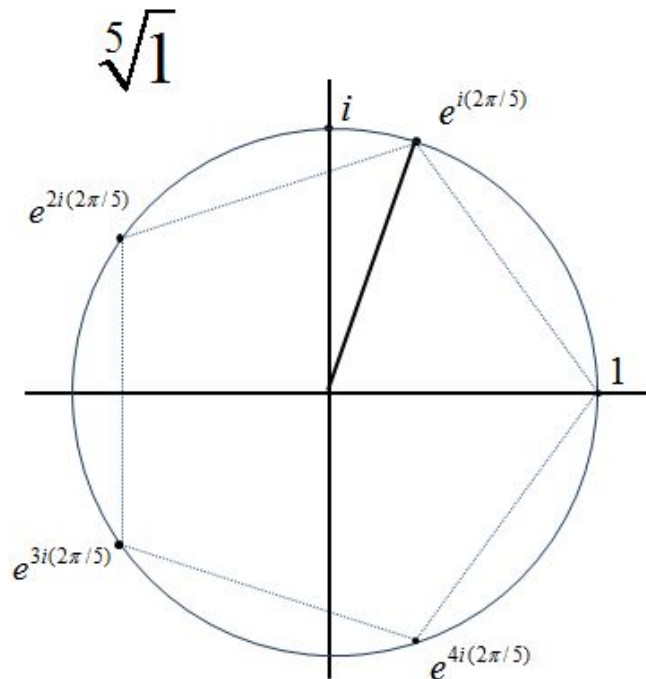


Figure 1.8: the fifth roots-of-unity

For our purposes, we'll consider the *primitive* N th root-of-unity to be the one that has the smallest arg (angle), namely, $e^{2\pi i/N}$, and we'll call all the other powers, ordinary, *non-primitive* roots. However, this is not how mathematicians would make the distinction, and *does not match the actual definition* which includes some of the other roots, a subtlety we can safely overlook.

As you can see, all N of the N th roots are the vertices of a regular polygon inscribed in the unit circle, with one vertex anchored at the real number 1. This is a fact about roots-of-unity that you'll want to remember. And now we have met all N of the distinct solutions to the equation $z^N = 1$.

1.5.2 Euler's Identity

By looking at the fourth root-of-unity, you can get some interesting relationships.

$$\begin{aligned} e^{i\pi/2} &= i, \\ e^{2\pi i} &= 1, \quad \text{and} \\ e^{\pi i} &= -1. \end{aligned}$$

(See Figure 1.9.) That last equation is also known as *Euler's identity* (distinct from Euler's formula). Sometimes it is written in the form

$$e^{\pi i} + 1 = 0.$$

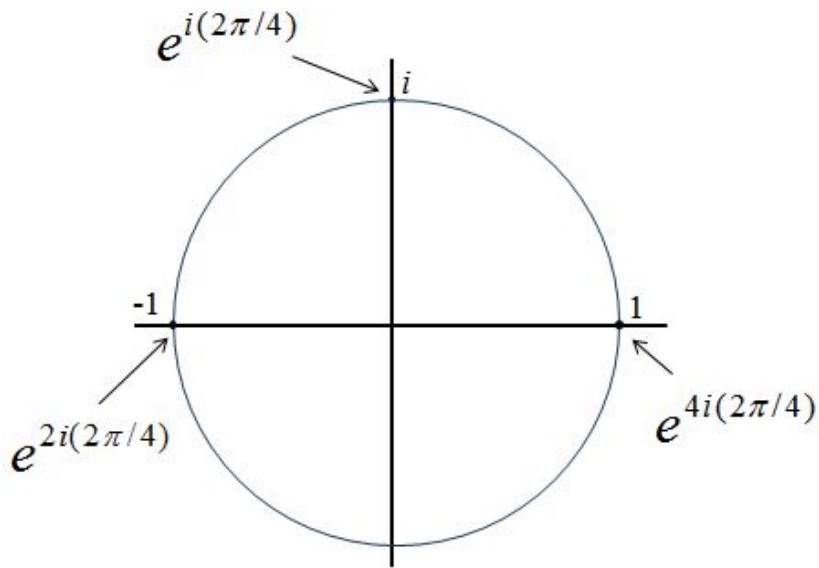


Figure 1.9: Three of the fourth roots-of-unity (find the fourth)

Take a moment to ponder this amazing relationship between the constants, i , e and π .

1.5.3 Summation Notation

There's a common notation used to write out long sums called the *summation notation*. We'll use it throughout the course, beginning with the next subsection, so let's formally introduce it here. Instead of using the ellipsis (...) to write a long sum, as in

$$a_1 + a_2 + a_3 + \dots + a_n ,$$

we symbolize like this

$$\sum_{k=1}^n a_k .$$

The index starting the sum is indicated below the large Greek Sigma, (Σ), and the final index in the sum is placed above it. For example, if we wanted to start the sum from a_0 and end at a_{n-1} , we would write

$$\sum_{k=0}^{n-1} a_k .$$

A common variation is the infinite sum, written as

$$\sum_{k=0}^{\infty} a_k ,$$

where the start of the sum can be anything we want it to be: 0, 1, -5 or even $-\infty$.

[**Exercise.** Write out the sums

$$1 + 2 + 3 + \dots + 1999,$$

and

$$0 + 2 + 4 + \dots + 2N,$$

using summation notation.]

1.5.4 Summing Roots-of-Unity and the Kronecker Delta

We finish this tutorial on complex arithmetic by presenting some facts about roots-of-unity that will come in handy in a few weeks. You can take some of these as exercises to confirm that you have mastered the ability to calculate with complex numbers.

For the remainder of this section, let's use the shorthand that I advertised earlier and call the primitive N th root-of-unity ω , omitting the subscript N , which will be implied.

We have seen that ω , as well as all of its integral powers,

$$\omega, \quad \omega^2, \quad \dots, \quad \omega^{N-1}, \quad \omega^N = \omega^0 = 1, \quad ,$$

are solutions of the N th degree polynomial equation

$$z^N - 1 = 0.$$

But these N complex numbers are also solutions of another polynomial

$$(z - 1)(z - \omega)(z - \omega^2) \cdots (z - \omega^{N-1}) = 0.$$

To see this, just plug any of the N th roots into this equation and see what happens. Any two polynomials that have exactly the same roots are equal, so

$$z^N - 1 = (z - 1)(z - \omega)(z - \omega^2) \cdots (z - \omega^{N-1}).$$

By high school algebra, you can easily verify that the LHS can also be factored to

$$z^N - 1 = (z - 1)(z^{N-1} + z^{N-2} + \dots + z^2 + z + 1).$$

We can therefore equate the RHS of the last two equations (and divide out the common $(z - 1)$) to yield

$$z^{N-1} + \dots + z + 1 = (z - \omega) \cdots (z - \omega^{N-1}).$$

This equation has a number of easily confirmed consequences that we will use going forward. We list each one as an exercise.

Exercises

(a) Show that when $0 \leq l < N$,

$$\omega^{l(N-1)} + \omega^{l(N-2)} + \dots + \omega^l + 1 = \begin{cases} N, & l = 0 \\ 0, & 0 < l < N \end{cases}.$$

Hint: Plug ω^l in for z in the last equation.

(b) Show that when $-N \leq l < N$,

$$\omega^{l(N-1)} + \omega^{l(N-2)} + \dots + \omega^l + 1 = \begin{cases} N, & l = 0, -N \\ 0, & -N < l < N, \quad l \neq 0 \end{cases}.$$

Hint: Prove that, for all l , $\omega^{l+N} = \omega^l$, and apply the last result.

(c) Show that for *any* integer l ,

$$\omega^{l(N-1)} + \omega^{l(N-2)} + \dots + \omega^l + 1 = \begin{cases} N, & l \equiv 0 \pmod{N} \\ 0, & l \not\equiv 0 \pmod{N} \end{cases}.$$

Hint: Add (or subtract) an integral multiple of N to (or from) l to bring it into the interval $[-N, N)$ and call l' the new value of l . Argue that $\omega^{l'(N-2)} = \omega^{l(N-2)}$, so this doesn't change the value of the sum. Finally, apply the last result.

(d) Show that for $0 \leq j < N$ and $0 \leq m < N$,

$$\omega^{(j-m)(N-1)} + \omega^{(j-m)(N-2)} + \dots + \omega^{(j-m)} + 1 = \begin{cases} N, & j = m \\ 0, & j \neq m \end{cases}.$$

Hint: Set $l = j - m$ and apply the last result.

Kronecker Delta

All of these exercises have a common theme: When we add all of the N th roots-of-unity raised to certain powers, there is a massive cancellation causing the sum to be zero except for special cases. I'll repeat the last result using *Kronecker delta symbol*, as this will be very important in some upcoming algorithms. First, what is the "Kronecker delta?"

Kronecker Delta. δ_{kj} , the **Kronecker delta**, is the mathematical way to express *anything* that is to be 0 unless the index $k = j$, in which case it is 1,

$$\delta_{kj} = \begin{cases} 1, & \text{if } k = j \\ 0, & \text{otherwise} \end{cases}.$$

You will see Kronecker delta throughout the course (and beyond), starting immediately, as I rewrite result (d) using it.

$$\sum_{k=0}^{N-1} \omega^{(j-m)k} = N\delta_{jm}.$$

We Are In the Air

Our flight is officially off the ground. We've learned or reviewed everything needed about complex arithmetic to sustain ourselves through the unusual terrain of quantum computing. There's a second subject that we will want to master, and that'll be our first fly-over. Look out the window, as it should be coming into view now: *linear algebra*.

Chapter 2

Real Vector Spaces

2.1 Vector Spaces for Quantum Computing

In each of these lessons we will inch a little closer to a complete understanding of the *qubit*. At the moment, all we know about a qubit is that it can be written as a collection of abstract symbols,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where last lecture revealed two of the symbols, α and β , to be *complex numbers*. Today, we add a little more information about it.

*The qubit, $|\psi\rangle$, is a **vector** quantity.*

Every vector lives in a world of other vectors, all of which bear certain similarities. That world is called a *vector space*, and two of the similarities that all vectors in any given vector space share are

- its *dimension* (i.e., is it two dimensional? three dimensional? 10 dimensional?), and
- the kinds of ordinary numbers – or *scalars* – that support the vector space operations (i.e., does it use real numbers? complex numbers? the tiny set of integers $\{0, 1\}$?).

In this lecture we'll restrict our study to *real vector spaces* – those whose scalars are the real numbers. We'll get to complex vector spaces in a couple days. As for *dimension*, we'll start by focusing on *two-dimensional* vector spaces, and follow that by meeting some higher dimensional spaces.

2.2 Vectors and Vector Spaces

\mathbb{R}^2 , sometimes referred to as *Euclidean 2-space*, will be our poster child for vector spaces, and we'll see that everything we learn about \mathbb{R}^2 applies equally well to higher dimensional vector spaces like \mathbb{R}^3 (three-dimensional), \mathbb{R}^4 (four-dimensional) or \mathbb{R}^n (n -dimensional, for any positive integer, n).

With that overview, let's get back down to earth and define the two-dimensional real vector space \mathbb{R}^2 .

2.2.1 Standard Equipment: The Axioms

Every vector space has some rules, called *the axioms*, that define its *objects* and the way in which those objects can be combined. When you learned about the integers, you were introduced to the objects, $(\dots - 3, -2, -1, 0, 1, 2, 3, \dots)$ and the rules $(2 + 2 = 4, (-3)(2) = -6, \text{etc.})$. We now define the axioms – the objects and rules – for the special vector space \mathbb{R}^2 .

The Objects

A vector space requires two sets of things to make sense: the *scalars* and the *vectors*.

Scalars

A vector space is based on some number system. For example, \mathbb{R}^2 is built on the *real numbers*, \mathbb{R} . These are the *scalars* of the vector space. In math lingo the scalars are referred to as the *underlying field*.

Vectors

The other set of objects that constitute a vector space are the *vectors*. In the case of \mathbb{R}^2 they are *ordered pairs*,

$$\mathbf{r} = \begin{pmatrix} 3 \\ -7 \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} 500 \\ 1 + \pi \end{pmatrix}, \quad \hat{\mathbf{x}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \hat{\mathbf{y}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

You'll note that I use **boldface** to name the vectors. That's to help distinguish a vector variable name like \mathbf{r} , \mathbf{x} or \mathbf{v} , from a scalar name, like a , x or α . Also, we will usually consider vectors to be written as *columns* like $\begin{pmatrix} 3 \\ -7 \end{pmatrix}$, not *rows* like $(3, -7)$, although this varies by author and context.

A more formal and complete description of the vectors in \mathbb{R}^2 is provided using set notation,

$$\mathbb{R}^2 \equiv \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \mid x, y \in \mathbb{R} \right\}.$$

(See figure 2.1.) This is somewhat incomplete, though, because it only tells what the

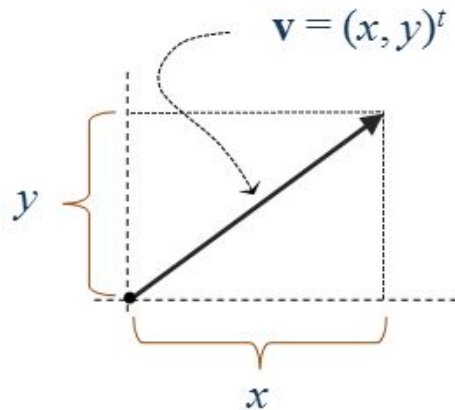


Figure 2.1: A vector in \mathbb{R}^2

set of vectors is and fails to mention the *scalars* or *rules*.

Notation: Vector *Transpose*. When writing vectors in a paragraph, I may write $(3, -7)$ when I'm being lazy, or $(3, -7)^t$, when I'm being good. The latter is pronounced “ $(3, -7)$ *transpose*”, and it means “turn this row vector into a column vector.” It goes both ways. If I had a column vector and wanted to turn it into a row-vector, I would tag a superscript $()^t$ onto the column vector.

Vocabulary. When we want to emphasize which set of scalars we are using for a given vector space, we would say that the vectors space is *defined over [insert name of scalars here]*. For example, the vector space \mathbb{R}^2 is a *vector space over the reals* (or *over \mathbb{R}*).

The Rules

Two *operations* and their properties are required for the above objects.

Vector Addition

We must be able to combine two *vectors* to produce a third *vector*,

$$\mathbf{v} + \mathbf{w} \longmapsto \mathbf{u}.$$

In \mathbb{R}^2 this is defined by adding vectors component-wise,

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 + x_2 \\ y_1 + y_2 \end{pmatrix}.$$

(See figure 2.2.)

Vector addition must obey the required rules. They are:

- **Zero Vector.** Every vector space must have a unique *zero vector* (a.k.a. *additive identity*), denoted by boldface $\mathbf{0}$, which has the property that $\mathbf{v} + \mathbf{0} = \mathbf{0} + \mathbf{v} = \mathbf{v}$, for all \mathbf{v} in the space. For \mathbb{R}^2 the zero vector is $(0, 0)^t$.

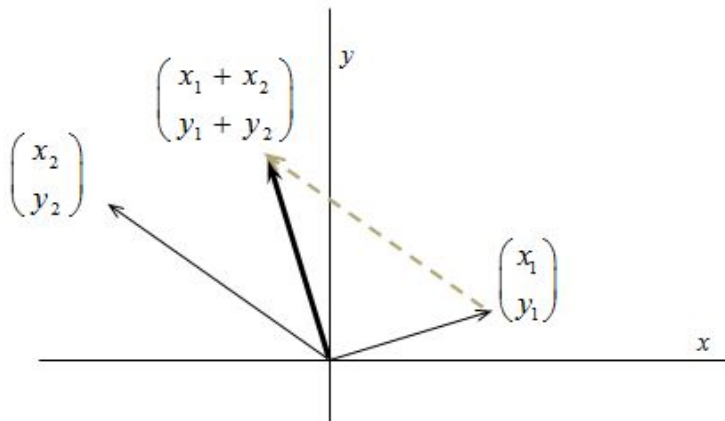


Figure 2.2: Vector addition in \mathbb{R}^2

- **Vector Opposites.** For each vector, \mathbf{v} , there must be an *additive inverse*, i.e., a unique $-\mathbf{v}$, such that $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$. In \mathbb{R}^2 the additive inverse of $(x, y)^t$ is $(-x, -y)^t$.
- **Commutativity and Associativity.** The vectors must satisfy the reasonable conditions $\mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}$ and $(\mathbf{v} + \mathbf{w}) + \mathbf{u} = \mathbf{v} + (\mathbf{w} + \mathbf{u})$, for all \mathbf{v} , \mathbf{w} and \mathbf{u} .

Scalar Multiplication

We can “scale” a vector by a *scalar*, that is, we can multiply a *vector* by a *scalar* to produce another *vector*,

$$c\mathbf{v} \mapsto \mathbf{w}.$$

In \mathbb{R}^2 this is defined in the expected way,

$$5 \begin{pmatrix} 3 \\ -7 \end{pmatrix} = \begin{pmatrix} 15 \\ -35 \end{pmatrix}, \quad c \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} cx_1 \\ cy_1 \end{pmatrix}.$$

(See figure 2.3.)

Scalar multiplication must obey the required rules. They are:

- **Scalar Identity.** $1\mathbf{v} = \mathbf{v}$, for any vector \mathbf{v} .
- **Associativity.** $(ab)\mathbf{v} = a(b\mathbf{v})$, for any vector \mathbf{v} and scalars a, b .
- **Distributivity.** $c(\mathbf{v} + \mathbf{w}) = c\mathbf{v} + c\mathbf{w}$ and $(c + d)(\mathbf{v}) = c\mathbf{v} + d\mathbf{v}$, for any vectors \mathbf{v}, \mathbf{w} and scalars c, d .

[**Exercise.** Assuming

$$\mathbf{r} = \begin{pmatrix} 3 \\ -7 \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} 500 \\ 1 + \pi \end{pmatrix}, \quad \hat{\mathbf{x}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

- verify the associativity of vector addition using these three vectors,
- multiply each by the scalar $1/\pi$, and
- verify the distributivity using the first two vectors and the scalar $1/\pi$.]

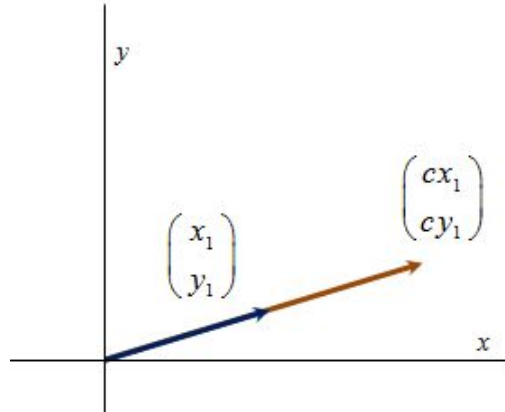


Figure 2.3: Scalar multiplication in \mathbb{R}^2

2.2.2 Optional Equipment: Inner Products, Modulus and Orthogonality

The vector spaces we encounter will have a *metric* – a way to measure distances. The metric is a side effect of a *dot product*, and we define this optional feature now.

[**Caution.** The phrase “optional equipment” means that not every vector space has an inner product, not that this is optional reading. It is crucial that you master this material for quantum mechanics and quantum computing since all of our vector spaces will have inner products and we will use them constantly.]

Dot (or Inner) Product

When a vector space has this feature, it provides a way to *multiply two vectors* in order to produce a *scalar*,

$$\mathbf{v} \cdot \mathbf{w} \longmapsto c.$$

In \mathbb{R}^2 this is called the *dot product*, but in other contexts it may be referred to as an *inner product*. There can be a difference between a *dot product* and an *inner product* (in complex vector spaces, for example), so don’t assume the terms are synonymous. However, for \mathbb{R}^2 they are, with both defined by

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = x_1x_2 + y_1y_2.$$

Inner products can be defined differently in different vector spaces,. However they are defined, they must obey certain properties to get the title *inner* or *dot product*. I won’t burden you with them all, but one that is very common is a *distributive property*.

$$\mathbf{v} \cdot (\mathbf{w}_1 + \mathbf{w}_2) = \mathbf{v} \cdot \mathbf{w}_1 + \mathbf{v} \cdot \mathbf{w}_2.$$

[**Exercise.** Look-up and list another property that an inner product must obey.]

[**Exercise.** Prove that the dot product in \mathbb{R}^2 , as defined above, obeys the distributive property.]

When we get to *Hilbert spaces*, there will be more to say about this.

Length (or Modulus or Norm)

An inner product, when present in a vector space, confers each vector with a *length* (a.k.a. *modulus* or *norm*), denoted by either $|\mathbf{v}|$ or $\|\mathbf{v}\|$. The length (in most situations, including ours) of each vector must be a non-negative real number, even for complex vector spaces coming later. So,

$$|\mathbf{v}| \geq 0.$$

For our friend \mathbb{R}^2 , if

$$\mathbf{v} = \begin{pmatrix} x \\ y \end{pmatrix},$$

then the *length of \mathbf{v}* is defined to be

$$\|\mathbf{v}\| \equiv \sqrt{\mathbf{v} \cdot \mathbf{v}} = \sqrt{\begin{pmatrix} x \\ y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}} = \sqrt{x^2 + y^2},$$

positive square root, assumed.

Orthogonality

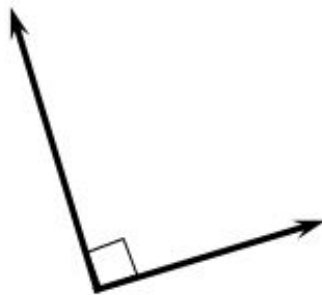


Figure 2.4: Orthogonal vectors

If two vectors, \mathbf{v} and \mathbf{w} , have a dot product which is zero,

$$\mathbf{v} \cdot \mathbf{w} = 0,$$

we say that they are *orthogonal* or *mutually perpendicular*. In the relatively visualizable spaces \mathbb{R}^2 and \mathbb{R}^3 , we can imagine the line segments $\overline{\mathbf{0}\mathbf{v}}$ and $\overline{\mathbf{0}\mathbf{w}}$ forming right-angles with one another. (See figure 2.4.)

Examples of Orthogonal Vectors. A well known orthogonal pair is

$$\left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}.$$

Another is

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix} \right\}.$$

[**Exercise.** Use the definition of dot product to show that each of the two sets of vectors, listed above, is orthogonal.]

Examples of Non-Orthogonal Vectors. An example of a set that is *not* orthogonal is

$$\left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}.$$

More Vocabulary. If a set of vectors is both orthogonal and each vector in the set has *unit length* (norm = 1), it is called *orthonormal*.

The set

$$\left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

is orthonormal, while the set

$$\left\{ \begin{pmatrix} 1.1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1.1 \end{pmatrix} \right\}$$

is orthogonal (check it) but not orthonormal.

Some Unexpected Facts

- A *length* seems like it should always be non-negative, but this won't be true in some common vector spaces of first-year physics (hint: special relativity).
- If $\mathbf{v} \neq \mathbf{0}$, we would expect $\mathbf{v} \cdot \mathbf{v} > 0$, strictly greater than zero. However, not so when we deal with the *mod-2 vector spaces* of quantum computing.

These two unexpected situations suggest that we need a term to distinguish the typical, well-behaved inner products from those that are off-beat or oddball. That term is *positive definiteness*.

Positive Definite Property. An inner-product is usually required to be *positive definite*, meaning

- $\mathbf{v} \cdot \mathbf{v} \geq 0$, and

- $\mathbf{v} \neq \mathbf{0} \Rightarrow \|\mathbf{v}\| > 0.$

When a proposed inner product fails to meet these conditions, it is often not granted the status *inner* (or *dot*) *product* but is instead called a *pairing*. When we come across a pairing, I'll call it to your attention, and we can take appropriate action.

[**Exercise.** Assume

$$\mathbf{r} = \begin{pmatrix} 3 \\ -7 \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} 500 \\ 1 + \pi \end{pmatrix} \quad \text{and} \quad \hat{\mathbf{x}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

- Compute the dot product of every pair in the group.
- Compute the norm of every vector in the group.
- Prove that the only vector in \mathbb{R}^2 which has zero norm is the zero vector, $\mathbf{0}$.
- For each vector, find an (i) orthogonal and (ii) non-orthogonal companion.]

2.2.3 A Bigger Vector Space: \mathbb{R}^3

Even though we have not learned what *dimension* means, we have an intuition that \mathbb{R}^2 is somehow two-dimensional: we graph its vectors on paper and it seems flat. Let's take a leap now and see what it's like to dip our toes into the higher dimensional vector space \mathbb{R}^3 (over \mathbb{R}). \mathbb{R}^3 is the set of all *triples* (or 3-tuples) of real numbers,

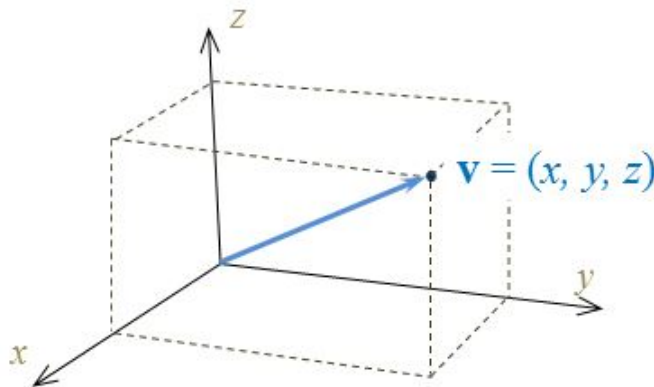


Figure 2.5: A vector in \mathbb{R}^3

$$\mathbb{R}^3 \equiv \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mid x, y, z \in \mathbb{R} \right\}.$$

(This only tells us what the *vectors* are, but I'm telling you now that the *scalars* continue to be \mathbb{R} .)

Examples of vectors in \mathbb{R}^3 are

$$\begin{pmatrix} 2 \\ -1 \\ 3.5 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} \pi/2 \\ 100 \\ -9 \end{pmatrix}.$$

It's harder to graph these objects, but it can be done using three-D sketches (See figure 2.5.).

[**Exercise.** Repeat some of the examples and exercises we did for \mathbb{R}^2 for this richer vector space. In particular, define vector addition, scalar multiplication, dot products, etc.]

2.2.4 Preview of a Complex Vector Space: \mathbb{C}^2

We'll roll out complex vector spaces in their full glory when we come to the *Hilbert space* lesson, but it won't hurt to put our cards on the table now. The most useful vector spaces in this course will be ones in which the *scalars* are the *complex numbers*. The simplest example is \mathbb{C}^2 .

Definition. \mathbb{C}^2 is the set of all ordered pairs of *complex* numbers,

$$\mathbb{C}^2 \equiv \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \mid x, y \in \mathbb{C} \right\}.$$

You can verify that this is a vector space and guess its dimension and how the inner product is defined. Then check your guesses online or look ahead in these lectures. All I want to do here is introduce \mathbb{C}^2 so you'll be ready to see vectors that have complex components.

2.3 Bases for a Vector Space

If someone were to ask, “*What is the single most widely used mathematical concept is in quantum mechanics and quantum computing,*” I would answer, “*the vector basis.*” (At least that's my answer today.) It is used or implied at every turn, so much so, that one forgets it's there. But it *is* there, always. Without a solid understanding of what a basis is, how it affects our window into a problem, and how different bases relate to one another, one cannot participate in the conversation. It's time to check off the “what is a basis” box.

Incidentally, we are transitioning now to properties that are not axioms. They are consequences of the axioms, i.e., we can *prove* them.

2.3.1 Linear Combination (or Superposition)

Whenever you have a finite set of two or more vectors, you can combine them using both scalar multiplication and vector addition. For example, with two vectors, \mathbf{v} , \mathbf{w} ,

and two scalars, a , b , we can form the vector \mathbf{u} by taking

$$\mathbf{u} = a\mathbf{v} + b\mathbf{w}.$$

Mathematicians call it a *linear combination* of \mathbf{v} and \mathbf{w} . Physicists call it a *superposition* of the two vectors. *Superposition* or *linear combination*, the idea is the same. We are "weighting" each of the vectors, \mathbf{v} and \mathbf{w} , by scalar weights, a and b , respectively, then adding the results. In a sense, the two scalars tell the relative amounts of each vector that we want in our result. (However, if you lean too heavily on that metaphor, you will find yourself doing some fast talking when your students ask you about negative numbers and complex scalars, so don't take it too far.)

The concept extends to sets containing more than two vectors. Say we have a finite set of n vectors $\{\mathbf{v}_k\}$ and corresponding scalars $\{c_k\}$. Now a linear combination would be expressed either long-hand or using summation notation,

$$\begin{aligned}\mathbf{u} &= c_0\mathbf{v}_0 + c_1\mathbf{v}_1 + \dots + c_{n-1}\mathbf{v}_{n-1} \\ &= \sum_{k=0}^{n-1} c_k\mathbf{v}_k.\end{aligned}$$

Linear combinations are fundamental to understanding vector bases.

[Exercise.] Form three different linear combinations of the vectors

$$\mathbf{r} = \begin{pmatrix} 3 \\ -7 \end{pmatrix}, \quad \mathbf{a} = \begin{pmatrix} 500 \\ 1 + \pi \end{pmatrix}, \quad \hat{\mathbf{x}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

showing the original scalars you chose for each one. Reduce each linear combination into the simplest form you can.]

[Exercise.] Play around with those three to see if you can express any one of them as a linear combination of the other two. **Hint:** You can, no matter how you split them up.]

2.3.2 The Notion of Basis

One can find a subset of the vectors (usually a very tiny fraction of them compared to the infinity of vectors in the space) which can be used to "generate" all the other vectors through linear combinations. When we have such a subset that is, in a sense, *minimal* (to be clarified), we call it a *basis* for the space.

The Natural Basis

In \mathbb{R}^2 , only two vectors are needed to produce – through linear combination – all the rest. The most famous basis for this space is the *standard* (or *natural* or *preferred*) basis, which I'll call \mathcal{A} for now,

$$\mathcal{A} = \{\hat{\mathbf{x}}, \hat{\mathbf{y}}\} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}.$$

For example, the vector $(15, 3)^t$ can be expressed as the linear combination

$$\begin{pmatrix} 15 \\ 3 \end{pmatrix} = 15 \hat{\mathbf{x}} + 3 \hat{\mathbf{y}}.$$

Note. In the diagram that follows, the vector pictured is not intended to be $(15, 3)^t$. (See figure 2.6.)

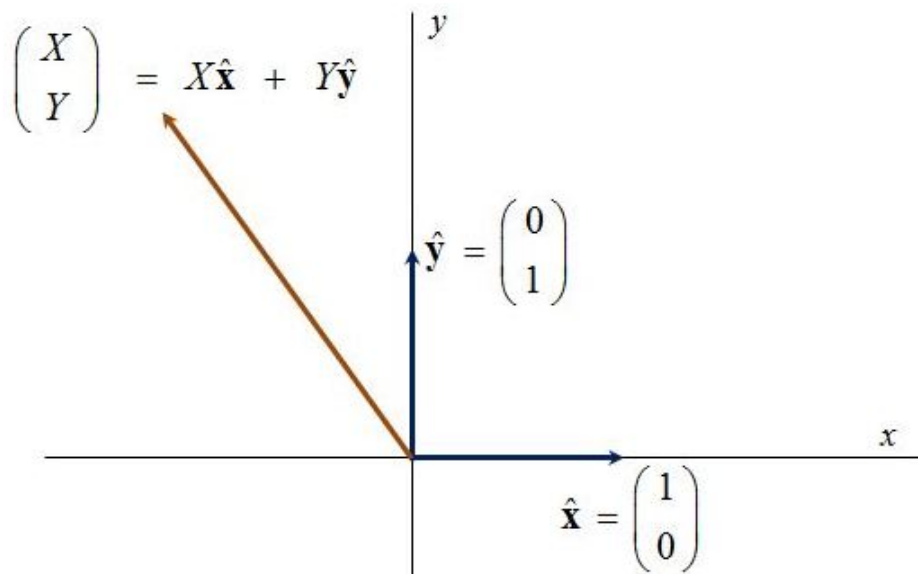


Figure 2.6: A vector expressed as linear combination of $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$

Notation. It is common to use the small-hat, $\hat{}$, on top of a vector, as in $\hat{\mathbf{x}}$, to denote a vector that has unit length. For the time being, we'll reserve the two symbols, $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ for the vectors in the natural basis of \mathbb{R}^2 .

Properties of a Basis

Hang on, though, because we haven't actually defined a basis yet; we only saw an example of one. A basis has to have two properties called (i) *linear independence* and (ii) *completeness* (or *spanning*).

- (i) **Linear Independence.** A set of vectors is *linearly independent* if we cannot express any one of them as a *linear combination* of the others. If we *can* express one as a linear combination of the others, then it is called a *linear dependent* set. *A basis must be linearly independent.*

We ask whether the following three vectors making up the set \mathcal{A}' is a basis. If so, it must be *linearly independent*. Is it?

$$\mathcal{A}' \equiv \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 2 \end{pmatrix} \right\}$$

Since we can express $(3, 2)^t$ as

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} = 3 \hat{\mathbf{x}} + 2 \hat{\mathbf{y}},$$

\mathcal{A}' is not a linearly independent set of vectors.

Restating, to exercise our new vocabulary, $(3, 2)^t$ is *weighted sum* of $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, where 3 is the *weight* of the $\hat{\mathbf{x}}$, and 2 is the *weight* of the $\hat{\mathbf{y}}$. Thus, it is a *linear combination* of those two and therefore *linearly dependent* on them. The requirement that every basis be *linear independent* means that there cannot be even one vector in the basis which can be expressed as a linear combination of the others, so \mathcal{A}' cannot be a basis for \mathbb{R}^2 .

[Exercise.] Is it also true that $(1, 0)^t$ can be expressed as a linear combination of the other two, $(0, 1)^t$ and $(3, 2)^t$? **Hint:** Yes. See if you can fiddle with the equation to find out what scalars are needed to get $(1, 0)^t$ as a linear combination of the other two.]

[Exercise.] Show that the zero vector $(0, 0)^t$ is never part of a linearly independent set. Hint: Find the coefficients of the remaining vectors that easily produce $(0, 0)^t$ as a linear combination.]

- (ii) **Completeness (or Spanning).** *Completeness* of a set of vectors means that the set *spans* the entire vector space, which in turn means that any vector in the space can be expressed as a linear combination of vectors in that set. *A basis must span the space.*

Does the following set of two vectors, \mathcal{A}'' , span \mathbb{R}^3 ?

$$\mathcal{A}'' \equiv \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

Since we *cannot* express $(3, -1, 2)^t$ as a linear combination of these two, i.e.,

$$\begin{pmatrix} 3 \\ -1 \\ 2 \end{pmatrix} \neq x \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + y \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

for any $x, y \in \mathbb{R}$, \mathcal{A}'' does not span the set of vectors in \mathbb{R}^3 . In other words, \mathcal{A}'' is not *complete*.

[Exercise.] Find two vectors, such that if either one (individually) were added to \mathcal{A}'' , the augmented set would span the space.]

[Exercise.] Find two vectors, such that, if either one were added to \mathcal{A}'' , the augmented set would still fail to be a spanning set.]

Definition of Basis

We now know enough to formally define a vector space basis.

Basis. A **basis** is a set of vectors that is **linearly independent** and **complete** (spans the space).

Another way to phrase it is that a basis is a *minimal* spanning set, meaning we can't remove any vectors from it without losing the spanning property.

Theorem. All bases for a given vector space have the same number of elements.

This is easy to prove (you can do it as an [exercise] if you wish). One consequence is that all bases for \mathbb{R}^2 must have two vectors, since we know that the natural basis has two elements. Similarly, all bases for \mathbb{R}^3 must have three elements.

Definition. The *dimension* of a vector space is the number of elements in any basis.

[**Exercise.** Describe some vector space (over \mathbb{R}) that is 10-dimensional. **Hint:** The set of five-tuples, $\{(x_0, x_1, x_2, x_3, x_4)^t \mid x_k \in \mathbb{R}\}$ forms a five-dimensional vector space over \mathbb{R} .]

Here is an often used fact that we should prove.

Theorem. If a set of vectors $\{\mathbf{v}_k\}$, is orthonormal, it is necessarily linearly independent.

Proof. We'll assume the theorem is false and arrive at a contradiction. So, we pretend that $\{\mathbf{v}_k\}$ is an orthonormal collection, yet one of them, say \mathbf{v}_0 , is a linear combination of the others,

$$\mathbf{v}_0 = \sum_{k=1}^{n-1} c_k \mathbf{v}_k,$$

where not *all* the c_k can be 0, since that would imply that $\mathbf{v}_0 = \mathbf{0}$, which cannot be a member of any orthonormal set (remember from earlier?). By orthonormality $\mathbf{v}_0 \cdot \mathbf{v}_k = 0$ for all $k \neq 0$, but of course $\mathbf{v}_0 \cdot \mathbf{v}_0 = 1$, so we get the following chain of equalities:

$$1 = \mathbf{v}_0 \cdot \mathbf{v}_0 = \mathbf{v}_0 \cdot \left(\sum_{k=1}^{n-1} c_k \mathbf{v}_k \right) = \sum_{k=1}^{n-1} c_k (\mathbf{v}_0 \cdot \mathbf{v}_k) = 0,$$

a contradiction. QED

Notice that even if the vectors were orthogonal, weaker than their being orthonormal, they would still have to be linearly independent.

Alternate Bases

There are many different pairs of vectors in \mathbb{R}^2 which can be used as a basis. Every basis has exactly two vectors (by our theorem). Here is an alternate basis for \mathbb{R}^2 :

$$\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1\} = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 4 \\ -1 \end{pmatrix} \right\}.$$

For example, the vector $(15, 3)^t$ can be expressed as the linear combination

$$\begin{pmatrix} 15 \\ 3 \end{pmatrix} = (27/5) \mathbf{b}_0 + (12/5) \mathbf{b}_1.$$

[**Exercise.** Multiply this out to verify that the *coefficients*, $27/5$ and $12/5$ work for that vector and basis.]

And here is yet a third basis for \mathbb{R}^2 :

$$\mathcal{C} = \{\mathbf{c}_0, \mathbf{c}_1\} = \left\{ \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}, \begin{pmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} \right\},$$

with the vector $(15, 3)^t$ expressible as the linear combination

$$\begin{pmatrix} 15 \\ 3 \end{pmatrix} = (9\sqrt{2}) \mathbf{c}_0 + (-6\sqrt{2}) \mathbf{c}_1.$$

[**Exercise.** Multiply this out to verify that the *coefficients*, $9\sqrt{2}$ and $-6\sqrt{2}$ work for that vector and basis.]

Note. In the diagrams, the vector pictured is not intended to be $(15, 3)^t$. (See figures 2.7 and 2.8.)

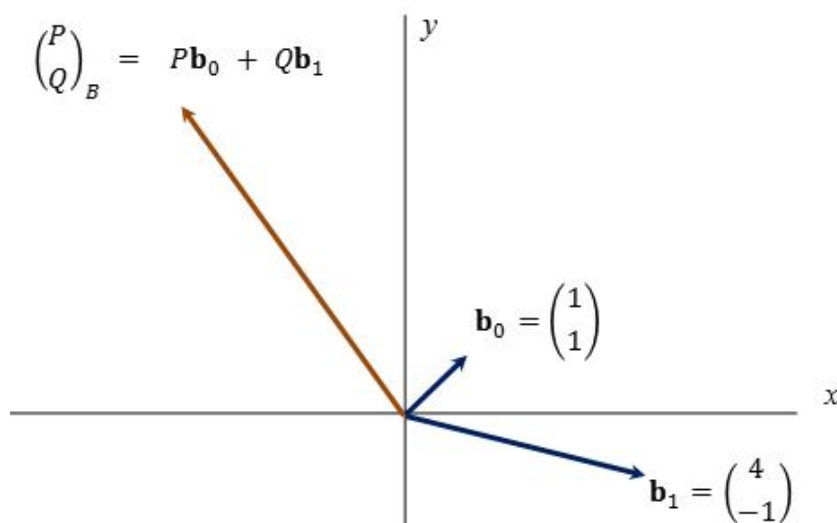


Figure 2.7: A vector expressed as linear combination of \mathbf{b}_0 and \mathbf{b}_1

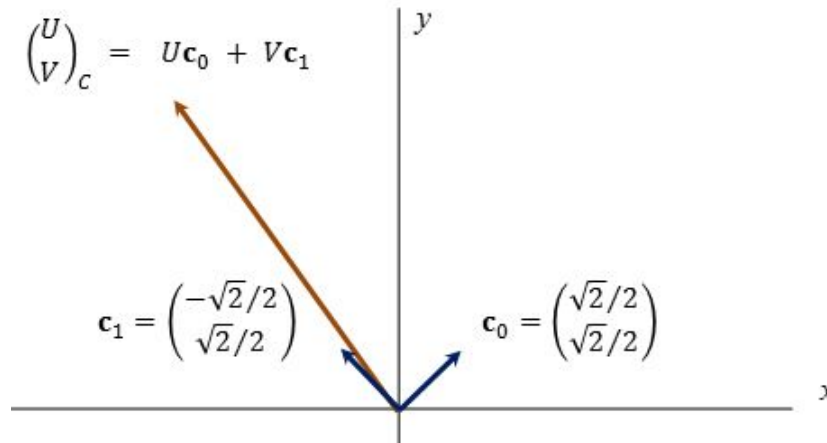


Figure 2.8: A vector expressed as linear combination of \mathbf{c}_0 and \mathbf{c}_1

Expanding Along a Basis. When we want to call attention to the specific basis we are using to express (i.e., to produce using the wrench of *linear combination*) a vector, \mathbf{v} , we would say we are *expanding* \mathbf{v} along the so-and-so basis, e.g., "we are *expanding* \mathbf{v} along the natural basis," or "let's *expand* \mathbf{v} along the \mathcal{B} basis."

Orthonormal Bases

We are particularly interested in bases whose vectors have unit length and are mutually perpendicular. Not surprisingly, because of the definition of orthonormal vectors, above, such bases are called *orthonormal bases*.

$$\begin{aligned} \mathcal{A} \text{ is orthonormal: } & \hat{\mathbf{x}} \cdot \hat{\mathbf{x}} = \hat{\mathbf{y}} \cdot \hat{\mathbf{y}} = 1, & \hat{\mathbf{x}} \cdot \hat{\mathbf{y}} &= 0 \\ \mathcal{B} \text{ is not orthonormal: } & \mathbf{b}_0 \cdot \mathbf{b}_0 = 2 \neq 1, & \mathbf{b}_0 \cdot \mathbf{b}_1 &= 3 \neq 0 \\ \mathcal{C} \text{ is orthonormal: } & ([\text{Exercise}]) \end{aligned}$$

If they are *mutually perpendicular* but do not have unit length, they are almost as useful. Such a basis is called an *orthogonal basis*. If you get a basis to be orthogonal, your hard work is done; you simply divide each basis vector by its *norm* in order to make it orthonormal.

2.3.3 Coordinates of Vectors

As a set of vectors, \mathbb{R}^2 consists *literally* of ordered pairs of real numbers, $\{ (x, y) \}$ or if you like, $\{ (x, y)^t \}$. Those are the vector objects of the vector space. However, each vector also has *coordinates* relative to some basis, namely the specific scalars needed to express that vector as a linear combination of the basis vectors. For example, if \mathbf{v} is expanded along the natural basis,

$$\mathbf{v} = v_x \hat{\mathbf{x}} + v_y \hat{\mathbf{y}},$$

its *coordinates* relative to the natural basis are v_x and v_y . In other words, the coordinates are just weighting factors needed to *expand* that vector in that given basis.

Vocabulary. Sometimes the term *coefficient* is used instead of *coordinate*.

If we have a different basis, like $\mathcal{B} = \{\mathbf{b}_0, \mathbf{b}_1\}$, and we expand \mathbf{v} along that basis,

$$\mathbf{v} = v_0 \mathbf{b}_0 + v_1 \mathbf{b}_1,$$

then its *coordinates* (*coefficients*) relative to the \mathcal{B} basis are v_0 and v_1 .

As you see, the same vector, \mathbf{v} , will have different coordinates relative to different bases. However, the ordered pair that describes the pure object – the vector, $(x, y)^t$, itself – does not change.

For some vector spaces, \mathbb{R}^2 being a prime example, it is easy to confuse the coefficients with the actual vector. Relative to the *natural* (*standard*) basis, the coordinates happen to be the same as the numbers in the vector itself, so the vector and the coordinates, when written in parentheses, are indistinguishable. If we need to clarify that we are expressing coordinates of a vector, and not the raw vector itself, we can label the ordered pair appropriately. Such a label would be the name of the basis being used. It's easier to *do* than it is to *say*. Here is the vector $(15, 3)^t$, first expanded along the natural basis,

$$\begin{pmatrix} 15 \\ 3 \end{pmatrix} = 15 \hat{\mathbf{x}} + 3 \hat{\mathbf{y}},$$

and now shown along with its coordinates in the standard basis,

$$\begin{pmatrix} 15 \\ 3 \end{pmatrix} = \begin{pmatrix} 15 \\ 3 \end{pmatrix}_{\mathcal{A}}.$$

For the non-preferred bases of the previous section, the coordinates for $(15, 3)$, similarly labeled, are written as

$$\begin{pmatrix} 15 \\ 3 \end{pmatrix} = \begin{pmatrix} 27/5 \\ 12/5 \end{pmatrix}_{\mathcal{B}} = \begin{pmatrix} 9\sqrt{2} \\ -6\sqrt{2} \end{pmatrix}_{\mathcal{C}}$$

Computing Expansion Coefficients along a Basis

In physics, it is common to use the term *expansion coefficients* to mean the *coordinates* relative to some basis. This terminology seems to be used especially if the basis happens to be *orthonormal*, although you may see the term used even if it isn't.

$$\begin{pmatrix} 15 \\ 3 \end{pmatrix} = (9\sqrt{2}) \mathbf{c}_0 + (-6\sqrt{2}) \mathbf{c}_1,$$

so $9\sqrt{2}$ and $-6\sqrt{2}$ are the “expansion coefficients” of

$$\begin{pmatrix} 15 \\ 3 \end{pmatrix}$$

along the \mathcal{C} basis.

If our basis happens to be *orthonormal*, there is a special trick we can use to find the coordinates – or expansion coefficients – relative to that orthonormal basis.

Let's say we would like to expand \mathbf{v} along the \mathcal{C} basis, but we only know it in the natural basis, \mathcal{A} . We also know the \mathcal{C} basis vectors, \mathbf{c}_0 and \mathbf{c}_1 , in the natural basis \mathcal{A} . In other words, we would like, but don't yet have,

$$\mathbf{v} = \alpha_0 \mathbf{c}_0 + \alpha_1 \mathbf{c}_1,$$

i.e., we don't know the scalar weights α_0, α_1 . An easy way to find the α_k is to "dot" \mathbf{v} with the two \mathcal{C} basis vectors. We demonstrate why this works by computing α_0 :

$$\begin{aligned} \mathbf{c}_0 \cdot \mathbf{v} &= \mathbf{c}_0 \cdot (\alpha_0 \mathbf{c}_0 + \alpha_1 \mathbf{c}_1) \\ &= \mathbf{c}_0 \cdot \alpha_0 \mathbf{c}_0 + \mathbf{c}_0 \cdot \alpha_1 \mathbf{c}_1 \\ &= \alpha_0 (\mathbf{c}_0 \cdot \mathbf{c}_0) + \alpha_1 (\mathbf{c}_0 \cdot \mathbf{c}_1) \\ &= \alpha_0 (1) + \alpha_1 (0) = \alpha_0 \end{aligned}$$

[**Exercise.** Justify each equality in this last derivation using the axioms of vector space and assumption of orthonormality of \mathcal{C} .]

[**Exercise.** This trick works almost as well with an *orthogonal* basis which does not happen to be *orthonormal*. We just have to add a step; when computing the expansion coefficient for the basis vector, \mathbf{c}_k , we must divide the dot product by $|\mathbf{c}_k|^2$. Prove this and give an example.]

Thus, dotting by \mathbf{c}_0 produced the expansion coefficient α_0 . Likewise, to find α_1 just dot the \mathbf{v} with \mathbf{c}_1 .

For the specific vector $\mathbf{v} = (15, 3)^t$ and the basis \mathcal{C} , let's verify that this actually works for, say, the 0th expansion coefficient:

$$\mathbf{c}_0 \cdot \mathbf{v} = \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} \cdot \begin{pmatrix} 15 \\ 3 \end{pmatrix} = 15\sqrt{2}/2 + 3\sqrt{2}/2 = 9\sqrt{2} \checkmark$$

The reason we could add the check-mark is that this agrees with the expression we had earlier for the vector \mathbf{v} expanded along the \mathcal{C} basis.

Remark. We actually did not have to know things in terms of the natural basis \mathcal{A} in order for this to work. If we had known the coordinates of \mathbf{v} in some other basis (it doesn't even have to be orthonormal), say \mathcal{B} , and we also knew coordinates of the \mathcal{C} basis vectors with respect to \mathcal{B} , then we could have done the same thing.

[**Exercise.** If you're up for it, prove this.]

2.3.4 Independence of Basis (or *Not?*)

The definition of inner product assumed that the n -tuples were, themselves, the vectors and not some coordinate representation expanded along a specific basis. Now

that we know a vector can have different coordinates relative to different bases, we ask “is the inner product formula that I gave independent of basis?,” i.e., can we use the coordinates, (d_x, d_y) and (e_x, e_y) relative to *some basis* – rather than the numbers in the raw vector ordered pair – to compute the inner product using the simple $d_x e_x + d_y e_y$? In general the answer is *no*.

[**Exercise.** Compute the length of the vector $(15, 3)^t$ by dotting it with itself. Now do the same thing, but this time compute using that vector’s coordinates relative to the three bases \mathcal{A} , \mathcal{B} and \mathcal{C} *through use of the imputed formula given above*. Do you get the same answers? Which bases’ coordinates give the right inner-product answer?]

However, when working with *orthonormal bases*, the answer is *yes*, one *can* use coordinates relative to that basis, instead of the pure vector coordinates, and apply the simple formula to the *coordinates*.

[**Exercise.** Explain the results of the last exercise in light of this new assertion.]

[**Exercise.** See if you can prove the last assertion.]

Note. There is a way to use non-orthonormal basis coordinates to compute dot products, but one must resort to a more elaborate matrix multiplication, the details of which we shall skip (but it’s a nice [**Exercise**] should you wish to attempt it).

2.4 Subspaces

The set of vectors that are scalar multiples of a single vector, such as

$$\left\{ \begin{pmatrix} a \\ .5a \end{pmatrix} \mid a \in \mathbb{R} \right\} = \left\{ a \begin{pmatrix} 2 \\ 1 \end{pmatrix} \mid a \in \mathbb{R} \right\},$$

is, itself, a vector space. It can be called a *subspace* of the larger space \mathbb{R}^2 . As an exercise, you can confirm that any two vectors in this set, when added together, produce a third vector which is also in the set. Same with scalar multiplication. So the subspace is said to be *closed under vector addition and scalar multiplication*. In fact, that’s what we mean by a subspace.

Vector Subspace. A subspace of a (parent) vector space is a **subset** of the parent vectors that is **closed** under the vector/scalar operations.

2.5 Higher Dimensional Vector Spaces

What we just covered lays the groundwork for more exotic – yet commonly used – vector spaces in physics and engineering. That’s why I first listed the ideas and facts in the more familiar setting of \mathbb{R}^2 (and \mathbb{R}^3). If you can abstract these ideas, rather than just memorize their use in \mathbb{R}^2 and \mathbb{R}^3 , it will serve you well, even in this short quantum computing course.

Axioms

We can extend directly from ordered pairs or triples to ordered n -tuples for any positive integer n :

$$\mathbb{R}^n \equiv \left\{ \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}, \quad x_k \in \mathbb{R}, \quad k = 0 \text{ to } n-1 \right\}$$

Set $n = 10$ and you are thinking about a 10-dimensional space.

The underlying field for \mathbb{R}^n is the same real number system, \mathbb{R} , that we used for \mathbb{R}^2 . The components are named $x_0, x_1, x_2, x_3, \dots$, instead of x, y, z, \dots (although, when you deal with relativity or most engineering problems, you'll use the four-dimensional symbols x, y, z and t , the last meaning *time*).

[**Exercise.** Define *vector addition* and *scalar multiplication* for \mathbb{R}^n following the lead set by \mathbb{R}^2 and \mathbb{R}^3 .]

Inner Product

The *dot product* is defined as you would expect. If

$$\mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix},$$

then

$$\mathbf{a} \cdot \mathbf{b} \equiv \sum_{k=1}^n a_k b_k.$$

[**Exercise.** Compute the length of the vector $(1, 1, 1, \dots, 1)^t$ in \mathbb{R}^n .]

Basis

The *standard* (or *natural* or *preferred*) basis in \mathbb{R}^n is

$$\begin{aligned} \mathcal{A} &= \left\{ \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \right\} \\ &= \{ \hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_{n-1} \}. \end{aligned}$$

It is clearly orthonormal (why “clearly?”), and any other basis,

$$\mathcal{B} = \{ \mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_{n-1} \} ,$$

for \mathbb{R}^n will therefore have n vectors. The orthonormal property would be satisfied by the alternate basis \mathcal{B} if and only if

$$\mathbf{b}_k \cdot \mathbf{b}_j = \delta_{kj} .$$

We defined the last symbol, δ_{kj} , in our complex arithmetic lesson, but since many of our readers will be skipping one or more of the early chapters, I’ll reprise the definition here.

Kronecker Delta. δ_{kj} , the **Kronecker delta**, is the mathematical way to express *anything* that is to be 0 unless the index $k = j$, in which case it is 1,

$$\delta_{kj} = \begin{cases} 1, & \text{if } k = j \\ 0, & \text{otherwise} \end{cases} .$$

Expressing any vector, \mathbf{v} , in terms of a basis, say \mathcal{B} , looks like

$$\mathbf{v} = \sum_{k=1}^n \alpha_k \mathbf{b}_k ,$$

and all the remaining properties and definitions follow exactly as in the case of the smaller dimensions.

Computing Expansion Coefficients

I explained how to compute the *expansion coefficient* of an arbitrary vector along an *orthonormal basis*. This move is done so frequently in a variety of contexts in quantum mechanics and electromagnetism that it warrants being restated here in the more general cases.

When the basis is orthonormal, we can find the expansion coefficients for a vector

$$\mathbf{v} = \sum_{k=1}^n \alpha_k \mathbf{b}_k ,$$

by *dotting* \mathbf{v} with the basis vectors one-at-a-time. In practical terms, this means “we dot with \mathbf{b}_j to get α_j ,”

$$\begin{aligned} \mathbf{b}_j \cdot \mathbf{v} &= \mathbf{b}_j \cdot \sum_{k=1}^n (\alpha_k \mathbf{b}_k) = \sum_{k=1}^n \mathbf{b}_j \cdot (\alpha_k \mathbf{b}_k) \\ &= \sum_{k=1}^n \alpha_k (\mathbf{b}_j \cdot \mathbf{b}_k) = \sum_{k=1}^n \alpha_k \delta_{jk} = \alpha_j . \end{aligned}$$

2.6 More Exercises

Prove any of following that interest you:

1. If you have a *spanning set* of vectors that is *not linearly independent*, you can find a subset that is a basis. (Argue why/how.) Give an example that demonstrates you cannot arbitrarily keep the “correct number” of vectors from the original spanning set and expect those to be your basis; you have to select your subset, carefully.
2. Demonstrate that the basis \mathcal{C} , in the examples above, is orthonormal.
3. **The Gram-Schmidt Process (Tricky).** Starting with an arbitrary basis of a subspace, you can construct an *orthonormal* basis which spans the same set as your arbitrary basis. This is not the same as selecting a subset as in exercise 1, but involves creating new vectors systematically from the existing basis. **Hint:** Start with any one of the original basis vectors and normalize it. (That’s the only easy one.) From there, pick any second basis vector from the original set (distinct from the first). Those two are linearly independent (why?), therefore they form a basis for a 2-dimensional subspace. Find a vector in that subspace that has unit length and is orthogonal to your first vector (the “easy” one). You now have two orthonormal vectors. Use the same logic to produce a 3-dimensional subspace to get a third orthonormal vector. By that time, you’ll see the pattern. To find the next orthonormal vector in the new subspace that will replace the old basis vector, use what you know about dot products, perpendicular vectors and projection of vectors. Draw pictures.
4. Show that the set of vectors in \mathbb{R}^2 which have length < 1 does not constitute a vector subspace \mathbb{R}^2 .
5. Prove that the set of points on the line $y = 3x - 1$ *does* not constitute a vector subspace of \mathbb{R}^2 .
6. Argue that all one-dimensional vector subspaces of \mathbb{R}^2 consist of lines through the origin.
7. Argue that all two-dimensional subspaces of \mathbb{R}^3 consist of planes through the origin.

And that does it for our overview of vector spaces. The extension of these concepts to the more exotic sounding vector spaces head – complex vector spaces, Hilbert spaces and spaces over the finite field \mathbb{Z}_2 – will be very easy if you understand what’s in this section and have done several of the exercises.

Chapter 3

Matrices

3.1 Matrices in Quantum Computing

If a *qubit* is the replacement for the *classical bit*, what replaces the *classical logic gate*? I gave you a sneak peek at the answer in the introductory lesson. There, I first showed you the truth table of the conventional logic gate known to all computer science freshmen as the *AND gate* (symbol \wedge).

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Then, I mentioned that in the quantum world these truth tables get replaced by something more abstract, called *matrices*. Like the truth table, a *matrix* contains the rules of engagement when a *qubit* steps into its foyer. The matrix for a quantum operator that we'll study later in the quarter is

$$\frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix},$$

which represents a special gate called the *second order Hadamard operator*. We'll meet that officially in a few weeks.

Our job today is to define matrices formally and learn the specific ways in which they can be manipulated and combined with the vectors we met the previous chapter.

3.2 Definitions

Definition of a Matrix. A *matrix* is a rectangular array of numbers, variables or pretty much anything. It has **rows** and **columns**. Each matrix has a particular **size** expressed as $[\# \text{ rows}] \times [\# \text{ columns}]$, for example, 2×2 , 3×4 , 7×1 , 10×10 , etc.

A 3×4 matrix (call it A) could be:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

A 4×2 matrix (call it B) might be:

$$\begin{pmatrix} -1 & -2 \\ -3 & -4 \\ -5 & -6 \\ -7 & -8 \end{pmatrix}$$

3.2.1 Notation

For this lesson I will number starting with 1, not 0, so

- row 1 = the first row = the top row of the matrix, and
- column 1 = the first column = the left column of the matrix.

I usually name matrices with non-boldface capital letters, A , B , etc. Occasionally, I will use boldface, **A**, **B**. The context will always be clear.

Sometimes you'll see the matrix represented abstractly as the (kl) th component surrounded by parentheses,

$$(A_{kl}),$$

which represents the *entire matrix* A , whose value in row k and column l is A_{kl} . This doesn't tell you the *size*, but you'll always know that from the context.

I'll typically use $m \times n$ or $p \times q$ to describe the size of some *general matrix* and $n \times n$ or $p \times p$ when I want to specify that we are dealing with a *square matrices*, i.e., one that has same number of rows as columns.

3.3 Matrix Multiplication

Matrix multiplication will turn out to be sensitive to order. In math lingo, it is not a *commutative* operation,

$$AB \neq BA.$$

Therefore, it's important that we note the order of the product in the definition. First, we can only multiply the matrices A ($n \times p$) and B ($q \times m$) if $p = q$. So we will only define the product AB for two matrices of sizes, A ($n \times p$) and B ($p \times m$). The size of the product will be $n \times m$. Symbolically,

$$(n \times p) \cdot (p \times m) = (n \times m) .$$

Note that the “inner” dimension gets annihilated, leaving the “outer” dimensions to determine the size of the product.

3.3.1 Row \times Column

We start by defining the product of a *row* by a *column*, which is just a short way of saying a $(1 \times l)$ matrix times an $(l \times 1)$ matrix, i.e., two “1-dimensional” matrices. It is the simple *dot product* of the two entitles as if they were vectors,

$$\begin{aligned} (1, \ 2, \ 3, \ 4) \begin{pmatrix} -5 \\ 6 \\ -7 \\ 8 \end{pmatrix} &= (1)(-5) + (2)(6) + (3)(-7) + (4)(8) \\ &= 18 . \end{aligned}$$

This is the definition of matrix multiplication in the special case when the first is a column vector and the second is a row vector. But that definition is used repeatedly to generate the product of two general matrices, coming up next. Here's an example when the vectors happen to have complex numbers in them.

$$\begin{aligned} (1, \ i, \ 3, \ 2 - 3i) \begin{pmatrix} -5 \\ 6 \\ -4i \\ 8 \end{pmatrix} &= (1)(-5) + (i)(6) + (3)(-4i) + (2 - 3i)(8i) \\ &= 19 + 10i . \end{aligned}$$

[For Advanced Readers. As you see, this is just the sum of the *simple products* of the coordinates, even when the numbers are complex. For those of you already familiar with *complex inner products* (a topic we will cover next week), please note that this is *not a complex inner product*; we do not take the complex conjugate of either vector. Even if the matrices are complex numbers, we take the ordinary complex product of the corresponding elements and add them.]

3.3.2 Definition of Matrix Multiplication

The full definition of two matrices AB is more-or-less forced on us by everything we have said so far. Based on the required size of the product and the definition of two

1-dimensional matrices, we must define the (kl) th element of the answer matrix to be the dot product of the k th row of A with the l th column of B . Let's look at it graphically before we see the formal definition.

We illustrate the computation of the 1-1 element of an answer matrix in Figure 3.1 and the computation of the 2-2 element of the same answer matrix in Figure 3.2.

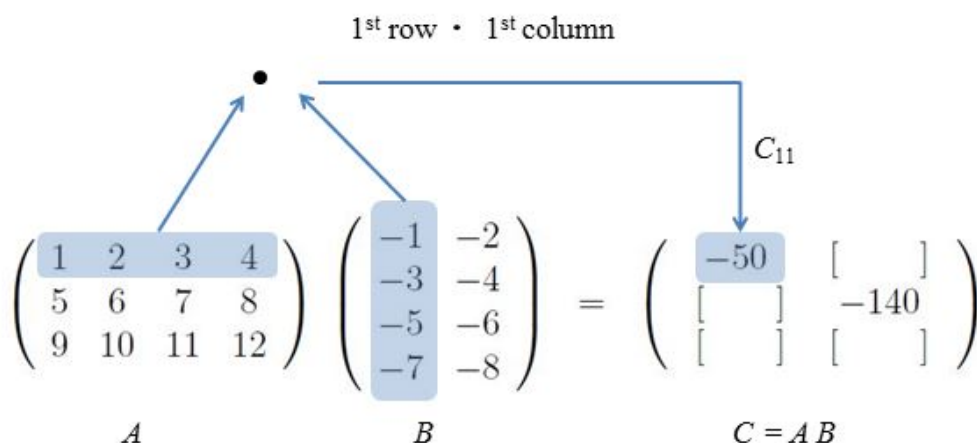


Figure 3.1: Dot-product of the first row and first column yields element 1-1

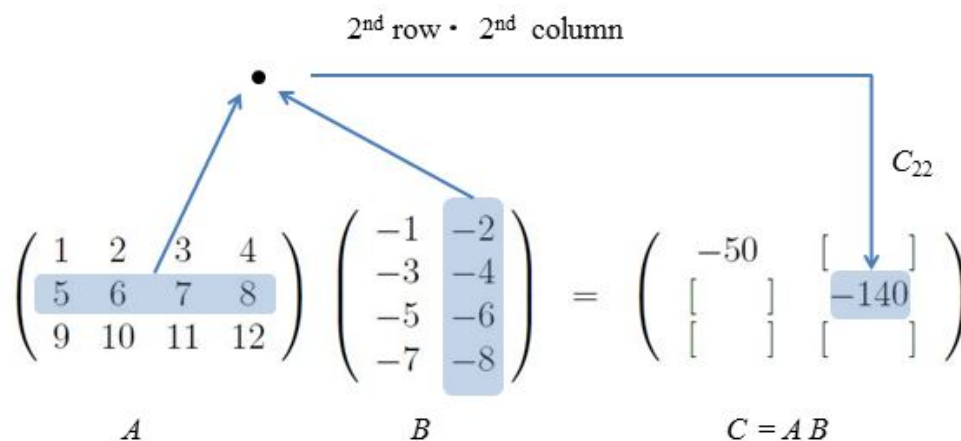


Figure 3.2: Dot-product of the second row and second column yields element 2-2

The Formal Definition of Matrix Multiplication. If A is an $n \times p$ matrix, and B is a $p \times m$ matrix, then $C = AB$ is an $n \times m$ matrix whose (kl) th element is given by

$$C_{kl} = (AB)_{kl} \equiv \sum_{j=1}^p A_{kj} B_{jl}, \quad (3.1)$$

where $k = 1, \dots, n$ and $l = 1, \dots, m$.

[**Exercise.** Fill in the rest of the elements in the product matrix, C , above.]

[**Exercise.** Compute the products

$$\begin{pmatrix} 1 & 2 & 1 \\ -2 & 0 & 4 \\ 5 & 5 & 5 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 2 & 0 & -1 \\ -2 & -1 & 1 & 4 \\ 5 & 0 & 5 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} . \quad]$$

[**Exercise.** Compute the first product above in the opposite order. Did you get the same answer?]

While matrix multiplication may not be *commutative*, it *is associative*,

$$A(BC) = (AB)C,$$

a property that is used frequently.

3.3.3 Product of a Vector by a Matrix

When a product happens to have an $n \times 1$ matrix in the second position, it is usually called *the product of a vector by a matrix*, because an $n \times 1$ matrix looks just like an n -dimensional vector. So, if \mathbf{v} is an n -dimensional vector and A is an $n \times n$ matrix, we already know what to do if we see $A\mathbf{v}$. For example,

$$\begin{pmatrix} 1 & 2i & 3 \\ 3 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 \\ 2.5i \\ -1 \end{pmatrix} = \begin{pmatrix} -4 \\ 11 + 2.5i \\ -1 \end{pmatrix} .$$

Some Observations

- **Position.** You cannot put the vector on the left of the matrix; the dimensions would no longer be compatible. You *can*, however, put the *transpose* of the vector on the left of the matrix, $\mathbf{v}^t A$. That does make sense and it has an answer (see exercise, below).
- **Linear Transformation.** Multiplying a vector by a matrix produces another vector. It is our first example of something called a *linear transformation*, which is a special kind of mapping that sends vectors to vectors. Our next lecture covers linear transformations in depth.

[**Exercise.** Using the \mathbf{v} and A from the last exercise, compute the product $\mathbf{v}^t A$.

[**Exercise.** Using the same A as above, let the vector $\mathbf{w} \equiv (-1, -2.5i, 1)^t$ and

- compute the product $A\mathbf{w}$, then
- compute $A(\mathbf{v} + \mathbf{w})$, and finally
- compare $A(\mathbf{v} + \mathbf{w})$ with $A\mathbf{v} + A\mathbf{w}$. Is this a coincidence?]

3.4 Matrix Transpose

We briefly covered the *transpose*, \mathbf{v}^t , of a *vector* \mathbf{v} . This is an action that converts column vectors to row vectors and vice versa,

$$\begin{aligned} \begin{pmatrix} 2, & i, & -\sqrt{3} \end{pmatrix}^t &= \begin{pmatrix} 2 \\ i \\ -\sqrt{3} \end{pmatrix} \\ \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}^t &= (1, 2, 3) \end{aligned}$$

That was a special case of the more general operation, namely, taking the *transpose of an entire matrix*. The transpose operation creates a new matrix whose *rows* are the *columns* of the original matrix (and whose *columns* are the *rows* of the original). More concisely, if A is the name of our original $n \times m$ matrix, its *transpose*, A^t , is the $m \times n$ matrix defined by

$$(A_{kl})^t = (A_{lk}).$$

In other words, the element in position kl of the original is moved to position lk of the transpose.

Examples:

$$\begin{aligned} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6i & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}^t &= \begin{pmatrix} 1 & 5 & 9 \\ 2 & 6i & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{pmatrix} \\ \begin{pmatrix} 1 & 2 & 0 & -i \\ -2 & -1 & 1 & 4 \\ 5 & 0 & 5 & 0 \end{pmatrix}^t &= \begin{pmatrix} 1 & -2 & 5 \\ 2 & -1 & 0 \\ 0 & 1 & 5 \\ -i & 4 & 0 \end{pmatrix} \end{aligned}$$

[**Exercise.** Make up two matrices, one square and one not square, and show the transpose of each.]

3.5 Matrix Addition and Scalar Multiplication

Matrices can be added (component-wise) and multiplied by a scalar (apply the scalar to all nm elements in the matrix). I'm going to let you be the authors of this section in two short exercises.

[**Exercise.** Make these definitions precise using a formula and give an example of each in a 3×3 case.]

[**Exercise.** Show that matrix multiplication is distributive over addition and both associative and commutative in combination with scalar multiplication, i.e.,

$$\begin{aligned} A(cB_1 + B_2) &= c(AB_1) + (AB_2) \\ &= (AB_2) + c(AB_1). \end{aligned}$$

3.6 Identity Matrix and Zero Matrix

Zero. A matrix whose elements are all 0 is called a *zero matrix* and can be written as 0 or (0) , e.g.,

$$0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{or} \quad (0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Clearly, when you add the zero matrix to another matrix it does not change anything in the other matrix. When you multiply the zero matrix by another matrix, including a vector, it will squash it to all 0s.

$$(0)A = (0) \quad \text{and} \quad (0)\mathbf{v} = \mathbf{0}$$

In words, the zero matrix is the *additive identity* for matrices.

Identity (or Unit). A square matrix whose diagonal elements (upper-left to lower-right) are all 1 and whose “off-diagonals” are all 0 is called an *identity matrix* or *unit matrix* and can be written variously as I , Id , 1 , $\mathbb{1}$ or \mathbb{I} , e.g.,

$$\mathbb{1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{or} \quad 1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The *identity matrix* has the property that when you apply it to (multiply it with)

another matrix or vector, it leaves that matrix or vector unchanged, e.g.,

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2i & 0 & -1 \\ -2 & -1 & 1 & 4 \\ 5 & 0 & 5 & 0 \\ 3-i & 2 & -2 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 2i & 0 & -1 \\ -2 & -1 & 1 & 4 \\ 5 & 0 & 5 & 0 \\ 3-i & 2 & -2 & 2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2i & 0 & -1 \\ -2 & -1 & 1 & 4 \\ 5 & 0 & 5 & 0 \\ 3-i & 2 & -2 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2i & 0 & -1 \\ -2 & -1 & 1 & 4 \\ 5 & 0 & 5 & 0 \\ 3-i & 2 & -2 & 2 \end{pmatrix}.$$

Notice that multiplication by a *unit matrix* has the same (non) effect whether it appears on either side of its co-multiplicand. The rule for both matrices and vectors is

$$\begin{aligned} \mathbb{1} M &= M \mathbb{1} = M, \\ \mathbb{1} \mathbf{v} &= \mathbf{v} \text{ and} \\ \mathbf{v}^t \mathbb{1} &= \mathbf{v}^t. \end{aligned}$$

In words, the unit matrix is the *multiplicative identity* for matrices.

3.7 Determinants

Associated with every *square* matrix is a scalar called its *determinant*. There is very little physics, math, statistics or any other science that we can do without a working knowledge of the determinant. Let's check that box now.

3.7.1 Determinant of a 2×2 Matrix

Consider any (real or complex) 2×2 matrix,

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

The *determinant* of A is defined (and written) as follows,

$$\det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \equiv ad - bc.$$

For example,

$$\begin{aligned} \begin{vmatrix} i & 1+i \\ \sqrt{6} & -5i \end{vmatrix} &\equiv (i)(-5i) - (1+i)(\sqrt{6}) \\ &= 5 - \sqrt{6} - i\sqrt{6} \\ &= (5 - \sqrt{6}) - i\sqrt{6}. \end{aligned}$$

[**Exercise.** Compute the following determinants:

$$\begin{aligned} \begin{vmatrix} 1 & -2 \\ 3 & 4 \end{vmatrix} &= ? \\ \begin{vmatrix} 4 & 1+i \\ 1-i & 2 \end{vmatrix} &= ? \end{aligned}]$$

3.7.2 Determinant of a 3×3 Matrix

We'll give an explicit definition of a 3×3 matrix and this will suggest how to proceed to the $n \times n$ case.

$$\begin{aligned} \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} &\equiv a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ &= a \left(\text{minor of } a \right) - b \left(\text{minor of } b \right) + c \left(\text{minor of } c \right) \end{aligned}$$

(Sorry, I had to use the variable name i , not for the $\sqrt{-1}$, but to mean the 3-3 element of the matrix, since I ran out of reasonable letters.) The latter defines the *minor* of a matrix element as the determinant of the smaller matrix constructed by crossing out that element's row and column (See Figure 3.3.)

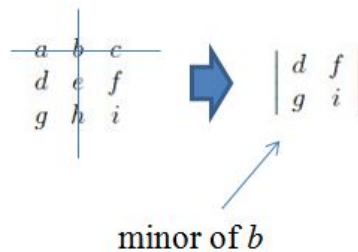


Figure 3.3: Minor of a matrix element

[**Exercise.** Compute the following determinants:

$$\begin{aligned} \begin{vmatrix} 1 & 2 & 3 \\ 2 & 1 & -2 \\ -1 & 3 & 4 \end{vmatrix} &= ? \\ \begin{vmatrix} 1 & 0 & 0 \\ 0 & 4 & 1+i \\ 0 & 1-i & 2 \end{vmatrix} &= ? \end{aligned}$$

Compare the last answer with the last 2×2 determinant exercise answer. Any thoughts?]

3.7.3 Determinant of an $n \times n$ Matrix

The 3×3 definition tells us to proceed recursively for any square matrix of size n . We define its determinant as an *alternating sum of the first row elements times their minors*,

$$\begin{aligned} \det(A) &= |A| \\ &\equiv A_{11} \left(\text{minor of } A_{11} \right) - A_{12} \left(\text{minor of } A_{12} \right) + A_{13} \left(\text{minor of } A_{13} \right) \\ &\quad + \cdots \\ &= \sum_{k=1}^n (-1)^{k+1} A_{1k} \left(\text{minor of } A_{1k} \right). \end{aligned}$$

I think you know what's coming. *Why row 1?* No reason at all (except that every square matrix has a *first* row). In the definition above we would say that *we expanded the determinant along the first row*, but we could have expanded it along any row – or any column, for that matter – and gotten the same answer.

However, there is one detail that has to be adjusted if we expand along some other row (or column). The expression

$$(-1)^{k+1}$$

has to be changed if we expand along the j th row (column) rather than the first row (column). The 1 above becomes j ,

$$(-1)^{k+j},$$

giving the formula

$$\det(A) = \sum_{k=1}^n (-1)^{k+j} A_{jk} \left(\text{minor of } A_{jk} \right),$$

good for any row (column) j .

[**Exercise.** Rewrite the definition of an $n \times n$ determinant expanded along the 3rd column; along the k th column.]

[**Exercise.** Expand one of the 2×2 determinants and one of the 3×3 determinants in the exercises above along a different row or column and confirm that you get the same answer as before.]

[**Exercise.** If you dare, prove that the determinant is independent of the choice of row/column used to expand it.]

3.7.4 Determinants of Products

$$\det(AB) = \det(A) \det(B).$$

We don't need to prove this or do exercises, but make a mental note as we need it in the following sections. (We'll easily prove it in the **next volume**, with the machinery we develop there.)

3.8 Matrix Inverses

3.8.1 Invertibility

Since we have the multiplicative identity (i.e., $\mathbb{1}$) for matrices we can ask whether, given an arbitrary square matrix A , the *inverse* of A can be found.

Definition: Invertible. An $n \times n$ matrix A , is said to be *invertible* \Leftrightarrow we can find a B such that $AB = BA = \mathbb{1}$. In that case we say the B is the *inverse* of A , and use the notation A^{-1} to describe B .

Shown in a couple different notations, just to encourage flexibility, a matrix inverse must satisfy (and is defined by)

$$\begin{aligned} M^{-1} M &= M M^{-1} = \mathbb{1} \quad \text{or} \\ A^{-1} A &= A A^{-1} = I. \end{aligned}$$

Here is one of two “little theorems” that we'll need when we introduce quantum mechanics.

Little Inverse Theorem “A”. $M\mathbf{v} = \mathbf{0}$ for some non-zero vector $\mathbf{v} \Leftrightarrow M$ is not invertible.

Proof of \Rightarrow . (By Contradiction) – We'll assume that M^{-1} exists and reach a contradiction. Let \mathbf{v} be some non-zero vector that M “sends” to $\mathbf{0}$. Then,

$$\mathbf{v} = \mathbb{1}\mathbf{v} = [M^{-1} M] \mathbf{v} = M^{-1}(M\mathbf{v}) = M^{-1}\mathbf{0} = \mathbf{0},$$

contradicting the choice of $\mathbf{v} \neq \mathbf{0}$.

Proof of \Leftarrow . (By Contradiction) – We'll assume there is **no** non-zero \mathbf{v} with $M\mathbf{v} = \mathbf{0}$ and arrive at a contradiction. (Details of the following, an [Exercise.]

- The above assumption means that $M\mathbf{v}_1 = M\mathbf{v}_2 \Rightarrow \mathbf{v}_1 = \mathbf{v}_2$.
- So, every \mathbf{w} in $\text{range}(M)$ can be assigned a *unique* \mathbf{v} in $\text{domain}(M) \mid M(\mathbf{v}) = \mathbf{w}$.
- We define a map $R : \text{range}(M) \rightarrow \text{domain}(M)$, by $R\mathbf{w} = \mathbf{v}$, where \mathbf{v} is the unique vector described in the last step.
- Show that R is a linear transformation.
- Show that $MR = RM = \mathbb{1}$, making $R = M^{-1}$, a contradiction.

QED

3.8.2 Non-Singularity and the Big Inverse Theorem

Definition: Non-Singular. An $n \times n$ matrix A , is said to be *non-singular* $\Leftrightarrow \det A \neq 0$. Conversely, if $\det A = 0$, we say that A is *singular*.

Vocabulary Variations. Sometimes you'll see *non-singularity* defined as the property of having an inverse – what we have called *invertibility*. But we're about to see that it doesn't matter which way the terms are presented, because the two notions are equivalent. If the matrix is *has an inverse* then *its determinant is non-zero*, and vice versa. This is codified in the next theorem.

Big Inverse Theorem. A matrix is *invertible* \Leftrightarrow it is *non-singular* (its determinant $\neq 0$).

While we don't have to prove this, it's actually fun and easy. In fact, we can get half way there in one line:

[**Exercise.** Prove that M *invertible* $\Rightarrow \det(M) \neq 0$ (in a single line). **Hint:** Use the fact about determinants of products.]

The other direction, \Leftarrow , is a consequence of a popular result in matrix equations. Since one or more of our quantum algorithms will refer to this result, we'll devote the next section to it and do an example. Before we leave this section, though, I need to place into evidence, our second "little theorem."

Little Inverse Theorem "B". $M\mathbf{v} = \mathbf{0}$ for some non-zero vector \mathbf{v} $\Leftrightarrow \det(M) = 0$.

Proof. *Little Theorem A* tells us that the $M\mathbf{v} = \mathbf{0} \Leftrightarrow M$ has no inverse, and the *Big Inverse Theorem* tells us that this is true $\Leftrightarrow \det(M) = 0$. QED

3.9 Matrix Equations and Cramer's Rule

3.9.1 Systems of Linear Equations

A system of *simultaneous linear equations* with n unknowns is a set of equations in variables x_1, x_2, \dots, x_n , which only involves sums *first order (linear)* terms of each variable, e.g.,

$$\begin{aligned} 4x_2 - x_5 + 3.2x_1 &= 19 + 5x_3 \\ x_1 + x_2 + x_3 + x_4 + x_5 &= 1 \\ 10x_4 + 22x_1 &= 85 - x_2 - x_1 \end{aligned}$$

To solve the system *uniquely*, it is necessary (but not sufficient) to have at least as many equations as there are unknowns. So the above system cannot have a unique

solution (although, since the equations are “consistent,” we can find some simpler relationships between the variables). Even if we *do* have exactly the same number of equations as unknowns, there still may not be a unique solution since one of the equations might – to cite one possibility – be a mere multiple of one of the others, adding no new information. Each equation has to add new information – it must be *independent* of all the others.

Matrix Equations

We can express this system of equations concisely using the language of matrix multiplication,

$$\begin{pmatrix} 3.2 & 4 & -5 & 0 & -1 \\ 1 & 1 & 1 & 1 & 1 \\ 23 & 1 & 0 & 10 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 19 \\ 1 \\ 85 \end{pmatrix}.$$

If we were able to get two more relationships between the variables, independent of these three, we would have a complete system represented by a square 5×5 matrix on the LHS. For example,

$$\begin{pmatrix} 3.2 & 4 & -5 & 0 & -1 \\ 1 & 1 & 1 & 1 & 1 \\ 23 & 1 & 0 & 10 & 0 \\ 2.5 & \pi & .09 & 50 & 1 \\ 2\pi & 0 & .83 & -1 & -17 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 19 \\ 1 \\ 85 \\ 0 \\ 4 \end{pmatrix}.$$

Setting M = the 5×5 matrix on the left, \mathbf{v} = the vector of unknowns, and \mathbf{c} = the vector of constants on the right, this becomes

$$M \mathbf{v} = \mathbf{c}.$$

How can we leverage the language of matrices to get a solution? We want to know what all the x_k are. That’s the same as having a vector equation in which \mathbf{v} is all alone on the LHS,

$$\mathbf{v} = \mathbf{d},$$

and the \mathbf{d} on the RHS is a vector of *constants*. The scent of a solution should be wafting in the breeze. If $M\mathbf{v} = \mathbf{c}$ were a scalar equation we would divide by M . Since it’s a matrix equation there are two differences:

1. Instead of dividing by M , we multiply each side of the equation (on the left) by M^{-1} .
2. M^{-1} may not even exist, so this only works if M is *invertible*.

If M is invertible, and we can calculate M^{-1} , we apply bullet 1 above,

$$\begin{aligned} M^{-1} M \mathbf{v} &= M^{-1} \mathbf{c} \\ \mathbb{1} \mathbf{v} &= M^{-1} \mathbf{c} \\ \mathbf{v} &= M^{-1} \mathbf{c} \end{aligned}$$

This leaves us with two action items.

1. Determine whether or not M is invertible.
2. If it is, compute the inverse, M^{-1} .

We know how to do item 1; if $\det(M)$ is non-zero, it can be inverted. For item 2, we have to learn how to solve a system of linear equations, something we are perfectly situated to do given the work we have already done in this lecture.

3.9.2 Cramer's Rule

Today's technique, called *Cramer's rule*, is a clean and easy way to invert a matrix, but not used much in practice due to its proclivity toward round-off error and poor computer performance. (We'll learn alternatives based on so-called *normal forms* and *Gaussian elimination* when we get into Simon's and Shor's quantum algorithms). But Cramer's rule is very cute – not to mention a key to our **big inverse theorem**, whose proof is still in need of a missing a part – so let's get enlightened.

Cramer's Rule. A system of n linear equations in n unknowns,

$$M \mathbf{v} = \mathbf{c},$$

(like the 5×5 system above) can be solved uniquely for the unknowns $x_k \Leftrightarrow \det(M) \neq 0$. In that case each x_k is given by

$$x_k = \frac{\det M_k}{\det M},$$

where M_k is the matrix M with its k th column replaced by the *constant vector* \mathbf{c} (see Figure 3.4).

We will not prove Cramer's rule in this volume. It's proof is useful in the sequel, so we'll give a proof there. For now, you can take it on faith, look up a proof, or refer to **Volume 2** (“in production” at the time of this writing).

$$M_k = \det \begin{pmatrix} M_{11} & \dots & \overset{k^{\text{th}} \text{ column}}{\downarrow} c_1 & \dots & M_{1n} \\ M_{21} & \dots & c_2 & \dots & M_{2n} \\ \vdots & \dots & \vdots & \dots & \vdots \\ \vdots & \dots & \vdots & \dots & \vdots \\ \vdots & \dots & \vdots & \dots & \vdots \\ M_{n1} & \dots & c_n & \dots & M_{nn} \end{pmatrix}$$

Figure 3.4: The numerator of Cramer's fraction

Example Application of Cramer's Rule

We can compute the relevant determinants of the 5×5 system above with the help of Mathematica or similar software which computes determinants for us. Or, we can write our own determinant calculator using very few instructions and recursion, and do it all ourselves. First, we check the main determinant,

$$\det(M) = -167077 \neq 0. \quad \checkmark$$

So M is non-singular \Rightarrow system is solvable. Now for Cramer's numerators (to six significant digits):

$$\begin{aligned} \det(M_1) &= -635709. \\ \det(M_2) &= 199789. \\ \det(M_3) &= 423127. \\ \det(M_4) &= 21996.3 \\ \det(M_5) &= -176281. \end{aligned}$$

Using this, we solve for the first unknown,

$$x_1 = \frac{-635709.}{-167077.} = 3.80489.$$

[**Exercise.** Compute the other four x_k and confirm that any one of the equations in the original system holds for these five values.]

Computing Matrix Inverses Using Cramer's Rule

As noted, Cramer's rule is not very useful for writing software, but we *can* apply it to the problem of finding a matrix inverse, especially when dealing with small, 2×2 , matrices by hand. Say we are given such a matrix

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

that we have confirmed to be non-singular by computing $ad - bc$. So we know it has an inverse, which we will temporarily write as

$$M^{-1} = \begin{pmatrix} e & f \\ g & h \end{pmatrix}.$$

The only thing we can say with certainty, at this point, is that

$$M M^{-1} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Our goal is to solve this matrix equation. We do so in two parts. First, we break this into an equation which only involves the *first column* of the purported inverse,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e \\ g \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

This is exactly the kind of 2-equation *linear system* we have already conquered. Cramer's rule tells us that it has a solution (since $\det(M) \neq 0$) and the solution is given by

$$e = \frac{\det \begin{pmatrix} 1 & b \\ 0 & d \end{pmatrix}}{\det M}$$

and

$$g = \frac{\det \begin{pmatrix} a & 1 \\ c & 0 \end{pmatrix}}{\det M}.$$

The same moves can be used on the *second column* of M^{-1} , to solve

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} f \\ h \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

[**Exercise.** Write down the corresponding quotients that give f and h .]

Example. We determine the invertibility of M and, if invertible, compute its inverse, where

$$M = \begin{pmatrix} -12 & 1 \\ 15 & 1 \end{pmatrix}.$$

The determinant is

$$\begin{vmatrix} -12 & 1 \\ 15 & 1 \end{vmatrix} = -12 - 15 = -27 \neq 0,$$

which tells us that the inverse does exist. Setting

$$M^{-1} = \begin{pmatrix} e & f \\ g & h \end{pmatrix},$$

we solve first for the left column $\begin{pmatrix} e \\ g \end{pmatrix}$. Following the procedure above,

$$e = \frac{\det \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}{\det M} = \frac{1}{-27} = -\frac{1}{27}$$

and

$$g = \frac{\det \begin{pmatrix} -12 & 1 \\ 15 & 0 \end{pmatrix}}{\det M} = \frac{-15}{-27},$$

which we don't simplify ... yet. Continuing on to solve for f and h results in the final inverse matrix

$$M^{-1} = \begin{pmatrix} -1/27 & 1/27 \\ 15/27 & 12/27 \end{pmatrix} = \frac{1}{27} \begin{pmatrix} -1 & 1 \\ 15 & 12 \end{pmatrix},$$

and we can see why we did not simplify the expression of g .

[**Exercise.** For the above example, confirm that $M M^{-1} = \text{Id.}$]

[**Exercise.** Compute the inverse of

$$M = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

using Cramer's rule. Check your work.]

[**Exercise.** Show that

$$M = \begin{pmatrix} 1 & 0 & 2 \\ 3 & 4 & 0 \\ -2 & 0 & 0 \end{pmatrix}$$

is non-singular, then compute its inverse using Cramer's rule. Check your work.]

3.9.3 Completion of the Proof of the Big Inverse Theorem

Remember this

Big Inverse Theorem. A matrix is *invertible* \Leftrightarrow it is *non-singular* (its determinant $\neq 0$).

?

Your proved \Rightarrow in an exercise. Using Cramer's rule, we can get the \Leftarrow :

Lemma: Non-Singularity Implies Invertibility.

$$\det(M) \neq 0 \Rightarrow M \text{ is invertible.}$$

Proof. The hypothesis of this Lemma, *non-singularity*, means that Cramer's Hypothesis for the system of n equations,

$$M\mathbf{v} = \mathbf{c},$$

is met, and we can invoke **Cramer** to solve for the unknowns $\mathbf{v} = (x_1, \dots, x_n)^t$. Let's choose to solve the system when \mathbf{c} is equal to the k^{th} *preferred* basis vector,

$$\hat{\mathbf{e}}_k = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow k^{\text{th}} \text{ position}.$$

(**Note.** I called the *preferred* basis $\{\hat{\mathbf{x}}_0, \dots, \hat{\mathbf{x}}_{n-1}\}$ in the past but am now introducing another common designation, $\{\hat{\mathbf{e}}_1, \dots, \hat{\mathbf{e}}_n\}$.) So, solving the system,

$$M\mathbf{v} = \hat{\mathbf{e}}_k$$

gives us a solution vector, $\mathbf{a}_k = (a_{k1}, a_{k2}, \dots, a_{kn})^t$, that satisfies

$$M\mathbf{a}_k = \hat{\mathbf{e}}_k.$$

We do this n times, each time for a different system, $M\mathbf{v} = \mathbf{c}$, using the same M , but where \mathbf{c} is a distinct *preferred* basis vector among the $\{\hat{\mathbf{e}}_k, k = 1, \dots, n\}$. Create a matrix, R , whose n columns consist of the solution vectors $\{\mathbf{a}_k\}$, and the above equality becomes

$$(M\mathbf{a}_1, \dots, M\mathbf{a}_k, \dots, M\mathbf{a}_n) = (\hat{\mathbf{e}}_1, \dots, \hat{\mathbf{e}}_k, \dots, \hat{\mathbf{e}}_n),$$

which is nothing other than

$$M(\mathbf{a}_1, \dots, \mathbf{a}_k, \dots, \mathbf{a}_n) = \mathbb{1},$$

and we have found a matrix,

$$A = (\mathbf{a}_1, \dots, \mathbf{a}_k, \dots, \mathbf{a}_n),$$

that satisfies

$$MA = \mathbb{1}.$$

We have shown that M has a “right” inverse, A , but matrix multiplication is not *commutative*, so we have not technically met the requirement for a matrix inverse (it must work on *either* side of the product). While it turns out that a *one-sided* inverse is a *two-sided* inverse for $n \times n$ matrices (which we haven't proven), we actually don't

need that result; we can *easily* show that the same A works as a “left” inverse, i.e., that $AM = \mathbb{1}$, using nothing external to this chapter. Watch.

First, we note that $\det A \neq 0$ (by applying the **product of determinant theorem** to $MA = \mathbb{1}$), so A satisfies the *non-singularity* condition needed by **Cramer’s rule**, allowing us, by the above argument, to give A its own right inverse, call it M' ,

$$AM' = \mathbb{1}.$$

Next, left-multiply both sides by M and simplify

$$\begin{aligned} M(AM') &= M\mathbb{1}, \\ (MA)M' &= M, \\ (\mathbb{1})M' &= M, \\ M' &= M, \end{aligned}$$

so, A ’s right inverse is the original M , and we have shown that

$$AM = MA = \mathbb{1},$$

i.e., M and A are *inverses* and, more to the point, M is *invertible*. QED

Chapter 4

Hilbert Space

4.1 Complex Vector Spaces for Quantum Computing

4.1.1 The Vector Nature of a Qubit

We are gradually constructing a mathematical language that can be used to accurately model the physical qubit and hardware that processes it. A typical qubit is expressed using the symbolism

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle.$$

In the sneak peek given in the introduction I leaked the meaning of the two numbers α and β . They embody the probabilistic nature of quantum bits. While someone might have prior knowledge that a qubit with the precise value “ $\alpha|0\rangle + \beta|1\rangle$ ” (whatever that means) was sitting in a memory location, if we tried to read the value of that location – that is, if we *measured* it – we would see evidence of neither α nor β , but instead observe only a classical bit; our measurement device would register one of two possible outcomes: “0” or “1.” Yet α and β do play a role here. They tell us our measurement would be

$$\begin{aligned} \text{“0”} & \quad \text{with probability} \quad |\alpha|^2 \quad \text{and} \\ \text{“1”} & \quad \text{with probability} \quad |\beta|^2. \end{aligned}$$

Obviously, there’s a lot yet to understand about how this unpredictable behavior can be put to good use, but for today we move a step closer by adding the following clue:

*The qubit shown above is a **vector** having unit length, and the α, β are its **coordinates** in what we shall see is the vector space’s natural basis $\{|0\rangle, |1\rangle\}$.*

In that sense, a qubit seems to correspond, if not physically at least mathematically, to a vector in some vector space where α and β are two scalars of the system.

4.1.2 The Complex Nature of a Qubit

We need to know whatever is *possible* to know about a hypothetical quantum circuit using paper and pencil so we can predict what the hardware will do before we build it. I'm *happy* to report that the mathematical language that works in this situation is one of the most accurate and well tested in the history of science: *quantum mechanics*. It *behooves me* to report that a *real* vector space like \mathbb{R}^2 or \mathbb{R}^3 doesn't work. To make accurate predictions about qubits and the logic circuitry that entangles them, we'll need to allow α and β to be *complex*. Thus, our vector space must be defined over a *complex* scalar field.

Besides the straightforward consequences resulting from the use of complex arithmetic, we'll have to define an *inner product* that differs from the simple *dot product* of real vector spaces. At that point we'll be working in what mathematicians call *complex Hilbert space* or simply *Hilbert space*. To make the correspondence between our math and the real world complete, we'll have to add one last tweak: we will consider only vectors of *unit length*. That is, we will need to work on something called *the projective sphere* a.k.a *projective Hilbert space*.

Today we learn how to manipulate vectors in a complex *Hilbert space* and the *projective sphere* within it.

4.2 The Complex Vector Space, \mathbb{C}^n

Since we are secure enough in the vocabulary of real vector spaces, there's no need to dilly-dally around with 2- or 3-dimensional spaces. We can go directly to the fully general n -dimensional complex space, which we call \mathbb{C}^n .

Its *scalars* are the complex numbers, \mathbb{C} , and its *vectors* are n -tuples of complex numbers,

$$\mathbb{C}^n \equiv \left\{ \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{pmatrix}, \quad c_k \in \mathbb{C}, \quad k = 0 \text{ to } n-1 \right\}$$

Except for the inner product, everything else works like the real space \mathbb{R}^n with the plot twist that the components are complex. In fact, the natural basis is actually identical to \mathbb{R}^n 's basis.

[**Exercise.** Prove that the same natural basis works for both \mathbb{R}^n and \mathbb{C}^n .]

There are, however, other bases for \mathbb{C}^n which have no counterpart in \mathbb{R}^n , since their components can be complex.

[**Exercise.** Drum up a basis for \mathbb{C}^n that has no real counterpart. Then find a vector in \mathbb{C}^n which is not in \mathbb{R}^n but whose coordinates relative to this basis are all real.]

[**Exercise.** Is $\mathbb{R}^n \subseteq \mathbb{C}^n$ as a set? As a subspace? Justify your answers.]

4.3 The Complex Inner Product

So, what's wrong with \mathbb{R}^n 's dot product in the complex case? If we were to define it as we do in the real case, for example as in \mathbb{R}^2 ,

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = x_1 x_2 + y_1 y_2,$$

we'd have a problem with lengths of vectors which we want to be ≥ 0 . Recall that lengths are defined by dotting a vector with itself (I'll remind you about the exact details in a moment). To wit, for a complex \mathbf{a} ,

$$\mathbf{a} \cdot \mathbf{a} \stackrel{?}{=} \sum_{k=0}^{n-1} a_k^2$$

is not necessarily *real*, never mind non-negative (try a vector whose components are all $1 + i$). When it *is* real, it could still be negative:

$$\begin{pmatrix} 5i \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 5i \\ 3 \end{pmatrix} = -25 + 9 = -16.$$

We don't want to have complex, imaginary or negative lengths typically, so we need a different definition of dot product for complex vector spaces. We define the product

$$\mathbf{a} \cdot \mathbf{b} \equiv \sum_{k=0}^{n-1} \bar{a}_k b_k = \sum_{k=0}^{n-1} (a_k^*) b_k.$$

When defined in this way, some authors prefer the term *inner product*, reserving *dot product* for the real vector space analog (although some authors say *complex dot product*, so you have to adapt.)

Notation. An alternative notation for the (complex) inner product is sometimes used,

$$\langle \mathbf{a}, \mathbf{b} \rangle,$$

and in quantum mechanics, you'll always see

$$\langle \mathbf{a} | \mathbf{b} \rangle.$$

Caution #1. The complex inner product is not *commutative*, i.e.,

$$\langle \mathbf{a} | \mathbf{b} \rangle \neq \langle \mathbf{b} | \mathbf{a} \rangle.$$

There is an important relationship between the two, however, namely

$$\langle \mathbf{a} | \mathbf{b} \rangle^* = \langle \mathbf{b} | \mathbf{a} \rangle .$$

[**Exercise.** Show that the definition of complex inner product implies the above result.]

Caution #2. The complex inner product can be defined by conjugating the b_k s, rather than the a_k s, and this would produce a different result, one which is the complex conjugate of *our* defined inner product. However, physicists – and we – conjugate the left-hand vector's coordinates because it produces nicer looking formulas.

Example. Let

$$\mathbf{a} = \begin{pmatrix} 1+i \\ 3 \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} 1-2i \\ 5i \end{pmatrix} .$$

Then

$$\begin{aligned} \langle \mathbf{a} | \mathbf{b} \rangle &= (1+i)^* (1-2i) + (3^*) 5i \\ &= (1-i) (1-2i) + (3) 5i \\ &= (1-i-2i-2) + 15i \\ &= -1 + 12i . \end{aligned}$$

[**Exercise.** Compute $\langle \mathbf{b} | \mathbf{a} \rangle$ and verify that it is the complex conjugate of $\langle \mathbf{a} | \mathbf{b} \rangle$.]

More Properties of the Complex Inner Product

In our linear algebra lesson we learned that inner products are *distributive*. In the second position, this means

$$\langle \mathbf{a} | \mathbf{b} + \mathbf{b}' \rangle = \langle \mathbf{a} | \mathbf{b} \rangle + \langle \mathbf{a} | \mathbf{b}' \rangle ,$$

and the same is true in the first position. For the record, we should collect two more properties that apply to the all-important complex inner product.

- It is *linear* in the *second* position. If c is a complex scalar,

$$c \langle \mathbf{a} | \mathbf{b} \rangle = \langle \mathbf{a} | c\mathbf{b} \rangle .$$

- It is *anti-linear* in the *first* position. If c is a complex scalar,

$$c \langle \mathbf{a} | \mathbf{b} \rangle = \langle c^* \mathbf{a} | \mathbf{b} \rangle .$$

Again, this asymmetry is reversed in much of mathematics and engineering. Quantum mechanics naturally leads to anti-linearity in the *left* vector position, while you will normally see it in the right vector position in other fields.

4.3.1 Norm and Distance

The *inner product* on \mathbb{C}^n confers it with a *metric*, that is, a way to measure things. There are two important concepts that emerge:

1. **Norm.** The *norm* of vector, now repurposed to our current complex vector space, is defined by

$$\|\mathbf{a}\| \equiv \sqrt{\sum_{k=0}^{n-1} |a_k|^2} = \sqrt{\sum_{k=0}^{n-1} (a_k)^* a_k} .$$

2. **Distance.** The *distance* between vectors \mathbf{a} and \mathbf{b} is

$$\text{dist}(\mathbf{a}, \mathbf{b}) \equiv \|\mathbf{b} - \mathbf{a}\| = \sqrt{\sum_{k=0}^{n-1} |b_k - a_k|^2} .$$

With the now correct definition of inner product we get the desired behavior when we compute a norm,

$$\begin{aligned} \|\mathbf{a}\|^2 &\equiv \langle \mathbf{a} | \mathbf{a} \rangle = \sum_{k=0}^{n-1} (a_k)^* a_k = \sum_{k=0}^{n-1} |a_k|^2 \\ &\geq 0 . \end{aligned}$$

Since the length (*norm* or *modulus*) of \mathbf{a} , $\|\mathbf{a}\|$, is the non-negative square root of this value, once again all is well: lengths are real and non-negative.

More Notation. You may see the modulus for a vector written in *normal* (not **bold**) face, as in

$$a = \|\mathbf{a}\| .$$

Another common syntax uses single, rather than double, bars,

$$|\mathbf{a}| = \|\mathbf{a}\| .$$

It's all acceptable.

Example (Continued). The norm (a.k.a. modulus) of \mathbf{a} in the earlier example is

$$\begin{aligned} a &= \|\mathbf{a}\| \equiv \sqrt{\langle \mathbf{a} | \mathbf{a} \rangle} = \sqrt{(1+i)^* (1+i) + (3^*) 3} \\ &= \sqrt{(1-i)(1+i) + (3) 3} \\ &= \sqrt{(1 - i + i + 1) + 9} \\ &= \sqrt{11} . \end{aligned}$$

[**Exercise.** Compute the norm of \mathbf{b} from that same example, above.]

Complex Orthonormality and Linear Independence

We had a little theorem about orthonormal sets in real vector spaces. It's just as true for complex vector spaces with the complex inner product.

Theorem. *If a set of vectors $\{\mathbf{a}_k\}$ is orthonormal, it is necessarily linearly independent.*

Proof. We'll assume the theorem is false and arrive at a contradiction. Say $\{\mathbf{a}_k\}$ is an orthonormal collection, yet one of them, \mathbf{a}_0 , is a linear combination of the others, i.e.,

$$\mathbf{a}_0 = \sum_{k=1}^{n-1} c_k \mathbf{a}_k.$$

By orthonormality $\langle \mathbf{a}_0 | \mathbf{a}_k \rangle = 0$ for all $k \neq 0$, and $\langle \mathbf{a}_0 | \mathbf{a}_0 \rangle = 1$, so

$$1 = \langle \mathbf{a}_0 | \mathbf{a}_0 \rangle = \left\langle \mathbf{a}_0 \left| \left(\sum_{k=1}^{n-1} c_k \mathbf{a}_k \right) \right. \right\rangle = \sum_{k=1}^{n-1} c_k \langle \mathbf{a}_0 | \mathbf{a}_k \rangle = 0,$$

a contradiction. QED

4.3.2 Expansion Coefficients

The dot-product trick for computing *expansion coefficients* along an orthonormal basis works in this regime, but we have to be careful due to this slight asymmetry. If we want to expand \mathbf{v} along an orthonormal $\mathcal{B} = \{\mathbf{b}_k\}$, we still “dot it” with the individual basis vectors. Say the (as yet unknown) coefficients of \mathbf{v} in this basis are (\dots, β_k, \dots) . We compute each one using

$$\begin{aligned} \langle \mathbf{b}_k | \mathbf{v} \rangle &= \left\langle \mathbf{b}_k \left| \sum_{j=0}^{n-1} \beta_j \mathbf{b}_j \right. \right\rangle \\ &= \sum_{j=0}^{n-1} \beta_j \langle \mathbf{b}_k | \mathbf{b}_j \rangle, \end{aligned}$$

and since orthonormality means

$$\langle \mathbf{b}_k | \mathbf{b}_j \rangle = \delta_{kj},$$

the last sum collapses to the desired β_k . However, we had to be careful to place our \mathbf{v} on the right side of the inner product. Otherwise, we would not get the k th expansion coefficient, but its —.

[**Exercise.** Fill in the blank].

Example. In \mathbb{C}^2 we expand $\mathbf{v} = \begin{pmatrix} 1+i \\ 1-i \end{pmatrix}$ along the natural basis \mathcal{A} ,

$$\mathcal{A} = \{\hat{\mathbf{e}}_0, \hat{\mathbf{e}}_1\} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}.$$

(We do this easy example because the answer is obvious: the coordinates along \mathcal{A} should match the pure vector components, otherwise we have a problem with our technique. Let's see ...)

We seek

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix}_{\mathcal{A}}.$$

The “dotting trick” says

$$\begin{aligned} v_0 &= \langle \hat{\mathbf{e}}_0 | \mathbf{v} \rangle = (e_{00})^* v_0 + (e_{01})^* v_1 \\ &= 1(1+i) + 0(1-i) = 1+i, \end{aligned}$$

and

$$\begin{aligned} v_1 &= \langle \hat{\mathbf{e}}_1 | \mathbf{v} \rangle = (e_{10})^* v_0 + (e_{11})^* v_1 \\ &= 0(1+i) + 1(1-i) = 1-i, \end{aligned}$$

so

$$\begin{pmatrix} 1+i \\ 1-i \end{pmatrix} = \begin{pmatrix} 1+i \\ 1-i \end{pmatrix}_{\mathcal{A}},$$

as expected (✓). Of course that wasn't much fun since natural basis vector components were real (0 and 1), thus there was nothing to conjugate. Let's do one with a little crunch.

Example. In \mathbb{C}^2 we expand the same $\mathbf{v} = \begin{pmatrix} 1+i \\ 1-i \end{pmatrix}$ along the basis \mathcal{B} ,

$$\mathcal{B} \equiv \{ \hat{\mathbf{b}}_0, \hat{\mathbf{b}}_1 \} = \left\{ \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}, \begin{pmatrix} i\sqrt{2}/2 \\ -i\sqrt{2}/2 \end{pmatrix} \right\}.$$

First, we confirm that this basis is *orthonormal* (because the dot-product trick only works for orthonormal bases).

$$\begin{aligned} \langle \hat{\mathbf{b}}_0 | \hat{\mathbf{b}}_1 \rangle &= (b_{00})^* b_{10} + (b_{01})^* b_{11} \\ &= (\sqrt{2}/2)(i\sqrt{2}/2) + (\sqrt{2}/2)(-i\sqrt{2}/2) \\ &= i/2 - i/2 = 0. \checkmark \end{aligned}$$

Also,

$$\begin{aligned} \langle \hat{\mathbf{b}}_0 | \hat{\mathbf{b}}_0 \rangle &= (b_{00})^* b_{00} + (b_{01})^* b_{01} \\ &= (\sqrt{2}/2)(\sqrt{2}/2) + (\sqrt{2}/2)(\sqrt{2}/2) \\ &= 1/2 + 1/2 = 1 \checkmark, \end{aligned}$$

and

$$\begin{aligned}
 \langle \hat{\mathbf{b}}_1 | \hat{\mathbf{b}}_1 \rangle &= (b_{10})^* b_{10} + (b_{11})^* b_{11} \\
 &= (-i\sqrt{2}/2)(i\sqrt{2}/2) + (i\sqrt{2}/2)(-i\sqrt{2}/2) \\
 &= 1/2 + 1/2 = 1 \quad \checkmark,
 \end{aligned}$$

which establishes orthonormality.

We seek

$$\begin{pmatrix} v_0 \\ v_1 \end{pmatrix}_{\mathcal{B}}.$$

The “dotting trick” says

$$\begin{aligned}
 v_0 &= \langle \hat{\mathbf{b}}_0 | \mathbf{v} \rangle = (b_{00})^* v_0 + (b_{01})^* v_1 \\
 &= (\sqrt{2}/2)(1+i) + (\sqrt{2}/2)(1-i) \\
 &= \sqrt{2},
 \end{aligned}$$

and

$$\begin{aligned}
 v_1 &= \langle \hat{\mathbf{b}}_1 | \mathbf{v} \rangle = (b_{10})^* v_0 + (b_{11})^* v_1 \\
 &= (-i\sqrt{2}/2)(1+i) + (i\sqrt{2}/2)(1-i) \\
 &= \sqrt{2},
 \end{aligned}$$

so

$$\begin{pmatrix} 1+i \\ 1-i \end{pmatrix} = \begin{pmatrix} \sqrt{2} \\ \sqrt{2} \end{pmatrix}_{\mathcal{B}}.$$

Finally, we check our work.

$$\begin{aligned}
 \sqrt{2} \mathbf{b}_0 + \sqrt{2} \mathbf{b}_1 &= \sqrt{2} \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} + \sqrt{2} \begin{pmatrix} i\sqrt{2}/2 \\ -i\sqrt{2}/2 \end{pmatrix} \\
 &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} i \\ -i \end{pmatrix} \\
 &= \begin{pmatrix} 1+i \\ 1-i \end{pmatrix} \quad \checkmark
 \end{aligned}$$

[**Exercise.** Work in \mathbb{C}^2 and use the same \mathbf{v} as above to get its coordinates relative to the basis \mathcal{C} ,

$$\mathcal{C} \equiv \{\hat{\mathbf{c}}_0, \hat{\mathbf{c}}_1\} = \left\{ \frac{1}{\sqrt{3}} \begin{pmatrix} 1+i \\ 1 \end{pmatrix}, \frac{1}{\sqrt{15}} \begin{pmatrix} 2+i \\ -3+i \end{pmatrix} \right\}.$$

Before starting, demonstrate that this basis is *orthonormal*.]

4.4 Hilbert Space

4.4.1 Definitions

A *Hilbert Space* is a *real* or *complex vector space* that

1. has an *inner product* and
2. is *complete*.

You already know what an *inner product* is. A vector space that has an inner product is usually called an *inner-product space*.

Completeness is different from the property that goes by the same name (unfortunately) involving vectors *spanning* a space. This is not *that*. Rather, *completeness* here means (please grab something stable and try not to hit your head) that *any Cauchy sequence of vectors in the space converges to some vector also in the space*. Unless you remember your advanced calculus, *completeness* won't mean much, and it's not a criterion that we will ever use, explicitly. However, if you want one more degree of explanation, read the next paragraph.

Completeness of an Inner Product (Optional Reading)

The inner product, as we saw, imbues the vector space with a *distance function*, $\text{dist}(\cdot, \cdot)$. Once we have a distance function, we can inspect any infinite sequence of vectors, like $\{\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \dots\}$, and test whether the distance between consecutive vectors in that sequence, $\text{dist}(\mathbf{a}_{k+1}, \mathbf{a}_k)$, get small fast (i.e., obeys the *Cauchy criterion* – please research the meaning of this phrase elsewhere, if interested). If it does, $\{\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \dots\}$ is called a *Cauchy sequence*. The vector space is *complete* if every Cauchy sequence of vectors in the space approaches a unique vector, *also* in the space, called the *limit of the sequence*.

Illustration of Completeness. The *completeness criterion* does not require a vector space in order that it be satisfied. It requires only a set and a *norm* (or *metric*) which allows us to measure distances. So we can ask about completeness of many sets that are not even vector spaces. The simplest example happens to *not* be a vector space.

Consider the interval $[0, 1]$ in \mathbb{R} which includes both endpoints. This set is complete with respect to the usual norm in \mathbb{R} . For example, the sequence $\{1 - \frac{1}{k}\}_{k=2}^{\infty}$ is Cauchy and converges to 1, which is, indeed, in $[0, 1]$. This does not establish that $[0, 1]$ is complete, but it indicates the kind of challenging sequence that might prove a set is not Cauchy. (See Figure 4.1)

For a counter-example, consider the interval $(0, 1)$ in \mathbb{R} which does not include either endpoint. The same sequence, just discussed, is still in the set, but its limit, 1, is not. So this set is not complete. (See Figure 4.2)

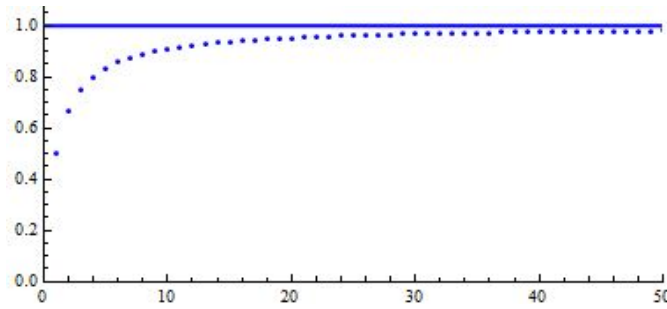


Figure 4.1: The Cauchy sequence $\{1 - \frac{1}{k}\}_{k=2}^{\infty}$ has its limit in $[0, 1]$

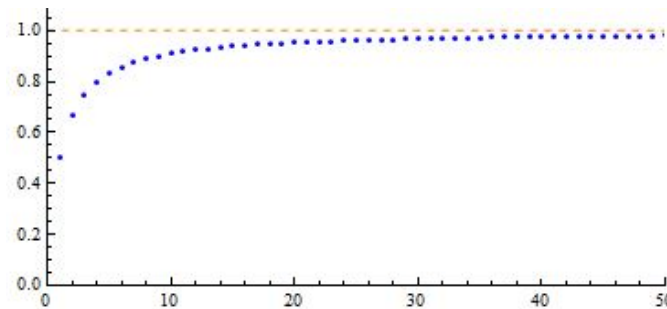


Figure 4.2: The Cauchy sequence $\{1 - \frac{1}{k}\}_{k=2}^{\infty}$ does not have its limit in $(0, 1)$

Notation

When I want to emphasize that we're working in a Hilbert space, I'll use the letter \mathcal{H} just as I use terms like \mathbb{R}^3 or \mathbb{C}^n to denote real or complex vector spaces. \mathcal{H} could take the form of a \mathbb{C}^2 or a \mathbb{C}^n , and I normally won't specify the dimension of \mathcal{H} until we get into tensor products.

4.4.2 Old Friends and New Acquaintances

Finite Dimensional Hilbert Spaces

Before we go further, let's give recognition to an old friend which happens to be a Hilbert space: \mathbb{R}^n with the usual dot-product. This Euclidean space (actually "these Euclidean spaces," since there is one for each positive n) does(do) meet the completeness criterion, so it(they) are Hilbert space(s).

As we have been discussing on this page so, too, are the complex inner-product spaces, \mathbb{C}^n , for each positive n .

For us, the most important Hilbert space will be \mathbb{C}^2 .

Thus, we already know two classes of inner-product spaces that are "Hilbert".

Infinite Dimensional Hilbert Spaces

When physicists use the term *Hilbert space*, they often mean something slightly more exotic: *function spaces*. These are vector spaces whose vectors consist of sufficiently well-behaved *functions*, and whose inner product is usually defined in terms of an *integral*.

The Space $L^2[a, b]$. For example, all complex-valued functions, $f(x)$, defined over the real interval $[a, b]$ and which are square-integrable, i.e.,

$$\int_a^b |f(x)|^2 dx < \infty,$$

form a vector space. We can define an inner product for any two such functions, f and g , using

$$\langle f | g \rangle \equiv \int_a^b f(x)^* g(x) dx,$$

and this inner-product will give a distance and norm that satisfy the completeness criterion. Hilbert spaces very much like these are used to model the momentum and position of sub-atomic particles.

4.4.3 Some Useful Properties of Hilbert Spaces

We will be making implicit use of the following consequences of real and complex inner-product spaces as defined above, so it's good to take a moment and meditate on each one.

Triangle Inequality

Both the real dot product of \mathbb{R}^n and the complex inner product of \mathbb{C}^n satisfy the triangle inequality condition: For any vectors, \mathbf{x} , \mathbf{y} and \mathbf{z} ,

$$\text{dist}(x, z) \leq \text{dist}(x, y) + \text{dist}(y, z).$$

Pictured in \mathbb{R}^2 , we can see why this is called the *triangle inequality*. (See Figure 4.3).

[**Exercise.** Pick three vectors in \mathbb{C}^3 and verify that the triangle inequality is satisfied. Do this at least twice, once when the three vectors do not all lie on the same complex line, $\{\alpha \mathbf{x} \mid \alpha \in \mathbb{C}\}$ and once when all three *do* lie on the same line and \mathbf{y} is “between” \mathbf{x} and \mathbf{z} .]

Cauchy-Schwarz Inequality

A more fundamental property of inner-product spaces is the *Cauchy-Schwarz inequality* which says that any two vectors, \mathbf{x} and \mathbf{y} , of an inner-product space satisfy

$$|\langle \mathbf{x} | \mathbf{y} \rangle|^2 \leq \|\mathbf{x}\|^2 \|\mathbf{y}\|^2.$$

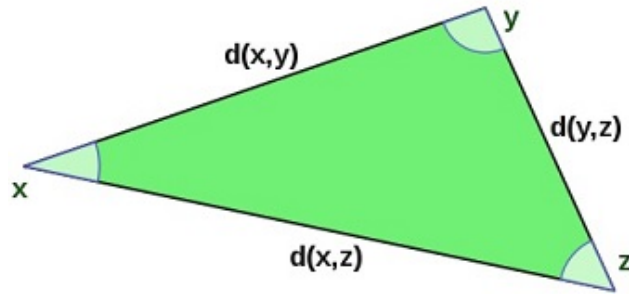


Figure 4.3: Triangle inequality in a metric space.svg from Wikipedia

The LHS is the absolute-value-squared of a complex scalar, while the RHS is the product of two vector norms (squared), each of which is necessarily a non-negative real. Therefore, it is an inequality between two non-negative values. In words, it says that the magnitude of an inner product is never more than the product of the two component vector magnitudes.

The Cauchy-Schwarz inequality becomes an *exact equality* if and only if the two vectors form a *linearly dependent* set, i.e., one is a scalar multiple of the other.

[**Exercise.** Pick two vectors in \mathbb{C}^3 and verify that the Cauchy-Schwarz inequality is satisfied. Do this at least twice, once when the two vectors are linearly independent and once when they are linearly dependent.]

4.5 Rays, Not Points

4.5.1 Modeling Quantum Systems

There is one final adjustment to be made if we are to accurately model quantum systems with these Hilbert spaces. It will simultaneously simplify your computations and confuse you. My hope is that there will be more “simplify” and less “confuse”.

Everything we do computationally will indeed take place in some Hilbert space $\mathcal{H} = \mathbb{C}^n$. In that regard, we have covered all the basic moves in this short lesson (minus *linear transformations*, which we present next time). However, in quantum mechanics, it is not true that every vector in \mathcal{H} corresponds to a distinct physical state of our system. Rather, the situation is actually a little easier (if we don’t resist). First a definition:

Definition of Ray. A *ray* (through the origin, $\mathbf{0}$) in a complex vector space is the set of all the scalar multiples of some non-zero vector,

$$\text{Ray belonging to } \mathbf{a} \neq \mathbf{0} \quad \equiv \quad \{\alpha \mathbf{a} \mid \alpha \in \mathbb{C}\}$$

Each possible physical state of our quantum system – a concept we will cover soon – will be represented by a *unit vector* (vector of length one) in \mathcal{H} . Stated differently, any two non-zero \mathcal{H} -vectors that differ by a complex scalar represent the same state,

so we may as well choose a vector of length one ($\|\mathbf{v}\| = 1$) that is on the same ray as either (or both) of those two; we don't have to distinguish between any of the infinite number of vectors on that ray.

This equivalence of all vectors on a given ray makes the objects in our mathematical model not *points* in \mathcal{H} , but *rays* through the *origin* of \mathcal{H} .

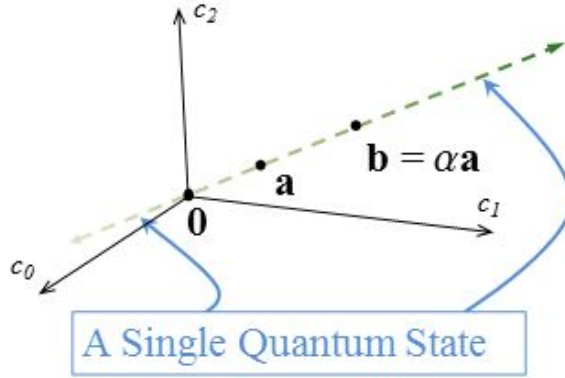


Figure 4.4: A “3-D” quantum state is a ray in its underlying $\mathcal{H} = \mathbb{C}^3$

If two complex n -tuples differ by a mere *scalar multiple*, α , they are to be considered the same state in our state space, i.e.,

$$\begin{aligned} \mathbf{b} &= \alpha \mathbf{a} \quad (\text{as } \mathcal{H}\text{-space vectors}) \\ &\Rightarrow \\ \mathbf{b} &\sim \mathbf{a} \quad (\text{as a physical state}), \end{aligned}$$

to be read “ \mathbf{a} is equivalent to \mathbf{b} ,” or “ \mathbf{a} and \mathbf{b} represent the same quantum state.” This identifies

[the quantum states modeled by \mathcal{H}]

with

[*rays* through the origin of that $\mathcal{H} = \mathbb{C}^n$].

(See Figure 4.4 for an example of an \mathcal{H} -state represented by its \mathbb{C}^3 ray.)

If we were to use brackets, $[\mathbf{a}]$, to mean “the ray represented by the n -tuple \mathbf{a} ,” then we could express a quantum state in \mathcal{H} , as

$$\text{An } \mathcal{H}\text{-“state”} \quad \longleftrightarrow \quad [\mathbf{a}] \quad = \quad \{\alpha \mathbf{a} \mid \alpha \in \mathbb{C}\}$$

Example. All three of these complex ordered-pairs lie on the same ray in \mathbb{C}^2 , so they all represent the same quantum state modeled by this two-dimensional \mathcal{H} :

$$\mathbf{a} = \begin{pmatrix} 2 \\ i \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} -10 \\ -5i \end{pmatrix} \quad \text{and} \quad \mathbf{c} = \begin{pmatrix} \frac{3-i}{5} \\ \frac{1+3i}{10} \end{pmatrix}$$

[**Exercise.** Elaborate.]

Dividing any one of them by its norm will produce a *unit vector* (vector with modulus one) which also represents the same ray. Using the simplest representative,

$$\frac{\mathbf{a}}{\|\mathbf{a}\|} = \frac{\begin{pmatrix} 2 \\ i \end{pmatrix}}{\sqrt{(2)(2) + (i)(-i)}} = \frac{\begin{pmatrix} 2 \\ i \end{pmatrix}}{\sqrt{5}} = \begin{pmatrix} \frac{2}{\sqrt{5}} \\ \frac{i}{\sqrt{5}} \end{pmatrix},$$

often written as

$$\frac{1}{\sqrt{5}} \begin{pmatrix} 2 \\ i \end{pmatrix}.$$

For example, if we ended up with \mathbf{a} as the answer to a problem, we could replace it by a *unit vector* along its ray,

$$\hat{\mathbf{a}} \equiv \frac{\mathbf{a}}{\|\mathbf{a}\|}.$$

(See Figure 4.5.)

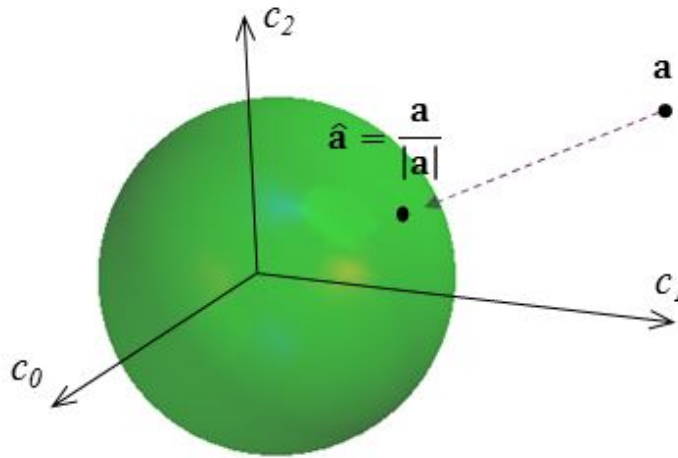


Figure 4.5: Dividing a vector by its norm yields a unit vector on the same ray

Computing a *unit length* alternative to a given vector in \mathcal{H} turns out to be of universal applicability in quantum mechanics, because it makes possible the computation of probabilities for each outcome of a measurement. The fact that a vector has norm = 1 corresponds to the various possible measurement probabilities adding to 1 (100% chance of getting *some* measurement). We'll see all this soon enough.

Caution. Figures 4.4 and 4.5 suggest that once you know the magnitude of \mathbf{a} , that will nail-it-down as a unique \mathbb{C}^3 representative at that distance along the ray. This is far from true. Each point pictured on the ray, itself, constitutes infinitely many different n -tuples in \mathbb{C}^n , all differing by a factor of $e^{i\theta}$, for some real θ .

Example (continued). We have seen that $\mathbf{a} = (2, i)^t$ has norm $\sqrt{5}$. A different \mathbb{C}^n representative for this \mathcal{H} -space vector is $(e^{\pi i/6}) \mathbf{a}$. We can easily see that $(e^{\pi i/6}) \mathbf{a}$ has the same length as \mathbf{a} . In words, $|e^{\pi i/6}| = 1$ (prove it or go back and review your complex arithmetic module), so multiplying by $e^{\pi i/6}$, while changing the \mathbb{C}^n vector, will not change its modulus (norm). Thus, that adjustment not only produces a different representative, it does so without changing the norm. Still, it doesn't hurt to calculate norm of the product the long way, just for exercise:

$$\begin{aligned}
 e^{\pi i/6} \mathbf{a} &= \begin{pmatrix} 2 e^{\pi i/6} \\ i e^{\pi i/6} \end{pmatrix} = \begin{pmatrix} 2 (\cos \pi/6 + i \sin \pi/6) \\ i (\cos \pi/6 + i \sin \pi/6) \end{pmatrix} \\
 &= \begin{pmatrix} 2 \cos \pi/6 + 2i \sin \pi/6 \\ -1 \sin \pi/6 + i \cos \pi/6 \end{pmatrix} \\
 &= \begin{pmatrix} 2 \frac{\sqrt{3}}{2} + 2i \frac{1}{2} \\ -1 \frac{1}{2} + i \frac{\sqrt{3}}{2} \end{pmatrix} \\
 &= \begin{pmatrix} \sqrt{3} + i \\ -\frac{1}{2} + \frac{i\sqrt{3}}{2} \end{pmatrix} \\
 &= \frac{1}{2} \begin{pmatrix} 2\sqrt{3} + 2i \\ -1 + i\sqrt{3} \end{pmatrix}
 \end{aligned}$$

I did the hard part: I simplified the *rotated* vector (we call the act of multiplying by $e^{i\theta}$ for real θ a *rotation* because of the geometric implication which you can imagine, look-up, or simply accept). All that's left to do in this computation is calculate the norm and see that it is $\sqrt{5}$.

[**Exercise.** Close the deal.]

4.5.2 $\mathbf{0}$ is not a Quantum State

The states we are to model correspond to *unit vectors* in \mathcal{H} . We all get that, now. But what are the implications?

- A state has to be a *normalizable* vector, which $\mathbf{0}$ is not. $\mathbf{0}$ will be the only vector in \mathcal{H} that does not correspond to a physical quantum state.
- As we already hammered home, every other vector does correspond to some state, but it isn't the *only* vector for that state. Any other vector that is a scalar multiple of it represents the same state.
- The mathematical entity that we get if we take the collection of all rays, $\{[\mathbf{a}]\}$, as its "points," is a new construct with the name: *complex projective sphere*. This new entity is, indeed, in one-to-one correspondence with the quantum states, but ...

- ... the complex projective sphere *is not a vector space*, so we don't want to go too far in attempting to define it; any attempt to make a formal mathematical entity just so that it corresponds, one-to-one, with the quantum states it models results in a non-vector space. Among other things, there is no $\mathbf{0}$ -vector in such a projective sphere, thus, no vector addition.

The Drill. With this in mind, we satisfy ourselves with the following process. It may lack concreteness until we start working on specific problems, but it should give you the feel for what's in store.

1. Identify a unit vector, $\hat{\mathbf{v}} \in \mathcal{H}$, corresponding to the quantum state of our problem.
2. Sometimes we will work with a scalar multiple, $\mathbf{v} = \alpha \hat{\mathbf{v}} \in \mathcal{H} = \mathbb{C}^n$, because it simplifies our computations and we know that this vector lies on the same ray as the original. Eventually, we'll *re-normalize* by dividing by α to bring it back to the projective sphere.
3. Often we will apply a *unitary* transformation, U , directly to $\hat{\mathbf{v}}$. $U\hat{\mathbf{v}}$ will be a unit vector because, by definition (upcoming lecture on *linear transformations*), unitary transformations preserve distances. Thus, U keeps vectors on the projective sphere.
4. In all cases, we will take care to apply valid operations to our unit “state vector,” making sure that we end up with an answer which is also a unit vector on the projective sphere.

4.5.3 Why?

There one question (at least) that you should ask and demand be answered.

Why is this called a “projective sphere?” Good question. Since the states of our quantum system are rays in \mathcal{H} , and we would prefer to visualize vectors as points, not rays, we go back to the underlying \mathbb{C}^n and *project* the entire ray (maybe *collapse* would be a better word) onto the surface of an n -dimensional sphere (whose *real* dimension is actually $2(n-1)$, but never mind that). We are *projecting* all those representatives onto a single point on the *complex n -sphere*. (See Figure 4.5.) **Cau-tion:** Each point on that sphere still has infinitely many representatives impossible to picture due to a potential scalar factor $e^{i\theta}$, for real θ .]

None of this is to say that scalar multiples, a.k.a. *phase changes*, never matter. When we start combining vectors in \mathcal{H} , their *relative phase* will become important, and so we shall need to retain individual scalars associated with each component n -tuple. Don't be intimidated; we'll get to that in cautious, deliberate steps.

4.6 Almost There

We have one last math lesson to dance through after which we will be ready to learn *graduate level quantum mechanics* (and do so without any prior knowledge of undergraduate quantum mechanics). This final topic is *linear transformations*. Rest up. Then attack it.

Chapter 5

Linear Transformations

5.1 Linear Transformations for Quantum Computing

5.1.1 A Concept More Fundamental Than the Matrix

In the last lecture we completed the mathematical preliminaries needed to formally define a *qubit*, the vector object that will model a physical memory location in a quantum computer. Of course, we haven't officially *defined* the qubit yet, but at least we have a pretty good idea that it will be associated with a vector on the projective sphere of some Hilbert space. We're fully armed to deal with qubits when the day comes (very soon, I promise).

This lesson will fill the remaining gap in your math: it provides the theory and computational skills needed to work with *quantum logic gates*, the hardware that *processes* qubits.

The introductory remarks of some prior lessons suggested that *a quantum logic gate will be described by a matrix*. Consider the Y operator, a quantum gate that we'll learn about later in the course. We'll see that Y takes *one qubit in* and produces *one qubit out*,

$$\text{---} \boxed{Y} \text{---} ,$$

and its matrix will be described by

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} .$$

Fair enough, except for one small problem.

A matrix for a logic gate implicitly assumes that there is an *underlying basis* that is being used in its construction. Since we know that every vector space can have many bases, it will turn out that “the” matrix for any logic gate like Y will mutate

depending which basis we are using. For example, there is a basis in which the same logic gate has a different matrix, i.e.,

$$Y = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

This should disturb you. If a matrix defines a quantum logic gate, and there can be more than one matrix describing that gate, how can we ever know anything?

There is a more fundamental concept than *the matrix* of linear algebra, that of *the linear transformation*. As you'll learn today, a linear transformation is the *basis-independent entity* that describes a logic gate. While a linear transformation's *matrix* will change depending on which underlying basis we use to construct it, its life-giving *linear transformation* remains fixed. You can wear different clothes, but underneath you're still you.

5.1.2 The Role of Linear Transformations in Quantum Computing

There are many reasons we care about linear transformations. I'll list a few to give you some context, and we'll discover more as we go.

- Every algorithm in quantum computing will require taking a *measurement*. In quantum mechanics, a *measurement* is associated with a certain type of linear transformation. Physicists call it by the name *Hermitian operator*, which we'll define shortly. Such beasts are, at their core, linear transformations. [**Beware:** I did not say that taking a measurement *was* a linear transformation; it is not. I said that every measurement *is associated with* a linear transformation.]
- In quantum computing we'll be replacing our old logic gates (AND, XOR, etc.) with a special kind of quantum gate whose basis independent entity is called a *unitary operator*. But a unitary operator is nothing more than a *linear transformation* which has some additional properties.
- We often want to express the same quantum state – or *qubit* – in different bases. One way to convert the coordinates of the qubit from one basis to another is to subject the coordinate vector to a *linear transformation*.

5.2 Definitions and Examples

5.2.1 Actions as well as Name Changes

Linear Transformations are the *verbs* of vector spaces. They can map vectors of one vector space, \mathcal{V} , into a *different* space, \mathcal{W} ,

$$\mathcal{V} \xrightarrow{T} \mathcal{W},$$

or they can move vectors around, keeping them in the *same* space,

$$\mathcal{V} \xrightarrow{T} \mathcal{V}.$$

They describe *actions* that we take on our vectors. They can move a vector by mapping it onto another vector. They can also be applied to vectors that don't move at all. For example, we often want to expand a vector along a basis that is different from the one originally provided, and linear transformations help us there as well.

5.2.2 Formal Definition of Linear Transformation

A *linear transformation*, T , is a map from a vector space (the *domain*) into itself or into another vector space (the *range*),

$$\mathcal{V} \xrightarrow{T} \mathcal{W} \quad (\mathcal{W} \text{ could be } \mathcal{V}),$$

that sends “input” vectors to “output” vectors,

$$T(\mathbf{v}) \mapsto \mathbf{w}.$$

That describes the *mapping* aspect (Figure 5.1). However, a *linear transformation*

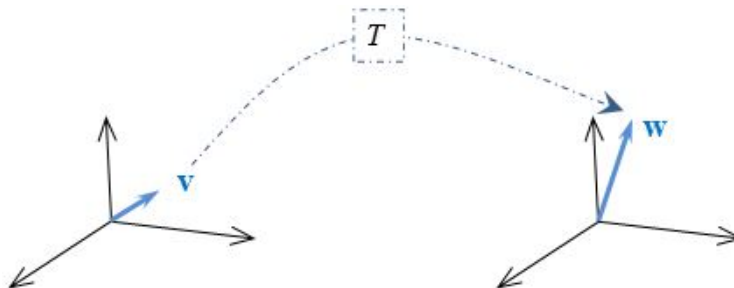


Figure 5.1: T mapping a vector \mathbf{v} in \mathbb{R}^3 to a \mathbf{w} in \mathbb{R}^3

has the following additional properties that allow us to call the mapping *linear*:

$$\begin{aligned} T(c\mathbf{v}) &= cT(\mathbf{v}) \quad \text{and} \\ T(\mathbf{v}_1 + \mathbf{v}_2) &= T(\mathbf{v}_1) + T(\mathbf{v}_2). \end{aligned}$$

Here, c is a *scalar* of the domain vector space and $\mathbf{v}_1, \mathbf{v}_2$ are domain *vectors*. These conditions have to be satisfied for all vectors and scalars.

Besides possibly being different spaces, the domain and range can also have different dimensions. However, today, we'll just work with linear transformation of a space into itself.

A linear transformation can be interpreted many different ways. When learning about them, it's best to think of them as mappings or positional changes which

convert a vector (having a certain direction and length) into a different vector (having a different direction and length).

Notation. Sometimes the parentheses are omitted when applying a linear transformation to a vector:

$$T\mathbf{v} \equiv T(\mathbf{v}).$$

Famous Linear Transformations

Here are a few useful linear transformations, some of which refer to \mathbb{R}^n or \mathbb{C}^n , some to function spaces not studied much in this course. I've included figures for a few that we'll meet later today.

\mathbf{v} is any vector in an n dimensional vector space on which the linear transformation acts. $\hat{\mathbf{x}}_k$ is the k th natural basis vector, v_k is the k th coordinate of \mathbf{v} in the natural basis, c is a scalar, and $\hat{\mathbf{n}}$ is any unit vector.

$$\mathbb{1}(\mathbf{v}) \equiv \mathbf{v} \quad (\text{Identity})$$

$$0(\mathbf{v}) \equiv \mathbf{0} \quad (\text{Zero})$$

$$S_c(\mathbf{v}) \equiv c\mathbf{v} \quad (\text{Scale}) \quad (\text{Figure 5.2})$$

$$\mathcal{P}_k(\mathbf{v}) \equiv v_k \hat{\mathbf{x}}_k \quad (\text{Projection onto } \hat{\mathbf{x}}_k) \quad (\text{Figure 5.3})$$

$$\mathcal{P}_{\hat{\mathbf{n}}}(\mathbf{v}) \equiv (\mathbf{v} \cdot \hat{\mathbf{n}}) \hat{\mathbf{n}} \quad (\text{Projection onto } \hat{\mathbf{n}}) \quad (\text{Figure 5.4})$$

$$D(\varphi) \equiv \varphi' \quad (\text{Differentiation})$$

$$\int^x (f) \equiv \int^x f(x') dx' \quad (\text{Anti-differentiation})$$

$$T_A(\mathbf{v}) \equiv A\mathbf{v} \quad (\text{Multiplication by matrix of constants, } A)$$

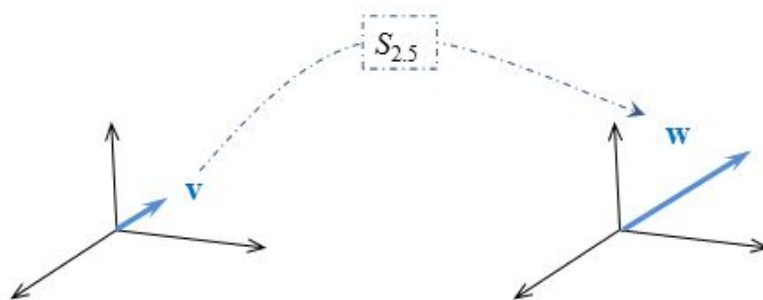


Figure 5.2: A scaling transformation

Let's pick one and demonstrate that it is, indeed, linear. We can prove both conditions at the same time using the vector $c\mathbf{v}_1 + \mathbf{v}_2$ as a starting point. I've

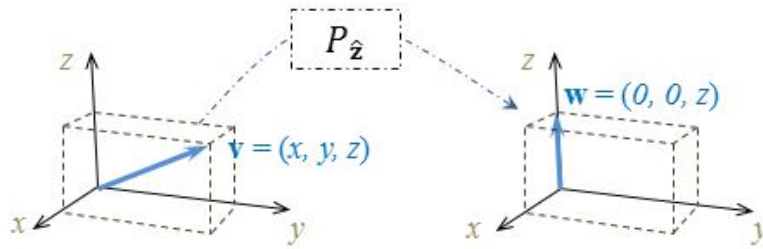


Figure 5.3: Projection onto the direction $\hat{\mathbf{z}}$, a.k.a. $\hat{\mathbf{x}}_3$

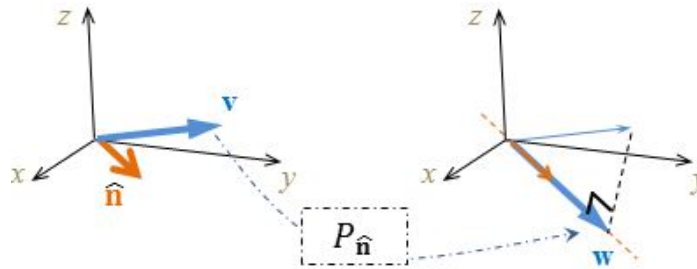


Figure 5.4: Projection onto an arbitrary direction $\hat{\mathbf{n}}$

selected the projection onto an arbitrary direction, $\hat{\mathbf{n}}$, as the example.

$$\begin{aligned}
 \mathcal{P}_{\hat{\mathbf{n}}}(c\mathbf{v}_1 + \mathbf{v}_2) &= [(c\mathbf{v}_1 + \mathbf{v}_2) \cdot \hat{\mathbf{n}}] \hat{\mathbf{n}} = [(c\mathbf{v}_1) \cdot \hat{\mathbf{n}} + \mathbf{v}_2 \cdot \hat{\mathbf{n}}] \hat{\mathbf{n}} \\
 &= [c(\mathbf{v}_1 \cdot \hat{\mathbf{n}}) + \mathbf{v}_2 \cdot \hat{\mathbf{n}}] \hat{\mathbf{n}} = [c(\mathbf{v}_1 \cdot \hat{\mathbf{n}})] \hat{\mathbf{n}} + [\mathbf{v}_2 \cdot \hat{\mathbf{n}}] \hat{\mathbf{n}} \\
 &= c[(\mathbf{v}_1 \cdot \hat{\mathbf{n}})] \hat{\mathbf{n}} + [\mathbf{v}_2 \cdot \hat{\mathbf{n}}] \hat{\mathbf{n}} \\
 &= c\mathcal{P}_{\hat{\mathbf{n}}}(\mathbf{v}_1) + \mathcal{P}_{\hat{\mathbf{n}}}(\mathbf{v}_2). \quad \text{QED}
 \end{aligned}$$

We also listed an example in which T_A was defined by the multiplicative action of a matrix, A , consisting of scalar constants (i.e., the matrix can't have variables or formulas in it). To make this precise, let A have size $m \times n$ and \mathbf{v} be any n -dimensional vector. (They both have to use the same field of scalars, say \mathbb{R} or \mathbb{C} .) From the rules of matrix multiplication we can easily see that

$$\begin{aligned}
 A(\mathbf{v} + \mathbf{w}) &= A(\mathbf{v}) + A(\mathbf{w}) \quad \text{and} \\
 A(c\mathbf{v}) &= cA(\mathbf{v}).
 \end{aligned}$$

Therefore,

$$T_A(\mathbf{v}) \equiv A\mathbf{v}$$

is linear. This is the linear transformation, T_A , *induced* by the matrix A . We'll look at it more closely in a moment.

[Exercise. Prove these two claims about matrix-multiplication.]

[**Exercise.** Look at these mapping of \mathbb{C}^3 into itself.

$$\begin{aligned} T_1 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} &\longmapsto \begin{pmatrix} 2x \\ 2y \\ 2z \end{pmatrix} \\ T_2 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} &\longmapsto \begin{pmatrix} 2ix \\ \sqrt{3} y \\ (4 - 1i)z \end{pmatrix} \\ T_3 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} &\longmapsto \begin{pmatrix} x + 2 \\ y + 2i \\ z + \sqrt{2} \end{pmatrix} \\ T_4 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} &\longmapsto \begin{pmatrix} -y \\ x \\ z \end{pmatrix} \\ T_5 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} &\longmapsto \begin{pmatrix} xy \\ y \\ z^2 \end{pmatrix} \\ T_6 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} &\longmapsto \begin{pmatrix} 0 \\ xyz \\ 0 \end{pmatrix} \end{aligned}$$

Which are linear, which are not? Support each claim with a proof or counter example.]

5.3 The Special Role of Bases

Basis vectors play a powerful role in the study of linear transformations as a consequence of linearity.

Let's pick any basis for our domain vector space (it doesn't have to be the natural basis or even be orthonormal),

$$\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \dots\} .$$

The definition of basis means we can express any vector as a linear combination of the \mathbf{b}_k s (uniquely) using the appropriate scalars, β_k , from the underlying field. Now, choose any random vector from the space and expand it along this basis (and remember there is only one way to do this for every basis),

$$\mathbf{v} = \sum_{k=1}^n \beta_k \mathbf{b}_k .$$

We now apply T to \mathbf{v} and make use of the linearity (the sum could be infinite) to get

$$T\mathbf{v} = T\left(\sum_{k=1}^n \beta_k \mathbf{b}_k\right) = \sum_{k=1}^n \beta_k T(\mathbf{b}_k)$$

What does this say? It tells us that if we know what T does to the *basis* vectors, we know what it does to *all* vectors in the space. Let's say T is some hard-to-determine function which is actually not known analytically (by formula), but by experimentation we are able to determine its action on the basis. We can extend that knowledge to any vector because the last result tells us that the coordinates of the vector combined with the known values of T on the basis are enough. In short, the small set of vectors

$$\{T(\mathbf{b}_1), T(\mathbf{b}_2), \dots\}$$

completely determines T .

5.3.1 Application: Rotations in Space

We now apply the theory to a linear transformation which seems to rotate points by 90° counter-clockwise in \mathbb{R}^2 . Let's call the rotation $R_{\pi/2}$, since $\pi/2$ radians is 90° .

The plan

We must first believe that a rotation is *linear*. I leave this to you.

[**Exercise.** Argue, heuristically, that $R_{\pi/2}$ is linear. **Hint:** Use the equations that define linearity, but proceed intuitively, since you don't yet have a formula for $R_{\pi/2}$.]

Next, we will use geometry to easily find the action of T on the two standard basis vectors, $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$. Finally, we'll extend that to all of \mathbb{R}^2 , using linearity. Now for the details.

The Details

Figures 5.5 and 5.6 show the result of the rotation on the two basis vectors.

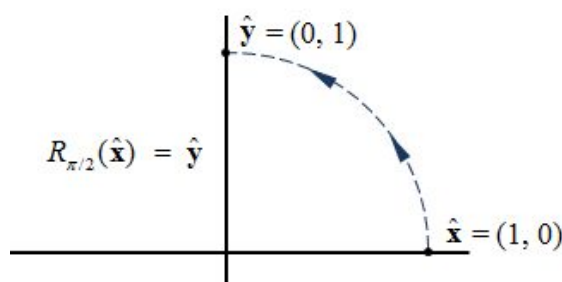


Figure 5.5: Rotation of $\hat{\mathbf{x}}$ counter-clockwise by $\pi/2$

These figures suggest that, in \mathbb{R}^2 ,

$$\begin{aligned} R_{\pi/2}(\hat{\mathbf{x}}) &= \hat{\mathbf{y}} \quad \text{and} \\ R_{\pi/2}(\hat{\mathbf{y}}) &= -\hat{\mathbf{x}}, \end{aligned}$$

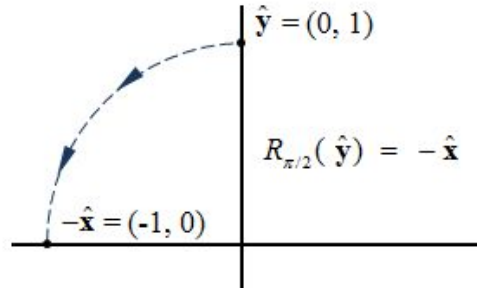


Figure 5.6: Rotation of $\hat{\mathbf{y}}$ counter-clockwise by $\pi/2$

therefore, for *any* \mathbf{v} , with natural basis coordinates $(v_x, v_y)^t$, linearity allows us to write

$$\begin{aligned} R_{\pi/2}(\mathbf{v}) &= R_{\pi/2}(v_x \hat{\mathbf{x}} + v_y \hat{\mathbf{y}}) \\ &= v_x R_{\pi/2}(\hat{\mathbf{x}}) + v_y R_{\pi/2}(\hat{\mathbf{y}}) \\ &= v_x \hat{\mathbf{y}} - v_y \hat{\mathbf{x}}. \end{aligned}$$

From knowledge of the linear transformation on the basis alone, we were able to derive a formula applicable to the entire space. Stated in column vector form,

$$R_{\pi/2} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix},$$

and we have our formula for all space (assuming the natural basis coordinates). It's that easy.

[**Exercise.** Develop the formula for a counter-clockwise rotation (again in \mathbb{R}^2) through an *arbitrary* angle, θ . Show your derivation based on its effect on the natural basis vectors.]

[**Exercise.** What is the formula for a rotation, $R_{z, \pi/2}$, about the z -axis in \mathbb{R}^3 through a 90° angle, counter-clockwise when looking down from the positive z -axis. Show your derivation based on its effect on the natural basis vectors, $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$.]

5.4 The Matrix of a Linear Transformation

5.4.1 From Matrix to Linear Transformation

You showed in one of your exercises above that any matrix A (containing scalar constants) induces a linear transformation T_A . Specifically, say A is a complex matrix of size $m \times n$ and \mathbf{v} is a vector in \mathbb{C}^n . Then the formula

$$T_A(\mathbf{v}) \equiv A\mathbf{v}$$

defines a mapping

$$\mathbb{C}^n \xrightarrow{T_A} \mathbb{C}^m,$$

which turns out to be linear. As you see, both A , and therefore, T_A , might map vectors into a different-sized vector space; sometimes $m > n$, sometimes $m < n$ and sometimes $m = n$.

5.4.2 From Linear Transformation to Matrix

We can go the other way. Starting with a linear transformation, T , we can construct a matrix M_T , that represents it. There is a little fine print here which we'll get to in a moment (and you may sense it even before we get there).

For simplicity, assume we are working in a vector space and using some natural basis $\mathcal{A} = \{\mathbf{a}_k\}$. (Think $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$ of \mathbb{R}^3 .) So every vector can be expanded along \mathcal{A} by

$$\mathbf{v} = \sum_{k=1}^n \alpha_k \mathbf{a}_k.$$

We showed that the action of T on the few vectors in basis \mathcal{A} completely determines its definition on *all* vectors \mathbf{v} . This happened because of linearity,

$$T\mathbf{v} = \sum_{k=1}^n \alpha_k T(\mathbf{a}_k).$$

Let's write the sum in a more instructive way as the formal (but not quite legal) "dot product" of a (*row of vectors*) with a (*column of scalars*). Shorthand for the above sum then becomes

$$T\mathbf{v} = \left(T(\mathbf{a}_1), T(\mathbf{a}_2), \dots, T(\mathbf{a}_n) \right) \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}.$$

Now we expand each vector $T(\mathbf{a}_k)$ vertically into its coordinates relative to the same basis, \mathcal{A} , and we will have a legitimate product,

$$\begin{pmatrix} (T\mathbf{a}_1)_1 & (T\mathbf{a}_2)_1 & \cdots & (T\mathbf{a}_n)_1 \\ (T\mathbf{a}_1)_2 & (T\mathbf{a}_2)_2 & \cdots & (T\mathbf{a}_n)_2 \\ \vdots & \vdots & \ddots & \vdots \\ (T\mathbf{a}_1)_n & (T\mathbf{a}_2)_n & \cdots & (T\mathbf{a}_n)_n \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}.$$

Writing a_{jk} in place of $(T\mathbf{a}_k)_j$, we have the simpler statement,

$$T\mathbf{v} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix},$$

which reveals that T is nothing more than multiplication by a matrix made up of the constants a_{jk} .

Executive Summary. To get a matrix, M_T , for any linear transformation, T , form a matrix whose columns are T applied to each basis vector.

We can then multiply any vector \mathbf{v} by the matrix $M_T = (a_{jk})$ to get $T(\mathbf{v})$.

Notation

Because this duality between matrices and linear transformations is so tight, we rarely bother to distinguish one from the other. If we start with a linear transformation, T , we just use T as its matrix (and do away with the notation M_T). If we start with a matrix, A , we just use A as its induced linear transformation, (and do away with the notation T_A).

Example

We've got the formula for a linear transformation that rotates vectors counter-clockwise in \mathbb{R}^2 , namely,

$$R_{\pi/2} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix}.$$

To compute its matrix relative to the natural basis, $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$, we form

$$\begin{aligned} M_{R_{\pi/2}} &= \left(R_{\pi/2}(\hat{\mathbf{x}}), R_{\pi/2}(\hat{\mathbf{y}}) \right) = \left(R_{\pi/2} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, R_{\pi/2} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \\ &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \end{aligned}$$

We can verify that it works by multiplying this matrix by an arbitrary vector,

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \cdot x + (-1) \cdot y \\ 1 \cdot x + 0 \cdot y \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix},$$

as required.

[Exercise.] Show that the matrix for the scaling transformation ,

$$S_{3i}(\mathbf{v}) \equiv 3i \mathbf{v},$$

is

$$M_{S_{3i}} = \begin{pmatrix} 3i & 0 \\ 0 & 3i \end{pmatrix},$$

and verify that it works by multiplying this matrix by an arbitrary vector to recover the definition of S_{3i} .

5.4.3 Dependence of a Matrix on Basis

When we are using an obvious and natural orthonormal basis (like $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$ for \mathbb{C}^3), everything is fairly straightforward. However, there will be times when we want a different basis than the one we thought, originally, was the *preferred* or *natural*. The rule is very strict, but simple.

In order to enable matrix multiplication to give us the result of some linear T applied to a vector \mathbf{v} , we must make sure T and \mathbf{v} are both expressed in a common basis. In that case, $\mathbf{w} = M_T \cdot \mathbf{v}$ only makes sense if both vectors and all the columns of M_T are expanded along that common basis.

In short, don't mix bases when you are expressing vectors and linear transformations. And don't forget what your underlying common basis is, especially if it's not the preferred basis.

Remember, there is always some innate definition of our vectors, independent of basis. Distinct from this, we have the expression of those vectors in a basis.

Say \mathbf{v} is a vector. If we want to express it using coordinates along the \mathcal{A} basis on Sunday, then “cabaret” along the \mathcal{B} basis all day Monday, that's perfectly fine. They are all the same object, and this notation could be employed to express that fact:

$$\mathbf{v} = \mathbf{v}|_{\mathcal{A}} = \mathbf{v}|_{\mathcal{B}}$$

This is confusing for beginners, because they usually start with vectors in \mathbb{R}^2 , which have *innate* coordinates, $\begin{pmatrix} x \\ y \end{pmatrix}$, that really don't refer to a basis. As we've seen, though, the vector happens to look the same as its expression in the natural basis, $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$, so without the extra notation, above, there's no way to know (nor is there usually the *need* to know) whether the author is referring to the vector, or its coordinates in the natural basis. If we ever want to disambiguate the conversation, we will use the notation, $\mathbf{v}|_{\mathcal{A}}$, where \mathcal{A} is some basis we have previously described.

The same goes for T 's matrix. If I want to describe T in a particular basis, I will say something like

$$T = T|_{\mathcal{A}} = T|_{\mathcal{B}},$$

and now I don't even need to use M_T , since the $|_{\mathcal{A}}$ or $|_{\mathcal{B}}$ implies we are talking about a matrix.

So the basis-free statement,

$$\mathbf{w} = T(\mathbf{v}),$$

can be viewed in one basis as

$$\mathbf{w}|_{\mathcal{A}} = T|_{\mathcal{A}} \cdot \mathbf{v}|_{\mathcal{A}},$$

and in another basis as

$$\mathbf{w}|_{\mathcal{B}} = T|_{\mathcal{B}} \cdot \mathbf{v}|_{\mathcal{B}}.$$

Matrix of M_T in a Non-Standard Basis

The development of our formula of a matrix M_T of a transformation, T , led to

$$M_T = \begin{pmatrix} T(\mathbf{a}_1), & T(\mathbf{a}_2), & \dots, & T(\mathbf{a}_n) \end{pmatrix}$$

when we were assuming the preferred basis,

$$\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \dots\} = \left\{ \begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ \vdots \end{pmatrix}, \dots \right\}.$$

There was nothing special about the preferred basis in this formula; if we had any basis – even one that was non-orthonormal – the formula would still be

$$T|_{\mathcal{B}} = \begin{pmatrix} T(\mathbf{b}_1)|_{\mathcal{B}}, & T(\mathbf{b}_2)|_{\mathcal{B}}, & \dots, & T(\mathbf{b}_n)|_{\mathcal{B}} \end{pmatrix}.$$

The way to see this most easily is to first note that in any basis \mathcal{B} , each basis vector \mathbf{b}_k , when expressed in its own \mathcal{B} -coordinates, looks exactly like the k th preferred basis element, i.e.,

$$\mathbf{b}_k|_{\mathcal{B}} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}|_{\mathcal{B}} \longleftarrow k\text{th element}.$$

[**Exercise.** prove it.]

Now we can see that the proposed $T|_{\mathcal{B}}$ applied to $\mathbf{b}_k|_{\mathcal{B}}$ gives the correct result. Writing everything in the \mathcal{B} basis (some labels omitted) and using matrix multiplication we have

$$\begin{pmatrix} \dots, & T(\mathbf{b}_k)|_{\mathcal{B}}, & \dots \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} = T(\mathbf{b}_k)|_{\mathcal{B}}. \quad \checkmark$$

Example

The transformation

$$T : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ (3i)y \end{pmatrix}$$

in the context of the vector space \mathbb{C}^2 is going to have the preferred-basis matrix

$$M_T = \begin{pmatrix} 1 & 0 \\ 0 & 3i \end{pmatrix}.$$

[**Exercise.** Prove it.]

Let's see what T looks like when expressed in the non-orthogonal basis

$$\mathcal{C} = \left\{ \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}.$$

[**Exercise.** Prove that this \mathcal{C} is a basis for \mathbb{C}^2 .]

Using our formula, we get

$$T|_{\mathcal{C}} = \left(T(\mathbf{c}_1)|_{\mathcal{C}}, T(\mathbf{c}_2)|_{\mathcal{C}} \right).$$

We compute each $T(\mathbf{c}_k)|_{\mathcal{C}}$, $k = 1, 2$.

First up, $T(\mathbf{c}_1)|_{\mathcal{C}}$:

$$T(\mathbf{c}_1) = T \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}.$$

Everything needs to be expressed in the \mathcal{C} basis, so we show that for this vector:

$$\begin{pmatrix} 2 \\ 0 \end{pmatrix} = 1 \begin{pmatrix} 2 \\ 0 \end{pmatrix} + 0 \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}|_{\mathcal{C}},$$

so this last column vector will be the first column of our matrix.

Next, $T(\mathbf{c}_2)|_{\mathcal{C}}$:

$$T(\mathbf{c}_2) = T \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 3i \end{pmatrix}.$$

We have to express this, too, in the \mathcal{C} basis, a task that requires a modicum of algebra:

$$\begin{pmatrix} 1 \\ 3i \end{pmatrix} = \alpha \begin{pmatrix} 2 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Solving this system should be no problem for you (exercise), giving,

$$\begin{aligned} \alpha &= \frac{1-3i}{2} \\ \beta &= 3i, \end{aligned}$$

so

$$T(\mathbf{c}_2) = \begin{pmatrix} 1 \\ 3i \end{pmatrix} = \left(\begin{pmatrix} \frac{1-3i}{2} \\ 3i \end{pmatrix} \right) \Big|_c$$

Plugging these in, we get,

$$T \Big|_c = \left(\begin{pmatrix} 1 & \frac{1-3i}{2} \\ 0 & 3i \end{pmatrix} \right) \Big|_c.$$

5.4.4 The Transformation in an Orthonormal Basis

When we have an orthonormal basis, things get easy. We start our analysis by working in a natural, orthonormal basis, where all vectors and matrices are simply written down without even thinking about the basis, even though it's always there, lurking behind the page. Today, we're using \mathcal{A} to designate the natural basis, so

$$\mathbf{v} = \mathbf{v}|_{\mathcal{A}}$$

would have the same coordinates on both sides of the equations, perhaps $\begin{pmatrix} 1 + i\sqrt{2} \\ 3.2 - i \end{pmatrix}$.

Formula for Matrix Elements Using the Inner Product Trick

Even though we already know how to get the matrix in the preferred basis, \mathcal{A} , let's analyze it, briefly. Take the scaling transformation $S_{\sqrt{2}/2}$ in \mathbb{R}^2 whose matrix is

$$\left(\begin{pmatrix} \sqrt{2}/2 & 0 \\ 0 & \sqrt{2}/2 \end{pmatrix} \right) \Big|_{\mathcal{A}}.$$

(You constructed the matrix for an S_c in an earlier exercise today using the scaling factor $c = 3i$, but just to summarize, we apply $S_{\sqrt{2}/2}$ to the two vectors $(1, 0)^t$ and $(0, 1)^t$, then make the output vectors the columns of the required matrix.) Remember, each column is computed by listing the coordinates of $T(\hat{\mathbf{x}})$ and $T(\hat{\mathbf{y}})$ in the natural basis. But for either vector, its coordinates can be expressed using the “dotting” trick because we have an orthonormal basis. So,

$$\left(\begin{pmatrix} \sqrt{2}/2 & 0 \\ 0 & \sqrt{2}/2 \end{pmatrix} \right) \Big|_{\mathcal{A}} = \begin{pmatrix} \langle \hat{\mathbf{x}} | \begin{pmatrix} \sqrt{2}/2 \\ 0 \end{pmatrix} \rangle & \langle \hat{\mathbf{x}} | \begin{pmatrix} 0 \\ \sqrt{2}/2 \end{pmatrix} \rangle \\ \langle \hat{\mathbf{y}} | \begin{pmatrix} \sqrt{2}/2 \\ 0 \end{pmatrix} \rangle & \langle \hat{\mathbf{y}} | \begin{pmatrix} 0 \\ \sqrt{2}/2 \end{pmatrix} \rangle \end{pmatrix}.$$

This is true for any T and the preferred basis in \mathbb{R}^2 . To illustrate, let's take the upper left component. It's given by the dot product

$$T_{11} = \langle \hat{\mathbf{x}} | T(\hat{\mathbf{x}}) \rangle.$$

Similarly, the other three are given by

$$\begin{aligned} T_{21} &= \langle \hat{\mathbf{y}} | T(\hat{\mathbf{x}}) \rangle , \\ T_{12} &= \langle \hat{\mathbf{x}} | T(\hat{\mathbf{y}}) \rangle \quad \text{and} \\ T_{22} &= \langle \hat{\mathbf{y}} | T(\hat{\mathbf{y}}) \rangle . \end{aligned}$$

In other words,

$$T|_{\mathcal{A}} = \begin{pmatrix} \langle \hat{\mathbf{x}} | T(\hat{\mathbf{x}}) \rangle & \langle \hat{\mathbf{x}} | T(\hat{\mathbf{y}}) \rangle \\ \langle \hat{\mathbf{y}} | T(\hat{\mathbf{x}}) \rangle & \langle \hat{\mathbf{y}} | T(\hat{\mathbf{y}}) \rangle \end{pmatrix} .$$

To make things crystal clear, let's rename the natural basis vectors

$$\mathcal{A} \equiv \{ \hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2 \},$$

producing the very illuminating

$$T|_{\mathcal{A}} = \begin{pmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{pmatrix} = \begin{pmatrix} \langle \hat{\mathbf{e}}_1 | T(\hat{\mathbf{e}}_1) \rangle & \langle \hat{\mathbf{e}}_1 | T(\hat{\mathbf{e}}_2) \rangle \\ \langle \hat{\mathbf{e}}_2 | T(\hat{\mathbf{e}}_1) \rangle & \langle \hat{\mathbf{e}}_2 | T(\hat{\mathbf{e}}_2) \rangle \end{pmatrix} .$$

But this formula, and the logic that led to it, would work for *any* orthonormal basis, not just \mathcal{A} , and in any vector space, not just \mathbb{R}^2 .

Summary. The jk th matrix element for the transformation, T , in an *orthonormal* basis,

$$\mathcal{B} = \{ \mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n \}$$

is given by

$$T_{jk}|_{\mathcal{B}} = \langle \hat{\mathbf{b}}_j | T(\hat{\mathbf{b}}_k) \rangle ,$$

so

$$T|_{\mathcal{B}} = \begin{pmatrix} \langle \hat{\mathbf{b}}_1 | T(\hat{\mathbf{b}}_1) \rangle & \langle \hat{\mathbf{b}}_1 | T(\hat{\mathbf{b}}_2) \rangle \\ \langle \hat{\mathbf{b}}_2 | T(\hat{\mathbf{b}}_1) \rangle & \langle \hat{\mathbf{b}}_2 | T(\hat{\mathbf{b}}_2) \rangle \end{pmatrix} .$$

Not only that, but we don't even have to start with a preferred basis to express our T and \mathcal{B} that are used in the formula. As long as T and \mathcal{B} are both expressed in the *same* basis – say some third \mathcal{C} , also orthonormal – we can use the coordinates and matrix elements relative to \mathcal{C} to compute $\langle \hat{\mathbf{b}}_j | T(\hat{\mathbf{b}}_k) \rangle$ and thus give us the matrix of $T_{\mathcal{B}}$.

Example 1

Let's represent the scaling transformation

$$S_{\sqrt{2}/2} = \left(\begin{array}{cc} \sqrt{2}/2 & 0 \\ 0 & \sqrt{2}/2 \end{array} \right) \Big|_{\mathcal{A}}.$$

in a basis that we encountered in a previous lesson (which was named \mathcal{C} then, but we'll call \mathcal{B} today, to make clear the application of the above formulas),

$$\mathcal{B} \equiv \{\mathbf{b}_1, \mathbf{b}_2\} = \left\{ \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}, \begin{pmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} \right\}.$$

We just plug in (intermediate \mathcal{B} labels omitted):

$$\begin{aligned} S_{\sqrt{2}/2} \Big|_{\mathcal{B}} &= \begin{pmatrix} \langle \hat{\mathbf{b}}_1 | T(\hat{\mathbf{b}}_1) \rangle & \langle \hat{\mathbf{b}}_1 | T(\hat{\mathbf{b}}_2) \rangle \\ \langle \hat{\mathbf{b}}_2 | T(\hat{\mathbf{b}}_1) \rangle & \langle \hat{\mathbf{b}}_2 | T(\hat{\mathbf{b}}_2) \rangle \end{pmatrix} \\ &= \begin{pmatrix} \langle \hat{\mathbf{b}}_1 | \frac{\sqrt{2}}{2} \hat{\mathbf{b}}_1 \rangle & \langle \hat{\mathbf{b}}_1 | \frac{\sqrt{2}}{2} \hat{\mathbf{b}}_2 \rangle \\ \langle \hat{\mathbf{b}}_2 | \frac{\sqrt{2}}{2} \hat{\mathbf{b}}_1 \rangle & \langle \hat{\mathbf{b}}_2 | \frac{\sqrt{2}}{2} \hat{\mathbf{b}}_2 \rangle \end{pmatrix} \\ &= \begin{pmatrix} \langle \hat{\mathbf{b}}_1 | \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \rangle & \langle \hat{\mathbf{b}}_1 | \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \rangle \\ \langle \hat{\mathbf{b}}_2 | \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \rangle & \langle \hat{\mathbf{b}}_2 | \begin{pmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \rangle \end{pmatrix} \\ &= \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{pmatrix} \Big|_{\mathcal{B}}. \end{aligned}$$

That's surprising (or not). The matrix is the same in the \mathcal{B} basis as it is in the \mathcal{A} basis.

- 1) Does it make sense?
- 2) Is this going to be true for all orthonormal bases and all transformations?

These are the kinds of questions you have to ask yourself when you are manipulating mathematical symbols in new and unfamiliar territory. I'll walk you through it.

1) Does it make sense?

Let's look at the result of

$$S_{\sqrt{2}/2} \begin{pmatrix} -3 \\ 10 \end{pmatrix}$$

in both bases.

- First we'll compute it in the \mathcal{A} -basis and transform the resulting output vector to the \mathcal{B} -basis.
- When we've done that, we'll start over, but this time first convert the starting vector and matrix into \mathcal{B} -basis coordinates and use those to compute the result, giving us an answer in terms of the \mathcal{B} -basis.
- We'll compare the two answers to see if they are equal.

By picking a somewhat random-looking vector, $(-3, 10)^t$, and discovering that both $T|_{\mathcal{A}}$ and $T|_{\mathcal{B}}$ turn it into equivalent output vectors, we will be satisfied that the result makes sense; apparently it is true that the matrix for $S_{\sqrt{2}/2}$ looks the same when viewed in both bases. But we haven't tested that, so let's do it.

\mathcal{A} -Basis. In the \mathcal{A} basis we already know that the output vector must be

$$\begin{pmatrix} -3\sqrt{2}/2 \\ 5\sqrt{2} \end{pmatrix} \Big|_{\mathcal{A}},$$

because that's just the application of the transformation to the innate vector, which is the same as applying the matrix to the *preferred* coordinates. We transform the output vector into \mathcal{B} -basis:

$$\begin{aligned} \begin{pmatrix} -3\sqrt{2}/2 \\ 5\sqrt{2} \end{pmatrix} \Big|_{\mathcal{A}} &= \begin{pmatrix} \left\langle \mathbf{b}_1 \middle| \begin{pmatrix} -3\sqrt{2}/2 \\ 5\sqrt{2} \end{pmatrix} \right\rangle \\ \left\langle \mathbf{b}_2 \middle| \begin{pmatrix} -3\sqrt{2}/2 \\ 5\sqrt{2} \end{pmatrix} \right\rangle \end{pmatrix} \Big|_{\mathcal{B}} \\ &= \begin{pmatrix} -\frac{3}{2} + 5 \\ \frac{3}{2} + 5 \end{pmatrix} \Big|_{\mathcal{B}} = \begin{pmatrix} \frac{7}{2} \\ \frac{13}{2} \end{pmatrix} \Big|_{\mathcal{B}} \end{aligned}$$

\mathcal{B} -Basis. Now, do it again, but this time convert the input vector, $\begin{pmatrix} -3 \\ 10 \end{pmatrix}$, to \mathcal{B} -coordinates, and run it through $S_{\sqrt{2}/2}|_{\mathcal{B}}$. First the input vector:

$$\begin{aligned} \begin{pmatrix} -3 \\ 10 \end{pmatrix} \Big|_{\mathcal{A}} &= \begin{pmatrix} \left\langle \mathbf{b}_1 \middle| \begin{pmatrix} -3 \\ 10 \end{pmatrix} \right\rangle \\ \left\langle \mathbf{b}_2 \middle| \begin{pmatrix} -3 \\ 10 \end{pmatrix} \right\rangle \end{pmatrix} \Big|_{\mathcal{B}} \\ &= \begin{pmatrix} -\frac{3\sqrt{2}}{2} + \frac{10\sqrt{2}}{2} \\ \frac{3\sqrt{2}}{2} + \frac{10\sqrt{2}}{2} \end{pmatrix} \Big|_{\mathcal{B}} = \begin{pmatrix} \frac{7\sqrt{2}}{2} \\ \frac{13\sqrt{2}}{2} \end{pmatrix} \Big|_{\mathcal{B}}. \end{aligned}$$

Finally, we apply $S_{\sqrt{2}/2}|_{\mathcal{B}}$ to these \mathcal{B} coordinates:

$$\begin{aligned} S_{\sqrt{2}/2}|_{\mathcal{B}} \begin{pmatrix} \frac{7\sqrt{2}}{2} \\ \frac{13\sqrt{2}}{2} \end{pmatrix} \Big|_{\mathcal{B}} &= \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} \end{pmatrix} \begin{pmatrix} \frac{7\sqrt{2}}{2} \\ \frac{13\sqrt{2}}{2} \end{pmatrix} \\ &= \begin{pmatrix} 7 \\ 13 \end{pmatrix} \Big|_{\mathcal{B}}. \end{aligned}$$

And ... we get the same answer. Apparently there is no disagreement, and other tests would give the same results, so it seems we made no mistake when we derived the matrix for $S_{\sqrt{2}/2}|_{\mathcal{B}}$ and got the same answer as $S_{\sqrt{2}/2}|_{\mathcal{A}}$.

This is not a proof, but it easy enough to do (next exercise). A mathematically-minded student might prefer to just do the proof and be done with it, while an applications-oriented person may prefer just to test it on a random vector to see if s/he is on the right track.

[**Exercise.** Prove that for *any* \mathbf{v} , you get the same result for $S_{\sqrt{2}/2}(\mathbf{v})$ whether you use coordinates in the \mathcal{A} basis or in the \mathcal{B} basis, thus confirming once-and-for-all that the matrix of this scaling transformation is the same in both bases.]

2) Do we always get the same matrix?

Certainly not – otherwise would I have burdened you with a formula for computing the matrix of a linear transformation in an arbitrary basis?

Example 2

We compute the matrices for a projection transformation,

$$\mathcal{P}_{\hat{\mathbf{n}}}(\mathbf{v}) \equiv (\mathbf{v} \cdot \hat{\mathbf{n}}) \hat{\mathbf{n}}, \quad \text{where } \hat{\mathbf{n}} = \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}.$$

in the two bases, above.

The \mathcal{A} Matrix. I'll use the common notation introduced earlier, $\hat{\mathbf{e}}_k$, for the k th natural basis vector.

$$\mathcal{P}_{\hat{\mathbf{n}}}|_{\mathcal{A}} = \begin{pmatrix} (\mathcal{P}_{\hat{\mathbf{n}}})_{11} & (\mathcal{P}_{\hat{\mathbf{n}}})_{12} \\ (\mathcal{P}_{\hat{\mathbf{n}}})_{21} & (\mathcal{P}_{\hat{\mathbf{n}}})_{22} \end{pmatrix} = \begin{pmatrix} \langle \hat{\mathbf{e}}_1 | \mathcal{P}_{\hat{\mathbf{n}}}(\hat{\mathbf{e}}_1) \rangle & \langle \hat{\mathbf{e}}_1 | \mathcal{P}_{\hat{\mathbf{n}}}(\hat{\mathbf{e}}_2) \rangle \\ \langle \hat{\mathbf{e}}_2 | \mathcal{P}_{\hat{\mathbf{n}}}(\hat{\mathbf{e}}_1) \rangle & \langle \hat{\mathbf{e}}_2 | \mathcal{P}_{\hat{\mathbf{n}}}(\hat{\mathbf{e}}_2) \rangle \end{pmatrix}.$$

Since

$$\begin{aligned} \mathcal{P}_{\hat{\mathbf{n}}}(\hat{\mathbf{e}}_1) &= \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} & \text{and} \\ \mathcal{P}_{\hat{\mathbf{n}}}(\hat{\mathbf{e}}_2) &= \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} \end{aligned}$$

(**exercise:** prove it), then

$$\mathcal{P}_{\hat{n}}|_{\mathcal{A}} = \begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$$

(**exercise:** prove it).

The \mathcal{B} Matrix. Now use the \mathcal{B} basis.

$$\mathcal{P}_{\hat{n}}|_{\mathcal{B}} = \begin{pmatrix} \langle \hat{\mathbf{b}}_1 | \mathcal{P}_{\hat{n}}(\hat{\mathbf{b}}_1) \rangle & \langle \hat{\mathbf{b}}_1 | \mathcal{P}_{\hat{n}}(\hat{\mathbf{b}}_2) \rangle \\ \langle \hat{\mathbf{b}}_2 | \mathcal{P}_{\hat{n}}(\hat{\mathbf{b}}_1) \rangle & \langle \hat{\mathbf{b}}_2 | \mathcal{P}_{\hat{n}}(\hat{\mathbf{b}}_2) \rangle \end{pmatrix} \Big|_{\mathcal{B}}.$$

Remember, for this to work, both $\hat{\mathbf{b}}_1$ and $\mathcal{P}_{\hat{n}}(\hat{\mathbf{b}}_1)$ have to be expressed in this same basis, and we almost always express them in the \mathcal{B} basis, since that's how we know everything. Now,

$$\begin{aligned} \mathcal{P}_{\hat{n}}(\hat{\mathbf{b}}_1) &= \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} & \text{and} \\ \mathcal{P}_{\hat{n}}(\hat{\mathbf{b}}_2) &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{aligned}$$

(**exercise:** prove it), so

$$\mathcal{P}_{\hat{n}}|_{\mathcal{B}} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \Big|_{\mathcal{B}}$$

(**exercise:** prove it), a very different matrix than $\mathcal{P}_{\hat{n}}|_{\mathcal{A}}$.

[**Exercise.** Using $(-3, 10)^t$ (or a general \mathbf{v}) confirm (prove) that the matrices $\mathcal{P}_{\hat{n}}|_{\mathcal{A}}$ and $\mathcal{P}_{\hat{n}}|_{\mathcal{B}}$ produce output coordinates of the identical intrinsic output vector, $\mathcal{P}_{\hat{n}}(-3, 10)^t$. **Hint:** use an argument similar to the one above.]

5.5 Some Special Linear Transformations for Quantum Mechanics

We will be using two special types of linear transformations frequently in quantum computing, *unitary* and *Hermitian* operators. Each has its special properties that are sometimes easier to state in terms of their matrices. For that, we need to define *matrix adjoints*.

5.5.1 The Adjoint of a Matrix

Given complex $n \times m$ matrix, M , we can form an $m \times n$ matrix, M^\dagger , by taking M 's *conjugate transpose*,

$$\begin{aligned} M^\dagger &\equiv (M^t)^* & \text{or, in terms of components,} \\ (M^\dagger)_{jk} &= M_{kj}^*. \end{aligned}$$

In other words, we form the *transpose* of M , then take the *complex conjugate* of every element in that matrix.

Examples.

$$\begin{pmatrix} 1-i & 3 & \pi \\ 0 & -2.5 & \sqrt{7}+7i \\ 6 & 7i & 88 \end{pmatrix}^\dagger = \begin{pmatrix} 1+i & 0 & 6 \\ 3 & -2.5 & -7i \\ \pi & \sqrt{7}-7i & 88 \end{pmatrix}$$

$$\begin{pmatrix} 1-i & 3+2i & \pi & 99 \\ 5 & -2.5 & \sqrt{7}+7i & e^i \end{pmatrix}^\dagger = \begin{pmatrix} 1+i & 5 \\ 3-2i & -2.5 \\ \pi & \sqrt{7}-7i \\ 99 & e^{-i} \end{pmatrix}$$

When either $m = 1$ or $n = 1$, the matrix is usually viewed as a *vector*. The adjoint operation turns column vectors into row vectors (while also transposing them), and vice versa:

$$\begin{pmatrix} 1+i \\ \sqrt{2}-2i \end{pmatrix}^\dagger = (1-i, \sqrt{2}+2i)$$

$$(3+7i, \sqrt{2})^\dagger = \begin{pmatrix} 3-7i \\ \sqrt{2} \end{pmatrix}$$

The Adjoint of a Linear Transformation

Because linear transformations always have a matrix associated with them (once we have established a basis), we can carry the definition of adjoint easily over to linear transformations.

Given a linear transformation, T with matrix M_T (in some agreed-upon basis), its *adjoint*, T^\dagger , is the linear transformation defined by the matrix (multiplication) M_T^\dagger . It can be stated in terms of its action on an arbitrary vector,

$$T^\dagger(\mathbf{v}) \equiv (M_T)^\dagger \cdot \mathbf{v}, \quad \text{for all } \mathbf{v} \in \mathcal{V}.$$

[Food for Thought. This definition requires that we have a basis established, otherwise we can't get a matrix. But is the adjoint of a linear transformation, in this definition, going to be different for different bases? Hit the CS 83A discussion forums, please.]

5.5.2 Unitary Operators

Informal Definition. A *unitary operator* is a linear transformation that preserves distances.

In a moment, we will make this precise, but first the big news.

Quantum Logic Gates Are Unitary Operators

$$\mathbf{v} \longrightarrow \boxed{U} \longrightarrow U\mathbf{v}$$

Quantum computing uses a much richer class of *logical operators* (or *gates*) than classical computing. Rather than the relatively small set of gates: XOR, AND, NAND, etc., in classical computing, quantum computers have infinitely many different logic gates that can be applied. On the other hand, the logical operators of quantum computing are of a special form not required of classical computing; they must be *unitary*, i.e., *reversible*.

Definition and Examples

Theoretical Definition. A linear transformation, U , is **unitary** (a.k.a. a **unitary operator**) if it preserves inner products, i.e.,

$$\langle U\mathbf{v} | U\mathbf{w} \rangle = \langle \mathbf{v} | \mathbf{w} \rangle ,$$

for all vectors \mathbf{v} , \mathbf{w} .

While this is a statement about the preservation of *inner products*, it has many implications, one of which is that *distances are preserved*, i.e., for all \mathbf{v} , $\|U\mathbf{v}\| = \|\mathbf{v}\|$.

Theorem. The following characterizations of unitary transformations are equivalent:

- i) A linear transformation is **unitary** if it preserves the lengths of vectors: $\|U\mathbf{v}\| = \|\mathbf{v}\|$.
- ii) A linear transformation is **unitary** if it preserves inner products, $\langle U\mathbf{v} | U\mathbf{w} \rangle = \langle \mathbf{v} | \mathbf{w} \rangle$.
- iii) A linear transformation is **unitary** if its matrix (in an orthonormal basis) has orthonormal rows (and columns).

Caution. This theorem is true, in part, because inner products are *positive-definite*, i.e., $\mathbf{v} \neq \mathbf{0} \Rightarrow \|\mathbf{v}\| > 0$ (lengths of non-zero vectors are strictly positive). However, we'll encounter some very important *pseudo* inner products that are not positive definite, and in those cases the three conditions will not be interchangeable. More on that when we introduce the *classical bit* and *quantum bit*, a few lectures hence.

A concrete example that serves as a classic specimen is a rotation, R_θ , in any Euclidean space, real or complex.

Notation. It is common to use the letter, U , rather than T , for a unitary operator (in the absence of a more specific designation, like R_θ).

Because of this theorem, we can use any of the three conditions as the definition of *unitarity*, and for practical reasons, we choose the third.

Practical Definition #1. A linear transformation, U , is *unitary* (a.k.a. a *unitary operator*) if its matrix (in any orthonormal basis) has *orthonormal columns* (or equivalently *orthonormal rows*), i.e., U is unitary \iff for any orthonormal basis,

$$\mathcal{B} = \{\mathbf{b}_k\}_{k=1}^n,$$

the *column vectors* of $U|_{\mathcal{B}}$ satisfy

$$\langle U(\mathbf{b}_j) | U(\mathbf{b}_k) \rangle = \delta_{jk}.$$

Note that we only need to verify this condition for a single orthonormal basis (exercise, below).

The Matrix of a Unitary Operator. A Matrix, M , is called *unitary* if its adjoint is also its inverse, i.e.,

$$M^\dagger M = MM^\dagger = \mathbf{1}.$$

It is easily seen by looking at our practical definition (*condition 3*) of a unitary operator that an operator is unitary \iff its matrix is unitary. This leads to yet another commonly stated, and equivalent, condition of a unitary operator.

Practical Definition #2. A *unitary operator*, U , is one in which

$$U^\dagger U = UU^\dagger = \mathbf{1}.$$

Some Unitary Operators

The 90° Rotations. The $\pi/2$ counter-clockwise rotation in \mathbb{R}^2 ,

$$R_{\pi/2} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix},$$

is unitary, since its matrix in the preferred basis, $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}\}$, is

$$M_{R_{\pi/2}} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

and it is easily seen that the columns are orthonormal.

General θ Rotations. In a previous exercise, you showed that θ counter-clockwise rotation in \mathbb{R}^2 was, in the standard basis,

$$M_{R_\theta} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}.$$

Dotting the two columns, we get

$$\begin{pmatrix} \cos \theta \\ -\sin \theta \end{pmatrix} \cdot \begin{pmatrix} \sin \theta \\ \cos \theta \end{pmatrix} = \cos \theta \sin \theta - \sin \theta \cos \theta = 0,$$

and dotting a column with itself (we'll just do the first column to demonstrate) gives

$$\begin{pmatrix} \cos \theta \\ -\sin \theta \end{pmatrix} \cdot \begin{pmatrix} \cos \theta \\ -\sin \theta \end{pmatrix} = \cos^2 \theta + \sin^2 \theta = 1,$$

so, again, we get orthonormality.

Phase Changes. A transformation which only modifies the arg of each coordinate,

$$\Pi_{\theta,\phi} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} e^{i\theta} x \\ e^{i\phi} y \end{pmatrix},$$

has the matrix

$$\Pi_{\theta,\phi} = \begin{pmatrix} e^{i\theta} & 0 \\ 0 & e^{i\phi} \end{pmatrix}.$$

Because this is a complex matrix, we have to apply the full inner-product machinery, which requires that we not forget to take conjugates. The inner product of the two columns is easy enough,

$$\left\langle \begin{pmatrix} e^{i\theta} \\ 0 \end{pmatrix} \middle| \begin{pmatrix} 0 \\ e^{i\phi} \end{pmatrix} \right\rangle = e^{-i\theta} \cdot 0 + 0 \cdot e^{i\phi} = 0,$$

but do notice that we had to take the complex conjugate of the first vector – even though failing to have done so would have been an error that still gave us the right answer. The inner product of a column with itself (we'll just show the first column) gives

$$\begin{aligned} \left\langle \begin{pmatrix} e^{i\theta} \\ 0 \end{pmatrix} \middle| \begin{pmatrix} e^{i\theta} \\ 0 \end{pmatrix} \right\rangle &= e^{-i\theta} \cdot e^{i\theta} + 0 \cdot 0 \\ &= e^0 + 0 = 1, \end{aligned}$$

and we have orthonormality. Once again, the complex conjugate was essential in the computation.

Some Non-Unitary Operators

Scaling by a Non-Unit Vector. Scaling by a complex (or real) c , with $|c| \neq 1$ has the matrix

$$S_c = \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix},$$

whose columns are orthogonal (do it), but whose column vectors are not unit length, since

$$\left\langle \begin{pmatrix} c \\ 0 \end{pmatrix} \middle| \begin{pmatrix} c \\ 0 \end{pmatrix} \right\rangle = c^* c + 0 \cdot 0 = |c|^2 \neq 1,$$

by construction.

Note. In our projective Hilbert spaces, such transformations don't really exist, since we consider all vectors which differ by a scalar multiple to be the same entity (state). While this example is fine for learning, and true for non-projective Hilbert spaces, it doesn't represent a real operator in quantum mechanics.

A Projection Operator. Another example we saw a moment ago is the projection onto a vector's 1-dimensional subspace in \mathbb{R}^2 (although this will be true of such a projection in \mathbb{R}^n , $n \geq 2$). That was

$$\mathcal{P}_{\hat{\mathbf{n}}}(\mathbf{v}) \equiv (\mathbf{v} \cdot \hat{\mathbf{n}}) \hat{\mathbf{n}}, \quad \text{where } \hat{\mathbf{n}} = \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}.$$

We only need look at $\mathcal{P}_{\hat{\mathbf{n}}}(\mathbf{v})$ in either of the two bases for which we computed its matrices to see that this is not unitary. Those are done above, but you can fill in a small detail:

[**Exercise.** Show that both matrices for this projection operator fail to have orthonormal columns.]

A Couple Loose Ends

[**Exercise.** Prove that, for orthonormal bases \mathcal{A} and \mathcal{B} , if $U|_{\mathcal{A}}$ has orthonormal columns, then $U|_{\mathcal{B}}$ does, too. Thus, the matrix condition for unitarity is independent of orthonormal basis.]

[**Exercise.** Prove that is *not* true that a unitary U will have a matrix with orthonormal columns for *all* bases. Do this by providing a counter-example as follows.

1. Use the rotation in $R_{\pi/2}$ in \mathbb{R}^2 and the non-orthogonal basis

$$\mathcal{D} = \left\{ \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\} = \{ \mathbf{d}_1, \mathbf{d}_2 \}$$

2. Write down the matrix for $R_{\pi/2}|_{\mathcal{D}}$ using the previously developed formula, true for *all* bases,

$$R_{\pi/2}|_{\mathcal{D}} = \left[\left(R_{\pi/2}(\mathbf{d}_1) \right)|_{\mathcal{D}}, \left(R_{\pi/2}(\mathbf{d}_2) \right)|_{\mathcal{D}} \right]$$

whose four components are, of course, unknown to us. We'll express them temporarily as

$$\begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix} \Big|_{\mathcal{D}}.$$

3. Computing the matrix in the previous step requires a little calculation: you *cannot* use the earlier “trick”

$$T_{jk}\Big|_{\mathcal{B}} = \left\langle \hat{\mathbf{b}}_j \left| T(\hat{\mathbf{b}}_k) \right. \right\rangle ,$$

since that formula only applies to an *orthonormal* basis \mathcal{B} , which \mathcal{D} is not. Instead, you must solve some simple equations. To get the matrix for $R_{\pi/2}\Big|_{\mathcal{D}}$, find the column vectors. The first column is $[R_{\pi/2}(2,0)^t]\Big|_{\mathcal{D}}$ (\mathcal{D} -coordinates taken *after* we have applied $R_{\pi/2}$). You can easily confirm that (in natural coordinates),

$$R_{\pi/2} \left[\begin{pmatrix} 2 \\ 0 \end{pmatrix} \Big|_{\mathcal{A}} \right] = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \Big|_{\mathcal{A}} ,$$

by drawing a picture. Now, get the \mathcal{D} -coordinates of this vector by solving the \mathcal{A} -coordinate equation,

$$\begin{pmatrix} 0 \\ 2 \end{pmatrix} = \alpha \begin{pmatrix} 2 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

for α and β .

4. Even though this will immediately tell you that $R_{\pi/2}\Big|_{\mathcal{D}}$ is not orthonormal, go on to get the full matrix by solving for the second column. $(\gamma, \delta)^t$, and showing that neither column is normalized and the two columns are not orthogonal.]

5.5.3 Hermitian Operators

While unitary operators are the linear transformations that can be used to represent *quantum logic gates*, there are types of operators associated with the *measurements* of a quantum state. These are the called *Hermitian operators*.

Preview Characterization. In quantum mechanics, a *Hermitian operator* will be an operator that is associated with some *observable*, that is, something about the system, such as velocity, momentum, spin that we can imagine *measuring*. For quantum computer scientists, the observable will be *state of a quantum bit*.

We will fully explore the connection between *Hermitian operators* and *observables* in the very next lesson (*quantum mechanics*), but right now let’s least see the mathematical definition of a *Hermitian operator*.

Definitions and Examples

Hermitian Matrices. A matrix, M , which is equal to its **adjoint** is called **Hermitian**, i.e.,

M is Hermitian

$$\Longleftrightarrow$$

$$M^\dagger = M.$$

[**Exercise.** Show that the matrices

$$\begin{pmatrix} 1 & 1-i & \pi \\ 1+i & -2.5 & \sqrt{7}+7i \\ \pi & \sqrt{7}-7i & 88 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 & 3 & 0 \\ 0 & 2 & -1 & 0 \\ 3 & -1 & 3 & 0 \\ 0 & 0 & 0 & \pi \end{pmatrix}$$

are *Hermitian*.]

[**Exercise.** Explain why the matrices

$$\begin{pmatrix} 1+i & 0 & 6 \\ 3 & -2.5 & -7i \\ \pi & \sqrt{7}-7i & 88 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 & 3 & 0 \\ 0 & 2 & -1 & 0 \\ 3 & -1 & 3 & 0 \end{pmatrix}$$

are *not Hermitian*.]

[**Exercise.** Explain why a *Hermitian* matrix has *real* elements along its diagonal.]

Hermitian Operators. A **Hermitian operator** is a linear transformation, T , whose matrix, M_T (in any basis) is Hermitian.

The definition of *Hermitian operator* implies that the basis chosen does not matter: either all of T 's matrices in *all* bases will be Hermitian or none will be. This is a fact, but we won't bother proving it.

Hermitian operators play a starring role in quantum mechanics, as we'll see shortly.

5.6 Enter the Quantum World

Well you did it. After a few weeks of intense but rewarding math, you are ready to learn some fully caffeinated quantum mechanics. I'll walk you through it in a single chapter that will occupy us for about a week at which time you'll be fully a certified quantum "mechanic."

Chapter 6

The Experimental Basis of Quantum Computing

6.1 The Physical Underpinning for Spin 1/2 Quantum Mechanics

This is the first of a three-chapter lesson on quantum mechanics. For our purposes in CS 83A, only the second of the three – the next chapter – contains the essential formalism that we’ll be using in our algorithms. However, a light reading of this first chapter will help frame the theory that comes next.

6.2 Physical Systems and Measurements

6.2.1 Quantum Mechanics as a Model of Reality

Quantum mechanics is not quantum physics. Rather, it is the collection of mathematical tools used to analyze physical systems which are, to the best of anyone’s ability to test, known to behave according to the laws of quantum physics.

As computer scientists, we will not concern ourselves with how the engineers implement the physical hardware that exhibits predictable quantum behavior (any more than we needed to know how they constructed a single classical bit capable of holding a 1 or a 0 out of beach sand in order to write classical software or algorithms). Certainly this is of great interest, but it won’t interfere with our ability to play our part in moving the field of quantum information forward. As of this writing, we don’t know which engineering efforts will result in the first generation of true quantum hardware, but our algorithms should work regardless of the specific solution they end up dropping at our doorstep.

That said, a brief overview of the physics will provide some stabilizing terra firma to which the math can be moored.

6.2.2 The Physical System, \mathcal{S}

Here's the set-up. We have a physical system, call it \mathcal{S} (that's a script "S"), and an apparatus that can measure some property of \mathcal{S} . Also, we have it on good authority that \mathcal{S} behaves according to quantum weirdness; 100 years of experimentation has confirmed certain things about the behavior of \mathcal{S} and its measurement outcomes. Here are some examples – the last few may be unfamiliar to you, but I will elaborate, shortly.

- The system is a proton and the measurement is the *velocity* (\propto *momentum*) of the proton.
- The system is a proton and the measurement is the *position* of the proton.
- The system is a hydrogen atom (one proton + one electron) and the measurement is the *potential energy* state of the atom.
- The system is an electron and the measurement is the *z-component magnitude* of the electron's *spin*.
- The system is an electron and the measurement is the *x-component magnitude* of the electron's *spin*.
- The system is an electron and the measurement is the magnitude of the electron's *spin projected onto the direction* \hat{n} .

In each case, we are measuring a real number that our apparatus somehow is capable of detecting. In practice, the apparatus is usually measuring something *related* to our desired quantity, and we follow that with a computation to get the value of interest (velocity, momentum, energy, z-component of spin, etc.).

6.2.3 Electron Spin as a Testbed for Quantum Mechanics

One usually studies *momentum* and *position* as the relevant measurable quantities, especially in a first course in quantum mechanics. For us, an electron's *spin* will be a better choice.

A Reason to Learn Quantum Mechanics Through Spin

The Hilbert spaces that we get if we measure momentum or position are infinite dimensional vector spaces, and the corresponding linear combinations become integrals rather than sums. We prefer to avoid calculus in this course. For spin, however, our vector spaces are two dimensional – about as simple as they come. Sums work great.

A Reason Computer Scientists Need to Understand Spin

The spin of an electron, which is known as a spin $1/2$ particle, has exactly the kind of property that we can incorporate into our algorithms. It will have classical aspects that allow it to be viewed as a classical bit (0 or 1), yet it still has quantum-mechanical aspects that enable us process it while it is in an “in-between” state – a *mixture* of 0 and 1.

Spin Offers a Minor Challenge

Unlike momentum or position, spin is not part of common vernacular, so we have to spend a moment defining it (in a way that will necessarily be incomplete and inaccurate since we don’t have a full 12 weeks to devote to the electromagnetism, classical mechanics and quantum physics required to do it justice).

6.3 A Classical Attempt at Spin $1/2$ Physics

6.3.1 An Imperfect Picture of Spin

The Scalar and Vector Sides of Spin

Spin is a property that every electron possesses. Some properties like *charge* and *mass* are the same for all electrons, while others like *position* and *momentum* vary depending on the electron in question and the exact moment at which we measure. The spin – or more accurate term *spin state* – of an electron has aspects of both. There is an *overall magnitude* associated with an electron’s spin state that does not change. It is represented by the number $1/2$, a value shared by all electrons at all times. But then each electron can have its own unique *vector orientation* that varies from electron-to-electron or moment-to-moment. We’ll sneak up on the true quantum definition of these two aspects of quantum spin in steps by first by thinking of an electron using inaccurate but intuitive imagery, and we’ll make adjustments as we perform experiments that progressively force us to change our attitude.

Electromagnetic Fields’ Use in Learning About Spin States

Electrons appear to interact with external electromagnetic fields *as if* they were tiny, negatively charged masses *rotating at a constant speed about an internal axis*. While they truly *are* negatively charged and massive, no actual rotation takes place. Nevertheless, the math resulting from this imagery can be used as a starting point to make predictions about how they will behave in such a field. The predictions won’t be right initially, but they’ll lead us to better models.

A Useful Mental Image of Spin

We indulge our desire to apply classical physics and – with an understanding that it’s not necessarily true – consider the electron to be a rotating, charged body. Such an assumption would imbue every electron with an *intrinsic angular momentum* that we call *spin*. If you have not studied basic physics, you can imagine a spinning top; it has a certain *mass distribution*, spins at a certain rate (*frequency*) and its rotational axis has a certain *direction*. Combining these three things into a single vector, we end up defining the *angular momentum* of the top or, in our case, the *spin* of the electron. (See Figure 6.1.)

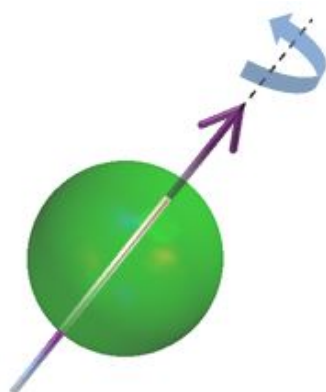


Figure 6.1: Classical angular momentum

6.3.2 A Naive Quantitative Definition of Electron Spin

Translating our imperfect model into the language of math, we initially define an electron’s spin (state) to be the *vector* \mathbf{S} which embodies two ideas,

1. first, its *quantity* of angular momentum (how “heavy” it is combined with how fast it’s rotating), which I will call S , and
2. second, the *orientation* (or *direction*) of its imagined rotational axis, which I will call $\hat{\mathbf{n}}_S$ or sometimes just $\hat{\mathbf{S}}$.

The first entity, S , is a *scalar*. The second, $\hat{\mathbf{n}}_S$, can be represented by a *unit vector* that points in the direction of the rotational axis (where we adjudicate up vs. down by a “right hand rule,” which I will let you recall from any one of your early math classes). (See Figure 6.2.)

So, the total spin vector will be written

$$\mathbf{S} = \begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix},$$

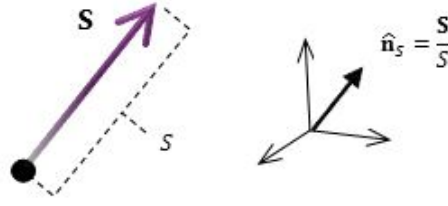


Figure 6.2: A classical idea for spin: A 3-D direction and a scalar magnitude

and we can break it into the two aspects, its scalar magnitude,

$$S \equiv |\mathbf{S}| = \sqrt{S_x^2 + S_y^2 + S_z^2},$$

and a unit vector that embodies only its orientation (direction)

$$\hat{\mathbf{n}}_S = \hat{\mathbf{S}} \equiv \frac{\mathbf{S}}{|\mathbf{S}|}.$$

S , the spin magnitude, is the same for all electrons under all conditions. For the record, its value is $\frac{\sqrt{3}}{2}\hbar$, where \hbar is a tiny number known as Plank's constant. (In street terminology that's "spin 1/2.") After today, we won't rely on the exact expression, but for now we will keep it on the books by making explicit the relationship

$$\mathbf{S} = \left(\frac{\sqrt{3}}{2}\hbar\right) \hat{\mathbf{n}}_S.$$

The constancy of its magnitude leaves the electron's spin *orientation*, $\hat{\mathbf{S}}$, as the only spin-related entity that can change from moment-to-moment or electron-to-electron.

6.3.3 Spherical Representation

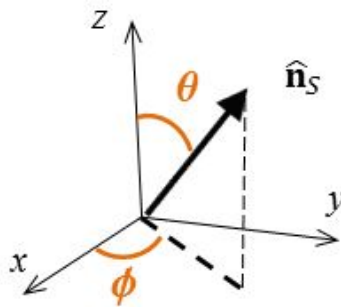


Figure 6.3: Polar and azimuthal angles for the (unit) spin direction

You may have noticed that we don't need all three components n_x , n_y and n_z . Since $\hat{\mathbf{n}}_S$ is unit length, the third can be derived from the other two. [**Exercise.** How?] A common way to express spin direction using only *two* real numbers is through the so-called *polar* and *azimuthal* angles, θ and ϕ (See Figure 6.3).

In fact, *Spherical coordinates* provide an alternate means of expressing *any* vector using these two angles plus the vector’s length, r . In this language a vector can be written $(r, \theta, \phi)_{\text{Sph}}$, rather than the usual (x, y, z) . (The subscript “Sph” is usually not shown if the context makes clear we are using spherical coordinates.) For example,

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \sqrt{3} \\ .615 \\ \pi/4 \end{pmatrix}_{\text{Sph}}.$$

In the language of spherical coordinates the vector $\hat{\mathbf{n}}_S$ is written $(1, \theta, \phi)_{\text{Sph}}$, where the first coordinate is always 1 because $\hat{\mathbf{n}}_S$ has unit length. The two remaining coordinates are the ones we just defined, the angles depicted in Figure 6.3.

θ and ϕ will be important alternatives to the Euclidean coordinates (n_x, n_y, n_z) , especially as we study the *Bloch sphere*, *density matrices* and *mixed states*, topics in the next quantum computing course, *CS 83B*.

6.4 Refining Our Model: Experiment #1

We proceed along classical grounds and, with the aid of some experimental physicists, design an experiment.

6.4.1 The Experiment

We prepare a bunch of electrons in equiprobable random states, take some measurements of their spin states and compare the result with what we would expect classically. Here are the details:

1. **The States.** Let’s instruct our experimental physicists to prepare an electron soup: billions upon billions of electrons in completely random spin orientations. No one direction (or range of directions) should be more represented than any other.



Figure 6.4: A soup of electrons with randomly oriented spins

2. **The Measurement.** Even classically there is no obvious way to measure the entire 3-D vector \mathbf{S} at once; we have to detect the three scalar components,

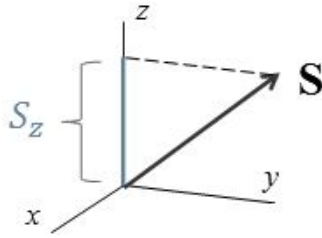


Figure 6.5: The z -projection of one electron's spin

individually. Therefore, we'll ask the physicists to measure only the real valued component S_z , the projection of \mathbf{S} onto the z -axis. They assure us they can do this without subjecting the electrons to any net forces that would (classically) modify the z -component of \mathbf{S} .

To aid the visualization, we imagine measuring each electron one-at-a-time and noting the z -component of spin after each "trial." (See Figure 6.5.)

3. **The Classical Expectation.** The results are easy to predict classically since the length of the spin is fixed at $\frac{\sqrt{3}}{2}\hbar$, and the electrons are oriented randomly; we expect S_z to vary between $+\frac{\sqrt{3}}{2}\hbar$ and $-\frac{\sqrt{3}}{2}\hbar$. For example, in one extreme case, we could find

↑ an electron whose spin is oriented "straight up," i.e., in the positive z -direction with $S_z = +\frac{\sqrt{3}}{2}\hbar$ possessing 100% of the vector's length.

Similar logic implies the detection of other electrons

- ↓ pointing straight down ($S_z = -\frac{\sqrt{3}}{2}\hbar$),
- lying entirely in the x - y plane ($S_z = 0$),
- ↗ pointing mostly up ($S_z = +.8\hbar$),
- ↘ slightly down ($S_z = -.1\hbar$),

etc.

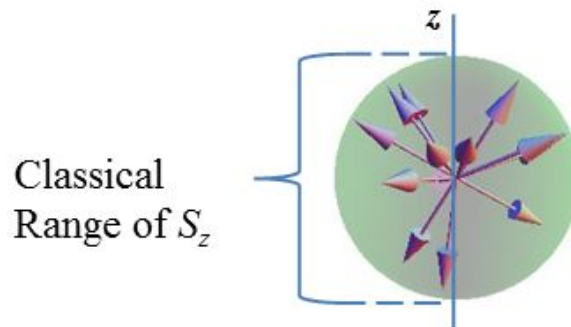


Figure 6.6: The Classical range of z -projection of spin

6.4.2 The Actual Results

We get no such cooperation from nature. In fact, we only – and always – get one of two z -values of spin: $S_z = +\hbar/2$ and $S_z = -\hbar/2$. Furthermore, the two readings appear to be somewhat random, occurring with about equal likelihood and no pattern:

$$\left(-\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots$$

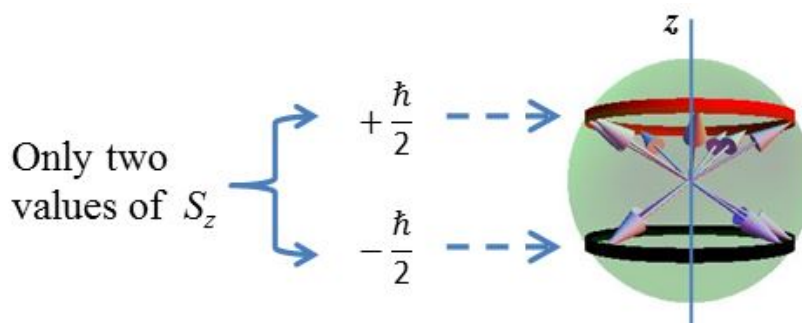


Figure 6.7: The measurements force S_z to “snap” into one of two values.

6.4.3 The Quantum Reality

There are two surprises here, each revealing its own quantum truth that we will be forced to accept.

- **Surprise #1.** There are infinitely many quantum spin states available for electrons to secretly experience. Yet when we measure the z -component of spin, the uncooperative particle always reports that this value is either $(+\frac{\hbar}{2})$ or $(-\frac{\hbar}{2})$, each choice occurring with about equal likelihood. We will call the z -component of spin the *observable* S_z (“*observable*” indicating that we can measure it) and accept from the vast body of experimental evidence that measuring the *observable* S_z forces the spin state to *collapse* such that its S_z “snaps” to either one of the two allowable values called *eigenvalues* of the *observable* S_z . We’ll call the $(+\frac{\hbar}{2})$ outcome the $+z$ outcome (or just the $(+)$ outcome) and the $(-\frac{\hbar}{2})$ outcome the $-z$ (or the $(-)$) outcome.
- **Surprise #2.** Even if we can somehow accept the collapse of the infinity of random states into the two measurable ones, we cannot help but wonder why the electron’s projection onto the z -axis is not the entire length of the vector, that is, either straight up at $(+\frac{\sqrt{3}}{2})\hbar$ or straight down at $(-\frac{\sqrt{3}}{2})\hbar$. The electron stubbornly wants to give us only a fraction of that amount, $\approx 58\%$. This corresponds to two groups. The “up group” which forms the angle $\theta \approx 55^\circ$

(0.955 rad) with the *positive* z -axis and “down group” which forms that same 55° angle with the *negative* z -axis. The explanation for not being able to get a measurement that has the full length $\left(\frac{\sqrt{3}}{2}\right)\hbar$ is hard to describe without a more complete study of quantum mechanics. Briefly it is due to the *Heisenberg uncertainty principle*. If the spin were to collapse to a state that was any closer to the vertical $\pm z$ -axis, we would have too much simultaneous knowledge about its x - and y - components (too close to 0) and its z -component (too close to $\left(\pm\frac{\sqrt{3}}{2}\right)\hbar$). (See Figure 6.8.) This would violate *Heisenberg*, which requires the combined *variation* of these observables be larger than a fixed constant. Therefore, S_z must give up some of its claim on the full spin magnitude, $\left(\frac{\sqrt{3}}{2}\right)\hbar$,

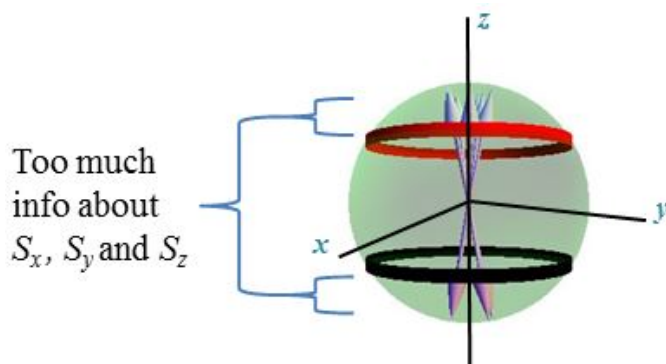


Figure 6.8: Near “vertical” spin measurements give illegally accurate knowledge of S_x , S_y and S_z .

resulting in a shorter S_z , specifically $\frac{1}{2}\hbar$.

6.4.4 A Follow-Up to Experiment #1

Before we move on to a new experimental set-up, we do one more test. The physicists tell us that a side-effect of the experimental design was that the electrons which measured a $+z$ are now separated from those that measured $-z$, and without building any new equipment, we can subject each group to a follow-up S_z measurement. The follow-up test, like the original, will not add any forces or torques that could change the S_z (although we are now beginning to wonder, given the results of the first set of trials). (See Figure 6.9.)

6.4.5 Results of the Follow-Up

We are reassured by the results. All of the electrons in the $(+)$ group faithfully measure $+z$, whether done immediately after the first test or after a period of time. Similarly, the S_z value of the $(-)$ group is unchanged; those rascals continue in their down- z orientation. We have apparently created two special states for the electrons which are distinguishable and verifiable. (See Figure 6.10.)

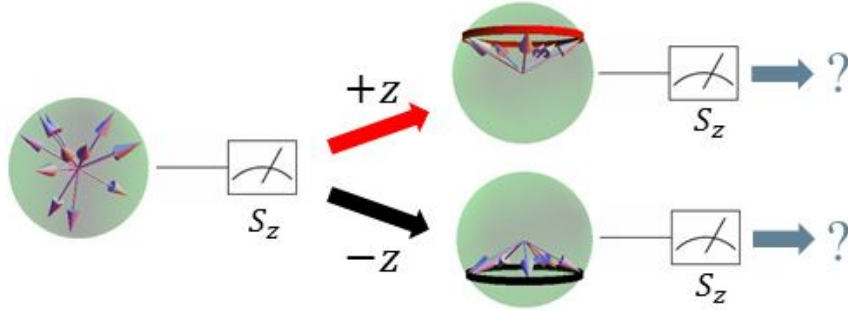


Figure 6.9: Plans for a follow-up to experiment # 1

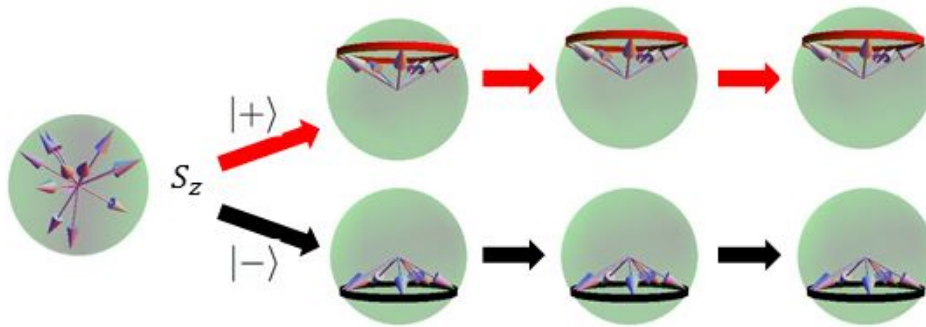


Figure 6.10: Results of follow-up measurements of S_z on the $+z$ ($|+\rangle$) and $-z$ ($|-\rangle$) groups.

$|+\rangle$ and $|-\rangle$ States. We give a name to the state of the electrons in the $(+)$ group: we call it the $|+\rangle_z$ state (or simply the $|+\rangle$ state, since we consider the z -axis to be the preferred axis in which to project the spin). We say that the $(-)$ group is in the $|-\rangle_z$ (or just the $|-\rangle$) state. Verbally, these two states are pronounced “plus ket” and “minus ket.”

6.4.6 Quantum Mechanics Lesson #1

Either by intuition or a careful study of the experimental record, we believe there are infinitely many spin values represented by the electrons in our random electron soup prepared by the physicists, yet when we measure the S_z component, each state *collapses* into one of two possible outcomes. Repeated testing of S_z after the first collapse, confirms that the collapsed state is *stable*: we have prepared the electrons in a special spin-up or spin-down state, $|+\rangle$ or $|-\rangle$.

$|+\rangle_x$ and $|-\rangle_x$ States. Certainly this would also be true had we projected the spin vector onto the x -axis (measuring S_x) or the y -axis (measuring S_y). For example, if we had tested S_x , we would have collapsed the electrons into a different pair of states, $|+\rangle_x$ and $|-\rangle_x$ (this time, we need the subscript x to distinguish it from our preferred z -axis states $\{|+\rangle, |-\rangle\}$).

What’s the relationship between the two z -states, $\{|+\rangle, |-\rangle\}$ and the x -states,

$\{|+\rangle_x, |-\rangle_x\}$? We'll need another experiment.

6.4.7 First Adjustment to the Model

The imagery of a spinning charged massive object has to be abandoned. Either our classical physics doesn't work at this subatomic level or the mechanism of electron spin is not what we thought it was – or both. Otherwise, we would have observed a continuum of values S_z values in our measurements. Yet the experiments that we have been doing are designed to measure spin *as if it were a 3-dimensional vector*, and the electrons are responding to something in our apparatus during the experiments, so while the model isn't completely accurate, the representation of spin by a vector, \mathbf{S} (or equivalently, two angular parameters, θ and ϕ , and one scalar S) may still have some merit. We're going to stop pretending that anything is really spinning, but we'll stick with the vector quantity \mathbf{S} and its magnitude S , just in case there's some useful information in there.

6.5 Refining Our Model: Experiment #2

The *output* of our first experiment will serve as the *input* of our second. (See Figure 6.11.)

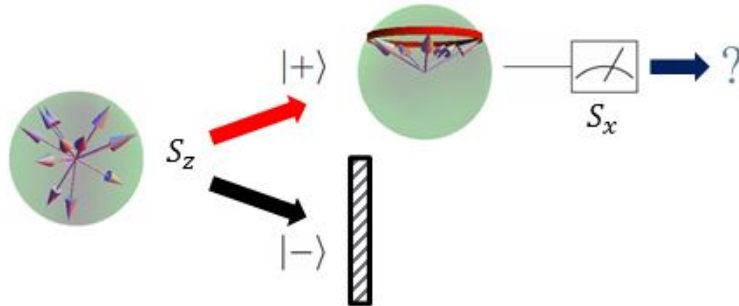


Figure 6.11: Experiment #2: $|+\rangle_z$ electrons enter an S_x measurement apparatus.

6.5.1 The Experiment

We will take only the electrons that collapsed to $|+\rangle$ as input to a new apparatus. We observed that this was essentially half of the original group, the other half consisting of the electrons that collapsed into the state $|-\rangle$ (which we now throw away).

Our second apparatus is going to measure the x -axis projection. Let's repeat this using the same bullet-points as we did for the first experiment.

1. **The States.** The input electrons are in a specific state, $|+\rangle$, whose z -spins always point (as close as possible to) up. This is in contrast to the first experiment where the electrons were randomly oriented.

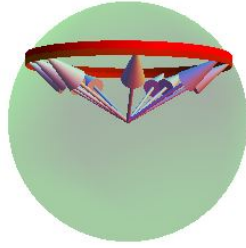


Figure 6.12: The input state for experiment # 2

2. **The Measurement.** Instead of measuring S_z , we measure S_x . We are curious

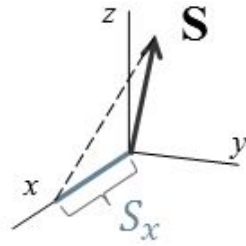


Figure 6.13: The x -projection of spin

to know whether electrons that start in the z -up state have any x -projection preference. We direct our physicists to measure only the real valued component S_x , the projection of \mathbf{S} onto the x -axis.

3. **The Classical Expectation.**

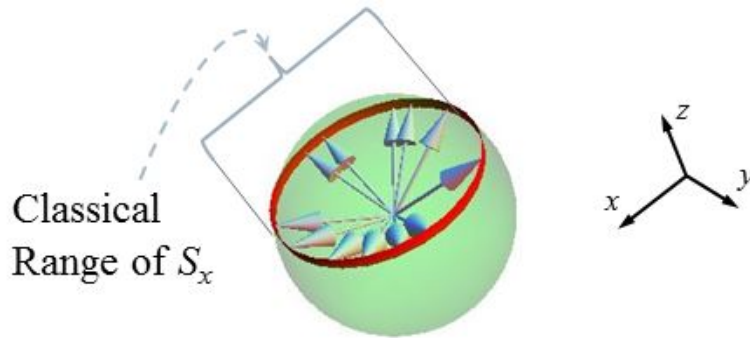


Figure 6.14: Viewed from top left, the classical range of x -projection of spin.

Clinging desperately to those classical ideas which have not been ruled out by the first experiment, we imagine S_x and S_y to be in any relative amounts that complement $|S_z|$, now firmly fixed at $\frac{\hbar}{2}$. This would allow values for those two such that $\sqrt{|S_x|^2 + |S_y|^2 + \frac{1}{4}\hbar^2} = \frac{\sqrt{3}}{2}\hbar$. If true, prior to this second measurement S_x would be smeared over a range of values. (See Figure 6.14.)

6.5.2 The Actual Results

As before, our classical expectations are dashed. We get one of two x -values of spin: $S_x = +\hbar/2$ or $S_x = -\hbar/2$. And again the two readings occur randomly with near equal probability:

$$\left(+\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots$$

Also, when we subject each output group to further S_x tests, we find that after the first S_x collapse each group is locked in its own state – *as long as we only test S_x* .

6.5.3 The Quantum Reality

Our first experiment prepared us to expect the “non-classical.” Apparently, when an electron starts out in a $|+\rangle_z$ spin state, an S_x measurement causes it to *collapse* into a state such that we only “see” one of two allowable *eigenvalues* of the *observable* S_x , namely $+\frac{\hbar}{2}$ (the (+) outcome) and $-\frac{\hbar}{2}$ (the (–) outcome).

Despite the quantum results, we believe we have at least isolated two groups of electrons, *both* in the $|+\rangle$ (z -up) state, but *one* in the $|-\rangle_x$ (x -“down”) state and *the other* in the $|+\rangle_x$ (x -“up”). (See Figure 6.15.) However, we had better test this belief.

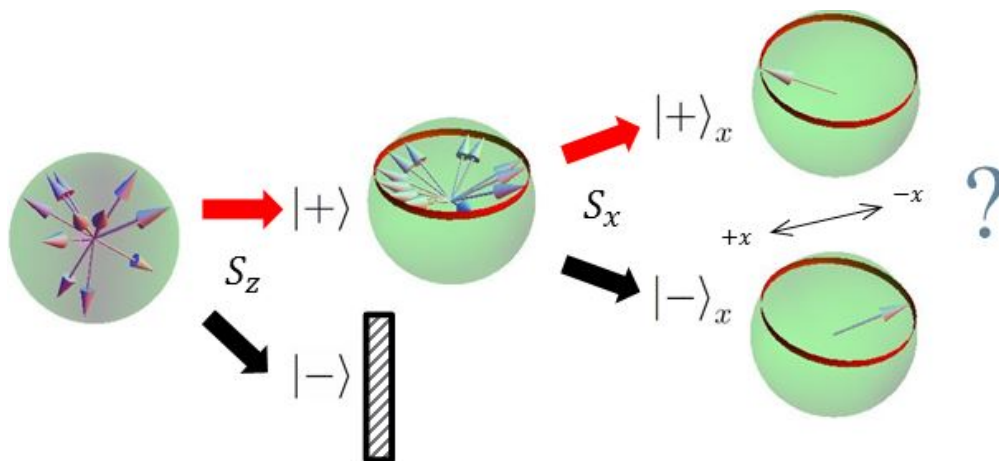


Figure 6.15: A guess about the states of two groups after experiment #2

6.5.4 A Follow-Up to Experiment #2

Using the two detectors the physicists built, we hone in on one of the two output groups of experiment, #2, choosing (for variety) the $|-\rangle_x$ group. We believe it to contain only electrons with $S_z = (+)$ and $S_x = (-)$. In case you are reading too fast, that’s z -up and x -down. To confirm, we run this group through a third test that measures S_z again, fully anticipating a boring run of (+) readings.

6.5.5 Results of the Follow-Up

Here is what we find:

$$\left(+\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots$$

We are speechless. There are now equal numbers of z -up and z -down spins in a group of electrons that we initially selected from a *purely* z -up pool. (See Figure 6.16.) Furthermore, the physicists assure us that nothing in the second apparatus could

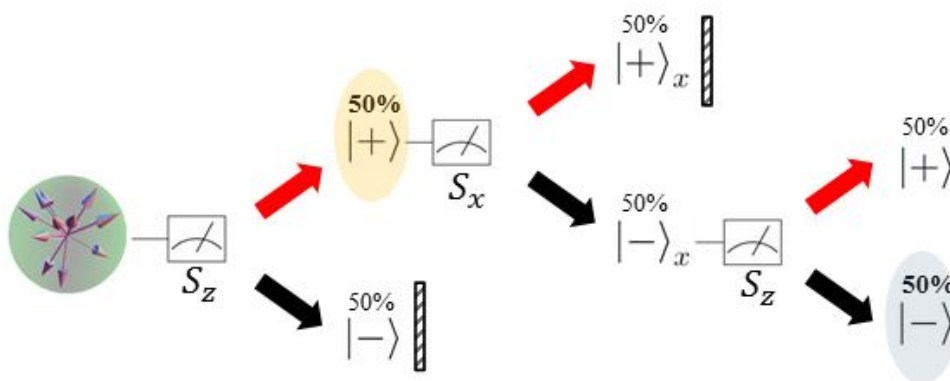


Figure 6.16: $|-\rangle$ electrons emerging from a group of $|+\rangle$ electrons

have “turned the electrons” viz-a-viz the z -component – there were no net z -direction forces.

6.5.6 Quantum Mechanics Lesson #2

Even if we accept the crudeness of only two measurable states per observable, we still can’t force both S_z and S_x into specific states at the same time. Measuring S_x destroys all information about S_z . (See Figure 6.17.)

After more experimentation, we find this to be true of any pair of distinct directions, S_x , S_y , S_z , or even $S_{\hat{n}}$, where \hat{n} is some arbitrarily direction onto which we project \mathbf{S} . There is no such thing as knowledge of three independent directional components such as S_x , S_y , and S_z . The theorists will say that S_z and S_x (and any such pair of different directions) are *incompatible observables*. Even if we lower our expectations to accept only discrete (in fact binary) outcomes $|+\rangle$, $|+\rangle_x$, $|-\rangle_y$, etc. we still can’t know or prepare two at once.

[Exception. If we select for $|+\rangle_x$ (or any $|+\rangle_{\hat{n}}$) followed by a selection for its polar opposite, $|-\rangle_x$ (or $|-\rangle_{\hat{n}}$), there will be predictably zero electrons in the final output.]

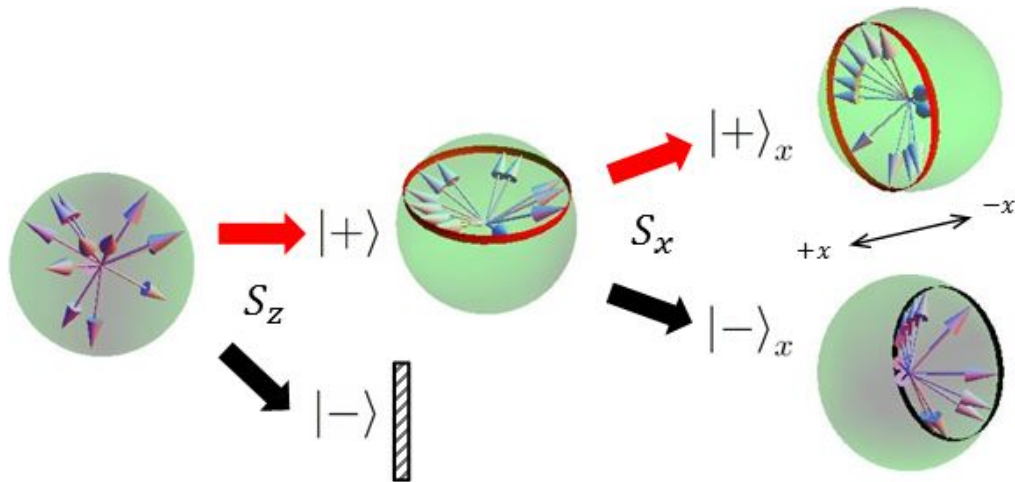


Figure 6.17: The destruction of $|+\rangle$ S_z information after measuring S_x

Preview

As we begin to construct a two-dimensional Hilbert space to represent spin, we already have a clue about what all this might mean. Since, the $|-\rangle_x$ output of the second apparatus was found (by the third apparatus) to contain portions of both $|+\rangle$ and $|-\rangle$, it might be expressible by the equation

$$|-\rangle_x = c_+ |+\rangle + c_- |-\rangle,$$

where c_+ and c_- are scalar “weights” which express how much $|+\rangle$ and $|-\rangle$ constitute $|-\rangle_x$. Furthermore, it would make sense that the scalars c_+ and c_- have equal magnitude if they are to reflect the observed (roughly) equal number of z -up and z -down spins detected by the third apparatus when testing the $|-\rangle_x$ group.

We can push this even further. Because we are going to be working with *normalized* vectors (recall *the projective sphere* in Hilbert space?), it will turn out that their common magnitude will be $1/\sqrt{2}$.

For this particular combination of vector states,

$$|-\rangle_x = \frac{1}{\sqrt{2}} |+\rangle - \frac{1}{\sqrt{2}} |-\rangle = \frac{|+\rangle - |-\rangle}{\sqrt{2}}.$$

In words (that we shall make precise in a few moments), the $|-\rangle_x$ vector can be expressed as a linear combination of the S_z vectors $|+\rangle$ and $|-\rangle$. This hints at the idea that $|+\rangle$ and $|-\rangle$ form a *basis* for a very simple 2-dimensional Hilbert space, the foundation of all quantum computing.

6.5.7 Second Adjustment to the Model

Every time we measure one of the three scalar coordinates S_x , S_y , or S_z , information we thought we had about the other two seems to vanish. The three dimensional vector

S is crumbling before our eyes. In its place, a new model is emerging – that of a *two dimensional* vector space whose two basis vectors appear to be the two z -spin states $|+\rangle$ and $|-\rangle$ which represent a quantum z -up and quantum z -down, respectively. This is a difficult transition to make, and I’m asking you to accept the concept without trying too hard to visualize it in your normal three-dimensional world view. Here are three counter-intuitive ideas, the seeds of which are present in the recent outcomes of experiment #2 and its follow-up:

1. Rather than electron spin being modeled by classical *three dimensional unit vectors*

$$\begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix} = \begin{pmatrix} 1 \\ \theta \\ \phi \end{pmatrix}_{\text{Sph}}$$

in a *real* vector space with basis $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$, we are heading toward a model where spin states are represented by *two dimensional unit vectors*

$$\begin{pmatrix} c_+ \\ c_- \end{pmatrix}$$

in a *complex* vector space with basis $\{|+\rangle_z, |-\rangle_z\}$.

2. In contrast to classical spin, where the unit vectors with z -components +1 and -1 are merely scalar multiples of the *same basis vector* $\hat{\mathbf{z}}$,

$$\begin{pmatrix} 0 \\ 0 \\ +1 \end{pmatrix} = (+1)\hat{\mathbf{z}} \quad \text{and} \quad \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = (-1)\hat{\mathbf{z}}$$

we are positing a model in which the two polar opposite z -spin states, $|+\rangle = |+\rangle_z$ and $|-\rangle = |-\rangle_z$, are *linearly independent* of one another.

3. In even starker contrast to classical spin, where the unit vector $\hat{\mathbf{x}}$ is linearly independent of $\hat{\mathbf{z}}$, the experiments seem to suggest that unit vector $|-\rangle_x$ can be formed by taking a linear combination of $|+\rangle$ and $|-\rangle$, specifically

$$|-\rangle_x = \frac{|+\rangle - |-\rangle}{\sqrt{2}}.$$

But don’t give up on the spherical coordinates θ and ϕ just yet. They have a role to play, and when we study *expectation values* and the *Bloch sphere*, you’ll see what that role is. Meanwhile, we have one more experiment to perform.

6.6 Refining Our Model: Experiment #3

6.6.1 What we Know

We have learned that measuring one of the scalars related to spin, S_z , S_x or any $S_{\hat{\mathbf{n}}}$, has two effects:

1. It collapses the electrons into one of two spin states for that observable, one that produces a reading of up $(+) = \left(+\frac{\hbar}{2}\right)$, and the other that produced a reading of down $(-) = \left(-\frac{\hbar}{2}\right)$.
2. It destroys any information about the other spin axes, or in quantum-ese, about the other spin observables.

(“Up/down” might be better stated as “left/right” when referring to x -coordinates, and “forward-backward” when referring to y -coordinates, but it’s safer to use the consistent terminology of *spin-up/spin-down* for any projection axis, not just the z -axis.)

There is something even more subtle we can learn, and this final experiment will tease that extra information out of the stubborn states.

6.6.2 The Experiment

We can prepare a pure state that is *between* $|+\rangle$ and $|-\rangle$ if we are clever. We direct our physicists to *rotate* the first apparatus *relative to the second apparatus* by an angle θ counter-clockwise from the z -axis (axis of rotation being the y -axis, which ensures that the x - z plane is rotated into itself (see Figure 6.18).

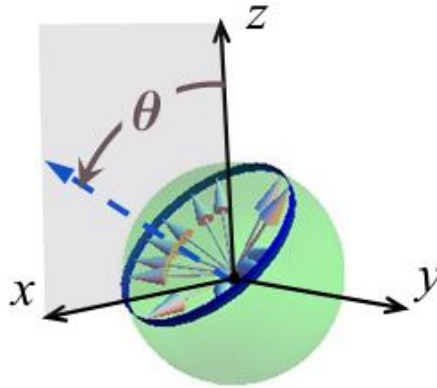


Figure 6.18: A spin direction with polar angle θ from $+z$, represented by $|\psi\rangle$

Let’s call this rotated state “ $|\psi\rangle$,” just so it has a name that is distinct from $|+\rangle$ and $|-\rangle$.

If we only rotate by a tiny θ , we have a high dose of $|+\rangle$ and a small dose of $|-\rangle$ in our rotated state, $|\psi\rangle$. On the other hand, if we rotate by nearly 180° (π radians), $|\psi\rangle$ would have mostly $|-\rangle$ and very little $|+\rangle$ in it. Before this lesson ends, we’ll prove that the right way to express the relationship between θ and the relative amounts of $|+\rangle$ and $|-\rangle$ contained in $|\psi\rangle$ is

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |+\rangle + \sin\left(\frac{\theta}{2}\right) |-\rangle .$$

By selecting the same (+) group coming out of the first apparatus (but now tilted at an angle θ) as input into the second apparatus, we have effectively changed our input states going into the second apparatus from purely $|+\rangle$ to purely $|\psi\rangle$.

We now measure S_z , the spin projected onto the z -axis. The exact features of what I just described can be stated using the earlier three-bullet format.

1. **The States.** This time the input electrons are in a specific state, $|\psi\rangle$, whose z -spin direction forms an angle θ from the z -axis (and for specificity, whose spherical coordinate for the azimuthal angle, ϕ , is 0). (See Figure 6.19.)

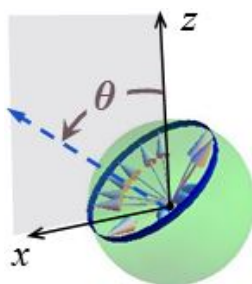


Figure 6.19: The prepared state for experiment #3, prior to measurement

2. **The Measurement.** We follow the preparation of this rotated state with an S_z measurement.
3. **The Classical Expectation.** We've been around the block enough to realize that we shouldn't expect a classical result. If this were a purely classical situation, the spin magnitude, $\frac{\sqrt{3}}{2}\hbar$, would lead to $S_z = \frac{\sqrt{3}}{2}\hbar \cos(\theta)$. But we already know the largest S_z ever "reads" is $\frac{1}{2}\hbar$, so maybe we attenuate *that* number by $\cos \theta$, and predict $\frac{\hbar}{2} \cos(\theta)$. Those are the only two ideas we have at the moment. (See Figure 6.20.)

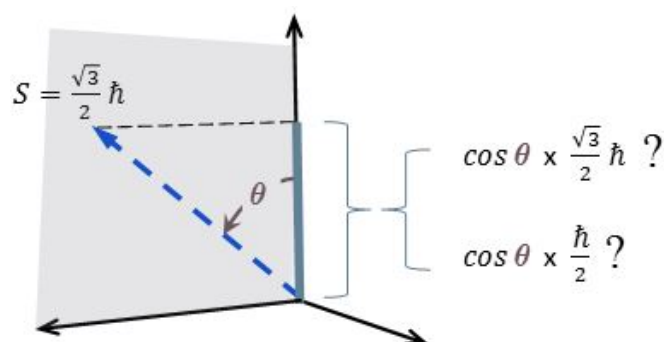


Figure 6.20: Classical, or semi-classical expectation of experiment #3's measurement

6.6.3 The Actual Results

It is perhaps not surprising that we always read one of two z -values of spin: $S_z = +\hbar/2$ and $S_z = -\hbar/2$. The two readings occur somewhat randomly:

$$\left(+\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(+\frac{\hbar}{2}\right) \cdots \left(-\frac{\hbar}{2}\right) \cdots$$

However, closer analysis reveals that they are not equally likely. As we try different θ s and tally the results, we get the summary shown Figure 6.21.

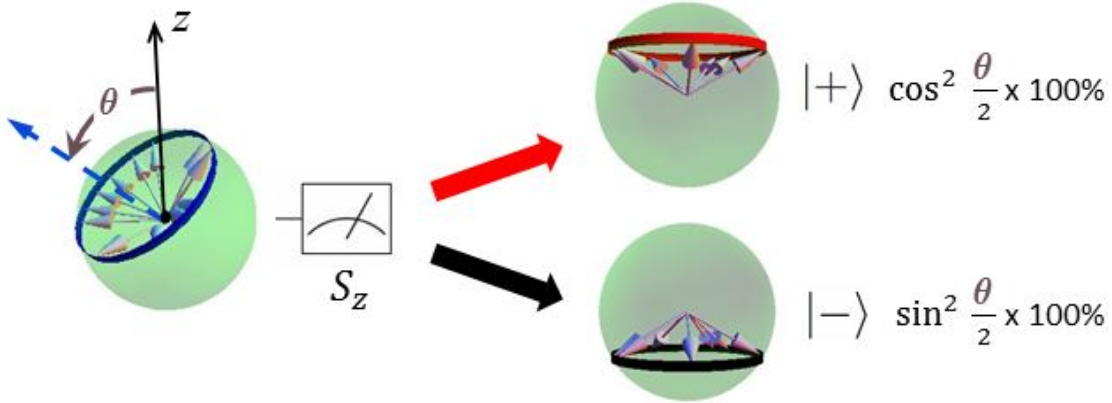


Figure 6.21: Probabilities of measuring $|\pm\rangle$ from starting state $|\psi\rangle$, θ from $+z$

In other words,

$$\begin{aligned} \left[\begin{array}{l} \% \text{ outcomes which} \\ = +\frac{\hbar}{2} \end{array} \right] &\approx \cos^2 \left(\frac{\theta}{2} \right) \cdot 100\% \quad \text{and} \\ \left[\begin{array}{l} \% \text{ outcomes which} \\ = -\frac{\hbar}{2} \end{array} \right] &\approx \sin^2 \left(\frac{\theta}{2} \right) \cdot 100\%. \end{aligned}$$

Notice how nicely this agrees with our discussion of experiment #2. There, we prepared a $|-\rangle_x$ state to go into the final S_z tester. $|+\rangle_x$ is intuitively 90° from the z -axis, so in that experiment our θ was 90° . That would make $\frac{\theta}{2} = 45^\circ$, whose cosine and sine are both $= \frac{\sqrt{2}}{2}$. For these values, the formula above gives a predicted 50% (i.e., equal) frequency to each outcome, (+) and (−), and that's exactly what we found when we measured S_z starting with $|-\rangle_x$ electrons.

6.6.4 The Quantum Reality

Apparently, no matter what quantum state our electron's spin is in, we always measure the magnitude projected onto an axis as $\pm\hbar/2$. We suspected this after the first two experiments, but now firmly believe it. The only thing we *haven't* tried is to rotate the first apparatus into its most general orientation, one that includes a non-zero azimuthal ϕ , but that would not change the results.

This also settles a debate that you might have been waging, mentally. Is spin, prior to an S_z measurement, actually in some combination of the states $|+\rangle$ and $|-\rangle$? *Yes*. Rotating the first apparatus relative to the second apparatus by a particular θ has a physical impact on the outcomes. Even though the electrons collapse into one of $|+\rangle$ and $|-\rangle$ after the measurement, there is a difference between $|\psi\rangle = .45|+\rangle + .89|-\rangle$ and $|\psi'\rangle = .71|+\rangle + .71|-\rangle$: The first produces 20% (+) measurements, and the second produces 50% (+) measurements.

6.6.5 Quantum Mechanics Lesson #3

If we prepare a state to be a normalized (length one) linear combination of the two S_z states, $|+\rangle$ and $|-\rangle$, the two coefficients predict the probability of obtaining each of the two possible eigenvalues $\pm\frac{\hbar}{2}$ for an S_z measurement. This is the meaning of the coefficients of a state when expressed relative to the *preferred* z -basis.

This holds for *any* direction. If we wish to measure the “projection of \mathbf{S} onto an arbitrary direction $\hat{\mathbf{n}}$ ” (in quotes because I’m using slightly inaccurate classical terminology), then we would expand our input state along the $\hat{\mathbf{n}}$ state vectors,

$$|\psi\rangle = \alpha|+\rangle_{\hat{\mathbf{n}}} + \beta|-\rangle_{\hat{\mathbf{n}}} ,$$

and $|\alpha|^2$ and $|\beta|^2$ would give the desired probabilities.

The quantum physicist will tell us that spin of an electron relative to a pre-determined direction can be in any of infinitely many states. However, when we measure, it will collapse into one of two possible states, one up and one down. Furthermore the relative number of the up measurements vs. down measurements is predicted precisely by the magnitude-squared of the coefficients of the prepared state.

Finally, we can use any two polar opposite directions in \mathbb{R}^3 (which we shall learn correspond to two orthonormal vectors in \mathbb{C}^2) as a “basis” for defining the spin state. $\{|+\rangle, |-\rangle\}$, the z -component measurements, are the most common, and have come to be regarded as the preferred basis, but we can use $\{|+\rangle_x, |-\rangle_x\}$, or any two oppositely directed unit vectors (i.e., orientations). Such pairs are considered “alternate bases” for the space of spin states. In fact, as quantum computer scientists, we’ll use these alternate bases frequently.

6.6.6 Third Adjustment to the Model

We’re now close enough to an actual model of quantum spin 1/2 particles to skip the game of making a minor adjustment to our evolving picture. In the next lesson we’ll add the new information about the meaning of the scalar coefficients and the probabilities they reflect and give the full Hilbert space description of spin 1/2 physics motivated by our three experiments.

Also, we’re ready to leave the physical world, at least until we study time evolution. We can thank our experimental physicists and let them get back to their job of designing a quantum computer. We have a model to build and algorithms to design.

6.7 Onward to Formalism

In physics the word “formalism” refers to the abstract mathematical notation necessary to scribble predictions about what a physical system such as a quantum computer will do if we build it. For our purposes, the formalism is what we need to know, and that’s the content of the next two chapters. They will provide a strict but limited set of rules that we can use to accurately understand and design quantum algorithms.

Of those chapters, only the next (the second of these three) is necessary for CS 83A, but what you have just read has prepared you with a sound intuition about the properties and techniques that constitute the formalism of quantum mechanics, especially in the case of the spin $1/2$ model used in quantum computing.

Chapter 7

Time Independent Quantum Mechanics

7.1 Quantum Mechanics for Quantum Computing

This is the second in a three-chapter introduction to quantum mechanics and the only chapter that is required for the first course, *CS 83A*. A brief and light reading of the prior lesson in which we introduce about a half dozen conceptual experiments will help give this chapter context. However, even if you skipped that on first reading, you should still find the following to be self-contained.

Our goal today is twofold.

1. We want you to master the notation used by physicists and computer scientists to scribble, calculate and analyze quantum algorithms and their associated logic circuits.
2. We want you to be able to recognize and make practical use of the direct correspondence between the math and the physical quantum circuitry.

By developing this knowledge, you will learn how manipulating symbols on paper affects the design and analysis of actual algorithms, hardware logic gates and measurements of output registers.

7.2 The Properties of Time Independent Quantum Mechanics

Let's do some quantum mechanics. We'll be stating a series of properties – I'll call them *traits* – that will either be unprovable but experimentally verified *postulates*, or provable *consequences* of those postulates. We'll be citing them all in our study of quantum computing.

In this lesson, *time* will not be a variable; the physics and mathematics pertain to a single instant.

7.2.1 The State Space of a Physical System

Physical System (Again)

Let's define (or *re-define* if you read the last chapter) a physical system \mathcal{S} to be some conceptual or actual apparatus that has a *carefully controlled and limited number of measurable states*. It's implied that there are very few aspects of \mathcal{S} that can change or be measured, since otherwise it would be too chaotic to lend itself to analysis.

Consider two examples. The first is studied early in an undergraduate quantum mechanics course; we won't dwell on it. The second forms the basis of quantum computing, so we'll be giving it considerable play time.

A Particle's Energy and Position in One-Dimension

We build hardware that allows a particle (mass = m) to move freely in *one dimension* between two boundaries, say from 0 cm to 5 cm. The particle can't get out of that 1-dimensional "box" but is otherwise free to roam around inside (no forces acting). We build an apparatus to test the particle's energy. Using elementary quantum mechanics we discover that the particle's energy measurement can only attain certain discrete values, E_0, E_1, E_2, \dots . Furthermore, once we know which energy, E_k , the particle has, we can form a probability curve that predicts the likelihood of finding the particle at various locations within the interval $[0, 5]$.

Spin: A Stern-Gerlach Apparatus

We shoot a single silver atom through a properly oriented inhomogeneous magnetic field. The atom's deflection, up or down, will depend on the atom's overall angular momentum. By placing two or more of these systems in series with various angular orientations, we can study the spin of *the outermost electron* in directions S_z, S_x, S_y or general $S_{\hat{n}}$. While not practical for quantum computation, this apparatus is exactly how the earlier spin-1/2 results were obtained.

[**Optional.**] If you are curious, here are a few details about the Stern-Gerlach apparatus.

- *Silver Atoms Work Well.* You might think this is too complex a system for measuring a single electron's spin. After all, there are 47 electrons and each has spin not to mention a confounding *orbital angular momentum* from its motion about the nucleus. However, orbital angular momentum is net-zero due to statistically random paths, and the spin of the inner 46 electrons cancel (they are paired, one up, one down). This leaves only the spin of the outermost

electron #47 to account for the spin of the atom as a whole. (Two other facts recommend the use of silver atoms. First, we can't use charged particles since the so-called *Lorentz* force would overshadow the subtle spin effects. Second, silver atoms are heavy enough that their deflection can be calculated based solely on classical equations.)

- *We Prepare a Fixed Initial State.* An atom can be prepared in a spin state associated with any pre-determined direction, $\hat{\mathbf{n}}$, prior to subjecting it to a final Stern-Gerlach tester. We do this by selecting a $|+\rangle$ electron from a preliminary Stern-Gerlach S_z tester then orient a second tester in an $\hat{\mathbf{n}}$ direction relative the original.
- *The Measurements and Outcomes.* The deflection of the silver atom is detected as it hits a collector plate at the far end of the last apparatus giving us the measurements $\pm\frac{\hbar}{2}$, and therefore the collapsed states $|+\rangle_x$, $|-\rangle_{\hat{\mathbf{n}}}$, etc. The results correspond precisely with our experiments #1, #2 and #3 discussed earlier.

Stern-Gerlach is the physical system to keep in mind as you study the math that follows.

7.3 The First Postulate of Quantum Mechanics

7.3.1 Trait #1 (The State Space)

For any physical system, \mathcal{S} , we associate a Hilbert space, \mathcal{H} . Each physical state in \mathcal{S} corresponds to some ray in \mathcal{H} , or stated another way, a point on the projective sphere (all unit vectors) of \mathcal{H} .

$$\text{physical state} \in \mathcal{S} \longleftrightarrow \mathbf{v} \in \mathcal{H}, \quad |\mathbf{v}| = 1.$$

The Hilbert space \mathcal{H} is often called the *state space of the system* \mathcal{S} , or just the *state space*.

Vocabulary and Notation. Physicists and quantum computer scientists alike express the vectors in the state space using “ket” notation. This means a vector in state space is written

$$|\psi\rangle,$$

and is usually referred to as a *ket*. The Greek letter ψ is typically used to label any old state. As needed we will be replacing it with specific and individual labels when we want to differentiate two state vectors, express something known about the vector, or discuss a famous vector that is universally labeled. Examples we will encounter include

$$|a\rangle, \quad |u_k\rangle, \quad |+\rangle, \quad \text{and} \quad |+\rangle_y.$$

When studying Hilbert spaces, I mentioned that a single physical state corresponds to an infinite number of vectors, all on the same ray, so we typically choose a “normalized” representative having unit length. It’s the job of quantum physicists to describe how to match the physical states with normalized vectors and ours as quantum computer scientists to understand and respect the correspondence.

7.3.2 The Fundamental State Space for Quantum Computing

Based on our experiments involving spin-1/2 systems from the (optional) previous chapter, we were led to the conclusion that any spin state can be described by a linear combination of two special states $|+\rangle$ and $|-\rangle$. If you skipped that chapter, this will serve to officially define those two states. They are the *natural basis kets* of a 2-dimensional complex Hilbert space, \mathcal{H} . In other words, we construct a simple vector space of complex ordered pairs with the usual complex inner product and decree the natural basis to be the two measurable states $|+\rangle$ and $|-\rangle$ in \mathcal{S} . Symbolically,

$$\begin{aligned}\mathcal{H} &\equiv \left\{ \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \mid \alpha, \beta \in \mathbb{C} \right\} \quad \text{with} \\ |+\rangle &\longleftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \\ |-\rangle &\longleftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} .\end{aligned}$$

In this regime, any physical spin state $|\psi\rangle \in \mathcal{S}$ can be expressed as a *normalized* vector expanded along this natural basis using

$$\begin{aligned}|\psi\rangle &= \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha |+\rangle + \beta |-\rangle , \quad \text{where} \\ |\alpha|^2 + |\beta|^2 &= 1 .\end{aligned}$$

The length requirement reflects that physical states reside on the *projective sphere* of \mathcal{H} .

[**Exercise.** Demonstrate that $\{|+\rangle, |-\rangle\}$ is an orthonormal pair. **Caution:** Even though this may seem trivial, be sure you are using the complex, not the real, inner product.]

The Orthonormality Expressions

In the heat of a big quantum computation, basis kets will kill each other off, turning themselves into the scalars 0 and 1 because the last exercise says that

$$\begin{aligned}\langle + | + \rangle &= \langle - | - \rangle = 1 , \quad \text{and} \\ \langle + | - \rangle &= \langle - | + \rangle = 0 .\end{aligned}$$

While this doesn't rise to the level of "trait", memorize it. Every quantum mechanic relies on it.

[**Exercise.** Demonstrate that the set $\{|+\rangle, |-\rangle\}$ forms a basis (the z -basis) for \mathcal{H} . **Hint:** Even though only the projective sphere models \mathcal{S} , we still have to account for the entire expanse of \mathcal{H} including all the vectors off the unit-sphere if we are going to make claims about "spanning the space."]

The x -Basis for \mathcal{H}

Let's complete a thought we started in our discussion of experiment #2 from last time. We did a little hand-waving to suggest

$$|-\rangle_x = \frac{|+\rangle - |-\rangle}{\sqrt{2}},$$

but now we can make this official (or, if you skipped the last chapter, let this serve as the definition of two new kets).

$$|-\rangle_x \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix},$$

that is, it is the vector in \mathcal{H} whose coordinates along the z -bases are as shown. We may as well define the $|+\rangle_x$ vector. It is

$$|+\rangle_x = \frac{|+\rangle + |-\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

[**Exercise.** Demonstrate that $\{|+\rangle_x, |-\rangle_x\}$ is an orthonormal pair.]

[**Exercise.** Demonstrate that the set $\{|+\rangle_x, |-\rangle_x\}$ forms a basis (the x -basis) for \mathcal{H} .]

7.3.3 Why Does a Projective \mathcal{H} model Spin-1/2?

There may be two questions (at least) that you are inclined to ask.

1. Why do we need a *complex* vector space – as opposed to a *real* one – to represent spin?
2. Why do spin states have to live on the projective sphere? Why not any point in \mathcal{H} or perhaps the sphere of radius 94022?

I can answer item 1 now (and item 2 further down the page). Obviously, there was nothing magical about the z -axis or the x -axis. I could have selected any direction in which to start my experiments at the beginning of the last lesson and then picked any other axis for the second apparatus. In particular, I might have selected the

same z -axis for the first measurement, but used the y -axis for the second one. Our interpretation of the results would then have suggested that $|-\rangle_y$ contain equal parts $|+\rangle$ and $|-\rangle$,

$$|-\rangle_y = c_+ |+\rangle + c_- |-\rangle ,$$

and similarly for $|+\rangle_y$. If we were forced to use real scalars, we would have to pick the same two scalars $\pm \frac{1}{\sqrt{2}}$ for c_{\pm} (although we could choose which got the $+$ and which got the $-$ sign, a meaningless difference). We'd end up with

$$|\pm\rangle_y \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ \pm 1 \end{pmatrix} \quad (\text{warning: not true}),$$

which would force them to be identical to the vectors $|+\rangle_x$ and $|-\rangle_x$, perhaps the order of the vectors, reversed. But this can't be true, since the x -kets and the y -kets can no more be identical to each other than either to the z -kets, and certainly neither are identical to the z -kets. (If they were, then repeated testing would never have split the original $|+\rangle$ into two equal groups, $|+\rangle_x$ and $|-\rangle_x$.) So there are just not enough real numbers to form a *third* pair of basis vectors in the y -direction, distinct from the x -basis and the z -basis.

If we allow complex scalars, the problem goes away. We can define

$$|\pm\rangle_y \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ \pm i \end{pmatrix} ,$$

Now all three pairs are totally different orthonormal bases for \mathcal{H} , yet each one contains “equal amounts” of $|+\rangle$ and $|-\rangle$.

7.4 The Second Postulate of Quantum Mechanics

Trait #1 seemed natural after understanding the experimental outcomes of the spin-1/2 apparatus. **Trait #2** is going to require that we cross a small abstract bridge, but I promise, you have all the math necessary to get to the other side.

7.4.1 Trait #2 (The Operator for an Observable)

An observable quantity \mathcal{A} (in \mathcal{S}) corresponds to an operator – a linear transformation – in \mathcal{H} . The operator for an observable is always Hermitian.

[**Note.** There is a **Trait #2a** that goes along with this, but you don't have the vocabulary yet, so we defer for the time being.]

In case you forgot what *Hermitian* means, the following symbolic version of **Trait #2**, which I'll call **Trait #2'**, should refresh your memory.

Trait #2' (Mathematical Version of Operator for an Observable):

$$\begin{aligned} \text{Observable } \mathcal{A} &\in \mathcal{S} \\ &\longleftrightarrow \\ T_{\mathcal{A}} : \mathcal{H} &\rightarrow \mathcal{H} \text{ linear and } T_{\mathcal{A}}^{\dagger} = T_{\mathcal{A}}. \end{aligned}$$

[**Review.** $T_{\mathcal{A}}^{\dagger} = T_{\mathcal{A}}$ is the *Hermitian condition*. The Hermitian condition on a *matrix* M means that M 's *adjoint* (conjugate transpose), M^{\dagger} , is the same as M . The Hermitian condition on a *linear transformation* means that its matrix (in any basis) is its own adjoint. Thus, **Trait #2'** says that the matrix, $T_{\mathcal{A}}$, which purports to correspond to the observable \mathcal{A} , must be self-adjoint. We'll reinforce all this in the first example.]

Don't feel bad if you didn't expect this kind of definition; nothing we did above suggested that there were linear transformations behind the observables S_z , S_x , etc. – never mind that they had to be *Hermitian*. I'm telling you now, there are and they do. This is an experimentally verified observation, not something we can prove, so it is up to the theorists to guess the operator that corresponds to a specific observable. Also, it is not obvious what we can *do* with the linear transformation of an observable, even if we discover what it is. All in due time.

7.4.2 The Observable S_z

The linear transformation for S_z , associated with the z -component of electron spin (a concept from the optional previous chapter), is represented by the matrix

$$\begin{pmatrix} \frac{\hbar}{2} & 0 \\ 0 & -\frac{\hbar}{2} \end{pmatrix} = \frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Blank stares from the audience.

Never fear. You'll soon understand why this matrix is the “operator” (in \mathcal{H}) chosen to represent the observable S_z (in the physical system \mathcal{S}). And if you did not read that chapter, just take this matrix as the definition of S_z .

Notation. Sometimes we abandon the constant $\hbar/2$ and use a simpler matrix to which we give the name σ_z a.k.a. the *Pauli spin matrix* (in the z -direction),

$$\sigma_z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Meanwhile, the least we can do is to confirm that the matrix for S_z is *Hermitian*.

[**Exercise.** Prove the matrix for S_z is Hermitian.]

We will now start referring to

- i) the *observable* “spin projected onto the z -axis,”

ii) its associated linear *operator*, and

iii) the *matrix* for that operator

all using the one name, S_z .

Which Basis? Everything we do in this chapter – and in the entire the volume – assumes that vectors and matrices are represented in the z -basis, which is the preferred basis that emerges from the S_z observable *unless explicitly stated*. There *are* other bases, but we’ll label things if and when we expand along them.

As for the linear operators associated with the measurements of the other two major spin directions S_x and S_y , they turn out to be

$$\begin{aligned} S_x &\longleftrightarrow \frac{\hbar}{2} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ S_y &\longleftrightarrow \frac{\hbar}{2} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \end{aligned}$$

both represented in terms of the z -basis (even though these operators do define their own bases).

Without the factor of $\hbar/2$ these, too, get the title of *Pauli spin matrices* (in the x - and y -directions, respectively). Collecting the three Pauli spin matrices for reference, we have

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$$

Tip. This also demonstrates a universal truth. Any observable expressed in “its own basis” is always a *diagonal matrix* with the numbers called the “eigenvalues” appearing along that diagonal. All three matrices are displayed in the natural basis – the z -basis. But because the z -basis “comes from” the observable S_z , only *it* takes the form of a diagonal. (Phrases like *eigenvalue*, an *observable’s basis* and a *basis coming from an observable* will be defined shortly.) Since neither S_x nor S_y are associated with the z -basis, their matrices expanded along the z -basis aren’t diagonal. But when we derive the basis coming from S_x , for example, we’ll find that suddenly *its* matrix (but not those of S_y or S_z) expressed in that new basis will be diagonal.

7.5 The Third Postulate of Quantum Mechanics

This trait will add some math vocabulary and a bit of quantum computational skill to our arsenal.

7.5.1 Trait #3 (The Eigenvalues of an Observable)

The only possible measurement outcomes of an observable quantity \mathcal{A} are special real numbers, a_1, a_2, a_3, \dots , associated with the operator’s matrix. The special numbers

a_1, a_2, a_3, \dots are known as **eigenvalues** of the matrix.

[**Note to Physicists.** I'm only considering finite or countable eigenvalues because our study of quantum computing always has only finitely many special a_k . In physics, when measuring position or momentum, the set of eigenvalues is continuous (non-enumerable).]

Obviously, we have to learn what an *eigenvalue* is. Whatever it is, when we compute the eigenvalues of, say, the matrix S_z , this trait tells us that they must be $\pm\hbar/2$, since those were the values we measured when we tested the observable S_z . If we had not already done the experiment but knew that the matrix for the observable was S_z above, **Trait #3** would allow us to predict the possible outcomes, something we will do now.

7.5.2 Eigenvectors and Eigenvalues

Given any complex or real matrix M , we can often find certain special non-zero vectors called *eigenvectors* for M . An eigenvector \mathbf{u} (for M) has the property that

$$\begin{aligned} \mathbf{u} &\neq \mathbf{0} \quad \text{and} \\ M\mathbf{u} &= a\mathbf{u}, \quad \text{for some (possibly complex) scalar } a. \end{aligned}$$

In words, applying M to the non-zero \mathbf{u} results in a scalar multiple of \mathbf{u} . The scalar multiple, a , associated with \mathbf{u} is called an *eigenvalue* of M , while \mathbf{u} is the (an) eigenvector belonging to a . This creates a possible collection of special eigenvector-eigenvalue pairs,

$$\left\{ \mathbf{u}_{a_k} \longleftrightarrow a_k \right\}_{k=1}^{n \text{ OR } \infty}$$

for any matrix M .

Note. There may be more than one eigenvector, $\mathbf{u}'_a, \mathbf{u}''_a, \mathbf{u}'''_a, \dots$, for a given eigenvalue, a , but in this course, we'll typically see the eigenvalues and eigenvectors uniquely paired. More on this, shortly.

[**Exercise.** Give two reasons we require eigenvectors to be non-zero. One should be based purely on the arithmetic of any vector space, and a second should spring from the physical need to use *projective* Hilbert spaces as our quantum system models.

Hint: Both reasons are very simple.]

[**Exercise.** Show that if \mathbf{u} is an eigenvector for M , then any scalar multiple, $\alpha\mathbf{u}$, is also an eigenvector.]

In light of the last exercise, when we find an eigenvector for a matrix, we consider all of its scalar multiples to be the same eigenvector; we only consider eigenvectors to be distinct if they are not scalar multiples of one another. Since we will be working with *projective* Hilbert spaces, whenever we solve for an eigenvector, we always *normalize* it (divide by its magnitude) to force it onto the projective sphere. From

there, we can also multiply by a unit-length complex number, $e^{i\theta}$, in order to put it into a particularly neat form.

Vocabulary. If an eigenvalue, a , corresponds to only one eigenvector, \mathbf{u}_a , it is called a *non-degenerate* eigenvalue. However, if a works for two or more (linearly independent, understood) eigenvectors, $\mathbf{u}'_a, \mathbf{u}''_a, \dots$, it is called a *degenerate* eigenvalue.

There are two facts that I will state without proof. (They are easy enough to be exercises.)

- **Uniqueness.** Non-Degenerate eigenvalues have unique (up to scalar multiple) eigenvectors. Degenerate eigenvalues do not have unique eigenvectors.
- **Diagonality.** When the eigenvectors of a matrix, M , form a basis for the vector space, we call it an *eigenbasis* for the space. M , expressed as a matrix *in its own eigenbasis*, is a diagonal matrix (0 everywhere except for diagonal from position 1-1 to n - n).

[**Exercise.** Prove one or both of these facts.]

7.5.3 The Eigenvalues and Eigenvectors of S_z

Let's examine $M = S_z$. Most vectors (say, $(1, 2)^t$, for example) are not eigenvectors of S_z . For example,

$$\frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \frac{\hbar}{2} \begin{pmatrix} 1 \\ -2 \end{pmatrix} \neq a \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

for any complex scalar, a . However, the vector $(1, 0)^t$ is an eigenvector, as

$$\frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{\hbar}{2} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

demonstrates. It also tells us that $\frac{\hbar}{2}$ is the eigenvalue associated with the vector $(1, 0)^t$, which is exactly what **Trait #3** requires.

[**Exercise.** Show that $(0, 1)^t$ and $-\frac{\hbar}{2}$ form another eigenvector-eigenvalue pair for S_z .]

This confirms that **Trait #3** works for S_z ; we have identified the eigenvalues for S_z and they do, indeed, represent the only measurable values of the observable S_z in our experiments.

All of this results in a more informative variant of **Trait #3**, which I'll call **Trait #3'**.

Trait #3': *The only possible measurement outcomes of an observable, \mathcal{A} , are the solutions $\{a_k\}$ to the eigenvector-eigenvalue equation*

$$T_{\mathcal{A}} |u_k\rangle = a_k |u_k\rangle .$$

The values $\{a_k\}$ are always real, and are called the eigenvalues of the observable, while their corresponding kets, $\{|u_k\rangle\}$ are called the eigenkets. If each eigenvalue has a unique eigenket associated with it, the observable is called non-degenerate. On the other hand, if there are two or more different eigenkets that make the equation true for the same eigenvalue, that eigenvalue is called a degenerate eigenvalue, and the observable is called degenerate.

You may be wondering why we can say that the eigenvalues of an observable are always real when we have mentioned that, for a general matrix operator, we can get complex eigenvalues. This is related to the theoretical definition of an observable which requires it to be of a special form that always has real eigenvalues.

7.6 Computing Eigenvectors and Eigenvalues

We take a short detour to acquire the skills needed to compute eigenvalues for any observable.

The Eigenvalue Theorem. *Given an $n \times n$ matrix M , its eigenvalues are solutions to the n^{th} order polynomial equation in the unknown λ*

$$\det(M - \lambda I) = 0.$$

The equation in the *Eigenvalue Theorem* is known as the “characteristic equation” for M .

Proof. Assume that a is an eigenvalue for M whose corresponding eigenvector is \mathbf{u} . Then

$$\begin{aligned} M\mathbf{u} &= a\mathbf{u} && \Rightarrow \\ M\mathbf{u} &= aI\mathbf{u} && \Rightarrow \\ (M - aI)\mathbf{u} &= \mathbf{0}. \end{aligned}$$

Keeping in mind that eigenvectors are always non-zero, we have shown that the matrix $M - aI$ maps a non-zero \mathbf{u} into $\mathbf{0}$. But that’s the hypothesis of the *Little Inverse Theorem* “B” of our matrix lesson, so we get

$$\det(M - aI) = 0.$$

In words, a is a solution to the characteristic equation. QED

This tells us to solve the characteristic equation for an observable in order to obtain its eigenvalues. Getting the eigenvectors from the eigenvalues is more of an art and is best done using the special properties that your physical system \mathcal{S} imbues to the state space \mathcal{H} . For spin-1/2 state spaces, it is usually very easy and can be done by hand.

7.6.1 The Eigenvalues and Eigenvectors of S_y

The characteristic equation is

$$\det(S_y - \lambda I) = 0,$$

which is solved like so:

$$\begin{aligned} \left| \frac{\hbar}{2} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \right| &= 0 \\ \left| \begin{pmatrix} -\lambda & -\frac{\hbar}{2}i \\ \frac{\hbar}{2}i & -\lambda \end{pmatrix} \right| &= 0 \\ \lambda^2 - \frac{\hbar^2}{4} &= 0 \\ \lambda &= \pm \frac{\hbar}{2} \end{aligned}$$

Of course, we knew the answer, because we did the experiments (and in fact, the theoreticians crafted the S_y matrix based on the results of the experimentalists). Now comes the fun part. We want to figure out the eigenvectors for these eigenvalues. Get ready to do your first actual quantum mechanical calculation.

Eigenvector for $(+\hbar/2)$. The eigenvector has to satisfy

$$S_y \mathbf{u} = +\frac{\hbar}{2} \mathbf{u},$$

so we view this as an equation in the unknown coordinates (expressed in the preferred z -basis where we have been working all along),

$$\frac{\hbar}{2} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \frac{\hbar}{2} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}.$$

This reduces to

$$\begin{aligned} -i v_2 &= v_1 & \text{and} \\ i v_1 &= v_2. \end{aligned}$$

There are two equations in two unknowns. *Wrong.* There are *four* unknowns (each coordinate, v_k , is a complex number, defined by two real numbers). This is somewhat expected since we know that the solution will be a ray of vectors all differing by a complex scalar factor. We can solve for any one of the vectors on this ray as a first step. We do this by guessing that this ray has some non-zero first coordinate (and if we guess wrong, we would try again, the second time knowing that it must therefore have a non-zero second coordinate – [Exercise. Why?]. Using this guess, we can pick $v_1 = 1$, since any non-zero first coordinate can be made to $= 1$ by a scalar multiple of the entire vector. With this we get the complex equations

$$\begin{aligned} -i v_2 &= 1 \\ i \cdot 1 &= v_2, \end{aligned}$$

revealing that $v_2 = i$, so

$$\mathbf{u} = \begin{pmatrix} 1 \\ i \end{pmatrix},$$

which we must (always) normalize by projecting onto the unit (“projective”) sphere.

$$\frac{\mathbf{u}}{|\mathbf{u}|} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix} = \frac{|+\rangle + i|-\rangle}{\sqrt{2}}.$$

The last equality is the expression of \mathbf{u} explicitly in terms of the z -basis $\{|+\rangle, |-\rangle\}$.

Alternate Method. We got lucky in that once we substituted 1 for v_1 , we were able to read off v_2 immediately. Sometimes, the equation is messier, and we need to do a little work. In that case, naming the real and imaginary parts of v_2 helps.

$$v_2 = a + bi,$$

and substituting this into the original equations containing v_2 , above, gives

$$\begin{aligned} -i(a + bi) &= 1 \\ i &= (a + bi), \end{aligned}$$

or

$$\begin{aligned} b - ai &= 1 \\ i &= a + bi. \end{aligned}$$

Let’s solve the second equation for a , then substitute into the first, as in

$$\begin{aligned} a &= i - bi \Rightarrow \\ b - (i - bi)i &= 1 \Rightarrow \\ 1 &= 1. \end{aligned}$$

What does this mean? It means that we get a very agreeable second equation; the b disappears resulting in a true identity (a *tautology* to the logician). We can, therefore let b be *anything*. Again, when given a choice, choose 1. So $b = 1$ and substituting that into any of the earlier equations gives $a = 0$. Thus,

$$v_2 = a + bi = 0 + 1 \cdot i = i,$$

the same result we got instantly the first time. We would then go on to normalize \mathbf{u} as before.

Wrong Guess for v_1 ? If, however, after substituting for a and solving the first equation, b disappeared and produced a falsehood (like $0 = 1$), then no b would be suitable. That would mean our original choice of $v_1 = 1$ was not defensible; v_1 could not have been a non-zero value. We would simply change that assumption, set $v_1 = 0$ and go on to solve for v_2 (either directly or by solving for a and b to get it). This

time, we would be certain to get a solution. In fact, any time you end up facing a *contradiction* ($3 = 4$) instead of a *tautology* ($7 = 7$), then your original guess for v_1 has to be changed. Just redefine v_1 (if you chose 1, change it to 0) and everything will work out.

Too Many Solutions? (Optional Reading) In our well-behaved spin-1/2 state space, each eigenvalue determines a single ray in the state-space, so it only takes one unit eigenvector to describe it; you might say that the subspace of \mathcal{H} spanned by the eigenvectors corresponding to each eigenvalue is 1-dimensional. All of its eigenvectors differ by scalar factor. But in other physical systems the eigenvector equation related to a single eigenvalue may yield too many solutions (even after accounting for the scalar multiples on the same ray we already know about). In other words, there may be *multiple* linearly-independent solutions to the equation for one eigenvalue. If so, we select an orthonormal set of eigenvectors that correspond to the degenerate eigenvalue, as follows.

1. First observe (you can prove this as an easy [exercise]) that the set of all eigenvectors belonging to that eigenvalue form a *vector subspace* of the state space.
2. Use basic linear algebra to find *any* basis for this subspace.
3. Apply a process named “Gram-Schmidt” to replace that basis by an *orthonormal* basis.

Repeat this for any eigenvalue that is degenerate. You will get an optional exercise that explains why we want an *orthonormal basis* for the *eigenspace* of a *degenerate eigenvalue*.

Eigenvector for ($-\hbar/2$). Now it’s your turn.

[Exercise. Find the eigenvector for the negative eigenvalue.]

7.6.2 The Eigenvalues and Eigenvectors of S_x

Roll up your sleeves and do it all for the third primary direction.

[Exercise. Compute the eigenvalues and eigenvectors for the observable S_x .]

Congratulations. You are now “doing” full-strength quantum mechanics (not quantum *physics*, but, yes, quantum *mechanics*).

7.6.3 Summary of Eigenvectors and Eigenvalues for Spin-1/2 Observables

Spoiler Alert. These are the results of the two last exercises.

The eigenvalues and eigenvectors for S_z , S_x , and S_y are:

$$\begin{aligned} S_z : \quad & +\frac{\hbar}{2} \longleftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad -\frac{\hbar}{2} \longleftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ S_x : \quad & +\frac{\hbar}{2} \longleftrightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad -\frac{\hbar}{2} \longleftrightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \\ S_y : \quad & +\frac{\hbar}{2} \longleftrightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ i \end{pmatrix}, \quad -\frac{\hbar}{2} \longleftrightarrow \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -i \end{pmatrix} \end{aligned}$$

Expressed explicitly in terms of the z -basis vectors we find

$$\begin{aligned} S_z : \quad & \frac{\hbar}{2} \leftrightarrow |+\rangle, \quad -\frac{\hbar}{2} \leftrightarrow |-\rangle \\ S_x : \quad & \frac{\hbar}{2} \leftrightarrow |+\rangle_x = \frac{|+\rangle + |-\rangle}{\sqrt{2}}, \quad -\frac{\hbar}{2} \leftrightarrow |-\rangle_x = \frac{|+\rangle - |-\rangle}{\sqrt{2}} \\ S_y : \quad & \frac{\hbar}{2} \leftrightarrow |+\rangle_y = \frac{|+\rangle + i|-\rangle}{\sqrt{2}}, \quad -\frac{\hbar}{2} \leftrightarrow |-\rangle_y = \frac{|+\rangle - i|-\rangle}{\sqrt{2}} \end{aligned}$$

We saw the x -kets and y -kets before when we were trying to make sense out of the 50-50 split of a $|-\rangle_x$ state into the two states $|+\rangle_z$ and $|-\rangle_z$. Now, the expressions re-emerge as the result of a rigorous calculation of the eigenvectors for the observables S_x and S_y . Evidently, the eigenvectors of S_x are the same two vectors that you showed (in an exercise) were an alternative orthonormal basis for \mathcal{H} , and likewise for the eigenvectors of S_y .

Using these expressions along with the distributive property of inner products, it is easy to show orthonormality relations like

$$\begin{aligned} {}_x\langle + | + \rangle_x &= 1, & \text{or} \\ {}_x\langle + | - \rangle_x &= 0. \end{aligned}$$

[**Exercise.** Prove the above two equalities as well as the remaining combinations that demonstrate that both the x -bases and y -basis are each (individually) orthonormal.]

7.7 Observables and Orthonormal Bases

Thus far, we have met a simple 2-dimension Hilbert space, \mathcal{H} , whose vectors correspond to states of a spin-1/2 physical system, \mathcal{S} . The correspondence is not 1-to-1, since an entire *ray* of vectors in \mathcal{H} correspond to the same physical state, but we are learning to live with that by remembering that we can – and will – *normalize* all vectors whenever we want to see a proper representative of that ray on the unit (*projective*) sphere. In addition, we discovered that our three most common observables – S_z , S_x and S_y – correspond to operators whose *eigenvalues* are all $\pm\frac{\hbar}{2}$ and whose

eigenvectors form three different 2-vector bases for the 2-dimensional \mathcal{H} . Each of the bases is an orthonormal basis.

I'd like to distill two observations and award them collectively the title of *trait*.

7.7.1 Trait #4 (Real Eigenvalues and Orthonormal Eigenvectors)

An observable \mathcal{A} in \mathcal{S} will always correspond to an operator

1. *whose eigenvalues, a_1, a_2, \dots , are real, and*
2. *whose eigenvectors $\mathbf{u}_{a_1}, \mathbf{u}_{a_2}, \dots$, form an orthonormal basis for the state space \mathcal{H} .*

I did not call this trait a “postulate,” because it isn’t; you can prove these two properties based on **Trait #2**, which connects observables to *Hermitian operators*. Although we won’t spend our limited time proving it, if you are interested, try the next few exercises.

Note. I wrote the trait as if all the eigenvalues were non-degenerate. It is still true, even for degenerate eigenvalues, although then we would have to label the eigenvectors more carefully.

[**Exercise.** Show that a Hermitian operator’s eigenvalues are real.]

[**Exercise.** Show that eigenvectors corresponding to *distinct* eigenvalues are orthogonal.]

[**Exercise.** In an optional passage, above, I mentioned that a *degenerate* eigenvalue determines not a single eigenvector, but a *vector subspace* of eigenvectors, from which we can always select an orthonormal basis. Use this fact, combined with the last exercise to construct a *complete orthonormal set* of eigenvectors, including those that are non-degenerate (whose eigenspace is one-dimensional) and those that are degenerate (whose eigenspace requires multiple vectors to span it).]

I told a small white lie a moment ago when I said that this is totally provable from the second postulate (**Trait #2**). There is one detail that I left out of the second postulate which is needed to prove these observations. You did not possess the vocabulary to understand it at the time, but now you do. In **Trait #2** I said that the observable had to correspond to a Hilbert-space operator. What I left out was the following mini-trait, which I’ll call

Trait #2a (Completeness of the Eigenbasis): *The eigenvectors of an observable **span** – and since they are orthogonal, constitute a **basis** for – the state space. Furthermore, **every** measurable quantity of the physical system \mathcal{S} corresponds to an observable, thus its eigenvectors can be chosen as a basis whenever convenient.*

If we had an observable whose eigenvectors turned out *not* to span the state space, we did a bad job of defining the state space and would have to go back and

figure out a better mathematical Hilbert space to model \mathcal{S} . Similarly, if there were a measurable quantity for which we could not identify a linear operator, we have not properly modeled \mathcal{S} .

7.7.2 Using the x - or y -Basis

While we've been doggedly using the z -axis kets to act as a basis for our 2-dimensional state space, we know we can use the x -kets or y -kets. Let's take the x -kets as an example.

First and most easily, when we express any basis vector in its own coordinates, the results always look like natural basis coordinates, i.e., $(0, 1)^t$ or $(1, 0)^t$. This was an exercise back in our linear algebra lecture, but you can take a moment to digest it again. So, if we were to switch to using the x -basis for our state space vectors we would surely see that

$$\begin{aligned} |+\rangle_x &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}_x \quad \text{and} \\ |-\rangle_x &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}_x. \end{aligned}$$

How would our familiar z -kets now look in this new basis? You can do this by starting with the expressions for the x -kets in terms of $|+\rangle$ and $|-\rangle$ that we already have,

$$\begin{aligned} |+\rangle_x &= \frac{|+\rangle + |-\rangle}{\sqrt{2}} \quad \text{and} \\ |-\rangle_x &= \frac{|+\rangle - |-\rangle}{\sqrt{2}}, \end{aligned}$$

and solve for the z -kets in terms of the x -kets. It turns out that doing so results in déjà vu,

$$\begin{aligned} |+\rangle &= \frac{|+\rangle_x + |-\rangle_x}{\sqrt{2}} \quad \text{and} \\ |-\rangle &= \frac{|+\rangle_x - |-\rangle_x}{\sqrt{2}}. \end{aligned}$$

It's a bit of a coincidence, and this symmetry is not quite duplicated when we express the y -kets in terms of $|+\rangle_x$ and $|-\rangle_x$. I'll do one, and you can do the other as an exercise.

$|+\rangle_y$ in the x -Basis. The approach is to first write down $|+\rangle_y$ in terms of $|+\rangle$ and $|-\rangle$ (already known), then replace those two z -basis kets with their x -representation,

shown above. Here we go.

$$\begin{aligned}
 |+\rangle_y &= \frac{|+\rangle + i|-\rangle}{\sqrt{2}} = \frac{\left(\frac{|+\rangle_x + |-\rangle_x}{\sqrt{2}}\right) + i\left(\frac{|+\rangle_x - |-\rangle_x}{\sqrt{2}}\right)}{\sqrt{2}} \\
 &= \frac{(1+i)|+\rangle_x + (1-i)|-\rangle_x}{2} \\
 &\stackrel{\times(1-i)}{=} \frac{2|+\rangle_x - 2i|-\rangle_x}{2} = |+\rangle_x - i|-\rangle_x \\
 &\stackrel{\text{normalize}}{\cong} \frac{|+\rangle_x - i|-\rangle_x}{\sqrt{2}}
 \end{aligned}$$

[**Exercise.** Show that

$$|-\rangle_y = \frac{|+\rangle_x + i|-\rangle_x}{\sqrt{2}}]$$

[**Exercise.** Express $|\pm\rangle$ and $|\pm\rangle_x$ in the y -basis.]

7.7.3 General States Expressed in Alternate Bases

Expressing $|+\rangle$ or $|-\rangle$ in a different basis, like the x -basis, is just a special case of something you already know about. Any vector, whose coordinates are given in the *preferred* basis, like

$$|\psi\rangle = \alpha|+\rangle + \beta|-\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix},$$

has an alternate representation when its coefficients are expanded along a different basis, say the x -basis,

$$|\psi\rangle = \alpha'|+\rangle_x + \beta'|-\rangle_x = \begin{pmatrix} \alpha' \\ \beta' \end{pmatrix}_x.$$

If we have everything expressed in the preferred basis, then one day we find ourselves in need of all our vectors' names in the alternate (say x) basis, how do we do it? Well there is one way I didn't develop (yet) and it involves finding a simple matrix by which you could multiply all the preferred vectors to produce their coefficients expanded along the alternate basis. But we usually don't need this matrix. Since we are using orthonormal bases, we can convert to the alternate coordinates directly using the dot-with-the-basis vector trick. Typically, we only have one or two vectors whose coordinates we need in this alternate basis, so we just apply that trick.

For example, to get a state vector, $|\psi\rangle$, expressed in the x -basis, we just form the two inner-products,

$$|\psi\rangle = \begin{pmatrix} {}_x\langle+|\psi\rangle \\ {}_x\langle-|\psi\rangle \end{pmatrix},$$

where the syntax

$${}_x\langle + | \psi \rangle$$

means we are taking the complex inner product of $|\psi\rangle$ on the right with the x -basis ket $|+\rangle_x$ on the left. (Don't forget that we have to take the Hermitian conjugate of the left vector for complex inner products.)

We are implying by context that the column vector on the RHS is expressed in x -coordinates since that's the whole point of the paragraph. But if we want to be super explicit, we could write it as

$$|\psi\rangle = \begin{pmatrix} {}_x\langle + | \psi \rangle \\ {}_x\langle - | \psi \rangle \end{pmatrix}_x, \quad \text{or} \quad \begin{pmatrix} {}_x\langle + | \psi \rangle \\ {}_x\langle - | \psi \rangle \end{pmatrix}_x,$$

with or without the long vertical line, depending on author and time-of-day.

Showing the same thing in terms of the x -kets explicitly, we get

$$\begin{aligned} |\psi\rangle &= {}_x\langle + | \psi \rangle |+\rangle_x + {}_x\langle - | \psi \rangle |-\rangle_x \\ &= \left\langle \frac{1}{\sqrt{2}} \middle| \psi \right\rangle |+\rangle_x + \left\langle \frac{1}{\sqrt{2}} \middle| \psi \right\rangle |-\rangle_x \end{aligned}$$

Notice that the coordinates of the three vectors, $|\pm\rangle_x$ and $|\psi\rangle$ are expressed in the preferred z -basis. We can compute inner products in *any* orthonormal bases, and since we happen to know everything in the z -basis, why not? Try not to get confused. We are looking for the coordinates in the x -basis, so we need to “dot” with the x -basis vectors, but we use z -coordinates of those vectors (and the z -coordinates of state $|\psi\rangle$) to compute those two scalars.

Example. The (implied z -spin) state vector

$$\begin{pmatrix} \frac{1+i}{\sqrt{6}} \\ -\frac{\sqrt{2}}{\sqrt{3}} \end{pmatrix}$$

has the x -spin coordinates,

$$\begin{pmatrix} \left\langle \frac{1}{\sqrt{2}} \middle| \frac{1+i}{\sqrt{6}} \right\rangle \\ \left\langle \frac{1}{\sqrt{2}} \middle| -\frac{\sqrt{2}}{\sqrt{3}} \right\rangle \\ \left\langle -\frac{1}{\sqrt{2}} \middle| \frac{1+i}{\sqrt{6}} \right\rangle \\ \left\langle -\frac{1}{\sqrt{2}} \middle| -\frac{\sqrt{2}}{\sqrt{3}} \right\rangle \end{pmatrix}_x$$

[**Exercise.** Compute and simplify.]

7.8 The Completeness (or Closure) Relation

7.8.1 Orthonormal Bases in Higher Dimensions

Our casual lesson about conversion from the z -basis to the x -basis has brought us to one of the most computationally useful tools in quantum mechanics. It's something that we use when doodling with pen and paper to work out problems and construct algorithms, so we don't want to miss the opportunity to establish it formally, right now. We just saw that

$$|\psi\rangle = {}_x\langle + | \psi \rangle_x |+\rangle_x + {}_x\langle - | \psi \rangle_x |-\rangle_x ,$$

which was true because $\{|+\rangle_x, |-\rangle_x\}$ formed an orthonormal basis for our state space.

In systems with higher dimensions we have a more general formula. Where do we get state spaces that have dimensions higher than two? A spin 1 system – photons – has 3 dimensions; a spin-3/2 system – delta particle – has 4-dimensions and later when we get into multi-qubit systems, we'll be taking tensor products of our humble 2-dimensional \mathcal{H} to form 8-dimensional or larger state spaces. And the state space that models position and momentum are infinite dimensional (but don't let that scare you – they are actually just as easy to work with – we use integrals instead of sums).

If we have an n -dimensional state space then we would have an orthonormal basis for that space, say,

$$\left\{ |u_k\rangle \right\}_{k=1}^n .$$

The $|u_k\rangle$ basis may or may not be a preferred basis – doesn't matter. Using the dot-product trick we can always expand any state in that space, say $|\psi\rangle$, along the basis just like we did for the x -basis in 2-dimensions. Only, now, we have a larger sum

$$|\psi\rangle = \sum_{k=1}^n \langle u_k | \psi \rangle |u_k\rangle .$$

This is a weighted sum of the u_k -kets by the scalars $\langle u_k | \psi \rangle$. There is no law that prevents us from placing the scalars on the right side of the vectors, as in

$$|\psi\rangle = \sum_{k=1}^n |u_k\rangle \langle u_k | \psi \rangle .$$

Now, we can take the $|\psi\rangle$ out of the sum

$$|\psi\rangle = \left(\sum_{k=1}^n |u_k\rangle \langle u_k| \right) |\psi\rangle .$$

Look at what we have. We are subjecting any state vector $|\psi\rangle$ to something that looks like an operator and getting that same state vector back again. In other words, that fancy looking *operator-sum* is nothing but an identity operator $\mathbb{1}$.

$$\left(\sum_{k=1}^n |u_k\rangle \langle u_k| \right) = \mathbb{1}.$$

This simple relation, called the *completeness* or *closure relation*, can be applied by inserting the sum into any state equation without changing it, since it is the same as inserting an identity operator (an identity matrix) into an equation involving vectors. We'll use it a little in this course, CS 83A, and a lot in the next courses CS 83B and CS 83C, here at Foothill College.

[**Exercise.** Explain how the sum, $\sum_k |u_k\rangle \langle u_k|$ is, in fact, a linear transformation that can act on a vector $|\psi\rangle$. **Hint:** After applying it to $|\psi\rangle$ and distributing, each term is just an inner-product (resulting in a scalar) times a vector. Thus, you can analyze a simple inner product first and later take the sum, invoking the properties of linearity.]

This is worthy of its own trait.

7.8.2 Trait #5 (Closure Relation)

Any orthonormal basis $\{|u_k\rangle\}$ for our Hilbert space \mathcal{H} satisfies the closure relation

$$\left(\sum_{k=1}^n |u_k\rangle \langle u_k| \right) = \mathbb{1}.$$

In particular, the eigenvectors of an observable will always satisfy the closure relation.

We won't work any examples as this will take us too far afield, and in our simple 2-dimensional \mathcal{H} there's not much to see, but we now have it on-the-record.

7.9 The Fourth Postulate of Quantum Mechanics

There are two versions of this one, and we only need the simpler of the two which holds in the case of *non-degenerate eigenvalues*, the situation that we will be using for our spin-1/2 state spaces. (Also, you'll note our *traits* are no longer in-sync with the *postulates*, an inevitable circumstance since there are more traits than postulates.)

7.9.1 Trait #6 (Probability of Outcomes)

If a system is in the (normalized) state $|\psi\rangle$, and this state is expanded along the eigenbasis $\{|u_k\rangle\}$ of some observable, \mathcal{A} , i.e.,

$$|\psi\rangle = \sum_{k=1}^n c_k |u_k\rangle,$$

then the probability that a measurement of \mathcal{A} will yield a non-degenerate eigenvalue a_k (associated with the eigenvector $|u_k\rangle$) is $|c_k|^2$.

Vocabulary. The expansion coefficients, c_k , for state $|\psi\rangle$, are often referred to as *amplitudes* by physicists.

In this language, the probability of obtaining a measurement outcome a_k for observable \mathcal{A} is the magnitude-squared of the *amplitude* c_k standing next to the eigenvector $|u_k\rangle$ associated with the outcome a_k .

The complex coordinates of the state determine the statistical outcome of repeated experimentation. This is about as quantum mechanical a concept as there is. It tells us the following.

- If a state is a *superposition* (non-trivial linear combination) of two or more eigenkets, we cannot know an outcome of a quantum measurement with certainty.
- This is not a lack of knowledge about the system, but a statement about what it means to know everything knowable about the system, namely the full description of the state.
- The coefficient (or *amplitude*) c_k gives you the probability of the outcome a_k , namely

$$\mathcal{P}(a_k)_{|\psi\rangle} = c_k^* c_k = |c_k|^2 ,$$

to be read, “when a system in state $|\psi\rangle$ is measured, its probability of outcome, a_k , is the square magnitude of c_k .” The observable in question is understood to be the one whose eigenvalues-eigenvectors are $\{a_k \leftrightarrow |u_k\rangle\}$.

Probability Example 1

Let’s analyze our original **Experiment #2** (last optional chapter), in which we started with a group of electrons in the $+z$ state, i.e., $|+\rangle$, and tested each of their spin projections along the x -axis, i.e., we measured the observable S_x . In order to predict the probabilities of the outcome of the S_x measurement using the fourth QM postulate – our **Trait #6** – we expand the pre-measurement state along the eigenvector basis for S_x and then examine each coefficient. The pre-measurements state is $|+\rangle$. We have previously expressed that state in terms of the S_x eigenvectors, and the result was

$$|+\rangle = \frac{|+\rangle_x + |-\rangle_x}{\sqrt{2}} .$$

The amplitude of each of the two possible outcome states (the two eigenstates $|+\rangle_x$ and $|-\rangle_x$) is $\frac{1}{\sqrt{2}}$. This tell us that each eigenstate outcome is equally likely and, in fact, determined by

$$\mathcal{P}\left(S_x = +\frac{1}{\sqrt{2}}\hbar\right)_{|+\rangle} = \left(\frac{1}{\sqrt{2}}\right)^* \left(\frac{1}{\sqrt{2}}\right) = \frac{1}{2} ,$$

while

$$\mathcal{P}\left(S_x = -\frac{1}{\sqrt{2}}\hbar\right)_{|+\rangle} = \left(\frac{1}{\sqrt{2}}\right)^* \left(\frac{1}{\sqrt{2}}\right) = \frac{1}{2}.$$

Notice that the two probabilities add to 1. Is this a happy coincidence? I think not. The first postulate of QM (our **Trait #1**) guarantees that we are using unit vectors to correspond to system states. If we had a non-unit vector that was supposed to represent that state, we'd need to normalize it first before attempting to compute the probabilities.

Probability Example 2

The follow-up to **Experiment #2** was the most shocking to us, so we should see how it is predicted by **Trait #6**. The starting point for this measurement was the output of the second apparatus, specifically, the $-x$ group: $|-\rangle_x$. We then measured S_z , so we need to know the coefficients of the state $|-\rangle_x$ along the S_z eigenbasis. We computed them already, and they're contained in

$$|-\rangle_x = \frac{|+\rangle - |-\rangle}{\sqrt{2}}.$$

The arithmetic we just did works exactly the same here, and our predictions are the same:

$$\mathcal{P}\left(S_z = +\frac{1}{\sqrt{2}}\hbar\right)_{|-\rangle_x} = \left(-\frac{1}{\sqrt{2}}\right)^* \left(-\frac{1}{\sqrt{2}}\right) = \frac{1}{2},$$

and

$$\mathcal{P}\left(S_z = -\frac{1}{\sqrt{2}}\hbar\right)_{|-\rangle_x} = \left(-\frac{1}{\sqrt{2}}\right)^* \left(-\frac{1}{\sqrt{2}}\right) = \frac{1}{2}.$$

Notice that, despite the amplitude's negative signs, the probabilities still come out non-negative.

[**Exercise.** Analyze the S_y measurement probabilities of an electron in the state $|-\rangle$. Be careful. This time we have a complex number to conjugate.]

Probability Example 3

The z -spin coordinates of a state, $|\psi\rangle$ are given by

$$\begin{pmatrix} \frac{1+i}{\sqrt{6}} \\ -\frac{\sqrt{2}}{\sqrt{3}} \end{pmatrix}.$$

The probability of detecting a z -UP spin is given by the coefficient (amplitude) of the $|+\rangle$, and is

$$\begin{aligned}\mathcal{P}\left(S_z = +\frac{1}{\sqrt{2}}\hbar\right)_{|\psi\rangle} &= \left(\frac{1-i}{\sqrt{6}}\right)\left(\frac{1+i}{\sqrt{6}}\right) \\ &= \frac{2}{6} = \frac{1}{3}.\end{aligned}$$

The probability of detecting an x -DOWN spin starting with that same $|\psi\rangle$ requires that we project that state along the x -basis,

$$|\psi\rangle = c_+ |+\rangle_x + c_- |-\rangle_x.$$

However, since we only care about the x -down state, we can just compute the $|-\rangle_x$ coefficient, which we do using the dot-product trick.

$$\begin{aligned}c_- &= \left\langle \begin{array}{c} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{array} \middle| \begin{array}{c} \frac{1+i}{\sqrt{6}} \\ -\frac{\sqrt{2}}{\sqrt{3}} \end{array} \right\rangle = \frac{1+i}{\sqrt{12}} + \frac{1}{\sqrt{3}} \\ &= \frac{1+i+2}{\sqrt{12}} = \frac{3+i}{\sqrt{12}}.\end{aligned}$$

Now we take the magnitude squared,

$$|c_-|^2 = \left(\frac{3-i}{\sqrt{12}}\right)\left(\frac{3+i}{\sqrt{12}}\right) = \frac{10}{12} = \frac{5}{6}.$$

[Exercise. Compute the $-z$ and $+x$ spin probabilities for this $|\psi\rangle$ and confirm that they complement their respective partners that we computed above. Explain what I mean by “complementing their partners.”]

[Exercise. Compute the $+y$ and $-y$ spin probabilities for this $|\psi\rangle$.]

7.10 The Fifth Postulate of Quantum Mechanics

After the first experiment of the last lesson, in which we measured the z -spin of random electrons and noted that the measurement caused them to split into two equal groups, one having all z -up states and the other having all z -down states, we did a follow-up. We tested S_z again on each output group separately. In case you missed it, here’s what happened. When we re-tested the $|+\rangle$ group, the results always gave us $+z$ readings, and when we re-tested the $|-\rangle$ group we always got $-z$ readings. It was as if, after the first measurement had divided our electrons into two equal groups of $+z$ and $-z$ spins, any subsequent tests suggested that the two groups were frozen into their two respective states, *as long as we only tested S_z* .

However, the moment we tested a different observable, like S_x , we disturbed that z -axis predictability. So the collapse of the system into the S_z state was only valid as long as we continued to test that one observable. This is our next trait.

7.10.1 Trait #7 (Post-Measurement Collapse)

If the measurement of an observable of system \mathcal{S} results in the eigenvalue, a_k , then the system "collapses" into the (an) eigenvector $|u_k\rangle$, associated with a_k . Further measurements on this collapsed state yields the eigenvalue a_k with 100% certainty.

Vocabulary Review. The eigenvectors of an observable are also known as *eigenkets*.

We've been saying all along that the eigenvalue a_k might be degenerate. The implication here is that there may be more than one possibility for the collapsed state, specifically, any of the eigenvectors $|u'_k\rangle$, $|u''_k\rangle$, $|u'''_k\rangle$, \dots which correspond to a_k . We won't encounter this situation immediately, but it will arise later in the course. The early easy cases will consist of non-degenerate eigenvalues whose probabilities are easily computed by the amplitudes of their respective unique eigenvectors. Later, when we get to degenerate eigenvalues, we won't be sure which of the eigenvectors – corresponding to that eigenvalue – represents the state into which the system collapsed. Yet, even knowing that it collapsed to the small subset of eigenvectors corresponding to a single eigenvalue (in the degenerate case) is invaluable information that plays directly into our algorithms.

Preparing Special System States

The impact of **Trait #7** is that engineers can prepare special states to act as input into our quantum hardware logic. This is akin to setting a register's value in a classical computer using an assignment statement, prior to beginning further logic.

Example: Preparing a Basis (Eigenvector) State. We did this in **Experiment #1**. After measuring the observable S_z , we ended up with two groups. By selecting either one of those groups, we will be getting either the $|+\rangle$ state or the $|-\rangle$ state, as we wish. Any future testing of S_z will confirm that we stay in those states, as long as we don't subject the system to forces that modify it.

Example: Preparing a Superposition State. We did this, too. In our late stages of tinkering with **Experiment #2**, we focused on the output of the second apparatus by choosing the $|-\rangle_x$ group for further investigation. Using our state space vocabulary, we expand the state $|-\rangle_x$ in terms of the z -eigenkets,

$$|+\rangle_x = \frac{|+\rangle + |-\rangle}{\sqrt{2}},$$

and realize that we have prepared a state which, with respect to the z -spin observable, is not a basis state but a linear combination – a *superposition* – of the two z -basis states (with equally weighted components). This kind of state preparation will be very important for quantum algorithms, because it represents starting out in a state which is neither 0 nor 1, but a combination of the the two. This allows us to work with a single state in our quantum processor and get two results for the price of one. A single qubit and a single machine cycle will simultaneously produce answers for both 0 and 1.

But, after we have prepared one of these states, how do we go about giving it to a quantum processor, and what *is* a quantum processor. That's answered in the first quantum computing lesson coming up any day now. Today, we carry on with pure quantum mechanics to acquire the full quiver of q-darts.

7.11 Summary of What Quantum Mechanics Tells Us About a System

We now have a complete set of numbers and mathematical entities that give us all there is to know about any quantum system. We know

1. the possible states $|\psi\rangle$ of the system (vectors in our state space),
2. the possible outcomes of a measurement of an *observable* of that system (the *eigenvalues*, a_k , of the observable),
3. the *eigenvectors* states (a.k.a. *eigenbasis*), $|u_k\rangle$, into which the system collapses after we detect a value, a_k , of that observable, and
4. the probabilities associated with each eigenvalue outcome (specifically, $|c_k|^2$, which are derived from the *amplitudes*, c_k , of $|\psi\rangle$ expanded along the *eigenbasis* $|u_k\rangle$).

In words, the system is in a state of probabilities. We can only get certain special outcomes of measurement. Once we measure, the system collapses into a special state associated with that special outcome. The probability that this measurement occurs is predicted by the coefficients of the state's eigenvector expansion.

7.12 Dirac's Bra-ket Notation

We take a short break from physics to talk about the universal notation made popular by the famous physicist Paul Dirac.

7.12.1 Kets and Bras

It's time to formalize something we've been using loosely up to this point: the *bracket*, or *bra-ket*, notation. The physicists' expression for a complex inner product,

$$\langle \mathbf{v} | \mathbf{w} \rangle, \quad \langle \eta | \psi \rangle, \quad \langle u_k | \psi \rangle, \quad \langle + | - \rangle,$$

can be viewed, not as a combination of two vectors in the same vector space, but rather as two vectors, each from *different* vector space. Take, for example,

$$\langle \eta | \psi \rangle.$$

The RHS of the inner product, $|\psi\rangle$, is the familiar vector in our state space, or *ket space*. Nothing new there. But the LHS, $\langle\eta|$, is to be thought of as a vector from a new vector space, called the *bra-space* (mathematicians call it the dual space). The bra space is constructed by taking *conjugate transpose* of the vectors in the ket space, that is,

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \longrightarrow \langle\psi| = (\alpha^*, \beta^*) .$$

Meanwhile, the *scalars* for the *bra space* are the same: the complex numbers, \mathbb{C} .

Examples

Here are some kets (not necessarily normalized) and their associated bras.

$$|\psi\rangle = \begin{pmatrix} 1 + i \\ \sqrt{2} - 2i \end{pmatrix} \longrightarrow \langle\psi| = (1 - i, \sqrt{2} + 2i)$$

$$|\psi\rangle = \begin{pmatrix} \sqrt{3}/2 \\ -i \end{pmatrix} \longrightarrow \langle\psi| = (\sqrt{3}/2, i)$$

$$|\psi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 5i \\ 0 \end{pmatrix} \longrightarrow \langle\psi| = \frac{1}{\sqrt{2}} (-5i, 0)$$

$$|\psi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \longrightarrow \langle\psi| = \frac{1}{\sqrt{2}} (1, 1)$$

$$|+\rangle \longrightarrow \langle+|$$

$$|-\rangle_y \longrightarrow {}_y\langle-|$$

[**Exercise.** Show that

$$|+\rangle + i|-\rangle \longrightarrow \langle+| - i\langle-| .$$

Hint: It's probably easier to do this without reading a hint, but if you're stuck ... write out the LHS as a single column vector and take the conjugate transpose. Meanwhile the RHS can be constructed by constructing the bras for the two z -basis vectors (again using coordinates) and combining them. The two efforts should result in the same vector.]

Vocabulary and Notation

Notice that bras are written as *row-vectors*, which is why we call them conjugate *transposes* of kets. The dagger (\dagger) is used to express the fact that a *ket* and *bra* bear

this *conjugate transpose* relationship,

$$\begin{aligned}\langle\psi| &= |\psi\rangle^\dagger \quad \text{and} \\ |\psi\rangle &= \langle\psi|^\dagger.\end{aligned}$$

This should sound very familiar. Where have we seen *conjugate transpose* before? **Answer:** When we defined the *adjoint* of a matrix. We even used the same dagger (\dagger) notation. In fact, you saw an example in which the matrix had only one row (or one column) – i.e., a *vector*. (See the lesson on *linear transformations*.) This is the same operation: *conjugate transpose*.

Be careful not say that a *ket* is the complex conjugate of a *bra*. Complex conjugation is used for scalars, only. Again, we say that a bra is the *adjoint* of the ket (and vice versa). If you want to be literal, you can always say *conjugate transpose*. An alternate term physicists like to use is *Hermitian conjugate*, “the bra is the *Hermitian conjugate* of the ket.”

Example

Let’s demonstrate that the sum of two *bras* is also a *bra*. What does that even *mean*? If $\langle\psi|$ and $\langle\eta|$ are *bras*, they must be the adjoints of two *kets*,

$$\begin{aligned}|\psi\rangle &= \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad \longleftrightarrow \quad \langle\psi| = (\alpha^*, \beta^*) . \\ |\eta\rangle &= \begin{pmatrix} \gamma \\ \delta \end{pmatrix} \quad \longleftrightarrow \quad \langle\eta| = (\gamma^*, \delta^*) .\end{aligned}$$

We add the bras component-wise,

$$\langle\eta| + \langle\psi| = (\alpha^* + \gamma^*, \beta^* + \delta^*) ,$$

which you can see is the Hermitian conjugate of the ket

$$|\eta\rangle + |\psi\rangle = \begin{pmatrix} \alpha + \gamma \\ \beta + \delta \end{pmatrix} ,$$

i.e.,

$$\langle\eta| + \langle\psi| = (|\eta\rangle + |\psi\rangle)^\dagger .$$

That’s all a *bra* needs to be: the Hermitian conjugate of some *ket*. So the sum is a bra.

There is (at least) one thing we must confirm. As always, when we define anything in terms of coordinates, we need to be sure that the definition is independent of our choice of basis (since coordinates arise from some basis). I won’t prove this, but you may choose to do so as an exercise.

[**Exercise.** Pick any three axioms of a vector space and prove that the bras in the bra space obey them.]

[**Exercise.** Show that the definition of bra space is independent of basis.]

Remain Calm. There is no cause for alarm. Bra space is simply a device that allows us manipulate the equations without making mistakes. It gives us the ability to talk about the LHS and the RHS of an inner product *individually* and symmetrically, unattached to the inner product.

Elaborate Example

We will use the bra notation to compute the inner product of the two somewhat complicated kets,

$$\begin{aligned} c|\psi\rangle + |\eta\rangle & \quad \text{on the left, and} \\ \frac{d|\phi\rangle - f|\theta\rangle}{g} & \quad \text{on the right,} \end{aligned}$$

where c , d , f and g are some complex scalars. The idea is very simple. We first take the Hermitian conjugate of the *intended left vector* by first

1. turning all of its kets into bras and
2. taking the complex conjugate of any scalars (in that left vector) which are outside a ket.

Next, we calculate, by

3. forming the desired inner product of the bra on the left with the ket on the right, and
4. using the distributive property to combine the component kets and bras.

Applying steps 1 and 2 on the left ket produces the bra

$$c^* \langle \psi | + \langle \eta | .$$

Step 3 gives us

$$\left(c^* \langle \psi | + \langle \eta | \right) \left(\frac{d|\phi\rangle - f|\theta\rangle}{g} \right) ,$$

and applying step 4 we get

$$\frac{c^* d \langle \psi | \phi \rangle + d \langle \eta | \phi \rangle - c^* f \langle \psi | \theta \rangle + f \langle \eta | \theta \rangle}{g} .$$

It seems overly complicated, but usually we apply it to simpler combinations and it is far less cumbersome than turning all the constituent kets into their coordinates, combining them, taking the conjugates of the left ket and doing the final “dot.”

Simple Example

$$\begin{aligned}
 {}_y\langle +|+\rangle &= \left\langle \left(\frac{|+\rangle + i|-\rangle}{\sqrt{2}} \right) \right| + \rangle \\
 &= \left(\frac{\langle +| - i\langle -|}{\sqrt{2}} \right) | + \rangle \\
 &= \frac{\langle +|+\rangle - i\langle -|+\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}}.
 \end{aligned}$$

The first thing we did was to express $|+\rangle_y$ in the z -basis without converting it to a bra. Then, we used the techniques just presented to convert that larger expression into a bra. From there, it was a matter of distributing the individual kets and bras and letting them neutralize each other. Normally, we would perform the first two steps at once, as the next example demonstrates.

$$\begin{aligned}
 {}_y\langle -|+\rangle_x &= \left(\frac{\langle +| + i\langle -|}{\sqrt{2}} \right) \left(\frac{|+\rangle + |-\rangle}{\sqrt{2}} \right) \\
 &= \frac{\langle +|+\rangle + i\langle -|+\rangle + \langle +|-\rangle + i\langle -|-\rangle}{2} \\
 &= \frac{1 + i}{2}.
 \end{aligned}$$

[**Exercise.** Compute ${}_y\langle +|+\rangle_x$ and ${}_x\langle -|+\rangle_y$.]

Summary

The bra space is a different vector space from the ket (our state) space. It is, however, an exact copy (an *isomorphism*) of the state space in the case of finite dimensions - all we ever use in quantum computing. You now have enough chops to prove this easily, so I leave it as an ...

[**Exercise.** Prove that the adjoint of the ket basis is a basis for bra space. **Hint:** Start with any bra. Find the ket from which it came (this step is not always possible in infinite dimensional Hilbert space, as your physics instructors will tell you). Expand that ket in any state space basis, then]

7.12.2 The Adjoint of an Operator

We now leverage the earlier definition of a matrix adjoint and extend our ability to translate expressions from the ket space to the bra space (and back). If A is an operator (or matrix) in our state space (not necessarily Hermitian - it could be any operator), then its adjoint A^\dagger can be viewed as an operator (or matrix) in the bra space.

$$A^\dagger : \langle \psi | \longmapsto \langle \phi |.$$

A^\dagger operates on bras, but since bras are row-vectors, it has to operate on the right, not left:

$$\langle\psi| A^\dagger \longmapsto \langle\phi|.$$

And the symmetry we would like to see is that the “output” $\langle\phi|$ to which A^\dagger maps $\langle\psi|$ is the bra corresponding to the ket $A|\psi\rangle$. That dizzying sentence translated into symbols is

$$\langle\phi| \longleftrightarrow (A|\psi\rangle)^\dagger.$$

Example. Start with our familiar 2-dimensional state space and consider the operator,

$$A = \begin{pmatrix} i & -i \\ 1 & 0 \end{pmatrix}.$$

Its adjoint is

$$A^\dagger = \begin{pmatrix} -i & 1 \\ i & 0 \end{pmatrix}.$$

A^\dagger maps the bra, $\langle\psi| = (1+i, 3)$ into

$$\langle\psi| A^\dagger = (1+i, 3) \begin{pmatrix} -i & 1 \\ i & 0 \end{pmatrix} = (1+2i, 1+i).$$

Meanwhile, back in ket space, A maps $|\psi\rangle = \langle\psi|^\dagger$ into

$$A|\psi\rangle = \begin{pmatrix} i & -i \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1-i \\ 3 \end{pmatrix} = \begin{pmatrix} 1-2i \\ 1-i \end{pmatrix}.$$

As you can see by comparing the RHS of both calculations, the adjoint of $A|\psi\rangle$ is $\langle\psi| A^\dagger$, in agreement with the claim.

7.12.3 The Adjoint Conversion Rules

The reason we added adjoint operators into the mix was to supply the final key to the processes of converting any combination of state space kets into bras and any combination of bras into kets. Say we desire to convert the *ket* expression

$$c A|\psi\rangle + d \langle +|\eta\rangle |\eta\rangle$$

into its *bra* counterpart. The rules will guide us, and they work for expressions far more complex with equal ease.

We state the rules, then we will try them out on this expression. This calls for a new trait.

7.12.4 Trait #8 (Adjoint Conversion Rules)

- The terms of a sum can be (but don't have to be) left in the same order.
- The order of factors in a product are reversed.
- Kets are converted to bras (i.e., take the adjoints).
- Bras are converted to kets (i.e., take the adjoints).
- Operators are converted to their adjoints.
- Scalars are converted to their complex conjugates.
- (Covered by the above, but stated separately anyway:) Inner products are reversed.
- When done (for readability only), rearrange each product so that the scalars are on the left of the vectors.

If we apply the adjoint conversion rules, except for the readability step, to the above combination we get

$$\left(c A |\psi\rangle + d \langle + | \eta \rangle | \eta \rangle \right)^\dagger = \langle \psi | A^\dagger c^* + \langle \eta | \langle \eta | + \rangle d^*,$$

which we rearrange to

$$c^* \langle \psi | A^\dagger + d^* \langle \eta | + \rangle \langle \eta |.$$

You'll get fast at this with practice. I don't want to spend any more real estate on the topic, since we don't apply it very much in our first course, CS 83A, but here are a couple exercises that will take care of any lingering urges.

[**Exercise.** Use the rules to convert the resulting bra of the last example back into a ket. Confirm that you get the ket we started with.]

[**Exercise.** Create a wild ket expression consisting of actual literal scalars, matrices and column vectors. Use the rules to convert it to a bra. Then use the same rules to convert the bra back to a ket. Confirm that you get the ket you started with.]

7.13 Expectation Values

Say we have modeled some physical quantum system, \mathcal{S} , with a Hilbert space, \mathcal{H} . Imagine, further, that we want to study some observable, \mathcal{A} , that has (all non-degenerate) eigenvalues $\{a_k\}$ with corresponding eigenkets $\{|u_k\rangle\}$. Most importantly, we assume that we can prepare many identical copies of \mathcal{S} , all in the same state $|\psi\rangle$.

(We did this very thing by selecting only z -up electrons in a Stern-Gerlach-like apparatus, for example.) We now look at our state expanded along the \mathcal{A} eigenbasis,

$$|\psi\rangle = \sum_k c_k |u_k\rangle.$$

How do the amplitudes, c_k , and their corresponding probabilities, $|c_k|^2$, make themselves felt, by us human experimenters?

The answer to this question starts by taking many repeated measurements of the observable \mathcal{A} on these many identical states $|\psi\rangle$ and recording our results.

[**Exercise.** Explain why we can't get the same results by repeating the \mathcal{A} measurements on a single system \mathcal{S} in state $|\psi\rangle$.]

7.13.1 The Mean of the Experiments

We'll take a large number, N , of measurements, record them, and start doing elementary statistics on the results. We label the measurements we get using

$$j\text{th measurement of } \mathcal{A} = m_j.$$

As a start, we compute average (or *mean*) of all N measurements,

$$\overline{m} = \frac{1}{N} \sum_{j=1}^N m_j.$$

If we take a large enough N , what do we expect this average to be? This answer comes from the statistical axiom called the *law of large numbers*, which says that this value will approach *the expectation value*, μ , as $N \rightarrow \infty$, that is,

$$\lim_{N \rightarrow \infty} \overline{m} = \mu.$$

This is good and wonderful, but I have not yet defined the *expectation value* μ . Better do that, fast.

[**Note.** I should really have labeled \overline{m} with N , as in \overline{m}_N , to indicate that each average depends on the number of measurements taken, as we are imagining that we can do the experiment with larger and larger N . But you understand this without the extra notation.]

7.13.2 Defining Expectation Value

This conveniently brings us back around to **Trait #6** (the *Fourth Postulate of QM*) concerning probabilities of outcomes. It asserts that the probability of detecting the

measurement (eigenvalue) a_k is given by its k th expansion coefficient along the \mathcal{A} basis (c_k), specifically by its magnitude-squared, $|c_k|^2$,

$$\mathcal{P}(a_k)_{|\psi\rangle} = c_k^* c_k = |c_k|^2 .$$

This motivates the definition. Expectation value, μ , is defined by summing up each possible outcome (the eigenvalues a_k), weighted by their respective probabilities, $|c_k|^2$:

$$\mu \equiv \sum_k |c_k|^2 a_k .$$

In case you don't see why this has the feeling of an expectation value (something we might expect from a typical measurement, if we were forced to place a bet), read it in English:

*The first measurable value times the probability of getting that value
plus
the second measurable value times the probability of getting that value
plus
... .*

In physics, rather than using the Greek letter μ , the notation for expectation value focuses attention on the observable we are measuring, \mathcal{A} ,

$$\langle \mathcal{A} \rangle_{|\psi\rangle} \equiv \sum_k |c_k|^2 a_k .$$

Fair Warning. In quantum mechanics, you will often see the expectation value of the observable, \mathcal{A} , written without the subscript, $|\psi\rangle$,

$$\langle \mathcal{A} \rangle ,$$

but this doesn't technically make sense. There is no such thing as an expectation value for an observable that applies without some assumed state; you must know which $|\psi\rangle$ has been prepared prior to doing the experiment. If this is not obvious to you, look up at the definition one more time: We don't have any c_k to use in the formula unless there is a $|\psi\rangle$ in the room, because

$$|\psi\rangle = \sum_k c_k |u_k\rangle .$$

When authors suppress the subscript state on the expectation value, it's usually because the context strongly implies the state or the state is explicitly described earlier and applies "until further notice."

Calculating an expectation value tells us one way we can use the amplitudes. This, in turn acts as an approximation of the average \overline{m} , of a set of experimental measurements on multiple systems in the identical state.

7.13.3 Computing Expectation Value

It seems like we should be done with this section. We have a formula for the *expectation value*, $\langle \mathcal{A} \rangle_{|\psi\rangle}$, so what else is there to do? It turns out that computing that sum isn't always as easy or efficient as computing the value a different way.

7.13.4 Trait #9 (Computing an Expectation Value)

The expectation value of an observable \mathcal{A} of a system in state $|\psi\rangle$ is usually computed using the expression $\langle \psi | \mathcal{A} | \psi \rangle$.

To add formality, we will also call this the

Expectation Value Theorem.

$$\langle \mathcal{A} \rangle_{|\psi\rangle} = \langle \psi | \mathcal{A} | \psi \rangle .$$

The \mathcal{A} on the RHS can be thought of either as the *operator* representing \mathcal{A} , or the *matrix* for the operator. The expression can be organized in various ways, all of which result in the same real number. For instance, we can first compute $\mathcal{A}|\psi\rangle$ and then take the inner product

$$\langle \psi | \left(\mathcal{A} | \psi \rangle \right) ,$$

or we can first apply \mathcal{A} to the bra $\langle \psi |$, and then dot it with the ket,

$$\left(\langle \psi | \mathcal{A} \right) | \psi \rangle .$$

If you do it this way, be careful *not take the adjoint* of \mathcal{A} . Just because we apply an operator to a bra does not mean we have to take its Hermitian conjugate. The formula says to use \mathcal{A} , *not* \mathcal{A}^\dagger , regardless of which vector we feed it.

[**Exercise.** Prove that the two interpretations of $\langle \psi | \mathcal{A} | \psi \rangle$ are equal by expressing everything in component form with respect to any basis. **Hint:** It's just (a row vector) \times (a matrix) \times (a column vector), so multiply it out both ways.]

Proof of the Expectation Value Theorem. This is actually one way to prove the last exercise nicely. Express everything in the \mathcal{A} -basis (i.e., the eigenkets of \mathcal{A} which **Trait #4** tells us form a basis for the state space \mathcal{H}). We already know that

$$|\psi\rangle = \sum_k c_k |u_k\rangle = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix}_{\mathcal{A}\text{-basis}} ,$$

which means (by our adjoint conversion rules)

$$\langle \psi | = \sum_k c_k^* \langle u_k | = (c_1^*, c_2^*, c_3^*, \dots, c_n^*)_{\mathcal{A}\text{-basis}} .$$

Finally, what's \mathcal{A} in its own eigenbasis? We know that any basis vector expressed in that basis' coordinates has a preferred basis look, $(1, 0, 0, 0, \dots)^t$, $(0, 1, 0, 0, \dots)^t$, etc. To that, add the definition of an eigenvector and eigenvalue

$$M\mathbf{u} = a\mathbf{u},$$

and you will conclude that, in its own eigenbasis, the matrix for \mathcal{A} is 0 everywhere except along its diagonal, which holds the eigenvalues,

$$\mathcal{A} = \begin{pmatrix} a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & \cdots & 0 \\ 0 & 0 & a_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_n \end{pmatrix}_{\mathcal{A}\text{-basis}}.$$

[Oops. I just gave away the answer to one of today's exercises. Which one?] We now have all our players in coordinate form, so

$$\begin{aligned} \langle \psi | \mathcal{A} | \psi \rangle &= (c_1^*, c_2^*, c_3^*, \dots, c_n^*) \begin{pmatrix} a_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & 0 & \cdots & 0 \\ 0 & 0 & a_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_n \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix} \\ &= (c_1^*, c_2^*, c_3^*, \dots, c_n^*) \begin{pmatrix} a_1 c_1 \\ a_2 c_2 \\ a_3 c_3 \\ \vdots \\ a_n c_n \end{pmatrix} \\ &= \sum_k |c_k|^2 a_k, \end{aligned}$$

which is the definition of $\langle \mathcal{A} \rangle_{|\psi\rangle}$. QED

7.13.5 Expectation Values in Spin-1/2 Systems

I'll do a few examples to demonstrate how this looks in our 2-dimensional world.

The Expectation Value of S_z Given the State $|+\rangle$

This is a great sanity check, since we know from **Trait #7** (the fifth postulate of QM) that S_z will always report a $+\frac{\hbar}{2}$ with certainty if we start in that state. Let's confirm it.

$$\langle + | S_z | + \rangle = (1, 0) \frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = +\frac{\hbar}{2}.$$

That was painless. Notice that this result is weaker than what we already knew from **Trait #7**. This is telling us that the *average* reading will approach the (+) eigenvalue in the long run, but in fact *every* measurement will be (+).

[**Exercise.** Show that the expectation value $\langle - | S_z | - \rangle = -\frac{\hbar}{2}$.]

The Expectation Value of S_x Given the State $|-\rangle_x$

Once again, we know the answer should be $-\frac{\hbar}{2}$, because we're starting in an eigenstate of S_x , but we will do the computation in the z -basis, which involves a wee-bit more arithmetic and will serve to give us some extra practice.

$$\begin{aligned} {}_x\langle - | S_x | - \rangle_x &= \frac{1}{\sqrt{2}}(1, -1) \frac{\hbar}{2} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \\ &= \frac{\hbar}{4}(1, -1) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \\ &= \frac{\hbar}{4}(1, -1) \begin{pmatrix} -1 \\ 1 \end{pmatrix} = -\frac{\hbar}{2}. \end{aligned}$$

[**Exercise.** Show that the expectation value ${}_x\langle + | S_x | + \rangle_x = +\frac{\hbar}{2}$.]

The Expectation Value of S_z Given the State $|-\rangle_y$

This time, it's not so obvious. However, we can guess. Since, the state

$$|-\rangle_y = \frac{|+\rangle - i|-\rangle}{\sqrt{2}}$$

we see that the probability for each outcome is 1/2. Over time half will result in an S_z measurement of $+\frac{\hbar}{2}$, and half will give us $-\frac{\hbar}{2}$, so the average should be close to 0. Let's verify that.

$$\begin{aligned} {}_y\langle - | S_z | - \rangle_y &= \frac{1}{\sqrt{2}}(1, +i) \frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -i \end{pmatrix} \\ &= \frac{\hbar}{4}(1, +i) \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -i \end{pmatrix} \\ &= \frac{\hbar}{4}(1, +i) \begin{pmatrix} 1 \\ i \end{pmatrix} = 0. \end{aligned}$$

[**Exercise.** Show that the expectation value ${}_y\langle + | S_z | + \rangle_y = 0$.]

7.14 It's About Time

We are done with all the math and physics necessary to do rigorous quantum computing at the basic level. We'll be adding a few lessons on math as the course progresses,

but for now you're ready to dive into the lectures on single qubit systems and early algorithms.

The next chapter is a completion of our quantum mechanics primer that covers the basics of *time evolution*, that is, it describes the laws by which quantum systems evolve over time. It isn't required for CS 83A, but you'll need it for the later courses in the sequence. You can skip it if you are so inclined or, if you "opt in" immediately, it'll provide the final postulates and traits that comprise a complete study of quantum formalism including the all important Schrödinger equation.

Whether you choose to go directly to the chapter on *qubits* or first learn the essentials of the *time dependent Schrödinger equation* you're in for a treat. Both subjects provide a sense of purpose and completion to all the hard work we've done up to this point.

Either way, it's about *time*.

Chapter 8

Time Dependent Quantum Mechanics

8.1 Time Evolution in Quantum Computing

Once a quantum system is put into a known state it will inevitably evolve over time. This might be a result of the natural laws of physics taking their toll on the undisturbed system or it may be that we are intentionally subjecting the system to a modifying force such as a quantum logic gate. In either case, the transformation is modeled by a linear operator that is *unitary*.

We've seen unitary matrices already, and even if you skip this chapter, you're equipped to go on to study qubits and quantum logic because the matrices and vectors involved do not have a time variable, t , in their respective coefficients - they are all *complex constants*.

But there are unitary transformation and quantum state vectors in which the elements themselves are functions of time, and it is that kind of evolution we will study today. It could represent the noise inherent in a system or it may just be a predictable change that results from the kind of hardware we are using.

8.2 The Hamiltonian

8.2.1 Total Energy

It turns out that the time evolution of a quantum system is completely determined by the *total energy* of the system (potential plus kinetic). There is a name for this quantity: *The Hamiltonian*. However the term is used differently in quantum mechanics than in classical mechanics, as we'll see in a moment.

8.2.2 From Classical Hamiltonian to Quantum Hamiltonian

We have to figure out a way to express the *energy* of a system \mathcal{S} using pen and paper so we can manipulate symbols, work problems and make predictions about how the system will look at 5 PM if we know how it started out at 8 AM. It sounds like a daunting task, but the 20th century physicists gave us a conceptually simple recipe for the process. The first step is to define a quantum operator – a matrix for our state space – that corresponds to the total energy. We'll call this recipe ...

Trait #10 (Constructing the Hamiltonian)

To construct an operator, H , that represents the energy of a quantum system:

1. *Express the classical energy, \mathcal{H} , formulaically in terms of basic classical concepts (e.g., position, momentum, angular momentum, etc.). You will have \mathcal{H} on the LHS of a formula and all the more basic terms on the RHS.*
2. *Replace the occurrence of the classical variables on the RHS by their (well known) quantum operators, and replace the symbol for classical energy, \mathcal{H} , on the LHS by its quantum symbol H .*

Vocabulary. Although the *total energy*, whether classical or quantum, is a scalar, the *Hamiltonian* in the quantum case is typically the *operator* associated with the (measurement of) that scalar, while the classical *Hamiltonian* continues to be synonymous with the scalar, itself. As you can see from the reading **Trait #10**, to distinguish the *classical* Hamiltonian, which works for macroscopic quantities, from the *quantum* Hamiltonian, we use \mathcal{H} for classical and H for quantum).

This is a bit hard to visualize until we do it. Also, it assumes we have already been told what operators are associated with the basic physical concepts on the RHS. As it happens, there are very few such basic concepts, and their quantum operators are well known. For example, 3-dimensional positional coordinates (x, y, z) correspond to three simple quantum operators X, Y and Z . Because I have not burdened you with the Hilbert space that models position and momentum, I can't give you a meaningful and short example using those somewhat familiar concepts, but in a moment you'll see every detail in our spin-1/2 Hilbert space, which is all we really care about.

8.3 The Hamiltonian for a Spin-1/2 System

8.3.1 A Classical Hamiltonian

Consider a stationary electron in a *constant* magnetic field. (**Warning:** this is not the Stern-Gerlach experiment since that requires a *non-homogeneous* magnetic-field, not to mention a moving electron.) Because the electron is stationary, all the energy is *potential* and depends only on the spin (direction and magnitude) of the electron

in relation to the magnetic field. (You may challenge that I forgot to account for the rotational kinetic energy, but an electron has no spatial extent, so there is no classical moment of inertia, and therefore no rotational kinetic energy.) We want to build a classical energy equation, so we treat spin as a classical 3-dimensional vector representing the *intrinsic angular momentum*, $(s_x, s_y, s_z)^t$. A *dot product* between this vector and the magnetic field vector, \mathbf{B} , expresses this potential energy and yields the following classical *Hamiltonian*

$$\mathcal{H} = -\gamma \mathbf{B} \cdot \mathbf{S},$$

where γ is a scalar known by the impressive name *gyromagnetic ratio* whose value is not relevant at the moment. (We are temporarily viewing the system as if it were classical in order to achieve step 1 in **Trait #10**, but please understand that it already has one foot in the quantum world simply by the inclusion of the scalar γ . Since scalars don't affect us, this apparent infraction doesn't disturb the process.)

Defining the z -Direction. The dot product only cares about the relationship between two vectors,

$$\mathbf{v} \cdot \mathbf{w} = vw \cos \theta,$$

where θ is the angle between them, so we can rotate the pair as a fixed assembly, that is, preserving angle θ . Therefore, let's establish a magnetic field (with magnitude B) pointing in the $+z$ -direction,

$$\mathbf{B} = B \hat{\mathbf{z}} = \begin{pmatrix} 0 \\ 0 \\ B \end{pmatrix},$$

and let the spin vector, \mathbf{S} , go along for the rotational ride. This does not produce a unique direction for the spin, but we only care about the polar angle θ , which we have constrained to remain unchanged. Equivalently, we can define the z -direction to be wherever our \mathbf{B} field points. Either way, we get a very neat simplification.

The classical spin has well-defined real-valued components s_x , s_y and s_z (not operators yet),

$$\mathbf{S} = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix},$$

and we use this vector to evaluate the dot product, above. Substituting gives

$$\begin{aligned} \mathcal{H} &= -\gamma \begin{pmatrix} 0 \\ 0 \\ B \end{pmatrix} \cdot \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} \\ &= -\gamma B s_z. \end{aligned}$$

This completes step 1 in **Trait #10**, and I can finally show you how easy it is to do step 2.

8.3.2 A Quantum Hamiltonian

We saw that (by a century of experimentation) the quantum operator corresponding to the classical z -component of spin is S_z . So we simply replace the classical s_z with our now familiar quantum operator operator S_z to get the quantum Hamiltonian,

$$H = -\gamma B S_z.$$

It's that simple.

Note that while S_z *may* change from moment-to-moment (we aren't sure yet), B and γ are constants, so the Hamiltonian is not time-dependent; it always gives an answer based solely on the relationship between \mathbf{S} and \mathbf{B} .

We now have a scalar relation between the Hamiltonian operator, H , and the z -spin projection operator, S_z , giving us a short-cut in our quest for H ; we need only substitute the matrix we computed earlier for S_z into this equation and get

$$H = -\gamma B \frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

8.4 The Energy Eigenkets

8.4.1 Relationship Between H and S_z

Because the operator H is a scalar multiple of our well studied S_z , we can easily find its eigenvectors (also known as the *energy eigenkets*) and eigenvalues; the eigenvectors are the same $|\pm\rangle$, and to get their respective eigenvalues we simply multiply those of S_z by $-\gamma B$. We'd better prove this, as it might not be obvious to everyone. We know that eigenvalue-eigenvector pairs for S_z are given by

$$\begin{aligned} S_z |+\rangle &= \left(\frac{\hbar}{2}\right) |+\rangle, & \left(|+\rangle \longleftrightarrow \frac{\hbar}{2}\right) & \text{ and} \\ S_z |-\rangle &= \left(-\frac{\hbar}{2}\right) |-\rangle, & \left(|-\rangle \longleftrightarrow -\frac{\hbar}{2}\right), & \end{aligned}$$

but we *now* know that S_z and B bear the scalar relationship,

$$S_z = \left(-\frac{1}{\gamma B}\right) H.$$

Substituting this into the S_z eigenvector expressions gives

$$\begin{aligned} \left(-\frac{1}{\gamma B}\right) H |+\rangle &= \left(\frac{\hbar}{2}\right) |+\rangle & \text{ and} \\ \left(-\frac{1}{\gamma B}\right) H |-\rangle &= \left(-\frac{\hbar}{2}\right) |-\rangle, \end{aligned}$$

or

$$\begin{aligned} H|+\rangle &= \left(-\frac{\gamma B\hbar}{2}\right)|+\rangle \quad \text{and} \\ H|-\rangle &= \left(\frac{\gamma B\hbar}{2}\right)|-\rangle . \end{aligned}$$

But this says that H has the *same two eigenvectors* as S_z , only they are associated with different eigenvalues,

$$\begin{aligned} |+\rangle &\longleftrightarrow -\frac{\gamma B\hbar}{2} , \\ |-\rangle &\longleftrightarrow +\frac{\gamma B\hbar}{2} . \end{aligned}$$

If we measure the energy of the system, we will get $-\frac{\gamma B\hbar}{2}$ if the electron collapses into the $|+\rangle$ state, pointing *as close as quantum mechanics allows* toward the $(+z)$ -axis, the direction of the \mathbf{B} -field. Meanwhile, if it collapses into the $|-\rangle$ state, we will get $+\frac{\gamma B\hbar}{2}$, pointing *as far as quantum mechanics allows* from $(+z)$ -axis, opposite the direction of the \mathbf{B} -field.

Does this make sense?

Yes. When a magnetic dipole (which is what electron spin represents, discounting the gyromagnetic ratio γ) is pointing in the direction of a magnetic field, energy is minimum. Imagine a compass needle pointing magnetic north. The potential energy of the compass-Earth system is at its *minimum*: It takes no energy to maintain that configuration. However, if the dipole is pointing *opposite* the direction of the magnetic field, energy is at its *maximum*: Imagine the compass needle pointing magnetic south. Now it takes energy to hold the needle in place. The potential energy of the compass-Earth system is maximum. The fact that the energy measurement is negative for the $|+\rangle$ but positive for the $|-\rangle$ faithfully represents physical reality.

Also, note that for a spin-1/2 system, any state vector, $|\psi\rangle$, has the same coordinate expression whether we expand it along the eigenkets of S_z or those of H , as they are the same two kets, $|+\rangle$ and $|-\rangle$.

8.4.2 Allowable Energies

Knowledge of the energy eigenvectors is essential to predicting time evolution of any state, as we are about to see. But first, we apply **Trait #3** (*the third postulate of QM: allowable eigenvalues of an observable*) to the Hamiltonian to make a trait-worthy observation that is universal in quantum physics.

Trait #11 (Quantization of Energy)

The only allowable (i.e., measurable) energies of a quantum system are the eigenvalues of the Hamiltonian.

Let's take a short side-trip to give the crucial postulate that expresses, in full generality, how *any* quantum state evolves based on the system's Hamiltonian operator, H .

8.5 Sixth Postulate of Quantum Mechanics

8.5.1 The Schrödinger Equation

The time-dependent Schrödinger Equation is the quantum-mechanical answer to the question, "How does a state evolve over time?" To start, we replace a general (but fixed) state with one which is a function of time,

$$|\psi\rangle \longrightarrow |\psi(t)\rangle .$$

As a consequence, its expansion coefficients will also be complex scalar *functions* of time.

$$|\psi(t)\rangle = \sum_{k=1}^n c_k(t) |u_k\rangle .$$

Everything we did earlier still holds if we freeze time at some t' . We would then evaluate the system at that instant as if it were not time-dependent and we were working with the fixed state

$$\begin{aligned} |\psi\rangle &= |\psi(t')\rangle = \sum_{k=1}^n c'_k |u_k\rangle , \\ c'_k &\equiv c_k(t') . \end{aligned}$$

To get to time $t = t'$ (or any future $t > 0$), though, we need to know the exact formula for those coefficients, $c_k(t)$, so we can plug-in $t = t'$ and produce this fixed state. That's where the *sixth postulate of quantum mechanics* comes in.

Trait #12 (The Time-Dependent Schrödinger Equation)

The time evolution of a state vector is governed by the Schrödinger Equation

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H(t) |\psi(t)\rangle .$$

Notice that the Hamiltonian can change from moment-to-moment, although it does not always do so. For our purposes, it is not time dependent, so we get a simpler form of the Schrödinger Equation,

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle .$$

This is still a time-dependent equation; we merely have a simplification acknowledging that t does not appear in the matrix for H .

The “Other” Schrödinger Equation? In case you’re wondering whether there’s a time-*independent* Schrödinger equation, the answer is, it depends on whom you ask. Purists say, not really, but most of us consider the eigenket-eigenvalue equation of **Trait #3**,

$$T_{\mathcal{A}} |u_k\rangle = a_k |u_k\rangle ,$$

when applied to the operator $\mathcal{A} = H$, to be the *time-independent Schrödinger equation*.

Solving the time-*independent* Schrödinger equation, which produces the energy eigenkets and eigenvalues, is typically the first step in the process of solving the more general time-*dependent* Schrödinger equation.

8.5.2 The Evolution of Spin in a Constant Magnetic Field

You are ready to solve your first Schrödinger equation. We consider the system of a stationary electron in a uniform z -up directed \mathbf{B} -field, whose Hamiltonian we have already “cracked” (meaning we solved the time-*independent* Schrödinger equation).

Because it’s so exciting, I’m going to summarize the set-up for you. The players are

- our state vector, $|\psi(t)\rangle$, with its two expansion coefficients, the unknown functions $c_1(t)$ and $c_2(t)$,

$$|\psi(t)\rangle = c_1(t) |+\rangle + c_2(t) |-\rangle = \begin{pmatrix} c_1(t) \\ c_2(t) \end{pmatrix}_z ,$$

- the Hamiltonian operator,

$$H = -\gamma B \frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} ,$$

- and the Schrödinger equation that relates the two,

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H |\psi(t)\rangle .$$

Let’s compute. Substitute the coordinate functions in for $|\psi\rangle$ in the Schrödinger equation,

$$i\hbar \frac{d}{dt} \begin{pmatrix} c_1(t) \\ c_2(t) \end{pmatrix} = -\gamma B \frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} c_1(t) \\ c_2(t) \end{pmatrix} .$$

This is equivalent to

$$\begin{pmatrix} i\hbar \frac{d}{dt} c_1 \\ i\hbar \frac{d}{dt} c_2 \end{pmatrix} = \begin{pmatrix} -\gamma B \frac{\hbar}{2} c_1 \\ \gamma B \frac{\hbar}{2} c_2 \end{pmatrix},$$

or

$$\begin{aligned} \frac{d}{dt} c_1 &= \frac{\gamma B i}{2} c_1 \quad \text{and} \\ \frac{d}{dt} c_2 &= -\frac{\gamma B i}{2} c_2. \end{aligned}$$

From calculus we know that the solutions to

$$\frac{dx}{dt} = kx,$$

with constant k , are the family of equations

$$x(t) = C e^{kt},$$

one solution for each complex constant C . (If you didn't know that, you can verify it now by differentiating the last equation.) The constant C is determined by the initial condition, at time $t = 0$,

$$C = x(0).$$

We have two such differential equations, one for each constant

$$k = \pm \frac{\gamma B i}{2},$$

so

$$\begin{aligned} c_1(t) &= C_1 e^{it(\gamma B/2)}, \\ c_2(t) &= C_2 e^{-it(\gamma B/2)}. \end{aligned}$$

The initial condition is that

$$|\psi(0)\rangle = |\psi\rangle,$$

our starting state. In other words, the initial conditions for our two equations are

$$\begin{aligned} c_1(0) &= \alpha_0 \quad \text{and} \\ c_2(0) &= \beta_0, \end{aligned}$$

where we are saying α_0 and β_0 are the two scalar coefficients of the state $|\psi\rangle$ at time $t = 0$. That gives us the constants

$$\begin{aligned} C_1 &= \alpha_0, \\ C_2 &= \beta_0, \end{aligned}$$

the complete formulas for the time-dependent coefficients,

$$\begin{aligned}c_1(t) &= \alpha_0 e^{it(\gamma B/2)}, \\c_2(t) &= \beta_0 e^{-it(\gamma B/2)}.\end{aligned}$$

and, finally, our time-dependent state in its full glory,

$$|\psi(t)\rangle = \alpha_0 e^{it(\gamma B/2)} |+\rangle + \beta_0 e^{-it(\gamma B/2)} |-\rangle.$$

We have solved the Schrödinger equation using first quarter calculus.

If you can manage to not be too confused, let's reduce our notation by using c_1 and c_2 (without the parameter (t)), rather than α_0 and β_0 , to mean the initial coefficients at time $t = 0$ giving us the slightly cleaner

$$|\psi(t)\rangle = c_1 e^{it(\gamma B/2)} |+\rangle + c_2 e^{-it(\gamma B/2)} |-\rangle.$$

We pause to consider how this can be generalized to any situation (in the case of finite or enumerable eigenvalues). I'll introduce an odd notation that physicists have universally adopted, namely that the eigenket associated with eigenvalue a , will be that same a inside the ket symbol, i.e., $|a\rangle$.

1. We first solved the system's *Hamiltonian* – the time-*in*dependent Schrödinger equation, if you like – to get the allowable energies, $\{E_k\}$ and their associated eigenkets, $\{|E_k\rangle\}$.
2. Next, we expanded the initial state, $|\psi\rangle$, along the energy basis,

$$|\psi\rangle = \sum_k c_k |E_k\rangle.$$

3. We solved the Schrödinger equation for each time-dependent amplitude, $c_k(t)$, which yielded

$$c_k(t) = c_k e^{-itE_k/\hbar}.$$

4. Finally, we “attached” the exponentials as factors to each of the terms in the original sum for $|\psi\rangle$ to get the full, time-dependent state,

$$|\psi(t)\rangle = \sum_k c_k e^{-itE_k/\hbar} |E_k\rangle.$$

This will be packaged into a *trait* in a minute. But first ...

8.5.3 Stationary States

Notice what this implies if our initial state happens to be one of the energy eigenstates. In our spin-1/2 system that would be either $|+\rangle$ or $|-\rangle$. Take the $|\psi\rangle = |+\rangle$. The result is

$$|\psi(t)\rangle = e^{it(\gamma B/2)} |+\rangle ,$$

Introducing the shorthand $\phi_t \equiv \gamma Bt/2$, the time evolution merely causes a “phase factor” of $e^{i\phi_t}$ to appear. But remember that our state-space does not differentiate between scalar multiples of state vectors, so

$$|\psi(t)\rangle = e^{i\phi_t} |+\rangle \cong |+\rangle = |\psi\rangle .$$

The state does not change over time.

Looked at another way, if you start out in state $|+\rangle$ and measure total energy, H , you get $-\frac{\gamma B\hbar}{2}$ with certainty (which we already knew, since the coefficient of $|+\rangle$ is 1, and $1^2 = 1$). This also means that the state into which $|+\rangle$ evolves over time t , also has 100% probability of remaining $|+\rangle$ as the coefficient of the $|+\rangle$ reveals:

$$|c_1(t)|^2 = c_1(t)^* c_1(t) = e^{-it(\gamma B/2)} e^{it(\gamma B/2)} = e^0 = 1 .$$

Its one and only expansion coefficient changes by a factor of $e^{i\phi_t}$ whose square magnitude is 1 regardless of t . This is big enough to call a *trait*.

Trait #13 (Stationary States)

An eigenstate of the Hamiltonian operator evolves in such a way that its measurement outcome does not change; it remains in the same eigenstate.

For this reason, eigenstates are often called *stationary states* of the system.

8.5.4 General (Non-Stationary) States

Does this mean that we can throw away the phase factors $e^{i\phi_t}$ for times $t \neq 0$? It may surprise you that the answer is a big fat **no**. Let's take the example of an initial state that is not an energy eigenket. One of the y -eigenstates will suffice,

$$|\psi\rangle \equiv |-\rangle_y = \frac{|+\rangle - i |-\rangle}{\sqrt{2}} .$$

The two initial coefficients are

$$\begin{aligned} c_1 &= \frac{1}{\sqrt{2}} \quad \text{and} \\ c_2 &= -\frac{i}{\sqrt{2}} . \end{aligned}$$

Allow this state to evolve for a time t according to the Schrödinger equation,

$$|\psi(t)\rangle = \frac{e^{it(\gamma B/2)} |+\rangle - i e^{-it(\gamma B/2)} |-\rangle}{\sqrt{2}}.$$

The state is unchanged if we multiply by an overall scalar multiple, so let's turn the $|+\rangle$ coefficient *real* by multiplying the entire fraction by its complex conjugate $e^{-it(\gamma B/2)}$, giving the equivalent state

$$|\psi(t)\rangle = \frac{|+\rangle - i e^{-it(\gamma B)} |-\rangle}{\sqrt{2}},$$

whose coefficients are

$$\begin{aligned} c_1(t) &= \frac{1}{\sqrt{2}} \quad \text{and} \\ c_2(t) &= -\frac{i e^{-it(\gamma B)}}{\sqrt{2}}. \end{aligned}$$

Comparing this to the original state, we see that while the coefficient of $|+\rangle$ never changes, the coefficient of $|-\rangle$ changes depending on t . To dramatize this, consider time $t' = \pi/(\gamma B)$ (an impossibly tiny fraction of a second if you were to compute it using real values for γ and a typical **B**-field).

$$\begin{aligned} |\psi(t')\rangle &= \frac{|+\rangle - i e^{-i\pi} |-\rangle}{\sqrt{2}} = \frac{|+\rangle - i(-1)|-\rangle}{\sqrt{2}} \\ &= \frac{|+\rangle + i |-\rangle}{\sqrt{2}} = |+\rangle_y. \end{aligned}$$

We started out in state $|-\rangle_y$, and a fraction of a second later found ourselves in state $|+\rangle_y$. That's a pretty dramatic change. If we were to measure S_y initially, we would, with certainty, receive a $-\hbar/2$ reading. Wait a mere $\pi/(\gamma B)$ seconds later, and a measurement would result, with equal certainty, in the opposite value $+\hbar/2$.

8.5.5 General Technique for Computing Time-Evolved States

We've set the stage for a technique that applies throughout quantum mechanics. We'll call it a *trait*.

Trait #14 (Evolution of Any Observable)

To determine the time-evolved probability of the outcome of any observable, \mathcal{A} , starting in the initial state $|\psi\rangle$,

1. *compute the Energy eigenvalues and eigenkets for the system, $\{E_k \leftrightarrow |E_k\rangle\}$, by solving the “time-independent” Schrödinger equation, $H|E_k\rangle = E_k|E_k\rangle$,*

2. expand $|\psi\rangle$ along the energy basis: $|\psi\rangle = \sum_k c_k |E_k\rangle$,
3. attach (as factors) the time-dependent phase factors, $e^{-itE_k/\hbar}$ to each term, $|\psi(t)\rangle = \sum_k c_k e^{-itE_k/\hbar} |E_k\rangle$,
4. “dot” this expression with the desired eigenket, $|u_j\rangle$ of \mathcal{A} , to find its amplitude, $\alpha_j(t) = \langle u_j | \psi(t) \rangle$, and
5. the magnitude-squared of this amplitude, $|\alpha_j(t)|^2$, will be the probability of an \mathcal{A} -measurement producing the eigenvalue a_k at time t .

The short version is that for any observable of interest, \mathcal{A} , you first solve the system’s time-*independent* Schrödinger equation and use its energy eigenbasis to express your state, $|\psi\rangle$. Incorporate the time dependence into that expression and “dot” that with the desired eigenstate of \mathcal{A} to get your amplitude and, ultimately, probability.

(I keep using quotes with the verb “dot”, because this is really an *inner*-product, rather than a real *dot*-product, requiring the left vector’s coordinates to be conjugated.)

Example

We continue to examine the evolution of an electron that starts in the y -down state, $|-\rangle_y$. We’ve already done the first three steps of **Trait #14** and found that after time t the state $|-\rangle_y$ evolves to

$$|(-)_t\rangle_y = \frac{|+\rangle - i e^{-it(\gamma B)} |-\rangle}{\sqrt{2}}.$$

(I used $|(-)_t\rangle_y$, rather than the somewhat confusing $|-(t)\rangle_y$, to designate the state’s dependence on time.)

That’s the official answer to the question “how does $|-\rangle_y$ evolve?”, but to see how we would use this information, we have to pick an observable we are curious about and apply **Trait #14**, steps 4 and 5.

Let’s ask about S_y , the y -projection of spin – specifically the probability of measuring a $|+\rangle_y$ at time t . Step 4 says to “dot” the time-evolved state with the vector $|+\rangle_y$, so the amplitude (step 4) is

$$c_{+y} = {}_y\langle + | (-)_t \rangle_y.$$

I’ll help you read it: the “left” vector of the inner product is the $+\frac{\hbar}{2}$ eigenket of the operator S_y , $|+\rangle_y$, independent of time. The “right” vector of the inner product is our starting state, $|-\rangle_y$, but evolved to a later time, t .

Because everything is expressed in terms of the z -basis, we have to be sure we stay in that realm. The z -coordinates of $|+\rangle_y$ are obtained from our familiar

$$|+\rangle_y = \frac{|+\rangle + i |-\rangle}{\sqrt{2}} = \begin{pmatrix} 1/\sqrt{2} \\ i/\sqrt{2} \end{pmatrix}_z.$$

I added the subscript z on the RHS to emphasize that we are displaying the vector $|+\rangle_y$ in the z -coordinates, as usual. If we are to use this on the left side of a complex inner product we have to take the conjugate of all components. This is easy to see in the coordinate form,

$${}_y\langle+| = (1/\sqrt{2}, -i/\sqrt{2})_z,$$

but let's see how we can avoid looking inside the vector by applying our *adjoint conversion rules* to the expression defining $|+\rangle_y$ to create a *bra* for this vector. I'll give you the result, and you can supply the (very few) details as an ...

[Exercise. Show that

$${}_y\langle+| = (|+\rangle_y)^\dagger = \frac{\langle+| - i \langle-|}{\sqrt{2}}.]$$

Getting back to the computation of the amplitude, c_{+y} , substitute the computed values into the inner product to get

$$\begin{aligned} c_{+y} &= \left(\frac{\langle+| - i \langle-|}{\sqrt{2}} \right) \left(\frac{|+\rangle - i e^{-it(\gamma B)} |-\rangle}{\sqrt{2}} \right) \\ &= \left(\frac{\langle+|+\rangle - e^{-i\gamma Bt} \langle-|-\rangle}{2} \right) = \frac{1 - e^{-i\gamma Bt}}{2}. \end{aligned}$$

(The last two equalities made use of the orthonormality of any observable eigenbasis (**Trait #4**).)

Finally, step 5 says that the probability of measuring $S_y = +\frac{\hbar}{2}$ at any time t is

$$|c_{+y}|^2 = c_{+y}^* c_{+y} = \left(\frac{1 - e^{i\gamma Bt}}{2} \right) \left(\frac{1 - e^{-i\gamma Bt}}{2} \right),$$

where we used the fact (see the complex number lecture) that, for real θ ,

$$(e^{-i\theta})^* = e^{i\theta}.$$

Let's simplify the notation by setting

$$\theta \equiv \gamma Bt$$

and complete the computation with that substitution,

$$\begin{aligned} |c_{+y}|^2 &= \frac{2 - e^{i\theta} - e^{-i\theta}}{4} \\ &= \frac{1}{2} - \frac{1}{2} \left(\frac{e^{i\theta} + e^{-i\theta}}{2} \right) \\ &= \frac{1}{2} - \frac{1}{2} \cos \theta. \end{aligned}$$

Undoing the substitution gives us the final result

$$P\left(S_y(t) = +\frac{\hbar}{2}\right) = \frac{1}{2} - \frac{1}{2} \cos(\gamma B t).$$

As you can see, the probability of measuring an up- y state oscillates between 0 and 1 sinusoidally over time. Note that this is consistent with our initial state at time $t = 0$: $\cos 0 = 1$, so the probability of measuring $+\frac{\hbar}{2}$ is zero; it *had* to be since we started in state $|-\rangle_y$, and when you are in an eigenstate ($|-\rangle_y$), the measurement of the observable corresponding to that eigenstate (S_y) is guaranteed to be the eigenstate's eigenvalue ($-\frac{\hbar}{2}$). Likewise, if we test precisely at $t = \pi/(\gamma B)$, we get $\frac{1}{2} - \frac{1}{2}(-1) = 1$, a certainty that we will detect $+\frac{\hbar}{2}$, the $|+\rangle_y$ eigenvalue.

We can stop the clock at times between those two extremes to get any probability we like.

[Exercise.] What is the probability of measuring $S_y(t) = +\frac{\hbar}{2}$ at the (chronologically ordered) times

- (a) $t = \pi/(6\gamma B)$,
- (b) $t = \pi/(4\gamma B)$,
- (c) $t = \pi/(3\gamma B)$,
- (d) $t = \pi/(2\gamma B)$.]

[Exercise.] Do the same analysis to get the probability that S_y measured at time t will be $-\frac{\hbar}{2}$. Confirm that at *any* time t , the two probabilities add to 1.]

8.6 Larmor Precession

We complete this lecture by combining *expectation value* with *time evolution* to get a famous result which tells us how we can relate the 3-dimensional real vector of a classical angular momentum vector to the quantum spin-1/2 electron, which is a 2-dimensional complex vector.

8.6.1 The Time-Evolved Spin State in a Uniform B-Field

We assume that we have many electrons in the same initial spin state, and we look at that state's expansion along the z -basis,

$$|\psi\rangle = c_1 |+\rangle + c_2 |-\rangle,$$

where, by normalization we know that

$$|c_1|^2 + |c_2|^2 = 1.$$

We let the state (of any one of these systems, since they are all the same) evolve for a time t . We have already solved the Schrödinger equation and found that the evolved state at that time will be

$$\begin{aligned} |\psi(t)\rangle &= c_1 e^{it(\gamma B/2)} |+\rangle + c_2 e^{-it(\gamma B/2)} |-\rangle \\ &= \begin{pmatrix} c_1 e^{it(\gamma B/2)} \\ c_2 e^{-it(\gamma B/2)} \end{pmatrix}. \end{aligned}$$

This, then is the state at time t , prior to measurement.

Rewriting $|\psi(t)\rangle$

It will help to represent this state by an equivalent vector that is a mere unit scalar multiple of itself. To do that, we first express c_1 and c_2 in *polar form*,

$$\begin{aligned} c_1 &= c e^{i\phi_1} & \text{and} \\ c_2 &= s e^{i\phi_2}, \end{aligned}$$

giving the equivalent state

$$|\psi(t)\rangle = \begin{pmatrix} c e^{i\phi_1} e^{it(\gamma B/2)} \\ s e^{i\phi_2} e^{-it(\gamma B/2)} \end{pmatrix}.$$

Then we multiply by the unit scalar $e^{-i(\frac{\phi_1+\phi_2}{2})}$ to get a more balanced equivalent state,

$$|\psi(t)\rangle = \begin{pmatrix} c e^{i(\frac{\phi_1-\phi_2}{2})} e^{it(\gamma B/2)} \\ s e^{-i(\frac{\phi_1-\phi_2}{2})} e^{-it(\gamma B/2)} \end{pmatrix}.$$

Now, we simplify by making the substitutions

$$\begin{aligned} \omega &= \gamma B & \text{and} \\ \phi_0 &= \phi_1 - \phi_2, \end{aligned}$$

to get the simple and balanced Hilbert space representative of our state,

$$|\psi(t)\rangle = \begin{pmatrix} c e^{i\phi_0/2} e^{it\omega/2} \\ s e^{-i\phi_0/2} e^{-it\omega/2} \end{pmatrix} = \begin{pmatrix} c e^{i(t\omega + \phi_0)/2} \\ s e^{-i(t\omega + \phi_0)/2} \end{pmatrix}.$$

We get a nice simplification by using the notation

$$\phi(t) \equiv \frac{\omega t + \phi_0}{2},$$

to express our evolving state very concisely as

$$|\psi(t)\rangle = \begin{pmatrix} c e^{i\phi(t)} \\ s e^{-i\phi(t)} \end{pmatrix}.$$

A Convenient Angle

There is one last observation before we start to compute. Since $|\psi\rangle$ is normalized,

$$|c|^2 + |s|^2 = 1,$$

the amplitudes c and s have moduli (absolute values) that are consistent with the sine and cosine of some angle. Furthermore, we can name that angle anything we like. Call it “ $\theta/2$ ” for reasons that will become clear in about 60 seconds. [**Start of 60 seconds.**]

We have proclaimed the angle θ to be such that

$$\begin{aligned} c &= \cos \frac{\theta}{2} & \text{and} \\ s &= \sin \frac{\theta}{2}, \end{aligned}$$

which is why I named the amplitudes c and s .

Also, we’re going to run into the two expressions cs and $c^2 - s^2$ a little later, so let’s see if we can write those in terms of our angle θ . The *addition law of sines* implies that

$$cs = \cos \frac{\theta}{2} \sin \frac{\theta}{2} = \frac{\sin \theta}{2},$$

while the *addition law of cosines* yields

$$c^2 - s^2 = \cos^2 \frac{\theta}{2} - \sin^2 \frac{\theta}{2} = \cos \theta.$$

By letting $\theta/2$ be the common angle that we used to represent c and s (instead of, say, θ) we ended up with plain old θ on the RHS of these formulas, which is the form we’ll need. [**End of 60 seconds.**]

Although we’ll start out using c and s for the moduli of $|\psi\rangle$ ’s amplitudes, we’ll eventually want to make these substitutions when the time comes. The angle θ will have a geometric significance.

8.6.2 Evolution of the Spin Expectation Values

Experiment #4: Measuring $|\psi(t)\rangle$ at Time t

We ask the physicists to measure one of S_x , S_y or S_z at time t . Now they can only measure *one* of those observables per electron, because once they do, $|\psi(t)\rangle$ will collapse into one of the six basis states, $|\pm\rangle$, $|\pm\rangle_x$ or $|\pm\rangle_y$, after which time, it becomes useless. But that’s okay; we’re swimming in $|\psi(t)\rangle$ electrons. We pick a very large number, N , (say N is a million). We test $3N$ electrons, measuring S_z on the first N , S_x on the second N , and S_y on the third N . We will have measured the state

$|\psi(t)\rangle$ $3N$ or 3 million times. The physicists record the measurements producing a certain number of $+z$ values, a certain number of $-z$, etc. We ask them to compute the average of the S_z results – a number between $-\frac{\hbar}{2}$ and $+\frac{\hbar}{2}$, and the same with the S_x and S_y results.

So they'll have three numbers in the end, \overline{m}_x , \overline{m}_y and \overline{m}_z .

But hold on a second. We don't have to bother the physicists, because when N is large, we know from the *the law of large numbers* that the average values of each of the three spin projections are approximated very closely by the *expectation values of the operators*. So, let's compute those instead of wasting a lot of time and money.

The Expectation Values at Time t

We do this for each observable S_z , S_x and S_y , individually, then find a way to combine the answers. We'll begin with $\langle S_z(t) \rangle$.

A. The Expectation Value for the z -spin Observable: $\langle S_z(t) \rangle$

Trait #9 (*the expectation value theorem*), tells us that we can compute this using

$$\langle \psi(t) | S_z | \psi(t) \rangle .$$

With the help of **Trait #8** (*the adjoint conversion rules*) and keeping in mind that c and s are real, we find

$$\begin{aligned} \langle \psi(t) | S_z | \psi(t) \rangle &= (c e^{-i\phi(t)}, s e^{i\phi(t)}) \frac{\hbar}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} c e^{i\phi(t)} \\ s e^{-i\phi(t)} \end{pmatrix} \\ &= (c e^{-i\phi(t)}, s e^{i\phi(t)}) \frac{\hbar}{2} \begin{pmatrix} c e^{i\phi(t)} \\ -s e^{-i\phi(t)} \end{pmatrix} \\ &= \frac{\hbar}{2} (c^2 - s^2) = \frac{\hbar}{2} \cos \theta . \end{aligned}$$

We can draw some quick conclusions from this (and subtler ones later).

1. The expectation value of S_z does not change with time.
2. If $|\psi\rangle = |+\rangle$, i.e., $c = 1$ and $s = 0$, the expectation value is $+\frac{\hbar}{2}$, as it must, since $|+\rangle$ is a stationary state and so *always* yields a $(+)$ measurement.
3. If $|\psi\rangle = |-\rangle$, i.e., $c = 0$ and $s = 1$, the expectation value is $-\frac{\hbar}{2}$, again consistent with $|-\rangle$ being the *other* stationary state, the one that always yields a $(-)$ measurement.

B. The Expectation Value for the x -spin Observable: $\langle S_x(t) \rangle$

We compute

$$\langle \psi(t) | S_x | \psi(t) \rangle .$$

Using our adjoint conversion rules again, we find

$$\begin{aligned} \langle \psi(t) | S_x | \psi(t) \rangle &= (c e^{-i\phi(t)}, s e^{i\phi(t)}) \frac{\hbar}{2} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c e^{i\phi(t)} \\ s e^{-i\phi(t)} \end{pmatrix} \\ &= (c e^{-i\phi(t)}, s e^{i\phi(t)}) \frac{\hbar}{2} \begin{pmatrix} s e^{-i\phi(t)} \\ c e^{i\phi(t)} \end{pmatrix} \\ &= \frac{\hbar}{2} (cs e^{-2i\phi(t)} + cs e^{2i\phi(t)}) . \end{aligned}$$

This is nice, but a slight rearrangement should give you a brilliant idea,

$$\langle \psi(t) | S_x | \psi(t) \rangle = cs \hbar \frac{e^{-2i\phi(t)} + e^{2i\phi(t)}}{2} .$$

Look back at our lesson on complex numbers, especially the consequences of the Euler formula, and you'll discover that the fraction simplifies to $\cos(2\phi(t))$. Now we have

$$\langle \psi(t) | S_x | \psi(t) \rangle = cs \hbar \cos(2\phi(t)) ,$$

which, after undoing our substitutions for cs and $\phi(t)$ we set up in the *convenient angle* section, looks like

$$\langle \psi(t) | S_x | \psi(t) \rangle = cs \hbar \cos(\omega t + \phi_0) = \frac{\hbar}{2} \sin \theta \cos(\omega t + \phi_0) .$$

Observe these consequences.

1. The expectation value of S_x varies sinusoidally with time at a frequency $\omega = \gamma B$, not counting the situation in item 2.
2. If $|\psi\rangle$ is either $|+\rangle$ or $|-\rangle$, then $cs = 0$ producing an expectation value of 0 at all times. This is consistent with the following two facts.
 - These two z -eigenkets, when expressed in the x -basis, have equal “doses” ($\frac{1}{\sqrt{2}}$) of $|+\rangle_x$ and $|-\rangle_x$, as you can tell from

$$|\pm\rangle = \frac{|+\rangle_x \pm |-\rangle_x}{\sqrt{2}} ,$$

so we would expect a roughly equal collapse into the $|+\rangle_x$ and $|-\rangle_x$ states, averaging to 0.

- We've already established that the two kets $|\pm\rangle$ are *stationary states* of H , so whatever holds at time $t = 0$, holds for all time.

C. The Expectation Value for the y -spin Observable: $\langle S_y(t) \rangle$

We compute

$$\langle \psi(t) | S_y | \psi(t) \rangle ,$$

The calculation proceeds much like that of $\langle S_x \rangle$.

$$\begin{aligned} \langle \psi(t) | S_y | \psi(t) \rangle &= \begin{pmatrix} c e^{-i\phi(t)} & s e^{i\phi(t)} \end{pmatrix} \frac{\hbar}{2} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} c e^{i\phi(t)} \\ s e^{-i\phi(t)} \end{pmatrix} \\ &= \begin{pmatrix} c e^{-i\phi(t)} & s e^{i\phi(t)} \end{pmatrix} \frac{\hbar}{2} \begin{pmatrix} -i s e^{-i\phi(t)} \\ i c e^{i\phi(t)} \end{pmatrix} \\ &= \frac{\hbar}{2i} (c s e^{-2i\phi(t)} - c s e^{2i\phi(t)}) \end{aligned}$$

Rearranging and applying one of our Euler formulas, we find

$$\begin{aligned} \langle \psi(t) | S_y | \psi(t) \rangle &= c s \hbar \frac{e^{-2i\phi(t)} - e^{2i\phi(t)}}{2i} \\ &= -c s \hbar \sin(2\phi(t)) \\ &= -c s \hbar \sin(\omega t + 2\phi_0) = -\frac{\hbar}{2} \sin \theta \sin(\omega t + \phi_0) . \end{aligned}$$

Again, some observations.

1. The expectation value of S_y varies sinusoidally with time at a frequency $\omega = \gamma B$, not counting the situation in item 2.
2. If $|\psi\rangle$ is either $|+\rangle$ or $|-\rangle$, then $cs = 0$ yielding an expectation value of 0 for all time. Again, this is consistent with these two z -eigenkets, being stationary states and having y -basis amplitudes of equal magnitude ($\frac{1}{\sqrt{2}}$),

$$\begin{aligned} |+\rangle &= \frac{|+\rangle_y + |-\rangle_y}{\sqrt{2}} \quad \text{and} \\ |-\rangle &= \frac{|+\rangle_y - |-\rangle_y}{i\sqrt{2}} , \end{aligned}$$

(from a prior exercise).

The Expectation Vector and Larmor Precession

In classical mechanics, *Larmor precession* describes the way in which a magnetic dipole's axis – a real 3-dimensional vector – revolves about a magnetic field vector.

In contrast, the quantum mechanical spin state-vector lives in a 2-dimensional Hilbert space, not 3-dimensional real space, so we don't have simultaneously measurable x , y , and z -components which we can study. However, we *can* define a real (and evolving) 3-dimensional vector $\mathbf{s}(t)$ to be

$$\mathbf{s}(t) \equiv \begin{pmatrix} \langle S_x \rangle_{|\psi(t)\rangle} \\ \langle S_y \rangle_{|\psi(t)\rangle} \\ \langle S_z \rangle_{|\psi(t)\rangle} \end{pmatrix}.$$

This $\mathbf{s}(t)$ is a true 3-dimensional (time-dependent) vector whose real coordinates are the three expectation values, $\langle S_x \rangle$, $\langle S_y \rangle$ and $\langle S_z \rangle$, at time, t .

In the previous section we showed that

$$\mathbf{s}(t) = \begin{pmatrix} \frac{\hbar}{2} \sin \theta \cos(\omega t + \phi_0) \\ -\frac{\hbar}{2} \sin \theta \sin(\omega t + \phi_0) \\ \frac{\hbar}{2} \cos \theta \end{pmatrix} = \frac{\hbar}{2} \begin{pmatrix} \sin \theta \cos(\omega t + \phi_0) \\ -\sin \theta \sin(\omega t + \phi_0) \\ \cos \theta \end{pmatrix}.$$

If this is not speaking to you, drop the factor of $\hbar/2$ and set $\phi(t) = \omega t + \phi_0$. What we get is the 3-dimensional vector

$$\mathbf{s}(t) \propto \begin{pmatrix} \sin \theta \cos \phi(t) \\ -\sin \theta \sin \phi(t) \\ \cos \theta \end{pmatrix}.$$

It is a unit vector in \mathbb{R}^3 whose spherical coordinates are $(1, \theta, \phi(t))$, i.e., it has a polar angle θ and azimuthal angle $\phi(t)$. (I don't use exclamation points, but if I did, I would use one here.) We are looking at a vector that has a fixed z -coordinate, but whose x and y -coordinates are in a clockwise circular orbit around the origin (*clockwise* because of the y -coordinate's minus sign – **[Exercise]**). This is called *precession*. Since our \mathbf{B} -field was defined to point in the $+z$ direction, we have discovered the meaning of the vector $\mathbf{s}(t) = (\langle S_x(t) \rangle, \langle S_y(t) \rangle, \langle S_z(t) \rangle)^t$.

This is something that recurs with regularity in quantum physics. The quantum state vectors, themselves, do not behave like classical vectors. However, their expectation values do.

8.6.3 Summary of Larmor Precession

When a motionless electron is initially in spin state

$$|\psi\rangle = \begin{pmatrix} c e^{i\phi_1} \\ s e^{i\phi_2} \end{pmatrix}.$$

within a uniform magnetic field \mathbf{B} pointing in the $+z$ direction, the Schrödinger equation tells us that its *expectation value vector*, $\mathbf{s}(t)$, evolves according to the formula

$$\mathbf{s}(t) = \frac{\hbar}{2} \begin{pmatrix} \sin \theta \cos(\omega t + \phi_0) \\ -\sin \theta \sin(\omega t + \phi_0) \\ \cos \theta \end{pmatrix}.$$

θ , ω and ϕ_0 are parameters related to $|\psi\rangle$ and \mathbf{B} as follows:

- θ is the polar angle that the real vector, $\mathbf{s}(t)$, makes with the z -axis in our \mathbb{R}^3 . It relates to $|\psi\rangle$ in that $\theta/2$ is the angle that expresses the magnitudes c and s of $|\psi\rangle$'s Hilbert-space coordinates,

$$\begin{aligned} c &= \cos \frac{\theta}{2} & \text{and} \\ s &= \sin \frac{\theta}{2}. \end{aligned}$$

$\theta/2$ ranges from 0 (when it defines the up- z state, $|+\rangle$) to $\pi/2$ (when it defines the down- z state, $|-\rangle$ state), allowing θ to range from 0 to π in \mathbb{R}^3 .

- ω is the *Larmor frequency*, defined by the magnitude of the \mathbf{B} -field and the constant, γ (the *gyromagnetic ratio*),

$$\omega \equiv \gamma B.$$

- ϕ_0 is the relative phase of the two amplitudes,

$$\phi_0 \equiv \phi_1 - \phi_2.$$

8.7 The End and the Beginning

This completes our tutorial on both time-independent and time-evolving quantum mechanics. If you studied this last lesson as part of the required reading for CS 83B, you are ready to move on to the next phase in that course. If you chose to read it as an optional section in CS 83A, good for you. It's time to start learning about *qubits*.

Chapter 9

The Qubit

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

9.1 Bits and Qubits as Vector Spaces

All the hard work is done. You have mastered the math, spin-physics and computational quantum mechanics needed to begin “doing” quantum computer science.

While useful to our understanding of quantum mechanics, the spin-1/2 physical system, \mathcal{S} , is no longer useful to us as computer scientists. We will work entirely inside its state space \mathcal{H} from this point forward, applying the postulates and traits of quantum mechanics directly to that abstract system, indifferent to the particular \mathcal{S} that the Hilbert space \mathcal{H} is modeling.

Although in practical terms a quantum bit, or *qubit*, is a variable superposition of two basis states in \mathcal{H} of the form shown at the top of this chapter, *formally* qubits – as well as classical bits – are actually *vector spaces*. You read that correctly. A single qubit is not a vector, but an entire vector space. We’ll see how that works shortly.

To get oriented, we’ll do a quick review of classical bits in the new formalism, then we’ll retrace our steps for qubits. Let’s jump in.

9.2 Classical Computation Models – Informal Approach

9.2.1 Informal Definition of Bits and Gates

In order to study the *qubit*, let’s establish a linguistic foundation by defining the more familiar *classical bit*.

Folksy Definition of a Bit

Here's one possible way to define a bit without "going formal."

A Bit (Informal). A "*bit*" is an entity, x , capable of being in one of two states which we label "0" and "1."

The main take-away is that 0 and 1 are not bits. They are the *values* or states that the bit can attain.

We can use the notation

$$x = 0$$

to mean that " x is in the state 0," an *observation* (or possibly a *question*) about the state bit x is in. We also use the same notation to express the *imperative*, "put x into the state 0." The latter is the programmer's *assignment* statement.

Folksy Definition of a Logical Operator a.k.a. *Gate*

What about the logical operators like AND or XOR which transform bits to other bits? We can define those, too, using similarly loose language.

Classical Logical (Boolean) Operator. A *logical operator* is a function that takes one or more bits as input and produces a single bit as output.

Logical operators are also called *logic gates* – or just *gates* – when implemented in circuits diagrams.

Note: We are only considering functions that have a single output bit. If one wanted to build a logic gate with multiple output bits it could be done by combining several single-output logic gates, one for each output bit.

Examples of Logical Operators

The Negation Operator. NOT (symbol \neg) is a logical operator on a single bit defined by the formula

$$\neg x \equiv \begin{cases} 0, & \text{if } x = 1 \text{ and} \\ 1, & \text{otherwise} \end{cases}.$$

A common alternate notation for the NOT operator is the "overline," $\bar{x} = \neg x$.

The Exclusive-Or Operator. XOR (symbol \oplus) is an operator on two bits defined by the formula

$$x \oplus y \equiv \begin{cases} 0, & \text{if } x = y \text{ and} \\ 1, & \text{otherwise} \end{cases}.$$

Mathematically, $x \oplus y$ is called “the mod-2 sum of x and y ,” language that is used throughout classical and quantum logic.

Unary and Binary Operators

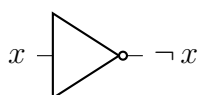
While logic gates can take any number of inputs, those that have one or two inputs are given special names.

*A **unary operator** is a gate that takes single input bit, and a **binary operator** is one that takes two input bits.*

As you can see, NOT is a unary operator while XOR is a binary operator.

Truth Tables

Informally, unary and binary operators are often described using *truth tables*. Here are the traditional gate symbols and truth tables for NOT and XOR.



x	$\neg x$ (or \bar{x})
0	1
1	0



x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

We could go on like this to define more operators and their corresponding logic gates. However, it’s a little disorganized and will not help you jump to a qubit, so let’s try something a little more formal.

Our short formal development of classical logic in this lesson will be restricted to the study of unary operators. We are learning about a single qubit today which is analogous to one classical bit and operators that act on only one classical bit, i.e., *unary operators*. It sounds a little boring, I know, but that’s because in classical logic, unary operators *are* boring (there are only four). But as you are about to see, in the quantum world there are infinitely many different unary operators, all useful.

9.3 Classical Computation Models – Formal Approach

The definition of a quantum bit is necessarily abstract, and if I were to define it at this point, you might not recognize the relationship between it and a classical bit. To be fair to qubits, we'll give a formal definition of a classical bit first, using the same language we will need in the quantum case. This will allow us to establish some vocabulary in the classical context that will be re-usable in the quantum world and give us a reference for comparing the two regimes on a level playing field.

9.3.1 A Miniature Vector Space

You are familiar with \mathbb{R}^2 , the vector space of ordered pairs of real numbers. I'd like to introduce you to an infinitely simpler vector space, $\mathcal{B} = \mathbb{B}^2$.

The Tiny “Field” \mathbb{B}

In place of the *field* \mathbb{R} of real numbers, we define a new field of numbers,

$$\mathbb{B} \equiv \{0, 1\}.$$

That's right, just the set containing two numbers. But we allow the set to have addition, \oplus , and multiplication, \cdot , defined by

\oplus is addition mod-2

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$0 \oplus 0 = 0$$

$$1 \oplus 1 = 0$$

\cdot is ordinary multiplication

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$0 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$

Of course \oplus is nothing other than the familiar **XOR** operation, although in this context we also get negative mod-2 numbers ($-1 = 1$) and subtraction mod-2 ($0 \ominus 1 = 0 \oplus (-1) = 1$), should we need them.

The Vector Space \mathcal{B}

We define $\mathcal{B} = \mathbb{B}^2$ to be the vector space whose scalars come from \mathbb{B} and whose vectors (*objects*) are ordered pairs of numbers from \mathbb{B} . This vector space is so small I can list

its objects on one line,

$$\mathcal{B} = \mathbb{B}^2 \equiv \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}.$$

I'm not going to bother proving that \mathcal{B} obeys all the properties of a vector space, and you don't have to either. But if you are interested, it's a fun ...

[**Exercise.** Show that \mathbb{B} obeys the properties of a field (multiplicative inverses, distributive properties, etc.) and that \mathcal{B} obeys the properties of a vector space.]

The Mod-2 Inner Product

Not only that, but there is an “inner product,”

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \odot \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = x_1 \cdot x_2 \oplus y_1 \cdot y_2.$$

You'll notice something curious about this “inner product” (which is why I put it in quotes and used the operator “ \odot ” rather than “ \cdot ”). The vector $(1, 1)^t$ is a non-zero vector which is *orthogonal to itself*. Don't let this bother you. In computer science, such non-standard inner products exist. (Mathematicians call them “pairings,” since they are not *positive definite*, i.e., it is *not* true that $\mathbf{v} \neq \mathbf{0} \Rightarrow \|\mathbf{v}\| > 0$. We'll just call \odot a *weird inner product*.) This one will be the key to *Simon's algorithm*, presented later in the course. The *inner product* gives rise to a *modulus* – or *length* – for vectors through the same mechanism we used in \mathbb{R}^2 ,

$$\|x\| = |x| \equiv \sqrt{x \odot x},$$

where I have shown two different notations for modulus. With this definition we find

$$\begin{aligned} \left\| \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\| &= \left\| \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\| = 1 \quad \text{and} \\ \left\| \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\| &= \left\| \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\| = 0. \end{aligned}$$

The strange – but *necessary* – equality on the lower right is a consequence of this oddball inner product on $\mathcal{B} = \mathbb{B}^2$.

[**Exercise.** Verify the above moduli.]

[**Notation.** Sometimes I'll use an *ordinary dot* for the inner product, $(x_1, y_1)^t \cdot (x_2, y_2)^t$ instead of the *circle dot*, $(x_1, y_1)^t \odot (x_2, y_2)^t$. When you see a vector on each side of “ \cdot ” you'll know that we really mean the *mod-2 inner product*, not mod-2 multiplication.]

Dimension of \mathcal{B}

If \mathcal{B} is a vector space, what is its dimension, and what is its natural basis? That's not hard to guess. The usual suspects will work. It's a short exercise.

[**Exercise.** Prove that \mathcal{B} is 2-dimensional by showing that

$$\left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\}$$

form an orthonormal basis. **Hint:** There are only four vectors, so express each in this basis. As for linear independence and orthonormality, I leave that to you.]

9.3.2 Formal Definition of a (Classical) Bit

Definition of Bit. A “*bit*” is (any copy of) the entire vector space \mathcal{B} .

Sounds strange, I know, making a bit equal to an entire vector space. We think of a bit as capable of holding a 1 or a 0. This is expressed as follows.

Definition of Bit Value. A “*bit’s value*” or “*state*” is any normalized (i.e., unit) vector in \mathcal{B} .

A bit, itself, is not committed to any particular value until we say which unit-vector in \mathcal{B} we are assigning it. Since there are only two unit vectors in \mathcal{B} , that narrows the field down to one of two values, which I’ll label as follows:

$$\begin{aligned} [0] &\equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix} && \text{and} \\ [1] &\equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \end{aligned}$$

The other two vectors $(0, 0)^t$ and $(1, 1)^t$, have length 0 so cannot be normalized (i.e., we cannot divide them by their length to form a unit vector).

How Programmers Can Think About Formal Bits

If the definition feels too abstract, try this out for size. As a programmer, you’re familiar with the idea of a *variable* (*LVALUE*) which corresponds to a *memory location*. That variable is capable of holding one specific *value* (the *RVALUE*) at any point in time, although there are many possible values we can assign it. The vector space \mathcal{B} is like that memory location: it is capable of holding one of several different values but is not committed to any until we make the assignment. Putting a value into a memory location with an assignment statement like “ $x = 51;$ ” corresponds to choosing one of the unit vectors in \mathcal{B} to be assigned to the bit. So the values allowed to be stored in the formal bit (copy \mathcal{B}) are any of the unit vectors in \mathcal{B} .

Multiple Bits

If we have several bits we have several copies of \mathcal{B} , and we can name each with a variable like \mathbf{x} , \mathbf{y} or \mathbf{z} . We can assign values to these variables with the familiar syntax

$$\begin{aligned}\mathbf{x} &= [1], \\ \mathbf{y} &= [1], \\ \mathbf{z} &= [0],\end{aligned}$$

etc.

A classical bit (uncommitted to a value) can also be viewed as a variable *linear combination* of the two basis vectors in \mathcal{B} .

Alternate Definition of Bit. A “*bit*” is a variable superposition of the two natural basis vectors of \mathcal{B} ,

$$\begin{aligned}\mathbf{x} &= \alpha [0] + \beta [1], \quad \text{where} \\ \alpha^2 \oplus \beta^2 &= 1.\end{aligned}$$

Since α and β are scalars of \mathbb{B} , they can only be 0 and 1, so the normalization condition implies exactly one of them is 1 and the other is 0.

Two questions are undoubtedly irritating you.

1. Is there any benefit, outside of learning quantum computation, of having such an abstract and convoluted definition of a classical bit?
2. How will this definition help us grasp the qubit of quantum computing?

Keep reading.

9.3.3 Formal Definition of a (Classical) Logical Operator

We will only consider *unary* (i.e., *one bit*) operators in this section. *Binary* operators come later.

Unary Operators

Definition of Logical Unary Operator. A “*logical unary operator*” or (“*one bit*” operator) is a linear transformation of \mathcal{B} that maps normalized vectors to other normalized vectors.

This is pretty abstract, but we can see how it works by looking at the only four logical unary operators in sight.

- **The constant-[0] operator** $A(\mathbf{x}) \equiv [0]$. This maps any bit into the 0-bit. (Don't forget, in \mathcal{B} , the 0-bit is not the 0-vector, it is the unit vector $(1, 0)^t$.) Using older, informal truth tables, we would describe this operator as follows:

x	[0]-op
0	0
1	0

In our new formal language, the constant-[0] operator corresponds to the linear transformation whose matrix is

$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix},$$

since, for any unit vector (bit value) $(\alpha, \beta)^t$, we have

$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \oplus \beta \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = [0],$$

the second-from-last equality is due to the fact that, by the normalization requirement on bits, exactly one of α and β must be 1.

- **The constant-[1] operator** $A(\mathbf{x}) \equiv [1]$. This maps any bit into the 1-bit. Informally, it is described by:

x	[1]-op
0	1
1	1

Formally, it corresponds to the linear transformation

$$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix},$$

since, for any unit vector (bit value) $(\alpha, \beta)^t$, we have

$$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ \alpha \oplus \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = [1].$$

- **The negation (or NOT) operator** $A(\mathbf{x}) \equiv \neg \mathbf{x}$. This maps any bit into its logical opposite. It corresponds to

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

since, for any $\mathbf{x} = (\alpha, \beta)^t$, we get

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \neg \mathbf{x}.$$

[**Exercise.** Using the formula, verify that $\neg[0] = [1]$ and $\neg[1] = [0]$.]

- **The *identity* operator** $A(\mathbf{x}) \equiv \mathbf{1}\mathbf{x} = \mathbf{x}$. This maps any bit into itself and corresponds to

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

[**Exercise.** Perform the matrix multiplication to confirm that $\mathbf{1}[0] = [0]$ and $\mathbf{1}[1] = [1]$.]

Linear Transformations that are Not Unary Operators

Apparently any linear transformation on \mathcal{B} other than the four listed above will not correspond to a logical unary operator. For example, the zero-operator

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

isn't listed. This makes sense since it does not map *normalized vectors* to *normalized vectors*. For example,

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} [0] = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \mathbf{0},$$

not a unit vector. Another example is the matrix

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix},$$

since it maps the bit $[1]$ to the non-normalizable $\mathbf{0}$,

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} [1] = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \mathbf{0}.$$

[**Exercise.** Demonstrate that

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

does not represent a logical operator by finding a unit vector that it maps to a non-unit vector, thus violating the definition of a *logical unary operator*.]

Defining logical operators as matrices provides a unified way to represent the seemingly random and unrelated four logical operators that we usually define using isolated truth tables. There's a second advantage to this characterization.

Reversible Logic Gates

A *reversible operator* or *logic gate* is one that can be undone by applying another operator (which might be the same as the original). In other words, its associated matrix has an *inverse*. For example, the constant-[1] operator is not reversible, because it forces its input to the output state [1],

$$\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

and there's no way to reconstruct $(\alpha, \beta)^t$ *reliably* from the constant bit [1]; it has erased information that can never be recovered. Thus, the operator, and its associated logic gate, is called *irreversible*.

Of the four logical operators on one classical bit, only two of them are *reversible*: the identity, $\mathbb{1}$, and negation, \neg . In fact, to reverse them, they can each be reapplied to the output to get the back original input bit value with 100% reliability.

Characterization of Reversible Operators. *An operator is “reversible” \Leftrightarrow its matrix is **unitary**.*

Example. We show that the matrix for \neg is unitary by looking at its columns. The first column is

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

which

- has unit length, since $(0, 1)^t \cdot (0, 1)^t = 1$, and
- is orthogonal to the second column vector, since $(0, 1)^t \cdot (1, 0)^t = 0$.

Reversibility is a very powerful property in quantum computing, so our ability to characterize it with a simple criterion such as *unitarity* is of great value. This is easy to check in the classical case, since there are only four unary operators: two have unitary matrices and two do not.

[**Exercise.** Show that the matrix for $\mathbb{1}$ is unitary and that the two constant operators' matrices are not.]

An Anomaly with Unitary Matrices in \mathcal{B}

We learned that *unitary transformations* have the following equivalent properties.

1. They preserve the lengths of vectors: $\|A\mathbf{v}\| = \|\mathbf{v}\|$,
2. they preserve inner products, $\langle A\mathbf{v} | A\mathbf{w} \rangle = \langle \mathbf{v} | \mathbf{w} \rangle$, and

3. their rows (or columns) are orthonormal.

While this is true of usual (i.e., *positive definite*) inner products, the *pairing* in \mathcal{B} causes these conditions to lose sync. All four possible logical operators on bits do preserve lengths and so meet condition 1; we’ve already shown that they map unit vectors to unit vectors, and we can also check that they map vectors of length zero to other vectors of length zero:

[**Exercise.** Verify that the constant-[1] operator maps the zero-length vectors $(1, 1)^t$ and $(0, 0)^t$ to the zero-length $(0, 0)^t$. Do the same for the constant-[0] operator. Thus both of these non-unitary operators preserve the (oddball) *mod-2 length*.]

However, the constant-[1] and [0] operators *do not preserve inner products*, nor do they have unitary matrices.

[**Exercise.** Find two vectors in \mathcal{B} that have inner product 0, yet whose two images under constant-[1] operator have inner product 1.]

This unusual situation has the consequence that, of the four linear transformations which qualify to be logical operators (preserving lengths) in \mathcal{B} , only two are reversible (have unitary matrices).

*Although the preservation-of-lengths condition (1), is enough to provide \mathcal{B} with four logical operators, that condition is not adequate to assure that they are all **unitary (reversible)**. Only two of the operators satisfy the stronger requirements 2 or 3.*

This distinction between *length-preserving operators* and *unitary operators* is not replicated in qubits, and the consequences will be profound.

9.4 The Qubit

We are finally ready to go quantum. At each turn there will be a classical concept and vocabulary available for comparison because we invested a few minutes studying the formal definitions in that familiar context.

9.4.1 Quantum Bits

Definition of Qubit. A “*quantum bit*,” a.k.a. “*qubit*,” is (any copy of) the entire vector space \mathcal{H} .

Notice that we don’t need to restrict ourselves, yet, to the *projective sphere*. That will come when we describe the allowed values a bit can take.

Comparison with Classical Logic

The *classical bit* was a 2-D vector space, \mathcal{B} , over the *finite* scalar field \mathbb{B} , and now we see that a *quantum bit* is a 2-D vector space, \mathcal{H} , over the *infinite* scalar field \mathbb{C} .

Besides the underlying scalar fields being different – the tiny 2-element \mathbb{B} vs. the infinite and rich \mathbb{C} – the vectors themselves are worlds apart. There are only four vectors in \mathcal{B} , while \mathcal{H} has infinitely many.

However, there is at least one similarity: *they are both two dimensional*. The natural basis for \mathcal{B} is $[0]$ and $[1]$, while the natural basis for \mathcal{H} is $|+\rangle$ and $|-\rangle$.

Quantum Computing Vocabulary

The quantum computing world has adopted different terms for many of the established quantum physics entities. This will be the first of several sections introducing that new vocabulary.

Symbols for Basis Vectors. To reinforce the connection between the qubit and the bit, we abandon the vector symbols $|+\rangle$ and $|-\rangle$ and in their places use $|0\rangle$ and $|1\rangle$ – same vectors, different names. This is true whether we are referring to the preferred z -basis, or any other orthonormal basis.

$$\begin{array}{ll} |+\rangle \longleftrightarrow |0\rangle & |-\rangle \longleftrightarrow |1\rangle \\ |+\rangle_x \longleftrightarrow |0\rangle_x & |-\rangle_x \longleftrightarrow |1\rangle_x \\ |+\rangle_y \longleftrightarrow |0\rangle_y & |-\rangle_y \longleftrightarrow |1\rangle_y \end{array}$$

Alternate x -Basis Notation. Many authors use the shorter notation for the x -basis,

$$\begin{array}{l} |0\rangle_x \longleftrightarrow |+\rangle \\ |1\rangle_x \longleftrightarrow |-\rangle, \end{array}$$

but I will eschew that for the time being; $|+\rangle$ and $|-\rangle$ already have a z -basis meaning in ordinary quantum mechanics, and using them for the x -basis too soon will cause confusion. However, be prepared for me to call $|+\rangle$ and $|-\rangle$ into action as x -basis CBS, particularly when we need the variable x for another purpose.

Computational Basis States. Instead of using the term *eigenbasis*, computer scientists refer to *computational basis states* (or *CBS* when I'm in a hurry). For example, we don't talk about the *eigenbasis of S_z* , $\{|+\rangle, |-\rangle\}$. Rather, we speak of the *preferred computational basis*, $\{|0\rangle, |1\rangle\}$. You are welcome to imagine it as being associated with the observable S_z , but we really don't care what physical observable led to this basis. We only care about the Hilbert space \mathcal{H} , not the physical system, \mathcal{S} , from which it arose. We don't even know what kind of physics will be used to build quantum computers (yet). Whatever physical hardware is used, it will give us the 2-D Hilbert space \mathcal{H} .

Alternate bases like $\{|0\rangle_x, |1\rangle_x\}$, $\{|0\rangle_y, |1\rangle_y\}$ or even $\{|0\rangle_{\hat{n}}, |1\rangle_{\hat{n}}\}$ for some direction \hat{n} , when needed, are also called *computational bases*, but we usually qualify them using the term *alternate computational basis*. We still have the short-hand terms *z-basis*, *x-basis*, etc., which avoid the naming conflict, altogether. These *alternate computational bases* are still defined by their expansions in the preferred, *z-basis* ($|1\rangle_x = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$, e.g.), and all the old relationships remain.

9.4.2 Quantum Bit Values

Definition of Qubit Value. The “*value*” or “*state*” of a qubit is any unit (or normalized) vector in \mathcal{H} .

In other words, a qubit is an entity – the Hilbert space \mathcal{H} – whose *value* can be any vector on the *projective sphere* of that space.

A qubit, itself, is not committed to any particular value until we say which specific unit-vector in \mathcal{H} we are assigning to it.

Normalization

Why do we restrict qubit values to the projective sphere? This is a quantum system, so states are always normalized vectors. That’s how we defined the state space in our quantum mechanics lesson. A qubit’s state (or the state representing any quantum system) has to reflect probabilities which sum to 1. That means the magnitude-squared of all the amplitudes must sum to 1. Well, that’s the *projective sphere*. Any vectors off that sphere cannot claim to reflect reality.

Certainly there will be times when we choose to work with un-normalized vectors, especially in CS 83B, but that will be a computational convenience that must eventually be corrected by normalizing the answers.

Comparison with Classical Logic

Just as a classical bit was capable of storing a state $[0]$ or $[1]$ (the two unit vectors in \mathcal{B}), qubits can be placed in specific *states* which are normalized vectors in \mathcal{H} . This time, however, there are infinitely many unit vectors in \mathcal{H} : we have the entire *projective sphere* from which to choose.

9.4.3 Usual Definition of Qubit and its Value

A more realistic working definition of qubit parallels that of the alternative formal definition of bit.

Alternative Definition of Qubit. A “*qubit*” is a variable superposition of the two natural basis vectors of \mathcal{H} ,

$$\begin{aligned} |\psi\rangle &= \alpha |0\rangle + \beta |1\rangle, & \text{where the complex scalars satisfy} \\ |\alpha|^2 + |\beta|^2 &= 1. \end{aligned}$$

I used the word “variable” to call attention to the fact that the qubit *stores* a value, but is not the value, *itself*.

Global Phase Factors

We never forget that even if we have a normalized state,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle,$$

it is not a unique representation; any length-1 scalar multiple

$$e^{i\theta} |\psi\rangle = e^{i\theta} \alpha |0\rangle + e^{i\theta} \beta |1\rangle, \quad \text{real } \theta,$$

still resides on the projective sphere, and since it is a scalar multiple of $|\psi\rangle$, it is a valid representative of same state or qubit value. We would say “ $|\psi\rangle$ and $e^{i\theta} |\psi\rangle$ differ by an overall, or global, phase factor θ ,” a condition that does not change anything, but can be used to put $|\psi\rangle$ into a more workable form.

Comparison with Classical Logic

The alternate/working expressions for bits in the two regimes look similar.

$$\text{Classical: } \mathbf{x} = \alpha [0] + \beta [1], \quad \alpha^2 \oplus \beta^2 = 1$$

$$\text{Quantum: } |\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1$$

In the classical case, α and β could only be 0 or 1, and they could not be the same (normalizability), leading to only two possible states for \mathbf{x} , namely, $[0]$ or $[1]$. In the quantum case, α and β can be any of an infinite combination of complex scalars, leading to an infinite number of distinct values for the state $|\psi\rangle$.

9.4.4 Key Difference Between Bits and Qubits

What’s the big idea behind qubits? They are considerably more difficult to define and study than classical bits, so we deserve to know what we are getting for our money.

Parallel Processing

The main motivation comes from our **Trait #6**, the *fourth postulate of quantum mechanics*. It tells us that until we measure the state $|\psi\rangle$, it has a probability of landing in either state, $|0\rangle$ or $|1\rangle$, the exact details given by the magnitudes of the complex amplitudes α and β .

As long as we don't measure $|\psi\rangle$, it is like the Centaur of Greek mythology: part $|0\rangle$ and part $|1\rangle$; when we process this beast with quantum logic gates, we will be sending *both* alternative binary values through the hardware in a single pass. That will change it to another normalized state vector (say $|\phi\rangle$), which has *different* amplitudes, and that can be sent through further logic gates, again retaining the potential to be part $|0\rangle$ and part $|1\rangle$, but with different probabilities.

Classical Computing as a Subset

Classical logic can be emulated using quantum gates.

This is slightly less obvious than it looks, and we'll have to study how one constructs even a simple AND gate using quantum logic. Nevertheless, we can correctly forecast that the computational basis states, $|0\rangle$ and $|1\rangle$, will correspond to the classical bit values [0] and [1]. These two lonely souls, however, are now swimming in a continuum of non-classical states.

Measurement Turns Qubits into Bits

Well, this is not such a great selling point. If qubits are so much better than bits, why not leave them alone? Worse still, our **Trait #7**, the *fifth postulate of quantum mechanics*, means that even if we *don't want* to collapse the qubit into a computational basis state, once we measure it, we will have done just that. We will lose the exquisite subtleties of α and β , turning the entire state into a $|0\rangle$ or $|1\rangle$.

Once you go down this line of thought, you begin to question the entire enterprise. We can't get any answers if we don't test the output states, and if they always collapse, what good did the amplitudes do? This skepticism is reasonable. For now, I can only give you some ideas, and ask you to wait to see the examples.

1. We can do a lot of processing “below the surface of the quantum ocean,” manipulating the quantum states without attempting an information-destroying measurement that “brings them up for air” until the time is right.
2. We can use **Trait #6**, the *fourth postulate*, in reverse: Rather than looking at amplitudes as a prediction of experimental outcomes, we can view the relative distribution of several measurement outcomes to guess at the amplitudes of the output state.
3. By preparing our states and quantum logic carefully, we can “load the dice” so

that the likelihood of getting an information-rich collapsed result will be greatly enhanced.

9.5 Quantum Operators (Unary)

9.5.1 Definition and Notation

Definition of a Unary Quantum Logical Operator. A “*unary quantum logical operator*” or “*unary quantum gate*” is a linear transformation of \mathcal{H} that maps normalized (unit) vectors to other normalized vectors.

The first important observation is that since \mathcal{H} is 2-dimensional, a unary quantum operator can be represented by a 2×2 matrix

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}.$$

Notation. I am now numbering starting from 0 rather than 1. This will continue for the remainder of the course.

The big difference between these matrices and those of classical unary operators is that here the coefficients come from the infinite pool of complex numbers. Of course, not every 2×2 matrix qualifies as a quantum gate, a fact we’ll get to shortly.

9.5.2 Case Study – Our First Quantum Gate, QNOT

We’ll examine every angle of this first example. It is precisely because of its simplicity that we can easily see the important differences between classical and quantum computational logic.

The Quantum NOT (or QNOT) Operator, X

The QNOT operator swaps the amplitudes of any state vector. It corresponds to

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

the same matrix that represents the NOT operator, \neg , of classical computing. The difference here is not in the operator but in the vast quantity of qubits to which we can apply it. Using $|\psi\rangle = (\alpha, \beta)^t$, as we will for this entire lecture, we find

$$X|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}.$$

In the special case of a CBS ket, we find that this does indeed change the state from $|0\rangle$ to $|1\rangle$ and vice versa.

[**Exercise.** Using the formula, verify that $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$.]

Notation and Vocabulary

The X operator is sometimes called the *bit flip* operator, because it “flips” the CBS coefficients, $\alpha \leftrightarrow \beta$. In the special case of a pure CBS input, like $|0\rangle$, it “flips” it to the *other* CBS, $|1\rangle$.

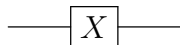
The reason QNOT is usually labeled using the letter X is that, other than the factor of $\frac{\hbar}{2}$, the matrix is the same as the spin-1/2 observable S_x . In fact, you’ll recall from the quantum mechanics lesson that QNOT is precisely the *Pauli spin matrix* in the x -direction,

$$X = \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

It’s best not to read anything too deep into this. The matrix that models an observable is used differently than one that performs a reversible operation on a qubit. Here, we are swapping amplitudes and therefore negating computational basis states. That’s the important take-away.

Gate Symbol and Circuits

In a circuit diagram, we would use



when we want to express an X gate or, less frequently, if we want to be overly explicit,



We might show the effect of the gate right on the circuit diagram,

$$\alpha |0\rangle + \beta |1\rangle \quad \text{---} \boxed{X} \text{---} \quad \beta |0\rangle + \alpha |1\rangle.$$

The *input* state is placed on the *left* of the gate symbol and the *output* state on the *right*.

Computational Basis State Emphasis

Because any linear operator is completely determined by its action on a basis, you will often see an operator like X defined *only* on the CBS states, and you are expected to know that this should be extended to the entire \mathcal{H} using linearity. In this case, the letters x and y are usually used to label a CBS, so

$$|x\rangle = \begin{cases} |0\rangle \\ \text{or} \\ |1\rangle \end{cases}$$

to be distinguished from $|\psi\rangle$, which can take on infinitely many *superpositions* of these two basis kets. With this convention, any quantum logic gate can be defined by its action on the $|x\rangle$ (and sometimes $|y\rangle$, $|z\rangle$ or $|w\rangle$, if we need more input kets). For the X gate, it might look like this

$$|x\rangle \text{ --- } \boxed{X} \text{ --- } |\neg x\rangle .$$

This expresses the two possible input states and says that $X|0\rangle = |\neg 0\rangle = |1\rangle$, while, $X|1\rangle = |\neg 1\rangle = |0\rangle$. Using alternative notation,

$$|x\rangle \text{ --- } \boxed{X} \text{ --- } |\bar{x}\rangle .$$

In fact, you'll often see the mod-2 operator for some CBS logic. If we used that to define X , it would look like this:

$$|x\rangle \text{ --- } \boxed{X} \text{ --- } |1 \oplus x\rangle .$$

[**Exercise.** Why does the last expression result in a logical negation of the CBS?]

You Must Remember This. The operators (\oplus , \neg , etc.) used inside the kets on the variables x and/or y apply *only* to the binary values 0 or 1 that label the basis states. *They make no sense for general states.* We must extend *linearly* to the rest of our Hilbert space.

Sample Problem

Given the definition of the bit flip operator, X , in terms of the CBS $|x\rangle$,

$$|x\rangle \text{ --- } \boxed{X} \text{ --- } |\bar{x}\rangle ,$$

what is the action of X on an arbitrary state $|\psi\rangle$, and what is the matrix for X ?

Expand $|\psi\rangle$ along the computational basis and apply X :

$$\begin{aligned} X|\psi\rangle &= X(\alpha|0\rangle + \beta|1\rangle) \\ &= \alpha X|0\rangle + \beta X|1\rangle \\ &= \alpha|\bar{0}\rangle + \beta|\bar{1}\rangle \\ &= \alpha|1\rangle + \beta|0\rangle \\ &= \beta|0\rangle + \alpha|1\rangle , \end{aligned}$$

in agreement with our original definition when viewed in coordinate form.

The matrix for X would be constructed, as with any linear transformation, by applying X to each basis ket and setting the columns of the matrix to those results.

$$M_X = \begin{pmatrix} X|0\rangle & X|1\rangle \end{pmatrix} = \begin{pmatrix} |1\rangle & |0\rangle \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} .$$

While the problem didn't ask for it, let's round out the study by viewing X in terms of a ket's CBS coordinates, $(\alpha, \beta)^t$. For that, we can read it directly off the final derivation of $X|\psi\rangle$, above, or apply the matrix, which I will do now.

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}.$$

One Last Time: In the literature, most logic gates are defined in terms of $|x\rangle$, $|y\rangle$, etc. This is only the action on the CBS, and it is up to us to fill in the blanks, get its action on the general $|\psi\rangle$ and produce the matrix for the gate.

Comparison with Classical Logic

We've seen that there is at least one classical operator, NOT, that has an analog, QNOT, in the quantum world. Their matrices look the same and they affect the classical bits, $[0] / [1]$, and corresponding CBS counterparts, $|0\rangle / |1\rangle$, identically, but that's where the parallels end.

What about the other three classical unary operators? You can probably guess that the quantum identity,

$$\mathbb{1} \equiv \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

exhibits the same similarities and differences as the QNOT did with the NOT. The matrices are identical and have the same effect on classical/CBS kets, but beyond that, the operators work in different worlds.

[**Exercise.** Express the gate (i.e., circuit) definition of $\mathbb{1}$ in terms of a CBS $|x\rangle$ and give its action on a general $|\psi\rangle$. Discuss other differences between the classical and quantum identity.]

That leaves the two constant operators, the $[0]$ -op and the $[1]$ -op. The simple answer is there are no quantum counterparts for these. The reason gets to the heart of quantum computation.

Unitarity is Not Optional

The thing that distinguished \neg and $\mathbb{1}$ from $[0]$ -op and the $[1]$ -op in the classical case was unitarity. The first two were unitary and the last two were not. How did those last two even sneak into the classical operator club? They had the property that they preserved the lengths of vectors. That's all an operator requires. But these constant ops were not reversible which implied that their matrices were not unitary. The quirkiness of certain operators being length-preserving yet non-unitary was a fluke of nature caused by the strange mod-2 inner product on \mathcal{B} . It allowed a distinction between length-preservation and unitarity.

In quantum computing, no such distinction exists. The self-same requirement that operators map unit vectors to other unit vectors in \mathcal{H} forces operators to be

unitary. This is because \mathcal{H} has a well-behaved (positive definite) inner product. We saw that with any positive definite inner product, unitarity and length-preservation are equivalent.

But if you don't want to be that abstract, you need only "try on" either constant op for size and see if it fits. Let's apply an attempted analog of the [0]-op to the unit vector $|0\rangle_x$. For fun, we'll do it twice. First try out the (non-unitary) matrix for the classical [0]-op in the quantum regime. We find

$$\begin{aligned} \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) &= \frac{1}{\sqrt{2}} \left(\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right) \\ &= \frac{1}{\sqrt{2}} \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} \sqrt{2} \\ 0 \end{pmatrix}, \end{aligned}$$

not a unit vector. To see a different approach we show it using a putative operator, A , defined by its CBS action that ignores the CBS input, always answering with a $|0\rangle$,

$$|x\rangle \text{ --- } \boxed{A} \text{ --- } |0\rangle .$$

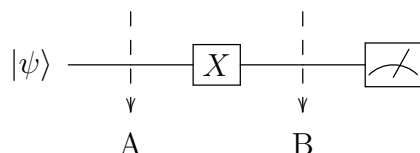
Apply A to the same unit vector $|0\rangle_x$.

$$\begin{aligned} A \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) &= \left(\frac{A|0\rangle + A|1\rangle}{\sqrt{2}} \right) = \left(\frac{|0\rangle + |0\rangle}{\sqrt{2}} \right) \\ &= \sqrt{2} |0\rangle, \end{aligned}$$

again, not normalized.

Measurement

Finally, we ask what impact QNOT has on the measurement probabilities of a qubit. Here is a picture of the circuit with two potential measurement "access points," A (*before* the gate) and B (*after*):



Of course, we know from *Trait #7 (fifth postulate of QM)* that once we measure the state it collapses into a CBS, so we cannot measure both points on the same "sample" of our system. If we measure at A, the system will collapse into either $|0\rangle$ or $|1\rangle$ and we will no longer have $|\psi\rangle$ going into X . So the way to interpret this diagram is to

visualize many different copies of the system in the same state. We measure some at access point A and others at access point B. The math tells us that

$$|\psi\rangle = \begin{cases} \alpha|0\rangle + \beta|1\rangle & \text{at Point A} \\ \beta|0\rangle + \alpha|1\rangle & \text{at Point B} \end{cases}$$

So, if we measure 1000 identical states at point A,

$$|\psi\rangle \longrightarrow \boxed{\text{meter}},$$

we will get

$$\begin{aligned} \# \text{ of "0"-measurements} &\approx 1000 \times |\alpha|^2 \quad \text{and} \\ \# \text{ of "1"-measurements} &\approx 1000 \times |\beta|^2, \end{aligned}$$

by **Trait #6**, the fourth QM postulate. However, if we do *not* measure those states, but instead send them through X and *only then* measure them (at B),

$$|\psi\rangle \longrightarrow \boxed{X} \longrightarrow \boxed{\text{meter}},$$

the same trait applied to the output expression tells us that the probabilities will be swapped,

$$\begin{aligned} \# \text{ of "0"-measurements} &\approx 1000 \times |\beta|^2 \quad \text{and} \\ \# \text{ of "1"-measurements} &\approx 1000 \times |\alpha|^2, \end{aligned}$$

Composition of Gates

Recall from the lesson on *linear transformation* that any *unitary* matrix, U , satisfies

$$U^\dagger U = U U^\dagger = \mathbb{1},$$

where U^\dagger is the *conjugate transpose* a.k.a, *adjoint*, of U . In other words its *adjoint* is also its *inverse*.

Because unitarity is required of *all* quantum gates – in particular QNOT – we know that

$$X^\dagger X = X X^\dagger = \mathbb{1},$$

But the matrix for X tells us that it is *self-adjoint*:

$$X^\dagger = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^\dagger = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = X.$$

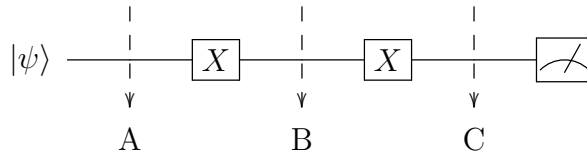
Therefore, in this case, we get the even stronger identity

$$X^2 = \mathbb{1}.$$

X is its own inverse. If we apply X (or any *self-adjoint* operator) consecutively without an intervening measurement, we should get our original state back. For QNOT, this means

$$|\psi\rangle \text{ --- } \boxed{X} \text{ --- } \boxed{X} \text{ --- } |\psi\rangle ,$$

which can be verified either algebraically, by multiplying the vectors and matrices, or experimentally, by taking lots of sample measurements on identically prepared states. Algebraically, for example, the state of the qubit at points A, B and C,



will be

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \longrightarrow \begin{pmatrix} \beta \\ \alpha \end{pmatrix} \longrightarrow \begin{pmatrix} \alpha \\ \beta \end{pmatrix} .$$

This leads to the expectation that for all quantum gates – *as long as we don't measure anything* – we can keep sending the output of one into the input of another, and while they will be transformed, the exquisite detail of the qubits' amplitudes remain intact. They may be hidden in algebraic changes caused by the quantum gates, but they will be retrievable due to the unitarity of our gates. However, make a measurement, and we will have destroyed that information; *measurements are not unitary operations*.

We've covered the only two classical gates that have quantum alter egos. Let's go on to meet some one-bit quantum gates that have no classical counterpart.

9.5.3 The Phase Flip, Z

If the *bit flip*, X , is the operator that coincides with the physical observable S_x , then the *phase flip*, Z , turns out to be the Pauli matrix σ_z associated with the S_z observable.

The Z operator, whose matrix is defined to be

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} ,$$

negates the second amplitude of a state vector, leaving the first unchanged,

$$Z|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix} .$$

Notation and Vocabulary

Z is called a *phase flip*, because it changes (maximally) the *relative* phase of the two amplitudes of $|\psi\rangle$. Why is multiplication by -1 a *maximal* phase change? Because

$$-1 = e^{i\pi},$$

so multiplying β by this scalar is a $\pi = 180^\circ$ rotation of β in the complex plane. You can't get more "maximal" than rotating something in the complex plane by 180° .

[**Exercise.** Graph β and $e^{i\pi}\beta$. If you are stuck, express β in polar form and use the Euler formula.]

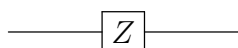
Furthermore, if we modify *only one* of the two amplitudes in a state's CBS expansion, that's a *relative* phase change, something we know *matters* (unlike *absolute* phase changes, which are inconsequential multiples of the entire ket by a complex unit).

[**Exercise.** Write out a summary of why a relative phase change matters. **Hint:** Review the sections about *time evolution of a quantum state* from the third (optional) time-dependent quantum mechanics lesson, particularly **Traits #13** and **#14**.]

The use of the symbol, Z , to represent this operator is, as already observed, due to its matrix being identical to σ_z of the S_z observable.

Gate Symbol and Circuits

In a circuit diagram we use



to express a Z gate. Its full effect on a general state in diagram form is

$$\alpha|0\rangle + \beta|1\rangle \longrightarrow \boxed{Z} \longrightarrow \alpha|0\rangle - \beta|1\rangle.$$

Computational Basis State Emphasis

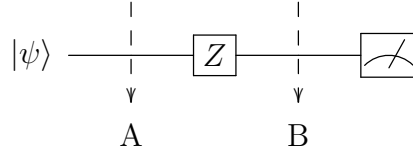
It is uncommon to try to express Z in compact computational basis form; it leaves $|0\rangle$ unchanged, and negates $|1\rangle$, an operation that has no classical counterpart (in \mathcal{B} , $-1 = 1$, so $-[1] = [1]$, and the operation would look like an *identity*). However, if pressed to do so, we could use

$$|x\rangle \longrightarrow \boxed{Z} \longrightarrow (-1)^x |x\rangle.$$

[**Exercise.** Show that this agrees with Z 's action on a general $|\psi\rangle$ by expanding $|\psi\rangle$ along the computational basis and using this formula while applying linearity.]

Measurement

It is interesting to note that Z has no measurement consequences on a single qubit since the *magnitude* of both amplitudes remain unchanged by the gate. The circuit



produces the A-to-B transition

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \longrightarrow \begin{pmatrix} \alpha \\ -\beta \end{pmatrix},$$

yielding the same probabilities, $|\alpha|^2$ and $|\beta|^2$, at both access points. Therefore, both $|\psi\rangle$ and $Z|\psi\rangle$ will have *identical measurement likelihoods*; if we have 1000 electrons or photons in spin state $|\psi\rangle$ and 1000 in $Z|\psi\rangle$, a measurement of all of them will throw about $|\alpha|^2 \times 1000$ into state $|0\rangle$ and $|\beta|^2 \times 1000$ into state $|1\rangle$.

However, two states that have the same measurement probabilities are not necessarily the same state.

The relative phase difference between $|\psi\rangle$ and $Z|\psi\rangle$ can be “felt” the moment we try to combine (incorporate into a larger expression) either state using superposition. Mathematically, we can see this by noticing that

$$\frac{|\psi\rangle + |\psi\rangle}{2} = |\psi\rangle,$$

i.e., we cannot create a *new* state from a single state, which is inherently *linearly dependent* with itself, while a *distinct* normalized state *can* be formed by

$$\begin{aligned} \frac{|\psi\rangle + Z|\psi\rangle}{2\alpha} &= \frac{(\alpha + \alpha)|0\rangle + (\beta - \beta)|1\rangle}{2\alpha} \\ &= \frac{2\alpha|0\rangle}{2\alpha} = |0\rangle. \end{aligned}$$

Unless it so happens that $\alpha = 1$ and $\beta = 0$, we get different results when using $|\psi\rangle$ and $Z|\psi\rangle$ in the second position of these two legal superpositions, demonstrating that they do, indeed, have different physical consequences.

9.5.4 The Bit-and-Phase Flip Operator, Y

After seeing the X and Z operators, we are compelled to wonder about the operator Y that corresponds to the Pauli matrix σ_y associated with the S_y observable. It is just as important as those, but has no formal name (we’ll give it one in a moment).

The Y operator is defined by the matrix

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},$$

which has the following effect on a general ket:

$$Y|\psi\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = -i \begin{pmatrix} \beta \\ -\alpha \end{pmatrix} \cong \begin{pmatrix} \beta \\ -\alpha \end{pmatrix},$$

that last equality accomplished by multiplying the result by the innocuous unit scalar, i .

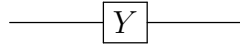
Notation and Vocabulary

Although Y has no official name, we will call it the *bit-and-phase flip*, because it flips both the bits and the relative phase, simultaneously.

The symbol Y is used to represent this operator because it is identical to the Pauli matrix σ_y .

Gate Symbol and Circuits

In a circuit diagram, we use



to express a Y gate. Its full effect on a general state in diagram form is

$$\alpha|0\rangle + \beta|1\rangle \text{ --- } \boxed{Y} \text{ --- } -i\beta|0\rangle + i\alpha|1\rangle.$$

Computational Basis State Emphasis

No one expresses Y in computational basis form, but it can be done.

[**Exercise.** Find a formula that expresses Y 's action on a CBS.]

What's with the i ? [**Optional Reading**]. Clearly, we can pull the i out of the matrix (or the output ket) with impunity since we can always multiply any normalized state by a unit scalar without changing the state. That makes us wonder what it's doing there in the first place. Superficially, the i makes $Y = \sigma_y$, a desirable completion to the pattern started with $X = \sigma_x$ and $Z = \sigma_z$. But more deeply, by attaching a factor of i to this operator, we can, with notational ease, produce a linear combination of the three matrices using *real* coefficients, n_x , n_y and n_z ,

$$n_x \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + n_y \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} + n_z \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} n_z & n_x - n_y i \\ n_x + n_y i & -n_z \end{pmatrix}.$$

This may seem a less than compelling justification, but the expression is of the syntactic form

$$\vec{\sigma} \cdot \hat{\mathbf{n}},$$

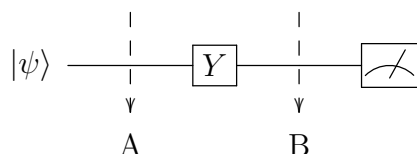
where, $\hat{\mathbf{n}}$ is a *real* 3-D unit vector $(n_x, n_y, n_z)^t$, $\vec{\sigma}$ is a “vector” of operators

$$\vec{\sigma} \equiv \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{pmatrix},$$

and their formal “dot” product, $\vec{\sigma} \cdot \hat{\mathbf{n}}$, represents the matrix for the observable $S_{\hat{\mathbf{n}}}$ (the measurement of spin in the most general direction defined by a unit vector $\hat{\mathbf{n}}$). This will be developed and used in the next course CS 83B.

Measurement

As a combination of both *bit flip* (which swaps probabilities) and *phase flip* (which does not change probabilities), Y has the same measurement consequence as a simple QNOT:



This gives an A-to-B transition

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \rightarrow \begin{pmatrix} -i\beta \\ i\alpha \end{pmatrix},$$

which causes the probabilities, $|\alpha|^2$ and $|\beta|^2$ to get swapped at access point B.

9.5.5 The Hadamard Gate, H

While all these quantum gates are essential, the one having the most far-reaching consequences and personifying the essence of quantum logic is the *Hadamard gate*.

The *Hadamard operator*, H , is defined by the matrix

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Traditionally, we first address its effect on the CBS states,

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{and} \\ H|1\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \end{aligned}$$

We immediately recognize that this “rotates” the z -basis kets onto the x -basis kets,

$$\begin{aligned} H|0\rangle &= |0\rangle_x \quad \text{and} \\ H|1\rangle &= |1\rangle_x . \end{aligned}$$

[**Exercise.** Prove that H is unitary.]

H of a General State

This is to be extended to an arbitrary state, $|\psi\rangle$, using the obvious rules. Rather than approach it by expanding $|\psi\rangle$ along the z -basis then extending linearly, it’s perhaps faster to view everything in terms of matrices and column vectors,

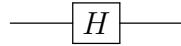
$$H|\psi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix} ,$$

which can be grouped

$$H|\psi\rangle = \left(\frac{\alpha + \beta}{\sqrt{2}} \right) |0\rangle + \left(\frac{\alpha - \beta}{\sqrt{2}} \right) |1\rangle$$

Gate Symbol and Circuits

In a circuit diagram, we use



to express an H gate. Its full effect on a general state in diagram form is

$$\alpha|0\rangle + \beta|1\rangle \quad \text{---} \boxed{H} \text{---} \quad \left(\frac{\alpha + \beta}{\sqrt{2}} \right) |0\rangle + \left(\frac{\alpha - \beta}{\sqrt{2}} \right) |1\rangle .$$

Computational Basis State Emphasis

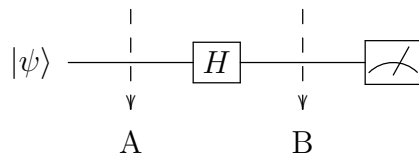
It turns out to be very useful to express H in compact computational basis form. I’ll give you the answer, and let you prove it for for yourself.

$$|x\rangle \quad \text{---} \boxed{H} \text{---} \quad \frac{|0\rangle + (-1)^x |1\rangle}{\sqrt{2}} .$$

[**Exercise.** Show that this formula gives the right result on each of the two CBS kets.]

Measurement

Compared to the previous gates, the Hadamard gate has a more complex and subtle effect on measurement probabilities. The circuit



gives an A-to-B transition

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \rightarrow \begin{pmatrix} \frac{\alpha+\beta}{\sqrt{2}} \\ \frac{\alpha-\beta}{\sqrt{2}} \end{pmatrix},$$

causing the output probabilities to go from $|\alpha|^2$ and $|\beta|^2$ at point A to

$$\begin{aligned} P(H|\psi\rangle \searrow |0\rangle) &= \frac{|\alpha + \beta|^2}{2} \\ P(H|\psi\rangle \searrow |1\rangle) &= \frac{|\alpha - \beta|^2}{2}, \end{aligned}$$

at point B.

Notation. The diagonal arrow, \searrow , is to be read “when measured, collapses to.”

To make this concrete, here are the transition probabilities of a few input states (details left as an exercise).

$ \psi\rangle$	Probabilities Before	Probabilities After
$ 0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$	$P(\searrow 0\rangle) = 1$ $P(\searrow 1\rangle) = 0$	$P(\searrow 0\rangle) = 1/2$ $P(\searrow 1\rangle) = 1/2$
$ 1\rangle_x = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$	$P(\searrow 0\rangle) = 1/2$ $P(\searrow 1\rangle) = 1/2$	$P(\searrow 0\rangle) = 0$ $P(\searrow 1\rangle) = 1$
$ \psi_0\rangle = \begin{pmatrix} i\sqrt{.3} \\ -\sqrt{.7} \end{pmatrix}$	$P(\searrow 0\rangle) = .3$ $P(\searrow 1\rangle) = .7$	$P(\searrow 0\rangle) = 1/2$ $P(\searrow 1\rangle) = 1/2$
$ \psi_1\rangle = \begin{pmatrix} 1/2 \\ \sqrt{3}/2 \end{pmatrix}$	$P(\searrow 0\rangle) = 1/4$ $P(\searrow 1\rangle) = 3/4$	$P(\searrow 0\rangle) = \frac{2+\sqrt{3}}{4}$ $P(\searrow 1\rangle) = \frac{2-\sqrt{3}}{4}$

[**Exercise.** Verify these probabilities.]

9.5.6 Phase-Shift Gates, S , T and R_θ

The *phase-flip* gate, Z , is a special case of a more general (relative) phase shift operation in which the coefficient of $|1\rangle$ is “shifted” by $\theta = \pi$ radians. There are two other common shift amounts, $\pi/2$ (the S operator) and $\pi/4$ (the T operator). Beyond that we use the most general amount, any θ (the R_θ operator). Of course, they can all be defined in terms of R_θ , so we’ll define that one first.

The *phase shift* operator, R_θ , is defined by the matrix

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix},$$

where θ is any real number. It leaves the coefficient of $|0\rangle$ unchanged and “shifts” (or “rotates”) $|1\rangle$ ’s coefficient by a relative angle θ , whose meaning we will discuss in a moment. Here’s the effect on a general state:

$$R_\theta |\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ e^{i\theta} \beta \end{pmatrix}.$$

The operators S and T are defined in terms of R_θ ,

$$S \equiv R_{\pi/2} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix},$$

and

$$T \equiv R_{\pi/4} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

Vocabulary

S is called the “phase gate,” and in an apparent naming error T is referred to as the “ $\pi/8$ gate,” but this is not actually an error as much as it is a change of notation. Like a state vector, any unitary operator on state space can be multiplied by a unit scalar with impunity. We’ve seen the reason, but to remind you, state vectors are *rays* in state space, all vectors on the ray considered to be the same state. Since we are also working on the projective sphere, any unit scalar not only represents the same state, but keeps the vector on the projective sphere.

With that in mind, if we want to see a more balanced version of T , we multiply it by $e^{-i\pi/8}$ and get the equivalent operator,

$$T \cong \begin{pmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{pmatrix}.$$

Measurement

These phase shift gates leave the probabilities alone for single qubit systems, just as the *phase flip* gate, Z , did. You can parallel the exposition presented for Z in the current situation to verify this.

9.6 Putting Unary Gates to Use

9.6.1 Basis Conversion

Every quantum gate is unitary, a fact that has two important consequences, the first of which we have already noted.

1. Quantum gates are reversible; their *adjoints* can be used to undo their action.
2. Quantum gates map any orthonormal CBS to another orthonormal CBS.

The second item is true using logic from our *linear algebra* lesson as follows. Unitary operators preserve inner products. For example, since the natural CBS $\{|0\rangle, |1\rangle\}$ – which we can also express using the more general notation $\{|x\rangle\}_{x=0}^1$ – satisfies the orthonormality relation

$$\langle x | y \rangle = \delta_{xy} \quad (\text{kronecker delta}),$$

then the unitarity of U means we also have

$$\langle x | U^\dagger U | y \rangle = \delta_{xy}.$$

(I'll remind you that the LHS of last equation is nothing but the inner product of $U|y\rangle$ with $U|x\rangle$ expressed in terms of the *adjoint conversion rules* introduced in our lesson on quantum mechanics).

That tells us that $\{U|0\rangle, U|1\rangle\}$ are orthonormal, and since the dimension of $\mathcal{H} = 2$ they also span the space. In other words, they form an orthonormal basis, as claimed.

Let's call this the *basis conversion property of unitary transformations* and make it a theorem.

Theorem (Basis Conversion Property). *If U is a unitary operator and \mathcal{A} is an orthonormal basis, then $U(\mathcal{A})$, i.e., the image of vectors \mathcal{A} under U , is another orthonormal basis, \mathcal{B} .*

[**Exercise.** Prove the theorem for any dimension, N . **Hint:** Let $\mathbf{b}_k = U(\mathbf{a}_k)$ be the k th vector produced by subjecting $\mathbf{a}_k \in \mathcal{A}$ to U . Review the *inner product-preserving* property of a unitary operator and apply that to any two vectors \mathbf{b}_k and \mathbf{b}_j in the image of \mathcal{A} . What does that say about the full set of vectors $\mathcal{B} = U(\mathcal{A})$? Finally, what do you know about the number of vectors in the basis for an N -dimensional vector space?]

QNOT and Bases

When applied to some gates, like QNOT, this is a somewhat trivial observation since QNOT maps the z -basis to itself:

$$\begin{aligned} X : |0\rangle &\longmapsto |1\rangle \\ X : |1\rangle &\longmapsto |0\rangle . \end{aligned}$$

Hadamard and Bases

In other situations, this is a very useful and interesting conversion. For example, if you look back at its effect on the CBS, the Hadamard gate takes the z -basis to the x -basis:

$$\begin{aligned} H : |0\rangle &\longmapsto |0\rangle_x \\ H : |1\rangle &\longmapsto |1\rangle_x , \end{aligned}$$

and, since every quantum gate's adjoint is its inverse, and H is self-adjoint (easy [Exercise]), it works in the reverse direction as well,

$$\begin{aligned} H : |0\rangle_x &\longmapsto |0\rangle \\ H : |1\rangle_x &\longmapsto |1\rangle . \end{aligned}$$

[Exercise. Identify the unitary operator that has the effect of converting between the z -basis and the y -basis.]

9.6.2 Combining Gates

We can place two or more one-qubit gates in series to create desired results as the next two sections will demonstrate.

An x -Basis QNOT

The QNOT gate (a.k.a. X), swaps the two CBS states, *but only relative to the z -basis*, because that's how we defined it. An easy experiment shows that this is not true of another basis, such as the x -basis:

$$\begin{aligned} X |1\rangle_x &= X \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \frac{X |0\rangle - X |1\rangle}{\sqrt{2}} \\ &= \frac{|1\rangle - |0\rangle}{\sqrt{2}} = -|1\rangle_x \cong |1\rangle_x . \end{aligned}$$

[Exercise. To what is the final equality due? What is $X |0\rangle_x$?]

[**Exercise.** Why is this not a surprise? **Hint:** Revive your quantum mechanics knowledge. X is proportional to the matrix for the observable, S_x , whose *eigenvectors* are $|0\rangle_x$ and $|1\rangle_x$, by definition. What *is* an eigenvector?]

If we wanted to construct a gate, $QNOT_x$, that *does* have the desired swapping effect on the x -basis we could approach it in a number of ways, two of which are

1. *brute force* using linear algebra, and
2. *a gate-combination* using the basis-transforming power of the H -gate.

I'll outline the first approach, leaving the details to you, then show you the second approach in its full glory.

Brute Force. We assert the desired behavior by *declaring* it to be true (and confirming that our guess results in a unitary transformation). Here, that means stating that $QNOT_x |0\rangle_x = |1\rangle_x$ and $QNOT_x |1\rangle_x = |0\rangle_x$. Express this as two equations involving the matrix for $QNOT_x$ and the x -basis kets in coordinate form (everything in z -basis coordinates, of course). You'll get four simultaneous equations for the four unknown matrix elements of $QNOT_x$. This creates a definition in terms of the natural CBS. We confirm it's unitary and we've got our gate.

[**Exercise.** Fill in the details for $QNOT_x$.]

Gate Combination. We know that $QNOT$ (i.e., X) swaps the z -CBS, and H converts between z -CBS and x -CBS. So we use H to map the x -basis to the z -basis, apply X to the z -basis and convert the results back to the x -basis:

$$- \boxed{H} - \boxed{X} - \boxed{H} -$$

Let's confirm that our plan worked.

Note. *Gates* operate from left-to-right, but *operator algebra* moves from right-to-left. When translating a circuit into a product of matrices, we must *reverse the order*. In this case, we can't "see" that effect, since the gate is symmetric, but it won't always be, so stay alert.

Substituting the matrices for their gates (and reversing order), we get

$$\begin{aligned} QNOT_x &\equiv (H)(X)(H) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \end{aligned}$$

and we have our matrix. It's easy to confirm that the matrix swaps the x -CBS kets and is identical to the matrix we would get using brute force (I'll let you check that).

Circuit Identities

The above example has a nice side-effect. By comparing the result with one of our basic gates, we find that $H \rightarrow X \rightarrow H$ is equivalent to Z ,

$$\text{---} \boxed{H} \text{---} \boxed{X} \text{---} \boxed{H} \text{---} = \text{---} \boxed{Z} \text{---}$$

There are more circuit identities that can be generated, some by looking at the matrices, and others by thinking about the effects of the constituent gates and confirming your guess through matrix multiplication.

Here is one you can verify,

$$\text{---} \boxed{H} \text{---} \boxed{Z} \text{---} \boxed{H} \text{---} = \text{---} \boxed{X} \text{---} ,$$

and here's another,

$$\text{---} \boxed{Z} \text{---} \boxed{Z} \text{---} = \text{---} \boxed{X} \text{---} \boxed{X} \text{---} = \text{---} \boxed{Y} \text{---} \boxed{Y} \text{---} = \text{---} \boxed{1} \text{---} .$$

The last pattern is true for any quantum logic gate, U , which is *self-adjoint* because then $U^2 = U^\dagger U = 1$, the first equality by “self-adjoint-ness” and the second by unitarity.

Some operator equivalences are not shown in gate form, but rather using the algebraic operators. For example

$$XZ = -iY$$

or

$$XYZ = -ZYX = i1 .$$

That's because the algebra shows a global phase factor which may appear awkward in gate form yet is still important if the combination is to be used in a larger circuit. As you may recall, even though a phase factor may not have observable consequences on the state alone, if that state is combined with other states prior to measurement, the *global* phase factor can turn into a *relative* phase difference, which *does* have observable consequences.

I will finish by reminding you that the algebra and the circuit are read in opposite order. Thus

$$XYZ = i1 .$$

corresponds to the circuit diagram

$$\text{---} \boxed{Z} \text{---} \boxed{Y} \text{---} \boxed{X} \text{---} = \text{---} \boxed{i1} \text{---}$$

This completes the basics of Qubits and their unary operators. There is one final topic that every quantum computer scientist should know. It is not going to be used much in this course, but will appear in CS 83B and CS 83C. It belongs in this chapter, so consider it recommended, but not required.

9.7 The Bloch Sphere

9.7.1 Introduction

Our goal is to find a visual 3-D representation for the qubits in \mathcal{H} . To that end, we will briefly allude to the lecture on quantum mechanics.

If you studied the optional *time evolution* of a general spin state corresponding to a special physical system – an electron in constant magnetic field \mathbf{B} – you learned that the *expectation value* of all three observables formed a *real* 3-D time-evolving vector,

$$\mathbf{s}(t) \equiv \begin{pmatrix} \langle S_x \rangle_{|\psi(t)\rangle} \\ \langle S_y \rangle_{|\psi(t)\rangle} \\ \langle S_z \rangle_{|\psi(t)\rangle} \end{pmatrix}.$$

which *precesses* around the direction of \mathbf{B} .

Do Not Panic. You don't have to remember or even re-study that section. This is merely a reference in case you want to connect that material with the following, which is otherwise self-contained.

The time evolution started out in an *initial spin state* at time $t = 0$, and we followed its development at later times. However, today, we need only consider the fixed state at its *initial* time. No evolution is involved.

With that scary introduction, let's calmly look at any general state's expansion along the preferred CBS,

$$|\psi\rangle = c_1 |0\rangle + c_2 |1\rangle.$$

9.7.2 Rewriting $|\psi\rangle$

We start by finding a more informative representation of $|\psi\rangle$. First, express c_1 and c_2 in *polar form*, giving the equivalent state

$$|\psi\rangle = \begin{pmatrix} c e^{i\phi_1} \\ s e^{i\phi_2} \end{pmatrix}.$$

Then multiply by the unit scalar $e^{-i(\frac{\phi_1+\phi_2}{2})}$ to get a more balanced equivalent state,

$$|\psi\rangle = \begin{pmatrix} c e^{i(\frac{\phi_1-\phi_2}{2})} \\ s e^{-i(\frac{\phi_1-\phi_2}{2})} \end{pmatrix}.$$

Now, simplify by making the substitution

$$\phi = \frac{\phi_1 - \phi_2}{2}$$

to get a balanced Hilbert space representative of our qubit,

$$|\psi\rangle = \begin{pmatrix} c e^{i\phi} \\ s e^{-i\phi} \end{pmatrix}.$$

As a qubit, $|\psi\rangle$ is normalized,

$$c^2 + s^2 = 1,$$

so c and s can be equated with the sine and cosine of some angle which we call $\frac{\theta}{2}$, i.e.,

$$\begin{aligned} c &= \cos \frac{\theta}{2} & \text{and} \\ s &= \sin \frac{\theta}{2}. \end{aligned}$$

We'll see why we pick $\frac{\theta}{2}$, not θ , next.

9.7.3 The Expectation Vector for $|\psi\rangle$

The three logic gates, X , Y and Z , are represented by unitary matrices, but they also happen to be *Hermitian*. This authorizes us to consider them *observables* defined by those matrices. In fact, we already related them to the observables S_x , S_y and S_z , the spin measurements along the principal axes, notwithstanding the removal of the factor $\frac{\hbar}{2}$. Therefore, each of these *observables* has an *expectation value* – a prediction about the *average* of many experiments in which we repeatedly send lots of qubits in the same state, $|\psi\rangle$, through one of these gates, measure the output (causing a *collapse*), compute the average for many trials, and do so for all three gates X , Y and Z . We define a real 3-D vector, \mathbf{s} , that collects the *expectation value* into one column, to be

$$\mathbf{s} \equiv \begin{pmatrix} \langle X \rangle_{|\psi\rangle} \\ \langle Y \rangle_{|\psi\rangle} \\ \langle Z \rangle_{|\psi\rangle} \end{pmatrix}.$$

At the end of the quantum mechanics lecture we essentially computed these expectation values. If you like, go back and plug $t = 0$ into the formula there. Aside from the factor of $\frac{\hbar}{2}$ (caused by the difference between $X/Y/Z$ and $S_x/S_y/S_z$), you will get

$$\mathbf{s} = \begin{pmatrix} \sin \theta \cos \phi \\ -\sin \theta \sin \phi \\ \cos \theta \end{pmatrix}.$$

By defining c and s in terms of $\frac{\theta}{2}$, we ended up with expectation values that had the whole angle, θ , in them. This is a unit vector in \mathbb{R}^3 whose spherical coordinates are $(1, \theta, -\phi)^t$, i.e., it has a polar angle θ and azimuthal angle $-\phi$. *It is a point on the unit sphere.*

9.7.4 Definition of the Bloch Sphere

The sphere in \mathbb{R}^3 defined by

$$\left\{ \hat{\mathbf{n}} \mid |\hat{\mathbf{n}}| = 1 \right\}$$

is called the *Bloch sphere* when the coordinates of each point on the sphere $\hat{\mathbf{n}} = (x, y, z)^t$ are interpreted as the three *expectation values* $\langle X \rangle$, $\langle Y \rangle$ and $\langle Z \rangle$ for some qubit state, $|\psi\rangle$. Each qubit value, $|\psi\rangle$, in \mathcal{H} corresponds to a point $\hat{\mathbf{n}}$ on the Bloch sphere.

If we use *spherical coordinates* to represent points on the sphere, then $\hat{\mathbf{n}} = (1, \theta, \phi)^t$ corresponds to the $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ in our Hilbert space \mathcal{H} according to

$$\hat{\mathbf{n}} = \begin{pmatrix} 1, \\ \theta, \\ \phi \end{pmatrix}_{\text{Sph}} \in \text{Bloch sphere} \quad \longleftrightarrow \quad |\psi\rangle = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) e^{-i\phi} \\ \sin\left(\frac{\theta}{2}\right) e^{i\phi} \end{pmatrix} \in \mathcal{H}.$$

Now we see that a polar angle, θ , of a point on the Bloch sphere gives the magnitudes of its corresponding qubit coordinates, but not *directly*; when θ is the polar angle, $\theta/2$ is used (through sine and cosine) for the qubit coordinate magnitudes.

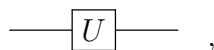
Chapter 10

Tensor Products

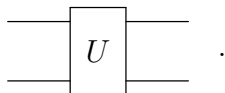
$$V \otimes W$$

10.1 Tensor Product for Quantum Computing

While *quantum unary gates* are far richer in variety and applicability than *classical unary gates*, there is only so much fun one can have with circuit elements, like



which have a single input. We need to combine qubits (a process called *quantum entanglement*), and to do that we'll need gates that have, at a minimum, two inputs,



(You may notice that there are also two outputs, an inevitable consequence of *unitarity* that we'll discuss in this hour.)

In order to feed two qubits into a *binary* quantum gate, we need a new tool to help us calculate, and that tool is the *tensor product* of the two single qubit state spaces

$$\mathcal{H} \otimes \mathcal{H}.$$

The concepts of tensors are no harder to master if we define the general tensor product of *any* two vector spaces, V and W of dimensions l and m , respectively,

$$V \otimes W,$$

and this approach will serve us well later in the course. We will then apply what we learn by setting $V = W = \mathcal{H}$.

The tensor product of more than two component spaces like

$$V_0 \otimes V_1 \otimes V_2 \otimes \cdots ,$$

or more relevant to us,

$$\mathcal{H} \otimes \mathcal{H} \otimes \mathcal{H} \otimes \cdots ,$$

presents no difficulty once we have mastered the “*order 2* tensors” (product of just two spaces). When we need a product of more than two spaces, as we shall in a future lecture, I’ll guide you. For now, let’s learn what it means to form the tensor product of just *two* vector spaces.

10.2 The Tensor Product of Two Vector Spaces

10.2.1 Definitions

Whenever we construct a new vector space like $V \otimes W$, we need to handle the required equipment. That means

1. specifying the *scalars* and *vectors*,
2. defining vector *addition* and *scalar multiplication*, and
3. confirming all the required *properties*.

Items 1 and 2 are easy, and we won’t be overly compulsive about **item 3**, so it should not be too painful. We’ll also want to cover the two – normally optional but for us *required* – topics,

4. defining the *inner product* and
5. establishing the *preferred basis*.

If you find this sort of abstraction drudgery, think about the fact that tensors are the requisite pillars of many fields including structural engineering, particle physics and general relativity. Your attention here will not go unrewarded.

Overview

The new vector space is based on the two vector spaces V (dimension = l) and W (dimension = m) and is called *tensor product* of V and W , written $V \otimes W$. The new space will turn out to have dimension = lm , the *product* of the two component space dimensions).

The Scalars of $V \otimes W$

Both V and W must have a common scalar set in order to form their inner product, and that set will be the scalar set for $V \otimes W$. For *real* vector spaces like \mathbb{R}^2 and \mathbb{R}^3 , the scalars for

$$\mathbb{R}^2 \otimes \mathbb{R}^3$$

would then be \mathbb{R} . For the Hilbert spaces of quantum physics (and quantum computing), the scalars are \mathbb{C} .

The Vectors in $V \otimes W$

Vectors of the tensor product space are formed in two stages. I like to Compartmentalize them as follows:

1. We start by populating $V \otimes W$ with the *formal symbols*

$$\mathbf{v} \otimes \mathbf{w}$$

consisting of one vector \mathbf{v} from V and another vector \mathbf{w} from W . $\mathbf{v} \otimes \mathbf{w}$ is called the *tensor product* of the two vectors \mathbf{v} and \mathbf{w} . There is no way to further merge these two vectors; $\mathbf{v} \otimes \mathbf{w}$ is as concise as their tensor product gets.

For example, in the case of $(5, -6)^t \in \mathbb{R}^2$ and $(\pi, 0, 3)^t \in \mathbb{R}^3$, they provide the tensor product vector

$$\begin{pmatrix} 5 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} \pi \\ 0 \\ 3 \end{pmatrix} \in \mathbb{R}^2 \otimes \mathbb{R}^3,$$

with no further simplification possible (notwithstanding the natural basis *coordinate* representation that we will get to, below).

Caution. No one is saying that all these formal products $\mathbf{v} \otimes \mathbf{w}$ are distinct from one another. When we define the *operations*, we'll see there is much duplication.

2. The formal vector products constructed in **step 1** produce only a small subset of the tensor product space $V \otimes W$ (that will come to be known as the “separable tensors of the product space.”) The most general vector is a finite sum of such symbols.

The full space $V \otimes W$ consists of all *finite* sums of the form

$$\sum_k \mathbf{v}_k \otimes \mathbf{w}_k, \quad \text{with} \\ \mathbf{v}_k \in V \quad \text{and} \quad \mathbf{w}_k \in W.$$

For example, in the case of the complex vector spaces $V = \mathbb{C}^3$ and $W = \mathbb{C}^4$, one such typical vector in the “product space” $\mathbb{C}^3 \otimes \mathbb{C}^4$ would be

$$\begin{pmatrix} 1+i \\ -6 \\ 3i \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} + \begin{pmatrix} 1-i \\ -6 \\ 3i \end{pmatrix} \otimes \begin{pmatrix} 1-i \\ \pi \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} i \\ i \\ \sqrt{7} \\ -i \end{pmatrix}.$$

Although this particular sum can’t be further simplified as a combination of “separable tensors” (those described in item 1, above), we can always simplify long sums so that there are at most lm terms in them. That’s because (as we’ll learn) there are lm *basis vectors* for the tensor product space.

Second Caution. Although these sums produce *all* the vectors in $V \otimes W$, they do so many times over. In other words, it will *not* be true that every sum created in this step is a distinct *tensor* from every *other* sum.

Vocabulary

- **Product Space.** The tensor product of two vector spaces is sometimes referred to as the *product space*.
- **Tensors.** Vectors in the product space are sometimes called *tensors*, emphasizing that they live in the tensor product space of two vector spaces. However, they are still vectors.
- **Separable Tensors.** Those vectors in $V \otimes W$ which arise in the **step 1** are called *separable tensors*; they can be “separated” into two component vectors, one from each space, whose product is the (separable) tensor. **Step 2** presages that most tensors in the product space are not separable. Separable tensors are sometimes called “pure” or “simple”.
- **Tensor Product.** We can use the term “tensor product” to mean either the product space or the individual separable tensors. Thus $V \otimes W$ is the *tensor product* of two spaces, while $\mathbf{v} \otimes \mathbf{w}$ is the (separable) *tensor product* of two vectors.

Vector Addition

This operation is built-into the definition of a tensor; since the general tensor is the sum of separable tensors, adding two of them merely produces another sum, which is automatically a tensor. The twist, if we can call it that, is how we equate any such sums that happen to represent the same tensor. This all expressed in the following two bullets.

- For any two tensors,

$$\zeta = \sum_k \mathbf{v}_k \otimes \mathbf{w}_k \quad \zeta' = \sum_j \mathbf{v}'_j \otimes \mathbf{w}'_j,$$

their vector sum is the combined sum, i.e.,

$$\zeta + \zeta' \equiv \sum_k \mathbf{v}_k \otimes \mathbf{w}_k + \sum_j \mathbf{v}'_j \otimes \mathbf{w}'_j,$$

which simply expresses the fact that a sum of two finite sums is itself a finite sum and therefore agrees with our original definition of a vector object in the product space. The sum may need simplification, but it is a valid object in the product space.

- The tensor product *distributes* over sums in the component space,

$$\begin{aligned} (\mathbf{v} + \mathbf{v}') \otimes \mathbf{w} &= \mathbf{v} \otimes \mathbf{w} + \mathbf{v}' \otimes \mathbf{w} \quad \text{and} \\ \mathbf{v} \otimes (\mathbf{w} + \mathbf{w}') &= \mathbf{v} \otimes \mathbf{w} + \mathbf{v} \otimes \mathbf{w}'. \end{aligned}$$

Practically, we only need to understand this as a “distributive property,” but in theoretical terms it has the effect of producing countless sets of equivalent vectors. That is, it tells how different formal sums in **step 2** of “The Vectors of $\mathbf{V} \otimes \mathbf{W}$ ” might represent the same actual tensor.

Commutativity. Vector addition commutes *by definition*; the formal sums in **step 2** are *declared* to be order-independent.

Such rules can be viewed as a way for different “looking” algebraic combinations of tensors to be declared identical, thereby causing infinitely many of the original formal sums to collapse to the same tensor. They confer a lawful organization to the product space. Let’s look at a few more properties that further this order-giving effect.

Scalar Multiplication

Let c be a scalar. We define its product with a tensor in two steps.

- **c Times a Separable Tensor.**

$$\begin{aligned} c(\mathbf{v} \otimes \mathbf{w}) &\equiv (c\mathbf{v}) \otimes \mathbf{w} \\ &\equiv \mathbf{v} \otimes (c\mathbf{w}). \end{aligned}$$

This definition requires – *declares* – that it doesn’t matter which component of the separable tensor gets the c . Either way, the two tensors formed are the same. It has the effect of establishing the equivalence of initially distinct formal symbols $\mathbf{v} \otimes \mathbf{w}$ in **step 1** of “The Vectors of $\mathbf{V} \otimes \mathbf{W}$.”

- **c Times a Sum of (Separable) Tensors.**

$$c(\mathbf{v}_0 \otimes \mathbf{w}_0 + \mathbf{v}_1 \otimes \mathbf{w}_1) \equiv c(\mathbf{v}_0 \otimes \mathbf{w}_0) + c(\mathbf{v}_1 \otimes \mathbf{w}_1).$$

Because any tensor can be written as the finite sum of separable tensors, this requirement covers the balance of the product space. You can distribute c over as large a sum as you like.

- **Order.** While we can place c on either the left or right of a vector, it is usually placed on the left, as in ordinary vector spaces.

The Requisite Properties of $V \otimes W$

Properties like *commutative* addition, *distributivity* of scalar multiplication are automatic consequences of the above definitions. I'll leave it as an optional exercise.

[**Exercise.** Prove that $V \otimes W$ satisfies the various operational requirements of a vector space.]

Inner Products in $V \otimes W$

If V and W possess inner products, an inner product is conferred to the product space $V \otimes W$ by

$$\langle \mathbf{v} \otimes \mathbf{w} \mid \mathbf{v}' \otimes \mathbf{w}' \rangle \equiv \langle \mathbf{v} \mid \mathbf{v}' \rangle \cdot \langle \mathbf{w} \mid \mathbf{w}' \rangle ,$$

where “ \cdot ” on the RHS is scalar multiplication. This only defines the tensor product of two separable tensors, however we extend this to *all* tensors by asserting a distributive property (or if you prefer the terminology, by “extending linearly”). For example,

$$\begin{aligned} & \langle \mathbf{v} \otimes \mathbf{w} \mid \mathbf{v}' \otimes \mathbf{w}' + \mathbf{v}'' \otimes \mathbf{w}'' \rangle \\ & \equiv \langle \mathbf{v} \otimes \mathbf{w} \mid \mathbf{v}' \otimes \mathbf{w}' \rangle + \langle \mathbf{v} \otimes \mathbf{w} \mid \mathbf{v}'' \otimes \mathbf{w}'' \rangle . \end{aligned}$$

[**Exercise.** Compute the inner product in $\mathbb{R}^2 \otimes \mathbb{R}^3$

$$\left\langle \begin{pmatrix} 5 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} \pi \\ 0 \\ 3 \end{pmatrix} \mid \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \right\rangle \Bigg]$$

[**Exercise.** Compute the inner product in $\mathbb{C}^3 \otimes \mathbb{C}^4$

$$\left\langle \begin{pmatrix} 1+i \\ -6 \\ 3i \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} + \begin{pmatrix} -i \\ -6 \\ 3i \end{pmatrix} \otimes \begin{pmatrix} i \\ \pi \\ 0 \\ 1 \end{pmatrix} \mid \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ i \\ 1 \\ i \end{pmatrix} \right\rangle \Bigg]$$

[**Exercise.** Prove that the definition of inner product satisfies all the usual requirements of a dot or inner product. Be sure to cover *distributivity* and *positive definiteness*.]

The Natural Basis for $V \otimes W$

Of all the aspects of tensor products, the one that we will use most frequently is the preferred basis.

Tensor Product Basis Theorem (Orthonormal Version). *If V has dimension l with orthonormal basis*

$$\left\{ \mathbf{v}_k \right\}_{k=0}^{l-1},$$

and W has dimension m , with orthonormal basis

$$\left\{ \mathbf{w}_j \right\}_{j=0}^{m-1},$$

then $V \otimes W$ has dimension lm and “inherits” a natural (preferred) orthonormal basis

$$\left\{ \mathbf{v}_k \otimes \mathbf{w}_j \right\}_{j,k=0}^{l-1, m-1}$$

from V and W . For example, the natural basis for $\mathbb{R}^2 \otimes \mathbb{R}^3$ is

$$\left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \right. \\ \left. \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}$$

Proof of Basis Theorem. I'll guide you through the proof, and you can fill in the gaps as an exercise if you care to.

Spanning. A basis must span the space. We need to show that any tensor can be expressed as a linear combination of the alleged basis vectors $\mathbf{v}_k \otimes \mathbf{w}_j$. This is an easy two parter:

1. Any $\mathbf{v} \in V$ can be expanded along the V - basis as

$$\mathbf{v} = \sum \alpha_k \mathbf{v}_k,$$

and any $\mathbf{w} \in W$ can be expanded along the W - basis as

$$\mathbf{w} = \sum \beta_j \mathbf{w}_j,$$

which implies that any *separable* tensor $\mathbf{v} \otimes \mathbf{w}$ can be expressed

$$\mathbf{v} \otimes \mathbf{w} = \sum \alpha_k \beta_j (\mathbf{v}_k \otimes \mathbf{w}_j).$$

[**Exercise.** Prove the last identity by applying linearity (distributive properties of the tensor product space).]

2. Any tensor is a sum of separable tensors, so item 1 tells us that it, too, can be expressed as a linear combination of $\mathbf{v}_k \otimes \mathbf{w}_j$. [**Exercise.** Demonstrate this algebraically.]

Linear Independence and Orthonormality. We rely on a little theorem to which I subjected you *twice*, first in the linear algebra lecture, then again in the Hilbert space lecture. It said that an *orthonormality* \Rightarrow *linearly independence*. We now show that the set $\{\mathbf{v}_k \otimes \mathbf{w}_j\}$ is orthonormal collection of vectors.

$$\begin{aligned} \langle \mathbf{v}_k \otimes \mathbf{w}_j | \mathbf{v}_{k'} \otimes \mathbf{w}_{j'} \rangle &= \langle \mathbf{v}_k | \mathbf{v}_{k'} \rangle \langle \mathbf{w}_j | \mathbf{w}_{j'} \rangle \\ &= \begin{cases} 1, & \text{if } k = k' \text{ and } j = j' \\ 0, & \text{otherwise} \end{cases} . \end{aligned}$$

This is the definition of orthonormality, so by our little theorem the vectors in $\{\mathbf{v}_k \otimes \mathbf{w}_j\}$ are linearly independent. QED

The basis theorem works even if the component bases are not orthonormal. Of course, in that case, the inherited tensor basis is not orthonormal.

Tensor Product Basis Theorem (General Version). *If V has dimension l with basis*

$$\left\{ \mathbf{v}_k \right\}_{k=0}^{l-1} ,$$

and W has dimension m , with basis

$$\left\{ \mathbf{w}_j \right\}_{j=0}^{m-1} ,$$

then $V \otimes W$ has dimension lm and “inherits” a natural (preferred) basis

$$\left\{ \mathbf{v}_k \otimes \mathbf{w}_j \right\}_{j, k = 0, 0}^{l-1, m-1}$$

from V and W .

If V and W each have an inner product (true for any of our vector spaces) this theorem follows immediately from the orthonormal basis theorem.

[**Exercise.** Give a one sentence proof of this theorem based on the orthonormal product basis theorem.]

The theorem is still true, even if the two spaces don't have inner products, but we won't bother with that version.

Practical Summary of the Tensor Product Space

While we have outlined a rigorous construction for a tensor product space, it is usually good enough for computer scientists to characterize the product space in terms of the tensor basis.

The produce space, $U = V \otimes W$ consists of *tensors*, \mathbf{u} , expressible as sums of the *separable basis*

$$\left\{ \mathbf{v}_k \otimes \mathbf{w}_j \mid j = 0, \dots, (n-1), \quad k = 0, \dots, (m-1) \right\},$$

weighted by *scalar* weights c_{kj} ,

$$\mathbf{u} = \sum_{\substack{k=0 \\ j=0}}^{(n-1)(m-1)} c_{kj} (\mathbf{v}_k \otimes \mathbf{w}_j).$$

The nm basis tensors, $\mathbf{v}_k \otimes \mathbf{w}_j$, are induced by the two component bases,

$$\begin{aligned} \mathcal{V} &= \left\{ \mathbf{v}_k \right\}_{k=0}^{n-1} \quad \text{and} \\ \mathcal{W} &= \left\{ \mathbf{w}_j \right\}_{j=0}^{m-1}. \end{aligned}$$

The *sums*, *products* and *equivalence* of tensor expressions are defined by the required *distributive* and *commutative* properties listed earlier, but can often be taken as the natural rules one would expect.

Conventional Order of Tensor Basis

While not universal, when we need to list the tensor basis linearly, the most common convention is to let the left basis index increment slowly and the right increment quickly. It is “*V-major / W-minor format*” if you will, an echo of the *row-major* (*column-minor*) ordering choice of arrays in computer science,

$$\begin{aligned} &\left\{ \mathbf{v}_0 \otimes \mathbf{w}_0, \quad \mathbf{v}_0 \otimes \mathbf{w}_1, \quad \mathbf{v}_0 \otimes \mathbf{w}_2, \quad \dots, \quad \mathbf{v}_0 \otimes \mathbf{w}_{m-1}, \right. \\ &\quad \mathbf{v}_1 \otimes \mathbf{w}_0, \quad \mathbf{v}_1 \otimes \mathbf{w}_1, \quad \mathbf{v}_1 \otimes \mathbf{w}_2, \quad \dots, \quad \mathbf{v}_1 \otimes \mathbf{w}_{m-1}, \\ &\quad \ddots \\ &\quad \left. \mathbf{v}_{l-1} \otimes \mathbf{w}_0, \quad \mathbf{v}_{l-1} \otimes \mathbf{w}_1, \quad \mathbf{v}_{l-1} \otimes \mathbf{w}_2, \quad \mathbf{v}_{l-1} \otimes \mathbf{w}_{m-1} \right\}. \end{aligned}$$

You might even see these basis tensors labeled using the shorthand like ζ_{kj} ,

$$\begin{aligned} &\left\{ \zeta_{00}, \quad \zeta_{01}, \quad \zeta_{02}, \quad \dots, \quad \zeta_{0(m-1)}, \right. \\ &\quad \zeta_{10}, \quad \zeta_{11}, \quad \zeta_{12}, \quad \dots, \quad \zeta_{1(m-1)}, \\ &\quad \ddots \\ &\quad \left. \zeta_{(l-1)0}, \quad \zeta_{(l-1)1}, \quad \zeta_{(l-1)2}, \quad \dots, \quad \zeta_{(l-1)(m-1)} \right\}. \end{aligned}$$

10.2.2 Tensor Coordinates from Component-Space Coordinates

We have defined everything relating to a tensor product space $V \otimes W$. Before we apply that to qubits, we must make sure we can quickly write down *coordinates* of a tensor in the *preferred (natural) basis*. This starts, as always, with (i) *separable tensors*, from which we move to the (ii) *basis tensors*, and finally graduate to (iii) *general tensors*.

Natural Coordinates of Separable Tensors

We are looking for the coordinates of a pure tensor, expressible as a product of two component vectors whose preferred coordinates we already know,

$$\mathbf{v} \otimes \mathbf{w} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{l-1} \end{pmatrix} \otimes \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \end{pmatrix}.$$

[**A Reminder.** We are numbering starting with 0, rather than 1, now that we are in *computing* lessons.]

I'm going to give you the answer immediately, and allow you to skip the explanation if you are in a hurry.

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{l-1} \end{pmatrix} \otimes \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \end{pmatrix} = \begin{pmatrix} c_0 \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \end{pmatrix} \\ c_1 \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \end{pmatrix} \\ \vdots \\ c_{l-1} \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} c_0 d_0 \\ c_0 d_1 \\ \vdots \\ c_0 d_{m-1} \\ c_1 d_0 \\ c_1 d_1 \\ \vdots \\ c_1 d_{m-1} \\ c_2 d_0 \\ c_2 d_1 \\ \vdots \\ c_2 d_{m-1} \\ \vdots \\ c_{l-1} d_0 \\ c_{l-1} d_1 \\ \vdots \\ c_{l-1} d_{m-1} \end{pmatrix}.$$

Example. For $(5, -6)^t \in \mathbb{R}^2$ and $(\pi, 0, 3)^t \in \mathbb{R}^3$, their tensor product in $\mathbb{R}^2 \otimes \mathbb{R}^3$ has natural basis coordinates given by

$$\begin{pmatrix} 5 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} \pi \\ 0 \\ 3 \end{pmatrix} = \begin{pmatrix} 5\pi \\ 0 \\ 15 \\ -6\pi \\ 0 \\ -18 \end{pmatrix}.$$

[**Exercise.** Give the coordinate representation of the tensor

$$\begin{pmatrix} 1+i \\ -6 \\ 3i \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

in the natural tensor basis.]

Explanation. I'll demonstrate the validity of the formula in the special case $\mathbb{R}^2 \otimes \mathbb{R}^3$. The derivation will work for any V and W (as the upcoming exercise shows).

Consider the tensor product of two general vectors,

$$\begin{pmatrix} c_0 \\ c_1 \end{pmatrix} \otimes \begin{pmatrix} d_0 \\ d_1 \\ d_2 \end{pmatrix}.$$

Let's zoom in on the meaning of each column vector which is, after all, shorthand notation for the following,

$$\left[c_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + c_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right] \otimes \left[d_0 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + d_1 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + d_2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right].$$

Now apply the linearity to equate the above with

$$\begin{aligned} & c_0 d_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + c_0 d_1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c_0 d_2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ & + c_1 d_0 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + c_1 d_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + c_1 d_2 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \end{aligned}$$

Next, identify each of the basis tensor products with their symbolic \mathbf{v}_k (for \mathbb{R}^2) and \mathbf{w}_j (for \mathbb{R}^3) to see it more clearly,

$$\begin{aligned} & c_0 d_0 (\mathbf{v}_0 \otimes \mathbf{w}_0) + c_0 d_1 (\mathbf{v}_0 \otimes \mathbf{w}_1) + c_0 d_2 (\mathbf{v}_0 \otimes \mathbf{w}_2) \\ & + c_1 d_0 (\mathbf{v}_1 \otimes \mathbf{w}_0) + c_1 d_1 (\mathbf{v}_1 \otimes \mathbf{w}_1) + c_1 d_2 (\mathbf{v}_1 \otimes \mathbf{w}_2). \end{aligned}$$

The basis tensors are listed in the conventional (“*V*-major / *W*-minor format”) allowing us to write down the coordinates of the tensor,

$$\begin{pmatrix} c_0 d_0 \\ c_0 d_1 \\ c_0 d_2 \\ c_1 d_0 \\ c_1 d_1 \\ c_1 d_2 \end{pmatrix},$$

as claimed. QED

[**Exercise.** Replicate the demonstration for the coordinates of a separable tensor $\mathbf{v} \otimes \mathbf{w}$ in any product space $\mathbf{V} \otimes \mathbf{W}$.]

Natural Coordinates of Basis Tensors

By definition and a previous exercise (see the lecture on *linear transformations*) we know that each basis vector, \mathbf{b}_k , when expressed in its own \mathcal{B} -coordinates looks exactly like the k th preferred basis element, i.e.,

$$\mathbf{b}_k \Big|_{\mathcal{B}} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow k\text{th element}.$$

The is true for our tensor basis, which means that once we embrace that basis, we have lm basis vectors

$$\left\{ \begin{pmatrix} 1 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \end{pmatrix} \right\} lm,$$

which don’t make reference to the two component vector spaces or the inherited *V-major / W-minor ordering* we decided to use. However, for this to be useful, we need an implied correspondence between these vectors and the inherited basis

$$\left\{ \begin{array}{l} \mathbf{v}_0 \otimes \mathbf{w}_0, \quad \mathbf{v}_0 \otimes \mathbf{w}_1, \quad \mathbf{v}_0 \otimes \mathbf{w}_2, \quad \dots, \quad \mathbf{v}_0 \otimes \mathbf{w}_{m-1}, \\ \mathbf{v}_1 \otimes \mathbf{w}_0, \quad \mathbf{v}_1 \otimes \mathbf{w}_1, \quad \mathbf{v}_1 \otimes \mathbf{w}_2, \quad \dots, \quad \mathbf{v}_1 \otimes \mathbf{w}_{m-1}, \\ \vdots \\ \mathbf{v}_{l-1} \otimes \mathbf{w}_0, \quad \mathbf{v}_{l-1} \otimes \mathbf{w}_1, \quad \mathbf{v}_{l-1} \otimes \mathbf{w}_2, \quad \mathbf{v}_{l-1} \otimes \mathbf{w}_{m-1} \end{array} \right\}.$$

This is all fine, as long as we remember that those self-referential tensor bases assume some agreed-upon ordering system, and in this course that will be the *V-major* ordering.

To illustrate this, say we are working with the basis vector

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \in \mathbb{R}^2 \otimes \mathbb{R}^3.$$

In the rare times when we need to relate this back to our original vector spaces we would count: The 1 is in position 4 (counting from 0), and relative to \mathbb{R}^2 and \mathbb{R}^3 this means

$$4 = \mathbf{1} \times 3 + \mathbf{1},$$

yielding the individual basis vectors $\#1$ in V and $\#1$ in W . Therefore, the correspondence is

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \longleftrightarrow \mathbf{v}_1 \otimes \mathbf{w}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

which we can confirm by multiplying out the RHS as we learned in the last section

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \cdot 0 \\ 0 \cdot 1 \\ 0 \cdot 0 \\ 1 \cdot 0 \\ 1 \cdot 1 \\ 1 \cdot 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

This will be easy enough with our 2-dimensional component spaces \mathcal{H} , but if you aren't prepared for it, you might find yourself drifting aimlessly when faced with a long column basis tensor and don't know what to do with it.

Natural Coordinates of General Tensors

Because the tensor space has dimension lm , we know that any tensor in the natural basis will look like

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{lm-2} \\ \zeta_{lm-1} \end{pmatrix}.$$

The only thing worth noting here is the correspondence between this and the component spaces V and W . If we are lucky enough to have a *separable* tensor in our hands, this would have the special form

$$\begin{pmatrix} c_0 d_0 \\ c_0 d_1 \\ c_0 d_2 \\ \vdots \\ c_1 d_0 \\ c_1 d_1 \\ c_1 d_2 \\ \vdots \\ c_2 d_0 \\ c_2 d_1 \\ c_2 d_2 \\ \vdots \end{pmatrix},$$

and we might be able to figure out the component vectors from this. However, in general, we don't have separable tensors. All we can say is that this tensor is a linear combination of the lm basis vectors, and just accept that it has the somewhat random components, ζ_i which we might label simply

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \zeta_2 \\ \vdots \\ \zeta_{lm-2} \\ \zeta_{lm-1} \end{pmatrix},$$

or we might label with an eye on our component spaces

$$\begin{pmatrix} \zeta_{00} \\ \zeta_{01} \\ \zeta_{02} \\ \vdots \\ \zeta_{0(m-1)} \\ \zeta_{10} \\ \zeta_{11} \\ \zeta_{12} \\ \vdots \\ \zeta_{1(m-1)} \\ \zeta_{20} \\ \zeta_{21} \\ \zeta_{22} \\ \vdots \\ \zeta_{2(m-1)} \\ \vdots \end{pmatrix},$$

with the awareness that these components ζ_{kj} *may not* be products of two factors $c_k d_j$ originating in two vectors $(c_0, \dots, c_{l-1})^t$ and $(d_0, \dots, d_{m-1})^t$.

One thing we *do* know: Any tensor can be written as a weighted-sum of, at most, lm separable tensors. (If this is not immediately obvious, please review the *tensor product basis theorem*.)

Example. Let's compute the coordinates of the non-separable tensor

$$\zeta = \begin{pmatrix} 3 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ \pi \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

in the natural basis. We do it by applying the above formula to each separable component and adding,

$$\zeta = \begin{pmatrix} 3 \begin{pmatrix} 1 \\ 2 \\ \pi \end{pmatrix} \\ -6 \begin{pmatrix} 1 \\ 2 \\ \pi \end{pmatrix} \end{pmatrix} + \begin{pmatrix} 1 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \\ 0 \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 3 \\ 6 \\ 3\pi \\ -6 \\ -12 \\ -6\pi \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \\ 1 + 3\pi \\ -6 \\ -12 \\ -6\pi \end{pmatrix}.$$

Tensors as Matrices

If the two component spaces have dimension l and m , we know that the product space has dimension lm . More recently we've been talking about how these lm coordinates

might be organized. Separable or not, a tensor is completely determined by its lm preferred coefficients, which suggests a rectangle of numbers,

$$\begin{pmatrix} \zeta_{00} & \zeta_{01} & \zeta_{02} & \cdots & \zeta_{0(m-1)} \\ \zeta_{10} & \zeta_{11} & \zeta_{12} & \cdots & \zeta_{1(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \zeta_{(l-1)0} & \zeta_{(l-1)1} & \zeta_{(l-1)2} & \cdots & \zeta_{(l-1)(m-1)} \end{pmatrix},$$

which happen to have a more organized structure in the separable case,

$$\begin{pmatrix} c_0 d_0 & c_0 d_1 & c_0 d_2 & \cdots & c_0 d_{m-1} \\ c_1 d_0 & c_1 d_1 & c_1 d_2 & \cdots & c_1 d_{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{l-1} d_0 & c_{l-1} d_1 & c_{l-1} d_2 & \cdots & c_{l-1} d_{m-1} \end{pmatrix}.$$

This matrix is not to be interpreted as a linear transformation of either component space – it is just a vector in the product space. (It *does* have a meaning as scalar-valued function, but we’ll leave that as a topic for courses in relativity, particle physics or structural engineering.)

Sometimes the lm *column vector* model serves tensor imagery perfectly well, while other times the $l \times m$ *matrix* model works better. It’s good to be ready to use either one as the situation demands.

10.3 Linear Operators on the Tensor Product Space

The product space is a *vector space* and, as such, supports *linear transformations*. Everything we know about them applies here: they have *matrix representations* in the natural basis, some are *unitary*, some not, some are *Hermitian*, some not, some have *inverses*, some don’t, etc. The only special and new topic that confronts us is how the linear transformation of the two component spaces, V and W , inform those of the product space, $V \otimes W$.

The situation will feel familiar. Just as tensors in $V \otimes W$ fall into two main classes, those that are *separable* (products, $\mathbf{v} \otimes \mathbf{w}$, of two vectors) and those that aren’t (they are built from sums of separable tensors), the *operators* on $V \otimes W$ have a corresponding breakdown.

10.3.1 Separable Operators

If A is a linear transformation (a.k.a. an “operator”) on V

$$A : V \longrightarrow V,$$

and B is a linear transformation on W

$$B : W \longrightarrow W,$$

then their *tensor product*, $A \otimes B$ is a linear transformation on the product space $V \otimes W$,

$$A \otimes B : V \otimes W \longrightarrow V \otimes W,$$

defined by its action on the separable tensors

$$[A \otimes B](\mathbf{v} \otimes \mathbf{w}) \equiv A\mathbf{v} \otimes B\mathbf{w},$$

and extended to general tensors linearly.

Note 1: We could have defined $A \otimes B$ first on just the lm basis vectors $\mathbf{v}_k \otimes \mathbf{w}_j$, since they span the space. However, it's so useful to remember that we can use this formula on any two component vectors \mathbf{v} and \mathbf{w} , that I prefer to make this the official definition.

Note 2: A and/or B need not map their respective vector spaces into themselves. For example, perhaps $A : V \mapsto V'$ and $B : W \mapsto W'$. Then $A \otimes B : V \otimes W \mapsto V' \otimes W'$. However, we will usually encounter the simpler case covered by the definition above.

One must verify that this results in a linear transformation (operator) on the product space by proving that for an $\boldsymbol{\zeta}, \boldsymbol{\eta} \in V \otimes W$ and scalar c ,

$$\begin{aligned} [A \otimes B](\boldsymbol{\zeta} + \boldsymbol{\eta}) &= [A \otimes B]\boldsymbol{\zeta} + [A \otimes B]\boldsymbol{\eta} \quad \text{and} \\ [A \otimes B](c\boldsymbol{\zeta}) &= c[A \otimes B]\boldsymbol{\zeta}. \end{aligned}$$

This is very easy to do as an ...

[**Exercise.** Prove this by first verifying it on separable tensors then showing that the extension to general tensors preserves the properties.]

Example. Let A be defined on \mathbb{R}^2 by

$$A\mathbf{v} = A \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} \equiv \begin{pmatrix} v_0 + 2v_1 \\ v_1 \end{pmatrix}$$

and B be defined on \mathbb{R}^3 by

$$B\mathbf{w} \equiv \pi\mathbf{w} = \begin{pmatrix} \pi w_0 \\ \pi w_1 \\ \pi w_2 \end{pmatrix}.$$

On separable tensors, then, $A \otimes B$ has the effect expressed by

$$[A \otimes B](\mathbf{v} \otimes \mathbf{w}) = \begin{pmatrix} v_0 + 2v_1 \\ v_1 \end{pmatrix} \otimes \begin{pmatrix} \pi w_0 \\ \pi w_1 \\ \pi w_2 \end{pmatrix},$$

and this is extended linearly to general tensors. To get specific, we apply $A \otimes B$ to the tensor

$$\boldsymbol{\zeta} = \begin{pmatrix} 3 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ \pi \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

to get

$$\begin{aligned}
[A \otimes B]\zeta &= \begin{pmatrix} 3 & -12 \\ & -6 \end{pmatrix} \otimes \begin{pmatrix} \pi \\ 2\pi \\ \pi^2 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ \pi \\ \pi \end{pmatrix} \\
&= \begin{pmatrix} -9 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} \pi \\ 2\pi \\ \pi^2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ \pi \\ \pi \end{pmatrix},
\end{aligned}$$

with no simplification obvious. However, an equivalent expression can be formed by extracting a factor of π :

$$[A \otimes B]\zeta = \pi \left[\begin{pmatrix} -9 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ \pi \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right].$$

We can always forsake the separable components and instead express this as a column vector in the product space by adding the two separable tensors,

$$\begin{aligned}
[A \otimes B]\zeta &= \begin{pmatrix} -9 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} \pi \\ 2\pi \\ \pi^2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ \pi \\ \pi \end{pmatrix} \\
&= \begin{pmatrix} -9\pi \\ -18\pi \\ -9\pi^2 \\ -6\pi \\ -12\pi \\ -6\pi^2 \end{pmatrix} + \begin{pmatrix} 0 \\ \pi \\ \pi \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -9\pi \\ -17\pi \\ \pi - 9\pi^2 \\ -6\pi \\ -12\pi \\ -6\pi^2 \end{pmatrix}.
\end{aligned}$$

10.3.2 The Matrix of a Separable Operator

We can write down the matrix for any separable operator using rules similar to those that gave us the coordinates of a separable tensor. As you recall, we used a *V-major format* which multiplied the entire vector \mathbf{w} by each coordinate of \mathbf{v} to build the result. The same will work here. We use an *A-major format*, i.e., the left operator's coordinates change more “slowly” than the right operator's. Stated differently, we

multiply each element of the left matrix by the entire matrix on the right.

$$\begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0(l-1)} \\ a_{10} & a_{11} & \cdots & a_{1(l-1)} \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix} \otimes \begin{pmatrix} b_{00} & b_{01} & \cdots & b_{0(m-1)} \\ b_{10} & b_{11} & \cdots & b_{1(m-1)} \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix} \\
= \begin{pmatrix} a_{00} \begin{pmatrix} b_{00} & b_{01} & \cdots \\ b_{10} & b_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} & a_{01} \begin{pmatrix} b_{00} & b_{01} & \cdots \\ b_{10} & b_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} & \cdots \\ a_{10} \begin{pmatrix} b_{00} & b_{01} & \cdots \\ b_{10} & b_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} & a_{11} \begin{pmatrix} b_{00} & b_{01} & \cdots \\ b_{10} & b_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

This works based on a *V-major column format* for the vectors in the product space. If we had used a *W-major* column format, then we would have had to define the product matrix using a *B-major* rule rather than the *A-major* rule given above.

Example. The matrices for the *A* and *B* of our last example are given by

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} \pi & 0 & 0 \\ 0 & \pi & 0 \\ 0 & 0 & \pi \end{pmatrix},$$

so the tensor product transformation $A \otimes B$ is immediately written down as

$$A \otimes B = \begin{pmatrix} \pi & 0 & 0 & 2\pi & 0 & 0 \\ 0 & \pi & 0 & 0 & 2\pi & 0 \\ 0 & 0 & \pi & 0 & 0 & 2\pi \\ 0 & 0 & 0 & \pi & 0 & 0 \\ 0 & 0 & 0 & 0 & \pi & 0 \\ 0 & 0 & 0 & 0 & 0 & \pi \end{pmatrix}.$$

Example. We've already applied the definition of a tensor operator directly to compute the above operator applied to the tensor

$$\zeta = \begin{pmatrix} 3 \\ -6 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \\ \pi \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 7 \\ 1+3\pi \\ -6 \\ -12 \\ -6\pi \end{pmatrix}$$

and found that, after reducing the result to a column vector,

$$[A \otimes B]\zeta = \begin{pmatrix} -9\pi \\ -17\pi \\ \pi - 9\pi^2 \\ -6\pi \\ -12\pi \\ -6\pi^2 \end{pmatrix}$$

It now behooves us to do a sanity check. We must confirm that multiplication of ζ by the imputed matrix for $A \otimes B$ will produce the same result.

$$[A \otimes B]\zeta = \begin{pmatrix} \pi & 0 & 0 & 2\pi & 0 & 0 \\ 0 & \pi & 0 & 0 & 2\pi & 0 \\ 0 & 0 & \pi & 0 & 0 & 2\pi \\ 0 & 0 & 0 & \pi & 0 & 0 \\ 0 & 0 & 0 & 0 & \pi & 0 \\ 0 & 0 & 0 & 0 & 0 & \pi \end{pmatrix} \begin{pmatrix} 3 \\ 7 \\ 1 + 3\pi \\ -6 \\ -12 \\ -6\pi \end{pmatrix} = ?$$

[**Exercise.** Do the matrix multiplication, fill in the question mark, and see if it agrees with the column tensor above.]

10.3.3 The Matrix of a General Operator

As with the vectors in the product space (i.e., *tensors*), we can represent any *operator* on the product space as sum of no more than $(lm)^2$ separable operators, since there are that many elements in a tensor product matrix. The following exercise will make this clear.

[**Exercise.** Let the dimensions of our two component spaces be l and m . Then pick two integers p and q in the range $0 \leq p, q < lm$. Show that the matrix

$$P_{pq}$$

which has a 1 in position (p, q) and 0 in all other positions, is separable. **Hint:** You need to find an $l \times l$ matrix and an $m \times m$ matrix whose tensor product has 1 in the right position and 0s everywhere else. Start by partitioning P_{pq} into sub-matrices of size $m \times m$. Which sub-matrix does the lonely 1 fall into? Where in that $m \times m$ sub-matrix does that lonely 1 fall?]

[**Exercise.** Show that the set of all $(lm)^2$ matrices

$$\left\{ P_{pq} \mid 0 \leq p, q < lm \right\},$$

“spans” the set of linear transformations on $V \otimes W$.]

10.3.4 Food for Thought

Before we move on to multi-qubit systems, here are a few more things you may wish to ponder.

[**Exercise.** Is the product of *unitary* operators unitary in the product space?]

[**Exercise.** Is the product of *Hermitian* operators Hermitian in the product space?]

[**Exercise.** Is the product of *invertible* operators invertible in the product space?]

Chapter 11

Two Qubits and Binary Quantum Gates

$$\begin{aligned} |\psi\rangle^2 = & \alpha |0\rangle_A |0\rangle_B + \beta |0\rangle_A |1\rangle_B \\ & + \gamma |1\rangle_A |0\rangle_B + \delta |1\rangle_A |1\rangle_B \end{aligned}$$

11.1 The Jump from One to Two

Classical Binary Systems

Elevating a *classical one-bit* system to a *classical two-bit* system doesn't seem to require any fancy math. We simply slap together two single bits and start defining binary logic gates like AND and OR which take both bits as input. That works because classical bits cannot become *entangled* the way in which quantum bits can. Nevertheless, we could use tensor products to define the classical two-bit system if we were inclined to see the formal definition. As with the one-bit systems, it would be rather dull and its only purpose would be to allow a fair comparison between *two classical bits* and *two quantum bits* (the latter *requiring* the tensor product). I won't bother with the formal treatment of two classical bits, but leave that as an exercise after you have seen how we do it in the quantum case.

Quantum Binary Systems

Tensor algebra allows us to make the leap from the 2-D Hilbert space of *one qubit* to a 4-D Hilbert space of *two qubits*. Once mastered, advancing to n qubits for $n > 2$ is straightforward. Therefore, we move carefully through the $n = 2$ case, where the concepts needed for higher dimensions are easiest to grasp.

11.2 The State Space for Two Qubits

11.2.1 Definition of a Two Quantum Bit (“Bipartite”) System

The reason that we “go tensor” for a two-qubit system is that the two bits may become *entangled* (to be defined below). That forces us to treat two bits as if they were a single state of a larger state space rather than keep them separate.

Definition of Two Qubits. A *Two-qubit system* is (any copy of) the entire product space $\mathcal{H} \otimes \mathcal{H}$.

Definition of a Two-Qubit Value. The “*value*” or “*state*” of a two-qubit system is any unit (or normalized) vector in $\mathcal{H} \otimes \mathcal{H}$.

In other words, *two qubits* form a single entity – the tensor product space $\mathcal{H} \otimes \mathcal{H}$ – whose value can be any vector (which happens also to be a tensor) on the *projective sphere* of that product space.

The two-qubit entity itself is not committed to any particular value until we say which specific unit-vector in $\mathcal{H} \otimes \mathcal{H}$ we are assigning it.

Vocabulary and Notation

Two qubits are often referred to as a *bipartite* system. This term is inherited from physics in which a composite system of two identical particles (thus *bi-parti-te*) can be “entangled.”

To distinguish the two otherwise identical component Hilbert spaces I may use subscripts, A for the left-space and B for the right space,

$$\mathcal{H}_A \otimes \mathcal{H}_B.$$

Another notation you might see emphasizes the *order* of the tensor product, that is, the number of component spaces – in our current case, two,

$$\mathcal{H}_{(2)}.$$

In this lesson, we are concerned with *order-2* products, with a brief but important section on *order-3* products at the very end.

Finally, note that in the lesson on tensor products, we used the common abstract names V and W for our two component spaces. In quantum computation the component spaces are usually called A and B . For example, whereas in that lecture I talked about “ V -major ordering” for the tensor coordinates, I’ll now refer to “ A -major ordering”.

11.2.2 The Preferred Bipartite CBS

For a single qubit, \mathcal{H} , we established names for our two CBS kets: “ $|0\rangle$ ” and “ $|1\rangle$.” Let’s do the same thing now for the four induced basis kets of our product space, $\mathcal{H}_{(2)}$. These states correspond to the two-bits of classical computing, and they allow us to think of two ordinary bits as being embedded within the rich continuum of a quantum bipartite state space.

Symbols for Basis Vectors

The tensor product of two 2-D vector spaces has dimension $2 \times 2 = 4$. Its inherited preferred basis vectors are the separable products of the component space vectors,

$$\{ |0\rangle \otimes |0\rangle, |0\rangle \otimes |1\rangle, |1\rangle \otimes |0\rangle, |1\rangle \otimes |1\rangle \}.$$

These are the CBS of the bipartite system. There are some shorthand alternatives in quantum computing.

$$\begin{aligned} |0\rangle \otimes |0\rangle &\longleftrightarrow |0\rangle |0\rangle \longleftrightarrow |00\rangle \longleftrightarrow |0\rangle^2 \\ |0\rangle \otimes |1\rangle &\longleftrightarrow |0\rangle |1\rangle \longleftrightarrow |01\rangle \longleftrightarrow |1\rangle^2 \\ |1\rangle \otimes |0\rangle &\longleftrightarrow |1\rangle |0\rangle \longleftrightarrow |10\rangle \longleftrightarrow |2\rangle^2 \\ |1\rangle \otimes |1\rangle &\longleftrightarrow |1\rangle |1\rangle \longleftrightarrow |11\rangle \longleftrightarrow |3\rangle^2 \end{aligned}$$

All three of the alternatives that lack the \otimes symbol are seen frequently in computer science, and we will switch between them freely based on the emphasis that the context requires.

The notation of the first two columns admits the possibility of labeling each of the component kets with the \mathcal{H} from whence it came, A or B , as in

$$\begin{aligned} |0\rangle_A \otimes |0\rangle_B &\longleftrightarrow |0\rangle_A |0\rangle_B \\ |0\rangle_A \otimes |1\rangle_B &\longleftrightarrow |0\rangle_A |1\rangle_B \\ \text{etc.} \end{aligned}$$

I will often omit the subscripts A and B when the context is clear and include them when I want to emphasize which of the two component spaces the vectors comes from. The labels are always expendable since the A -space ket is the one on the *left* and the B -space ket is the one on the *right*. I will even include and/or omit them in the same string of equalities, since it may be clear in certain expressions, but less so in others:

$$\begin{aligned} U\left((\beta |0\rangle_A + \delta |1\rangle_A) |1\rangle_B \right) &= U(\beta |0\rangle |1\rangle + \delta |1\rangle |1\rangle) = \beta |0\rangle |0\rangle + \delta |0\rangle |1\rangle \\ &= |0\rangle_A (\beta |0\rangle_B + \delta |1\rangle_B) \end{aligned}$$

(This is an equation we will develop later in this lesson.)

The densest of the notations in the “ \longleftrightarrow ” stack a couple paragraphs back is the *encoded* version which expresses the ket as an integer from 0 to 3. We should reinforce this correspondence at the outset and add the *coordinate representation* of each basis ket under the implied *A-major ordering* of the vectors suggested by their presentation, above.

basis ket	$ 0\rangle 0\rangle$	$ 0\rangle 1\rangle$	$ 1\rangle 0\rangle$	$ 1\rangle 1\rangle$
encoded	$ 0\rangle^2$	$ 1\rangle^2$	$ 2\rangle^2$	$ 3\rangle^2$
coordinates	$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

This table introduces an exponent-like notation, $|x\rangle^2$, which is needed mainly in the encoded form, since an integer representation for a CBS does not disclose its tensor order (2 in this case) to the reader, while the other representations clearly imply that we are looking at two-qubits.

11.2.3 Separable Bipartite States

The CBS are four special *separable tensors*. There is an infinity of other separable tensors in $\mathcal{H} \otimes \mathcal{H}$ of the form

$$|\psi\rangle \otimes |\varphi\rangle \longleftrightarrow |\psi\rangle|\varphi\rangle .$$

Note that, unlike the CBS symbolism, there is no further alternative notation for a general separable tensor. In particular, $|\psi\varphi\rangle$ makes no sense.

11.2.4 Alternate Bipartite Bases

Inherited Second Order x -CBS

We can construct other computational bases like the inherited x -basis,

$$\{ |0\rangle_x |0\rangle_x , \ |0\rangle_x |1\rangle_x , \ |1\rangle_x |0\rangle_x , \ |1\rangle_x |1\rangle_x \} ,$$

or, using the common alternate notation the x -basis, as

$$\{ |+\rangle |+\rangle , \ |+\rangle |-\rangle , \ |-\rangle |+\rangle , \ |-\rangle |-\rangle \} .$$

You might see it condensed, as

$$\{ |++\rangle , \ |+-\rangle , \ |-+\rangle , \ |--\rangle \} ,$$

or *super condensed* (my own notation),

$$\{ |0\rangle_{\pm}^2, |1\rangle_{\pm}^2, |2\rangle_{\pm}^2, |3\rangle_{\pm}^2 \}.$$

In this last version, I'm using the subscript \pm to indicate “ x basis” and I'm encoding the ket labels into decimal integers, 0 through 3, for the four CBS states.

Inherited Second Order Mixed CBS

While rare, we could inherit from the z -basis for our A -space, and the x -basis for our B -space, to create the hybrid

$$\{ |0\rangle |0\rangle_x, |0\rangle |1\rangle_x, |1\rangle |0\rangle_x, |1\rangle |1\rangle_x \}.$$

[**Exercise.** How do we know that this is an orthonormal basis for the product space?]

Notation

Whenever we choose to label a state as a CBS of a non-standard basis, we leave no room to label the component spaces, $|0\rangle_A |1\rangle_B$, which then must be inferred from their position (A left and B right).

[**Exception.** If we use the alternate notation $|+\rangle = |0\rangle_x$ and $|-\rangle = |1\rangle_x$, then we once again have room for the labels A and B : $|+\rangle_A |+\rangle_B$, $|-\rangle_A |+\rangle_B$, etc.]

No matter what basis we use, if pressed to show *both* the individual basis of interest *and* label the component state spaces, we could express everything in the z -basis and thus avoid the subscript conflict, entirely, as with

$$|0\rangle |0\rangle_x = |0\rangle_A \left(\frac{|0\rangle_B + |1\rangle_B}{\sqrt{2}} \right).$$

The basis we use as input(s) to the quantum circuit – and along which we measure the output of the same circuit – is what we mean when we speak of *the* computational basis. By convention, this is the z -basis. It corresponds to the classical bits

$$|0\rangle |0\rangle \longleftrightarrow [00]$$

$$|0\rangle |1\rangle \longleftrightarrow [01]$$

$$|1\rangle |0\rangle \longleftrightarrow [10]$$

$$|1\rangle |1\rangle \longleftrightarrow [11]$$

Nevertheless, in principle there is nothing preventing us from having hardware that measures output along a different orthonormal basis.

[**Future Note.** In fact, in the later courses, CS 83B and 83C, we will be considering measurements which are not only made with reference to a different observable (than S_z) but are not even bases: they are neither orthonormal nor linearly independent. These go by the names *general measurement operators* or *positive operator valued measures*, and are the theoretical foundation for encryption and error correction.]

Example

We expand the separable state

$$|0\rangle_x |1\rangle_y$$

along the (usual) computational basis in $\mathcal{H} \otimes \mathcal{H}$.

$$\begin{aligned} |0\rangle_x |1\rangle_y &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - i|1\rangle}{\sqrt{2}} \right) \\ &= \frac{|00\rangle - i|01\rangle + |10\rangle - i|11\rangle}{2}. \end{aligned}$$

As a sanity check we compute the modulus-squared of the product directly, i.e., by summing the magnitude-squared of the four amplitudes,

$$\begin{aligned} \left\| |0\rangle_x |1\rangle_y \right\|^2 &= \left(\frac{1}{2} \right) \left(\frac{1}{2} \right) + \left(\frac{i}{2} \right) \left(\frac{-i}{2} \right) + \left(\frac{1}{2} \right) \left(\frac{1}{2} \right) + \left(\frac{i}{2} \right) \left(\frac{-i}{2} \right) \\ &= 1, \end{aligned}$$

as it darn well had *better* be. ✓

This is the kind of test you can perform in the midst of a large computation to be sure you haven't made an arithmetic error.

We might have computed the modulus-squared of the tensor product using one of two other techniques, and it won't hurt to confirm that we get the same 1 as an answer. The techniques are

- the definition of *inner product* in the tensor space (the product of *component inner-products*),

$$\begin{aligned} \left\| |0\rangle_x |1\rangle_y \right\|^2 &= \left\langle |0\rangle_x \otimes |1\rangle_y \mid |0\rangle_x \otimes |1\rangle_y \right\rangle \\ &= {}_x \langle 0 | 0 \rangle_x \cdot {}_y \langle 1 | 1 \rangle_y = 1 \cdot 1 = 1 \quad \checkmark, \end{aligned}$$

- or the *adjoint conversion rules* to form the *left bra* for the inner product,

$$\begin{aligned} \left\| |0\rangle_x |1\rangle_y \right\|^2 &= \left({}_y \langle 1 | {}_x \langle 0 | \right) \left(|0\rangle_x |1\rangle_y \right) = {}_y \langle 1 | \left({}_x \langle 0 | 0 \rangle_x \right) |1\rangle_y \\ &= {}_y \langle 1 | 1 \rangle_y = 1 \quad \checkmark. \end{aligned}$$

However, neither of these would have been as thorough a check of our arithmetic as the first approach.

[Exercise. Expand the separable state

$$\left(\sqrt{.1} |0\rangle + i\sqrt{.9} |1\rangle \right) \otimes \left(i\sqrt{.7} |0\rangle + \sqrt{.3} |1\rangle \right)$$

along the computational basis in $\mathcal{H} \otimes \mathcal{H}$. Confirm that it is a unit tensor.]

How the Second Order x -Basis Looks when Expressed in the Natural Basis

If we combine the separable expression of the x -CBS kets,

$$|+\rangle|+\rangle, \quad |+\rangle|-\rangle, \quad |-\rangle|+\rangle, \quad \text{and} \quad |-\rangle|-\rangle.$$

with the expansion of each component ket along the natural basis,

$$\begin{aligned} |+\rangle &= H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{and} \\ |-\rangle &= H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}, \end{aligned}$$

the four x -kets look like this, when expanded along the natural basis:

$$\begin{aligned} |+\rangle|+\rangle &= \frac{|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle}{2}, \\ |+\rangle|-\rangle &= \frac{|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle}{2}, \\ |-\rangle|+\rangle &= \frac{|0\rangle|0\rangle + |0\rangle|1\rangle - |1\rangle|0\rangle - |1\rangle|1\rangle}{2}, \\ |-\rangle|-\rangle &= \frac{|0\rangle|0\rangle - |0\rangle|1\rangle - |1\rangle|0\rangle + |1\rangle|1\rangle}{2}. \end{aligned}$$

Notice something here that we will use in a future lecture:

*When expanded along the z -basis, the x -basis kets have **equal numbers of + and – terms** except for the zeroth CBS ket, $|00\rangle_x$, whose coordinates are all +1.*

I know it sounds silly when you say it out loud, but believe me, it will be very useful.

11.2.5 Non-Separable Bipartite Tensors

The typical tensor in a two-qubit system is a normalized finite sum of separable tensors – that’s how we started defining product tensors. While such a tensor is not necessarily separable, it can at least be expressed as a *superposition* of the four CBS kets,

$$|\psi\rangle^2 = \alpha|0\rangle|0\rangle + \beta|0\rangle|1\rangle + \gamma|1\rangle|0\rangle + \delta|1\rangle|1\rangle.$$

The “exponent 2” on the LHS is, as mentioned earlier, a clue to the reader that $|\psi\rangle$ lives in a *second-order* tensor product space, a detail that might not be clear without looking at the RHS. In particular, nothing is being “squared.”

11.2.6 Usual Definition of Two Qubits and their Values

This brings us to the common definition of a two-qubit system, which avoids the above formalism.

Alternative Definition of Two Qubits. *“Two qubits” are represented by a variable superposition of the four tensor basis vectors of $\mathcal{H} \otimes \mathcal{H}$,*

$$|\psi\rangle^2 = \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \gamma |1\rangle |0\rangle + \delta |1\rangle |1\rangle$$

where the complex scalars satisfy

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1.$$

Using our alternate notation, we can write this superposition either as

$$|\psi\rangle^2 = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle$$

or

$$|\psi\rangle^2 = \alpha |0\rangle^2 + \beta |1\rangle^2 + \gamma |2\rangle^2 + \delta |3\rangle^2.$$

We may also use alternate notation for scalars, especially when we prepare for higher-order product spaces:

$$|\psi\rangle^2 = c_{00} |00\rangle + c_{01} |01\rangle + c_{10} |10\rangle + c_{11} |11\rangle$$

or

$$|\psi\rangle^2 = c_0 |0\rangle^2 + c_1 |1\rangle^2 + c_2 |2\rangle^2 + c_3 |3\rangle^2.$$

11.3 Fundamental Two-Qubit Logic Gates

11.3.1 Binary Quantum Operators

Definition and Terminology

*A **binary quantum operator** is a unitary transformation, U , on the two-qubit system $\mathcal{H} \otimes \mathcal{H}$.*

As you can see, with all the vocabulary and skills we have mastered, definitions can be very short now and still have significant content. For instance, we already know that some binary quantum operators will be separable and others will not. The simplest and most common gate, in fact, is not separable.

Binary quantum operators also go by the names *two-qubit gates*, *binary qubit operators*, *bipartite operators* and various combinations of these.

Complete Description of Binary Quantum Operators

When we study specific binary qubit gates, we will do five things:

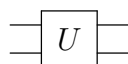
- \circlearrowleft : Show the *symbol* for that gate,
- $|x\rangle|y\rangle$: define the gate on the *computation basis states*,
- (\dots) : construct the *matrix* for the gate,
- $|\psi\rangle^2$: examine the behavior of the gate on an *general state* (i.e., one that is not necessarily separable), and
- \searrow : discuss the *measurement* probabilities of the output registers.

11.3.2 General Learning Example

We'll go through the checklist on a not-particularly-practical gate, but one that has the generality needed to cover future gates.

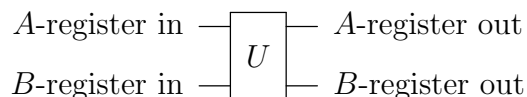
\circlearrowleft : The Symbol

Every binary qubit gate has two input lines, one for each input qubit, and two output lines, one for each output qubit. The label for the unitary transformation associated with the gate, say U , is placed inside a box connected to its inputs and outputs.



Although the data going into the two input lines can become "entangled" inside the gate, we consider the top half of the gate to be a separate *register* from the lower half. This can be confusing to new students, as we can't usually consider each output line to be independent of its partner the way the picture suggests. More (a lot more) about this shortly.

Vocabulary. The top input/output lines form an *upper A register* (or *A channel*) while the bottom form a *lower B register* (or *B channel*).



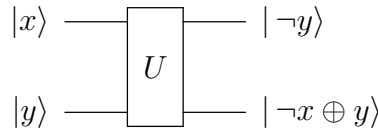
The labels A and B are usually implied, not written.

$|x\rangle |y\rangle$: Action on the CBS

Every operator is defined by its action on the basis, and in our case that's the computational basis. For binary gates, the symbolism for the general CBS is

$$\begin{aligned} |x\rangle \otimes |y\rangle &\cong |x\rangle |y\rangle, \\ \text{where } x, y &\in \{0, 1\}. \end{aligned}$$

To demonstrate this on our “learning” gate, U , we define its action on the CBS, which in turn defines the gate:



It is very important to treat the LHS as a single two-qubit input state, not two separate single qubits, and likewise with the output. In other words, it is really saying

$$\begin{aligned} U(|x\rangle \otimes |y\rangle) &\equiv |\neg y\rangle \otimes |\neg x \oplus y\rangle \\ \text{or, using shorter notation,} \\ U(|x\rangle |y\rangle) &\equiv |\neg y\rangle |\neg x \oplus y\rangle \end{aligned}$$

Furthermore, $|x\rangle |y\rangle$ only represents the four CBS, so we have to extend this linearly to the entire Hilbert space.

Let's make this concrete. Taking one of the four CBS, say $|10\rangle$, the above definition tells us to substitute $1 \rightarrow x$ and $0 \rightarrow y$, to get the gate's output,

$$U(|1\rangle |0\rangle) = |\neg 0\rangle |\neg 1 \oplus 0\rangle = |1\rangle |0\rangle.$$

[Exercise. Compute the effect of U on the other three CBS.]

(\dots) : The Matrix

In our linear transformation lesson, we proved that the matrix M_T that represents an operator T can be written by applying T to each basis vector, \mathbf{a}_k , and placing the answer vectors in the columns of the matrix,

$$M_T = \begin{pmatrix} T(\mathbf{a}_1), & T(\mathbf{a}_2), & \dots, & T(\mathbf{a}_n) \end{pmatrix}.$$

$T(\mathbf{v})$ is then just $M_T \cdot \mathbf{v}$. Applying the technique to U and the CBS $\{|x\rangle |y\rangle\}$ we get

$$M_U = \begin{pmatrix} U|00\rangle, & U|01\rangle, & U|10\rangle, & U|11\rangle \end{pmatrix}.$$

Each of these columns must be turned into the *coordinate representation* – in the inherited tensor basis – of the four U -values. Let's compute them. (Spoiler alert: this was the last exercise):

$$U|00\rangle = |\neg 0\rangle |\neg 0 \oplus 0\rangle = |1\rangle |1 \oplus 0\rangle = |1\rangle |1\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Similarly, we get

$$U|01\rangle = |\neg 1\rangle |\neg 0 \oplus 1\rangle = |0\rangle |1 \oplus 1\rangle = |0\rangle |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$U|10\rangle = |\neg 0\rangle |\neg 1 \oplus 0\rangle = |1\rangle |0 \oplus 0\rangle = |1\rangle |0\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix},$$

$$U|11\rangle = |\neg 1\rangle |\neg 1 \oplus 1\rangle = |0\rangle |0 \oplus 1\rangle = |0\rangle |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

giving us the matrix

$$U \cong M_U = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

which is, indeed, unitary. Incidentally, not every recipe you might conjure for the four values $U(|x\rangle|y\rangle)$ will produce a unitary matrix and therefore not yield a reversible – and thus valid – quantum gate. (We learned last time that non-unitary matrices do not keep state vectors on the projective sphere and therefore do not correspond to physically sensible quantum operations.)

[Exercise.] Go through the same steps on the putative operator defined by

$$U(|x\rangle|y\rangle) \equiv |x \oplus \neg y\rangle |\neg x \oplus y\rangle.$$

Is this matrix unitary? Does U constitute a realizable quantum gate?

$|\psi\rangle^2$: Behavior on General State

The general state is a superposition of the four basis kets,

$$|\psi\rangle^2 = \alpha|0\rangle|0\rangle + \beta|0\rangle|1\rangle + \gamma|1\rangle|0\rangle + \delta|1\rangle|1\rangle,$$

whose coordinates are

$$|\psi\rangle^2 = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix},$$

so matrix multiplication gives

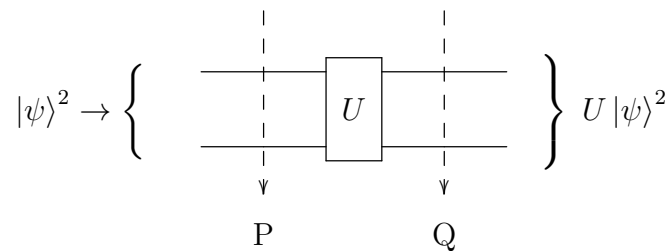
$$\begin{aligned} U|\psi\rangle^2 &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \beta \\ \delta \\ \gamma \\ \alpha \end{pmatrix} \\ &= \beta|0\rangle|0\rangle + \delta|0\rangle|1\rangle + \gamma|1\rangle|0\rangle + \alpha|1\rangle|1\rangle. \end{aligned}$$

Evidently, U leaves the amplitude of the CBS ket, $|1\rangle|0\rangle$, alone and permutes the other three amplitudes.

↘ : Measurement

[**Note:** Everything in this section, and in general when we speak of measurement, assumes that we are measuring the states relative to the *natural computational basis*, the z -basis, unless explicitly stated otherwise. This is $\{|0\rangle, |1\rangle\}$ for a single register and $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ for both registers. We can measure relative to other CBSs, but then some of the results below cannot be applied exactly as stated. Of course, I would not finish the day without giving you an example.]

Now that we know what U does to input states we can make some basic observations about how it changes the measurement probabilities. Consider a state's amplitudes both before and after the application of the gate (access points P and Q, respectively):



Easy Observations by Looking at Expansion Coefficients. Trait #6 (QM's *fourth postulate*) tells us that a measurement of the input state (point P)

$$|\psi\rangle^2 = \alpha|0\rangle|0\rangle + \beta|0\rangle|1\rangle + \gamma|1\rangle|0\rangle + \delta|1\rangle|1\rangle,$$

collapses it to $|00\rangle$ with probability $|\alpha|^2$. Meanwhile, a look at the $U|\psi\rangle^2$'s amplitudes (point Q),

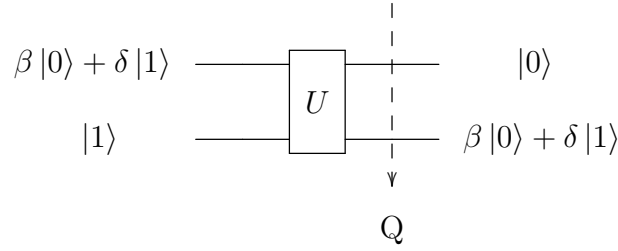
$$U|\psi\rangle^2 = \beta|0\rangle|0\rangle + \delta|0\rangle|1\rangle + \gamma|1\rangle|0\rangle + \alpha|1\rangle|1\rangle,$$

reveals that measuring the output there will land it on the state $|00\rangle$ with probability $|\beta|^2$. This *was* the input's probability of landing on $|01\rangle$ *prior* to the gate; U *has shifted the probability that a ket will register a "01" on our meter to the probability that it will register a "00."* In contrast, a glance at $|\psi\rangle$'s pre- and post- U amplitudes of the CBS $|10\rangle$ tells us that the probability of *this* state being measured after U is the same as before: $|\gamma|^2$.

Measurement of Separable Output States. By looking at the expansion coefficients of the general output state, we can usually concoct a simple input state that produces a separable output. For example, taking $\gamma = \alpha = 0$ gives a separable input, $(\beta|0\rangle_A + \delta|1\rangle_A) \otimes |1\rangle_B$, as well as the following separable output:

$$\begin{aligned} U\left((\beta|0\rangle_A + \delta|1\rangle_A)|1\rangle_B\right) &= U(\beta|0\rangle|1\rangle + \delta|1\rangle|1\rangle) = \beta|0\rangle|0\rangle + \delta|0\rangle|1\rangle \\ &= |0\rangle_A(\beta|0\rangle_B + \delta|1\rangle_B) \end{aligned}$$

We consider a post-gate measurement at access point Q:



Measuring the A -register at the output (point Q) will yield a “0” with *certainty* (the coefficient of the separable CBS component, $|0\rangle$, is 1) yet will tell us *nothing* about the B -register, which has a $|\beta|^2$ probability of yielding a “0” and $|\delta|^2$ chance of yielding a “1”, just as it did before we measured A . Similarly, measuring B at point Q will collapse that output register into one of the two B -space CBS states (with the probabilities $|\beta|^2$ and $|\delta|^2$) but will not change a subsequent measurement of the A -register output, still certain to show us a “0”.

(If this seems as though I’m jumping to conclusions, it will be explained formally when we get the *Born rule*, below.)

A slightly less trivial separable output state results from the input,

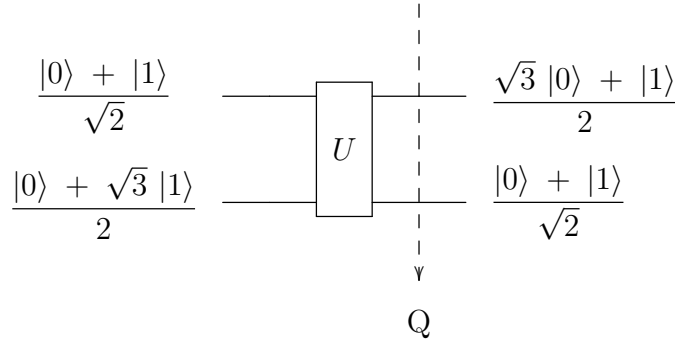
$$\begin{aligned} |\psi\rangle^2 &= \left(\frac{\sqrt{2}}{4}\right)|00\rangle + \left(\frac{\sqrt{6}}{4}\right)|01\rangle + \left(\frac{\sqrt{2}}{4}\right)|10\rangle + \left(\frac{\sqrt{6}}{4}\right)|11\rangle \\ &= \left(\frac{|0\rangle_A + |1\rangle_A}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle_B + \sqrt{3}|1\rangle_B}{2}\right). \end{aligned}$$

(As it happens, this input state is separable, but that’s not required to produce a separable output state, the topic of this example. I just made it so to add a little symmetry.)

The output state can be written down instantly by permuting the amplitudes according to U 's formula,

$$\begin{aligned} U|\psi\rangle^2 &= \left(\frac{\sqrt{6}}{4}\right)|00\rangle + \left(\frac{\sqrt{6}}{4}\right)|01\rangle + \left(\frac{\sqrt{2}}{4}\right)|10\rangle + \left(\frac{\sqrt{2}}{4}\right)|11\rangle \\ &= \left(\frac{\sqrt{3}|0\rangle_A + |1\rangle_A}{2}\right) \otimes \left(\frac{|0\rangle_B + |1\rangle_B}{\sqrt{2}}\right), \end{aligned}$$

and I have factored it for you, demonstrating the output state's separability. Measuring either output register at access point Q,



has a non-zero probability of yielding one of the two CBS states for its respective \mathcal{H} , but it won't affect the measurement of the *other* output register. For example, measuring the *B-qubit-out* will land it in $|0\rangle_B$ or $|1\rangle_B$ with equal probability. Regardless of which result we get, it will not affect a future measurement of the *A-qubit-out* which has a 3/4 chance of measuring “0” and 1/4 chance of showing us a “1.”

Measuring One Register of a Separable State. This is characteristic of separable states, whether they be input or output. *Measuring either register does not affect the probabilities of the other register.* It only collapses the component vector of the tensor, leaving the other vector un-collapsed.

11.3.3 Quantum Entanglement

The Measurement of Non-Separable Output States

As a final example before we get into real gates, we look at what happens when we try to measure a non-separable output state of our *general learning circuit* just presented. Consider the input,

$$\begin{aligned} |\psi\rangle^2 &= \left(\frac{1}{\sqrt{2}}\right)|00\rangle + \left(\frac{1}{\sqrt{2}}\right)|01\rangle \\ &= |0\rangle_A \left(\frac{|0\rangle_B + |1\rangle_B}{\sqrt{2}}\right), \end{aligned}$$

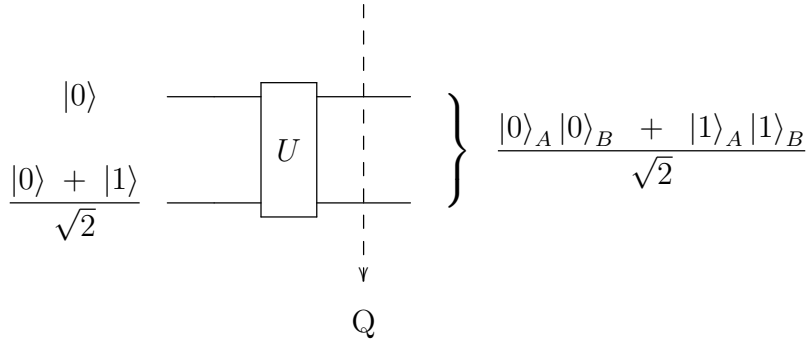
a separable state that we also know under the alias,

$$|0\rangle |0\rangle_x.$$

Applying U , this time using matrix notation (for fun and practice), yields

$$\begin{aligned} U|\psi\rangle^2 &= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \\ &= \frac{|0\rangle_A |0\rangle_B + |1\rangle_A |1\rangle_B}{\sqrt{2}}, \end{aligned}$$

clearly *not* factorable. Furthermore, unlike the separable output states we have studied, a measurement of either register forces its partner to collapse.



For example, if we measure B 's output, and find it to be in state $|1\rangle_B$, since the output ket has only one CBS tensor associated with that $|1\rangle_B$, namely $|1\rangle_A |1\rangle_B$, as we can see from its form

$$\frac{|0\rangle |0\rangle + |1\rangle |1\rangle}{\sqrt{2}},$$

we are forced to conclude that the A -register must have collapsed into its $|1\rangle$ state. If this is not clear to you, imagine that the A -register had *not* collapsed to $|1\rangle$. It would then be possible to measure a “0” in the A -register. However, such a turn of events would have landed a $|1\rangle$ in the B -register and $|0\rangle$ in the A -register, a combination that is patently absent from the output ket's CBS expansion, above.

Stated another way (if you are still unsure), there is only *one bipartite state* here, and if, when expanded along the CBS basis, one of the four CBS kets is missing from that expansion that CBS ket has a zero probability of being the result of a measurement collapse. Since $|0\rangle |1\rangle$ is not in the expansion, this state is not accessible through a measurement. (And by the way, the same goes for $|1\rangle |0\rangle$.)

Definition of Quantum Entanglement

An *entangled state* in a product space is one that is *not separable*.

Non-Locality

Entangled states are also said to be *non-local*, meaning that if you are in a room with only one of the two registers, you do not have full control over what happens to the data there; an observer of the other register in a different room may measure his qubit and affect your data even though you have done nothing. Furthermore, if you measure the data in that register, your efforts are not confined to your room but extend to the outside world where the other register is located. Likewise, *separable states* are considered *local*, since they *do* allow full segregation of the actions on separate registers. Each observer has total control of the destiny of his register, and his actions don't affect the other observer.

The Entanglement Connection

Non-separable states are composed of entangled constituents. While each constituent may be *physically* separated in space and time from its partner in the other register, the two parts do *not* have independent world lines. Whatever happens to one affects the other.

This is the single most important and widely used phenomenon in quantum computing, so be sure to digest it well.

Partial Collapse

In this last example a measurement and collapse of one register completely determined the full and unique state of the output. However, often things are subtler. Measuring one register may have the effect of only *partially* collapsing its partner. We'll get to that when we take up the *Born rule*.

Measurements Using a Different CBS

Everything we've done in the last two sections is true as long as we have a consistent CBS from start to finish. The definition of U , its matrix and the measurements have all used the same CBS. But funny things happen if we use a different measurement basis than the one used to define the operator or express its matrix. Look for an example in a few minutes.

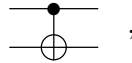
Now let's do everything again, this time for a *famous* gate.

11.3.4 The Controlled-NOT (CNOT) Gate

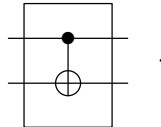
This is the simplest and most commonly used binary gate in quantum computing.

• : The Symbol

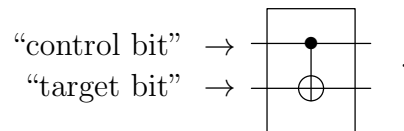
The gate is usually drawn without an enclosing box as in



but I sometimes box it,



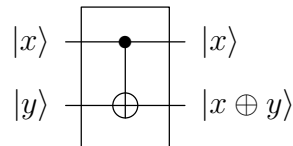
The A -register is often called the *control bit*, and the B -register the *target bit*,



We'll explain that terminology in the next bullet.

$|x\rangle|y\rangle$: Action on the CBS

The CNOT gate has the following effect on the computational basis states:



When viewed on the tiny set of four CBS tensors, it appears to leave the A -register unchanged and to negate the B -register qubit or leave it alone, based on whether the A -register is $|1\rangle$ or $|0\rangle$:

$$|y\rangle \mapsto \begin{cases} |y\rangle, & \text{if } x = 0 \\ |\neg y\rangle, & \text{if } x = 1 \end{cases}$$

Because of this, the gate is described as a *controlled-NOT operator*. The A -register is called the *control bit* or *control register*, and the B -register is the *target bit* or *target register*. We cannot use this simplistic description on a general state, however, as the next sections will demonstrate.

(...) : The Matrix

We compute the column vectors of the matrix by applying CNOT to the CBS tensors to get

$$\begin{aligned} M_{\text{CNOT}} &= \left(\text{CNOT} |00\rangle, \text{CNOT} |01\rangle, \text{CNOT} |10\rangle, \text{CNOT} |11\rangle \right) \\ &= \left(|00\rangle, |01\rangle, |11\rangle, |10\rangle \right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{aligned}$$

which, as always, we must recognize as unitary.

[Exercise.] Prove that this is not a separable operator. **Hint:** What did we say about the form of a separable operator's matrix?

$|\psi\rangle^2$: Behavior on General State

Applying CNOT to the general state,

$$|\psi\rangle^2 = \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \gamma |1\rangle |0\rangle + \delta |1\rangle |1\rangle,$$

we get

$$\begin{aligned} \text{CNOT} |\psi\rangle^2 &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \delta \\ \gamma \end{pmatrix} \\ &= \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \delta |1\rangle |0\rangle + \gamma |1\rangle |1\rangle. \end{aligned}$$

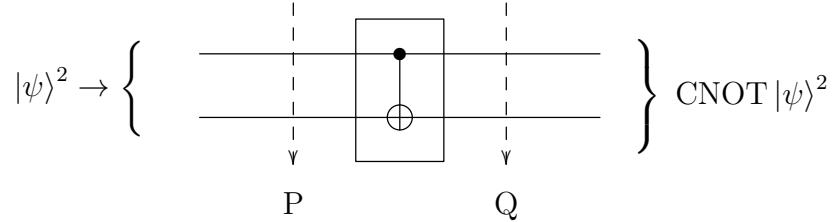
A Meditation. If you are tempted to read on, feeling that you understand everything we just covered, see how quickly you can answer this:

[Exercise.] The CNOT is said to leave the source register unchanged and flip the target register *only if* the source register input is $|1\rangle$. Yet the matrix for CNOT seems to *always* swap the last two amplitudes, $\gamma \leftrightarrow \delta$, of *any* ket. Explain this.]

Caution. If you cannot do the last exercise, you should not continue reading, but review the last few sections or ask a colleague for assistance until you see the light. This is an important consequence of what we just covered. It is best that you apply that knowledge to solve it rather than my blurting out the answer for you.

↘ : Measurement

First, we'll consider the amplitudes before and after the application of CNOT (access points P and Q, respectively):



A measurement of the *input* state (point P),

$$|\psi\rangle^2 = \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \gamma |1\rangle |0\rangle + \delta |1\rangle |1\rangle ,$$

will yield a “00” with probability $|\alpha|^2$ and “01” with probability $|\beta|^2$. A *post-gate* measurement of $CNOT |\psi\rangle^2$ (point Q),

$$CNOT |\psi\rangle^2 = \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \delta |1\rangle |0\rangle + \gamma |1\rangle |1\rangle ,$$

will yield those first two readings with the same probabilities since their ket's respective amplitudes are not changed by the gate. However, the probabilities of getting a “10” vs. a “11” reading are swapped. They go from $|\gamma|^2$ and $|\delta|^2$ before the gate to $|\delta|^2$ and $|\gamma|^2$, after.

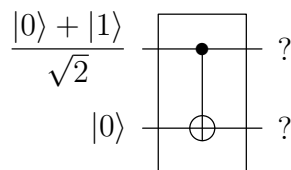
Separable Output States. There's nothing new to say here, as we have covered all such states in our learning example. Whenever we have a separable output state, measuring one register has no affect on the other register. So while a measurement of *A* causes it to collapse, *B* will continue to be in a superposition state until we measure it (and vice versa).

Quantum Entanglement for CNOT

A separable bipartite state *into* CNOT gate does not usually result in a separable state *out of* CNOT. To see this, consider the separable state

$$|\psi\rangle^2 = |0\rangle_x \otimes |0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) |0\rangle$$

going into CNOT:



When presented with a superposition state into either the A or B register, back away *very slowly* from your circuit diagram. Turn, instead, to the linear algebra, which never lies. The separable state should be resolved to its *tensor basis form* by distributing the product over the sums,

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) |0\rangle = \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |10\rangle .$$

[**Exception:** If you have a *separable operator* as well as *separable input state*, we don't need to expand the input state along the CBS, as the definition of separable operator allows us to apply the component operators individually to the component vectors. CNOT is not separable, so we have to expand.]

Now, apply CNOT using linearity,

$$\begin{aligned} \text{CNOT} \left(\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |10\rangle \right) &= \frac{1}{\sqrt{2}} \text{CNOT}(|00\rangle) + \frac{1}{\sqrt{2}} \text{CNOT}(|10\rangle) \\ &= \frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle \\ &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} . \end{aligned}$$

This is the *true output* of the gate for the presented input. It is not separable as is obvious by its simplicity; there are only two ways we might factor it: pulling out an A -ket (a vector in the first \mathcal{H} space) or pulling out a B -ket (a vector in the second \mathcal{H} space), and neither works.

(Repeat of) Definition. An *entangled state* in a product space is one that is *not separable*.

Getting back to the circuit diagram, we see there is nothing whatsoever we can place in the question marks that would make that circuit sensible. Anything we might try would make it appear as though we had a separable product on the RHS, which we do not. The best we can do is consolidate the RHS of the gate into a *single bipartite qubit*, indeed, an *entangled state*.

$$\left. \begin{array}{c} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ |0\rangle \end{array} \right\} \left[\begin{array}{c} \bullet \\ | \\ \oplus \end{array} \right] \rightarrow \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

With an entangled output state such as this, measuring one output register causes the collapse of *both* registers. We use this property frequently when designing quantum algorithms.

Individual Measurement of Output Registers. Although we may have an entangled state at the output of a gate, we are always allowed to measure each

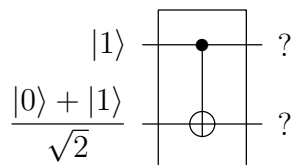
register separately. No one can stop us from doing so; the two registers are distinct physical entities at separate locations in the computer (or universe). Entanglement and non-locality mean that the registers are connected to one another. Our intent to measure one register must be accompanied by the awareness that, when dealing with an entangled state, doing so will affect the other register's data.

When CNOT's *Control Register* gets a CBS ...

Now, consider the separable state

$$|\psi\rangle^2 = |1\rangle \otimes |0\rangle_x = |1\rangle \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right)$$

going into CNOT:



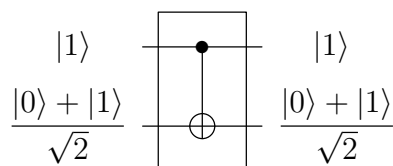
I have chosen the *A*-register input to be $|1\rangle$ for variety. It could have been $|0\rangle$ with the same (as of yet, undisclosed) outcome. The point is that this time our *A*-register is a CBS while the *B*-register is a superposition. We know from experience to ignore the circuit diagram and turn to the linear algebra.

$$|1\rangle \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) = \frac{1}{\sqrt{2}} |10\rangle + \frac{1}{\sqrt{2}} |11\rangle ,$$

and we apply CNOT

$$\begin{aligned} \text{CNOT} \left(\frac{1}{\sqrt{2}} |10\rangle + \frac{1}{\sqrt{2}} |11\rangle \right) &= \frac{1}{\sqrt{2}} \text{CNOT}(|10\rangle) + \frac{1}{\sqrt{2}} \text{CNOT}(|11\rangle) \\ &= \frac{1}{\sqrt{2}} |11\rangle + \frac{1}{\sqrt{2}} |10\rangle \\ &= |1\rangle \left(\frac{|1\rangle + |0\rangle}{\sqrt{2}} \right) . \end{aligned}$$

Aha – *separable*. That's because the *control-bit* (the *A*-register) is a CBS; it does not change during the linear application of CNOT so will be conveniently available for factoring at the end. Therefore, for this input, we are authorized to label the output registers, individually.



The two-qubit output state is unchanged. Not so fast. You have to do an ...

[**Exercise.** We are told that a $|1\rangle$ going into CNOT's *control register* means we flip the B -register bit. Yet, the output state of this binary gate is the same as the input state. Explain. **Hint:** Try the same example with a B -register input of $\sqrt{.3} |0\rangle + \sqrt{.7} |1\rangle$.]

[**Exercise.** Compute CNOT of an input tensor $|1\rangle |1\rangle_x$. Does CNOT leave this state unchanged?]

Summary. A CBS ket going into the *control register* (A) of a CNOT gate allows us to preserve the two registers at the output: we do, indeed, get a separable state out, with the control register output identical to the control register input. This is true even if a superposition goes into the *target register* (B). If a *superposition* goes into the *control* register, however, all bets are off (i.e., entanglement emerges at the output).

What the Phrase “The A -Register is Unchanged” *Really* Means

The A , or *control*, register of the CNOT gate is said to be unaffected by the CNOT gate, although this is overstating the case; it gives the false impression that a separable bipartite state *into* CNOT results in a separable state *out*, which we see is not the case. Yet, there are at least two ways to interpret this characterization.

1. When a CBS state (of the preferred, z -basis) is presented to CNOT's A -register, the output state is, indeed, separable, with the A -register unchanged.
2. If a non-trivial superposition state is presented to CNOT's A -register, the measurement probabilities (relative to the z -basis) of the A -register output are preserved.

We have already demonstrated **item 1**, so let's look now at **item 2**. The general state, expressed along the natural basis is

$$|\psi\rangle^2 = \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \gamma |1\rangle |0\rangle + \delta |1\rangle |1\rangle ,$$

and CNOTing this state gives

$$\text{CNOT} |\psi\rangle^2 = \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \delta |1\rangle |0\rangle + \gamma |1\rangle |1\rangle .$$

If we were to measure this state, we know that it would collapse to a CBS in $\mathcal{H} \otimes \mathcal{H}$. With what probability would we find that the first CBS (register- A) in this state had collapsed to $|0\rangle$? $|1\rangle$?

We have not had our formal lesson on probability yet, but you can no doubt appreciate that, of the four possible CBS outcomes, two of them find the A -reg collapsing to $|0\rangle_A$ (even though they are *independent* events because the B -register of

one outcome is $|0\rangle_B$ and of the other outcome is $|1\rangle_B$). Therefore, to get the overall probability that A collapses to $|0\rangle_A$, we simply add those two probabilities:

$$\begin{aligned} P(A\text{-reg output} \searrow |0\rangle) &= P\left(\text{CNOT}|\psi\rangle \searrow |00\rangle\right) + P\left(\text{CNOT}|\psi\rangle \searrow |01\rangle\right) \\ &= |\alpha|^2 + |\beta|^2, \end{aligned}$$

which is exactly the same probability of measuring a 0 on the input, $|\psi\rangle$, prior to applying CNOT.

[**Exercise.** We did not compute the probability of measuring a “0” on the input, $|\psi\rangle$. Do that to confirm the claim.]

[**Exercise.** What **trait** of QM (and *postulate*) tells us that the individual probabilities are $|\alpha|^2$ and $|\beta|^2$?]

[**Exercise.** Compute the probabilities of measuring a “1” in the A -register both before and after the CNOT gate. **Caution:** This doesn’t mean that we would measure the *same* prepared state before and after CNOT. Due to the collapse of the state after any measurement, we must prepare many identical states and measure *some* before and *others* after then examine the outcome frequencies to see how the experimental probabilities compare.]

Measuring Using a Different CBS

Now we come to the example that I promised: measuring in a basis different from the one used to define and express the matrix for the gate. Let’s present the following four bipartite states

$$|00\rangle_x, |01\rangle_x, |10\rangle_x, \text{ and } |11\rangle_x$$

to the input of CNOT and look at the output (do a measurement) in terms of the x -basis (which consist of those four tensors).

$|00\rangle_x$: This one is easy because the z -coordinates are all the same.

$$|00\rangle_x = |0\rangle_x |0\rangle_x = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

CNOT applied to $|00\rangle_x$ swaps the last two z -coordinates, which are identical, so it is unchanged.

$$\text{CNOT} |00\rangle_x = |00\rangle_x$$

(Use any method you learned to confirm this rigorously.)

$|01\rangle_x$: Just repeat, but watch for signs.

$$|01\rangle_x = |0\rangle_x |1\rangle_x = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

Let's use the matrix to get a quick result.

$$\text{CNOT } |01\rangle_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}$$

From here it's easier to expand along the z -basis so we can factor,

$$\begin{aligned} \text{CNOT } |01\rangle_x &= \frac{1}{2} \left(|0\rangle |0\rangle - |0\rangle |1\rangle - |1\rangle |0\rangle + |1\rangle |1\rangle \right) \\ &= \frac{1}{2} \left(|0\rangle |0\rangle - |1\rangle |0\rangle - |0\rangle |1\rangle + |1\rangle |1\rangle \right) \\ &= \frac{1}{2} \left((|0\rangle - |1\rangle) |0\rangle - (|0\rangle - |1\rangle) |1\rangle \right) \\ &= \frac{1}{2} \left((|0\rangle - |1\rangle) (|0\rangle - |1\rangle) \right) \\ &= \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = |1\rangle_x |1\rangle_x = |11\rangle_x. \end{aligned}$$

What is this? Looking at it in terms of the x -basis it left the B -register unchanged at $|1\rangle_x$ but flipped the A -register from $|0\rangle_x$ to $|1\rangle_x$.

Looking back at the $|00\rangle_x$ case, we see that when the B -register held a qubit in the $S_x = "0"$ state, the A -register was unaffected relative to the x -basis.

This looks suspiciously as though the B -register is now the *control* bit and A is the *target* bit and, in fact, a computation of the remaining two cases, $|10\rangle_x$ and $|11\rangle_x$, would bear this out.

$$|10\rangle_x : \text{CNOT } |10\rangle_x = |10\rangle_x \text{ [Exercise.]}$$

$$|11\rangle_x : \text{CNOT } |11\rangle_x = |01\rangle_x \text{ [Exercise.]}$$

Summarizing, we see that

$$\begin{aligned} \text{CNOT } |00\rangle_x &= |00\rangle_x \\ \text{CNOT } |01\rangle_x &= |11\rangle_x \\ \text{CNOT } |10\rangle_x &= |10\rangle_x \\ \text{CNOT } |11\rangle_x &= |01\rangle_x, \end{aligned}$$

demonstrating that, in the x -basis, the B -register is the *control* and the A is the *target*.

[**Preview.** We are going to revisit this in a circuit later today. For now, we'll call it an “upside-down” action of the CNOT gate relative to the x -basis and later see how to turn it into an actual “upside-down” CNOT gate for the natural CBS kets. When we do that, we'll call it “C \uparrow NOT,” because it will be controlled from bottom-up in the z -basis. So for, however, we've only produced this bottom-up behavior in the x -basis, so the gate name does not change.]

What About Measurement? I advertised this section as a study of measurement, yet all I did so far was make observations about the separable components of the output – which was an eye-opener in itself. Still, let's bring it back to the topic of measurement.

Take any of the input states, say $|10\rangle_x$. Then the above results say that

$$\text{CNOT} |10\rangle_x = |10\rangle_x .$$

To turn this into a statement about measurement probabilities, we “dot” the output state with the x -basis kets to get the four amplitudes. By orthogonality of CBS,

$$\begin{aligned} {}_x\langle 00 | 10 \rangle_x &= 0 \\ {}_x\langle 01 | 10 \rangle_x &= 0 \\ {}_x\langle 10 | 10 \rangle_x &= 1 \\ {}_x\langle 11 | 10 \rangle_x &= 0 , \end{aligned}$$

producing the measurement probability of 100% for the state, $|10\rangle_x$. In other words, for this state – which has a B input of 0 (in x -coordinates) – its output remains 0 with certainty, while A 's 1 (again, x -coordinates) is unchanged, also with certainty. On the other hand, the input state $|11\rangle_x$ gave us an output of $|01\rangle_x$, so the output amplitudes become

$$\begin{aligned} {}_x\langle 00 | 01 \rangle_x &= 0 \\ {}_x\langle 01 | 01 \rangle_x &= 1 \\ {}_x\langle 10 | 01 \rangle_x &= 0 \\ {}_x\langle 11 | 01 \rangle_x &= 0 . \end{aligned}$$

Here the input – whose B x -basis component is 1 – turns into an output with the B x -basis component remaining 1 (with certainty) and an A x -basis input 1 becoming flipped to 0 at the output (also with certainty).

This demonstrates that such statements like “The A -register is left unchanged” or “The A register is the control qubit,” are loosey-goosey terms that must be taken with a grain of salt. They are vague for non-separable states (as we saw, earlier) and patently false for measurements in alternate CBSs.

11.3.5 The Second-Order Hadamard Gate

We construct the second order Hadamard gate by forming the *tensor product* of two first order gates, so we'd better first review that operator.

Review of the First Order Hadamard Gate

Recall that the *first order Hadamard gate* operates on the 2-dimensional Hilbert space, \mathcal{H} , of a *single qubit* according to its effect on the CBS states

$$\begin{aligned} H|0\rangle &= |0\rangle_x = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{and} \\ H|1\rangle &= |1\rangle_x = \frac{|0\rangle - |1\rangle}{\sqrt{2}}, \end{aligned}$$

which, in an exercise, you showed was equivalent to the CBS formula

$$|x\rangle \longrightarrow \boxed{H} \longrightarrow \frac{|0\rangle + (-1)^x |1\rangle}{\sqrt{2}}.$$

We learned that its matrix is

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

and that it affects a general qubit state, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, according to

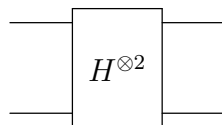
$$H|\psi\rangle = \left(\frac{\alpha + \beta}{\sqrt{2}}\right)|0\rangle + \left(\frac{\alpha - \beta}{\sqrt{2}}\right)|1\rangle.$$

🔗 : Definition and Symbol

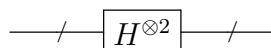
Definition. The *second order Hadamard gate*, also called the *two-qubit* or *binary* Hadamard gate, is the tensor product of two single-qubit Hadamard gates,

$$H^{\otimes 2} \equiv H \otimes H.$$

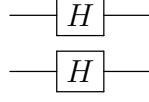
Notation. You will see both $H \otimes H$ and $H^{\otimes 2}$ when referring to the second order Hadamard gate. In a circuit diagram, it looks like this



when we want to show the separate A and B register, or like this



when we want to condense the input and output pipes into a multi-pipe. However, it is often drawn as two individual H gates applied in parallel,



$|x\rangle|y\rangle$: Action on the CBS

When a two-qubit operator is defined as a tensor product of two single-qubit operators, we get its action on CBS kets free-of-charge compliments of the definition of *product operator*. Recall that the tensor product of two operators T_1 and T_2 , *requires* that its action on separable tensors be

$$[T_1 \otimes T_2](\mathbf{v} \otimes \mathbf{w}) \equiv T_1 \mathbf{v} \otimes T_2 \mathbf{w}.$$

This forces the action of $H \otimes H$ on the $\mathcal{H}_{(2)}$ computational basis state $|0\rangle|0\rangle$ (for example) to be

$$[H \otimes H] \left(|0\rangle|0\rangle \right) = H|0\rangle H|0\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right).$$

Separability of CBS Output. Let's pause a moment to appreciate that when an operator in $\mathcal{H}_{(2)}$ is a pure product of individual operators, as this one is, *CBS states* always map to *separable states*. We can see this in the last result, and we know it will happen for the other three CBS states.

CBS Output in z -Basis Form. Separable or not, it's always good to have the basis-expansion of the gate output for the four CBS kets. Multiplying it out for $H^{\otimes 2}(|0\rangle|0\rangle)$, we find

$$[H \otimes H] \left(|0\rangle|0\rangle \right) = \frac{|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle}{2} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

Doing the same thing for all four CBS kets, we get the identities

$$\begin{aligned} H^{\otimes 2} |0\rangle|0\rangle &= \frac{|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle}{2} \\ H^{\otimes 2} |0\rangle|1\rangle &= \frac{|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle}{2} \\ H^{\otimes 2} |1\rangle|0\rangle &= \frac{|0\rangle|0\rangle + |0\rangle|1\rangle - |1\rangle|0\rangle - |1\rangle|1\rangle}{2} \\ H^{\otimes 2} |1\rangle|1\rangle &= \frac{|0\rangle|0\rangle - |0\rangle|1\rangle - |1\rangle|0\rangle + |1\rangle|1\rangle}{2}. \end{aligned}$$

Condensed Form #1. There is a single-formula version of these four CBS results, and it is needed when we move to three or more qubits, so we had better develop it now. However, it takes a little explanation, so we allow a short side trip.

First, let's switch to encoded notation ($0 \leftrightarrow 00$, $1 \leftrightarrow 01$, $2 \leftrightarrow 10$ and $3 \leftrightarrow 11$), and view the above in the equivalent form,

$$\begin{aligned} H^{\otimes 2} |0\rangle^2 &= \frac{|0\rangle^2 + |1\rangle^2 + |2\rangle^2 + |3\rangle^2}{2} \\ H^{\otimes 2} |1\rangle^2 &= \frac{|0\rangle^2 - |1\rangle^2 + |2\rangle^2 - |3\rangle^2}{2} \\ H^{\otimes 2} |2\rangle^2 &= \frac{|0\rangle^2 + |1\rangle^2 - |2\rangle^2 - |3\rangle^2}{2} \\ H^{\otimes 2} |3\rangle^2 &= \frac{|0\rangle^2 - |1\rangle^2 - |2\rangle^2 + |3\rangle^2}{2}. \end{aligned}$$

Next, I will show (with your help) that all four of these can be summarized by the single equation (note that x goes from the encoded value 0 to 3)

$$\begin{aligned} H^{\otimes 2} |x\rangle^2 \\ = \frac{(-1)^{x \odot 0} |0\rangle^2 + (-1)^{x \odot 1} |1\rangle^2 + (-1)^{x \odot 2} |2\rangle^2 + (-1)^{x \odot 3} |3\rangle^2}{2}, \end{aligned}$$

where the operator \odot in the exponent of (-1) is the *mod-2 dot product* that was defined for the unusual vector space, \mathbb{B}^2 , during the lesson on single qubits. I'll repeat it here in the current context. For two CBS states,

$$\begin{aligned} x &= x_1 x_0 \quad \text{and} \\ y &= y_1 y_0, \end{aligned}$$

where the RHS represents the two-bit string of the CBS ("00," "01," etc.), we define

$$x \odot y \equiv x_1 \cdot y_1 \oplus x_0 \cdot y_0,$$

that is, we multiply corresponding bits and take their mod-2 sum. To get overly explicit, here are a few computed mod-2 dot products:

$x = x_1 x_0$	$y = y_1 y_0$	$x \odot y$
3 = 11	3 = 11	0
1 = 01	1 = 01	1
0 = 00	2 = 10	0
1 = 01	3 = 11	1
1 = 01	2 = 10	0

I'll now confirm that the last expression presented above for $H^{\otimes 2} |x\rangle^2$, when applied to the particular CBS $|3\rangle^2 \cong |1\rangle |1\rangle$, gives the right result:

$$\begin{aligned}
H^{\otimes 2} |3\rangle^2 &= H^{\otimes 2} |11\rangle \\
&= \frac{(-1)^{11 \odot 00} |0\rangle^2 + (-1)^{11 \odot 01} |1\rangle^2 + (-1)^{11 \odot 10} |2\rangle^2 + (-1)^{11 \odot 11} |3\rangle^2}{2} \\
&= \frac{(-1)^0 |0\rangle^2 + (-1)^1 |1\rangle^2 + (-1)^1 |2\rangle^2 + (-1)^0 |3\rangle^2}{2} \\
&= \frac{(|0\rangle^2 - |1\rangle^2 - |2\rangle^2 + |3\rangle^2)}{2},
\end{aligned}$$

which matches the final of our four CBS equations for $H^{\otimes 2}$. Now, for the help I warned you would be giving me.

[Exercise. Show that this formula produces the remaining three CBS expressions for $H^{\otimes 2}$.]

We now present the condensed form using summation notation,

$$H^{\otimes 2} |x\rangle^2 = \frac{1}{2} \sum_{y=0}^3 (-1)^{x \odot y} |y\rangle^2.$$

Most authors typically don't use the " \odot " for the mod-2 dot product, but stick with a simpler " \cdot ", and add some verbiage to the effect that "this is a mod-2 dot product...", in which case you would see it as

$$H^{\otimes 2} |x\rangle^2 = \frac{1}{2} \sum_{y=0}^3 (-1)^{x \cdot y} |y\rangle^2.$$

Putting this result into the circuit diagram gives us

$$|x\rangle^2 \rightarrow \left\{ \begin{array}{c} \text{---} \text{---} \text{---} \\ \text{---} \text{---} \end{array} \begin{array}{|c|} \hline H^{\otimes 2} \\ \hline \end{array} \begin{array}{c} \text{---} \text{---} \text{---} \\ \text{---} \text{---} \end{array} \right\} \frac{1}{2} \sum_{y=0}^3 (-1)^{x \cdot y} |y\rangle^2$$

or, using the multi-channel pipe notation,

$$|x\rangle^2 \text{ --- } \boxed{H^{\otimes 2}} \text{ --- } \frac{1}{2} \sum_{y=0}^3 (-1)^{x \cdot y} |y\rangle^2$$

Condensed Form #2. You may have noticed that the separability of the CBS outputs, combined with the expression we already had for a single-qubit Hadamard,

$$H |x\rangle = \frac{|0\rangle + (-1)^x |1\rangle}{\sqrt{2}},$$

gives us another way to express H in terms of the CBS. Once again, invoking the definition of a product operator, we get

$$\begin{aligned} [H \otimes H] \left(|x\rangle |y\rangle \right) &= H |x\rangle \otimes H |y\rangle \\ &= \left(\frac{|0\rangle + (-1)^x |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + (-1)^y |1\rangle}{\sqrt{2}} \right), \end{aligned}$$

which, with much less fanfare than condensed form #1, produces a nice separable circuit diagram definition for the binary Hadamard:

$$\begin{array}{ccc} |x\rangle & \text{---} & \boxed{H^{\otimes 2}} & \text{---} & \frac{|0\rangle + (-1)^x |1\rangle}{\sqrt{2}} \\ |y\rangle & \text{---} & & \text{---} & \frac{|0\rangle + (-1)^y |1\rangle}{\sqrt{2}} \end{array}$$

which I will call “condensed form #2”.

[**Caution:** The y in this circuit diagram labels the B -register CBS ket, a totally different use from its indexing of the summation formula in condensed form #1. However, in both condensed forms the use of y is standard.]

The reason for the hard work of condensed form #1 is that it is easier to generalize to higher order qubit spaces, and proves more useful in those contexts.

[**Exercise.** Multiply-out form #2 to show it in terms of the four CBS tensor kets.]

[**Exercise.** The result of the last exercise will look very different from condensed form #1. By naming your bits and variables carefully, demonstrate that both results produce the same four scalar amplitudes “standing next to” the CBS tensor terms.]

(...) : The Matrix

This time we have two different approaches available. As before we can compute the column vectors of the matrix by applying $H^{\otimes 2}$ to the CBS tensors. However, the separability of the operator allows us to use the theory of tensor products to write down the product matrix based on the two component matrices. The need for frequent sanity checks wired into our collective computer science mindset induces us to do both.

Method #1: Tensor Theory.

Using the standard A -major (B -minor) method, a separable operator $A \otimes B$ can

be immediately written down using our formula

$$\begin{aligned}
& \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0(l-1)} \\ a_{10} & a_{11} & \cdots & a_{1(l-1)} \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix} \otimes \begin{pmatrix} b_{00} & b_{01} & \cdots & b_{0(m-1)} \\ b_{10} & b_{11} & \cdots & b_{1(m-1)} \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix} \\
&= \begin{pmatrix} a_{00} \begin{pmatrix} b_{00} & b_{01} & \cdots \\ b_{10} & b_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} & a_{01} \begin{pmatrix} b_{00} & b_{01} & \cdots \\ b_{10} & b_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} & \cdots \\ a_{10} \begin{pmatrix} b_{00} & b_{01} & \cdots \\ b_{10} & b_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} & a_{11} \begin{pmatrix} b_{00} & b_{01} & \cdots \\ b_{10} & b_{11} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}
\end{aligned}$$

For us, this means

$$\begin{aligned}
H \otimes H &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \cdot \left(\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right) & 1 \cdot \left(\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right) \\ 1 \cdot \left(\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right) & -1 \cdot \left(\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \right) \end{pmatrix} \\
&= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix},
\end{aligned}$$

a painlessly obtained result.

Method #2: Linear Algebra Theory.

Our earlier technique which works for any transformation, separable or not, instructs us to place the output images of the four CBS kets into columns of the desired

matrix. Using our four expressions for $H^{\otimes 2} |x\rangle |y\rangle$ presented initially, we learn

$$\begin{aligned}
H^{\otimes 2} |00\rangle &= \frac{1}{2} \left(|00\rangle + |01\rangle + |10\rangle + |11\rangle \right) = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \\
H^{\otimes 2} |01\rangle &= \frac{1}{2} \left(|00\rangle - |01\rangle + |10\rangle - |11\rangle \right) = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \\
H^{\otimes 2} |10\rangle &= \frac{1}{2} \left(|00\rangle + |01\rangle - |10\rangle - |11\rangle \right) = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} \\
H^{\otimes 2} |11\rangle &= \frac{1}{2} \left(|00\rangle - |01\rangle - |10\rangle + |11\rangle \right) = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
M_{H^{\otimes 2}} &= \begin{pmatrix} H^{\otimes 2} |00\rangle, & H^{\otimes 2} |01\rangle, & H^{\otimes 2} |10\rangle, & H^{\otimes 2} |11\rangle \end{pmatrix} \\
&= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.
\end{aligned}$$

happily, the same result.

[**Exercise.** Complete the sanity check by confirming that this is a unitary matrix.]

$|\psi\rangle^2$: Behavior on General State

To a general

$$|\psi\rangle^2 = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix},$$

we apply our operator

$$H^{\otimes 2} |\psi\rangle^2 = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \alpha + \beta + \gamma + \delta \\ \alpha - \beta + \gamma - \delta \\ \alpha + \beta - \gamma - \delta \\ \alpha - \beta - \gamma + \delta \end{pmatrix},$$

which shows the result of applying the two-qubit Hadamard to any state.

↘ : Measurement

There is no concise phrase we can use to describe how the binary Hadamard affects measurement probabilities of a general state. We must be content to describe it in terms of the algebra. For example, testing at point P,

$$|\psi\rangle^2 \rightarrow \left\{ \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \\ \text{---} \text{---} \text{---} \text{---} \\ \downarrow \quad \downarrow \\ \text{P} \quad \text{Q} \end{array} \begin{array}{c} \boxed{H^{\otimes 2}} \end{array} \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \\ \text{---} \text{---} \text{---} \text{---} \\ \downarrow \quad \downarrow \\ \text{Q} \end{array} \right\} H^{\otimes 2} |\psi\rangle^2$$

collapses $|\psi\rangle^2$ to $|00\rangle$ with probability $|\alpha|^2$ (as usual). Waiting, instead, to take the measurement of $H^{\otimes 2} |\psi\rangle^2$ (point Q), would produce a collapse to that same $|00\rangle$ with the probability

$$\left\| \frac{\alpha + \beta + \gamma + \delta}{2} \right\|^2 = \frac{(\alpha + \beta + \gamma + \delta)^* (\alpha + \beta + \gamma + \delta)}{4},$$

an accurate but relatively uninformative fact.

Measurement of CBS Output States. However, there *is* something we can say when we present any of the four CBS tensors to our Hadamard. Because it's separable, we can see its output clearly:

$$\begin{array}{ccc} \begin{array}{c} |x\rangle \\ |y\rangle \end{array} & \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \\ \text{---} \text{---} \text{---} \text{---} \\ \downarrow \quad \downarrow \\ \text{P} \quad \text{Q} \end{array} & \begin{array}{c} \boxed{H^{\otimes 2}} \end{array} & \begin{array}{c} \text{---} \text{---} \text{---} \text{---} \\ | \quad | \quad | \quad | \\ \text{---} \text{---} \text{---} \text{---} \\ \downarrow \quad \downarrow \\ \text{Q} \end{array} & \begin{array}{c} \frac{|0\rangle + (-1)^x |1\rangle}{\sqrt{2}} \\ \frac{|0\rangle + (-1)^y |1\rangle}{\sqrt{2}} \end{array} \end{array}$$

This tells us that if we measure *either* register at Q, we will read a “0” or “1” with equal (50%) probability, and if we measure *both* registers at Q, we’ll see a “00,” “01,” “10” or “11” also with equal (25%) probability. Compare this to the probabilities at P, where we are going to get either of our two input qubit values, “x” and “y” with 100% probability (or, if we measure both, the bipartite input value “xy” with 100% probability) since they are CBS states at the time of measurement.

Converting Between z -Basis and x -Basis using $H^{\otimes 2}$

The induced z -basis for $\mathcal{H}_{(2)}$ is

$$|0\rangle|0\rangle, \quad |0\rangle|1\rangle, \quad |1\rangle|0\rangle, \quad \text{and} \quad |1\rangle|1\rangle,$$

while the induced x -basis for $\mathcal{H}_{(2)}$ is

$$|0\rangle_x|0\rangle_x, \quad |0\rangle_x|1\rangle_x, \quad |1\rangle_x|0\rangle_x, \quad \text{and} \quad |1\rangle_x|1\rangle_x,$$

or in alternate notation,

$$|+\rangle|+\rangle, \quad |+\rangle|-\rangle, \quad |-\rangle|+\rangle, \quad \text{and} \quad |-\rangle|-\rangle.$$

Since H converts to-and-from the bases $\{|0\rangle, |1\rangle\}$ and $\{|0\rangle_x, |1\rangle_x\}$ in $\mathcal{H}_{(1)}$, it is a short two-liner to confirm that the separable $H^{\otimes 2}$ converts to-and-from their induced second order counterparts in $\mathcal{H}_{(2)}$. I'll show this for one of the four CBS kets and you can do the others. Let's take the third z -basis ket, $|1\rangle|0\rangle$,

$$\begin{aligned} H^{\otimes 2} |1\rangle|0\rangle &= [H \otimes H] (|1\rangle \otimes |0\rangle) = (H|1\rangle) \otimes (H|0\rangle) \\ &= |1\rangle_x \otimes |0\rangle_x = |1\rangle_x|0\rangle_x \cong |-\rangle|+\rangle \end{aligned}$$

[**Exercise.** Do the same for the other three CBS kets.]

This gives us a very useful way to view $H^{\otimes 2}$ when we are thinking about bases:

$H^{\otimes 2}$ is the transformation that takes the z -CBS to the x -CBS (and since it is its own inverse, also takes x -CBS back to the z -CBS).

Algebraically, using the alternate $|\pm\rangle$ notation for the x -basis, the forward direction is

$$\begin{aligned} H^{\otimes 2} |00\rangle &= |++\rangle, \\ H^{\otimes 2} |01\rangle &= |+-\rangle, \\ H^{\otimes 2} |10\rangle &= |-+\rangle \quad \text{and} \\ H^{\otimes 2} |11\rangle &= |--\rangle, \end{aligned}$$

and the inverse direction is

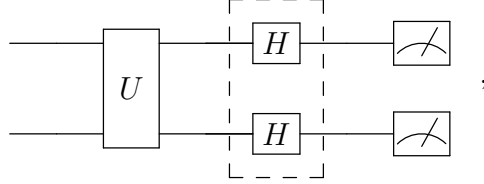
$$\begin{aligned} H^{\otimes 2} |++\rangle &= |00\rangle, \\ H^{\otimes 2} |+-\rangle &= |01\rangle, \\ H^{\otimes 2} |-+\rangle &= |10\rangle \quad \text{and} \\ H^{\otimes 2} |--\rangle &= |11\rangle. \end{aligned}$$

11.4 Measuring Along Alternate Bases

In the earlier study, when we were applying the naked CNOT gate directly to kets, I described the final step as “measuring the output relative to the x -basis.” At the time it was a bit of an informal description. How would it be done *practically*?

11.4.1 Measuring Along the x -Basis

We apply the observation of the last section. When we are ready to measure along the x -basis, we insert the quantum gate that turns the x -basis into the z -basis, then measure in our familiar z -basis. For a general circuit represented by a single operator U we would use



to measure along the x -basis. (The meter symbols imply a natural z -basis measurement, unless otherwise stated.)

11.4.2 Measuring Along *any* Separable Basis

Say we have some *non-preferred* basis,

$$\mathcal{C} = \{ |\xi_0\rangle, |\xi_1\rangle \},$$

of our first order Hilbert space, $\mathcal{H}_{(1)}$. Let's also assume we have the unary operator, call it T , that converts from the natural z -CBS to the \mathcal{C} -CBS,

$$\begin{aligned} T : z\text{-basis} &\longrightarrow \mathcal{C}, \\ T|0\rangle &= |\xi_0\rangle, \\ T|1\rangle &= |\xi_1\rangle. \end{aligned}$$

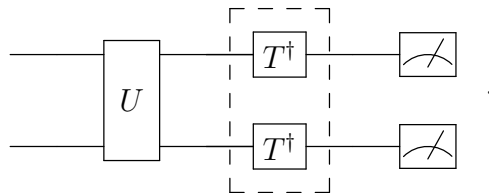
Because T is *unitary*, its inverse is T^\dagger , meaning that T^\dagger takes a \mathcal{C} -CBS back to a z -CBS.

$$\begin{aligned} T^\dagger : \mathcal{C} &\longrightarrow z\text{-basis}, \\ T^\dagger |\xi_0\rangle &= |0\rangle, \\ T^\dagger |\xi_1\rangle &= |1\rangle. \end{aligned}$$

Moving up to our second order Hilbert space, $\mathcal{H}_{(2)}$, the operator that converts from the induced 4-element z -basis to the induced 4-element \mathcal{C} -basis is $T \otimes T$, while to go in the reverse direction we would use $T^\dagger \otimes T^\dagger$,

$$\begin{aligned} (T^\dagger \otimes T^\dagger) |\xi_{00}\rangle &= |00\rangle, \\ (T^\dagger \otimes T^\dagger) |\xi_{01}\rangle &= |01\rangle, \\ (T^\dagger \otimes T^\dagger) |\xi_{10}\rangle &= |10\rangle, \\ (T^\dagger \otimes T^\dagger) |\xi_{11}\rangle &= |11\rangle. \end{aligned}$$

To measure a bipartite output state along the induced \mathcal{C} -basis, we just apply $T^\dagger \otimes T^\dagger$ instead of the Hadamard gate $H \otimes H$ at the end of the circuit:



Question to Ponder

How do we vocalize the measurement? Are we measuring the two qubits in the \mathcal{C} basis or in the z -basis? It is crucial to our understanding that we tidy up our language. There are two ways we can say it and they are equivalent, but each is very carefully worded, so please meditate on them.

1. If we apply the T^\dagger gate first, then subsequent measurement will be *in the z -basis*.
2. If we are talking about the original output registers of U before applying the T^\dagger gates to them, we would say we “measuring that pair *along the \mathcal{C} basis*.” This interpretation implies that we “intend to go on to send the qubits though the T^\dagger s, then measure them in the z -basis.”

How We Interpret the Final Results

Let’s say we plan to measure the output along the \mathcal{C} basis, which means we’ll apply the T^\dagger s before a z -basis measurement.

Assume, further, that we happen to know that, just after the main circuit but before the final basis-transforming $T^\dagger \otimes T^\dagger$, we were in one of the four \mathcal{C} -CBS states.

A final z -basis reading must, of course, always give us binary number “ x ” ($x = 0, 1, 2$ or 3). But under our stated conditions, detecting an x tells us that the original bipartite state before the T^\dagger s was $|\xi_x\rangle$. Why?

Well, the assumption that we had a \mathcal{C} -CBS ket before the T^\dagger s means that they converted it, with certainty, to its corresponding z -basis ket, and when you measure a CBS ket in its own basis, no collapse takes place – it’s already in a stationary state. So, there’s only one deterministic path that led to the x , namely, $|\xi_x\rangle$.

However, if we have some *superposition state* of \mathcal{C} -CBS kets, $|\psi\rangle^2$, at the end of our main circuit but before the basis-transforming $T^\dagger \otimes T^\dagger$, we have to describe things probabilistically. Let’s express this superposition as

$$|\psi\rangle^2 = c_{00} |\xi_{00}\rangle + c_{01} |\xi_{01}\rangle + c_{10} |\xi_{10}\rangle + c_{11} |\xi_{11}\rangle .$$

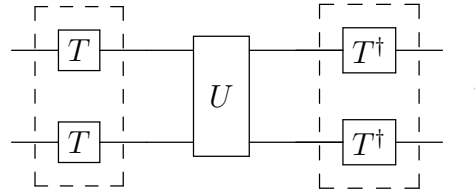
By linearity,

$$(T^\dagger \otimes T^\dagger) |\psi\rangle^2 = c_{00} |00\rangle + c_{01} |01\rangle + c_{10} |10\rangle + c_{11} |11\rangle ,$$

a result that tells us the probabilities of detecting the four natural CBS kets on our final meters are the same as the probabilities of our having detected the corresponding \mathcal{C} -CBS kets *prior* to the final $T^\dagger \otimes T^\dagger$ gate. Those probabilities are, of course, $|c_x|^2$, for $x = 0, 1, 2$ or 3 .

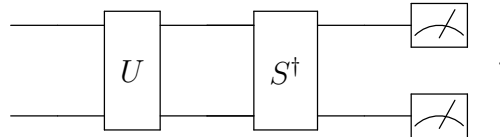
11.4.3 The Entire Circuit Viewed in Terms of an Alternate Basis

If we wanted to operate the circuit *entirely* using the alternate CBS, i.e., giving it input states defined using alternate CBS coordinates as well as measuring along the alternate basis, we would first create a circuit (represented here by a single operator U) that works in terms of the z -basis, then surround it with T and T^\dagger gates,



11.4.4 Non-Separable Basis Conversion Gates

While on the topic, I should mention *non-separable* basis conversion transformations (to/from the z -basis). A basis conversion operator doesn't have to be separable in order for it to be useful in measuring along that basis. Like all quantum gates it is *unitary*, so everything we said about separable conversion operators will work. To measure along *any* basis, we find the *binary* operator, call it S , that takes the z -basis to the other basis and use S^\dagger prior to measurement.

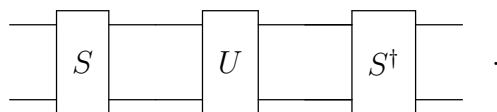


An Entire Circuit in a Non-Separable Basis

If we wanted to design a circuit in terms of input coordinates *and* measurements relative to a non-preferred (and non-separable) basis, we would

1. first define the circuit in terms of the natural z -basis, then
2. sandwich it between S and S^\dagger gates.

The circuit diagram for this is



We'll be using both separable and non-separable basis conversion operators today and in future lessons.

11.5 Variations on the Fundamental Binary Qubit Gates

We've seen three examples of two-qubit unitary operators: a general learning example, the *CNOT* and the $H^{\otimes 2}$. There is a tier of binary gates which are derivative of one or more of those three, and we can place this tier in a kind of "secondary" category and thereby leverage and/or parallel the good work we've already done.

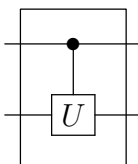
11.5.1 Other Controlled Logic Gates

The Controlled- U Gate

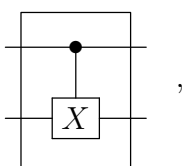
The CNOT is a special case of a more general gate which "places" any *unary* (one-qubit) operation in the B -register to be controlled by the qubit in the A -register. If U is any unary gate, we can form a binary qubit gate called a *controlled- U gate* (or *U -operator*). Loosely stated, it applies the unary operator on one register's CBS *conditionally*, based on the CBS qubit going into the other register.

⌚ : The Symbol

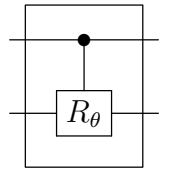
The *controlled- U gate* is drawn like the CNOT gate with CNOT's \oplus operation replaced by the unary operator, U , that we wish to control:



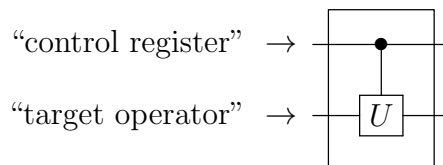
For example, a controlled-*bit flip* (= controlled-QNOT) operator would be written



whereas a controlled-phase shift operator, with a shift angle of θ , would be



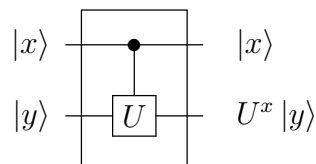
The A -register maintains its role of *control bit/register*, and the B -register the *target register* or perhaps more appropriately, *target (unary) operator*.



There is no accepted name for the operator, but I'll use the notation CU (i.e., CX , $C(R_\theta)$, etc.) in this course.

$|x\rangle |y\rangle$: Action on the CBS

It is easiest and most informative to give the effect on the CBS when we know which specific operator, U , we are controlling. Yet, even for a general U we can give formal expression using the power (exponent) of the matrix U . As with ordinary integer exponents, U^n simply means “multiply U by itself n times” with the usual convention that $U^0 = \mathbb{1}$:



The abstract expression is equivalent to the more understandable

$$|y\rangle \mapsto \begin{cases} |y\rangle, & \text{if } x = 0 \\ U |y\rangle, & \text{if } x = 1 \end{cases}$$

It will help if we see the four CBS results as explicitly:

$$\begin{aligned}
CU |00\rangle &= |0\rangle |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
CU |01\rangle &= |0\rangle |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\
CU |10\rangle &= |1\rangle U |0\rangle = \begin{pmatrix} 0 \\ 0 \\ U_{00} \\ U_{10} \end{pmatrix} \\
CU |11\rangle &= |1\rangle U |1\rangle = \begin{pmatrix} 0 \\ 0 \\ U_{01} \\ U_{11} \end{pmatrix}
\end{aligned}$$

Here the U_{jk} are the four matrix elements of M_U .

[**Exercise.** Verify this formulation.]

(...) : The Matrix

The column vectors of the matrix are given by applying CU to the CBS tensors to get

$$\begin{aligned}
M_{CU} &= \left(CU |00\rangle, \quad CU |01\rangle, \quad CU |10\rangle, \quad CU |11\rangle \right) \\
&= \left(|00\rangle, \quad |01\rangle, \quad |1\rangle U |0\rangle, \quad |1\rangle U |1\rangle \right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{pmatrix},
\end{aligned}$$

which can be more concisely expressed as

$$CU = \left(\begin{array}{c|c} \mathbf{1} & 0 \\ \hline 0 & U \end{array} \right)$$

or, the even cleaner

$$= \left(\begin{array}{c|c} \mathbf{1} & \\ \hline & U \end{array} \right).$$

[**Exercise.** Prove that this is a separable operator for some U and not-separable for others. **Hint:** What did we say about CNOT in this regard?]

[**Exercise.** Confirm unitarity.]

$|\psi\rangle^2$: Behavior on General State

Applying CU to the general state,

$$|\psi\rangle^2 = \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \gamma |1\rangle |0\rangle + \delta |1\rangle |1\rangle ,$$

we get

$$\begin{aligned} CU |\psi\rangle^2 &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ U_{00} \gamma + U_{01} \delta \\ U_{10} \gamma + U_{11} \delta \end{pmatrix} \\ &= \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + (U_{00} \gamma + U_{01} \delta) |1\rangle |0\rangle + (U_{10} \gamma + U_{11} \delta) |1\rangle |1\rangle . \end{aligned}$$

\searrow : Measurement

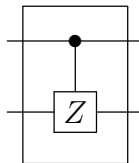
There's nothing new we have to add here, since measurement probabilities of the target-register will depend on the specific U being controlled, and we have already established that the measurement probabilities of the control register are unaffected (see CNOT discussion).

The Controlled-Z Gate

Let's get specific by looking at CZ.

\bullet : The Symbol

Its symbol in a circuit is



$|x\rangle |y\rangle$: Action on the CBS

Recall that we had what appeared to be an overly abstract expression for the unary Z -gate operating on a CBS $|x\rangle$,

$$Z |x\rangle = (-1)^x |x\rangle ,$$

but this actually helps us understand the CZ action using a similar expression:

$$\text{CZ} |x\rangle |y\rangle = (-1)^{xy} |x\rangle |y\rangle .$$

[**Exercise.** Prove this formula is correct. **Hint:** Apply the *wordy* definition of a controlled Z -gate (“*leaves the B -reg alone if ... and applies Z to the B -reg if ...*”) to the four CBS tensors and compare what you get with the formula $(-1)^{xy} |x\rangle |y\rangle$ for each of the four combinations of x and y .]

To see the four CBS results as explicitly,

$$\begin{aligned} \text{CZ} |00\rangle &= \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\ \text{CZ} |01\rangle &= \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\ \text{CZ} |10\rangle &= \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\ \text{CZ} |11\rangle &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} \end{aligned}$$

$\begin{pmatrix} \cdots \end{pmatrix}$: The Matrix

The column vectors of the matrix were produced above, so we can write it down instantly:

$$\text{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

[**Exercise.** Prove that this (a) is not a separable operator and (b) is unitary.]

$|\psi\rangle^2$: Behavior on General State

Applying CU to the general state,

$$|\psi\rangle^2 = \alpha |0\rangle |0\rangle + \beta |0\rangle |1\rangle + \gamma |1\rangle |0\rangle + \delta |1\rangle |1\rangle ,$$

we get

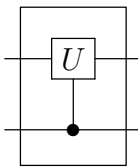
$$CZ |\psi\rangle^2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ -\delta \end{pmatrix}.$$

↘ : Measurement

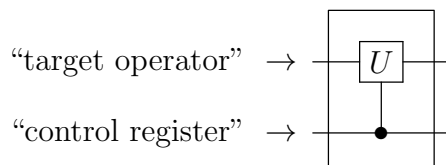
As we noticed with the unary Z -operator, the probabilities of measurement (along the preferred CBS) are not affected by the controlled- Z . However, the state *is* modified. You can demonstrate this the same way we did for the unary Z : combine $|\psi\rangle^2$ and $CZ |\psi\rangle^2$ with a second tensor and show that you (can) produce distinct results.

Swapping Roles in Controlled Gates

We could have (and still can) turn any of our controlled gates upside down:

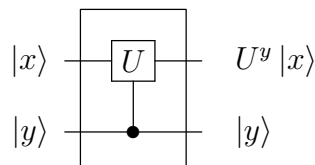


Now the B -register takes on the role of *control*, and the A -register becomes the *target*:



Let's refer to this version of a controlled gate using the notation (caution: not seen outside of this course) $(C\uparrow)U$

Everything has to be adjusted accordingly if we insist – which we do – on continuing to call the upper register the A -register, producing A -major (B -minor) matrices and coordinate representations. For example, the action on the CBS becomes



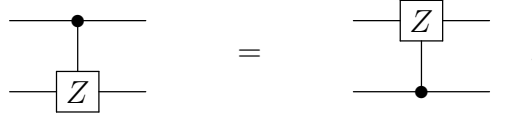
and the matrix we obtain (make sure you can compute this this) is

$$(C\uparrow)U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & U_{00} & 0 & U_{01} \\ 0 & 0 & 1 & 0 \\ 0 & U_{10} & 0 & U_{11} \end{pmatrix},$$

where the U_{jk} are the four matrix elements of M_U .

With this introduction, try the following exercise.

[**Exercise.** Prove that the binary quantum gates CZ and $(C\uparrow)Z$ are identical. That is, show that



Hint: If they are equal on the CBS, they are equal *period*.]

11.5.2 More Separable Logic Gates

The binary Hadamard gate, $H \otimes H$, is just one example of many possible separable gates which, by definition, are constructed as the tensor product of two unary operators. Take any two single-qubit gates, say X and H . We form the product operator, and use tensor theory to immediately write down its matrix,

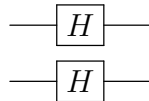
$$\begin{aligned} X \otimes H &= \begin{pmatrix} 0 \cdot M_H & 1 \cdot M_H \\ 1 \cdot M_H & 0 \cdot M_H \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix}. \end{aligned}$$

[**Exercise.** Verify unitarity.]

Vocabulary. Separable operators are also called *local operators*.

Separable Operators on Separable States. All separable operators take separable states to other separable states, therefore, they preserve *locality* (which is why they are also known as *local operators*). In particular, their action on the CBS result in separable outputs.

One of the notational options we had for the binary Hadamard gate,



provided a visual representation of this locality. It correctly pictures the non-causal separation between the two channels when separable inputs are presented to the gate. This is true for all separable operators like our current $X \otimes H$,

$$\begin{array}{ccc} |\psi\rangle & \text{---} \boxed{X} \text{---} & X|\psi\rangle \\ \otimes & & \otimes \\ |\varphi\rangle & \text{---} \boxed{H} \text{---} & H|\varphi\rangle \end{array}$$

demonstrating the complete isolation of each channel, *but only when separable states are sent to the gate*. If an entangled state goes in, an entangled state will come out. The channels are still separate, but the input and output must be displayed as unified states,

$$|\psi\rangle^2 \rightarrow \left\{ \begin{array}{c} \text{---} \boxed{X} \text{---} \\ \text{---} \boxed{H} \text{---} \end{array} \right\} \rightarrow (X \otimes H) |\psi\rangle^2$$

Separable Operators on Entangled States

A single-qubit operator, U , applied to one qubit of an entangled state is equivalent to a separable binary operator like $\mathbb{1} \otimes U$ or $U \otimes \mathbb{1}$. Such “local” operations will generally modify *both* qubits of the entangled state, so a separable operator’s effect is not restricted to only one channel for such states. To reinforce this concept, let U be any unary operator on the A -space and form the separable operator

$$U \otimes \mathbb{1} : \mathcal{H}_A \otimes \mathcal{H}_B \longrightarrow \mathcal{H}_A \otimes \mathcal{H}_B.$$

Let’s apply this to a general $|\psi\rangle^2$, allowing the possibility that it’s an entangled state.

$$\begin{aligned} (U \otimes \mathbb{1}) |\psi\rangle^2 &= [U \otimes \mathbb{1}] \left(\alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle \right) \\ &= \alpha (U|0\rangle) |0\rangle + \beta (U|0\rangle) |1\rangle + \gamma (U|1\rangle) |0\rangle + \delta (U|1\rangle) |1\rangle. \end{aligned}$$

Replacing $U|0\rangle$ and $U|1\rangle$ with their expansions along the CBS (and noting that the matrix elements, U_{jk} , are the weights), we get

$$\begin{aligned} (U \otimes \mathbb{1}) |\psi\rangle^2 &= \alpha (U_{00} |0\rangle + U_{10} |1\rangle) |0\rangle + \beta (U_{00} |0\rangle + U_{10} |1\rangle) |1\rangle \\ &\quad + \gamma (U_{01} |0\rangle + U_{11} |1\rangle) |0\rangle + \delta (U_{01} |0\rangle + U_{11} |1\rangle) |1\rangle. \end{aligned}$$

Now, distribute and collect terms for the four CBS tensor basis, to see that

$$(U \otimes \mathbb{1}) |\psi\rangle^2 = (\alpha U_{00} + \gamma U_{01}) |00\rangle + \dots$$

and without even finishing the regrouping we see that the amplitude of $|00\rangle$ can be totally different from its original amplitude, proving that the new entangled state is different, and potentially just as entangled.

[Exercise. Complete the unfinished expression I started for $(U \otimes \mathbb{1}) |\psi\rangle^2$, collect terms for the four CBS kets, and combine the square-magnitudes of $|0\rangle_B$ to get the *probability* of B measuring a “0”. Do the same for the $|1\rangle_B$ to get the probability of B measuring a “1”. If you compute carefully, you’ll find that the probabilities are unaffected by the application of $U \otimes \mathbb{1}$. (**Hint:** use the unitarity of U to make various terms vanish). **Conclusion:** While the amplitudes of $(U \otimes \mathbb{1}) |\psi\rangle^2$ are all changed, the measurement probabilities are not. This *does not mean* the before and

after states are effectively the same. (They combine differently with almost any third state.)]

Vocabulary. This is called performing a *local operation* on one qubit of an entangled pair.

The Matrix. It will be handy to have the matrix for the operator $U \otimes \mathbb{1}$ at the ready for future algorithms. It can be written down instantly using the rules for separable operator matrices (see the lesson on tensor products),

$$\begin{pmatrix} U_{00} & 0 & U_{01} & 0 \\ 0 & U_{00} & 0 & U_{01} \\ U_{10} & 0 & U_{11} & 0 \\ 0 & U_{10} & 0 & U_{11} \end{pmatrix}.$$

Example. Alice and Bob each share one qubit of the entangled pair

$$|\beta_{00}\rangle \equiv \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

which you'll notice I have named $|\beta_{00}\rangle$ for reasons that will be made clear later today when we get to the Bell states. Alice will hold the *A*-register qubit (on the left of each product) and Bob the *B*-register qubit, so you may wish to view the state using the notation

$$\frac{|0\rangle_A |0\rangle_B + |1\rangle_A |1\rangle_B}{\sqrt{2}}.$$

Alice sends her qubit (i.e., the *A*-register) through a local QNOT operator,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

and Bob does nothing. This describes the full *local* operator

$$X \otimes \mathbb{1}$$

applied to the entire bipartite state. We want to know the effect this has on the total entangled state, so we apply the matrix for $X \otimes \mathbb{1}$ to the state. Using our pre-computed matrix for the general $U \otimes \mathbb{1}$ with $U = X$, we get

$$\begin{aligned} (X \otimes \mathbb{1}) |\beta_{00}\rangle &= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \\ &= \frac{|01\rangle + |10\rangle}{\sqrt{2}}. \end{aligned}$$

We could have gotten the same result, perhaps more quickly, by distributing $X \otimes \mathbb{1}$ over the superposition and using the identity for separable operators,

$$[S \otimes T](\mathbf{v} \otimes \mathbf{w}) = S(\mathbf{v}) \otimes T(\mathbf{w}).$$

Either way, the result will be used in our first quantum algorithm (*superdense coding*), so it's worth studying carefully. To help you, here is an exercise.

[Exercise.] Using the entangled state $|\beta_{00}\rangle$ as input, prove that the following local operators applied on Alice's end produce the entangled output states shown:

- $(Z \otimes \mathbb{1}) |\beta_{00}\rangle = (|00\rangle - |11\rangle) / \sqrt{2},$
- $(iY \otimes \mathbb{1}) |\beta_{00}\rangle = (|01\rangle - |10\rangle) / \sqrt{2}$

and, for completeness, the somewhat trivial

- $(\mathbb{1} \otimes \mathbb{1}) |\beta_{00}\rangle = (|00\rangle + |11\rangle) / \sqrt{2}. \quad]$

11.6 The Born Rule and Partial Collapse

We've had some experience with the famous entangled state

$$|\beta_{00}\rangle = \frac{|0\rangle|0\rangle + |1\rangle|1\rangle}{\sqrt{2}}.$$

If we measure the A -register of this state, it will of course collapse to either $|0\rangle_A$ or $|1\rangle_A$ as all good single qubits must. That, in turn, forces the B -register into its version of that *same state*, $|0\rangle_B$ or $|1\rangle_B$, meaning if A collapsed to $|0\rangle_A$, B would collapse to $|0\rangle_B$, and likewise for $|1\rangle$; there simply are no terms present in $|\beta_{00}\rangle$'s CBS expansion in which A and B are in different states. In more formal language, the CBS terms in which the A qubit and B qubit differ have *zero amplitude* and therefore zero collapse probability.

To make sure there is no doubt in your mind, consider a different state,

$$|\psi\rangle^2 = \frac{|0\rangle|1\rangle + |1\rangle|0\rangle}{\sqrt{2}}.$$

This time, if we measured the A -register and it collapsed to $|0\rangle_A$, that would force the B -register to collapse to $|1\rangle_B$.

[Exercise.] Explain this last statement. What happens to the B -register if we measure the A -register and it collapses to $|1\rangle_A$?

All of this is an example of a more general phenomenon in which we measure one qubit of an entangled bipartite state.

11.6.1 The Born Rule for a Two-Qubit System

Take the most general state

$$|\psi\rangle^2 = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix},$$

and rearrange it so that either the A -kets or B -kets are factored out of common terms. Let's factor the A -kets for this illustration.

$$|\psi\rangle^2 = |0\rangle_A (\alpha|0\rangle_B + \beta|1\rangle_B) + |1\rangle_A (\gamma|0\rangle_B + \delta|1\rangle_B).$$

(I labeled the state spaces of each ket to reinforce which kets belong to which register, but position implies this information even without the labels. I will often label a particular step in a long computation when I feel it helps, leaving the other steps unlabeled.)

What happens if we measure A and get a “0”? Since there is only one term which matches this state, namely,

$$|0\rangle (\alpha|0\rangle + \beta|1\rangle),$$

we are forced to conclude that the B -register is left in the non-normalized state

$$\alpha|0\rangle + \beta|1\rangle.$$

There are a couple things that may be irritating you at this point.

1. We don't actually have a QM postulate that seems to suggest this claim.
2. We are not comfortable that B 's imputed state is not normalized.

We are on firm ground, however, because when the postulates of quantum mechanics are presented in their full generality, both of these concerns are addressed. The *fifth postulate* of QM (**Trait #7**), which addresses post-measurement collapse has a generalization sometimes called the *generalized Born rule*. For the present, we'll satisfy ourselves with a version that applies only to a bipartite state's one-register measurement. We'll call it the ...

Trait #15 (Born Rule for Bipartite States): *If a bipartite state is factored relative to the A -register,*

$$|\psi\rangle^2 = |0\rangle (\alpha|0\rangle + \beta|1\rangle) + |1\rangle (\gamma|0\rangle + \delta|1\rangle),$$

a measurement of the A -register will cause the collapse of the B -register according to

$$\begin{aligned} A \searrow 0 &\Rightarrow B \searrow \frac{\alpha |0\rangle + \beta |1\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}} \\ A \searrow 1 &\Rightarrow B \searrow \frac{\gamma |0\rangle + \delta |1\rangle}{\sqrt{|\gamma|^2 + |\delta|^2}}. \end{aligned}$$

Note how this handles the non-normality of the state $\alpha |0\rangle + \beta |1\rangle$: we divide through by the norm $\sqrt{|\alpha|^2 + |\beta|^2}$. (The same is seen for the alternative state $\gamma |0\rangle + \delta |1\rangle$).

Vocabulary. We'll call this simply the “*Born Rule*.”

Trait #15 has a partner which tells us what happens if we first factor out the B -ket and measure the B -register.

[**Exercise.** State the *Born Rule* when we factor and measure the B -register.]

Checking the Born Rule in Simple Cases

You should always confirm your understanding of a general rule by trying it out on simple cases to which you already have an answer.

Example. The state we encountered,

$$|\psi\rangle^2 = \frac{|0\rangle|0\rangle + |1\rangle|1\rangle}{\sqrt{2}},$$

has

$$\begin{aligned} \alpha &= \delta = 1 & \text{and} \\ \beta &= \gamma = 0, \end{aligned}$$

so if we measure the A -register and find it to be in the state $|0\rangle_A$ (by a measurement of “0” on that register), the *Born rule* tells us that the state remaining the B -register should be

$$\frac{\alpha |0\rangle + \beta |1\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}} = \frac{1 \cdot |0\rangle + 0 \cdot |1\rangle}{\sqrt{1^2 + 0^2}} = |0\rangle,$$

which is what we concluded on more elementary grounds. ✓

[**Exercise.** Use this technique to prove that if we start in the same state and, instead, measure the B -register with a result of “1,” then the A -register is left in the state $|1\rangle_A$ with certainty.]

Example. Let's take a somewhat less trivial bipartite state. Consider

$$|\psi\rangle^2 = \frac{|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle}{2},$$

and imagine that we test the B -register and find that it “decided” to collapse to $|0\rangle_B$. To see what state this leaves the A -register in, factor out the B -kets of the original to get an improved view of $|\psi\rangle^2$,

$$|\psi\rangle^2 = \frac{(|0\rangle_A + |1\rangle_A)|0\rangle_B + (|0\rangle_A - |1\rangle_A)|1\rangle_B}{2}.$$

Examination of this expression tells us that the A -register corresponding to $|0\rangle_B$ is some *normalized* representation of the vector $|0\rangle + |1\rangle$. Let’s see if the Born rule gives us that result. The expression’s four scalars are

$$\begin{aligned}\alpha &= \beta = \gamma = \frac{1}{2} \quad \text{and} \\ \delta &= -\frac{1}{2},\end{aligned}$$

so a B -register collapse to $|0\rangle_B$ will, according to the Born rule, leave the A -register in the state

$$\frac{\alpha|0\rangle + \gamma|1\rangle}{\sqrt{|\alpha|^2 + |\gamma|^2}} = \frac{\frac{1}{2}|0\rangle + \frac{1}{2}|1\rangle}{\sqrt{(1/2)^2 + (1/2)^2}} = \frac{|0\rangle + |1\rangle}{\sqrt{2}},$$

again, the expected normalized state.

[**Exercise.** Show that a measurement of $B \searrow 1$ for the same state results in an A -register collapse to $(|0\rangle - |1\rangle)/\sqrt{2}$.]

Application of Born Rule to a Gate Output

The Born rule gets used in many important quantum algorithms, so there’s no danger of over-doing our practice. Let’s take the separable gate $\mathbb{1} \otimes H$, whose matrix you should (by now) be able to write down blindfolded,

$$\mathbb{1} \otimes H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & 1 & 1 \\ & & 1 & -1 \end{pmatrix}.$$

Hand it the most general $|\psi\rangle^2 = (\alpha, \beta, \gamma, \delta)^t$, which will produce gate output

$$\frac{1}{2} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \\ \gamma + \delta \\ \gamma - \delta \end{pmatrix}$$

We measure the B register and get a “1”. To see what’s left in the A -register, we factor the B -kets at the output,

$$\begin{aligned}(\mathbb{1} \otimes H)|\psi\rangle^2 &= \left((\alpha + \beta)|0\rangle + (\gamma + \delta)|1\rangle\right)|0\rangle \\ &\quad + \left((\alpha - \beta)|0\rangle + (\gamma - \delta)|1\rangle\right)|1\rangle.\end{aligned}$$

(At this point, we have to pause to avoid notational confusion. The “ α ” of the Born rule is actually our current $(\alpha - \beta)$, with similar unfortunate name conflicts for the other three Born variables, all of which I’m sure you can handle.) The Born rule says that the A -register will collapse to

$$\frac{(\alpha - \beta) |0\rangle + (\gamma - \delta) |1\rangle}{\sqrt{|\alpha - \beta|^2 + |\gamma - \delta|^2}},$$

which is as far as we need to go, although it won’t hurt for you to do the ...

[**Exercise.** Simplify this and show that it is a normal vector.]

11.7 Multi-Gate Circuits

We have all the ingredients to make countless quantum circuits from the basic binary gates introduced above. We’ll start with the famous Bell states.

11.7.1 A Circuit that Produces Bell States

There are four pairwise entangled states, known as *Bell states* or *EPR pairs* (for the physicists Bell, Einstein, Podolsky and Rosen who discovered their special qualities). We’ve already met one,

$$|\beta_{00}\rangle \equiv \frac{|00\rangle + |11\rangle}{\sqrt{2}},$$

and the other three are

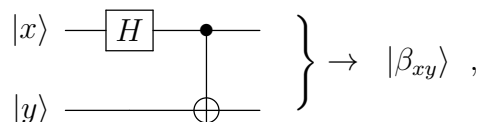
$$\begin{aligned} |\beta_{01}\rangle &\equiv \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \\ |\beta_{10}\rangle &\equiv \frac{|00\rangle - |11\rangle}{\sqrt{2}} \quad \text{and} \\ |\beta_{11}\rangle &\equiv \frac{|01\rangle - |10\rangle}{\sqrt{2}}. \end{aligned}$$

The notation I have adopted above is after *Nielsen and Chuang*, but physicists also use the alternative symbols

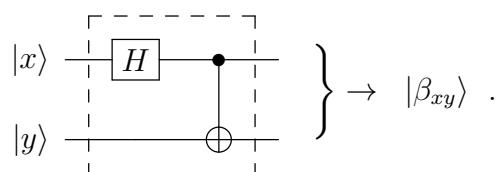
$$\begin{aligned} |\beta_{00}\rangle &\longrightarrow |\Phi^+\rangle, \\ |\beta_{01}\rangle &\longrightarrow |\Psi^+\rangle, \\ |\beta_{10}\rangle &\longrightarrow |\Phi^-\rangle \quad \text{and} \\ |\beta_{11}\rangle &\longrightarrow |\Psi^-\rangle. \end{aligned}$$

In addition, I am not using the superscript to denote a bipartite state ($|\beta_{00}\rangle^2$), since the double index on β (β_{00}) tells the story.

The circuit that produces these four states using the standard CBS basis for $\mathcal{H}_{(2)}$ as inputs is



which can be seen as a combination of a unary Hadamard gate with a CNOT gate. We could emphasize that this is a binary gate in its own right by calling it BELL and boxing it,



In concrete terms, the algebraic expression,

$$\text{BELL} (|x\rangle |y\rangle) = |\beta_{xy}\rangle$$

is telling us that

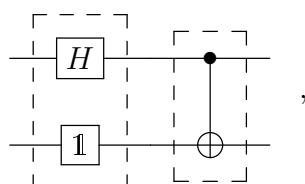
$$\begin{aligned} \text{BELL} (|0\rangle |0\rangle) &= |\beta_{00}\rangle , \\ \text{BELL} (|0\rangle |1\rangle) &= |\beta_{01}\rangle , \\ \text{BELL} (|1\rangle |0\rangle) &= |\beta_{10}\rangle \quad \text{and} \\ \text{BELL} (|1\rangle |1\rangle) &= |\beta_{11}\rangle . \end{aligned}$$

The Bell States Form an Orthonormal Basis

When studying unary quantum gates, we saw that they take orthonormal bases to orthonormal bases. The same argument – unitarity – proves this to be true of any dimension Hilbert space. Consequently, the Bell states form an orthonormal basis for $\mathcal{H}_{(2)}$.

The Matrix for BELL

The matrix for this gate can be constructed using the various techniques we have already studied. For example, you can use the standard linear algebra approach of building columns for the matrix from the four outputs of the gate. For variety, let's take a different path. The *A*-register Hadamard has a *plain quantum wire* below it, meaning the *B*-register is implicitly performing an *identity operator* at that point. So we could write the gate using the equivalent symbolism



a visual that demonstrates the application of two *known* matrices in series, that is,

$$\begin{aligned} \text{BELL} &= (CNOT)(H \otimes \mathbb{1}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}. \end{aligned}$$

At this point, you should do a few things.

[Exercise.

1. Explain why the order of the matrices is opposite the order of the gates in the diagram.
2. Confirm the matrix multiplication, above.
3. Confirm that the matrix is unitary.
4. Confirm that the matrix gives the four Bell states when one presents the four CBS states as inputs. (See below for a hint.)
5. Demonstrate that BELL is not separable. **Hint:** What do we know about the matrix of a separable operator?]

I'll do **item 4** for the input state $|10\rangle$, just to get the blood flowing.

$$\begin{aligned} \text{BELL } |10\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix} \\ &= \frac{|00\rangle - |11\rangle}{\sqrt{2}} = |\beta_{10}\rangle. \quad \checkmark \end{aligned}$$

Four Bell States from One

We did an example earlier that demonstrated that

$$(X \otimes \mathbb{1}) |\beta_{00}\rangle = (|01\rangle + |10\rangle) / \sqrt{2},$$

and followed it with some exercises. We can now list those results in the language of the EPR pairs.

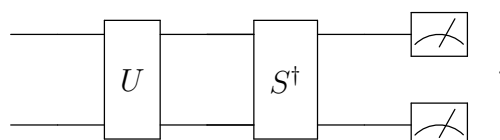
$$\begin{aligned} (\mathbf{1} \otimes \mathbf{1}) |\beta_{00}\rangle &= |\beta_{00}\rangle \\ (X \otimes \mathbf{1}) |\beta_{00}\rangle &= |\beta_{01}\rangle \\ (Z \otimes \mathbf{1}) |\beta_{00}\rangle &= |\beta_{10}\rangle \\ (iY \otimes \mathbf{1}) |\beta_{00}\rangle &= |\beta_{11}\rangle \end{aligned}$$

This might be a good time to appreciate one of today's earlier observations: *a local (read "separable") operation on an entangled state changes the entire state, affecting both qubits of the entangled pair.*

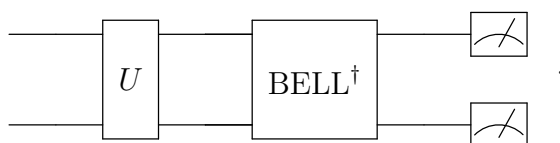
BELL as a Basis Transforming Operator

The operator BELL takes natural CBS kets to the four Bell kets, the latter shown to be an orthonormal basis for $\mathcal{H}_{(2)}$. But that's exactly what we call a basis transforming operator. Viewed in this light BELL, like $H^{\otimes 2}$, can be used when we want to change our basis. Unlike $H^{\otimes 2}$, however, BELL is *not separable* (a recent exercise) and *not its own inverse* (to be proven in a few minutes).

Measuring Along the Bell Basis. We saw that to measure along *any* basis, we find the *binary* operator, call it S , that takes the z -basis to the other basis and use S^\dagger prior to measurement.



Thus, to measure along the BELL basis (and we will, next lecture), we plug in BELL for S ,



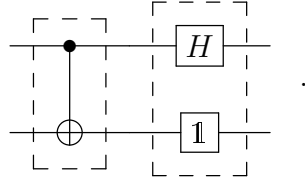
And what *is* $BELL^\dagger$? Using the *adjoint conversion rules*, and remembering that the order of operators in the circuit is opposite of that in the algebra, we find

$$\begin{aligned} BELL^\dagger &= \left[(CNOT) (H \otimes \mathbf{1}) \right]^\dagger = (H \otimes \mathbf{1})^\dagger (CNOT)^\dagger \\ &= (H \otimes \mathbf{1}) (CNOT), \end{aligned}$$

the final equality a consequence of the fact that CNOT and $H \otimes \mathbf{1}$ are both self-adjoint.

[**Exercise.** Prove this last claim using the matrices for these two binary gates.]

In other words, we just reverse the order of the two sub-operators that comprise BELL. This makes the circuit diagram for BELL^\dagger come out to be



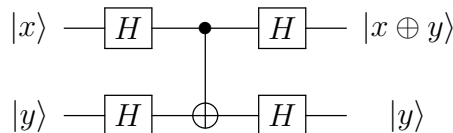
The matrix for BELL^\dagger is easy to derive since we just take the transpose (everything's real so no complex conjugation necessary):

$$\text{BELL}^\dagger = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}^\dagger = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}.$$

11.7.2 A Circuit that Creates an Upside-Down CNOT

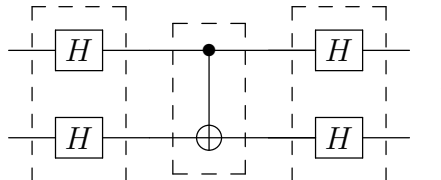
Earlier today we demonstrated that the CNOT gate, when presented and measured relative to the x -basis, behaved "upside down". Now, I'm going to show you how to turn this into a circuit that creates an "upside down" CNOT in the standard z -basis CBS.

First, the answer. Here is your circuit, which we'll call " $\text{C}\uparrow\text{NOT}$," because it is controlled from bottom-up:



Notice that it has a right-side up (normal) CNOT gate at its core.

The first thing we can do is verify the claim indicated at the output registers. As we just did with the BELL gate, we can do this most directly by multiplying the matrices of the three binary sub-gates evident by grouping this gate as



That corresponds to the matrix product

$$\begin{aligned}
C \uparrow \text{NOT} &= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \\
&= \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 4 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.
\end{aligned}$$

This is an easy matrix to apply to the four CBS kets with the following results:

$$\begin{aligned}
C \uparrow \text{NOT} : |00\rangle &\mapsto |00\rangle \\
C \uparrow \text{NOT} : |01\rangle &\mapsto |11\rangle \\
C \uparrow \text{NOT} : |10\rangle &\mapsto |10\rangle \\
C \uparrow \text{NOT} : |11\rangle &\mapsto |01\rangle
\end{aligned}$$

We can now plainly see that the B -register is controlling the A -register's QNOT operation, as claimed.

Interpreting the Upside-Down CNOT Circuit

We now have two different studies of the upside-down CNOT. The first study concerned the “naked” CNOT and resulted in the observation that, relative to the x -CBS, the B -register controlled the QNOT (X) operation on the A -register, thus it looks upside-down if you are an x -basis ket. The second and current study concerns a new circuit that had the CNOT surrounded by Hadamard gates and, taken as a whole is a truly upside-down CNOT viewed in the ordinary z -CBS. How do these two studies relate?

The key to understanding this comes from our recent observation that $H^{\otimes 2}$ can be viewed as a way to convert between the x -basis and the z -basis (in either direction since it's its own inverse).

Thus, we use the first third of the three-part circuit to let $H^{\otimes 2}$ take z -CBS kets to x -CBS kets. Next, we allow CNOT to act on the x -basis, which we saw from our earlier study caused the B -register to be the control qubit – *because and only because we are looking at x -basis kets*. The output will be x -CBS kets (since we put x -CBS kets into the central CNOT). Finally, in the last third of the circuit we let $H^{\otimes 2}$ convert the x -CBS kets back to z -CBS kets.

11.8 More than Two Qubits

We will officially introduce n -qubit systems for $n > 2$ next week, but we can find ways to use binary qubit gates in circuits that have more than two inputs immediately as long as we operate on no more than two qubits at a-time. This will lead to our first quantum algorithms.

11.8.1 Order-3 Tensor Products

If a *second order* product space is the tensor product of two vector spaces,

$$W = A \otimes B,$$

then it's easy to believe that a *third order* product space, would be constructed from three component spaces,

$$W = A \otimes B \otimes C.$$

This can be formalized by relying on the second order construction, and applying it twice, e.g.,

$$W = (A \otimes B) \otimes C.$$

It's actually less confusing to go through our order-2 tensor product development and just extend all the definitions so that they work for three component spaces. For example, taking a page from our tensor product lecture, we would start with the formal vector symbols

$$\mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}$$

and produce all *finite sums* of these things. Then, as we did for the $A \otimes B$ product space, we could define tensor addition and scalar multiplication. The basic concepts extend directly to third order product spaces. I'll cite a few highlights.

- The dimension of the product space, W , is the product of the three dimensions,

$$\dim(W) = \dim(A) \cdot \dim(B) \cdot \dim(C)$$

and W has as its basis

$$\{ \mathbf{w}_{jkl} \equiv \mathbf{a}_j \otimes \mathbf{b}_k \otimes \mathbf{c}_l \},$$

where $\{\mathbf{a}_j\}$, $\{\mathbf{b}_k\}$ and $\{\mathbf{c}_l\}$ are the bases of the three component spaces.

- The vectors (tensors) in the product space are uniquely expressed as superpositions of these basis tensors so that a typical tensor in W can be written

$$\mathbf{w} = \sum_{j,k,l} c_{jkl} (\mathbf{a}_j \otimes \mathbf{b}_k \otimes \mathbf{c}_l),$$

where c_{jkl} are the amplitudes of the CBS kets, scalars which we had been naming α , β , γ , etc. in a simpler era.

- A *separable operator* on the product space is one that arises from three component operators, T_A , T_B and T_C , each defined on its respective component space, A , B and C . This separable tensor operator is defined first by its action on separable order-3 tensors

$$[T_A \otimes T_B \otimes T_C](\mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}) \equiv T_A(\mathbf{a}) \otimes T_B(\mathbf{b}) \otimes T_C(\mathbf{c})$$

and since the basis tensors are of this form, that establishes the action of $T_A \otimes T_B \otimes T_C$ on the basis which in turn extends the action to the whole space.

If any of this seems hazy, I encourage you to refer back to the tensor product lecture and fill in details so that they extend to three component spaces.

[**Exercise.** Replicate the development of an order-2 tensor product space from our past lecture to order-3 using the above definitions as a guide.]

11.8.2 Three Qubit Systems

Definition of Three Qubits. *Three qubits are, collectively, the tensor product space $\mathcal{H} \otimes \mathcal{H} \otimes \mathcal{H}$, and the value of those qubits can be any tensor having unit length.*

Vocabulary and Notation. Three qubits are often referred to as a *tripartite* system.

To distinguish the three identical component spaces, we sometimes use subscripts,

$$\mathcal{H}_A \otimes \mathcal{H}_B \otimes \mathcal{H}_C.$$

The *order* of the tensor product, this time three, can be used to label the state space:

$$\mathcal{H}_{(3)}.$$

11.8.3 Tripartite Bases and Separable States

The tensor product of three 2-D vector spaces has dimension $2 \times 2 \times 2 = 8$. Its inherited preferred basis tensors are the separable products of the component space vectors,

$$\left\{ \begin{array}{l} |0\rangle \otimes |0\rangle \otimes |0\rangle, \quad |0\rangle \otimes |0\rangle \otimes |1\rangle, \quad |0\rangle \otimes |1\rangle \otimes |0\rangle, \quad |0\rangle \otimes |1\rangle \otimes |1\rangle, \\ |1\rangle \otimes |0\rangle \otimes |0\rangle, \quad |1\rangle \otimes |0\rangle \otimes |1\rangle, \quad |1\rangle \otimes |1\rangle \otimes |0\rangle, \quad |1\rangle \otimes |1\rangle \otimes |1\rangle \end{array} \right\}.$$

The shorthand alternatives are

$$\begin{aligned}
|0\rangle \otimes |0\rangle \otimes |0\rangle &\longleftrightarrow |0\rangle |0\rangle |0\rangle \longleftrightarrow |000\rangle \longleftrightarrow |0\rangle^3 \\
|0\rangle \otimes |0\rangle \otimes |1\rangle &\longleftrightarrow |0\rangle |0\rangle |1\rangle \longleftrightarrow |001\rangle \longleftrightarrow |1\rangle^3 \\
|0\rangle \otimes |1\rangle \otimes |0\rangle &\longleftrightarrow |0\rangle |1\rangle |0\rangle \longleftrightarrow |010\rangle \longleftrightarrow |2\rangle^3 \\
|0\rangle \otimes |1\rangle \otimes |1\rangle &\longleftrightarrow |0\rangle |1\rangle |1\rangle \longleftrightarrow |011\rangle \longleftrightarrow |3\rangle^3 \\
|1\rangle \otimes |0\rangle \otimes |0\rangle &\longleftrightarrow |1\rangle |0\rangle |0\rangle \longleftrightarrow |100\rangle \longleftrightarrow |4\rangle^3 \\
|1\rangle \otimes |0\rangle \otimes |1\rangle &\longleftrightarrow |1\rangle |0\rangle |1\rangle \longleftrightarrow |101\rangle \longleftrightarrow |5\rangle^3 \\
|1\rangle \otimes |1\rangle \otimes |0\rangle &\longleftrightarrow |1\rangle |1\rangle |0\rangle \longleftrightarrow |110\rangle \longleftrightarrow |6\rangle^3 \\
|1\rangle \otimes |1\rangle \otimes |1\rangle &\longleftrightarrow |1\rangle |1\rangle |1\rangle \longleftrightarrow |111\rangle \longleftrightarrow |7\rangle^3
\end{aligned}$$

The notation of the first two columns admits the possibility of labeling each of the component kets with the \mathcal{H} from which it came, A , B or C ,

$$\begin{aligned}
|0\rangle_A \otimes |0\rangle_B \otimes |0\rangle_C &\longleftrightarrow |0\rangle_A |0\rangle_B |0\rangle_C \\
|0\rangle_A \otimes |0\rangle_B \otimes |1\rangle_C &\longleftrightarrow |0\rangle_A |0\rangle_B |1\rangle_C \\
&\text{etc.}
\end{aligned}$$

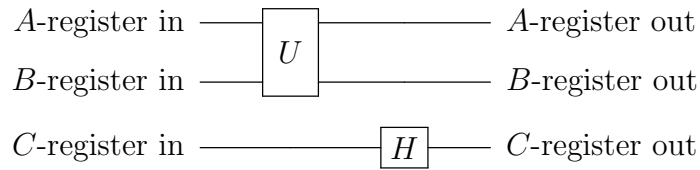
The densest of the notations expresses the CBS ket as an integer from 0 to 7. We reinforce this correspondence and add the *coordinate representation* of each basis ket:

$ 000\rangle$	$ 001\rangle$	$ 010\rangle$	$ 011\rangle$	$ 100\rangle$	$ 101\rangle$	$ 110\rangle$	$ 111\rangle$
$ 0\rangle^3$	$ 1\rangle^3$	$ 2\rangle^3$	$ 3\rangle^3$	$ 4\rangle^3$	$ 5\rangle^3$	$ 6\rangle^3$	$ 7\rangle^3$
$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

Note that that the “exponent 3” is needed mainly in the encoded form, since an integer representation for a CBS does not disclose its tensor order (3) to the reader, while the other representations clearly reveal that the context is three-qubits.

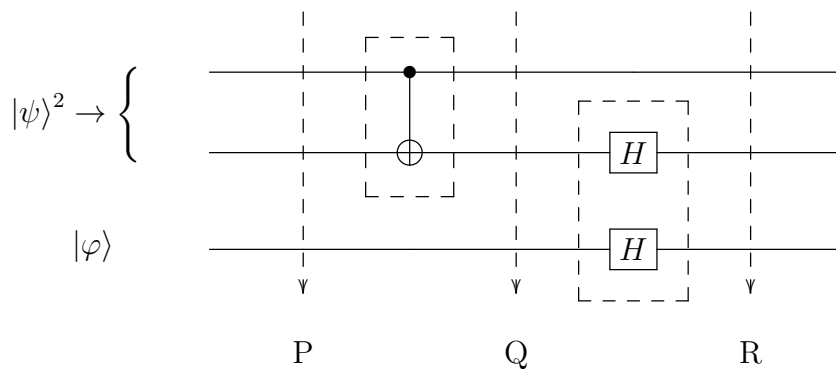
The Channel Labels. We will use the same labeling scheme as before, but more input lines means more labels. For three lines, we would name the registers A , B and

C , as in the hypothetical circuit



Working with Three Qubits

As I mentioned in the introduction, the current regime allows three or more inputs as long as we only apply operators to two at-a-time. Let's look at a circuit that meets that condition.



This circuit is receiving an *order-3 tensor* at its inputs. The first two registers, A and B , get a (potentially) entangled bipartite state $|\psi\rangle^2$ and the third, C , gets a single qubit, $|\varphi\rangle$. We analyze the circuit at the three access points, P , Q and R .

A Theoretical Approach. We'll first do an example that is more general than we normally need, but provides a surefire fallback technique if we are having a hard time. We'll give the input states the *most general form*,

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} \quad \text{and} \quad |\varphi\rangle = \begin{pmatrix} \eta \\ \xi \end{pmatrix}$$

Access Point P. The initial tripartite tensor is

$$|\psi\rangle^2 |\varphi\rangle = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} \otimes \begin{pmatrix} \eta \\ \xi \end{pmatrix} = \begin{pmatrix} \alpha\eta \\ \alpha\xi \\ \beta\eta \\ \beta\xi \\ \gamma\eta \\ \gamma\xi \\ \delta\eta \\ \delta\xi \end{pmatrix}$$

Access Point Q. The first gate is a CNOT applied only to the entangled $|\psi\rangle^2$, so the overall effect is just

$$\left(\text{CNOT} \otimes \mathbf{1} \right) (|\psi\rangle^2 |\varphi\rangle) = \left(\text{CNOT} |\psi\rangle^2 \right) \otimes |\varphi\rangle$$

by the rule for applying a separable operator to a separable state. (Although the first two qubits are entangled, when the input is grouped as a tensor product of $\mathcal{H}_{(2)} \otimes \mathcal{H}$, $|\psi\rangle^2 \otimes |\varphi\rangle$ is recognized as a separable second-order tensor.)

Applying the CNOT explicitly, we get

$$\begin{aligned} \left(\text{CNOT} |\psi\rangle^2 \right) \otimes |\varphi\rangle &= \text{CNOT} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} \otimes \begin{pmatrix} \eta \\ \xi \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \delta \\ \gamma \end{pmatrix} \otimes \begin{pmatrix} \eta \\ \xi \end{pmatrix} \\ &= \begin{pmatrix} \alpha\eta \\ \alpha\xi \\ \beta\eta \\ \beta\xi \\ \delta\eta \\ \delta\xi \\ \gamma\eta \\ \gamma\xi \end{pmatrix}. \end{aligned}$$

We needed to multiply out the separable product in preparation for the next phase of the circuit, which appears to operate on the last two registers, B and C .

Access Point R. The final operator is local to the last two registers, and takes the form

$$\mathbf{1} \otimes H^{\otimes 2}$$

Although it feels as though we might be able to take a short cut, the intermediate tripartite state and final operator are sufficiently complicated to warrant treating it as

a fully entangled three-qubit state and just doing the big matrix multiplication.

$$\begin{aligned}
\left[\mathbf{1} \otimes H^{\otimes 2} \right] \begin{pmatrix} \alpha\eta \\ \alpha\xi \\ \beta\eta \\ \beta\xi \\ \delta\eta \\ \delta\xi \\ \gamma\eta \\ \gamma\xi \end{pmatrix} &= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix} \begin{pmatrix} \alpha\eta \\ \alpha\xi \\ \beta\eta \\ \beta\xi \\ \delta\eta \\ \delta\xi \\ \gamma\eta \\ \gamma\xi \end{pmatrix} \\
&= \frac{1}{2} \begin{pmatrix} \alpha\eta + \alpha\xi + \beta\eta + \beta\xi \\ \alpha\eta - \alpha\xi + \beta\eta - \beta\xi \\ \alpha\eta + \alpha\xi - \beta\eta - \beta\xi \\ \alpha\eta - \alpha\xi - \beta\eta + \beta\xi \\ \delta\eta + \delta\xi + \gamma\eta + \gamma\xi \\ \delta\eta - \delta\xi + \gamma\eta - \gamma\xi \\ \delta\eta + \delta\xi - \gamma\eta - \gamma\xi \\ \delta\eta - \delta\xi - \gamma\eta + \gamma\xi \end{pmatrix},
\end{aligned}$$

which could be written as the less compact basis expansion

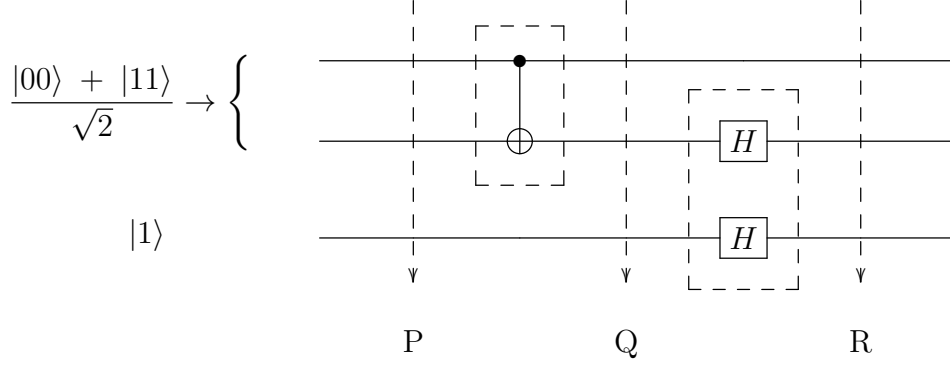
$$\begin{aligned}
&\frac{1}{2} \left[(\alpha\eta + \alpha\xi + \beta\eta + \beta\xi) |000\rangle + (\alpha\eta - \alpha\xi + \beta\eta - \beta\xi) |001\rangle \right. \\
&+ (\alpha\eta + \alpha\xi - \beta\eta - \beta\xi) |010\rangle + (\alpha\eta - \alpha\xi - \beta\eta + \beta\xi) |011\rangle \\
&+ (\delta\eta + \delta\xi + \gamma\eta + \gamma\xi) |100\rangle + (\delta\eta - \delta\xi + \gamma\eta - \gamma\xi) |101\rangle \\
&\left. + (\delta\eta + \delta\xi - \gamma\eta - \gamma\xi) |110\rangle + (\delta\eta - \delta\xi - \gamma\eta + \gamma\xi) |111\rangle \right].
\end{aligned}$$

This isn't very enlightening because the coefficients are so general. But a concrete example shows how the process can be streamlined.

A Practical Approach. Usually we have specific and nicely symmetric input tensors. In this case, let's pretend we know that

$$\begin{aligned}
|\psi\rangle^2 &\equiv \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad \text{and} \\
|\varphi\rangle &\equiv |1\rangle.
\end{aligned}$$

For reference, I'll repeat the circuit for this specific input.



Access Point P. We begin in this state

$$\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}} \right) |1\rangle .$$

Access Point Q. Apply the two-qubit CNOT gate to registers A and B , which has the overall effect of applying $\text{CNOT} \otimes \mathbb{1}$ to the full tripartite tensor.

$$\begin{aligned} \left(\text{CNOT} \otimes \mathbb{1} \right) \left(|\psi\rangle^2 |\varphi\rangle \right) &= \left(\text{CNOT} |\psi\rangle^2 \right) \otimes \left(\mathbb{1} |\varphi\rangle \right) \\ &= \left(\text{CNOT} \left[\frac{|00\rangle + |11\rangle}{\sqrt{2}} \right] \right) \otimes \left(\mathbb{1} |1\rangle \right) \\ &= \left(\frac{\text{CNOT} |00\rangle + \text{CNOT} |11\rangle}{\sqrt{2}} \right) \otimes |1\rangle \\ &= \left(\frac{|00\rangle + |10\rangle}{\sqrt{2}} \right) \otimes |1\rangle \\ &= \frac{|001\rangle + |101\rangle}{\sqrt{2}} , \end{aligned}$$

a state that can be factored to our advantage:

$$\frac{|001\rangle + |101\rangle}{\sqrt{2}} = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) |01\rangle$$

Access Point R. Finally we apply the second two-qubit gate $H^{\otimes 2}$ to the B and C registers, which has the overall effect of applying $\mathbb{1} \otimes H^{\otimes 2}$ to the full tripartite state. The factorization we found makes this an easy separable proposition,

$$\left[\mathbb{1} \otimes H^{\otimes 2} \right] \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |01\rangle \right) = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \otimes H^{\otimes 2} |01\rangle .$$

Referring back to the second order Hadamard on the two states in question, i.e.,

$$H^{\otimes 2} |01\rangle = \frac{|00\rangle - |01\rangle + |10\rangle - |11\rangle}{2},$$

we find that

$$\begin{aligned} & \left[\mathbf{1} \otimes H^{\otimes 2} \right] \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |01\rangle \right) \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|00\rangle - |01\rangle + |10\rangle - |11\rangle}{2} \right) \end{aligned}$$

which can be factored into a fully separable

$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right).$$

On the other hand, if we had been keen enough to remember that $H^{\otimes 2}$ is just the separable $H \otimes H$, which has a special effect on a separable state like $|01\rangle = |0\rangle |1\rangle$, we could have gotten to the factored form faster using

$$\begin{aligned} & \left[\mathbf{1} \otimes H^{\otimes 2} \right] \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |01\rangle \right) = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes [H \otimes H] (|0\rangle |1\rangle) \\ &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes H |0\rangle \otimes H |1\rangle \\ &= \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right). \end{aligned}$$

The moral is, don't worry about picking the wrong approach to a problem. If your math is sound, you'll get to the end zone either way.

Double Checking Our Work. Let's pause to see how this compares with the general formula. Relative to the general $|\psi\rangle^2$ and $|\varphi\rangle$, the specific amplitudes we have in this example are

$$\begin{aligned} \alpha &= \delta = \frac{1}{\sqrt{2}}, & \xi &= 1 \quad \text{and} \\ \beta &= \gamma = \eta = 0 \end{aligned}$$

which causes the general expression

$$\begin{aligned} & \frac{1}{2} \left[(\alpha\eta + \alpha\xi + \beta\eta + \beta\xi) |000\rangle + (\alpha\eta - \alpha\xi + \beta\eta - \beta\xi) |001\rangle \right. \\ & + (\alpha\eta + \alpha\xi - \beta\eta - \beta\xi) |010\rangle + (\alpha\eta - \alpha\xi - \beta\eta + \beta\xi) |011\rangle \\ & + (\delta\eta + \delta\xi + \gamma\eta + \gamma\xi) |100\rangle + (\delta\eta - \delta\xi + \gamma\eta - \gamma\xi) |101\rangle \\ & \left. + (\delta\eta + \delta\xi - \gamma\eta - \gamma\xi) |110\rangle + (\delta\eta - \delta\xi - \gamma\eta + \gamma\xi) |111\rangle \right] \end{aligned}$$

to reduce to

$$\begin{aligned}
& \frac{1}{2} \left[(\alpha\xi) |000\rangle + (-\alpha\xi) |001\rangle + (\alpha\xi) |010\rangle + (-\alpha\xi) |011\rangle \right. \\
& \quad \left. + (\delta\xi) |100\rangle + (-\delta\xi) |101\rangle + (\delta\xi) |110\rangle + (-\delta\xi) |111\rangle \right] \\
&= \frac{1}{2} \left[\left(\frac{1}{\sqrt{2}} |000\rangle - \frac{1}{\sqrt{2}} |001\rangle + \frac{1}{\sqrt{2}} |010\rangle - \frac{1}{\sqrt{2}} |011\rangle \right) \right. \\
& \quad \left. + \frac{1}{\sqrt{2}} |100\rangle - \frac{1}{\sqrt{2}} |101\rangle + \frac{1}{\sqrt{2}} |110\rangle - \frac{1}{\sqrt{2}} |111\rangle \right].
\end{aligned}$$

A short multiplication reveals this to be the answer we got for the input $\left(\frac{|00\rangle + |11\rangle}{\sqrt{2}} \right) |1\rangle$, without the general formula.

[**Exercise.** Verify that this is equal to the directly computed output state. **Hint:** If you start with the first state we got, prior to the factorization, there is less to do.]

[**Exercise.** This had better be a normalized state as we started with unit vectors and applied unitary gates. Confirm it.]

The Born Rule for Tripartite Systems

The Born rule can be generalized to any dimension and stated in many ways. For now, let's state the rule for an *order-three* Hilbert space with registers A , B and C , and in a way that favors factoring out the AB -registers.

Trait #15' (Born Rule for Tripartite States): If we have a tripartite state that can be expressed as the sum of four terms

$$|\varphi\rangle^3 = |0\rangle_{AB}^2 |\psi_0\rangle_C + |1\rangle_{AB}^2 |\psi_1\rangle_C + |2\rangle_{AB}^2 |\psi_2\rangle_C + |3\rangle_{AB}^2 |\psi_3\rangle_C ,$$

each of which is the product of a distinct CBS ket for $\mathcal{H}_A \otimes \mathcal{H}_B$ and some general first order (typically un-normalized) ket in the space \mathcal{H}_C ,

$$|k\rangle_{AB}^2 |\psi_k\rangle_C ,$$

then if we measure the first two registers, thus forcing their collapse into one of the four basis states,

$$\{ |0\rangle_{AB}^2 , |1\rangle_{AB}^2 , |2\rangle_{AB}^2 , |3\rangle_{AB}^2 \} ,$$

the C register will be left in a *normalized* state associated with the measured CBS

ket. In other words,

$$\begin{aligned}
A \otimes B \searrow |0\rangle^2 &\implies C \searrow \frac{|\psi_0\rangle}{\sqrt{\langle\psi_0|\psi_0\rangle}}, \\
A \otimes B \searrow |1\rangle^2 &\implies C \searrow \frac{|\psi_1\rangle}{\sqrt{\langle\psi_1|\psi_1\rangle}}, \\
A \otimes B \searrow |2\rangle^2 &\implies C \searrow \frac{|\psi_2\rangle}{\sqrt{\langle\psi_2|\psi_2\rangle}} \quad \text{and} \\
A \otimes B \searrow |3\rangle^2 &\implies C \searrow \frac{|\psi_3\rangle}{\sqrt{\langle\psi_3|\psi_3\rangle}}.
\end{aligned}$$

Note the prime (') in the *tripartite* **Trait #15'**, to distinguish this from the unprimed **Trait #15** for *bipartite* systems. Also, note that I suppressed the state-space subscript labels A , B and C which are understood by context.

We'll use this form of the Born rule for *quantum teleportation* in our next lecture.

11.8.4 End of Lesson

This was a landmark week, incorporating all the basic ideas of quantum computing. We are now ready to tackle our first quantum algorithms.

Chapter 12

First Quantum Algorithms

12.1 Three Algorithms

This week we will see how *quantum* circuits and their associated algorithms can be used to achieve results impossible in the world of *classical* digital computing. We'll cover three topics:

- Superdense Coding
- Quantum Teleportation
- Deutsch's Algorithm

The first two demonstrate quantum communication possibilities, and the third provides a learning framework for many quantum algorithms which execute “faster” (in a sense) than their classical counterparts.

12.2 Superdense Coding

12.2.1 Sending Information by Qubit

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle ,$$

seems like it holds an infinite amount of information. After all, α and β are complex numbers, and even though you can't choose them arbitrarily ($|\alpha|^2 + |\beta|^2$ must be 1), the mere fact that α can be any complex number whose magnitude is ≤ 1 means it could be an unending sequence of never-repeating digits, like 0.4193980022903... . If a *sender* \mathcal{A} (an assistant quantum researcher named “Alice”) could pack $|\psi\rangle$ with that α (and compatible β) and send it off in the form of a single photon to a *receiver* \mathcal{B} (another helper whose name is “Bob”) a few time zones away, \mathcal{A} would be sending an infinite string of digits to \mathcal{B} encoded in that one sub-atomic particle.

The problem, of course, arises when the when \mathcal{B} tries to look inside the received state. All he can do is measure it once and *only* once (**Trait #7**, the *fifth postulate* of QM), at which point he gets a “0” or “1” and both α and β are wiped off the face of the Earth. That one measurement tells \mathcal{B} very little.

[**Exercise.** But it does tell him *something*. What?]

In short, to communicate $|\alpha|$, \mathcal{A} would have to prepare and send an infinite number of identical states, then \mathcal{B} would have to receive, test and record them. Only *then* would \mathcal{B} know $|\alpha|$ and $|\beta|$ (although neither α nor β). This is no better than classical communication.

We have to lower our sights.

A Most Modest Wish

We are wondering what information, exactly, \mathcal{A} (Alice) can send \mathcal{B} (Bob) in the form of a single qubit. We know it’s not infinite. At the other extreme is the most modest super-classical capability we could hope for: *two classical bits* for the price of one. I think that we need no lecture to affirm the claim that, in order to send a two-digit binary message, i.e., one of

$$\begin{aligned} 0 &= \text{“}00\text{”} , \\ 1 &= \text{“}01\text{”} , \\ 2 &= \text{“}10\text{”} , \\ 3 &= \text{“}11\text{”} , \end{aligned}$$

we would have to send more than one classical bit – we’d need two. Can we pack at least this meager amount of classical information into one qubit with the confidence that \mathcal{B} would be able read the message?

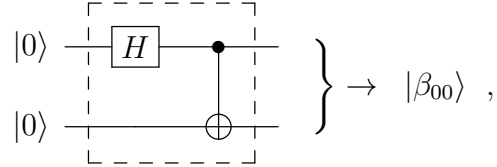
12.2.2 The Superdense Coding Algorithm

We can, and to do so, we use the four *Bell states* (*EPR pairs*) from the *two qubit lecture*,

$$\begin{aligned} |\beta_{00}\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} , \\ |\beta_{01}\rangle &= \frac{|01\rangle + |10\rangle}{\sqrt{2}} , \\ |\beta_{10}\rangle &= \frac{|00\rangle - |11\rangle}{\sqrt{2}} \quad \text{and} \\ |\beta_{11}\rangle &= \frac{|01\rangle - |10\rangle}{\sqrt{2}} . \end{aligned}$$

Building the Communication Equipment

\mathcal{A} and \mathcal{B} prepare the state $|\beta_{00}\rangle$. (This can be done, for example, by sending a $|00\rangle$ through the BELL gate,



as we learned.) \mathcal{A} takes the A register of the entangled state $|\beta_{00}\rangle$ and \mathcal{B} takes the B register. \mathcal{B} gets on a plane, placing his qubit in the overhead bin and travels a few time zones away. This can all be done long before the classical two-bit message is selected by \mathcal{A} , but it has to be done. It can even be done by a third party who sends the first qubit of this EPR pair to \mathcal{A} and the second to \mathcal{B} .

Defense of Your Objection. The sharing of this qubit does not constitute sending more than one qubit of information (the phase yet to come), since it is analogous to establishing a radio transmission protocol or message envelope, which would have to be done even with classical bits. It is part of the equipment that \mathcal{B} and \mathcal{A} use to communicate data, not the data itself.

Notation. In the few cases where we need it (and one is coming up), let's build some notation. When a potentially entangled two-qubit state is separated physically into two registers or by two observers, we need a way to talk about each individual qubit. We'll use

$$|\psi\rangle^2 \Big|_A \quad \text{for the } A \text{ register (or } \mathcal{A}\text{'s) qubit, and}$$

$$|\psi\rangle^2 \Big|_B \quad \text{for the } B \text{ register (or } \mathcal{B}\text{'s) qubit.}$$

Note that, unless $|\psi\rangle^2$ happens to be separable – and $|\beta_{00}\rangle$ is clearly *not* – we will be faced with the reality that

$$|\psi\rangle^2 \neq |\psi\rangle^2 \Big|_A \otimes |\psi\rangle^2 \Big|_B .$$

[**Note.** This does *not* mean that the A register and B register can't exist in physically independent locations and be measured or processed independently by different observers. As we learned, one observer can modify or measure either qubit individually. What it *does* mean is that the two registers are entangled so modifying or measuring one *will affect the other*. Together they form a single state.]

With this language, the construction and distribution of each half of the entangled $|\beta_{00}\rangle$ to \mathcal{A} and \mathcal{B} can be symbolized by

$$|\beta_{00}\rangle \longrightarrow \left\{ \begin{array}{l} \longrightarrow |\beta_{00}\rangle \Big|_A \text{ goes to } \mathcal{A} \\ \longrightarrow |\beta_{00}\rangle \Big|_B \text{ goes to } \mathcal{B} \end{array} \right.$$

\mathcal{A} Encodes the Message

When \mathcal{A} is ready to ship one of the four bit strings to \mathcal{B} , she decides – or is informed – which it is to be and takes the following action. She submits her half of the bipartite state to one of four local gates according to the table

\mathcal{A} to Send	\mathcal{A} Applies	Equivalent Binary Gate	New Bipartite State
"00"	(nothing)	$\mathbb{1} \otimes \mathbb{1}$	$ \beta_{00}\rangle$
"01"	X	$X \otimes \mathbb{1}$	$ \beta_{01}\rangle$
"10"	Z	$Z \otimes \mathbb{1}$	$ \beta_{10}\rangle$
"11"	iY	$iY \otimes \mathbb{1}$	$ \beta_{11}\rangle$

(The " $\otimes \mathbb{1}$ "s in the *equivalent binary gate* column reflect the fact that \mathcal{B} is not touching his half of $|\beta_{00}\rangle$, which is effectively the identity operation as far as the B register is concerned.)

And how do we know that the far right column is the result of \mathcal{A} 's local operation? We apply the relevant matrix to $|\beta_{00}\rangle$ and read off the answer (see section *Four Bell States from One* in the *two qubit* lecture).

Compatibility Note. Most authors ask \mathcal{A} to apply Z if she wants to encode "01" and X if she wants to encode "10", but doing so results in the state $|\beta_{10}\rangle$ for "01" and $|\beta_{01}\rangle$ for "10", not a very nice match-up and is why I chose to present the algorithm with those two gates swapped. Of course, it really doesn't matter *which* of the four operators \mathcal{A} uses for each encoding, as long as \mathcal{B} uses the same correspondence to decode.

If we encapsulate the four possible operators into one symbol, SD (for ***Super Dense***), which takes on the proper operation based on the message to be encoded, \mathcal{A} 's job is to apply the local circuit,

$$|\beta_{00}\rangle \Big|_A \quad \text{---} \boxed{SD} \text{---} \quad \left[(SD \otimes \mathbb{1}) |\beta_{00}\rangle \right] \Big|_A$$

Notice that to describe \mathcal{A} 's half of the output state, we need to first show the full effect of the bipartite operator and only then restrict attention to \mathcal{A} 's qubit. We cannot express it as a function of \mathcal{A} 's input, $|\beta_{00}\rangle \Big|_A$, alone.

\mathcal{A} Sends Her Qubit

The message is now encoded in the bipartite state, but for \mathcal{B} to decode it, he needs both qubits. \mathcal{A} now sends her qubit to \mathcal{B} .

$$\mathcal{A} \xrightarrow{\left[(SD \otimes \mathbb{1}) |\beta_{00}\rangle \right] \Big|_A} \mathcal{B}$$

\mathcal{B} Measures Along the Bell Basis to Read the Message

When he gets \mathcal{A} 's qubit, \mathcal{B} has the complete *entangled* bipartite state

$$\left. \begin{array}{l} \left[(SD \otimes \mathbb{1}) |\beta_{00}\rangle \right] \Big|_A \\ \left[(SD \otimes \mathbb{1}) |\beta_{00}\rangle \right] \Big|_B \end{array} \right\} (SD \otimes \mathbb{1}) |\beta_{00}\rangle ,$$

so he can now measure both qubits to determine which of the four Bell states he has. Once that's done he reads the earlier table from right-to-left to recover the classical two-bit message.

Refresher: Measuring Along the Bell Basis. Since this is the first time we will have applied it in an algorithm, I'll summarize one way that \mathcal{B} can measure his entangled state along the Bell basis. When studying *two qubit logic* we learned that to measure a bipartite state along a non-standard basis (call it \mathcal{C}), we find the *binary* operator that takes the z -basis to the other basis, call it S , and use S^\dagger prior to measurement:

$$(\text{some } \mathcal{C}\text{-basis state}) \longrightarrow \left\{ \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline S^\dagger \\ \hline \end{array} \begin{array}{c} \text{---} \\ \text{---} \end{array} \right\} \text{measure along } z\text{-basis}$$

In this situation, S is BELL, whose adjoint we computed in that same lecture,

$$\begin{array}{|c|} \hline \text{BELL}^\dagger \\ \hline \end{array} = \begin{array}{|c|} \hline \begin{array}{c} \bullet \\ | \\ \oplus \end{array} \begin{array}{|c|} \hline H \\ \hline \end{array} \\ \hline \end{array} .$$

Adding the *measurement symbols* (the “meters”) along the z -basis, circuit becomes

$$(\text{one of the four BELL states}) \longrightarrow \left\{ \begin{array}{c} \text{---} \\ \text{---} \end{array} \begin{array}{|c|} \hline \text{BELL}^\dagger \\ \hline \end{array} \begin{array}{c} \boxed{\diagup} \\ \boxed{\diagdown} \end{array} \right\} .$$

In terms of matrices, \mathcal{B} subjects his two-qubit state to the matrix for BELL^\dagger (also computed last time),

$$\text{BELL}^\dagger = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}.$$

Bob’s Action and Conclusion. Post-processing with the BELL^\dagger gate turns the four Bell states into four z -CBS kets; if \mathcal{B} follows that gate with a z -basis measurement and sees a “01”, he will conclude that he had received the Bell state $|\beta_{01}\rangle$ from \mathcal{A} , and likewise for the other states. So his role, after receiving the qubit sent by \mathcal{A} , is to

1. apply BELL^\dagger to his two qubits, and
2. read the encoded message according to his results using the table

\mathcal{B} measures	\mathcal{B} Concludes \mathcal{A} ’s Message to him is
“00”	“00”
“01”	“01”
“10”	“10”
“11”	“11”

In other words, the application of the BELL^\dagger gate allowed \mathcal{B} to interpret his z -basis measurement reading “ xy ” as the message, itself.

The following exercise should help crystallize the algorithm.

[**Exercise.** Assume \mathcal{A} wants to send the message “11.”

- i) Apply $iY \otimes \mathbb{1}$ to $|\beta_{00}\rangle$ and confirm that you get $|\beta_{11}\rangle$ out.
- ii) Multiply the 4×4 matrix for BELL^\dagger by the 4×1 state vector for $|\beta_{11}\rangle$ to show that \mathcal{B} recovers the message “11.”

A Circuit Representation of Superdense Coding

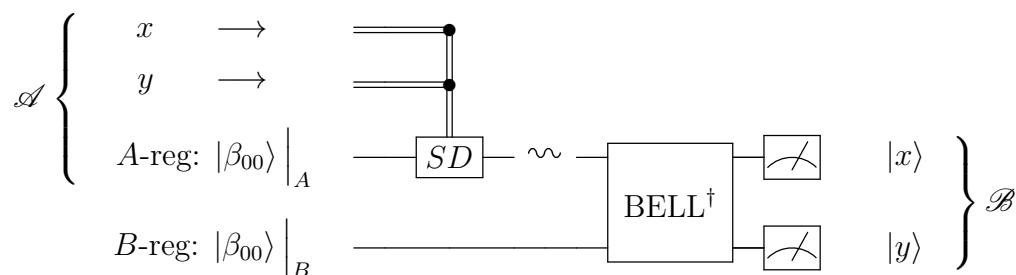
We can get a circuit for the overall superdense coding algorithm by adding some new notation.

Classical Wires. Double lines (=) indicate the transfer of *classical bits*. We use them to move one or more ordinary digits within a circuit.

Decisions Based on Classical Bits. We insert a dot symbol, \bullet , into a classical line to indicate a general *controlled operation*, based on the content of that classical data ([1] means apply an operator, [0] means don't).

Noiseless Transmission Between Communicators. To indicate the (typically radio) transmission of either quantum or classical data between sender and receiver, we use the wavy line, \sim .

With this notation, the superdense coding circuit can be expressed as:



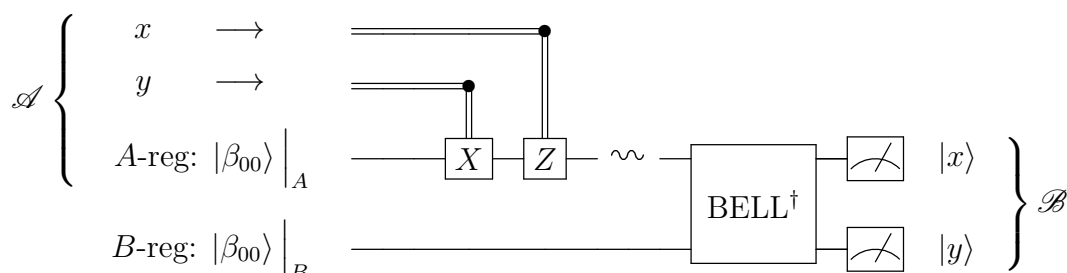
The notation tells the story. \mathcal{A} uses her two-bit classical message “ xy ” (traveling on the *double lines*) to control (*filled circles*) which of the four operations ($SD = \mathbf{1}$, X , Z or iY) she will apply to her qubit. After sending her qubit to \mathcal{B} , \mathcal{B} measures both qubits along the *Bell basis* to recover the message “ xy ” now sitting in the output registers in natural z -basis form $|x\rangle|y\rangle$.

[**Exercise.** Measurement involves collapse and uncertainty. Why is \mathcal{B} so certain that his two measurements will always result in a true reproduction of the message “ xy ” sent by \mathcal{A} ? **Hint:** For each of the four possible messages, what bipartite state is he holding at the moment of measurement?]

This can actually be tightened up. You’ve seen several unary operator identities in the *single qubit lecture*, one of which was $XZ = -iY$. A slight revision of this (verify as an exercise) is

$$ZX = iY,$$

which enables us to define the elusive SD operation: we place a controlled- X gate and controlled- Z gate in the A -channel under \mathcal{A} ’s supervision. Each gate is controlled by one of the two classical bits in her message. They work just like a quantum Controlled- U gate, only simpler: if the classical control bit is 1, the target operation is applied, if the bit is 0, it is not.



For example, if both bits are 1, both gates get applied and result in the desired behavior: “11” $\Rightarrow ZX \cong iY$.

[**Exercise.** Remind us why the gates X and Z appear reversed in the circuit relative to the algebraic identity $iY = ZX$.]

The Significance of Superdense Coding

This technique may not seem tremendously applicable considering its unimpressive 2-bit to 1-bit compression, but consider sending a large classical message, even one that is already as densely compressed as classical logic will allow. This is a 2-to-1 improvement over the best classical technique when applied to the output of classical compression. The fact that we have to send lots of entangled Bell states before our message takes nothing away from our ability to send *information* in half the time (or space) as before.

12.3 Quantum Teleportation

We re-enlist the help of our two most excellent researchers, Alice and Bob, and continue to refer to them by their code names \mathcal{A} and \mathcal{B} .

In *superdense coding* \mathcal{A} sent \mathcal{B} one qubit, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, in order to reconstruct two classical bits. *Quantum teleportation* is the mirror image of this process. \mathcal{A} wants to send \mathcal{B} the qubit, $|\psi\rangle$, by sending him just two *classical* bits of information.

Giving Teleportation Context

You might ask why she doesn’t simply send \mathcal{B} the one qubit and be done with it. Why be so indirect and translate the quantum information into classical bits? There are many answers, two of which I think are important.

1. As a practical matter, it may be impossible, or at least difficult, for \mathcal{A} to send \mathcal{B} qubit information due to its instability and/or expense. By contrast, humans have engineered highly reliable and economical classical communication channels. Sending two bits is child’s play.
2. Sending the original qubit rather than two classical bits is somewhat beside the point. The very fact \mathcal{A} can get the infinitely precise data embedded in the continuous scalars α and β by sending something as crude as an integer from 0 to 3 should come as unexpectedly marvelous news, and we want to know why and how this can be done.

Caveats

There is the usual caveat. Just because \mathcal{B} gets the qubit doesn't mean he can *know* what it is. He can no more examine its basis coefficients than \mathcal{A} (or anyone in her local lab who didn't already know their values) could. What we are doing here is getting the qubit over to \mathcal{B} 's lab so he can use it on his end for any purpose that \mathcal{A} could have (before the teleportation).

And then there's the unusual caveat. In the process of executing the teleportation, \mathcal{A} loses her copy of $|\psi\rangle$. We'll see why as we describe the algorithm.

An Application of the Born Rule

I like to take any reasonable opportunity to restate important tools so as to establish them in your mind. The *Born rule* is so frequently used that it warrants such a review before we apply it to teleportation.

The *Born rule for 3-qubit systems* (**Trait #15'**) tells us (and I am paraphrasing with equally precise expressions) that if we have a tripartite state which can be expressed as the sum of four terms (where the first factors of each term are AB -basis kets):

$$|\varphi\rangle^3 = |0\rangle_{AB}^2 |\psi_0\rangle_C + |1\rangle_{AB}^2 |\psi_1\rangle_C + |2\rangle_{AB}^2 |\psi_2\rangle_C + |3\rangle_{AB}^2 |\psi_3\rangle_C ,$$

then an AB -register measurement along the natural basis will force the corresponding C -register collapse according to

$$\begin{aligned} A \otimes B \searrow |0\rangle_{AB}^2 &\implies C \searrow \frac{|\psi_0\rangle_C}{\| |\psi_0\rangle_C \|} , \\ A \otimes B \searrow |1\rangle_{AB}^2 &\implies C \searrow \frac{|\psi_1\rangle_C}{\| |\psi_1\rangle_C \|} , \\ &\text{etc.} \end{aligned}$$

There are two consequences that will prepare us for understanding quantum teleportation as well as anticipating other algorithms that might employ this special technique.

Consequence #1. The rule works for *any* orthonormal basis in channels A and B , not just the natural basis. Whichever basis we choose for the first two registers A and B , it is *along that basis* that we must make our two-qubit measurements. So, if we use the Bell basis, $\{ |\beta_{jk}\rangle \}$, then a state in the form

$$\begin{aligned} |\varphi\rangle^3 = |\beta_{00}\rangle_{AB} |\psi_{00}\rangle_C + |\beta_{01}\rangle_{AB} |\psi_{01}\rangle_C \\ + |\beta_{10}\rangle_{AB} |\psi_{10}\rangle_C + |\beta_{11}\rangle_{AB} |\psi_{11}\rangle_C , \end{aligned}$$

when measured along *that* basis will force the corresponding C -register collapse according to

$$\begin{aligned} A \otimes B \searrow |\beta_{00}\rangle_{AB} &\implies C \searrow \frac{|\psi_{00}\rangle_C}{\| |\psi_{00}\rangle_C \|}, \\ A \otimes B \searrow |\beta_{01}\rangle_{AB} &\implies C \searrow \frac{|\psi_{01}\rangle_C}{\| |\psi_{01}\rangle_C \|}, \end{aligned}$$

etc.

This follows from the **Trait #7**, *Post-Measurement Collapse*, which tells us that AB will collapse to one of the four CBS states – regardless of which CBS we use – forcing C into the state that is glued to its partner in the above expansion.

The division by each $\| |\psi_{jk}\rangle \|$ (or, if you prefer, $\sqrt{\langle \psi_{jk} | \psi_{jk} \rangle}$) is necessary because the overall tripartite state, $|\varphi\rangle^3$ can only be normalized when the $|\psi_{jk}\rangle$ have non-unit (in fact < 1) lengths.

[Exercise. We already know that $|\beta_{jk}\rangle$ are four normalized CBS kets. Show that if the $|\psi_{jk}\rangle$ were normal vectors in \mathcal{H}_C , then $|\varphi\rangle^3$ would not be a normal vector. **Hint:** Write down ${}^3\langle \varphi | \varphi \rangle^3$ and apply orthonormality of the Bell states.]

Consequence #2. If we know that the four general states, $|\psi_{jk}\rangle$, are just four variations of a single known state, we may be able to glean even more specific information about the collapsed C -register. To cite the example needed today, say we know that all four $|\psi_{jk}\rangle$ use the same two scalar coordinates, α and β , only in slightly different combinations,

$$\begin{aligned} |\varphi\rangle^3 &= |\beta_{00}\rangle_{AB} \left(\frac{\alpha |0\rangle_C + \beta |1\rangle_C}{2} \right) + |\beta_{01}\rangle_{AB} \left(\frac{\beta |0\rangle_C + \alpha |1\rangle_C}{2} \right) \\ &+ |\beta_{10}\rangle_{AB} \left(\frac{\alpha |0\rangle_C - \beta |1\rangle_C}{2} \right) + |\beta_{11}\rangle_{AB} \left(\frac{-\beta |0\rangle_C + \alpha |1\rangle_C}{2} \right). \end{aligned}$$

(Each denominator 2 is needed to produce a normal state $|\varphi\rangle^3$; we cannot absorb it into α and β , as those scalars are fixed by the normalized $|\psi\rangle$ to be teleported. However, the *Born rule* tells us that the collapse of the C -register will get rid of this factor, leaving only one of the four numerators in the C -register.) Such a happy state-of-affairs will allow us to convert any of the four collapsed states in the C -register to the one state,

$$\alpha |0\rangle + \beta |1\rangle$$

by mere application of a simple unary operator. For example, if we find that AB collapses to $|\beta_{00}\rangle$ (by reading a “00” on our measuring apparatus), then C will have *already* collapsed to the state $\alpha |0\rangle + \beta |1\rangle$. Or, if AB collapses to $|\beta_{11}\rangle$ (meter reads “11”), then we apply the operator iY to C to recover $\alpha |0\rangle + \beta |1\rangle$, because

$$iY(-\beta |0\rangle + \alpha |1\rangle) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} -\beta \\ \alpha \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

You’ll refer back to these two facts as we unroll the quantum teleportation algorithm.

12.3.1 The Quantum Teleportation Algorithm

We continue to exploit the *EPR pairs* which I list again for quick reference:

$$\begin{aligned} |\beta_{00}\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \\ |\beta_{01}\rangle &= \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \\ |\beta_{10}\rangle &= \frac{|00\rangle - |11\rangle}{\sqrt{2}} \quad \text{and} \\ |\beta_{11}\rangle &= \frac{|01\rangle - |10\rangle}{\sqrt{2}}. \end{aligned}$$

Building the Communication Equipment

\mathcal{A} and \mathcal{B} prepare – and each get one qubit of – the bipartite state $|\beta_{00}\rangle$.

\mathcal{A} Produces a General Qubit for Teleportation

When \mathcal{A} is ready to *teleport* a qubit to \mathcal{B} , she manually produces – or is given – any \mathcal{H} -space qubit

$$|\psi\rangle = |\psi\rangle_C \equiv \alpha |0\rangle_C + \beta |1\rangle_C.$$

The subscript C indicates that we have a qubit separate from the two entangled qubits already created and distributed to our two “messengers,” a qubit which lives in its own space with its own (natural) CBS basis $\{|0\rangle_C, |1\rangle_C\}$.

By tradition for this algorithm, we place the C -channel *above* the A/B -Channels:

$$\begin{aligned} |\psi\rangle_C &\longrightarrow \text{register } C \\ |\beta_{00}\rangle_{AB} &\longrightarrow \begin{cases} \text{register } A \\ \text{register } B \end{cases} \end{aligned}$$

The algebraic state of the entire system, then, starts out to be

$$|\varphi\rangle^3 = |\psi\rangle_C |\beta_{00}\rangle_{AB} = \left(\alpha |0\rangle_C + \beta |1\rangle_C \right) \otimes \left(\frac{|0\rangle_A |0\rangle_B + |1\rangle_A |1\rangle_B}{\sqrt{2}} \right).$$

The Plan

\mathcal{A} starts by *teleporting* a qubit to \mathcal{B} . No information is actually *sent*. Rather, \mathcal{A} does something to her entangled qubit, $|\beta_{00}\rangle|_A$, which instantaneously modifies \mathcal{B} ’s qubit, $|\beta_{00}\rangle|_B$, faster than the speed of light. This is the meaning of the word *teleportation*.

She then follows that up by taking a measurement of her two qubits, getting two classical bits of information – the outcomes of the two register readings. Finally, she sends the result of that measurement as a classical two-bit message to \mathcal{B} (sorry, we have to obey the Einstein’s speed limit for this part). \mathcal{B} will use the two classical bits he receives from Alice to tweak his qubit (already modified by Alice’s teleportation) into the desired state, $|\psi\rangle$.

\mathcal{A} Expresses the System State in the Bell Basis (No Action Yet)

In the z -basis, all the information about $|\psi\rangle$ is contained in \mathcal{A} ’s C -register. She wants to move that information over to \mathcal{B} ’s B -register. Before she even does anything physical, she can accomplish most of the hard work by just *rearranging* the tripartite state $|\varphi\rangle^3$ in a factored form expanded along a CA Bell-basis rather than a CA z -basis. In other words, we’d like to see

$$|\varphi\rangle^3 \stackrel{?}{=} |\beta_{00}\rangle_{CA} |\psi_{00}\rangle_B + |\beta_{01}\rangle_{CA} |\psi_{01}\rangle_B \\ + |\beta_{10}\rangle_{CA} |\psi_{10}\rangle_B + |\beta_{11}\rangle_{CA} |\psi_{11}\rangle_B ,$$

where the $|\psi_{jk}\rangle_B$ are (for the moment) four unknown B -channel states. We can only arrive at such a CA Bell basis expression if the two channels A and C become entangled, which they are not, initially. We’ll get to that.

In our short review of the Born rule, above, I gave you a preview of the actual expression we’ll need. This is what we would like/wish/hope for:

$$|\varphi\rangle^3 \stackrel{?}{=} |\beta_{00}\rangle_{CA} \left(\frac{\alpha |0\rangle_B + \beta |1\rangle_B}{2} \right) + |\beta_{01}\rangle_{CA} \left(\frac{\beta |0\rangle_B + \alpha |1\rangle_B}{2} \right) \\ + |\beta_{10}\rangle_{CA} \left(\frac{\alpha |0\rangle_B - \beta |1\rangle_B}{2} \right) + |\beta_{11}\rangle_{CA} \left(\frac{-\beta |0\rangle_B + \alpha |1\rangle_B}{2} \right) .$$

Indeed, if we could accomplish that, then \mathcal{A} would only have to measure her two qubits along the Bell basis, forcing a collapse into one of the four Bell states and by the Born rule collapsing \mathcal{B} ’s register into the one of his four matching states. A glance at the above expressions reveals that this gets us 99.99% of the way toward placing $|\psi\rangle$ into \mathcal{B} ’s B -register, i.e., manufacturing $|\psi\rangle_B$, a teleported twin to Alice’s original $|\psi\rangle_A$. We’ll see how \mathcal{B} gets the last .01% of the way there, but first, we prove the validity of the hoped-for expansion.

We begin with the desired expression and reduce it to the expression we know to be our actual starting point, $|\varphi\rangle^3$. (**Warning:** After the first expression, I’ll be

dropping the state-space subscripts $A/B/C$ and letting position do the job.)

$$\begin{aligned}
& |\beta_{00}\rangle_{CA} \left(\frac{\alpha|0\rangle_B + \beta|1\rangle_B}{2} \right) + |\beta_{01}\rangle_{CA} \left(\frac{\beta|0\rangle_B + \alpha|1\rangle_B}{2} \right) \\
& + |\beta_{10}\rangle_{CA} \left(\frac{\alpha|0\rangle_B - \beta|1\rangle_B}{2} \right) + |\beta_{11}\rangle_{CA} \left(\frac{-\beta|0\rangle_B + \alpha|1\rangle_B}{2} \right) \\
& = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \left(\frac{\alpha|0\rangle + \beta|1\rangle}{2} \right) + \frac{|01\rangle + |10\rangle}{\sqrt{2}} \left(\frac{\beta|0\rangle + \alpha|1\rangle}{2} \right) \\
& + \frac{|00\rangle - |11\rangle}{\sqrt{2}} \left(\frac{\alpha|0\rangle - \beta|1\rangle}{2} \right) + \frac{|01\rangle - |10\rangle}{\sqrt{2}} \left(\frac{-\beta|0\rangle + \alpha|1\rangle}{2} \right) \\
& = \frac{1}{2\sqrt{2}} \left(\begin{aligned} & \alpha|000\rangle + \alpha|110\rangle + \beta|001\rangle + \beta|111\rangle \\ & + \beta|010\rangle + \beta|100\rangle + \alpha|011\rangle + \alpha|101\rangle \\ & + \alpha|000\rangle - \alpha|110\rangle - \beta|001\rangle + \beta|111\rangle \\ & - \beta|010\rangle + \beta|100\rangle + \alpha|011\rangle - \alpha|101\rangle \end{aligned} \right).
\end{aligned}$$

Half the terms cancel and the other half reinforce to give

$$\begin{aligned}
& |\beta_{00}\rangle_{CA} \left(\frac{\alpha|0\rangle_B + \beta|1\rangle_B}{2} \right) + |\beta_{01}\rangle_{CA} \left(\frac{\beta|0\rangle_B + \alpha|1\rangle_B}{2} \right) \\
& + |\beta_{10}\rangle_{CA} \left(\frac{\alpha|0\rangle_B - \beta|1\rangle_B}{2} \right) + |\beta_{11}\rangle_{CA} \left(\frac{-\beta|0\rangle_B + \alpha|1\rangle_B}{2} \right) \\
& = \frac{1}{2\sqrt{2}} \left(2\alpha|000\rangle + 2\beta|100\rangle + 2\alpha|011\rangle + 2\beta|111\rangle \right) \\
& = \frac{1}{\sqrt{2}} \left(\alpha|0\rangle + \beta|1\rangle \right) |00\rangle + \frac{1}{\sqrt{2}} \left(\alpha|0\rangle + \beta|1\rangle \right) |11\rangle \\
& = (\alpha|0\rangle + \beta|1\rangle) \left(\frac{|00\rangle + |11\rangle}{\sqrt{2}} \right) \\
& = |\psi\rangle_C |\beta_{00}\rangle_{AB},
\end{aligned}$$

a happy ending. This was \mathcal{A} 's original formulation of the tripartite state in terms of the z -basis, so it is indeed the same as the Bell expansion we were hoping for.

Next, we take action to make use of this alternate formulation of our system state. (Remember, we haven't actually *done anything* yet.)

\mathcal{A} Measures the Registers CA Along the Bell Basis

The last derivation demonstrated that

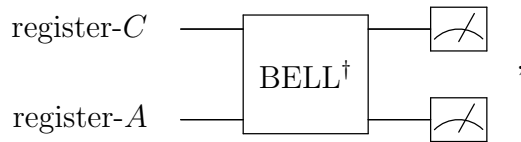
$$\begin{aligned}
 & |\psi\rangle_C |\beta_{00}\rangle_{AB} \\
 = & |\beta_{00}\rangle_{CA} \left(\frac{\alpha |0\rangle_B + \beta |1\rangle_B}{2} \right) + |\beta_{01}\rangle_{CA} \left(\frac{\beta |0\rangle_B + \alpha |1\rangle_B}{2} \right) \\
 & + |\beta_{10}\rangle_{CA} \left(\frac{\alpha |0\rangle_B - \beta |1\rangle_B}{2} \right) + |\beta_{11}\rangle_{CA} \left(\frac{-\beta |0\rangle_B + \alpha |1\rangle_B}{2} \right),
 \end{aligned}$$

that is, the to-be-teleported $|\psi\rangle$ in Alices's C -register seems to “shows up” (in a modified form) in \mathcal{B} 's B -register without anyone having taken any action – all we did was rearrange the terms. However, this rearrangement *is only valid if \mathcal{A} intends to measure the AC -register along the BELL basis*. Such a measurement, as we have seen, always has two parts.

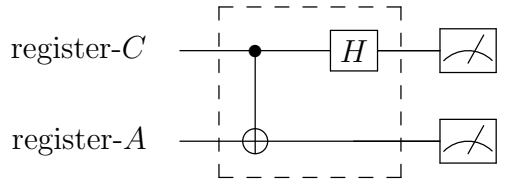
1. \mathcal{A} applies a BELL^\dagger gate to her AC -registers (the operator that takes the non-standard basis to the z -basis).
2. \mathcal{A} measures the resultant AC registers in a standard z -basis.

The first of these two parts, which effects the instantaneous transfer of $|\psi\rangle$ from the A -channel to the B -channel, corresponds to the *teleportation* step of “the plan.” The second part is where \mathcal{A} 's measurement selects one of the four “near”- $|\psi\rangle$ s for \mathcal{B} 's B -register.

The circuit \mathcal{A} needs is



or more explicitly,



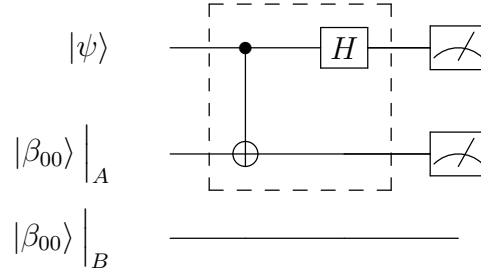
After applying the gate (but before the measurement), the original tripartite state,

$|\varphi\rangle^3$, will be transformed to

$$\begin{aligned}
\left(\text{BELL}^\dagger \otimes \mathbb{1} \right) |\varphi\rangle^3 &= \text{BELL}^\dagger |\beta_{00}\rangle_{CA} \otimes \mathbb{1} \left(\frac{\alpha |0\rangle_B + \beta |1\rangle_B}{2} \right) \\
&+ \text{BELL}^\dagger |\beta_{01}\rangle_{CA} \otimes \mathbb{1} \left(\frac{\beta |0\rangle_B + \alpha |1\rangle_B}{2} \right) \\
&+ \text{BELL}^\dagger |\beta_{10}\rangle_{CA} \otimes \mathbb{1} \left(\frac{\alpha |0\rangle_B - \beta |1\rangle_B}{2} \right) \\
&+ \text{BELL}^\dagger |\beta_{11}\rangle_{CA} \otimes \mathbb{1} \left(\frac{-\beta |0\rangle_B + \alpha |1\rangle_B}{2} \right) \\
&= |00\rangle_{CA} \left(\frac{\alpha |0\rangle_B + \beta |1\rangle_B}{2} \right) + |01\rangle_{CA} \left(\frac{\beta |0\rangle_B + \alpha |1\rangle_B}{2} \right) \\
&+ |10\rangle_{CA} \left(\frac{\alpha |0\rangle_B - \beta |1\rangle_B}{2} \right) + |11\rangle_{CA} \left(\frac{-\beta |0\rangle_B + \alpha |1\rangle_B}{2} \right),
\end{aligned}$$

Now when \mathcal{A} measures her two qubits along the z -basis, she will actually be measuring the pre-gate state along Bell basis. After the measurement only one of the four terms will remain (by collapse) and \mathcal{B} will have a near- $|\psi\rangle$ left in his B -register.

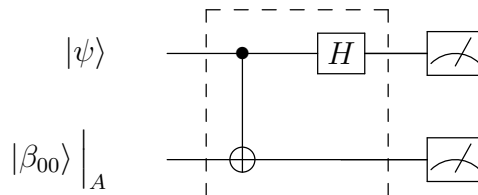
The circuit that describes \mathcal{A} 's local Bell basis measurement with \mathcal{B} 's qubit going along for the ride is



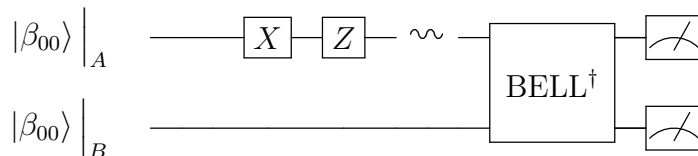
where the final meters are, as always, natural z -basis measurements.

One Qubit of an Entangled Pair

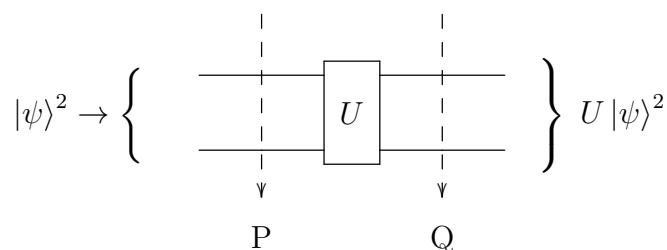
In case you hadn't noticed, twice today we've seen something that might seem to be at odds with our previous lessons. I'm talking about a single, entangled, qubit being fed into one channel of a binary quantum gate, like



or



Earlier, I admonished you to not expect to see – or be allowed to write – a non-separable bipartite state broken into two parts, each going into the individual channels of a binary gate. Rather we need to consider it as a non-separable entity going into both channels at once, as in:



However, the new notation that I have provided today, one half of an entangled qubit,

$$|\psi\rangle^2 \Big|_A \quad \text{for the } A \text{ register (or } \mathcal{A}\text{'s) qubit, and}$$

$$|\psi\rangle^2 \Big|_B \quad \text{for the } B \text{ register (or } \mathcal{B}\text{'s) qubit,.}$$

allows us to write these symbols as individual inputs into either input of a binary quantum gate without violating the cautionary note. Why? Because earlier, the separate inputs we disallowed were individual components of a separable tensor (when no such separable tensor existed). We were saying that you cannot mentally place a tensor symbol, \otimes , between the two individual inputs. Here, the individual symbols are not elements in the two component spaces, and there is no danger of treating them as separable components of a bipartite state, and no \otimes is implied.

\mathcal{A} Sends Her Measurement Results to \mathcal{B}

\mathcal{A} now has a two (classical) bit result of her measurement: “ xy ” = “00”, “01”, “10” or “11.” She sends “ xy ” to \mathcal{B} through a classical channel, which takes time to get there.

\mathcal{B} Uses the Received Message to Extract $|\psi\rangle$ from His Qubit

\mathcal{B} 's qubit is in one of the four collapsed states

$$\begin{aligned} & \alpha |0\rangle_B + \beta |1\rangle_B \\ & \beta |0\rangle_B + \alpha |1\rangle_B \\ & \alpha |0\rangle_B - \beta |1\rangle_B \\ & -\beta |0\rangle_B + \alpha |1\rangle_B . \end{aligned}$$

This happened as a result of \mathcal{A} 's Bell basis measurement (instantaneous faster-than-light speed transfer, ergo “teleportation”). That’s the 99.99% I spoke of earlier. To get the final .01% of the way there, he needs to look at the two classical bits he received (which took time to reach him). They tell him which of those four states his qubit landed in. If it’s anything other than “00” he needs to apply a local unary operator to his B -register to “fix it up,” so it will be in the original $|\psi\rangle$. The rule is

\mathcal{B} Receives	\mathcal{B} Applies	\mathcal{B} Recovers
“00”	(nothing)	$ \psi\rangle$
“01”	X	$ \psi\rangle$
“10”	Z	$ \psi\rangle$
“11”	iY	$ \psi\rangle$

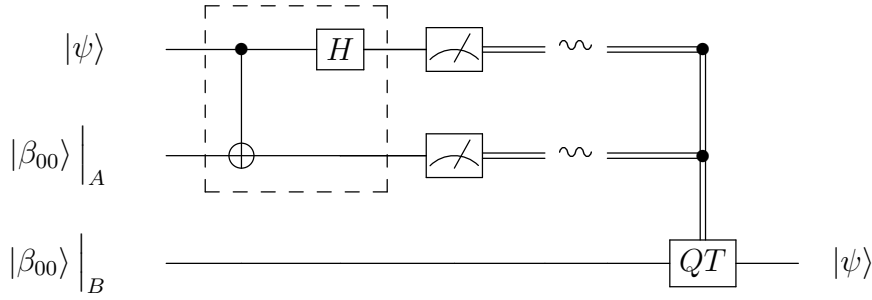
Nothing helps make an abstract description concrete like doing a calculation, so I recommend an ...

[Exercise.

- i) Express the operator $\text{BELL}^\dagger \otimes \mathbb{1}$ as an 8×8 matrix with the help of the section “The Matrix of a Separable Operator,” in the lesson on *tensor products*.
- ii) Express the state $|\varphi\rangle^3 = (\alpha |0\rangle + \beta |1\rangle) |\beta_{00}\rangle$ as an 8×1 column vector by multiplying it out (use this initial state description, not the “re-arranged” version that used the Bell basis).
- iii) Multiply the 8×8 operator matrix by the 8×1 state vector to get \mathcal{A} 's output state (prior to measurement) in column vector form.
- iv) Expand the last answer along the z -basis.
- v) Factor that last result in such a way that it looks the same as the answer we got when we applied $\text{BELL}^\dagger \otimes \mathbb{1}$ to the “re-arranged” version of $|\varphi\rangle^3$.]
- vi) In the case where \mathcal{B} receives the classical message “10” from \mathcal{A} , apply the corresponding “fix-it-up operator” shown in the table to his collapsed qubit and thereby prove that he recovers the exact teleported state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$.

A Circuit Representation of Quantum Teleportation

Just as we used “ SD ” to be one of four possible operators in the superdense coding algorithm, we will use “ QT ” to mean one of four operators ($\mathbb{1}$, X , Z , iY) that \mathcal{B} must apply to his register based on the message he receives from \mathcal{A} . With this shorthand, the quantum teleportation circuit can be expressed as:

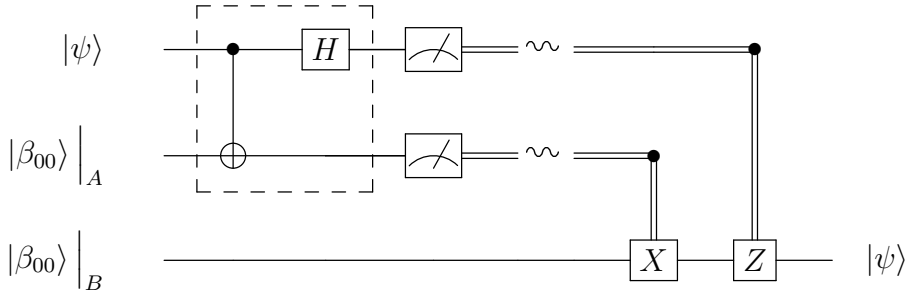


The circuit says that after taking the measurements (the *meter symbols*), \mathcal{A} “*radios*” the classical data (*double lines* and *wavy lines*) to \mathcal{B} who uses it to control (*filled circles*) which of the four operations he will apply to his qubit.

Once again, we use the identity

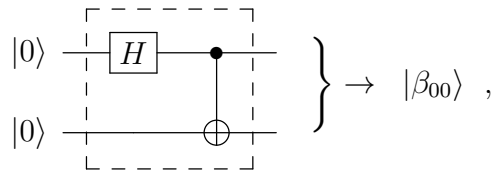
$$ZX = iY$$

to more precisely define the QT operation. We place a controlled- X and controlled- Z gates into \mathcal{B} ’s pipeline to get an improved circuit description.

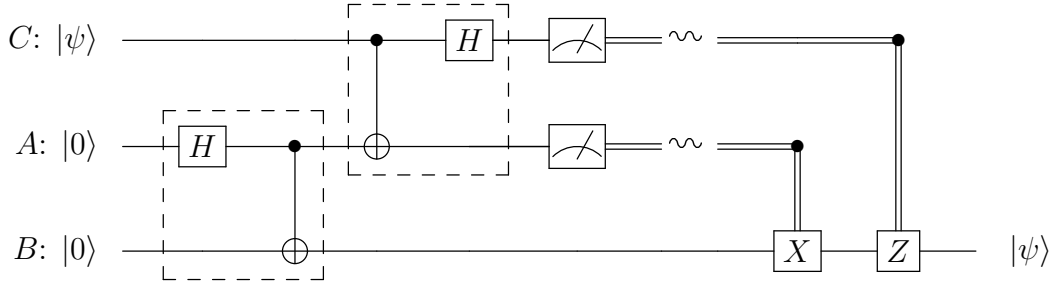


(Don’t forget that operators are applied from left-to-right in circuits, but right-to-left in algebra.)

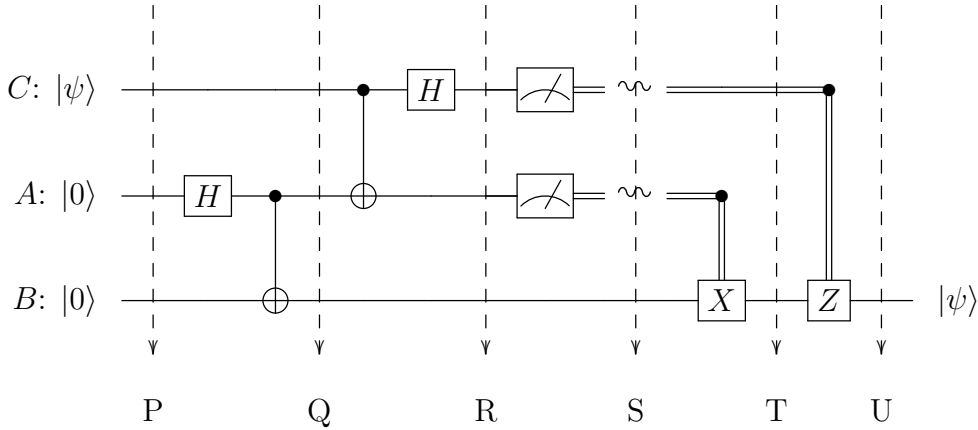
Many authors go a step further and add the initial gate that creates the AB -channel Bell state $|\beta_{00}\rangle$ from CBS kets:



which leads to



Here is the circuit with *access points* marked for an exercise:



Observe that the tripartite state

$$|\varphi\rangle^3 = |\psi\rangle_C |0\rangle_A |0\rangle_B$$

going into the entire circuit is transformed by various gates and measurements along the way. It continues to exist as a tripartite state to the very end, but you may not recognize it as such due to the classical wires and transmission of classical information around access points R and S, seemingly halting the qubit flow to their right. Yet the full order-3 state lives on. It is simply unnecessary to show the full state beyond that point, because registers C and A, after collapse, will contain one of the four CBS kets, $|x\rangle_C |y\rangle_A$, for $xy = 00, 01, 10$ or 11 . But those two registers never change after the measurement, and when Bob applies a measurement to his local register B, say iY perhaps, he will be implicitly applying the separable operator $\mathbb{1} \otimes \mathbb{1} \otimes iY$ to the full separable tripartite state.

[Exercise. Using *natural coordinates* for everything, compute the state of the vector $|\varphi\rangle^3$ as it travels through the access points, P-U: $|\varphi\rangle_P^3$, $|\varphi\rangle_Q^3$, $|\varphi\rangle_R^3$, $|\varphi\rangle_S^3$, $|\varphi\rangle_T^3$ and $|\varphi\rangle_U^3$. For points S, T and U you will have to know what measurement \mathcal{A} reads and sends to \mathcal{B} , so do those three points *twice*, once for a reading of $CA = "01"$ and once for a reading of $CA = "11"$. **HINT:** Starting with the easy point P, apply transformations carefully to the basis kets using *separable notation* like $(\mathbb{1} \otimes \text{BELL})$ or $(\text{BELL}^\dagger \otimes \mathbb{1})$. When you get to post-measurement classical pipes, apply the *Born Rule* which will select exactly one term in the sum.]

Why Teleportation Works.

Consider the main steps in the teleportation algorithm. We begin with three channels, the first of which contains *all* the quantum information we want to teleport, and the last two *none* of it,

$$|\varphi\rangle^3 = |\psi\rangle_C |\beta_{00}\rangle_{AB}.$$

Once we have the idea to entangle channels A and C by converting to the Bell basis (perhaps driven by the fact that one of the Bell states is in the AB register pair) we end up with a state in the general form,

$$\begin{aligned} |\varphi\rangle^3 \cong & |\beta_{00}\rangle_{CA} |\psi_{00}\rangle_B + |\beta_{01}\rangle_{CA} |\psi_{01}\rangle_B \\ & + |\beta_{10}\rangle_{CA} |\psi_{10}\rangle_B + |\beta_{11}\rangle_{CA} |\psi_{11}\rangle_B. \end{aligned}$$

Without even *looking* at any of four $|\psi_{jk}\rangle$ kets in the B -channel, we are convinced that 100% of the $|\psi\rangle$ information is now sitting inside that register, waiting to be tapped. Why?

The reason is actually quite simple.

Quantum gates – including basis transformations – are always *unitary* and thus *reversible*. If the Bell-basis operator had failed to transfer *all* the $|\psi\rangle$ information into the B -register, then, since *none* of it is left in the AC -registers (they contain only Bell states), there would be no hope of getting an inverse gate to recover our starting state which holds the full $|\psi\rangle$. Thus, producing an expression that left channels A and C bereft of any a trace of $|\psi\rangle$ information must necessarily produce a B -channel contains it *all*.

12.4 Introduction to Quantum Oracles

Superdense coding and *quantum teleportation* may seem more like *applications* of quantum computation than quantum *algorithms*. They enable us to transmit data in a way that is not possible using classical engineering. In contrast, *Deutsch's* little problem really *feels* algorithmic in nature and, indeed, its solution provides the template for the many quantum algorithms that succeed it.

We take a short side-trip to introduce *Boolean functions* then construct our first *quantum oracles* used in Deutsch's and later algorithms.

12.4.1 Boolean Functions and Reversibility

Classical unary gates – of which there are a grand total of four – contain both *reversible* operators ($\mathbb{1}$ and \neg) and *irreversible* ones (the $[0]$ -op and $[1]$ -op). Quantum operators require reversibility due to their unitarity, so there are no quantum analogs

for the latter two. Binary gates provide even more examples of irreversible classical operations for which there are no quantum counterparts.

These are specific examples of a general phenomenon that is more easily expressed in terms of classical *Boolean functions*.

Boolean Functions. A *Boolean function* is a function that has one or more binary digits (0 or 1) as *input*, and *one binary digit* as *output*.

From Unary Gates to Boolean Functions

A *classical unary gate* takes a single classical bit *in* and produces a single classical bit *out*. In the language of functions, it is nothing other than a *Boolean function* of one bit, i.e.,

$$f : \{0, 1\} \longrightarrow \{0, 1\},$$

or using the notation $\mathbb{B} \equiv \{0, 1\}$ introduced in the *single qubit lecture*,

$$f : \mathbb{B} \longrightarrow \mathbb{B}.$$

(As we defined it, \mathbb{B} had a richer structure than that of a simple set; it had a mod-2 addition operation \oplus that we will find useful in the definitions and computations to come.)

Reversible Example. To avoid becoming unmoored by abstractions, we revisit the negation operator in the language of Boolean functions. If we define

$$f(x) \equiv \neg x, \quad \text{for } x \in \mathbb{B},$$

then in terms of a *truth table*, the Boolean function f is

x	$f(x)$
0	1
1	0

This f is *reversible*, and in fact is *its own* inverse, since $f(f(x)) = x$.

Irreversible Example. On the other hand, if we define

$$g(x) \equiv 1, \quad \text{for } x \in \mathbb{B},$$

with a *truth table* of

x	$g(x)$
0	1
1	1

we have the quintessential example of an *irreversible* function.

Boolean functions (*classical* implied) will now become the *subject* of study. Computer science seeks to answer questions about such functions or create Boolean functions that do useful things. On the other hand, we will be using *quantum* circuits composed of *quantum* operators to answer questions about these *classical Boolean functions*. In other words, we have not abandoned the classical functions (I'll drop the modifier “Boolean” for now) in the least. On the contrary: they are the principle players in our narrative.

Binary Gates as Boolean Functions

The language naturally extends to functions of more than one input bit. To keep things simple, let's talk about *two bits*.

A *two bit function* (*classical* and *Boolean* implied) takes *two bits in* and produces one bit *out*. In other words,

$$f : \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\} \longrightarrow \{0, 1\},$$

or in \mathbb{B} notation,

$$f : \mathbb{B}^2 \longrightarrow \mathbb{B}.$$

(Column vs. row is not important here, so I'll use whichever fits better into the written page without the $()^t$ baggage.)

Note that we are avoiding the term “binary,” replacing it instead with “two bit” to avoid confusion arising from the fact that we are using binary digits for every input slot, whether a unary input or a multi-bit input.

Irreversibility in the Two (or Greater) Bit Case. Since two-input Boolean functions, like *all* Boolean functions, have a single bit out, they are inherently irreversible; we cannot undo the destruction that results from the loss of one or more bits.

(Necessarily) Irreversible Example. A typical two bit function that, like all two+ bit functions, is necessarily irreversible is the XOR, i.e.,

$$f(x, y) \equiv x \oplus y, \quad \text{for } \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{B}^2,$$

with the truth table

(x, y)	$f(x, y)$
(0, 0)	0
(0, 1)	1
(1, 0)	1
(1, 1)	0

12.4.2 The Quantum Oracle of a Boolean Function

Although our quantum algorithms will use quantum gates, they will often have to incorporate the classical functions that are the center of our investigations. But how can we do this when all quantum circuits are required to use unitary – and therefore *reversible* – gates? There is a well known classical technique for turning an otherwise irreversible function into one that is reversible. The technique pre-dates quantum computing, but we’ll look at it only in the quantum context, and if you’re interested in the classical analog, you can mentally “down-convert” ours by ignoring its superposition capability and focus only on the CBS inputs.

Oracles for Unary Functions

Suppose we are given a *black box* that computes some unary function, $f(x)$, even one that may be initially unknown to us. The term *black box* suggests that we don’t know what’s on the inside or how it works.

$$x \quad \text{---} \boxed{f} \text{---} f(x)$$

It can be shown that, using this black box – along with certain fundamental quantum gates – one can build a *new* gate that

- takes *two* bits in,
- has *two* bits out,
- is *unitary* (and therefore *reversible*),
- computes the function f when presented the proper input, and
- does so with the same efficiency (technically, the same *computational complexity*, a term we will define in a later lesson), as the black box f , whose irreversible function we want to reproduce.

We won’t describe how this works but, instead, take it as a given and call the new, larger circuit “ U_f ,” the *quantum oracle for f* . Its action on CBS kets and its circuit diagram are defined by

$$\begin{array}{lcl} \text{Data register:} & |x\rangle \text{---} & \boxed{U_f} \text{---} |x\rangle \\ \text{Target register:} & |y\rangle \text{---} & \boxed{U_f} \text{---} |y \oplus f(x)\rangle \end{array} ,$$

which also gives the name *data register* to the *A*-channel and *target register* to the *B*-channel.

First, notice that the output of the target register is a CBS; *inside* the ket we are XOR-ing two classical binary values y and $f(x)$, producing another binary value which, in turn, defines a CBS ket: either $|0\rangle$ or $|1\rangle$.

Example. We compute the matrix for U_f when $f(x) = 0$, the constant (and irreversible) [0]-op. Starting with the construction of the matrix of *any* linear transformation and moving on from there,

$$\begin{aligned} U_f &= (U_f|00\rangle, U_f|01\rangle, U_f|10\rangle, U_f|11\rangle) \\ &= (|0\rangle|0 \oplus f(0)\rangle, |0\rangle|1 \oplus f(0)\rangle, |1\rangle|0 \oplus f(1)\rangle, |1\rangle|1 \oplus f(1)\rangle) \\ &= (|0\rangle|f(0)\rangle, |0\rangle|\overline{f(0)}\rangle, |1\rangle|f(1)\rangle, |1\rangle|\overline{f(1)}\rangle), \end{aligned}$$

where we use the alternate notation for negation, $\bar{a} = \neg a$. So far, everything we did applies to the quantum oracle for *any* function f , so we'll put a pin in it for future use. Now, going on to apply it to $f = [0]$ -op,

$$\begin{aligned} U_{[0]\text{-op}} &= (|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle, |1\rangle|1\rangle) \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \end{aligned}$$

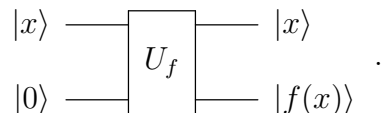
an interesting result in its own right, $U_{[0]\text{-op}} = \mathbf{1}$, but nothing to which we should attribute any deep meaning. Do note, however, that such a nice result makes it self-evident that U_f is not only *unitary* but its own inverse, as we show next it *always will be*.

U_f is Always its Own Inverse. We compute on the tensor CBS, and the result will be extensible to the entire $\mathcal{H} \otimes \mathcal{H}$ by linearity:

$$\begin{aligned} (U_f U_f)|xy\rangle &= U_f(U_f|xy\rangle) = U_f(|x\rangle|y \oplus f(x)\rangle) \\ &= |x\rangle| (y \oplus f(x)) \oplus f(x) \rangle \\ &= |x\rangle| y \oplus (f(x) \oplus f(x)) \rangle = |x\rangle|y\rangle = |xy\rangle \quad \text{QED} \end{aligned}$$

[**Exercise.** Why is $f(x) \oplus f(x) = 0$?]

U_f Computes $f(x)$. This is a simple consequence of the circuit definition, because if we plug $0 \rightarrow y$, we get



[**Exercise.** What do we get if we plug $1 \rightarrow y$?]

[**Exercise.** Compute the quantum oracles for the other three unary functions and observe that their matrices reveal them each to be unitary, not to mention self inverses.]

Notice that output of U_f for a CBS is always a *separable state*

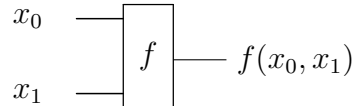
$$|x\rangle \otimes |y \oplus f(x)\rangle ,$$

since $y \oplus f(x)$ is always either 0 or 1.

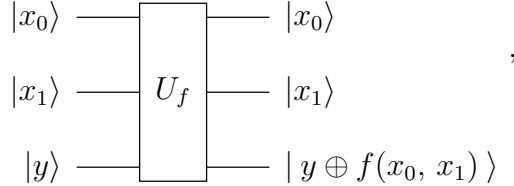
[**Exercise.** CBS input kets producing separable (even CBS) output kets *do not* a separable *operator* make. Prove this by looking at the three (U_f)s you computed in the last exercise and finding two which are patently non-separable.]

Oracles for Functions of Two Boolean Inputs

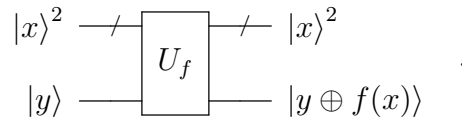
Everything extends smoothly to more than one-bit gates, so we only need outline the analysis for two-qubits. We are given *black box* of a two-input Boolean function, $f(x_0, x_1)$,



This time, we assume that circuit theory enables us to build a *three-in, three-out* oracle, U_f , defined by



usually shortened by using the *encoded form* of the CBS kets, $|x\rangle^2$, where $x \in \{0, 1, 2, 3\}$,



The key points are the same:

- U_f is its own inverse.
- U_f emulates $f(x)$ by setting $y = 0$,

$$U_f (|x\rangle^2 |0\rangle) = |x\rangle^2 |f(x)\rangle .$$

[**Exercise.** Compute the quantum oracle (in matrix form) for the classical AND gate.]

12.5 Deutsch's Problem

Our first quantum algorithm answers a question about an *unknown unary function* $f(x)$. It does not find the exact form of this function, but seeks only to answer a general question about its character. Specifically, we ask whether the function is *one-to-one* (distinct inputs produce distinct outputs) or *constant* (both inputs are mapped to the same output.)

Obviously, we can figure this out by evaluating *both* $f(0)$ and $f(1)$, after which we would know the answer, not to mention have a complete description of f . But the point is to see what we can learn about f without doing both evaluations of the function; we only want to do *one evaluation*. In a classical world if we only get to query f once we have to choose between inputs 0 or 1, and getting the output for our choice will not tell us whether the function is one-to-one or constant.

All the massive machinery we have accumulated in the past weeks can be brought to bear on this simple problem very neatly to demonstrate how quantum parallelism will beat classical computing in certain problems. It will set the stage for all quantum algorithms.

12.5.1 Definitions and Statement of the Problem

For this and a subsequent algorithm, we will define a property that a Boolean function might (or might not) have. We continue to assume that *function* means *Boolean function*.

Balanced and Constant Functions

Balanced Function. A *balanced function* is one that takes on the value 0 for exactly half of the possible inputs (and therefore 1 on the other half).

Two examples of balanced functions of two inputs are XOR and $\mathbb{1}_y : (x, y) \mapsto y$:

(x, y)	$XOR(x, y)$	(x, y)	$\mathbb{1}_y(x, y)$
$(0, 0)$	0	$(0, 0)$	0
$(0, 1)$	1	$(0, 1)$	1
$(1, 0)$	1	$(1, 0)$	0
$(1, 1)$	0	$(1, 1)$	1

Two unbalanced function of two inputs are AND and the $[1]$ -op:

(x, y)	$AND(x, y)$	(x, y)	$[1](x, y)$
(0, 0)	0	(0, 0)	1
(0, 1)	0	(0, 1)	1
(1, 0)	0	(1, 0)	1
(1, 1)	1	(1, 1)	1

Constant Functions. *Constant functions* are functions that always produce the same output regardless of the input. There are only two constant functions for any number of inputs: either the $[0]$ -op or the $[1]$ -op. See the truth table for the $[1]$ -op, above; the truth table for the $[0]$ -op would, of course, have 0s in the right column instead of 1s.

Balanced and Constant Unary Function

There are only four unary functions. Therefore the terms *balanced* and *constant* might seem heavy handed. The two *constant* functions are obviously the $[0]$ -op or the $[1]$ -op, and the other two are *balanced*. In fact, the balanced unary functions already have a term that describes them: *one-to-one*. There's even a simpler term in balanced functions in the unary case: *not constant*. To see this, let's lay all of our cards "on the table," pun intended.

x	1	x	\neg	x	$[0]$	x	$[1]$
0	0	0	1	0	0	0	1
1	1	1	0	1	0	1	1

So exactly two of our unary ops are *constant* and the other two are *balanced* = *one-to-one* = *not constant*.

The reason we complicate things by adding the vocabulary *constant* vs. *balanced* is that we will eventually move on to functions of more than one input, and in *those cases*,

- not all functions will be either balanced or one-to-one (e.g., the binary AND function isn't either), and
- balanced functions will not be one-to-one (e.g., binary XOR function is balanced but not one-to-one)

Deutsch's Problem

We are now ready to state Deutsch's problem using vocabulary that will help when we go to higher-input functions.

Deutsch’s Problem. *Given an unknown unary function that we are told is either **balanced** or **constant**, determine which it is in **one query** of the quantum oracle, U_f .*

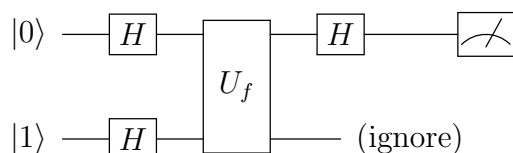
Notice that we are not asking to determine the exact function, just which category it belongs to. Even so, we cannot do it classically in a single query.

12.5.2 Deutsch’s Algorithm

The algorithm consists of building a circuit and measuring the A -register – *once*. That’s it. Our conclusion about f is determined by the result of the measurement: if we get a “0” the function is *constant*, if we get “1” the function is *balanced*. We will have gotten an answer about f with only one query of the oracle and thereby obtained a $\times 2$ improvement over a classical algorithm.

The Circuit

We combine the *quantum oracle* for f with a few *Hadamard gates* in a very small circuit:



Because there are only four unary functions, the temptation is to simply plug each one into U_f and confirm our claim. That’s not a bad *exercise* (which I’ll ask you to do), but let’s understand how one arrives at this design so we can use the ideas in other algorithms.

The Main Ideas Behind Deutsch’s Solution

Classical computing is embedded in quantum computing when we restrict our attention to the finite number of CBS kets that swim around in the infinite quantum ocean of the full state space. For the simplest *Hilbert space* imaginable – the first-order space, $\mathcal{H} = \mathcal{H}_{(1)}$ – those CBS kets consist of the two natural basis vectors $\{|0\rangle, |1\rangle\}$, corresponding to the classical bits [0] and [1]. We should expect that *no improvements to classical computing can be achieved by using only z -basis states (i.e., the CBS) for any algorithm or circuit*. Doing so would be using our Hilbert space as though it were the finite set $\{|0\rangle, |1\rangle\}$, a state of affairs that does nothing but simulate the classical world.

There are two quantum techniques that motivate the algorithm.

#1: Quantum Parallelism. This is the most general of the two ideas and will be used in all our algorithms. Any *non-trivial* superposition of the two CBS kets

$$\alpha|0\rangle + \beta|1\rangle, \quad \text{both } \alpha \text{ and } \beta \neq 0,$$

takes us off this classical plane into quantum hyperspace where all the fun happens. When we send such a non-trivial superposition through the quantum oracle, we are implicitly processing *both* z -basis kets – and therefore both classical states, $[0]$ and $[1]$ – simultaneously. This is the first big idea that fuels quantum computing and explains how it achieves its speed improvements. (The second big idea is quantum *entanglement*, but we’ll feature that one a little later.)

The practical impact of this technique in Deutsch’s algorithm is that we’ll be sending a perfectly balanced (or *maximally mixed*) superposition,

$$|0\rangle_x = \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle ,$$

through the *data register* (the A -channel) of the oracle, U_f .

#2: The Phase Kick-Back Trick. This isn’t quite as generally applicable as quantum parallelism, but it plays a role in several algorithms including some we’ll meet later in the course. It goes like this. If we feed the *other* maximally mixed state,

$$|1\rangle_x = \frac{1}{\sqrt{2}} |0\rangle - \frac{1}{\sqrt{2}} |1\rangle ,$$

into the *target register* (the B -channel) of U_f , we can transfer – or *kick-back* – 100% of the information about the unknown function $f(x)$ from the B -register output to the A -register output.

You’ve actually experienced this idea earlier today when you studied quantum teleportation. Recall that by merely rearranging the initial configuration of our input state we were able to effect a seemingly magical transfer of $|\psi\rangle$ from one channel to the other. In the current context, presenting the x -basis ket, $|1\rangle_x$, to the target register will have a similar effect.

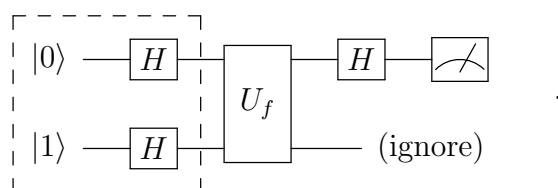
Temporary Change in Notation

Because we are going to make heavy use of the x -basis kets here and the variable x is being used as the Boolean input to the function $f(x)$, I am going to call into action our alternate x -basis notation,

$$\begin{aligned} |+\rangle &\equiv |0\rangle_x & \text{and} \\ |-\rangle &\equiv |1\rangle_x . \end{aligned}$$

Preparing the Oracle’s Input: The Two Left Hadamard Gates

Together, the two techniques explain the first part of Deutsch’s circuit (in the dashed-box),



We recognize H as the operator that takes z -basis kets to x -basis kets, thus manufacturing a $|+\rangle$ (i.e., $|0\rangle_x$) for the *data register input* and $|-\rangle$ (i.e., $|1\rangle_x$) for the *target register input*,

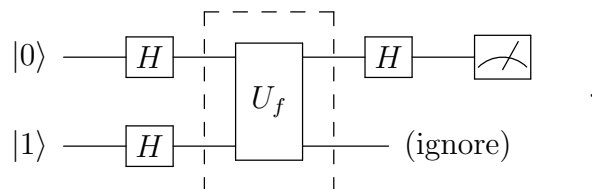
$$\begin{aligned} |0\rangle &\longrightarrow \boxed{H} \longrightarrow |+\rangle \\ |1\rangle &\longrightarrow \boxed{H} \longrightarrow |-\rangle \end{aligned} .$$

In other words, the Hadamard gate converts the two natural basis kets (easy states to prepare) into superposition inputs for quantum oracle. The *top* gate sets up *quantum parallelism* for the circuit, and the bottom one sets up the *phase kick-back*. For reference, algebraically these two gates perform

$$\begin{aligned} H|0\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \quad \text{and} \\ H|1\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle . \end{aligned}$$

Analyzing the Oracle

The real understanding of how the algorithm works comes by analyzing the kernel of the circuit, the oracle (in the dashed-box),



Step 1. CBS Into Both Channels. We creep up slowly on our result by first considering a CBS ket into both registers, a result we know immediately by definition of U_f ,

$$\begin{aligned} \text{Data register: } &|x\rangle \longrightarrow \boxed{U_f} \longrightarrow |x\rangle \\ \text{Target register: } &|y\rangle \longrightarrow \boxed{U_f} \longrightarrow |y \oplus f(x)\rangle \end{aligned} ,$$

or algebraically,

$$U_f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle$$

Step 2. CBS Into Data and Superposition into Target. We stick with a CBS $|x\rangle$ going into the data register, but now allow the superposition $|-\rangle$ to go into

the target register. Extend the above linearly,

$$\begin{aligned}
U_f(|x\rangle|-\rangle) &= U_f\left(|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\right) = \frac{U_f(|x\rangle|0\rangle) - U_f(|x\rangle|1\rangle)}{\sqrt{2}} \\
&= \frac{|x\rangle|0 \oplus f(x)\rangle - |x\rangle|1 \oplus f(x)\rangle}{\sqrt{2}} \\
&= \frac{|x\rangle|f(x)\rangle - |x\rangle|\overline{f(x)}\rangle}{\sqrt{2}} = |x\rangle\left(\frac{|f(x)\rangle - |\overline{f(x)}\rangle}{\sqrt{2}}\right).
\end{aligned}$$

This amounts to

$$\begin{aligned}
U_f(|x\rangle|-\rangle) &= |x\rangle\begin{cases} \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{when } f(x) = 0 \\ \frac{|1\rangle - |0\rangle}{\sqrt{2}}, & \text{when } f(x) = 1 \end{cases} \\
&= |x\rangle(-1)^{f(x)}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).
\end{aligned}$$

Since it's a scalar, $(-1)^{f(x)}$ can be moved to the left and be attached to the A -register's $|x\rangle$, a mere rearrangement of the terms,

$$U_f(|x\rangle|-\rangle) = (-1)^{f(x)}|x\rangle\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = \left((-1)^{f(x)}|x\rangle\right)|-\rangle,$$

and we have successfully (like magic) moved all of the information about $f(x)$ from the B -register to the A -register, where it appears as an overall phase factor in the scalar's exponent, $(-1)^{f(x)}$.

The Oracle's part of the circuit would process this intermediate step's data as follows.

$$\begin{array}{ccc}
|x\rangle & \text{---} & \boxed{U_f} & \text{---} & (-1)^{f(x)}|x\rangle \\
|-\rangle & \text{---} & & \text{---} & |-\rangle
\end{array},$$

Although we have a ways to go, let's pause to summarize what we have accomplished so far.

- **Quantum Mechanics:** This is a non-essential, theoretical note to test your memory of our quantum mechanics lesson. We have proven that $|x\rangle|-\rangle$ is an *eigenvector* of U_f with *eigenvalue* $(-1)^{f(x)}$ for $x = 0, 1$. By an exercise you'll have the opportunity to work at the end of this chapter, we learn that all oracles are Hermitian. The current observation – that this oracle's eigenvalues are -1 and +1 – is compatible with the “Hermiticity” of its U_f [**Exercise:** Why?]

- **Quantum Computing:** The information about $f(x)$ is encoded – “kicked-back” – in the A (*data*) register’s output. That’s where we plan to look for it in the coming step. Viewed this way, the B -register retains no useful information; just like in teleportation, a rearrangement of the data sometimes creates a perceptual shift of information from one channel to another that we can exploit by measuring along a different basis – something we will do in a moment.

Step 3. Superpositions into Both Registers. Finally, we want the state $|+\rangle$ to go into the data register so we can process both $f(0)$ and $f(1)$ in a single pass. The effect is to present the separable $|+\rangle \otimes |-\rangle$ to the oracle and see what comes out. Applying linearity to the last result we get

$$\begin{aligned}
 U_f(|+\rangle|-\rangle) &= U_f\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}|-\rangle\right) \\
 &= \frac{U_f(|0\rangle|-\rangle) + U_f(|1\rangle|-\rangle)}{\sqrt{2}} \\
 &= \frac{(-1)^{f(0)}|0\rangle|-\rangle + (-1)^{f(1)}|1\rangle|-\rangle}{\sqrt{2}} \\
 &= \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}|-\rangle.
 \end{aligned}$$

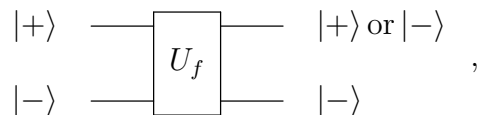
By combining the *phase kick-back* with *quantum parallelism*, we’ve managed to get an expression containing both $f(0)$ and $f(1)$ in the A -register. We now ask the question that Deutsch posed in the context of this simple expression, “What is the difference between the *balanced* case ($f(0) \neq f(1)$) and the *constant* case ($f(0) = f(1)$)?” *Answer:* When constant, the two terms in the numerator have the same sign and when balanced, they have different signs, to wit,

$$U_f(|+\rangle|-\rangle) = \begin{cases} (\pm 1) \frac{|0\rangle + |1\rangle}{\sqrt{2}}|-\rangle, & \text{if } f(0) = f(1) \\ (\pm 1) \frac{|0\rangle - |1\rangle}{\sqrt{2}}|-\rangle, & \text{if } f(0) \neq f(1) \end{cases}$$

We don’t care about a possible overall phase factor or (-1) in front of all this since it’s a unit scalar in a state space. Dumping it and noticing that the A -register has x -basis kets in both cases, we get the ultimate simplification,

$$U_f(|+\rangle|-\rangle) = \begin{cases} |+\rangle|-\rangle, & \text{if } f(0) = f(1) \\ |-\rangle|-\rangle, & \text{if } f(0) \neq f(1) \end{cases}$$

the perfect form for an x -basis measurement. Before we do that, let’s have a look at the oracle’s input and output states,

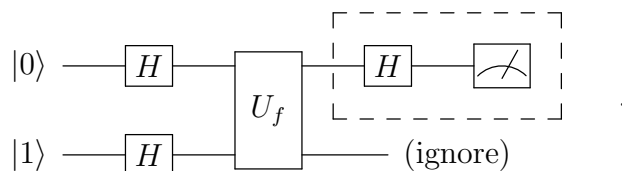


Measurement

We only care about the A -register, since the B -register will always collapse to $|-\rangle$. The conclusion?

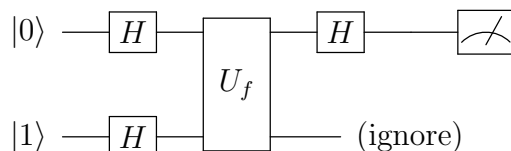
After measuring the A -register along the x -basis, if we collapse to $|+\rangle$, f is constant, and if we collapse to $|-\rangle$, f is balanced.

Of course an x -basis measurement is nothing more than a z -basis measurement after applying the $x \leftrightarrow z$ basis transforming unitary H . This explains the insertion of the final Hadamard gate in the upper right (dashed,



Deutsch's Algorithm in Summary

We've explained the purpose of all the components in the circuit and how each plays a role in leveraging *quantum parallelism* and *phase kick-back*. The result is extremely easy to state. We run the circuit



one time only and measure the *data register output* in the natural basis.

- If we read “0,” f is constant.
- If we read “1,” f is balanced.

This may not seem like game changing result; a quantum speed up of $2\times$ in a problem that is both trivial and without any real world application, but it demonstrates that there is a difference between quantum computing and classical computing. It also lays the groundwork for the more advanced algorithms to come.

In all the oracles we've studied, their rows (and columns) consisted of only 1s and 0s, and were their own inverse. Show that this implies they are Hermitian.

[Exercise. In all the oracles we've studied, their rows (and columns) consisted of only 1s and 0s, and each one was its own inverse. Show that this implies they were all Hermitian.]

12.6 n Qubits and More Algorithms

We're just getting started, though. In the next lesson we'll attack a general n -qubit computer and two algorithms that work on that kind of system, so get ready for more fun.

Chapter 13

Multi-Qubit Systems and Algorithms

13.1 Moving Up from 2 Qubits to n Qubits

This week we generalize our work with one and two-qubit computation to include n -qubits for any integer $n > 2$. We'll start by defining n th order tensor products, a natural extension of what we did for 2nd and 3rd order products, then apply that to n th order state spaces, $\mathcal{H}_{(n)}$.

It's common for non-math major undergraduates to be confounded by higher order tensors due to the vast number of coordinates and large dimensions, so I'll give you a running start by doing a short recap of second and third order tensor products first. This will establish a pattern that should make the larger orders go down more smoothly.

13.2 General Tensor Products

13.2.1 Recap of Order-2 Tensor Products

Objects of the Product Space and Induced Basis

We learned that the tensor product of two vector spaces, A and B , having dimensions d_A and d_B , respectively, is the *product space*,

$$W = A \otimes B,$$

whose vectors (a.k.a. *tensors*) consist of objects, \mathbf{w} , expressible as weighted sums of a “separable basis” (defined, below), i.e.,

$$\mathbf{w} = \sum_{\substack{k=0, \\ j=0}}^{\substack{d_A-1, \\ d_B-1}} c_{kj} (\mathbf{a}_k \otimes \mathbf{b}_j),$$

where the c_{kj} are the scalar weights and also serve as the *coordinates* of \mathbf{w} along that basis.

The separable basis tensors appearing in the above linear combination are the $d_A d_B$ vectors

$$\left\{ \mathbf{a}_k \otimes \mathbf{b}_j \mid k = 0, \dots, (d_A - 1) \quad \text{and} \quad j = 0, \dots, (d_B - 1) \right\},$$

induced by the two component bases,

$$\begin{aligned} \mathcal{A} &= \left\{ \mathbf{a}_k \right\}_{k=0}^{d_A-1} \quad \text{and} \\ \mathcal{B} &= \left\{ \mathbf{b}_j \right\}_{j=0}^{d_B-1}. \end{aligned}$$

The *sums*, *products* and *equivalence* of tensor expressions were defined by the required *distributive* and *commutative* properties, but can often be taken as the natural rules one would expect.

Separable Operators in the Product Space

A *separable operator* on the product space is one that arises from two component operators, T_A and T_B , each defined on its respective component space, A and B . This separable tensor operator is defined first by its action on separable order-2 tensors,

$$[T_A \otimes T_B](\mathbf{a} \otimes \mathbf{b}) \equiv T_A(\mathbf{a}) \otimes T_B(\mathbf{b}),$$

and since the basis tensors are of this form, it establishes the action of $T_A \otimes T_B$ on the basis which, in turn, extends the action to the whole space.

13.2.2 Recap of Order-3 Tensor Products

We also outlined the same process for a *third-order* tensor product space

$$W = A \otimes B \otimes C$$

in order to acquire the vocabulary needed to present a few of the early quantum algorithms involving three channels. Here is a summary of that section.

Objects of the Product Space and Induced Basis

Assuming A , B and C have dimensions d_A , d_B and d_C , respectively, the basis for the product space is the set of $d_A d_B d_C$ separable tensors

$$\left\{ \mathbf{a}_k \otimes \mathbf{b}_j \otimes \mathbf{c}_l \right\},$$

where

\mathbf{a}_k is the k^{th} vector in the basis for A ,
 \mathbf{b}_j is the j^{th} vector in the basis for B and
 \mathbf{c}_l is the l^{th} vector in the basis for C .

A general *tensor* \mathbf{w} in the product space is expressible as a weighted sum of these basis tensors,

$$\mathbf{w} = \sum_{\substack{k=0, \\ j=0, \\ l=0}}^{\substack{d_A-1, \\ d_B-1, \\ d_C-1}} c_{kjl} (\mathbf{a}_k \otimes \mathbf{b}_j \otimes \mathbf{c}_l),$$

where the c_{kjl} are the scalar weights (or coordinates) that define \mathbf{w} .

Separable Operators in the Product Space

A *separable operator* on the product space is one that arises from three component operators, T_A , T_B and T_C , each defined on its respective component space, A , B and C . This separable tensor operator is defined first by its action on separable order-3 tensors,

$$[T_A \otimes T_B \otimes T_C] (\mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}) \equiv T_A(\mathbf{a}) \otimes T_B(\mathbf{b}) \otimes T_C(\mathbf{c}),$$

and since the basis tensors are of this form, that establishes the action of $T_A \otimes T_B \otimes T_C$ on the basis which, in turn, extends the action to the whole space.

13.2.3 Higher Order Tensor Products

We now formally generalize these concepts to *any* order tensor product space

$$W = A_0 \otimes A_1 \otimes \cdots \otimes A_{n-2} \otimes A_{n-1},$$

for $n \geq 2$. We'll label the dimensions of the component spaces by

$$\begin{aligned} \dim(A_0) &= d_0, \\ \dim(A_1) &= d_1, \\ &\vdots \\ \dim(A_{n-2}) &= d_{n-2} \quad \text{and} \\ \dim(A_{n-1}) &= d_{n-1}. \end{aligned}$$

The tensor product space will have dimension

$$\dim(W) = d_0 d_1 \cdots d_{n-2} d_{n-1} = \prod_{k=0}^{n-1} d_k$$

which seems really big (and *is* big in fields like general relativity), but for us each component space is \mathcal{H} which has dimension two, so $\dim(W)$ will be the – still large but at least palatable – number 2^n .

Objects of the Product Space and Induced Basis

The vectors – a.k.a. *tensors* – of the space consist of those \mathbf{w} expressible as weighted sums of the separable basis

$$\left\{ \mathbf{a}_{0k_0} \otimes \mathbf{a}_{1k_1} \otimes \mathbf{a}_{2k_2} \otimes \cdots \otimes \mathbf{a}_{(n-1)k_{n-1}} \right\}_{k_0, k_1, \dots, k_{n-1} = 0, 0, \dots, 0}^{d_0-1, d_1-1, \dots, d_{n-1}-1}.$$

The somewhat daunting subscript notation says that

$$\begin{aligned} \mathbf{a}_{0k_0} &\text{ is the } (k_0)^{\text{th}} \text{ vector in the basis for } A_0, \\ \mathbf{a}_{1k_1} &\text{ is the } (k_1)^{\text{th}} \text{ vector in the basis for } A_1, \\ \mathbf{a}_{2k_2} &\text{ is the } (k_2)^{\text{th}} \text{ vector in the basis for } A_2, \\ &\vdots \\ \mathbf{a}_{(n-1)k_{n-1}} &\text{ is the } (k_{n-1})^{\text{th}} \text{ vector in the basis for } A_{n-1}, \end{aligned}$$

If we write this algebraically, the typical \mathbf{w} in W has a unique expansion along the tensor basis weighted by the scalars $c_{k_0 k_1 \dots k_{n-1}}$,

$$\mathbf{w} = \sum c_{k_0 k_1 \dots k_{n-1}} (\mathbf{a}_{0k_0} \otimes \mathbf{a}_{1k_1} \otimes \mathbf{a}_{2k_2} \otimes \cdots \otimes \mathbf{a}_{(n-1)k_{n-1}}).$$

This notation is an order of magnitude more general than we need, but it is good to have down for reference. We'll see that the expression takes on a much more manageable form when we get into the state spaces of quantum computing.

The *sums*, *products* and *equivalence* of tensor expressions have definitions analogous to their lower-order prototypes. You'll see examples as we go.

Separable Operators in the Product Space

A *separable operator* on the product space is one that arises from n component operators, T_0, T_1, \dots, T_{n-1} , each defined on its respective component space, A_0, A_1, \dots, A_{n-1} . This separable tensor operator is defined first by its action on separable order- n tensors

$$\begin{aligned} [T_0 \otimes T_1 \otimes \dots \otimes T_{n-1}] (\mathbf{v}_0 \otimes \mathbf{v}_1 \otimes \dots \otimes \mathbf{v}_{n-1}) \\ \equiv T_0(\mathbf{v}_0) \otimes T_1(\mathbf{v}_1) \otimes \dots \otimes T_{n-1}(\mathbf{v}_{n-1}), \end{aligned}$$

and since the basis tensors are always separable,

$$\mathbf{a}_{0k_0} \otimes \mathbf{a}_{1k_1} \otimes \dots \otimes \mathbf{a}_{(n-1)k_{n-1}},$$

this establishes the action of $T_0 \otimes T_1 \otimes \dots \otimes T_{n-1}$ on the basis,

$$\begin{aligned} [T_0 \otimes T_1 \otimes \dots \otimes T_{n-1}] (\mathbf{a}_{0k_0} \otimes \mathbf{a}_{1k_1} \otimes \dots \otimes \mathbf{a}_{(n-1)k_{n-1}}) \\ \equiv T_0(\mathbf{a}_{0k_0}) \otimes T_1(\mathbf{a}_{1k_1}) \otimes \dots \otimes T_{n-1}(\mathbf{a}_{(n-1)k_{n-1}}), \end{aligned}$$

which, in turn, extends the action to the whole space.

Notation

Sometimes we use the \prod or \bigotimes notation to shorten expressions. In these forms, the product space would be written in one of the two equivalent ways

$$W = \bigotimes_{k=0}^{n-1} A_k = \prod_{k=0}^{n-1} A_k,$$

the induced basis in one of

$$\left\{ \bigotimes_{j=0}^{n-1} \mathbf{a}_{jk_j} \right\}_{k_0, k_1, \dots, k_{n-1} = 0, 0, \dots, 0}^{d_0-1, d_1-1, \dots, d_{n-1}-1} = \left\{ \prod_{j=0}^{n-1} \mathbf{a}_{jk_j} \right\}_{k_0, k_1, \dots, k_{n-1} = 0, 0, \dots, 0}^{d_0-1, d_1-1, \dots, d_{n-1}-1},$$

a separable operator as one of

$$\bigotimes_{k=0}^{n-1} T_k = \prod_{k=0}^{n-1} T_k,$$

and a separable operator's action on a separable state as either

$$\left[\bigotimes_{k=0}^{n-1} T_k \right] \left(\bigotimes_{k=0}^{n-1} \mathbf{v}_k \right) = \bigotimes_{k=0}^{n-1} T_k(\mathbf{v}_k)$$

or

$$\left[\prod_{k=0}^{n-1} T_k \right] \left(\prod_{k=0}^{n-1} \mathbf{v}_k \right) = \prod_{k=0}^{n-1} T_k(\mathbf{v}_k).$$

13.3 n -Qubit Systems

The next step in this lecture is to define the precise state space we need for a quantum computer that supports n qubits. I won't back up all the way to two qubits as I did for the tensor product, but a short recap of *three qubits* will prove useful in the transition to n qubits.

13.3.1 Recap of Three Qubits

A *three qubit system* is modeled by a third order tensor product of three identical copies of our friendly spin-1/2 Hilbert space, \mathcal{H} . We can use either the *order*-notation or *component space label* notation to signify the product space,

$$\mathcal{H}_{(3)} \cong \mathcal{H}_A \otimes \mathcal{H}_B \otimes \mathcal{H}_C .$$

The dimension is $2 \times 2 \times 2 = 8$.

Three Qubit CBS and Coordinate Convention

The natural three qubit tensor basis is constructed by forming all possible separable products of component space basis vectors, and we continue to use our CBS ket notation. The CBS for $\mathcal{H}_{(3)}$ is therefore

$$\left\{ \begin{array}{l} |0\rangle \otimes |0\rangle \otimes |0\rangle , \quad |0\rangle \otimes |0\rangle \otimes |1\rangle , \quad |0\rangle \otimes |1\rangle \otimes |0\rangle , \quad |0\rangle \otimes |1\rangle \otimes |1\rangle , \\ |1\rangle \otimes |0\rangle \otimes |0\rangle , \quad |1\rangle \otimes |0\rangle \otimes |1\rangle , \quad |1\rangle \otimes |1\rangle \otimes |0\rangle , \quad |1\rangle \otimes |1\rangle \otimes |1\rangle \end{array} \right\} ,$$

with the often used shorthand options

$$\begin{array}{lllll} |0\rangle \otimes |0\rangle \otimes |0\rangle & \longleftrightarrow & |0\rangle |0\rangle |0\rangle & \longleftrightarrow & |000\rangle & \longleftrightarrow & |0\rangle^3 \\ |0\rangle \otimes |0\rangle \otimes |1\rangle & \longleftrightarrow & |0\rangle |0\rangle |1\rangle & \longleftrightarrow & |001\rangle & \longleftrightarrow & |1\rangle^3 \\ |0\rangle \otimes |1\rangle \otimes |0\rangle & \longleftrightarrow & |0\rangle |1\rangle |0\rangle & \longleftrightarrow & |010\rangle & \longleftrightarrow & |2\rangle^3 \\ |0\rangle \otimes |1\rangle \otimes |1\rangle & \longleftrightarrow & |0\rangle |1\rangle |1\rangle & \longleftrightarrow & |011\rangle & \longleftrightarrow & |3\rangle^3 \\ |1\rangle \otimes |0\rangle \otimes |0\rangle & \longleftrightarrow & |1\rangle |0\rangle |0\rangle & \longleftrightarrow & |100\rangle & \longleftrightarrow & |4\rangle^3 \\ |1\rangle \otimes |0\rangle \otimes |1\rangle & \longleftrightarrow & |1\rangle |0\rangle |1\rangle & \longleftrightarrow & |101\rangle & \longleftrightarrow & |5\rangle^3 \\ |1\rangle \otimes |1\rangle \otimes |0\rangle & \longleftrightarrow & |1\rangle |1\rangle |0\rangle & \longleftrightarrow & |110\rangle & \longleftrightarrow & |6\rangle^3 \\ |1\rangle \otimes |1\rangle \otimes |1\rangle & \longleftrightarrow & |1\rangle |1\rangle |1\rangle & \longleftrightarrow & |111\rangle & \longleftrightarrow & |7\rangle^3 \end{array} .$$

The notation of the first two columns admits the possibility of labeling each of the component kets with the \mathcal{H} whence it came, A , B or C ,

$$\begin{aligned} |0\rangle_A \otimes |0\rangle_B \otimes |0\rangle_C &\longleftrightarrow |0\rangle_A |0\rangle_B |0\rangle_C , \\ |0\rangle_A \otimes |0\rangle_B \otimes |1\rangle_C &\longleftrightarrow |0\rangle_A |0\rangle_B |1\rangle_C , \\ \text{etc.} \end{aligned}$$

The densest of the notations expresses the CBS ket as an integer from 0 to 7. This can all be summarized by looking at the third order coordinate representation of these eight tensors:

$ 000\rangle$	$ 001\rangle$	$ 010\rangle$	$ 011\rangle$	$ 100\rangle$	$ 101\rangle$	$ 110\rangle$	$ 111\rangle$
$ 0\rangle^3$	$ 1\rangle^3$	$ 2\rangle^3$	$ 3\rangle^3$	$ 4\rangle^3$	$ 5\rangle^3$	$ 6\rangle^3$	$ 7\rangle^3$
$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

A typical three qubit value is a normalized superposition of the eight CBS, e.g.,

$$\begin{aligned} &\frac{|3\rangle^3 + |5\rangle^3}{\sqrt{2}}, \quad \frac{|2\rangle^3 + e^{99i} |3\rangle^3 + i |7\rangle^3}{\sqrt{3}}, \\ &\sqrt{.1} |0\rangle^3 + \sqrt{-.6} |2\rangle^3 - \sqrt{.05} |4\rangle^3 - (\sqrt{.05} + i\sqrt{.2}) |6\rangle^3, \end{aligned}$$

or most generally,

$$\sum_{k=0}^7 c_k |k\rangle^3,$$

where

$$\sum_{k=0}^7 |c_k|^2 = 1.$$

13.3.2 Three Qubit Logic Gates; the Toffoli Gate

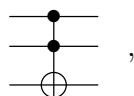
Order three quantum logic gates are unitary operators on $\mathcal{H}_{(3)}$. They can be constructed by taking, for example,

- the separable product of three first order operators,
- the separable product of a second order and a first order operator, or
- any operator defined by a possibly non-separable unitary matrix.

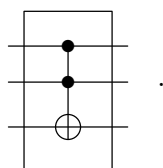
In this course, we won't be studying third order gates other than certain separable products of first order gates (like $H^{\otimes 3}$ discussed in the next section). However, let's take a sneak peek at the non-separable *Toffoli gate* which plays a role in reversible computation and some of our work in the later courses CS 83B and CS 83C.

⌋ : The Symbol

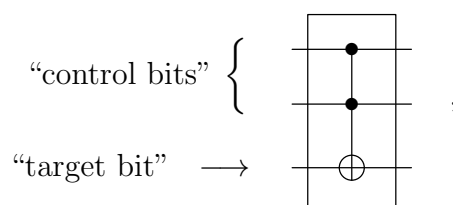
The gate is usually drawn without an enclosing box as in



but can be boxed to emphasize its overall effect on a tripartite state,



The A and B registers are the *control bits*, and the C register the *target bit*,

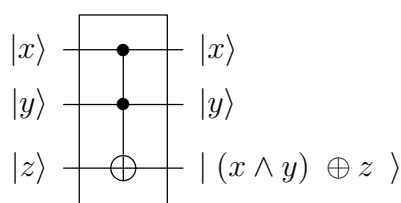


two terms that will become clear in the next bullet point.

At times I'll use all caps, as in TOFFOLI, to name the gate in order to give it the status of its simpler cousin, CNOT.

$|x\rangle |y\rangle$: Action on the CBS

The TOFFOLI gate has the following effect on the computational basis states:



In terms of the eight CBS tensors, it leaves the A and B registers unchanged and negates the C register qubit or leaves it alone based on whether the AND of the control bits is “1” or “0”:

$$|z\rangle \mapsto \begin{cases} |z\rangle, & \text{if } x \wedge y = 0 \\ |\neg z\rangle, & \text{if } x \wedge y = 1 \end{cases}$$

It is a *controlled-NOT operator*, but the control consists of two bits rather than one.

Remember, not every CBS definition we can drum up will result in a unitary operator, especially when we start defining the output kets in terms of arbitrary classical operations. In an exercise during your two qubit lesson you met a bipartite “gate” which seemed simple enough but turned out not to be unitary. So we must confirm this property in the next bullet.

(...) : The Matrix

We compute the column vectors of the matrix by applying TOFFOLI to the CBS tensors to get

$$\begin{aligned} M_{\text{TOFFOLI}} &= \left(\text{TOFFOLI } |000\rangle, \text{ TOFFOLI } |001\rangle, \dots, \text{ TOFFOLI } |111\rangle \right) \\ &= \left(|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |111\rangle, |110\rangle \right) \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{aligned}$$

which is an identity matrix until we reach the last two rows (columns) where it swaps those rows (columns). It *is* unitary.

[**Exercise.** Prove that this is *not* a separable operator.]

$|\psi\rangle^2$: Behavior on General State

Applying TOFFOLI to the general state,

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_7 \\ c_6 \end{pmatrix}.$$

This is as far as we need to go on the Toffoli gate. Our interest here is in higher order gates that are separable products of unary gates.

13.3.3 n Qubits

Definition and Notation

Definition of n Qubits. *n qubits are, collectively, the tensor product of n identical copies of the 2-D Hilbert space \mathcal{H} , and the **value** of the n qubits at any given moment is a particular tensor in this space having unit length.*

Note. As usual, we consider two unit-tensors which differ by a phase factor, $e^{i\theta}$ for real θ , to be the same n qubit value.

We can designate this state space using the notation

$$\mathcal{H}_{(n)} = \overbrace{\mathcal{H} \otimes \mathcal{H} \otimes \dots \otimes \mathcal{H}}^n = \bigotimes_{k=0}^{n-1} \mathcal{H}.$$

n Qubit Computational Basis States

The natural n qubit tensor basis is constructed by forming all possible separable products of component space basis vectors. The CBS for $\mathcal{H}_{(n)}$ is therefore

$$\left\{ \begin{aligned} &|0\rangle \otimes \dots \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle, \quad |0\rangle \otimes \dots \otimes |0\rangle \otimes |0\rangle \otimes |1\rangle, \quad |0\rangle \otimes \dots \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle, \\ &|0\rangle \otimes \dots \otimes |0\rangle \otimes |1\rangle \otimes |1\rangle, \quad |0\rangle \otimes \dots \otimes |1\rangle \otimes |0\rangle \otimes |0\rangle, \quad \dots, \\ &\dots, \quad |1\rangle \otimes \dots \otimes |1\rangle \otimes |1\rangle \otimes |0\rangle, \quad |1\rangle \otimes \dots \otimes |1\rangle \otimes |1\rangle \otimes |1\rangle \end{aligned} \right\},$$

with the shorthand options

$$\begin{aligned}
|0\rangle \otimes \cdots \otimes |0\rangle \otimes |0\rangle &\longleftrightarrow |0\rangle \cdots |0\rangle |0\rangle \longleftrightarrow |0 \cdots 000\rangle \longleftrightarrow |0\rangle^n \\
|0\rangle \otimes \cdots \otimes |0\rangle \otimes |1\rangle &\longleftrightarrow |0\rangle \cdots |0\rangle |1\rangle \longleftrightarrow |0 \cdots 001\rangle \longleftrightarrow |1\rangle^n \\
|0\rangle \otimes \cdots \otimes |1\rangle \otimes |0\rangle &\longleftrightarrow |0\rangle \cdots |1\rangle |0\rangle \longleftrightarrow |0 \cdots 010\rangle \longleftrightarrow |2\rangle^n \\
|0\rangle \otimes \cdots \otimes |1\rangle \otimes |1\rangle &\longleftrightarrow |0\rangle \cdots |1\rangle |1\rangle \longleftrightarrow |0 \cdots 011\rangle \longleftrightarrow |3\rangle^n \\
&\vdots \\
|1\rangle \otimes \cdots \otimes |1\rangle \otimes |0\rangle &\longleftrightarrow |1\rangle \cdots |1\rangle |0\rangle \longleftrightarrow |1 \cdots 110\rangle \longleftrightarrow |2^n - 2\rangle^n \\
|1\rangle \otimes \cdots \otimes |1\rangle \otimes |1\rangle &\longleftrightarrow |1\rangle \cdots |1\rangle |1\rangle \longleftrightarrow |1 \cdots 111\rangle \longleftrightarrow |2^n - 1\rangle^n
\end{aligned}$$

Dimension of $\mathcal{H}_{(n)}$

As you can tell by counting, there are 2^n basis tensors in the product space, which makes sense because the dimension of the product space is the product of the dimensions of the component spaces; since $\dim(\mathcal{H}) = 2$, $\dim(\mathcal{H}_{(n)}) = 2 \times 2 \times \cdots \times 2 = 2^n$, \checkmark .

For n th order CBS kets we usually label each component ket using the letter x with its corresponding *space label*,

$$|x_{n-1}\rangle \otimes |x_{n-2}\rangle \otimes |x_{n-3}\rangle \otimes \cdots \otimes |x_0\rangle, \quad x_k \in \{0, 1\},$$

with the more common and denser alternatives

$$|x_{n-1}\rangle |x_{n-2}\rangle \cdots |x_1\rangle |x_0\rangle = |x_{n-1} x_{n-2} \cdots x_3 x_2 x_1 x_0\rangle,$$

and densest of them all,

$$|x\rangle^n, \quad x \in \{0, 1, 2, 3, \dots, 2^n - 1\}.$$

For example, for $n = 5$,

$$\begin{aligned}
|0\rangle^5 &\longleftrightarrow |00000\rangle, \\
|1\rangle^5 &\longleftrightarrow |00001\rangle, \\
|2\rangle^5 &\longleftrightarrow |00010\rangle, \\
|8\rangle^5 &\longleftrightarrow |01000\rangle, \\
|23\rangle^5 &\longleftrightarrow |10111\rangle,
\end{aligned}$$

and, in general,

$$|x\rangle^5 \longleftrightarrow |x_4 x_3 x_2 x_1 x_0\rangle.$$

13.3.4 n Qubit Logic Gates

Quantum logic gates of order $n > 3$ are nothing more than unitary operators of order $n > 3$, which we defined above. There's no need to say anything further about a *general* n th order logic gate. Instead, let's get right down to the business of describing the *specific* example that will pervade the remainder of the course.

The n th Order Hadamard Gate, $H^{\otimes n}$

We generalize the two-qubit Hadamard gate, $H^{\otimes 2}$, to n -qubits naturally. It is the local operator that behaves like n individual unary H gates if presented with a separable input state,

$$n \text{ copies } \left\{ \begin{array}{c} \text{---} \boxed{H} \text{---} \\ \text{---} \boxed{H} \text{---} \\ \vdots \\ \text{---} \boxed{H} \text{---} \end{array} \right\},$$

or described in terms of the CBS,

$$|x_{n-1}x_{n-2} \cdots x_1x_0\rangle \longrightarrow \left\{ \begin{array}{c} \text{---} \boxed{H^{\otimes n}} \text{---} H|x_{n-1}\rangle \\ \text{---} \boxed{H^{\otimes n}} \text{---} H|x_{n-2}\rangle \\ \vdots \\ \text{---} \boxed{H^{\otimes n}} \text{---} H|x_1\rangle \\ \text{---} \boxed{H^{\otimes n}} \text{---} H|x_0\rangle \end{array} \right\}.$$

Algebraically, we developed the formula for the second order Hadamard,

$$H^{\otimes 2} |x\rangle^2 = \frac{1}{2} \sum_{y=0}^3 (-1)^{x \odot y} |y\rangle^2,$$

where “ \odot ” stood for the mod-2 dot product. It is only a matter of expending more time and graphite to prove that this turns into the higher order version,

$$H^{\otimes n} |x\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{x \odot y} |y\rangle^n.$$

Putting this result into the circuit diagram gives us

$$|x\rangle^n \text{ --- } \boxed{H^{\otimes n}} \text{ --- } \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{x \odot y} |y\rangle^n.$$

Vector Notation

We'll sometimes present the formula using vector dot products. If x and y are viewed as *vectors* of 1s and 0s, we would represent them using boldface \mathbf{x} and \mathbf{y} ,

$$x \leftrightarrow \mathbf{x} = \begin{pmatrix} x_{n-1} \\ x_{n-2} \\ \vdots \\ x_1 \\ x_0 \end{pmatrix}, \quad y \leftrightarrow \mathbf{y} = \begin{pmatrix} y_{n-1} \\ y_{n-2} \\ \vdots \\ y_1 \\ y_0 \end{pmatrix}.$$

When so expressed, the dot product between vector \mathbf{x} and vector \mathbf{y} is considered the *mod-2 dot product*,

$$\mathbf{x} \cdot \mathbf{y} = x_{n-1} y_{n-1} \oplus x_{n-2} y_{n-2} \oplus \cdots \oplus x_1 y_1 \oplus x_0 y_0.$$

This results in an equivalent form of the Hadamard gate using vector notation,

$$H^{\otimes n} |x\rangle^n = \left(\frac{1}{\sqrt{2}} \right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{x} \cdot \mathbf{y}} |y\rangle^n.$$

I'll remind you about this when the time comes.

Higher Order Basis Conversion

We'll be doing higher level basis conversions frequently, especially between the z -basis and the x -basis. Let's review and extend our knowledge.

The induced z -basis for $\mathcal{H}_{(n)}$ is

$$|0\rangle|0\rangle\cdots|0\rangle|0\rangle, \quad |0\rangle|0\rangle\cdots|0\rangle|1\rangle, \quad |0\rangle|0\rangle\cdots|1\rangle|0\rangle, \quad \dots, \quad |1\rangle|1\rangle\cdots|1\rangle|1\rangle,$$

while the induced x -basis for $\mathcal{H}_{(n)}$ is

$$|0\rangle_x|0\rangle_x\cdots|0\rangle_x|0\rangle_x, \quad |0\rangle_x|0\rangle_x\cdots|0\rangle_x|1\rangle_x, \quad |0\rangle_x|0\rangle_x\cdots|1\rangle_x|0\rangle_x, \\ \dots, \quad |1\rangle_x|1\rangle_x\cdots|1\rangle_x|1\rangle_x.$$

Using our alternate order-one x -basis notation,

$$\begin{aligned} |+\rangle &\equiv |0\rangle_x & \text{and} \\ |-\rangle &\equiv |1\rangle_x, \end{aligned}$$

the induced x -basis CBS can also be written without using the letter “ x ” as a label,

$$\begin{aligned} |+\rangle|+\rangle\cdots|+\rangle|+\rangle, \quad |+\rangle|+\rangle\cdots|+\rangle|-\rangle, \quad |+\rangle|+\rangle\cdots|-\rangle|+\rangle, \\ \dots, \quad |-\rangle|-\rangle\cdots|-\rangle|-\rangle. \end{aligned}$$

Since H converts to and from these the x and z bases in \mathcal{H} , it easy to confirm that the separable $H^{\otimes n}$ converts to-and-from these two bases in $\mathcal{H}_{(n)}$.

[**Exercise.** Do it.]

Notation. In order to make the higher order x -CBS kets of $\mathcal{H}_{(n)}$ less confusing (we need “ x ” as an encoded integer specifying the CBS state), I’m going to call to duty some non-standard notation that I introduced in our *two qubit lecture*: I’ll use the subscript “ \pm ” to indicate a CBS relative to the x -basis:

$$|y\rangle_{\pm}^n \equiv n^{\text{th}}\text{-order encoded input, } y, \text{ relative to the } x\text{-basis.}$$

That is, if y is an integer from 0 to $2^n - 1$, when you see the \pm subscript on the CBS ket you know that its binary representation is telling us which x -basis (*not* z -basis) ket it represents. So,

$$\begin{aligned} |0\rangle_{\pm}^n &= |+\rangle |+\rangle \cdots |+\rangle |+\rangle |+\rangle , \\ |1\rangle_{\pm}^n &= |+\rangle |+\rangle \cdots |+\rangle |+\rangle |-\rangle , \\ |2\rangle_{\pm}^n &= |+\rangle |+\rangle \cdots |+\rangle |-\rangle |+\rangle , \\ |3\rangle_{\pm}^n &= |+\rangle |+\rangle \cdots |+\rangle |-\rangle |-\rangle , \\ &\text{etc.} \end{aligned}$$

(Without the subscript “ \pm ,” of course, we mean the usual z -basis CBS.) This frees up the variable x for use inside the ket,

$$|x\rangle_{\pm}^n \equiv n^{\text{th}}\text{-order encoded input, } x, \text{ relative to the } x\text{-basis.}$$

With this notation, the change-of-basis is described simply as

$$H^{\otimes n} |x\rangle^n = |x\rangle_{\pm}^n$$

and in the other direction as

$$H^{\otimes n} |x\rangle_{\pm}^n = |x\rangle^n .$$

Natural Coordinates for the x -Basis

In the last section we were looking at the separable form of the CBS for an n th order Hilbert space. Let’s count for a moment. Whether we have an x -basis, z -basis or any *other* basis induced from the n component \mathcal{H} s, there are n factors in the separable factorization, i.e.,

$$\overbrace{|0\rangle |1\rangle |1\rangle \cdots |0\rangle |1\rangle}^{n \text{ components}} \quad \text{or} \quad \overbrace{|+\rangle |-\rangle |-\rangle \cdots |+\rangle |-\rangle}^{n \text{ components}}$$

But when expanded along any basis these states have 2^n components (because the product space is 2^n dimensional). From our *linear algebra* and *tensor product* lessons

we recall that a basis vector, \mathbf{b}_k , expanded along its own basis, \mathcal{B} , contains a single 1 and the rest 0s. In coordinate form that looks like

$$\mathbf{b}_k \Big|_{\mathcal{B}} = \left(\begin{array}{c} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array} \right) \Big|_{\mathcal{B}} \longleftarrow k\text{th element}.$$

This column vector is very tall in the current context, whether a z -basis ket,

$$|x\rangle^n = 2^n \text{ rows } \left\{ \left(\begin{array}{c} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array} \right) \right\} \Big|_z \longleftarrow x\text{th element},$$

an x -basis CBS ket,

$$|x\rangle_{\pm}^n = 2^n \text{ rows } \left\{ \left(\begin{array}{c} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{array} \right) \right\} \Big|_{\pm} \longleftarrow x\text{th element},$$

or more to the point of this exploration, an x -basis ket expanded along the z -basis,

$$|x\rangle_{\pm}^n = 2^n \text{ rows } \left\{ \left(\begin{array}{c} ? \\ \vdots \\ ? \\ \vdots \\ ? \end{array} \right) \right\} \Big|_z.$$

Actually, we know what those ?s are because it is the $H^{\otimes n}$ which turns the z -CBS into an x -CBS,

$$|x\rangle^n \xrightarrow{H^{\otimes n}} |x\rangle_{\pm}^n,$$

and we have already seen the result of $H^{\otimes n}$ applied to any $|x\rangle^n$, namely,

$$|x\rangle_{\pm}^n = H^{\otimes n} |x\rangle = \left(\frac{1}{\sqrt{2}} \right)^n \sum_{y=0}^{2^n-1} (-1)^{x \odot y} |y\rangle^n,$$

which, when written out looks something like

$$\frac{|0\rangle^n \pm |1\rangle^n \pm |2\rangle^n \pm |3\rangle^n \pm \dots \pm |2^{n-1}-1\rangle^n}{(\sqrt{2})^n}.$$

In coordinate form that's

$$|x\rangle_{\pm}^n = 2^n \text{ rows } \left\{ \frac{1}{(\sqrt{2})^n} \begin{pmatrix} 1 \\ \pm 1 \\ \vdots \\ \pm 1 \\ \vdots \\ \pm 1 \end{pmatrix} \right\}_z.$$

But we can do better. Not all possible sums and differences will appear in the sum, so not all possible combinations of +1 and -1 will appear in an x -basis ket's column vector (not counting the scalar factor $(\frac{1}{\sqrt{2}})^n$). An x -CBS ket, $|x\rangle_{\pm}$, will have exactly the same number of +s as -s in its expansion (and +1s, -1s in its coordinate vector) — except for $|0\rangle_{\pm}$, which has all +s (+1s). How do we know this?

We start by looking at the lowest dimension, $\mathcal{H} = \mathcal{H}_{(1)}$, where there were two easy-to-grasp x -kets in z -basis form,

$$\begin{aligned} |+\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{and} \\ |-\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \end{aligned}$$

The claim is easily confirmed here with only two kets to check. Stepping up to second order, the x -kets expanded along the z -basis were found to be

$$\begin{aligned} |+\rangle |+\rangle &= \frac{|0\rangle |0\rangle + |0\rangle |1\rangle + |1\rangle |0\rangle + |1\rangle |1\rangle}{2} \\ |+\rangle |-\rangle &= \frac{|0\rangle |0\rangle - |0\rangle |1\rangle + |1\rangle |0\rangle - |1\rangle |1\rangle}{2} \\ |-\rangle |+\rangle &= \frac{|0\rangle |0\rangle + |0\rangle |1\rangle - |1\rangle |0\rangle - |1\rangle |1\rangle}{2} \\ |-\rangle |-\rangle &= \frac{|0\rangle |0\rangle - |0\rangle |1\rangle - |1\rangle |0\rangle + |1\rangle |1\rangle}{2}, \end{aligned}$$

also easily seen to satisfy the claim. Let's make a prediction.

Little Lemma. *For an order- n $\mathcal{H}_{(n)}$, a typical x CBS ket look like*

$$\frac{|0\rangle^n \pm |1\rangle^n \pm |2\rangle^n \pm |3\rangle^n \pm \dots \pm |2^{n-1}-1\rangle^n}{(\sqrt{2})^n},$$

where — except for $|0\rangle_{\pm}^n$ which has all plus signs — the sum will always have an equal numbers of $+s$ and $-s$.

[Caution.] This doesn't mean that *every* sum with an equal number of positive and negative coefficients is necessarily an x CBS ket; there are still more ways to distribute the $+s$ and $-s$ equally than there are CBS kets, so the distribution of the plus and minus signs has to be even further restricted if the superposition above is to represent an x -basis ket. But just knowing that all x CBS tensors, when expanded along the z -basis, are “balanced” in this sense, will help us understand and predict quantum circuits.]

Proof of Lemma. We already know that the lemma is true for first and second order state spaces because we are staring directly into the eyes of the two x -bases, above. But let's see why the Hadamard operators tell the same story. The matrix for $H^{\otimes 2}$, which is used to convert the second order z -basis to an x -basis, is

$$H \otimes H = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

If we forget about the common factor $\frac{1}{2}$, it has a first column of $+1$ s, and all its remaining columns have equal numbers of $+1$ s and -1 s. If we apply $H^{\otimes 2}$ to $|0\rangle^2 = (1, 0, 0, 0)^t$ we get the first column, all $+1$ s. If we apply it to any $|x\rangle^2$, for $x > 0$, say, $|2\rangle^2 = (0, 0, 1, 0)^t$, we get one of the *other* columns, each one of which has an equal numbers of $+1$ s and -1 s.

To reproduce this claim for any higher order Hadamard, we just show that the matrix for $H^{\otimes n}$ (which generates the n th order x -basis) will also have all $+1$ s in the left column and equal number of $+1$ s and -1 s in the other columns. This is done formally by *mathematical induction*, but we can get the gist by noting how we extend the claim from $n = 2$ to $n = 3$. By definition,

$$H^{\otimes 3} = H \otimes H \otimes H = H \otimes (H^{\otimes 2}),$$

or, in terms of matrices,

$$H^{\otimes 3} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

By our technique for calculating tensor product matrices, we know that the matrix on the right will appear *four times* in the 8×8 product matrix, with the lower right copy being negated (due to the -1 in the lower right of the smaller left matrix). To

wit,

$$H^{\otimes 3} = \left(\frac{1}{\sqrt{2}} \right)^3 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}$$

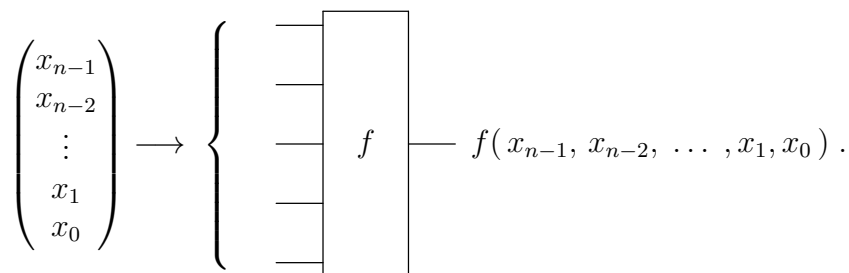
Therefore, except for the first column (all +1s), the tensor product's columns will all be a *doubling* (vertical stacking) of the columns of the balanced 4×4 (or a negated 4×4). Stacking two balanced columns above one another produces a column that is twice as tall, but still balanced. QED

[**Exercise.** Give a rigorous proof by showing how one extends an order $(n - 1)$ Hadamard matrix to an order n Hadamard matrix.]

13.3.5 Oracles for n Qubit Functions

We saw that *quantum oracles* for functions of a *single boolean variable* helped produce some early quantum algorithms. Now that we are graduating to n qubit circuits, useful in studying Boolean functions of n binary inputs, it's time to upgrade our definition of quantum oracles to cover multi-input *fs*. (We retain the assumption that our functions produce only a single Boolean output value – they're not vector functions).

We are given a *black box* for an n -input Boolean function, $f(x_{n-1}, x_{n-2}, \dots, x_1, x_0)$,

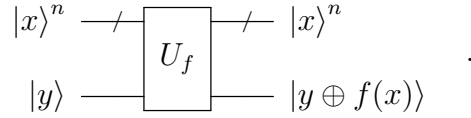


We assume that circuit theory enables us to build an $(n + 1)$ -in, $(n + 1)$ -out oracle, U_f , defined on the 2^{n+1} CBS kets

$$\left\{ |x\rangle^n |y\rangle \right\},$$

where $x \in \{0, 1, 2, \dots, 2^n - 1\}$ is in *encoded* form, $y \in \{0, 1\}$, and the circuit's

action on these CBS is



- A consequence of the definition is that U_f is its own inverse (try it).
- It is also easy to see that U_f emulates $f(x)$ by setting $y = 0$,

$$U_f(|x\rangle^n |0\rangle) = |x\rangle^n |f(x)\rangle .$$

- We assume (it does not follow from the definition) that the oracle is of the same *spatial circuit complexity* as $f(x)$, i.e., it grows in size at the same rate as f grows relative to the number of inputs, n . This is usually demonstrated to be true for common individual functions by manually presenting circuits that implement oracles for those functions.

13.4 Significant Deterministic Speed-Up: The Deutsch-Jozsa Problem

Deutsch’s algorithm enabled us to see how quantum computing could solve a problem faster than classical computing, but the speed up was limited to $2\times$, forget about the expense required to build the quantum circuit; it’s not enough to justify the investment. We now restate Deutsch’s problem for functions of n Boolean inputs and in that context call it the “Deutsch-Jozsa Problem.” We will find that the classical solution grows (in time) exponentially as n increases, not counting the increasing oracle size, while the quantum algorithm we present next has a constant time solution. This is a significant speed-up relative to the oracle and does give us reason to believe that quantum algorithms may be of great value. (The precise meaning of relative vs. absolute speed-up will be presented in our up-coming lesson devoted to *quantum oracles*, but we’ll discuss a couple different ways to measure the speed-up *informally* later today.)

We continue to study functions that have *Boolean* inputs and outputs, specifically n binary inputs and one binary output,

$$f : \{0, 1\}^n \longrightarrow \{0, 1\} .$$

The Deutsch-Jozsa Problem. *Given an unknown function,*

$$f(x_{n-1}, x_{n-2}, \dots, x_1, x_0) ,$$

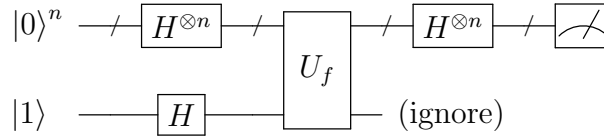
*of n inputs that we are told is either **balanced** or **constant**, determine which it is in **one query** of the quantum oracle, U_f .*

13.4.1 Deutsch-Jozsa Algorithm

The algorithm consists of building a circuit very similar to that in Deutsch's circuit and measuring the data register *once*. Our conclusion about f is the same as in the unary case: if we get a “0” the function is *constant*, if we get “1” the function is *balanced*. We'll analyze the speed-up after we prove this claim.

The Circuit

We replace the unary Hadamard gates of Deutsch's circuit with n th order *Hadamard gates* to accommodate the wider data register lines, but otherwise, the circuit layout is organized the same:

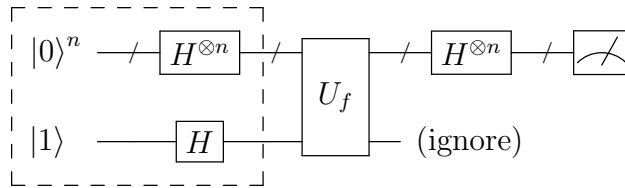


The circuit reveals that we will be

- applying *quantum parallelism* by allowing the upper Hadamard to produce a perfectly mixed input state, $|0\rangle_{\pm}^n$, into U_f 's data register, and
- using the phase kick-back trick by putting $|-\rangle$ into U_f 's target register.

Preparing the Oracle's Input: The Two Left Hadamard Gates

The first part of the Deutsch-Jozsa circuit (in the dashed box) prepares states that are needed for *quantum parallelism* and *phase kick-back* just as the lower-order Deutsch circuit did,



The H and $H^{\otimes n}$ operators take z -basis kets to x -basis kets in the first order \mathcal{H} , and the n th order $\mathcal{H}_{(n)}$ spaces, respectively, thus manufacturing a $|0\rangle_{\pm}^n$ for the *data register input* and $|-\rangle$ for the target register input,

$$\begin{aligned} |0\rangle^n &\xrightarrow{H^{\otimes n}} |0\rangle_{\pm}^n \\ |1\rangle &\xrightarrow{H} |-\rangle \end{aligned}$$

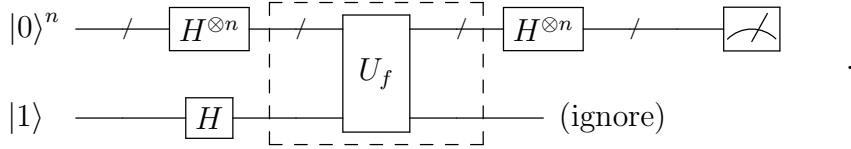
The *top* gate sets up *quantum parallelism* and the bottom sets up the *phase kick-back*. For reference, here is the algebra:

$$H^{\otimes n} |0\rangle^n = |0\rangle_{\pm}^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} |y\rangle^n = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} |y\rangle^n \quad \text{and}$$

$$H |1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle .$$

Analyzing the Oracle

Next, we consider the effect of the oracle on these two x -basis inputs (dashed box),



We'll do it in stages, as before, to avoid confusion and be sure we don't make mistakes.

Step 1. CBS Into Both Channels. When a natural CBS ket goes into both registers, the definition of U_f tells us what comes out:

$$\begin{array}{lcl} \text{Data register:} & |x\rangle^n & \xrightarrow{\quad} |x\rangle^n \\ \text{Target register:} & |y\rangle & \xrightarrow{\quad} |y \oplus f(x)\rangle \end{array} \quad ,$$

algebraically,

$$U_f(|x\rangle^n |y\rangle) = |x\rangle^n |y \oplus f(x)\rangle .$$

Step 2. CBS Into Data and Superposition into Target. Continuing on with a general CBS $|x\rangle^n$ into the data register, we next allow the superposition $|-\rangle$ into the target register.

$$\begin{aligned} U_f(|x\rangle^n |-\rangle) &= U_f\left(|x\rangle^n \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\right) = \frac{U_f(|x\rangle^n |0\rangle) - U_f(|x\rangle^n |1\rangle)}{\sqrt{2}} \\ &= \frac{|x\rangle^n |0 \oplus f(x)\rangle - |x\rangle^n |1 \oplus f(x)\rangle}{\sqrt{2}} \\ &= \frac{|x\rangle^n |f(x)\rangle - |x\rangle^n |\overline{f(x)}\rangle}{\sqrt{2}} = |x\rangle^n \left(\frac{|f(x)\rangle - |\overline{f(x)}\rangle}{\sqrt{2}}\right) . \end{aligned}$$

This amounts to

$$\begin{aligned}
 U_f(|x\rangle^n |-\rangle) &= |x\rangle^n \begin{cases} \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{when } f(x) = 0 \\ \frac{|1\rangle - |0\rangle}{\sqrt{2}}, & \text{when } f(x) = 1 \end{cases} \\
 &= |x\rangle^n (-1)^{f(x)} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right).
 \end{aligned}$$

The scalar, $(-1)^{f(x)}$ can be *kicked back*,

$$U_f(|x\rangle^n |-\rangle) = (-1)^{f(x)} |x\rangle^n \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) = \left((-1)^{f(x)} |x\rangle^n \right) |-\rangle,$$

and once again the information about $f(x)$ is converted into an overall phase factor in the data register, $(-1)^{f(x)} |x\rangle^n$.

From a circuit standpoint, we have accomplished

$$\begin{array}{lcl}
 \text{Data register:} & |x\rangle^n & \xrightarrow{\quad} \boxed{U_f} \xrightarrow{\quad} (-1)^{f(x)} |x\rangle^n \\
 \text{Target register:} & |-\rangle & \xrightarrow{\quad} \boxed{U_f} \xrightarrow{\quad} |-\rangle
 \end{array}$$

Step 3. Superpositions into Both Registers. Finally, we send the full output of $H^{\otimes n} |0\rangle^n$,

$$|0\rangle_{\pm}^n = |++\cdots+\rangle,$$

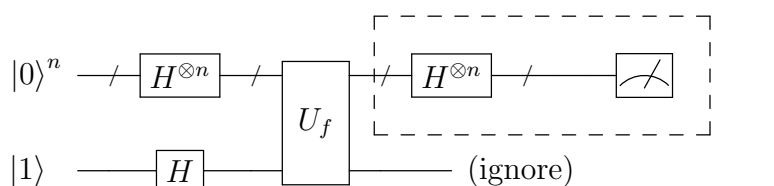
into the data register so we can process $f(x)$ for all x in a single pass and thereby leverage *quantum parallelism*. The net effect is to present the separable $|0\rangle_{\pm}^n \otimes |-\rangle$ to the oracle. Applying linearity to the last result we find

$$\begin{aligned}
 U_f(|0\rangle_{\pm}^n |-\rangle) &= U_f \left(\left(\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} |y\rangle^n \right) |-\rangle \right) \\
 &= \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} U_f(|y\rangle^n |-\rangle) \\
 &= \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} \left((-1)^{f(y)} |y\rangle^n \right) |-\rangle \\
 &= \left(\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{f(y)} |y\rangle^n \right) |-\rangle.
 \end{aligned}$$

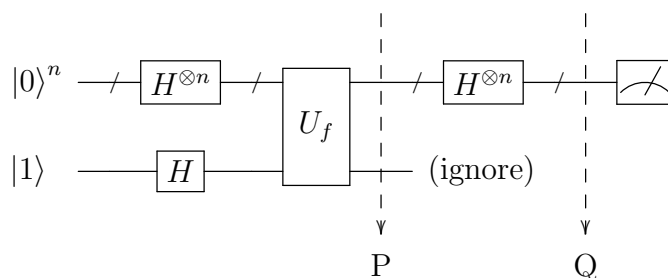
The Final Hadamard Gate

[**Warning.** The version of the argument I give next is easy enough to follow and will “prove” the algorithm, but it may leave you with a “huh?” feeling. That’s because it does not explain how one arrives at the decision to apply the final Hadamard. I’ll present a more illuminating alternative at the end of this lesson that will be more satisfying but which requires that you activate a few more little gray cells.]

We are ready to apply the n th order Hadamard gate in the upper right (dashed box),



To that end, we consider how it changes the state at access point P into a state at the final access point Q:



In this phase, we are subjecting the full $(n + 1)$ st order separable

$$\left(\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{f(y)} |y\rangle^n \right) |-\rangle$$

to the separable transformation $H^{\otimes n} \otimes \mathbf{1}$, which allows us to consider each component of the separable operation individually. We only care about the Hadamard part, since

it is the output of the data register we will test. It produces the output

$$\begin{aligned}
H^{\otimes n} \left(\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{f(y)} |y\rangle^n \right) &= \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{f(y)} H^{\otimes n} |y\rangle^n \\
&= \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} \left[(-1)^{f(y)} \left(\frac{1}{\sqrt{2^n}} \sum_{z=0}^{2^n-1} (-1)^{y \odot z} |z\rangle^n \right) \right] \\
&= \frac{1}{2^n} \sum_{z=0}^{2^n-1} \left[\underbrace{\left(\sum_{y=0}^{2^n-1} (-1)^{f(y)} (-1)^{y \odot z} \right)}_{G(z)} |z\rangle^n \right],
\end{aligned}$$

where we have regrouped the sum and defined a scalar function, $G(z)$, of the summation index z . So, the final output is an expansion along the z -basis,

$$\text{data register at access point Q} = \frac{1}{2^n} \sum_{z=0}^{2^n-1} G(z) |z\rangle^n.$$

We now look only at the coefficient, $G(0)$, of the very first CBS ket, $|0\rangle^n$. This will tell us something about the other $2^n - 1$ CBS coefficients, $G(z)$, for $z > 0$. We break it into two cases.

- **f is constant.** In this case, $f(y)$ is the same for all y , either 0 or 1; call it c . We evaluate the coefficient of $|0\rangle^n$ in the expansion, namely $\frac{G(0)}{2^n}$.

$$\frac{G(0)}{2^n} = \frac{1}{2^n} \sum_{y=0}^{2^n-1} (-1)^c (-1)^{y \odot 0} = (-1)^c \frac{2^n}{2^n} = \pm 1,$$

thereby forcing the coefficients of all other z -basis kets in the expansion to be 0 (why?). So in the *constant* case we have a CBS ket $|0\rangle^n$ at access point Q *with certainty* and are therefore guaranteed to get a reading of “0” if we measure the state.

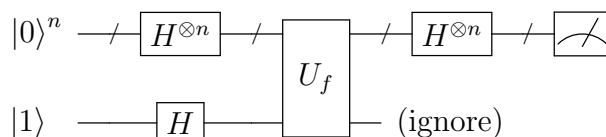
- **f is balanced.** This time the coefficient of $|0\rangle^n$ in the expansion is

$$\frac{G(0)}{2^n} = \frac{1}{2^n} \sum_{y=0}^{2^n-1} (-1)^{f(y)} (-1)^{y \odot 0} = \frac{1}{2^n} \sum_{y=0}^{2^n-1} (-1)^{f(y)},$$

but a *balanced* f promises an equal number $f(y) = 0$ and $f(y) = 1$, so the sum has an equal number of +1s and -1s, forcing it to be 0. Therefore, the probability of a measurement causing a collapse to the state $|0\rangle^n$ is 0 (the amplitude-squared of the CBS state $|0\rangle^n$). We are guaranteed to *never* get a reading of “0” when we measure the data register at access point Q.

The Deutsch-Jozsa Algorithm in Summary

We've explained the purpose of all the components in the circuit and how each plays a role in leveraging *quantum parallelism* and *phase kick-back*. The result is extremely easy to state. We run the circuit



one time only and measure the *data register output* in the natural basis.

- If we read “0” then f is constant.
- If we read “ x ” for any *other* x , (i.e., $x \in [1, 2^n - 1]$), then f is balanced.

13.4.2 Quantum vs. Classical Time Complexity

What have we actually accomplished in efficiency?

The Classical Time Complexity

In order to know the answer to the Deutsch-Jozsa problem deterministically, i.e., with 100% certainty, we would have to evaluate the function f for more than half of the possible inputs, i.e., at least

$$\frac{2^n}{2} + 1 = 2^{n-1} + 1$$

times. That is, we'd plug just over half of the possible x values into f (say, $x = 0, 1, 2, \dots, 2^{n-1} + 1$), and if they were all the same, we'd know the function must be constant. If any two were distinct, we know it is balanced. Of course, we may get lucky and find that $f(0) \neq f(1)$, in which case we can declare victory (*balanced*) very quickly, but we cannot count on that. We could be very unlucky and get the same output for the first $2^n/2$ computations, only to know the answer with certainty, on the $2^n/2 + 1$ st. (if it's the same as the others: *constant*, if not: *balanced*.)

While we have not had our official lecture on time complexity, we can see that as the number of binary inputs, n , grows, the number of required evaluations of f , $2^{n-1} + 1$, grows exponentially with n . However, when we consider that there are $N = 2^n$ encoded integers that are allowed inputs to f , then as N grows, the number of evaluations of f , $\frac{N}{2} + 1$, grows only linearly with N .

The classical problem has a solution which is exponential in n (the number of binary inputs, or linear in $N = 2^n$ (the number of integer inputs).

The Quantum Time Complexity

We have solved the problem with one evaluation of U_f which is assumed to have the same the same circuit complexity as f . Now you might say that this is a constant time solution, i.e., it does not grow at all with n , because no matter how large n is, we only need to evaluate U_f once. In that light, the quantum solution is *constant-time*; it doesn't grow at all with n . We simply measure the output of the data register, x , done using

```
if (x > 0)
```

and we're done.

You might argue that we have overstated the case, because in order to detect the output of the circuit, we have to query all n bits of the data registers to know whether we get “000...00” or an integer other than that. No computer can do that for arbitrarily large n without having an increasingly large circuit or increasingly long testing algorithm. So in practical terms, this is an evaluation of U_f followed by n one-bit queries, something that requires n *if* statements. That grows *linearly* with n or, using encoded integer counting ($N = 2^n$), *logarithmically* (even better). Either way, the quantum algorithm has a better time complexity (*linear vs. exponential* in n or *logarithmic vs. linear* in N) than its classical counterpart. So the speed-up is real.

But there's another way to view things that puts the quantum algorithm in an even more favorable light. Whether quantum or classical, the number of binary registers to test is the same: n . So we can really ignore that hardware growth when we speak of time complexity *relative to* the classical case; the quantum algorithm can be said to solve the problem in constant time *relative to* the classical algorithm.

Reminder. I'll define terms like *logarithmic*, *linear* and *exponential time complexity* in the next lecture.

The Deterministic Result

Either way you look at it, if you require 100% certainty of the solution, we have found an algorithm that is “faster” than the classical solution.

The Non-Deterministic (Probabilistic) Result

If, however, we allow a small error possibility for the classical case, as is only fair since we might expect our quantum circuit to be prone to error (a topic of the next course), then the classical algorithm grows neither exponentially with n nor linearly with N , but in fact is a constant time algorithm, just like the Deutsch-Jozsa. I'll give you an outline of the reason now, and after our probability lesson, we can make it rigorous.

Classical Algorithm Admitting a Small Error Probability $\epsilon \ll 1$.

Let's consider the following classical algorithm.

The M -and-Guess Algorithm. Let M be some positive integer (think 20). Given a Boolean function, $f(x)$ of $x \in [0, n-1]$ which is either *balanced* or *constant*, i.e., one that satisfies the Deutsch-Jozsa hypothesis, we evaluate $f(x)$ M times, each time at a random $x \in [0, n-1]$. We call each evaluation a “*trial*.” If we get two different outputs, $f(x') \neq f(x'')$, by the time we complete our M trials, we declare victory: f is balanced without a doubt. On the other hand, if we get the same output for all M trials, we declare *near* victory: We report that f is constant, with a *pretty good certainty*.

How often will we get the wrong answer using this algorithm?

The only way we can fail is if the function is *balanced* yet we declare it to be *constant* after M trials. That only happens if we are unlucky enough to get M straight 0s or M straight 1s from a balanced f .

We'll call that eventuality, the *event*

$$\mathcal{S} \wedge \mathcal{B},$$

which is a symbolic way to say, “ f was *Balanced* yet (technically AND) all trial outcomes were the *Same*.”

Since a balanced f means there is a 50-50 chance of getting a 1 or a 0 on any trial, this unlucky outcome is akin to flipping a fair coin M times and getting either all heads or all tails. As you can intuit by imagining 20 heads or 20 tails in a sequence of 20 fair coin tosses, this is quite unlikely. We'll explain it rigorously in the upcoming lesson on probability, but the answer is that the probability of this event occurring, designated $P(\mathcal{S} \wedge \mathcal{B})$, is

$$P(\mathcal{S} \wedge \mathcal{B}) = 2 \times \left(\frac{1}{2}\right)^M \times \left(\frac{1}{2}\right) = \frac{1}{2^M}.$$

The factor of 2 out front is due to the fact that the error on a balanced function can occur two different ways, all 1s or all 0s. The final factor $1/2$ is a result of an assumption – which could be adjusted if not true – that we are getting a constant function or balanced function with equal likelihood.

So we decide beforehand the error probability we are willing to accept, say some $\varepsilon \ll 1$, and select M so that

$$\frac{1}{2^M} \leq \varepsilon.$$

This will allow our classical algorithm to complete (with the same tiny error probability, ε , in a fixed number of evaluations, M , of the function f regardless of the number of inputs, n . To give you an idea,

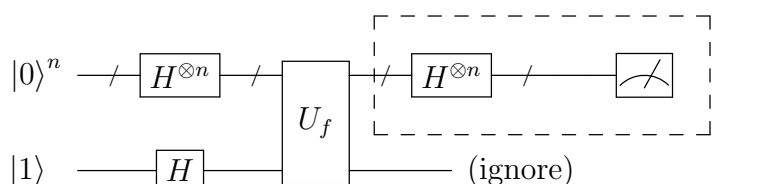
$$P(\mathcal{S} \wedge \mathcal{B}) \leq \begin{cases} 0.000001, & \text{for } M = 20 \\ 9 \times 10^{-16}, & \text{for } M = 50 \end{cases}.$$

Since the error probability does not increase with increasing n , the classical algorithm has a constant time solution, meaning that we can solve it with the same time complexity as the quantum Deutsch-Jozsa algorithm. (We will define terms like *complexity* and *constant time* precisely very soon, but you get the general idea.) Therefore, no realistic speed-up is gained using quantum computing if we accept a vanishingly small error result.

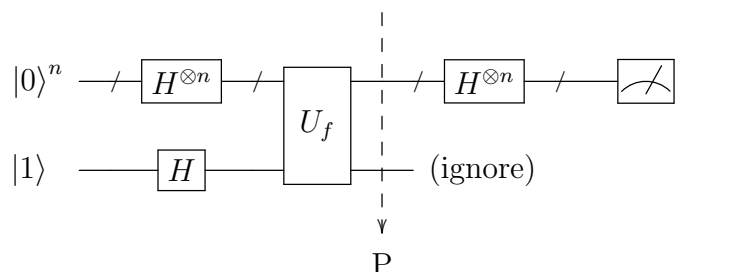
This does not diminish the importance of the deterministic solution which does show a massive computational speed increase, but we must always temper our enthusiasm with a dose of reality.

13.4.3 Alternate Proof of the Deutsch-Jozsa Algorithm

I'd like to offer a slightly more elaborate – but also more illustrative – argument for the final Hadamard gate and how we might guess that it is the correct way to complete the circuit (dashed box),



Recall that we had established that at access point P,



the *data register* was in the state

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{f(y)} |y\rangle^n.$$

The hypothesis of Deutsch-Jozsa tells us that f is either *balanced* or *constant* which has implications about this state.

1. If f is *constant*, then all the coefficients, $(-1)^{f(y)}$, are the same, and we are looking at $|0\rangle_{\pm}^n$ (or possibly -1 times this state, observationally equivalent).
2. If f is *balanced*, then half the $(-1)^{f(y)}$ are $+1$ and half are -1 . Now, our *little lemma* reminds us that this condition suggests – *but does not guarantee* – that a balanced state might, at access point P, be an x -CBS state *other* than $|0\rangle_{\pm}^n$.

The *constant* case 1, guarantees that we land in the x -CBS state, $|0\rangle_{\pm}^n$. The balanced case 2, suggests that we *might* end up in one of the other x -CBS states, $|x\rangle_{\pm}^n$, for $x > 1$. Let's pretend that in the balanced case we are lucky enough to land exactly in one of those other CBS states. If so, when we measure at access point P *along the x -basis*,

1. a measurement of “0” would imply that f was *constant*, and
2. a measurement of “ x ,” for $x > 0$, would imply that f was *balanced*.

This is because measuring any CBS state along its *own* basis gives, with 100% probability, the value of that state; that state's amplitude is 1 and all the rest of the CBS states' amplitudes are 0.

The Bad News. Alas, we are not able to assert that all balanced f s will produce x -CBS kets since there are more ways to distribute the + and – signs equally than there are x -CBS kets.

The Good News. We *do* know something that will turn out to be pivotal: a balanced f will *never* have the CBS ket, $|0\rangle_{\pm}^n$ in its expansion. Let's prove it.

If we give the data register's state at access point P the name $|\beta\rangle^n$,

$$|\beta\rangle^n = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{f(y)} |y\rangle^n,$$

then we know that a balanced f means that

$$|\beta\rangle^n = \frac{1}{\sqrt{2^n}} \left\{ \begin{pmatrix} \pm 1 \\ \pm 1 \\ \pm 1 \\ \vdots \\ \pm 1 \\ \pm 1 \end{pmatrix} \right\} \text{ equal numbers of } +1 \text{ and } -1.$$

Furthermore, its $|0\rangle_{\pm}^n$ coefficient is given by the dot-with-the-basis-ket trick (all coefficients are real, so we can use a simple dot-product),

$${}_{\pm}^n \langle 0 | \beta \rangle^n = \frac{1}{\sqrt{2^n}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \cdot \frac{1}{\sqrt{2^n}} \begin{pmatrix} \pm 1 \\ \pm 1 \\ \pm 1 \\ \vdots \\ \pm 1 \\ \pm 1 \end{pmatrix}.$$

Aside from the scalar factors $1/\sqrt{2}$, the *left vector* has all 1s, while the *right vector* has half +1s and half –1s, i.e., their dot product is 0: we are assured that there is no presence of the 0th CBS ket $|0\rangle_{\pm}^n$ in the expansion of a balanced f . ✓

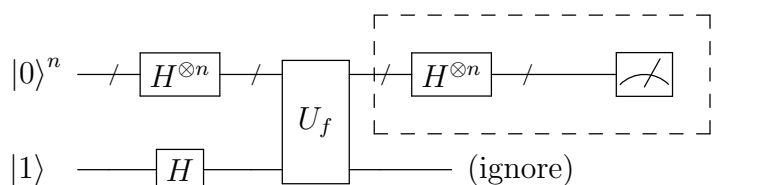
We have shown that the amplitude of the data register's $|0\rangle_{\pm}^n$ is 0 whenever f is balanced, and we already knew that its amplitude is 1 whenever f is constant, so measuring at access point P along the x -basis will

- collapse to $|0\rangle_{\pm}^n$ if f is constant, guaranteed, and
- never collapse to $|0\rangle_{\pm}^n$ if f is balanced, guaranteed.

Conclusion: If we measure along the x -basis at access point P, a reading of “0” means *constant* and a reading of “ x ,” for any $x > 0$, means *balanced*.

Measurement

The x -basis measurement we seek is nothing more than a z -basis measurement after applying the n th order $x \leftrightarrow z$ basis transforming unitary $H^{\otimes n}$. This explains the final n th order Hadamard gate in the upper right (dashed box),



After that, when we measure the data register it will either cause a collapse to the state

- $|0\rangle^n$, which corresponds to the pre- $H^{\otimes n}$ state of $|0\rangle_{\pm}^n$, and therefore indicated a *constant* f , or
- anything *else*, which corresponds to a pre- $H^{\otimes n}$ state that did not contain even “trace amounts” of $|0\rangle_{\pm}^n$ before the final gate and therefore indicates a *balanced* f .

The argument led to the same conclusion but forced us to think about the direct output of the oracle in terms of the x -basis, thereby guiding the decision to apply the final Hadamard gate. Not only that, we get a free algorithm out of it, and I’ll let you guess what it is.

A New Problem and a New Algorithm: You Discover it.

[**Exercise.** While not all balanced functions lead to x CBS kets at access point P, several do. Describe them in words or formulas.]

Let’s call the collection of functions in the last exercise \mathcal{B}_x .

[**Exercise.** How many functions are in the set \mathcal{B}_x ?]

[**Exercise.** If you are told that an unknown function is in the set \mathcal{B}_x , formulate an algorithm using the Deutsch-Jozsa circuit that will, in a single evaluation of U_f , determine the *entire truth table* of the unknown function.]

[**Exercise.** How many evaluations of the unknown function f would be needed to do this *deterministically* using a classical approach?]

[**Exercise.** If you were to allow for a non-deterministic outcome classically, would you be able to get a *constant time* solution (one whose number of evaluations of f would be independent of n for a fixed error, ε)?]

[**Exercise.** After attempting this problem, read the next section (Bernstein-Vazirani) and compare your results and algorithm with that seemingly distinct problem. Are the two truly different problems and algorithms or is there a relationship between them?]

13.5 True Non-Deterministic Speed-Up: The Bernstein-Vazirani Problem

The *quantum Deutsch-Jozsa algorithm* offers a deterministic exponential speed-up over the classical algorithm, but realistically when we accept a small error, both it and its classical alternative are “constant time,” i.e., they can be solved in a fixed time independent of the number of inputs. We saw this when analyzing the classical algorithm.

The first problem that showed a clear separation in time complexity was the *Bernstein-Vazirani problem*, in which classically, even when one accepts an error the solution still grows in time with the number of inputs, n . Meanwhile, the quantum solution is constant time – only one evaluation of the oracle, U_f , is needed regardless of the number of inputs.

Reminder. We’ll cover *probability* and *time complexity* in formal lessons, but for our current purpose we can let intuition guide our computations, just as we did earlier with classical Deutsch-Jozsa analysis.

We continue to study functions that have *Boolean* inputs and outputs, specifically n binary inputs and one binary output,

$$f : \{0, 1\}^n \longrightarrow \{0, 1\} .$$

The Bernstein-Vazirani Problem, *Given an unknown function,*

$$f(x_{n-1}, x_{n-2}, \dots, x_1, x_0) ,$$

of n inputs that are known to be defined by a mod-2 dot product with an n (binary) digit constant, a ,

$$f(x) = a \odot x ,$$

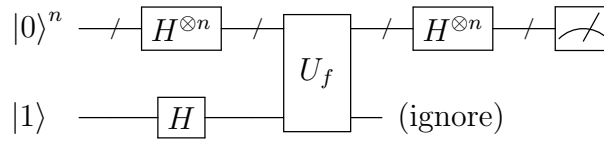
*find a in **one query** of the quantum oracle, U_f .*

13.5.1 The Bernstein-Vazirani Algorithm

The algorithm uses the same circuit with the same inputs and the same single data register measurement as Deutsch-Jozsa. However this time, instead of asking whether we see a “0” or a non-“0” at the output, we look at the full output: its value will be our desired unknown, a .

The Circuit

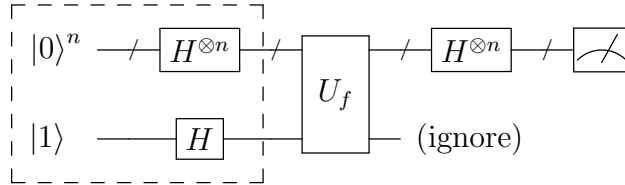
For quick reference, here it is again:



The $|0\rangle$ going into the top register provides the quantum parallelism and the $|1\rangle$ into the bottom offers a phase kick-back that transfers information about f from the target output to the data output.

Preparing the Oracle’s Input: The Two Left Hadamard Gates

Same as Deutsch-Jozsa. The first part of the circuit prepares states that are needed for *quantum parallelism* and *phase kick-back*,



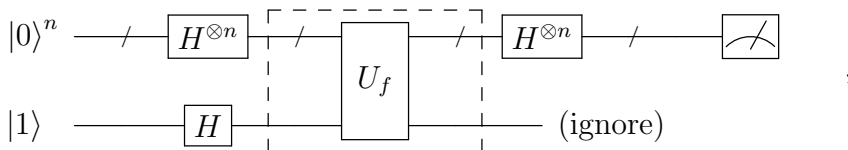
The two registers going into the oracle are, again

$$H^{\otimes n} |0\rangle^n = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} |y\rangle^n \quad \text{and}$$

$$H |1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle .$$

Analyzing the Oracle

The oracle (dashed box) has the same output,



namely,

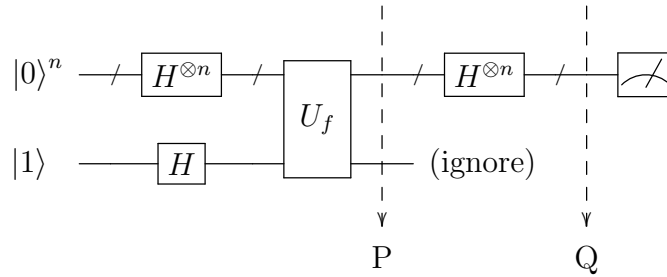
$$U_f (|0\rangle_{\pm}^n |-\rangle) = \left(\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{f(y)} |y\rangle^n \right) |-\rangle ,$$

but we can plug in the exact formula for $f(y)$,

$$U_f (|0\rangle_{\pm}^n |-\rangle) = \left(\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{a \odot y} |y\rangle^n \right) |-\rangle ,$$

The Final Hadamard Gate

We apply the final n th order Hadamard gate. At access point P,



the data register holds the ket

$$\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{a \odot y} |y\rangle^n$$

while at point Q, it ends up as (refer to the individual steps in the Deutsch-Josza derivation, with $a \odot y$ in place of $f(y)$),

$$\begin{aligned} & H^{\otimes n} \left(\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{a \odot y} |y\rangle^n \right) \\ &= \frac{1}{2^n} \sum_{z=0}^{2^n-1} \left[\underbrace{\left(\sum_{y=0}^{2^n-1} (-1)^{a \odot y} (-1)^{y \odot z} \right)}_{G(z)} |z\rangle^n \right] , \end{aligned}$$

which also defines a scalar function $G(z)$, used to simplify the analysis. So, the final output is an expansion along the z -basis,

$$\text{data register at access point Q} = \frac{1}{2^n} \sum_{z=0}^{2^n-1} G(z) |z\rangle^n .$$

Consider $G(z)$ in the two cases:

- $z = a$.

$$G(a) = \sum_{y=0}^{2^n-1} (-1)^{a \odot y} (-1)^{y \odot a} = \sum_{y=0}^{2^n-1} 1 = 2^n,$$

so the amplitudes of the CBS ket $|a\rangle$ is

$$\frac{G(a)}{2^n} = \frac{2^n}{2^n} = 1.$$

- $z \neq a$. We don't even have to sweat the computation for the amplitudes for the other kets, because once we know that $|a\rangle$ has amplitude 1, the others have to be 0. (Why?)

We have shown that at access point Q, the CBS state $|a\rangle$ is sitting in the data register. Since it *is* a CBS state, it won't collapse to anything other than what it already was, and we are guaranteed to get a reading of “a,” our sought-after n -bit binary number.

Time Complexity

Because the quantum circuit evaluates U_f only once, this is a *constant time* solution. What about the classical solution?

Deterministic. Classically we would need a full n evaluations of f in order to get all n coordinates of a . That is, we would use the input value

$$e_k \equiv \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow k\text{th element}.$$

in order to compute the k th coordinate of a based on

$$f(e_k) = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ \vdots \\ a_k \\ \vdots \\ a_{n-1} \end{pmatrix} = a_k.$$

After n passes we would have all n coordinates of a and be done. Thus, the classical algorithm *grows linearly* with the number of inputs n . This kind of growth is called *linear growth* or *linear time complexity* as it requires longer to process more inputs, but if you double the number of inputs, it only requires twice as much time. This is not as bad as the *exponential* growth of the *classical deterministic Deutsch-Jozsa* algorithm.

Alternatively, we can measure the classical deterministic solution to the current problem in terms of the *encoded integer size*, $N = 2^n$. In that case the classical algorithm is *logarithmic* in N , which doesn't sound as bad as *linear*, even though this is just a different accounting system.)

Non-Deterministic. What if, classically, we evaluate f a fixed number of times, M , and allow for some error, ε close to 0? Can we succeed if M is independent of the number of inputs, n ? No. In fact, even if allowed M to grow with n by taking $M = n - 1$, we would *still* be forced to guess at the last coordinate. This would produce a 50% error since the last coordinate could be 1 or 0 with equal probability. We can't even make a good guess (small error ε close to 0) if we skip a measely one of the n evaluations, never mind skipping the many evaluations that would be hoisted on us if we let M be constant and watched n grow far beyond M .

So in practical terms, the classical solution is not constant time, and we have a clear separation between quantum and classical solutions to the question. This is a stronger result than quantum computing provided to the Duetsch-Josza problem where, when we allowed a small error, there was no real difference between the quantum and classical solutions.

13.6 Generalized Born Rule

We can't end this lesson without providing a final generalization of **Traits #15** and **#15'**, the *Born rule* for bipartite and tripartite systems. We'll call it **Traits #15''**, the *generalized Born rule*. The sentiment is the same as its smaller order cousins.

In rough language it says that when we have a special kind of sum of separable states from two high-dimensional spaces, A and B , an A -measurement will cause the overall state to collapse to one of the separable terms, thereby selecting the B -state of that term.

13.6.1 Trait #15'' (Generalized Born Rule for $(n+m)$ th order States)

Assume that we have an $(n+m)$ th order state, $|\varphi\rangle^{n+m}$, in the product space $A \otimes B = \mathcal{H}_{(n)} \otimes \mathcal{H}_{(m)}$ with the property that $|\varphi\rangle^{n+m}$ can be written as the following kind of sum:

$$|\varphi\rangle^{n+m} = |0\rangle_A^n |\psi_0\rangle_B^m + |1\rangle_A^n |\psi_1\rangle_B^m + \cdots + |2^n - 1\rangle_A^n |\psi_{2^n-1}\rangle_B^m .$$

In this special form, notice that each term is a separable product of a distinct CBS ket from A and some general state from B , i.e., the k th term is

$$|k\rangle_A^n |\psi_k\rangle_B^m .$$

We know by QM **Trait #7** (*post-measurement collapse*), that the state of the component space $A = \mathcal{H}_{(n)}$ must collapse to one of the CBS states, call it

$$|k_0\rangle_A^n .$$

The generalized Born rule assures us that this will force the component space $B = \mathcal{H}_{(m)}$ to collapse to the matching state,

$$|\psi_{k_0}\rangle_B^m ,$$

only it will become normalized after the collapse to the equivalent

$$\frac{|\psi_{k_0}\rangle_B^m}{\sqrt{\langle\psi_{k_0}|\psi_{k_0}\rangle}} .$$

(Note that in the last expression I suppressed the superscripts m in the denominator and subscripts, B , everywhere, to avoid clutter.)

Discussion

The assumption of this rule is that the component spaces A and B are in an entangled state which can be expanded as a sum, all terms of which have A basis factors. Well, *any* state in $A \otimes B$ can be expressed this way; all we have to do is express it along the full 2^{n+m} product basis kets, then collect terms having like A -basis kets and factor out the common ket in each term. So the assumption isn't so much about the *state* $|\varphi\rangle^{n+m}$ as it is about how the state is *written*.

The next part reminds us that when an observer of the state space A takes a measurement along the natural basis, her only possible outcomes are one of the $2^n - 1$ basis kets:

$$\{ |0\rangle_A^n , |1\rangle_A^n , |2\rangle_A^n , \dots , |2^n - 1\rangle_A^n \} ,$$

so only one term in the original sum survives. That term tells us what a B -state space observer now has before him:

$$\begin{aligned} A \searrow |0\rangle^n &\implies B \searrow \frac{|\psi_0\rangle^m}{\sqrt{\langle\psi_0|\psi_0\rangle}} , \\ A \searrow |1\rangle^n &\implies B \searrow \frac{|\psi_1\rangle^m}{\sqrt{\langle\psi_1|\psi_1\rangle}} , \\ A \searrow |2\rangle^n &\implies B \searrow \frac{|\psi_2\rangle^m}{\sqrt{\langle\psi_2|\psi_2\rangle}} , \end{aligned}$$

and, in general,

$$A \searrow |k\rangle^n \implies B \searrow \frac{|\psi_k\rangle^m}{\sqrt{\langle\psi_k|\psi_k\rangle}} , \quad \text{for } k = 0, 1, \dots, 2^n - 1 .$$

This does *not* tell us what a B -state observer would measure, however, since the state he is left with, call it

$$\frac{|\psi_{k_0}\rangle^m}{\sqrt{\langle\psi_{k_0}|\psi_{k_0}\rangle}},$$

is not assumed to be a CBS ket of the B space. It is potentially a superposition of CBS kets, itself, so has a wide range of possible collapse probabilities. However, just knowing the amplitudes of the state $|\psi_{k_0}\rangle_B^m$ (after normalization) narrows down what B is likely to find. This measurement collapse, forced by an observer of A , but experienced by an observer of the entangled B , is the crux of the remainder of the course.

Size of the Superposition. We've listed the sum as potentially having the maximum number of 2^n terms, based on the underlying assumption that each term has an A -basis ket. However, it often has fewer terms, in which case the rule still applies, only then there are fewer collapse possibilities.

Role of A and B . There was nothing special about the component state space A . We could have expanded the original state,

$$|\varphi\rangle^{n+m}$$

in such a way that each term in the sum had a B -space CBS ket, $|k\rangle_B^m$ and the A -space partners were general states, $|\psi_k\rangle_A^n$.

13.7 Towards Advanced Quantum Algorithms

Even though *Bernstein – Vazirani* provided a separation between quantum and classical computing, the classical problem was still *easy*, meaning it did not grow exponentially with n ; the improvement is not yet dramatic enough to declare quantum computing a game changing technology. For that, we will need to study two landmark algorithms,

- the quantum solution to *Simon's problem*, and
- the quantum solution to *Shor's problem*.

They, in turn, require we add a little more math to our diet, so we take a small-but-interesting side trip in next time.

Chapter 14

Probability Theory

14.1 Probability in Quantum Computing

14.1.1 Probability for Classical Algorithms

In the few quantum algorithms we’ve had, the quantum circuit solved the problem in a single pass of the circuit: one query of the oracle. We didn’t need probability for that. We *did* use probability, informally, to estimate how long a classical algorithm would take to complete if we allowed for experimental error. In the *Deutsch-Jozsa* problem, when we considered a classical solution using the *M-and-guess* method, we expressed the event of failure (“all trial outcomes were the *Same* yet (read “and”) *f* is *Balanced*”) by

$$\mathcal{S} \wedge \mathcal{B},$$

and argued intuitively that the probability of this happening was

$$P(\mathcal{S} \wedge \mathcal{B}) = \frac{1}{2^M}.$$

That analysis was necessary to evaluate the worthiness of the quantum alternative’s constant-time solution.

14.1.2 Probability for Quantum Algorithms

Soon, we will study quantum algorithms that require multiple queries of a quantum circuit and result in an overall performance which is *non-deterministic*, i.e., *probabilistic*. Here’s a preview of a small section of *Simon’s quantum algorithm*:

... (previous steps) ...

- Repeat the following loop at most $n + T$ times or until we get $n - 1$ linearly independent vectors, whichever comes first.

... (description of loop) ...

- If the above loop ended after $n + T$ full passes, we failed.
- Otherwise, we succeeded. Add an n th vector, w_{n-1} , which is ...

... (following steps) ...

Estimating the probabilities of this algorithm will require more than intuition; it will require a few probability laws and formulas.

Today we'll cover those laws and formulas. We'll use them to solidify our earlier classical estimations, and we'll have them at-the-ready for the upcoming probabilistic quantum algorithms.

14.2 The Essential Vocabulary: Events vs. Probabilities

When we build and analyze quantum circuits, we'll be throwing around the terms *event* and *probability*. *Events* (word-like things) are more fundamental than *probabilities* (number-like things). Here is an informal description (rigor to follow).

14.2.1 Events

An event is something that happens, happened, will happen or might happen.

Events are described using English, Mandarin, Russian or some other natural language. They are not numbers, but descriptions. There is *no such thing* as the event "8." There *is* the event that *Salim rolls an 8 at dice*, Or *Han missed the 8 PM train*.

- The Higgs particle might decay into four muons.



- An application error occurred at 6:07 PM.

Number of events: 66			
Level	Date and Time	Source	Event ID
Error	8/16/2014 6:07:00 PM	Application Er...	1000

- Ms. Park shot a 1-under-par yesterday at Pebble Beach.



We will often use a script letter like $\mathcal{A}, \mathcal{B}, \mathcal{C}, \dots$ to designate events, as in

- “... Let \mathcal{E} be the event that two sample measurements of our quantum circuit *are equal* ...”,
- “... Let \mathcal{I} be the event that all the vectors we select at random *are linearly independent* ...”, or
- “... Let \mathcal{C} be the event that the number given to us *is relatively prime to 100* ...”.

14.2.2 Probabilities

Probabilities are the numeric likelihoods that certain events will occur. They are always positive numbers between 0 and 1, inclusive. If the probability of an event is 0 it cannot occur, if it is 1 it will occur with 100% certainly, if it is .7319 it will occur 73.19% of the time, and so on. We express the probabilities of events using $P()$ notation, like so

$$\begin{aligned} P(\text{Ms. Park will shoot a birdie on the 18th hole}) &= .338, \quad \text{or} \\ P(\text{at least one measurement is } > -13.6\text{eV}) &= .021. \end{aligned}$$

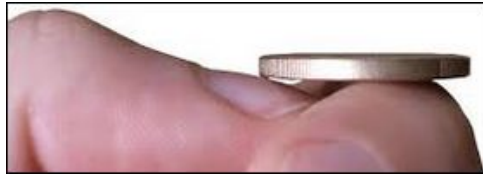
In the heat of an analysis, we’ll be using the script letters that symbolize the events under consideration. They will be defined in context, so you’ll always know what they stand for. The corresponding probabilities of the events will be expressed, then, using syntax like

$$\begin{aligned} P(\mathcal{E}) &= .99996, \\ P(\mathcal{I}) &\leq .1803, \quad \text{or} \\ P(\mathcal{C}) &> .99996. \end{aligned}$$

14.3 The Quantum Coin Flip

Many probability lessons use the familiar *coin flip* as a source of elementary examples. But we quantum computer scientists can use a phenomenon which behaves exactly like a coin flip but is much more interesting: a measurement of the quantum superposition state

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}}.$$



If we prepare this state by applying a Hadamard gate, H , to the basis state $|0\rangle$, our “coin” is waiting at the output gate. In other words, the coin is $H|0\rangle$:

$$|0\rangle \longrightarrow \boxed{H} \longrightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}}.$$

[**Exercise.** We recognize this state under an alias; it also goes by the name $|+\rangle$. Recall why.]

Measuring the output state $H|0\rangle$ is our actual “toss.” It causes the state to collapse to either $|0\rangle$ or $|1\rangle$, which we would experience by seeing a “0” or “1” on our meter.

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \longrightarrow \boxed{\text{meter}} \searrow |0\rangle \text{ or } |1\rangle$$

(Here, \searrow means *collapses to*.)

That’s equivalent to getting a *heads* or *tails*. Moreover, the probability of getting either one of these outcomes (the *eigenvalues*) is determined by the *amplitudes* of their respective CBS kets (the *eigenvectors*). Since both amplitudes are $1/\sqrt{2}$, the probabilities are

$$\begin{aligned} P(\text{measuring } 0) &= \left| \frac{1}{\sqrt{2}} \right|^2 = \frac{1}{2}, \quad \text{and} \\ P(\text{measuring } 1) &= \left| \frac{1}{\sqrt{2}} \right|^2 = \frac{1}{2}. \end{aligned}$$

So we have a perfectly good coin. Suddenly learning probability theory seems a lot more appealing and as a bonus we’ll be doing a little quantum computing along the way.

[**Exercise.** We could have used $H|1\rangle$ as our coin. Explain why.]

[**Exercise.** The above presupposes that we will measure the output state along the z -basis, $\{|0\rangle, |1\rangle\}$. What happens if, instead, we measure the same state along the x -basis, $\{|+\rangle, |-\rangle\}$. Can we use this method as a fair coin flip?]

14.4 Experimental Outcomes and the Sample Space

We now give the concepts *event* and *probability* a bit more rigor.

14.4.1 Outcomes

Every experiment or set of measurements associated with our quantum algorithms will consist of a set of *all possible outcomes*. An *outcome* is the most basic result we can imagine.

A Single Qubit Coin Flip

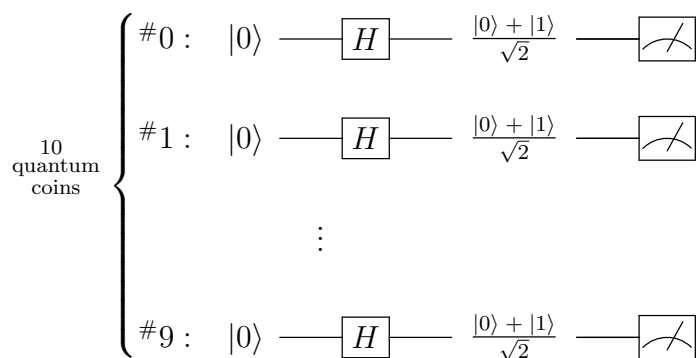
If the experiment is to prepare exactly one quantum coin, i.e., the state $H|0\rangle$, and then measure it, there are two possible outcomes:

- “Meter reads 0” \longleftrightarrow “state collapsed to $|0\rangle$ ” and
- “Meter reads 1” \longleftrightarrow “state collapsed to $|1\rangle$.”

Usually, though, we simply say that the *possible outcomes are 0 and 1*.

A Ten Qubit Coin Flip

A more involved experiment would be to prepare ten identical-but-distinct quantum coins, labeled #0 - #9 (because we are computer scientists), then measure each one, resulting in ten measurements.



Now, defining the outcomes isn't so clear.

Maybe we care about the number of 0s and 1s, but not which preparations (say, #1, #7, #9) measured 0 and which (say, #0, #2, #3, #4, #5, #6, #8) measured 1. The outcomes could be defined by 11 possible “head” counts, where *head* means getting a 0:

- “Total 0s = 0”
- “Total 0s = 1”
- “Total 0s = 2”

- ⋮
- “Total 0s = 10”

Again, we could abbreviate this by saying, “the possible outcomes are 0-10.”

One problem with this definition of “outcome” is that some are more likely than others. It is usually beneficial to define outcomes so that they are all equally – or nearly equally – likely. So, we change our outcomes to be the many *ten-tuples* consisting of

- “Results were (0,0,0,0,0,0,0,0,0,0)”
- “Results were (0,0,0,0,0,0,0,0,0,1)”
- “Results were (0,0,0,0,0,0,0,0,1,0)”
- ⋮
- “Results were (1,1,1,1,1,1,1,1,1,1)”

There are now a lot more outcomes ($2^{10} = 1024$), but they each have the same likelihood of happening. (If you don’t believe me, list the eight outcomes for three coins and start flipping.) A shorter way to describe the second breakdown of outcomes is,

“a possible outcome has the form

$$(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9),$$

where x_k is the k th measurement result.”

14.4.2 Requirements when Defining Outcomes

While there are many ways to partition all the possible results of your experiment, there is usually one obvious way that presents itself. In the ten qubit coin toss, we saw that the second alternative had some advantages. But there are actually requirements that make certain ways of dividing up the would-be outcomes *illegal*. To be *legal*, a partition of outcomes must satisfy two conditions.

1. Outcomes must be *mutually exclusive* (a.k.a. *disjoint*).
2. Outcomes must collectively represent *every possible result* of the experiment.

[**Exercise.** Explain why the two ways we defined the outcomes of the ten qubit coin toss are both *legal*.]

14.4.3 An Incorrect Attempt at Defining Outcomes

It's natural to believe that organizing the ten qubit coin toss by looking at each individual qubit outcome (such as “the 4th qubit measured a 0”) as being a reasonable partition of the experiment. After all, there are ten individual circuits. Here is the (unsuccessful) attempt at using that as our outcome set.

“The **outcomes** are to be the individual measurement results

$$\begin{aligned}\mathcal{Z}_3 &\equiv \text{measurement of qubit \#3 is 0,} \\ \mathcal{Z}_8 &\equiv \text{measurement of qubit \#8 is 0,} \\ \mathcal{O}_5 &\equiv \text{measurement of qubit \#5 is 1,} \\ \mathcal{O}_0 &\equiv \text{measurement of qubit \#0 is 1,}\end{aligned}$$

and so on.”

(\mathcal{Z} would mean that the event detects a *Zero*, while \mathcal{O} , script-*O*, means the event detects a *One*. Meanwhile, the subscript indicates which of the ten measurements we are describing.)

There are ten measurements, each one can be either zero or one, and the above organization produces 20 alleged outcomes. However, this does not satisfy the two requirements of “outcome.”

[Exercise. Explain why?]

14.4.4 Events

Definitions

Event. *An **event** is a subset of outcomes.*

Simple Event. *An event that contains exactly one outcome is called a **simple event** (a.k.a. **elementary event**).*

Please recognize that *outcomes* are not *events*. A set *containing an outcome* is an event (a *simple* one, to be precise).

Compound Event. *An event that contains more than one outcome is called a **compound event**.*

Describing Events

Events can be described either by the actual sets, using set notation, or an English (or French or Vietnamese) sentence.

Examples of *simple event* descriptions for our ten qubit coin toss experiment are

- $\{ (0, 0, 1, 1, 1, 0, 0, 0, 1, 1) \}$
- $\{ (1, 1, 1, 1, 1, 1, 0, 0, 1, 0) \}$
- $\{ (0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \}$
- “The first five qubits measure 0 and the last five measure 1.”
- “All ten qubits measure 1.”
- “Qubit #0 measures 0, and as the qubit # increases, the measurements alternate.”

Examples of *compound event* descriptions for our ten qubit coin toss experiment are

- $\{ (0, 0, 1, 1, 1, 0, 0, 0, 1, 1), (1, 1, 1, 1, 1, 1, 0, 0, 1, 0) \}$
- “The first five qubits measure 0.”
- “The fourth qubit measures 1.”
- “As the qubit # increases, the measurements alternate.”

[**Exercise.** Describe five simple events, and five compound events. Use some set notation and some natural English descriptions. You can use set notation that leverages formulas rather than listing all the members, individually.]

14.4.5 The Sample Space

Sample Space. *The **sample space** is the set of all possible outcomes. It is referred to using the Greek letter Ω .*

In our ten qubit coin toss, Ω is the set of all ordered ten-tuples consisting of 0s and 1s,

$$\Omega = \left\{ (x_0, x_1, x_2, \dots, x_9) \mid x_k \in \{0, 1\} \right\}.$$

At the other extreme, there is the *null event*.

Null Event. *The event consisting of no outcomes, a.k.a. the empty set, is the **null event**. It is represented by the symbol, \emptyset .*

14.4.6 Set Operations

Unions

One way to express Ω is as a compound event consisting of the *set union* of simple events. Let's do that for the *ten qubit coin flip* using big- \cup notation, which is just like summation notation, Σ , only for unions,

$$\Omega \equiv \bigcup_{x_k \in \{0,1\}} \{ (x_0, x_1, x_2, \dots, x_9) \} .$$

Example. We would like to represent the event, \mathcal{F} , that the first four quantum coin flips in our ten qubit experiment are all the same. One expression would be

$$\mathcal{F} = \bigcup_{w, x_k \in \{0,1\}} \{ (w, w, w, w, x_0, x_1, \dots, x_5) \} .$$

Example. We would like to represent the event, \mathcal{F}' , that the first four quantum coin flips in our ten qubit experiment are all the same, but the first five are *not* all the same. One expression would be

$$\mathcal{F}' = \bigcup_{w, x_k \in \{0,1\}} \{ (w, w, w, w, w \oplus 1, x_0, x_1, \dots, x_4) \} ,$$

where \oplus is addition mod-2.

[**Exercise.** Find a representation of the event, \mathcal{E} , that the first and last quantum coin flip in our ten qubit experiment are different.]

[**Exercise.** Find a representation of the event, \mathcal{O} , that the number of 1s in our quantum coin flip in our ten qubit experiment is *odd*.]

[**Exercise.** Find a representation of the event, \mathcal{H} , that the *Hamming weight*, i.e., the *sum* of the outcomes, of our ten qubit quantum coin flip is ≥ 6 .]

Intersections

We can use *intersections* to describe sets, too. For example, if we wanted an event \mathcal{A} in our ten qubit coin flip to be one in which both a) the first four are the same and also b) the first and last are different, we might express that using

$$\mathcal{A} = \mathcal{F} \cap \mathcal{E} ,$$

where \mathcal{F} and \mathcal{E} were defined by me (and you) a moment ago.

Of course, we could also try representing that event directly, but once we have a few compound events defined, we often leverage them to produce new ones.

[**Exercise.** Use *intersections* and the events already defined above to describe the event, \mathcal{R} , in our ten qubit coin flip in which both a) the Hamming weight has absolute value ≥ 5 and b) the sum of the 1s is odd.]

Differences

What if we wanted to discuss the ten qubit coin flip event, \mathcal{D} , which had *odd sums*, but whose *first four flips are not equal*? We could leverage our definitions of \mathcal{O} and \mathcal{F} and use *difference notation*, represented by the either of the equivalent operators: “−” or “\.” For example,

$$\begin{aligned}\mathcal{D} &= \mathcal{O} - \mathcal{F} \quad \text{or} \\ \mathcal{D} &= \mathcal{O} \setminus \mathcal{F}.\end{aligned}$$

That notation instructs the reader to start with \mathcal{O} , then remove all the events that satisfy (or are \in) \mathcal{F} .

Complements and the \neg Operator

When we start with the entire sample space Ω , and subtract and event, \mathcal{S} ,

$$\Omega - \mathcal{S},$$

we use a special term and notation: the *complement* of \mathcal{S} , written in several ways, depending on the author or context,

$$\begin{aligned}\mathcal{S}', \\ \overline{\mathcal{S}}, \\ \mathcal{S}^c, \quad \text{or} \\ \neg \mathcal{S}.\end{aligned}$$

All are usually read “not \mathcal{S} ”, and the last reprises the *logical negation operator*, \neg .

14.5 Alternate Views of the Ten Qubit Coin Flip

This is a good time to introduce other ways to view the outcomes of this ten qubit experiment that will help us when we get to some quantum algorithms later in the course.

A 10-Dimensional Space With Coordinates 0 and 1

For each outcome,

$$(x_0, x_1, x_2, \dots, x_9),$$

we look at it as a ten component vector whose coordinates are 0 or 1. The vectors can be written in either row or column form,

$$(x_0, x_1, x_2, \dots, x_9) = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_9 \end{pmatrix}.$$

An outcome is already in a vector-like format so it's not hard to see the correspondence. For example, the outcome $(0, 1, 0, \dots, 1)$ corresponds to the vector

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 1 \end{pmatrix}.$$

This seems pretty natural, but as usual, I'll complicate matters by introducing the *addition* of two such vectors. While adding vectors is nothing new, the concept doesn't seem to have much meaning when you think of them as *outcomes*. (What does it mean to “add two outcomes of an experiment?”) Let's not dwell on that for the moment but proceed with the definition. We add vectors in this space by taking their *component-wise mod-2 sum* \oplus or, equivalently, their component-wise **XOR**,

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \equiv \begin{pmatrix} 0 \oplus 1 \\ 1 \oplus 0 \\ 0 \oplus 1 \\ \vdots \\ 1 \oplus 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 0 \end{pmatrix}.$$

To make this a vector space, I'd have to tell you its scalars, (just the two numbers 0 and 1), operations on the scalars (simple multiplication, “.” and mod-2 addition, “ \oplus ”), etc. Once the details were filled in we would have the “ten dimensional vectors mod-2,” or $(\mathbb{Z}_2)^{10}$. The fancy name expresses the fact that the vectors have 10 components (the superscript 10 in $(\mathbb{Z}_2)^{10}$), in which each component comes from the set $\{0, 1\}$ (the subscript 2 in $(\mathbb{Z}_2)^{10}$). You might recall from our formal treatment of classical bits that we had a two dimensional mod-2 vector space, $\mathcal{B} = \mathbb{B}^2$, which in this new notation is $(\mathbb{Z}_2)^2$.

We can create mod-2 vectors of any dimension, of course, like the five-dimensional $(\mathbb{Z}_2)^5$ or more general n -dimensional $(\mathbb{Z}_2)^n$. The number of components 10, 5 or n , tells you the dimension of the vector space.

The Integers from 0 to $2^{10} - 1$

A second view of a *ten qubit coin flip* is that of a ten bit integer from 0 to 1023, constructed by concatenating all the results,

$$x_0x_1x_2 \cdots x_9.$$

Examples of the correspondence between these last two views are:

mod-2 vector		binary		integer
$(0, 0, \dots, 0, 0, 0)$	\leftrightarrow	$00 \cdots 000$	\leftrightarrow	0
$(0, 0, \dots, 0, 0, 1)$	\leftrightarrow	$00 \cdots 001$	\leftrightarrow	1
$(0, 0, \dots, 0, 1, 0)$	\leftrightarrow	$00 \cdots 010$	\leftrightarrow	2
$(0, 0, \dots, 0, 1, 1)$	\leftrightarrow	$00 \cdots 011$	\leftrightarrow	3
$(0, 0, \dots, 1, 0, 0)$	\leftrightarrow	$00 \cdots 100$	\leftrightarrow	4
\vdots		\vdots		\vdots
$(x_0, x_1, x_2, \dots, x_9)$	\leftrightarrow	$x_0x_1x_2, \dots, x_9$	\leftrightarrow	x

This gives us additional vocabulary to describe events. A few examples of some interesting event-descriptions are:

- All outcomes between 500 and 700. (Uses integer interpretation.)
- All outcomes divisible by 7. (Uses integer interpretation.)
- All outcomes which are *relatively prime* (a.k.a. *coprime*) to a given number. (Uses integer interpretation.)
- All outcomes that are *linearly independent of* the set (event) containing the three vectors

$$\{ (0, \dots, 0, 0, 1), (0, \dots, 0, 1, 0), (0, \dots, 1, 0, 0) \}$$

(Uses vector interpretation.) See below, for a review of *linear independence*.

The last two are of particular importance when we consider quantum period-finding.

14.6 Mod-2 Vectors in Probability Computations for Quantum Algorithms

In order to manufacture some particularly important examples for our upcoming algorithms, we will take a moment to shore-up our concept of *linear independence*, especially as it pertains to the mod-2 vector space $(\mathbb{Z}_2)^n$.

14.6.1 Ordinary Linear Independence and Spanning

Definition of Linear Independence (Review)

Linear Independence. In a vector space, a set of vectors $\{\mathbf{v}_0, \dots, \mathbf{v}_{n-1}\}$ is **linearly independent** if you cannot form a non-trivial linear combination of them which produces the zero vector, i.e.,

$$\begin{aligned} c_0\mathbf{v}_0 + c_1\mathbf{v}_1 + \dots + c_{n-1}\mathbf{v}_{n-1} &= \mathbf{0} \\ \implies \\ \text{all } c_k &= 0. \end{aligned}$$

Another way to say this is that no vector in the set can be expressed as a linear combination of the others.

[**Exercise.** Prove that the last statement is equivalent to the definition.]

[**Exercise.** Show that the zero vector can never be a member of a linearly independent set.]

[**Exercise.** Show that a *singleton* (a set consisting of any single non-zero vector) is a linearly independent set.]

Definition of Span of a Set of Vectors (Review)

The *span* of a set of vectors is the (usually larger) set consisting of all vectors that can be constructed by taking linear combinations of the original set.

The Span of a Set of Vectors. The span of a set of m vectors $\mathcal{S} = \{\mathbf{v}_k\}_{k=0}^{m-1}$ is

$$\left\{ c_0\mathbf{v}_0 + c_1\mathbf{v}_1 + \dots + c_{m-1}\mathbf{v}_{m-1} \mid c_k \text{ are scalars} \right\}.$$

The set \mathcal{S} does not have to be a linearly independent set. If it is not, then it means we can omit one or more of its vectors without reducing its span.

[**Exercise.** Prove it.]

When we say that a vector, \mathbf{w} , is *in the span of* \mathcal{S} , we mean that \mathbf{w} can be written as a linear combination of the \mathbf{v}_k s in \mathcal{S} . Again, this does not require that the original $\{\mathbf{v}_k\}$ be linearly independent.

[**Exercise.** Make this last definition explicit using formulas.]

When a vector, \mathbf{w} , is not in the span of \mathcal{S} , adding \mathbf{w} to \mathcal{S} will increase \mathcal{S} 's span.

14.6.2 Linear Independence and Spanning in the Mod-2 Sense

Because the only scalars available to “weight” each vector in a mod-2 linear combination are 0 and 1, the *span* of any set of mod-2 vectors reduces to simple sums.

The Span of a Set of Mod-2 Vectors. *The span of a set of m mod-2 vectors $\mathcal{S} = \{\mathbf{v}_k\}_{k=0}^{m-1}$ is*

$$\left\{ \mathbf{v}_{l_0} + \mathbf{v}_{l_1} + \dots + \mathbf{v}_{l_s} \mid \mathbf{v}_{l_k} \in \mathcal{S} \right\}.$$

Abstract Example

Say we have four mod-2 vectors (of any dimension),

$$\mathcal{S} = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}.$$

Because the scalars are only 0 or 1, vectors in the span are all possible sums of these vectors. If one of the vectors doesn't appear in a sum, that's just a way to say its corresponding weighting scalar is 0. If it does appear, then its weighting scalar is 1. Here are some vectors in the span of \mathcal{S} :

$$\begin{aligned} &\mathbf{0} \quad (\text{the zero vector}) \\ &\mathbf{v}_2 \\ &\mathbf{v}_0 + \mathbf{v}_3 \\ &\mathbf{v}_0 + \mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3 \\ &\mathbf{v}_1 + \mathbf{v}_1 \quad (= \mathbf{0}, \text{ also the zero vector}) \end{aligned}$$

[**Exercise.** For a mod-2 vector space, how many vectors are in the span of the empty set, \emptyset ? How many are in the span of $\{\mathbf{0}\}$? How many are in the span of a set consisting of a single (specific) non-zero vector? How many are in the span of a set of two (specific) linearly independent vectors? **Bonus:** How many in the span of a set of m (specific) linearly independent vectors? **Hint:** If you're stuck, the next examples will help sort things out.]

Concrete Example

Consider the two five-dimensional vectors in $(\mathbb{Z}_2)^5$

$$\{(1, 0, 0, 1, 1), (1, 1, 1, 1, 0)\}.$$

Their span is all possible sums (and let's not forget to include the zero vector):

$$\begin{aligned} \mathbf{0} &= (0, 0, 0, 0, 0) \\ (1, 0, 0, 1, 1) \\ (1, 1, 1, 1, 0) \\ (1, 0, 0, 1, 1) + (1, 1, 1, 1, 0) &= (0, 1, 1, 0, 1) \end{aligned}$$

Two Linearly Independent Vectors

A set of two mod-2 vectors, $\{\mathbf{v}_0, \mathbf{v}_1\}$ (of any dimension, say 10), are linearly independent in the mod-2 sense when

- neither vector is zero ($\mathbf{0}$), and
- the two vectors are distinct, i.e., $\mathbf{v}_0 \neq \mathbf{v}_1$.

[**Exercise.** Prove that this follows from the definition of linear independence using the observation that the only scalars available are 0 and 1.]

Finding a Third Independent Vector Relative to an Independent Set of Two

Consider the same two five-dimensional vectors in $(\mathbb{Z}_2)^5$

$$\{ (1, 0, 0, 1, 1), (1, 1, 1, 1, 0) \}.$$

This set is linearly independent as we know from the above observations. If we wanted to add a third vector, \mathbf{w} , to this set such that the augmented set of three vectors,

$$\{ (1, 0, 0, 1, 1), (1, 1, 1, 1, 0), \mathbf{w} \},$$

would also be linearly independent, what would \mathbf{w} have to look like? It is easiest to describe the *illegal* vectors, i.e., those \mathbf{w} which are linearly *dependent* on the two, then make sure we avoid those. For \mathbf{w} to be linearly *dependent* on these two, it would have to be in their *span*. We already computed the span and found it to be the four vectors

$$\begin{aligned} & (0, 0, 0, 0, 0), \\ & (1, 0, 0, 1, 1), \\ & (1, 1, 1, 1, 0) \quad \text{and} \\ & (0, 1, 1, 0, 1). \end{aligned}$$

Thus, a vector \mathbf{w} is linearly independent of the original two exactly when it is not in that set,

$$\mathbf{w} \notin \{ \mathbf{0}, (1, 0, 0, 1, 1), (1, 1, 1, 1, 0), (0, 1, 1, 0, 1) \},$$

or taking the complement, when

$$\mathbf{w} \in \overline{\{ \mathbf{0}, (1, 0, 0, 1, 1), (1, 1, 1, 1, 0), (0, 1, 1, 0, 1) \}}.$$

(I used the over-bar, $\overline{\mathcal{E}}$, rather than the \mathcal{E}^c notation to denote *complement*, since the c tends to get lost in this situation.)

How many vectors is this? Count all the vectors in the space ($2^5 = 32$) and subtract the four that we know to be linearly dependent. That makes $32 - 4 = 28$ such \mathbf{w} independent of the original two.

Discussion. This analysis didn't depend on which two vectors were in the original linearly independent set. If we had started with *any* two distinct non-zero vectors, they would be independent and there would be 28 ways to extend them to a set of *three* independent vectors. Furthermore, the only role played by the dimension 5 was that we subtracted the 4 from $2^5 = 32$ to get 28. If we had been working in the 7-dimensional $(\mathbb{Z}_2)^7$ and asked the same question starting with two specific vectors, we would have arrived at the conclusion that there were $2^7 - 4 = 128 - 4 = 124$ vectors independent of the first two. If we started with two linearly independent 10-dimensional vectors, we would have gotten $2^{10} - 4 = 1024 - 4 = 1020$ choices. And if we started in the space of 3-dimensional vectors, $(\mathbb{Z}_2)^3$, there would be $2^3 - 4 = 8 - 4 = 4$ independent \mathbf{w} from which to choose.

[**Exercise.** If we had two independent vectors in the 2-dimensional $(\mathbb{Z}_2)^2$, how many ways would there have been to select a third vector linearly dependent from the first two?]

14.6.3 Probability Warm-Up: Counting

We're trying to learn how to *count*, a skill that every combinatorial mathematician needs to master, and one that even we computer scientists would do well to develop.

Finding a Third Vector that is Not in the Span of Two Random Vectors

Let's describe an event that is more general than the one in the last example.

$\mathcal{E} \equiv$ the event that, after selecting two 5-dimensional mod-2 vectors \mathbf{x} and \mathbf{y} at random (from the entire space), a third random selection \mathbf{w} (also from the entire space) will not be in the span of the first two.

Notation. Since we are selecting vectors in a specific sequence, we'll use the notation

$$(\mathbf{x}, \mathbf{y})$$

to represent the event where \mathbf{x} is the first pick and \mathbf{y} is the second pick. (To count accurately, we must consider order, which is why we don't use braces: “{” or “}.”) Similarly, the selection of the third \mathbf{w} after the first two could be represented by

$$(\mathbf{x}, \mathbf{y}, \mathbf{w}).$$

How many outcomes are in event \mathcal{E} ?

This time, the first two vectors are not a fixed pair, nor are they required to be linearly independent. We simply want to know how many ways the third vector (or

“five qubit coin flip,” if you are remembering how these vectors arose) can avoid being in the span of the first two vectors (or “five qubit coin flips”). No other restrictions.

We break \mathcal{E} into two major cases:

1. \mathbf{x} and \mathbf{y} form a linearly independent set (mostly answered), and
2. \mathbf{x} and \mathbf{y} are not linearly independent. This case contains three sub-cases:
 - (i) Both \mathbf{x} and \mathbf{y} are $\mathbf{0}$,
 - (ii) exactly one of \mathbf{x} and \mathbf{y} is $\mathbf{0}$, and
 - (iii) $\mathbf{x} = \mathbf{y}$, but neither is $\mathbf{0}$.

Let’s do the larger case 2 first, then come back and finish up what we started earlier to handle case 1.

Harder Case 2. For \mathbf{x} and \mathbf{y} *not* linearly independent, we count each sub-case as follows.

- (i) There is only one configuration of \mathbf{x} and \mathbf{y} in this sub-case, namely $(\mathbf{0}, \mathbf{0})$. In such a situation, the only thing we require of \mathbf{w} is that it not be $\mathbf{0}$. There are $32 - 1 = 31$ such \mathbf{w} . Therefore, there are 31 simple events, $(\mathbf{0}, \mathbf{0}, \mathbf{w})$, in this case.
- (ii) In this sub-case there are 31 configurations of the form $(\mathbf{0}, \mathbf{y})_{\mathbf{y} \neq \mathbf{0}}$ and 31 of the form $(\mathbf{x}, \mathbf{0})_{\mathbf{x} \neq \mathbf{0}}$. That’s a total of 62 ways that random choices of \mathbf{x} and \mathbf{y} can lead us to this sub-case. Meanwhile, for each such configuration there are $32 - 2 = 30$ ways for \mathbf{w} to be different from the first two vectors. Putting it together, there are $62 \cdot 30 = 1860$ simple events in this sub-case.
- (iii) There are 31 configurations of $\mathbf{x} = \mathbf{y} \neq \mathbf{0}$ in this sub-case, namely $(\mathbf{x}, \mathbf{x})_{\mathbf{x} \neq \mathbf{0}}$. Meanwhile, for each such configuration any \mathbf{w} that is neither $\mathbf{0}$ nor \mathbf{x} will work. There are $32 - 2 = 30$ such \mathbf{w} . Putting it together, there are $31 \cdot 30 = 930$ simple events in this sub-case.

Summarizing, the number of events with \mathbf{x} and \mathbf{y} not linearly independent and \mathbf{w} not in their span is $31 + 1860 + 930 = 2821$.

Easier Case 1. This situation can be described by $(\mathbf{x}, \mathbf{y})_{\mathbf{x} \neq \mathbf{y} \neq \mathbf{0}}$. That means the first choice can’t be $\mathbf{0}$ so there are 31 possibilities for \mathbf{x} , and the second one can’t be $\mathbf{0}$ or \mathbf{x} , so there are 30 choices left for \mathbf{y} . That’s $31 \cdot 30$ ways to get into this major case. Meanwhile, there are 28 \mathbf{w} s not in the span *for each of those individual outcomes* (result of last section), providing the linearly-independent case with $31 \cdot 30 \cdot 28 = 26040$ simple events.

Combining Both Cases. We add the two major cases to get $2821 + 26040 = 28861$ outcomes in event \mathcal{E} . If you are thinking of this as three *five qubit coin flip*

outcomes – 15 individual flips, total – there are 28861 ways in which the third group of five will be linearly independent of the first two.

Is this likely to happen? How many possible flips are there in the sample space Ω ? The answer is that there are 2^{15} (or, if you prefer, 32^3) = 32768. (The latter expression comes from **3** *five qubit* events, each event coming from a set of $2^5 = \mathbf{32}$ outcomes.) That means 28861 of the 32768 possible outcomes will result in the third outcome not being in the span of the first two. A simple division tells us that this will happen 88% of the time.

A third vector selected at random from $(\mathbb{Z}_2)^5$ is far more likely to be independent of any two previously selected vectors than it is to be in their span.

This was a seat-of-the-pants calculation. We'd better develop a more reproducible methodology so we don't have to work so hard every time we need to count.

14.7 Fundamental Probability Theory

14.7.1 The Axioms

Consider the set of all possible events of an experiment,

$$\text{Events} \equiv \{ \mathcal{E} \}.$$

A *probability measure* of an experiment is a function, $P()$, with domain = *Events* and range = non-negative real numbers,

$$P : \text{Events} \longrightarrow \mathbb{R}_{\geq 0},$$

which satisfies the following three axioms:

1. All $P()$ values are non-negative (already stated),

$$P(\mathcal{E}) \geq 0.$$

2. The probabilities of *something* happening is certain,

$$P(\Omega) = 1.$$

3. The probabilities of *mutually exclusive* (*disjoint*) events, $\mathcal{E}_0, \dots, \mathcal{E}_{n-1}$, can be added,

$$P\left(\bigcup_{k=0}^{n-1} \mathcal{E}_k\right) = \sum_{k=0}^{n-1} P(\mathcal{E}_k).$$

[**Exercise.** Prove the following consequences from the above axioms:

- For any event, $P(\mathcal{E}) \leq 1$.
- $P(\emptyset) = 0$.
- If $\mathcal{E} \subseteq \mathcal{F}$, $P(\mathcal{E}) \leq P(\mathcal{F})$.]

14.7.2 Assigning Probabilities in Finite, Equiprobable, Sample Spaces

Definitions

It may not always be obvious how to assign probabilities to events even when they are simple events. However, when the sample space is finite and all simple events are *equiprobable*, we can always do it. We just count and divide.

Caution. There is no way to *prove* that simple events are equiprobable. This is something we deduce by experiment. For example, the probability of a coin flip coming up *tails* (or the z -spin measurement of an electron in state $|0\rangle_x$ being “1”) is said to be .5, but we don’t know that it is for sure. We conclude it to be so by doing lots of experiments.

Size of an Event. *The **size of an event**, written $|\mathcal{E}|$, is the number of outcomes that constitute it,*

$$|\mathcal{E}| \equiv \# \text{outcomes} \in \mathcal{E}.$$

Probability of an Event. *The probability of an event, \mathcal{E} , when all simple events are equiprobable, is*

$$P(\mathcal{E}) \equiv \frac{|\mathcal{E}|}{|\Omega|}.$$

This is not a definition, but a consequence of the axioms plus the assumption that simple events are finite and equiprobable.

Example 1

In the *ten qubit coin flip*, consider the event, \mathcal{F} , in which the first four flip results are the same (0 or all 1). We compute the probability by counting how many *simple events* (or, equivalently, how many *outcomes*) meet that criterion. What we know about these events is that the first four flips are the same, so they are either all 0 or all 1.

- **All 0.** The number of events in this case is the number of ten-tuples of the form,

$$(0, 0, 0, 0, x_0, x_1, \dots, x_5),$$

which we can see is the same as the number of integers of the form $x_0x_1x_2x_3x_4x_5$. That's 000000 through 111111, or $0 \rightarrow 63 = 64$.

- **All 1.** The number of events in this case is the number of ten-tuples of the form,

$$(1, 1, 1, 1, x_0, x_1, \dots, x_5),$$

also equal to 64.

So the number of outcomes in this event is 128. Meanwhile the sample space has 1024 outcomes, giving

$$P(\mathcal{F}) = \frac{128}{1024} = 1/8 = .125.$$

Example 2

Next, consider the event $\widehat{\mathcal{F}}$ in which the first four individual flips come up the same (= “**Example 1**”), but with the additional constraint that the fifth qubit measurement be different from those four. Now, the outcomes fall into the two categories

- **Four 0s followed by a 1.**

$$(0, 0, 0, 0, 1, x_0, x_1, \dots, x_4),$$

- **Four 1s followed by a 0.**

$$(1, 1, 1, 1, 0, x_0, x_1, \dots, x_4),$$

Again, we look at the number of possibilities for the “free range” bits x_0, \dots, x_4 , which is 32 for each of the two categories, making the number of outcomes in this event $32 + 32 = 64$, so the probability becomes

$$P(\widehat{\mathcal{F}}) = \frac{64}{1024} = 1/16 = .0625.$$

Example 3

We do a *five qubit coin flip* twice. That is, we measure five quantum states once, producing a mod-2 vector, $\mathbf{x} = (x_0, x_1, \dots, x_4)$, then repeat, getting a second vector, $\mathbf{y} = (y_0, y_1, \dots, y_4)$. It's like doing one *ten qubit coin flip*, but we are organizing

things naturally into two equal parts. Instead of the outcomes being single vectors with ten mod-2 components, outcomes are *pairs* of vectors, each member of the pair having five mod-2 components,

$$\begin{aligned}\Omega &\equiv \left\{ \left((x_0, x_1, \dots, x_4), (y_0, y_1, \dots, y_4) \right) \mid x_k, y_k \in \{0, 1\} \right\} \\ &\text{or, more concisely,} \\ \Omega &\equiv \left\{ (\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in (\mathbb{Z}_2)^5 \right\} .\end{aligned}$$

Consider the event, \mathcal{I} , in which *the two vectors form a linearly independent set*.

We've already discussed the exact conditions for a set of two mod-2 vectors to be linearly independent:

- neither vector is zero ($\mathbf{0}$), and
- the two vectors are distinct, i.e., $\mathbf{x} \neq \mathbf{y}$.

We compute $P(\mathcal{I})$ by counting events. For an outcome $(\mathbf{x}, \mathbf{y}) \in \mathcal{I}$, \mathbf{x} can be any *non-0* five-tuple, (x_0, x_1, \dots, x_4) , and \mathbf{y} must be different from both $\mathbf{0}$ and \mathbf{x} . That makes 31 vectors $\mathbf{x} \neq \mathbf{0}$, each supporting 30 linearly independent \mathbf{y} s. That's $31 \times 30 = 930$ outcomes in \mathcal{I} . $|\Omega|$ continues to be 1024, so

$$P(\mathcal{I}) = \frac{|\mathcal{I}|}{|\Omega|} = \frac{930}{1024} \approx .908 .$$

As you can see, it is very likely that two *five qubit coin flips* will produce a linearly independent set; it happens $> 90\%$ of the time.

[Exercise. Revise the above example so that we take three, rather than two, *five qubit coin flips*. Now the sample space is all *triples* of these five-tuples, $(\mathbf{x}, \mathbf{y}, \mathbf{w})$. What is the probability of the event, \mathcal{T} , that all three “flip-tuples” are linearly independent? **Hint:** We already covered the more lenient case in which \mathbf{x} and \mathbf{y} were allowed to be any two vectors and \mathbf{w} was not in their span. Repeat that analysis but exclude the cases where \mathbf{x} and \mathbf{y} form a linearly *dependent* set.]

[Exercise. Make up three interesting event descriptions in this experiment and compute the probability of each.]

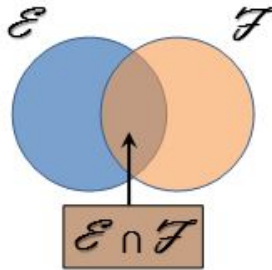
14.8 Big Theorems and Consequences of the Probability Axioms

14.8.1 Unions

The third axiom tells us about the probability of unions of *disjoint* events, $\{\mathcal{E}_k\}$, namely,

$$P\left(\bigcup_{k=0}^{n-1} \mathcal{E}_k\right) = \sum_{k=0}^{n-1} P(\mathcal{E}_k) .$$

What happens when the events are *not* mutually exclusive? A simple diagram in the case of two events tells the story. If we were to add the probabilities of two events,



\mathcal{E} and \mathcal{F} , that had a non-empty intersection we would be counting the intersection twice. To fix the error we just subtract off the probability of the intersection,

$$P(\mathcal{E} \cup \mathcal{F}) = P(\mathcal{E}) + P(\mathcal{F}) - P(\mathcal{E} \cap \mathcal{F}) .$$

When there are more than two sets, the intersections involve more combinations and get harder to write out. But the concept is the same, and all we need to know is that

$$P\left(\bigcup_{k=0}^{n-1} \mathcal{E}_k\right) = \sum_{k=0}^{n-1} P(\mathcal{E}_k) - P(\text{various intersections}) .$$

The reason this is always enough information is that we will be using the formula to bound the probability from above, so the equation, as vague as it is, clearly implies

$$P\left(\bigcup_{k=0}^{n-1} \mathcal{E}_k\right) \leq \sum_{k=0}^{n-1} P(\mathcal{E}_k) .$$

14.8.2 Conditional Probability and Bayes' Law

Conditional Probability

Very often we will want to know the probability of some event, \mathcal{E} , under the assumption that another event, \mathcal{F} , is true. For example, we might want to know the

probability that three quantum coin flips are linearly independent under the assumption that the first two are (known to be) linearly independent. The notation for the event “ \mathcal{E} given \mathcal{F} ” is

$$\mathcal{E} \mid \mathcal{F},$$

and the notation for the event’s probability is

$$P(\mathcal{E} \mid \mathcal{F}).$$

This is something we can count using common sense. Start with our formula for an event in a finite sample space of equiprobable simple events (always the setting for us),

$$P(\mathcal{E}) \equiv \frac{|\mathcal{E}|}{|\Omega|}.$$

Next, think about what it means to say “under the assumption that another event, \mathcal{F} , is true.” It means that our sample space, Ω , suddenly shrinks to \mathcal{F} , and in that smaller sample space, we are interested in the probability of the event $\mathcal{E} \cap \mathcal{F}$, so

$$P(\mathcal{E} \mid \mathcal{F}) = \frac{|\mathcal{E} \cap \mathcal{F}|}{|\mathcal{F}|}, \quad \text{whenever } \mathcal{F} \neq \emptyset.$$

Of course, if $\mathcal{F} = \emptyset$ everything is zero so there is no formula needed.

Bayes’ Law

Study this last expression until it makes sense. Once you have it, divide the top and bottom by the size of our original sample space, $|\Omega|$, to produce the equivalent identity,

$$P(\mathcal{E} \mid \mathcal{F}) = \frac{P(\mathcal{E} \cap \mathcal{F})}{P(\mathcal{F})}, \quad \text{whenever } P(\mathcal{F}) > 0.$$

This is often taken to be the definition of conditional probability, but we can view it as a natural consequence of the meaning of the phrase “ \mathcal{E} given \mathcal{F} .” It is also a simplified form of *Bayes’ law*, and I will often refer to this as *Bayes’ law* (or *rule* or *formula*), since this simple version is all we will ever need.

14.8.3 Statistical Independence

Two Independent Events

Intuitively, two events are statistically independent – or simply *independent* – if they don’t affect one another. If one is known to be true, the probability of the other is

unaffected. Stated in terms of conditional probabilities, this is nothing more than the declaration that

$$P(\mathcal{E} \mid \mathcal{F}) = P(\mathcal{E}),$$

or, looking at the mirror image,

$$P(\mathcal{F} \mid \mathcal{E}) = P(\mathcal{F}).$$

If we substitute the first of these into *Bayes' law* we find

$$P(\mathcal{E}) = \frac{P(\mathcal{E} \cap \mathcal{F})}{P(\mathcal{F})}, \quad \text{whenever } P(\mathcal{F}) > 0,$$

and rearranging gets us

$$P(\mathcal{E} \cap \mathcal{F}) = P(\mathcal{E})P(\mathcal{F}),$$

which, by the way, is true even for the degenerate case, $\mathcal{F} = \emptyset$. This is the *official definition of statistically independent events*, and working backwards you would derive the intuitive meaning that we started with. In words, two events are *independent* \Leftrightarrow

“the probability of the intersection **is equal to** the product of the probabilities.”

Multiple Independent Events

The idea carries over to any number of events, although the notation becomes thorny. It's easier to first say it in words, then show the formula. In words,

n events are *independent*

\Leftrightarrow

“the probability of the intersection [*of any subset of the n events*] **is equal to** the product of the probabilities [*of events in that subset*].”

Formulaically, we have to resort to double indexing.

n events, $\{\mathcal{E}_k\}_{k=0}^{n-1}$, are *independent*

\Leftrightarrow

$$P\left(\bigcap_{\substack{1 \leq k_0 < k_1 < \dots \\ \dots < k_l < n}} \mathcal{E}_{k_i}\right) = \prod_{\substack{1 \leq k_0 < k_1 < \dots \\ \dots < k_l < n}} P(\mathcal{E}_{k_i}).$$

14.8.4 Other Forms and Examples

I'll list a few useful consequences of the definitions and the formulas derived above. Unless otherwise noted, they are all easy to prove and you can select any of these as an exercise.

Events

$$\begin{aligned}(\mathcal{E} \cap \mathcal{F})^c &= \mathcal{E}^c \cup \mathcal{F}^c \\ \mathcal{E} \cap (\mathcal{F} \cup \mathcal{G}) &= (\mathcal{E} \cap \mathcal{F}) \cup (\mathcal{E} \cap \mathcal{G})\end{aligned}$$

Probabilities

$$\begin{aligned}P(\mathcal{E}) &= P(\mathcal{E} \cap \mathcal{F}) + P(\mathcal{E} \cap \mathcal{F}^c) \\ P(\mathcal{E} \cap \mathcal{F}) &= P(\mathcal{E} | \mathcal{F}) P(\mathcal{F})\end{aligned}$$

Example 1

In our *ten qubit coin flip*, what is the probability that the 3rd, 6th and 9th flips are identical?

We'll call the event \mathcal{E} . It is the union of two *disjoint* events, the first requiring that all three flips be 0, \mathcal{Z} , and the second requiring that all three be 1, \mathcal{O} ,

$$\begin{aligned}\mathcal{E} &= \mathcal{Z} \cup \mathcal{O}, \quad \text{with } \mathcal{Z} \cap \mathcal{O} = \emptyset \\ \implies \\ P(\mathcal{E}) &= P(\mathcal{Z}) + P(\mathcal{O}).\end{aligned}$$

There is no mathematical difference between \mathcal{Z} and \mathcal{O} , so we compute $P()$ of either one, say \mathcal{O} . It is the intersection of the three statistically independent events, namely that the 3rd, 6th and 9th flips, individually, come up 1, which we call \mathcal{O}_3 , \mathcal{O}_6 and \mathcal{O}_9 , respectively. The probability of each is, of course, .5, so we get,

$$\begin{aligned}P(\mathcal{O}) &= P(\mathcal{O}_3 \cap \mathcal{O}_6 \cap \mathcal{O}_9) \\ &= P(\mathcal{O}_3) P(\mathcal{O}_6) P(\mathcal{O}_9). \\ &= .5 \cdot .5 \cdot .5 = .125\end{aligned}$$

Plugging into the sum, we get

$$P(\mathcal{E}) = P(\mathcal{Z}) + P(\mathcal{O}) = .125 + .125 = .25.$$

Example 2

We do the *five qubit coin flip* five times. We examine the probability of the event \mathcal{I}_5 defined as “the five five-tuples are linearly independent.” Our idea is to write $P(\mathcal{I}_5)$ as a product of conditional probabilities. Let

$$\begin{aligned}\mathcal{I}_j &\equiv \text{event that the vector outcomes} \\ &\quad \#0, \quad \#1, \quad \dots \quad \#(j-1) \\ &\quad \text{are linearly-independent.}\end{aligned}$$

Our goal is to compute the probability of \mathcal{I}_5 .

(It will now be convenient to use the over-bar notation $\overline{\mathcal{E}}$ to denote *complement*.)

Combining the basic identity,

$$\begin{aligned}P(\mathcal{I}_5) &= P(\mathcal{I}_5 \cap \overline{\mathcal{I}_4}) \\ &\quad + P(\mathcal{I}_5 \cap \mathcal{I}_4)\end{aligned}$$

(**exercise:** why?), with the observation that

$$P(\mathcal{I}_5 \cap \overline{\mathcal{I}_4}) = 0$$

(**exercise:** why?), we can write

$$\begin{aligned}P(\mathcal{I}_5) &= P(\mathcal{I}_5 \cap \mathcal{I}_4) \\ &= P(\mathcal{I}_5 \mid \mathcal{I}_4) P(\mathcal{I}_4)\end{aligned}$$

(**exercise:** why?).

Now apply that same process to the right-most factor repeatedly, and you end up with

$$\begin{aligned}P(\mathcal{I}_5) &= P(\mathcal{I}_5 \mid \mathcal{I}_4) P(\mathcal{I}_4 \mid \mathcal{I}_3) \\ &\quad P(\mathcal{I}_3 \mid \mathcal{I}_2) P(\mathcal{I}_2 \mid \mathcal{I}_1) P(\mathcal{I}_1) \\ &= \prod_{j=1}^5 P(\mathcal{I}_j \mid \mathcal{I}_{j-1}),\end{aligned}$$

where, the $j = 1$ term contains the curious event, \mathcal{I}_0 . This corresponds to *no* coin-flip = *no* vector being selected or tested. It’s different from the *zero vector* = $(0, 0, 0, 0, 0)^t$, which is an actual flip possibility; \mathcal{I}_0 is no flip at all, i.e., the *empty set*, \emptyset . But we know that \emptyset is linearly independent *always* because it vacuously satisfies the condition that one cannot produce the zero vector as a linear combination of vectors from the set – since *there are no vectors* to combine. So $P(\mathcal{I}_0) = 1$. Thus, the last factor in the product,

$$P(\mathcal{I}_1 \mid \mathcal{I}_0),$$

reduces to

$$P(\mathcal{J}_1),$$

but it's cleaner to include the conditional probability in all factors when using product notation.

[**Note.** We'll be computing the individual factors in Simon's algorithm. This is as far as we need to take it today.]

14.9 Wedge and Vee Notation and Lecture Recap

Some texts and papers use the alternate notation, \wedge instead of \cap for intersections, and \vee instead of \cup for unions. When \wedge and \vee are used, \neg is typically the negation (complement) operator of choice, so the three go together. This is seen more commonly in electrical engineering and computer science than math and physics. I introduced the concepts in this lesson using more traditional \cap, \cup , and c notation but we'll use both in upcoming lectures. Normally, I will reserve \cap, \cup , and c for non-event sets (like sets of integers) and \wedge, \vee , and \neg for events.

We'll take this opportunity to repeat some of the more important results needed in future lectures using the new notation.

14.9.1 Disjoint Events

The probabilities of *mutually exclusive* (*disjoint*) events, $\mathcal{E}_0, \dots, \mathcal{E}_{n-1}$, can be added,

$$P\left(\bigvee_{k=0}^{n-1} \mathcal{E}_k\right) = \sum_{k=0}^{n-1} P(\mathcal{E}_k).$$

14.9.2 Partition of the Sample Space by Complements.

Any event, \mathcal{F} , and its complement, $\neg \mathcal{F}$, partition the space, leading to the identity

$$\begin{aligned} P(\mathcal{E}) &= P(\mathcal{E} \wedge \mathcal{F}) \\ &\quad + P(\mathcal{E} \wedge \neg \mathcal{F}). \end{aligned}$$

14.9.3 Bayes' Law.

Bayes' law (or *rule* or *formula*) in its simple, special case, can be expressed as

$$P(\mathcal{E} | \mathcal{F}) = \frac{P(\mathcal{E} \wedge \mathcal{F})}{P(\mathcal{F})}, \quad \text{whenever } P(\mathcal{F}) > 0.$$

14.9.4 Statistical Independence

n events, $\{\mathcal{E}_k\}_{k=0}^{n-1}$, are *statistically independent* \Leftrightarrow

$$P\left(\bigwedge \mathcal{E}_{k_i}\right) = \prod P(\mathcal{E}_{k_i})$$

for any subset of the indexes, $\{k_l\} \subseteq \{0, 1, \dots, n-1\}$.

This completes the probability theory results needed for the Foothill College courses in Quantum computing. You are now fully certified to study the statistical aspects of the algorithms.

14.10 Application to Deutsch-Jozsa

In a recent lecture, I gave you an informal argument that the Deutsch-Jozsa problem had a constant time solution using a classical algorithm *if we accept a small but constant error*. We now have the machinery to give a rigorous derivation and also show two different sampling strategies. It is a bit heavy handed to apply such rigorous machinery to what seemed to be a relatively obvious result, but the practice that we get provides a good warm-up for times when the answers are not so obvious. Such times await in our upcoming lessons on Simon's and Shor's algorithms.

First, a summary of the **classical Deutsch-Jozsa algorithm**:

The M -and-Guess Algorithm. Let M be some positive integer. Given a Boolean function, $f(x)$ of $x \in [0, n-1]$ which is either *balanced* or *constant*, we evaluate $f(x)$ M times, each time at a random $x \in [0, n-1]$. If we get two different outputs, $f(x') \neq f(x'')$, we declare f *balanced*. If we get the same output for all M trials, we declare f *constant*.

The only way we fail is if the function is *balanced* (event \mathcal{B}) yet all M trials produce the same output (event \mathcal{S}). This is the event

$$\mathcal{S} \wedge \mathcal{B}.$$

whose probability we must compute.

Assumption. Since we only care about the length of the algorithm as the number of possible encoded inputs, 2^n , gets very large, there is no loss of generality if we assume $M < 2^n$. To be sure, we can adjust the algorithm so that in those cases in which $M \geq 2^n$ we sample f *non-randomly* at *all* 2^n input values. In those cases we will know f completely after the M trials and will have zero error. It is only when $M < 2^n$ that we have to do the analysis.

14.10.1 Sampling *with* Replacement

The way the algorithm is stated, we are admitting the possibility of randomly selecting the same x more than once during our M trials. This has two consequences in the *balanced* case – the only case that could lead to error:

1. If we have a balanced function, it results in a very simple and consistent probability for obtaining a 1 (or 0) in any single trial, namely $P = \frac{1}{2}$.
2. It is a worst case scenario. It produces a larger error than we would get if we were to recast the algorithm to be smarter. So, if we can guarantee a small constant error for all n in *this* case it will be even better if we improve the algorithm slightly.

The algorithm as stated uses a “sampling with replacement” technique and is the version that I summarized in the original presentation. We’ll dispatch that rigorously first and move on to a smarter algorithm in the section that follows.

14.10.2 Analysis Given a Balanced f

The experiment consists of sampling f M times then announcing our conclusion: *balanced* or *constant*. We wish to compute the probability of error, which we take to mean the unlucky combination of being handed a balanced function yet observing either all 0s or all 1s in our M samples. Before attacking the desired probability of error,

$$P(\mathcal{S} \wedge \mathcal{B}) ,$$

we consider the case in which we are *given* a balanced function – the only way we could fail. The probability to be computed in *that* case is expressed by

$$P(\mathcal{S}|\mathcal{B}) .$$

So we first do our counting under the assumption that we have a *balanced function*. In this context, we are not asking about a “wrong guess,” since under this assumption there is no guessing going on; we *know* we have a balanced function. Rather, we are computing the probability that, given a balanced function, we get M identical results in our M trials.

Another way to approach this preliminary computation is to declare our sample space, Ω , to be all experimental outcomes based on a *balanced function*. Shining the light on this interpretation, we get to leave out the phrase “*given that f is balanced*” throughout this section. Our choice of sample space implies it.

Notation for Outcomes Yielding a *One*

Let \mathcal{O}_k be the event that [we observe $f(x) = 1$ on the ***kth trial***]. (Script \mathcal{O} stands for *One*.) Because f is balanced (in this section),

$$P(\mathcal{O}_k) = \frac{1}{2} .$$

Let \mathcal{O} (no index) be the unlucky event that $[we\ get\ f(x) = 1\ for\ \textit{all}\ M\ trials,$
 $k = 0, \dots, M - 1]$. This corresponds to the algebraic identity

$$\mathcal{O} = \bigcap_{k=0}^{M-1} \mathcal{O}_k.$$

Notation for Outcomes Yielding a *Zero*

Likewise, let \mathcal{Z}_k be the event that $[we\ observe\ f(x) = 0\ on\ the\ \textit{kth}\ trial]$. (Script \mathcal{Z} for \mathcal{Z} ero.) Of course, we also have

$$P(\mathcal{Z}_k) = \frac{1}{2}.$$

Finally, let \mathcal{Z} (no index) be the event that $[we\ get\ f(x) = 0\ for\ \textit{all}\ M\ trials]$.

The Probability of M Identical Readings

The event \mathcal{S} means all outcomes were the \mathcal{S} ame, i.e, we got either all 1s or all 0s. \mathcal{S} is the disjoint union,

$$\mathcal{S} = \mathcal{O} \vee \mathcal{Z} \quad (\text{disjoint}),$$

so

$$\begin{aligned} P(\mathcal{S}) &= P(\mathcal{O} \vee \mathcal{Z}) = P(\mathcal{O}) + P(\mathcal{Z}) \\ &= 2 P(\mathcal{O}) = 2 P\left(\bigcap_{k=0}^{M-1} \mathcal{O}_k\right) = 2 \prod_{k=0}^{M-1} P(\mathcal{O}_k). \end{aligned}$$

The first line uses the mutual exclusivity of \mathcal{O} and \mathcal{Z} , while the last line relies on the statistical independence of the $\{\mathcal{O}_k\}$. Plugging in $\frac{1}{2}$ for the terms in the product, we find that

$$P(\mathcal{S}) = 2 \prod_{k=0}^{M-1} \frac{1}{2} = 2 \left(\frac{1}{2}\right)^M = \frac{1}{2^{M-1}}.$$

14.10.3 Completion of Sampling with Replacement for Unknown f

We might be tempted to say that $\left(\frac{1}{2}\right)^{M-1}$ is the probability of failure in our *M-and-guess algorithm*, but that would be rash. We computed it under the assumption of a balanced f . To see the error clearly, imagine that our function provider gives us a balanced function only 1% of the time. For the other 99% of the functions, when we

guess “constant” we will be doing so on a *constant function* and will be correct; our chances of being wrong in this case are diminished to

$$.01 \times \frac{1}{2^{M-1}}.$$

To see how this comes out of our theory, we give the correct expression for a wrong guess. Let \mathcal{W} be the event consisting of a wrong guess. It happens when both f is *balanced* (\mathcal{B}) AND we get M *identical results* (\mathcal{S}),

$$\mathcal{W} = \mathcal{S} \wedge \mathcal{B}.$$

Bayes’ law gives us the proper path,

$$P(\mathcal{W}) = P(\mathcal{S} \wedge \mathcal{B}) = P(\mathcal{S}|\mathcal{B}) P(\mathcal{B}).$$

It was actually, $P(\mathcal{S}|\mathcal{B})$, not $P(\mathcal{W})$, that we computed in the last section, so we plug in that number,

$$P(\mathcal{W}) = \left(\frac{1}{2^{M-1}} \right) P(\mathcal{B}),$$

and now understand that the probability of being wrong is attenuated by $P(\mathcal{B})$, which is usually taken to be $1/2$ due to fair sampling. But understand that the world is not always fair, and we may be given more of one kind of function than the other. Using the usual value for $P(\mathcal{B})$, this yields the proper probability of guessing wrong,

$$P(\mathcal{W}) = \left(\frac{1}{2^{M-1}} \right) \left(\frac{1}{2} \right) = \frac{1}{2^M}.$$

There are many different versions of how one samples the function in a classical Deutsch-Jozsa solution, so other mechanisms might yield a different estimate. However, they are all used as upper bounds and all give the same end result: classical methods can get the right answer with exponentially small error in constant time.

14.10.4 Sampling *without* Replacement

Now, let’s make the obvious improvement of avoiding duplicate evaluations of f for the randomly selected x . Once we generate a random x , we somehow take it out of contention for the next random selection. (I won’t go into the details on how we guarantee this, but you can surmise that it will only add a *constant* time penalty – something independent of n – to the algorithm.)

[Exercise. Write an algorithm that produces M *distinct* x values at random. You can assume a random number generator that returns many more than M distinct values (with possibly some repeats), since a typical random number generator will return at least 32,768 different ints, while M is on the order of 20 or 50. **Hint:** It’s

okay if your algorithm depends on M , since M is not a function of n . It could even be on the order of M^2 or M^3 , so long as it does not rely on n .]

Intuitively, this should reduce the error because every time we remove another x whose $f(x) = 1$ from our pot, there are fewer x s capable of leading to that 1, while the full complement of $\frac{2^n}{2}$ x s that give $f(x) = 0$ are still present. Thus, chances of getting $f(x) = 1$ on future draws should diminish each trial from the original value of $\frac{1}{2}$. We prove this by doing a careful count.

Analysis Given a Balanced f

As before, we will do the heavy lifting assuming the sample space $\Omega = \mathcal{B}$, and when we're done we can just multiply by $1/2$ (which assumes equally likely reception of *balanced* vs. *constant* function.).

$M = 2$ Samples

The $\{\mathcal{O}_k\}$ are no longer independent. To see how to deal with that, we'll look at $M = 2$ which only contains the two events \mathcal{O}_0 and \mathcal{O}_1 .

The first trial is the easiest. Clearly,

$$P(\mathcal{O}_0) = \frac{1}{2}.$$

Incorporating second trial requires conditional probability. We compute with care. Our interest is $P(\mathcal{O})$, and

$$P(\mathcal{O}) = P(\mathcal{O}_1 \wedge \mathcal{O}_0) = P(\mathcal{O}_1 \mid \mathcal{O}_0) P(\mathcal{O}_0).$$

What is $P(\mathcal{O}_1 \mid \mathcal{O}_0)$? We get it by counting. There is one fewer x capable of producing $f(x) = 1$ (specifically, $\frac{2^n}{2} - 1$), and the total number of x s from which to choose is also smaller by one, now at $2^n - 1$, so

$$P(\mathcal{O}_1 \mid \mathcal{O}_0) = \frac{\frac{2^n}{2} - 1}{2^n - 1} = \frac{2^{n-1} - 1}{2^n - 1},$$

making

$$P(\mathcal{O}) = P(\mathcal{O}_1 \mid \mathcal{O}_0) P(\mathcal{O}_0) = \left(\frac{2^{n-1} - 1}{2^n - 1} \right) \left(\frac{1}{2} \right).$$

Let's write $\frac{1}{2}$ in a more complicated – but also more suggestive – way, to give

$$P(\mathcal{O}) = P(\mathcal{O}_1 \wedge \mathcal{O}_0) = \left(\frac{2^{n-1} - 1}{2^n - 1} \right) \left(\frac{2^{n-1}}{2^n} \right).$$

$M = 3$ Samples

You may be able to guess what will happen next. Here we go.

$$P(\mathcal{O}) = P(\mathcal{O}_2 \wedge \mathcal{O}_1 \wedge \mathcal{O}_0) = P\left(\mathcal{O}_2 \mid [\mathcal{O}_1 \wedge \mathcal{O}_0]\right) P(\mathcal{O}_1 \wedge \mathcal{O}_0).$$

We know the value $P(\mathcal{O}_1 \wedge \mathcal{O}_0)$, having computed it in the $M = 2$ case, so direct your attention to the factor $P\left(\mathcal{O}_2 \mid [\mathcal{O}_1 \wedge \mathcal{O}_0]\right)$. The event $\mathcal{O}_1 \wedge \mathcal{O}_0$ means there are now *two* fewer x s left that would cause $f(x) = 1$, and the total number of x s from which to choose now stands at $2^n - 2$, so

$$P\left(\mathcal{O}_2 \mid [\mathcal{O}_1 \wedge \mathcal{O}_0]\right) = \frac{\frac{2^n}{2} - 2}{2^n - 2} = \frac{2^{n-1} - 2}{2^n - 2}.$$

Substituting into our expression for $P(\mathcal{O})$ we get

$$P(\mathcal{O}) = P(\mathcal{O}_2 \wedge \mathcal{O}_1 \wedge \mathcal{O}_0) = \left(\frac{2^{n-1} - 2}{2^n - 2}\right) \left(\frac{2^{n-1} - 1}{2^n - 1}\right) \left(\frac{2^{n-1}}{2^n}\right).$$

The General Case of M Samples

It should now be clear that after M trials,

$$\begin{aligned} P(\mathcal{O}) &= P(\mathcal{O}_{M-1} \wedge \cdots \wedge \mathcal{O}_2 \wedge \mathcal{O}_1 \wedge \mathcal{O}_0) \\ &= \left(\frac{2^{n-1} - (M-1)}{2^n - (M-1)}\right) \cdots \left(\frac{2^{n-1} - 2}{2^n - 2}\right) \left(\frac{2^{n-1} - 1}{2^n - 1}\right) \left(\frac{2^{n-1}}{2^n}\right), \end{aligned}$$

or, more compactly,

$$P(\mathcal{O}) = \prod_{k=0}^{M-1} \frac{2^{n-1} - k}{2^n - k}.$$

This covers the eventually of getting all 1s when f is balanced, and if we include the alternate way to get unlucky, all 0s, we have

$$P(\mathcal{S} \text{ without replacement}) = 2 \prod_{k=0}^{M-1} \frac{2^{n-1} - k}{2^n - k}$$

Completion of Analysis for Unknown f

We already know what to do. If we believe we'll be getting about equal numbers of balanced and constant functions, we multiply the last result by $1/2$,

$$\begin{aligned} P(\mathcal{W} \text{ without replacement}) &= \left(\frac{1}{2}\right) \times 2 \prod_{k=0}^{M-1} \frac{2^{n-1} - k}{2^n - k} \\ &= \prod_{k=0}^{M-1} \frac{2^{n-1} - k}{2^n - k}. \end{aligned}$$

Bounding the Wrong-Guess Probability

The probability of guessing wrong in the “*with replacement*” algorithm (under equal likelihood of getting the two types of functions) was

$$P(\mathcal{W} \text{ with replacement}) = \prod_{k=0}^{M-1} \frac{1}{2},$$

so we want to confirm our suspicion that we have improved our chances of guessing correctly. We compare the current case with this past case and ask

$$\prod_{k=0}^{M-1} \frac{2^{n-1} - k}{2^n - k} \quad \left\{ \begin{array}{l} < \\ = \\ > \end{array} \right\} \prod_{k=0}^{M-1} \frac{1}{2} ?$$

Intuitively we already guessed that it must be less, but we can now confirm this with hard figures. We simply prove that each term in the left product is less than (or in one case, equal to) each term in the right product.

We want to show that

$$\frac{2^{n-1} - k}{2^n - k} \leq \frac{1}{2}.$$

Now is the time we realize that the old grade school fraction test they taught us from our childhood, usually called “cross-multiplication,”

$$\frac{a}{b} \leq \frac{c}{d} \Leftrightarrow ad \leq cb,$$

actually had an application. (*whoda think?*) The inequality relation involving the fractions is equivalent to asking

$$\begin{aligned} 2(2^{n-1} - k) &\leq 2^n - k ? \\ 2^n - 2k &\leq 2^n - k ? \\ -2k &\leq -k ? \end{aligned}$$

The answer is *yes*. In fact, except for $k = 0$ where both sides are equal, the LHS is *strictly less* than the RHS. Thus, the product and therefore the probability of error is actually *smaller* in the “*without replacement*” algorithm. The *constant time* result of the earlier algorithm therefore guaranteed a constant time result here, but we should have fewer wrong guesses now.

14.11 A Condition for Constant Time Complexity in Non-Deterministic Algorithms

This section will turn out to be critically important to our analysis of some quantum algorithms, ahead. I’m going to define a few terms here before our official coverage, but they’re easy and we’ll give these new terms full air time in future lectures.

14.11.1 Non-Deterministic Algorithms

At the start of this lesson, I alluded to a type of algorithm, \mathcal{A} , that was *non-deterministic* (or as I sometimes say to avoid the hyphen, *probabilistic*). This means that given any acceptably small error tolerance, $\varepsilon > 0$, the probability that \mathcal{A} will give an inaccurate result is $< \varepsilon$.

We'll say that \mathcal{A} is *probabilistic with error tolerance ε* .

There is a theorem that we can easily state and prove which will help us in our future quantum algorithms. It gives a simple condition that a *probabilistic* algorithm will succeed in *constant time*. First, a few more definitions.

14.11.2 Preview of Time Complexity – An Algorithm's Dependence on Size

Assume an algorithm, \mathcal{A} , is potentially dependent on an integer, N , that describes the size of the problem to be solved.

Example 1

\mathcal{A} might be an algorithm to sort the data and N is the number of data records to be sorted.

Example 2

\mathcal{A} might be an algorithm to determine whether a function, f , is balanced and N is the number of inputs to the function.

We'll say that the algorithm \mathcal{A} *has size N* .

This does not mean that it will take N units of time for \mathcal{A} to execute, or even that its execution time is dependent in any way on N ; it only means that \mathcal{A} solves a problem associated with a certain number, N , of data elements or inputs, and N is allowed to take on an increasingly large value, making the algorithm potentially more difficult (or not). \mathcal{A} might complete very quickly, independent of its size, or it might take longer and longer to complete, as its size, N , grows.

14.11.3 Looping Algorithms

Often in quantum computing, we have an algorithm that repeats an identical measurement (test, experiment) in a loop, and that measurement can be categorized in one of two ways: success (\mathcal{S}) or failure (\mathcal{F}). Assume that a measurement (test, experiment) only need succeed one time in any of the loop passes to end the algorithm with a declaration of victory: total success. Only if it fails after all loop passes is the algorithm considered to have failed. Finally, the events \mathcal{S} and \mathcal{F} for any one loop

pass are usually *statistically independent* of the outcomes on previous loop passes, a condition we will assume is met.

We'll say that \mathcal{A} is a *probabilistic looping algorithm* (which I may shorten to just *looping algorithm* when the context is clear).

14.11.4 Probabilistic Algorithms with Constant Time Complexity

Say we have a probabilistic looping algorithm \mathcal{A} of size N that can be shown to complete with the desired confidence (error tolerance) in a fixed number of loop passes, T , where T is independent of N .

We'll call \mathcal{A} a *constant time algorithm*.

This is a desirable property for \mathcal{A} , since it gives us virtual success no matter how large the problem size, N , is. There's a simple criterion that guarantees a looping algorithm \mathcal{A} has this nice constant time property.

14.11.5 A Simple Test for Constant Time Algorithms

We can now state the theorem, which I'll call the *CTC theorem for looping algorithms* (CTC for "constant time complexity"). By the way, this is my terminology. Don't try to use it in mixed company.

The CTC Theorem for Looping Algorithms. *Assume that \mathcal{A} is a **probabilistic, looping** algorithm having size N . If we can show that the probability of success for a **single loop pass** is bounded away from zero, i.e.,*

$$P(\mathcal{S}) \geq p > 0,$$

*with p independent of the size N , then \mathcal{A} is a **constant time algorithm**.*

Be sure not to miss the fact that the lower bound for success, p , is not dependent on N . If, for example, the probability of success were positive but happened to depend on N (say, $p(N) = 1/N$), then p would be vanishingly small for larger problem sizes: the hypothesis of the theorem, in such a situation, would not be met. On the other hand, if p meets the condition – it's not dependent on N – then no matter how small it is ($p = .000000000001$, say), the theorem tells us our algorithm is *constant time*.

Proof. Let \mathcal{S}_k be the event of success in the k th loop pass. The hypothesis is that

$$P(\mathcal{S}_k) \geq p, \quad \text{for all } k \geq 1.$$

We are allowing the algorithm to have an error with probability ε . Pick T such that

$$(1 - p)^T < \varepsilon,$$

a condition we can guarantee for large enough T since $(1 - p) < 1$. Note that p being independent of the size N implies that T is also independent of N . After having established T , we repeat \mathcal{A} 's loop T times. The event of failure of our algorithm at the completion of all T loop passes, which we'll call \mathcal{F}_{tot} , can be expressed in terms of the individual loop pass failures,

$$\mathcal{F}_{\text{tot}} = (\neg \mathcal{S}_1) \wedge (\neg \mathcal{S}_2) \wedge \cdots \wedge (\neg \mathcal{S}_T).$$

Since the events are statistically independent, we can convert this to a probability using a simple product,

$$\begin{aligned} P(\mathcal{F}_{\text{tot}}) &= P(\neg \mathcal{S}_1) P(\neg \mathcal{S}_2) \cdots P(\neg \mathcal{S}_T) \\ &= (1 - p)^T < \varepsilon. \end{aligned}$$

We have shown that we can get \mathcal{A} to succeed with failure probability $< \varepsilon$ if we allow \mathcal{A} 's loop to proceed a fixed number of times, T , independent of its size, N . This is the definition of *constant time complexity*. QED

Explicit Formula for T

To solve for T explicitly, we turn our condition on the integer T into an equality on a real number t ,

$$(1 - p)^t = \varepsilon,$$

solve for t by taking the \log_{1-p} of both sides,

$$t = \log_{1-p}(\varepsilon),$$

then pick any integer $T > t$. Of course, taking a log having a non-standard base like $1 - p$, which is some real number between 0 and 1, is not usually a calculator-friendly proposition; calculators, not to mention programming language math APIs, tend to give us the option of only \log_2 or \log_{10} . No problem, because ...

[**Exercise.** Show that

$$\log_A x = \frac{\log x}{\log A},$$

where logs on the RHS are *both* base 2, *both* base 10, or *both* any other base for that matter.]

Using the exercise to make the condition on t a little more palatable,

$$t = \frac{\log(\varepsilon)}{\log(1 - p)},$$

and combining that with the need for an integer $T > t$, we offer a single formula for T ,

$$T = \left\lfloor \frac{\log(\varepsilon)}{\log(1 - p)} \right\rfloor + 1,$$

where $\lfloor x \rfloor$ is notation for the *floor* of x , or the greatest integer $\leq x$.

Examples with Two Different ps for Success

It's important to realize that we don't care whether the *event of success* in each loop pass, \mathcal{S} , is highly probable or highly improbable. We only care that it is bounded away from 0 by a fixed amount, p , independent of the size of the algorithm, N . Two examples should crystallize this. In both examples, we assume that we would like to assure an error probability less than $\varepsilon = .000001 = 10^{-6}$ – that's one in a million. How many times do we have to loop?

Example 1. $P(\mathcal{S}) = .002$ (Very Improbable). We'll use \log_{10} , since my calculator likes that.

$$\begin{aligned} T &= \left\lfloor \frac{\log(10^{-6})}{\log(.998)} \right\rfloor + 1 = \left\lfloor \frac{-6}{-.00086945871262889} \right\rfloor + 1 \\ &= \lfloor 6900.8452188 \rfloor + 1 = 6900 + 1 = 6901, \end{aligned}$$

or, if we want to pick a nice round number, 7000 loop passes.

Example 2. $P(\mathcal{S}) = .266$ (Reasonably Probable). If we have a more reasonable chance of success for each measurement in a single pass – a little better than 25% – we would expect to get the same level of confidence, error $\varepsilon = 10^{-6}$, with far fewer loop passes. Let's see how much faster the algorithm “converges.”

$$\begin{aligned} T &= \left\lfloor \frac{\log(10^{-6})}{\log(.734)} \right\rfloor + 1 = \left\lfloor \frac{-6}{-0.134303940083929467} \right\rfloor + 1 \\ &= \lfloor 44.67478762 \rfloor + 1 = 44 + 1 = 45, \end{aligned}$$

or, rounding up, 50 loop passes.

Although our required number of loop passes is understandably sensitive to the desired accuracy of our algorithm, it is not dependent on the amount of data or number of inputs on which our algorithm works. The algorithm terminates in *constant time* regardless of the amount of data *within* each of two cases above.

Chapter 15

Computational Complexity

15.1 Computational Complexity in Quantum Computing

Algorithms are often described by their *computational complexity*, a quantitative expression about how an algorithm's time and space requirements grow as the data set or problem size grows. We've already seen examples where the separation between the quantum and classical algorithms favors quantum, but the arguments given at the time were a little hand-wavy; we didn't have formal definitions to lean on. We filled in our loosely worded probability explanations by supplying a lesson on basic *probability theory*, and now it's time to do the same with *computational complexity*.

15.2 An Algorithm's Sensitivity to Size

We design an algorithm to be applied to an unknown function, a number or a set of data. It might determine the period of the function, factor the number or sort the data. The number of inputs to the function (or size of the number to be factored or amount of data to process) could be anything from one to ten trillion or beyond. There are many possible solutions to our problem, but once we settle on one that seems good, we ask the question, "how does our algorithm's *running time* increase as N increases?" where N represents the number of inputs (or number to be factored or amount of data) on which the algorithm operates.

15.2.1 Some Examples of Time Complexity

The way an algorithm's running time grows as the size of the problem increases – its "growth rate" – is known in scientific disciplines as its *time complexity*. To get a feel for different growth rates, let's assume that it acts on *data sets* (not *function inputs* or *a number to be factored*). The following examples don't *define* the various time

complexities we are about to study, but they do give you taste of their consequences.

Constant Time Complexity

- The algorithm does not depend on the size of the data set, N . It appears to terminate in a fixed running time (C seconds) no matter how large N is. Such an algorithm is said to have *constant time complexity* (or be a *constant-time* algorithm).

Polynomial Time Complexity

- The algorithm takes C seconds to process N data items. We double N , and the running time seems to *double* – it takes $2C$ seconds to process $2N$ items. If we apply it to $8N$ data items, the running time seems to take $8C$ seconds. Here, the algorithm exhibits *linear time complexity* (or be a *linear* algorithm).
- The algorithm takes C seconds to process N data items. We double N , and now the running time seems to *quadruple* – it takes $C(2^2) = 4C$ seconds to process $2N$ items. If we apply it to $8N$ data items the running time seems to take $C(8^2) = 64C$ seconds. Now, the algorithm will likely have *quadratic time complexity* (or be a *quadratic* algorithm).
- The algorithm takes C seconds to process N data items. We double N , and now the running time seems to increase by a factor of 2^3 – it takes $C(2^3) = 8C$ seconds to process $2N$ items. If we apply it to $8N$ data items the running time seems to take $C(8^3) = 512C$ seconds. Now, the algorithm will likely have *cubic time complexity*.

The previous examples – constant, linear, quadratic – all fall into the general category of *polynomial time complexity* which includes growth rates limited by some fixed power of N (N^2 for *quadratic*, N^3 for *cubic*, N^5 for *quintic*, etc.).

Non-Polynomial Time Complexity

Sometimes we can't find a p such that N^p reflects the growth in time as the data grows. We need a different functional form. Examples include *logarithmic* and *exponential* growth. I won't give an example of the former here – we'll define it rigorously in the next section. But here's what exponential growth feels like.

- The algorithm processes N items in C seconds (assume N is large enough that $C > 1$). When we *double* N the running time seems to take C^2 seconds. If we apply it to $3N$ data items the running time takes C^3 seconds and to $7N$ data items the running time takes C^7 . This is *longer/slower/worse* than polynomial complexity for any polynomial size. Now, the algorithm probably has *exponential* growth).

(This last example doesn't describe *every* exponential growth algorithm, by the way, but an algorithm satisfying this for some $C > 1$ would likely be exponential.)

15.2.2 Time Complexity vs. Space Complexity

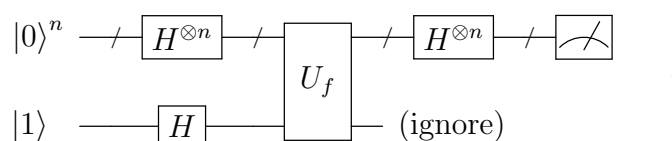
I've limited our discussion to the realm of time. However, if we were allowed to make hardware circuitry that grows as the data set grows (or utilize a larger number of processors from a very large pool of existing farm of computers) then we may not need more time. We would be trading *time complexity* for *space complexity*.

The general term that describes the growth rate of the algorithm in both *time* and *space* is *computational complexity*.

For the purposes of our course, we'll usually take the hardware out of the picture when measuring complexity and only consider time. There are two reasons for this:

- Our interest will usually be in *relative speed-up* of quantum over classical methods. For that, we will be using a hardware *black box* that does the bulk of the computation. We will be asking the question, "how much time does the quantum algorithm save us over the classical algorithm when we use the same – or spatially equivalent – black box in both regimes?"
- Even when we take space into consideration, the circuitry in our algorithms for this course will grow *linearly* at worst, (often *logarithmically*) and we have much bigger fish to fry. We're trying to take a very expensive *exponential algorithm classically* and find a *polynomial algorithm using quantum computation*. Therefore, the linear or logarithmic growth of the hardware will be overpowered by the time cost in both cases and therefore ignorable.

For example, the circuit we used for both *Deutsch-Josza* and *Bernstein-Vazirani*,



had $n + 1$ inputs (and outputs), so it grows *linearly* with n (and only *logarithmically* with the encoded $N = 2^n$). Furthermore, since this is true for both classical and quantum algorithms, such growth can be ignored when we *compare* the two regimes.

15.2.3 Notation

To kick things off, we establish the following symbolism for the *time taken by an algorithm to deal with a problem of size N* (where you can continue to think of N as

the amount of data, while keeping in mind it might be the number of inputs or the size of a number to be factored).

$$T_Q(N) \equiv \text{time required by algorithm } Q \text{ to process } N \text{ elements.}$$

We now explore ways to quantify $T_Q(N)$.

15.3 *Big-O* Growth

15.3.1 Conflicting Ways to Measure an Algorithm

When you begin to parse the meaning of an algorithm's *running time*, you quickly come to a realization. Take a simple *linear search* algorithm on some random (unsorted) data array, `myArray[k]`. We will plow through the array from element 0 through element $N - 1$ and stop if and when we find the *search key*, x :

```

for ( k = 0,  found = false ;
      k < N  &&  !found ;
      k++)
{
    if ( x == myArray[k] )
    {
        found = true;
        foundPosition = k;
    }
}

if ( found )
    cout << x << " found at position " << k << endl;

```

If x is in location `myArray[0]`, the algorithm terminates instantly independent of the array size: *constant time*. If it is in the last location, `myArray[N-1]` (or not in the list at all), the algorithm will take $N - 1$ steps to complete, a time that increases *linearly* with N .

If we can't even adjudicate the speed of an algorithm for a single data set, how do we categorize it in terms of all data sets of a fixed size? We do so by asking three of four types of more nuanced questions. The most important category of question is "what happens in the worst case?" This kind of time complexity is called *big-O*.

To measure *big-O* time complexity, we stack the cards against ourselves by constructing the worst possible data set that our algorithm could encounter. In the search example above, that would be the case in which x was in that *last position* searched. This clears up the ambiguity about where we might find it and tells us that the *big-O* complexity is going to be *linear*. But wait – we haven't *officially* defined what it means to be *linear*.

15.3.2 Definition of *Big-O*

Let's say we have an ordinary function of N , call it $f(N)$. $f(N)$ could be anything. One example is

$$f(N) = N^2 + 3N + 75.$$

Another might be

$$f(N) = N \log N + 2.$$

We wish to compare the growth rate of $T_Q(N)$ with the function $f(N)$. We say that

$$\begin{aligned} T_Q(N) &= O(f(N)) \\ &\iff \\ T_Q \text{ grows no faster than } f(N). \end{aligned}$$

We are giving our timing on algorithm Q an *upper bound* using the function $f(N)$. In words, we say that “*the timing for algorithm Q is big- O of f* ”. But we still have a loose end; the wording “grows no faster than” is not very rigorous, so let's clear that up.

$$\begin{aligned} T_Q(N) &= O(f(N)) \\ &\iff \\ \text{there exist positive constants } n_0 \text{ and } c \text{ such that} \end{aligned}$$

$$T_Q(N) \leq c|f(N)|, \quad \text{for all } N \geq n_0.$$

This means that while $T_Q(N)$ might start out being much greater than $c|f(N)|$ for small N , eventually it “improves” as N increases to the extent that $T_Q(N)$ will become and stay $\leq c|f(N)|$ for all N once we get past $N = n_0$.

Note. Since our comparison function, $f(x)$, will always be non-negative, I will drop the absolute value signs in many of the descriptions going forward.

15.3.3 Common Terminology for Certain *Big-O* Growth Rates

Now we can officially define the terms like *quadratic* or *exponential* growth.

Descriptions of Growth Functions

$f(N)$	Informal term for $O(f(N))$
1	constant
$\log N$	logarithmic
$\log^2 N$	log-squared
N	linear
$N \log N$	(no special term)
N^2	quadratic
N^3	cubic
$N^k,$ $k \geq 0$, integer	polynomial
$N^k \log N^l,$ $k, l \geq 0$, integer	(also) polynomial
2^N	exponential
$N!$	factorial

15.3.4 Factors and Terms We Can Ignore

You'll note that the table of common *big-O* terminology doesn't contain functions like $10N^2$ or $N^2 + N$. There's a good reason for this.

Ignore a Constant Factor K

Instead of declaring an algorithm to be $O(1000N)$, we will say it is $O(N)$ (i.e., *linear*). Instead of $O(1.5N^3)$ we will call it $O(N^3)$ (i.e., *cubic*). Here's why.

Theorem. *If $T_Q(N) = O(Kf(N))$, for some constant K , then $T_Q(N) = O(f(N))$.*

The theorem says we can ignore constant factors and use the simplest version of the function possible, i.e., $O(N^2)$ vs. $O(3.5N^2)$.

[**Exercise.** Prove it. **Hint.** It's easy.]

For Polynomial *big-O* complexity, Ignore all but the Highest Power Term

We only need monomials, never binomials or beyond, when declaring a *big-O*. For example if T_Q is $O(N^4 + N^2 + N + 1)$ it's more concisely $O(N^4)$. Here's why.

Lemma. *If j and k are non-negative integers satisfying $j < k$, then $N^j \leq N^k$, for all $N \geq 1$.*

[**Exercise.** Prove it. **Hint.** It's easy.]

Now we prove the main claim of the section: for *big-O*, we can ignore all but the highest power term in a polynomial.

Theorem. *If*

$$T_Q = O\left(\sum_{j=0}^k a_j N^j\right)$$

then

$$T_Q = O(N^k) .$$

Proof. We have positive constants n_0 and c such that

$$T_Q(N) \leq c \left| \sum_{j=0}^k a_j N^j \right| , \quad \text{for all } N \geq n_0 .$$

Pick a positive number a greater than all the coefficients a_j , which will allow us to write

$$c \left| \sum_{j=0}^k a_j N^j \right| \leq ca \sum_{j=0}^k N^j .$$

Next, if the n_0 happened to be $= 0$ in the *big-O* criteria, “upgrade” it to 1. Now n_0 is ≥ 1 so we can apply the lemma and replace all the N^j with the higher power, N^k ,

$$ca \sum_{j=0}^k N^j \leq ca \sum_{j=0}^k N^k = ca(k+1) N^k , \quad \text{for all } N \geq n_0 ,$$

and we have a new constant, $c' \equiv ca(k+1)$, with

$$T_Q(N) \leq c' |N^k| , \quad \text{for all } N \geq n_0 ,$$

making T_Q *big-O* of N^k . QED

15.4 Ω Growth

15.4.1 Definition of Ω

Sometimes we want the opposite relationship between algorithm Q and a function f . We want to demonstrate that Q has *worse* (or at least no better) performance than f when the data set on which it operates grows. We say that

$$\begin{aligned} T_Q(N) &= \Omega(f(N)) \\ &\iff \\ T_Q \text{ grows at least as fast as } f(N). \end{aligned}$$

We are giving our timing on algorithm Q a *lower bound* using the function $f(N)$. In words, we say that “the timing for algorithm Q is **omega** of f ”. Quantitatively,

$$\begin{aligned} T_Q(N) &= \Omega(f(N)) \\ &\iff \end{aligned}$$

there exist positive constants n_0 and c such that

$$T_Q(N) \geq c|f(N)|, \quad \text{for all } N \geq n_0.$$

This means that while $T_Q(N)$ might start out being much smaller than $c|f(N)|$ for small N , eventually it will “degrade” as N increases to the extent that $T_Q(N)$ will become and stay $\geq c|f(N)|$ for all N once we get to $N = n_0$.

There are similar theorems as those we proved for *big-O* complexity that would apply to Ω time complexity, but they are not critical to our work, so I’ll leave them as an exercise.

[**Exercise.** State and prove lemmas and theorems analogous to the ones we proved for *big-O*, but applicable to Ω growth.]

15.5 Θ Growth

Next, we express the fact that an algorithm Q is said to grow at *exactly* (a term which is not universally used because it could be misinterpreted) the same rate as some mathematical expression $f(N)$ using the notation

$$T_Q(N) = \Theta(f(N)).$$

We mean that T_Q grows neither faster nor slower than $f(N)$. In words, we say “the timing for algorithm Q is theta $f(N)$ ”. Officially,

$$\begin{aligned} T_Q(N) &= \Theta(f(N)) \\ &\iff \\ &\text{both} \\ T_Q(N) &= O(f(N)) \quad \text{and} \quad T_Q(N) = \Omega(f(N)) \end{aligned}$$

Ideally, this is what we want to know about an algorithm. Sometimes, when programmers informally say an algorithm is *big-O of* N or $N \log N$, they really mean that it is *theta of* N or $\log N$, because they have actually narrowed down the growth rate to being precisely linear or logarithmic. Conversely, if a programmer says an algorithm is *linear* or *logarithmic* or $N \log N$, we don't know what they mean without qualification by one of categories, usually *big-O* or Θ .

15.6 *Little-o* Growth

Less frequently, you may come across *little-o* notation, that is, $T_Q = o(f(N))$. This simply means that we not only have an upper bound in $f(N)$, but this upper bound is, in some sense, *too high*. One way to say this is that

$$\begin{aligned} T_Q(N) &= o(f(N)) \\ &\iff \\ T_Q(N) &= O(f(N)) \text{ , but } T_Q(N) \neq \Theta(f(N)) \text{ , .} \end{aligned}$$

15.7 Easy vs. Hard

Computational theorists use the term *easy* to refer to a problem whose (known) algorithm has *polynomial time complexity*. We don't necessarily bother specifying the exact power of the bounding monomial; usually the powers are on the order of five or less. However in this course, we'll prove exactly what they are when we need to.

Hard problems are ones whose only known algorithms have *exponential time complexity*.

A large part of the promise of quantum computing is that it can use quantum parallelism and entanglement to take problems that are classically *hard* and find quantum algorithms that are *easy*. This is called *exponential speed-up* (sometimes qualified with the terms *relative* or *absolute*).

For the remainder of the course we will only tackle two remaining algorithms, but they will be whoppers. They both exhibit *exponential speed up*.

15.8 Wrap-Up

This section was a necessarily brief and incomplete study of time complexity because we only needed the most fundamental aspects of *big-O* and Θ growth for the most obvious and easy-to-state classes: *exponential growth*, *polynomial growth* (and a few cases of $N \log N$ growth). When we need them, the analysis we do should be self-explanatory, especially with this short section of definitions on which you can fall back.

Chapter 16

Computational Basis States and Modular Arithmetic

$$|15 \oplus 3\rangle^4 \leftrightarrow |12\rangle^4 \leftrightarrow |1100\rangle \leftrightarrow (1, 1, 0, 0)^t$$

16.1 Different Notations Used in Quantum Computing

In the quantum computing literature you'll encounter alternative ways to express the states of a qubit or the inner workings of an algorithm. If you're schooled in only one style, you might be puzzled when you come across unfamiliar verbiage, especially when the author changes dialects in mid-utterance. Today, I want to consolidate a few different ways of talking about Hilbert space and computational basis states. This will prepare you for the algorithms ahead and enable you to read more advanced papers which assume the reader can make such transitions seamlessly. It's also convenient to have this single resource to consult if you're reading a derivation and suddenly have a queasy feeling as the notation starts to get away from you.

16.2 Notation and Equivalence of Three Environments

16.2.1 First Environment – n -qubit Hilbert Space, $\mathcal{H}_{(n)}$

Single Qubit Hilbert Spaces

Recall that a one-qubit Hilbert space, \mathcal{H} , consists of the 2-D complex vector space with basis vectors $|0\rangle$ and $|1\rangle$, making its typical state a superposition (always normalized,

of course) of those two vectors, as in

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle.$$

Multi Qubit Hilbert Spaces

Multi-qubit computers operate in a *tensor product* of such spaces, one for each qubit. This product is a 2^n -dimensional Hilbert space, which I sometimes label with the subscript (n) for clarity, as in $\mathcal{H}_{(n)}$. We can think of it as

$$\mathcal{H}_{(n)} = \overbrace{\mathcal{H} \otimes \mathcal{H} \otimes \dots \otimes \mathcal{H}}^n = \bigotimes_{k=0}^{n-1} \mathcal{H}.$$

The computational basis states, or CBS, of this product space are the 2^n vectors of the form

$$|x\rangle^n = |x_{n-1}\rangle \otimes |x_{n-2}\rangle \otimes |x_{n-3}\rangle \otimes \dots \otimes |x_0\rangle = \bigotimes_{k=0}^{n-1} |x_k\rangle,$$

where each $|x_k\rangle$ is either $|0\rangle$ or $|1\rangle$, i.e., a CBS of the k th *one qubit* space. We index in decreasing order from x_{n-1} to x_0 because we'll want the right-most bit to correspond to the least significant digit of the binary number $x_{n-1} \dots x_1 x_0$.

Different CBS Notations

One shorthand we've used in the past for this CBS is

$$|x_{n-1}\rangle |x_{n-2}\rangle \dots |x_1\rangle |x_0\rangle,$$

and two other common notations we'll need are the decimal integer (*encoded*) form, x , and its binary representation,

$$|x\rangle^n, \quad x \in \{0, 1, 2, 3, \dots, 2^n - 1\} \quad \text{or} \\ |x_{n-1} x_{n-2} \dots x_3 x_2 x_1 x_0\rangle, \quad x_k \in \{0, 1\}.$$

For example, for $n = 3$,

$$\begin{aligned} |0\rangle^3 &\longleftrightarrow |000\rangle, \\ |1\rangle^3 &\longleftrightarrow |001\rangle, \\ |2\rangle^3 &\longleftrightarrow |010\rangle, \\ |3\rangle^3 &\longleftrightarrow |011\rangle, \\ |4\rangle^3 &\longleftrightarrow |100\rangle, \end{aligned}$$

and, in general,

$$|x\rangle^3 \longleftrightarrow |x_2 x_1 x_0\rangle.$$

The 2-dimensional $\mathcal{H} = \mathcal{H}_{(1)}$ and its 2^n -dimensional products $\mathcal{H}_{(n)}$ are models we use for quantum computing. However, the problems that arise naturally in math and computer science are based on simpler number systems. Let's have a look at two such systems and show that they are equivalent to the CBS of our Hilbert space(s), \mathcal{H} .

16.2.2 The Second Environment: The Finite Group $(\mathbb{Z}_2)^n$

Simple Integers

We're all familiar with the integers, \mathbb{Z} ,

$$\mathbb{Z} \equiv \{ \dots, -3, -2, -1, 0, 1, 2, 3, \dots \}, \quad \text{usual } +,$$

where I have explicitly stated the operation of interest, namely ordinary addition. (We're not particularly interested in multiplication at this time.) This is sometimes called the *group of integers*, and as a set we know it's infinite, stretching toward $\pm\infty$ in the two directions.

Stepping Stone: \mathbb{Z}_N , or mod N Arithmetic

Another group that you may not have encountered in a prior course is the *finite* group \mathbb{Z}_N , consisting of only the N integers from 0 to $N - 1$,

$$\mathbb{Z}_N \equiv \{ 0, 1, 2, \dots, N - 1 \}, \quad \text{"+" is } (+ \bmod N),$$

and this time we are using *addition modulo N* as the principal operation; if $x + y \geq N$, we bring it back into the set by taking its remainder after dividing by N :

$$x + y \bmod N \equiv (x + y) \% N.$$

The negative of each $x \in \mathbb{Z}_N$ is defined by

$$-x \bmod N \equiv (N - x).$$

Subtraction is defined using the above two definitions, as you would expect,

$$x - y \bmod N \equiv (x + -y) \% N.$$

To make this concrete, we consider the group \mathbb{Z}_{15} .

Example Operations mod-15

$$\begin{aligned} 7 + 2 &= 9 \\ 7 + 10 &= 2 \\ 14 + 14 &= 13 \\ -1 &= 14 \\ -7 &= 8 \\ 10 &= -5 \\ 14 - 2 &= 12 \\ 2 - 8 &= 9 \\ 4 - 4 &= 0 \end{aligned}$$

Stepping Stone: \mathbb{Z}_2 with \oplus Arithmetic

While \mathbb{Z}_N is important in its own right (and we'll be using it in the future), an important special case you'll want to embrace is \mathbb{Z}_N for $N = 2$. The above definition of \mathbb{Z}_N works for \mathbb{Z}_2 , but when $N = 2$ some special notation kicks in:

$$\mathbb{Z}_2 \equiv \{0, 1\}, \quad \text{"}\oplus\text{" is } (+ \bmod 2).$$

A few consequences of mod-2 addition are

$$\begin{aligned} 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 0 \oplus 0 &= 0 \\ 1 \oplus 1 &= 0 \\ -1 &= 1 \\ 0 - 1 &= 0 \ominus 1 = 1 \end{aligned}$$

Of course \oplus is nothing other than the familiar **XOR** operation, although in this context, we get subtraction and negative mod-2 numbers defined, as well. Also, while we should be consistent and call subtraction \ominus , the last example shows that we can, and usually do, use the ordinary “ $-$ ” operator, even in mod-2 arithmetic. If there is the potential for confusion, we would tag on the parenthetical “ $(\bmod 2)$.”

An Old Friend with a New Name

We studied \mathbb{Z}_2 under a different name during our introductory lecture on *a single qubit* (although you may have skipped that optional section; it was a study of classical bits and classical gates using the formal language of vector spaces in preparation for defining the qubit). At the time we didn't use the symbolism \mathbb{Z}_2 for the two-element group, but called it \mathbb{B} , the two-element “field” on which we built the vector space \mathcal{B} for the *formal* definition of *classical* bits and operators.

Connection Between \mathbb{Z}_2 and \mathcal{H}

While not officially recognized by any government or municipality, it will help to observe a very simple relationship between the mod-2 group, \mathbb{Z}_2 , and the single-qubit space, $\mathcal{H} = \mathcal{H}_{(1)}$. It's little more than the fact that the dimension of $\mathcal{H}_{(1)}$, 2, equals the size of the group \mathbb{Z}_2 , also 2. Moreover, each CBS of $\mathcal{H}_{(1)}$ is labeled using the digits 0 and 1:

$$\begin{array}{ccc} \mathbb{Z}_2 & & \mathcal{H}_{(1)} \\ 0 & \leftrightarrow & |0\rangle \\ 1 & \leftrightarrow & |1\rangle \end{array}$$

I hasten to add that this connection does not go beyond the 1:1 correspondence listed above and, in particular, *does not* extend to the mod-2 addition in \mathbb{Z}_2 vs. the vector addition in \mathcal{H} ; those are totally separate and possess no similarities. Also, *only the basis states* in \mathcal{H} are part of this correspondence; the general state, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ has no place in the analogy. As tenuous as it may seem, this connection will help us in the up-coming analysis.

Second Environment Completed: $(\mathbb{Z}_2)^n$ with \oplus Arithmetic

As a set, $(\mathbb{Z}_2)^n$ is simply the n -tuples that have either 0 or 1 as their coordinates, that is,

$$(\mathbb{Z}_2)^n \equiv \left\{ (x_{n-1}, x_{n-2}, \dots, x_2, x_1, x_0) \right\},$$

each $x_k = 0$ or 1 ,

or, in column vector notation,

$$\begin{aligned} (\mathbb{Z}_2)^n &\equiv \left\{ \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \right\} \\ &= \left\{ \begin{pmatrix} x_{n-1} \\ x_{n-2} \\ \vdots \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} \right\}. \end{aligned}$$

Notice that we label the 0th coordinate on the far right, or bottom, and the $(n-1)$ st coordinate on the far left, or top. This facilitates the association of these vectors with binary number representations (coming soon) in which the LSB is on the right, and the MSB is on the left (as in binary $1000 = 8$, while $0001 = 1$).

The additive operation stems from the “ \mathbb{Z}_2 ” in its name: it’s the component-wise mod-2 addition, \oplus or, equivalently **XOR**, e.g.,

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

I’ll usually write the elements of $(\mathbb{Z}_2)^n$ in boldface, as in **x** or **y**, to emphasize the vector point-of-view.

Common Notation. This set is often written $\{0, 1\}^n$, especially when we don’t care about the addition operation, only n -tuples of 0s and 1s that are used as inputs to Boolean functions.

$(\mathbb{Z}_2)^n$ is a Vector Space

I’ve already started calling the objects in $(\mathbb{Z}_2)^n$ *vectors*, and this truly is an official designation. Just as \mathbb{R}^n is an n -dimensional vector space over the reals, and $\mathcal{H}_{(n)}$ is a 2^n -dimensional vector space over the complex numbers, $(\mathbb{Z}_2)^n$ is a vector space over \mathbb{Z}_2 . The natural question arises, *what does this even mean?*

You know certain things about all vector spaces, a few of which are

- There is some kind of *scalar multiplication*, $c\mathbf{v}$.
- There is always some *basis* and therefore a *dimension*.
- There is often an *inner product* defining *orthogonality*.

All this is true of $(\mathbb{Z}_2)^n$, although the details will be defined as they crop up. For now, we only care about the vector notation and \oplus addition. That is, unless you want to do this ...

[**Exercise.** Describe all of the above and anything else that needs to be confirmed to authorize us to call $(\mathbb{Z}_2)^n$ a vector space.]

Caution. For general N , $(\mathbb{Z}_N)^n$ is not a vector space. The enlightened among you can help the uninitiated understand this in your forums, but it is not something we will need. What kind of N *will* lead to a vector space?

Recall. Once again, think back to the vector space that we called $\mathcal{B} = \mathbb{B}^2$. It was the formal structure that we used to define classical bits. Using the more conventional language of this lesson, \mathcal{B} is the four-element, 2-dimensional vector space $(\mathbb{Z}_2)^2$.

Connection Between $(\mathbb{Z}_2)^n$ and $\mathcal{H}_{(n)}$

Let’s punch up our previous analogy, bringing it into higher dimensions. We relate the n -component vectors, $(\mathbb{Z}_2)^n$, and the multi-qubit space, $\mathcal{H} = \mathcal{H}_{(n)}$. As before,

the dimension of $\mathcal{H}_{(n)}$, 2^n , equals the size of the group $(\mathbb{Z}_2)^n$, also 2^n . The vectors in $(\mathbb{Z}_2)^n$ corresponds nicely to the CBS in $\mathcal{H}_{(n)}$:

$$\begin{array}{ccc}
(\mathbb{Z}_2)^n & & \mathcal{H}_{(n)} \\
(0, 0, \dots, 0, 0, 0)^t & \leftrightarrow & |00 \cdots 000\rangle \\
(0, 0, \dots, 0, 0, 1)^t & \leftrightarrow & |00 \cdots 001\rangle \\
(0, 0, \dots, 0, 1, 0)^t & \leftrightarrow & |00 \cdots 010\rangle \\
(0, 0, \dots, 0, 1, 1)^t & \leftrightarrow & |00 \cdots 011\rangle \\
(0, 0, \dots, 1, 0, 0)^t & \leftrightarrow & |00 \cdots 100\rangle \\
\vdots & & \vdots \\
(x_{n-1}, \dots, x_2, x_1, x_0)^t & \leftrightarrow & |x_{n-1} \cdots x_2 x_1 x_0\rangle
\end{array}$$

Again, there is no connection between the respective addition operations, and the correspondence does not include superposition states of $\mathcal{H}_{(n)}$. Still, the basis states in $\mathcal{H}_{(n)}$ line up with the vectors in $(\mathbb{Z}_2)^n$, and that's the important thing to remember.

16.2.3 The Third Environment: The Finite Group \mathbb{Z}_{2^n} with \oplus Arithmetic

One of our stepping stones above included the *finite* group \mathbb{Z}_N with mod- N $+$ as its additive operation,

$$\mathbb{Z}_N \equiv \{0, 1, 2, \dots, N-1\}.$$

Now we're going to make two modifications to this. First, we'll restrict the size, N , to powers of 2, i.e., $N = 2^n$, for some n ,

$$\mathbb{Z}_{2^n} \equiv \{0, 1, 2, \dots, 2^n - 1\},$$

so each x in \mathbb{Z}_N has exactly n binary digits.

$$\begin{aligned}
x &\longleftrightarrow x_{n-1} x_{n-2} \cdots x_2 x_1 x_0, \quad \text{notationally, or} \\
x &= \sum_{k=0}^{n-1} x_k 2^k, \quad \text{as a sum of powers-of-2.}
\end{aligned}$$

The second change will be to the addition operation. It will be neither normal addition nor mod- N addition. Instead, we define $x + y$ using the *bit-wise* \oplus operator.

$$\begin{aligned}
\text{If } x &= x_{n-1} x_{n-2} \cdots x_2 x_1 x_0, \\
\text{and } y &= y_{n-1} y_{n-2} \cdots y_2 y_1 y_0, \\
\text{then } x \oplus y &\equiv (x_{n-1} \oplus y_{n-1}) \cdots (x_2 \oplus y_2)(x_1 \oplus y_1)(x_0 \oplus y_0).
\end{aligned}$$

Note that the RHS of the last line is not a product, but the *binary representation* using its base-2 digits (e.g., 11010001101). Another way to say it is

$$x \oplus y \equiv \sum_{k=0}^{n-1} (x_k \oplus y_k) 2^k.$$

To eliminate any confusion between this group and the same set under ordinary mod- N (mod- 2^n) addition, let's call ours by its full name,

$$(\mathbb{Z}_{2^n}, \oplus) .$$

Examples of addition in $(\mathbb{Z}_{2^n}, \oplus)$ are

$$\begin{aligned} 1 \oplus 1 &= 0, \\ 1 \oplus 2 &= 3, \\ 1 \oplus 5 &= 4, \\ 4 \oplus 5 &= 1, \\ 5 \oplus 11 &= 14, \\ 13 \oplus 13 &= 0, \quad \text{and} \\ 15 \oplus 3 &= 12. \end{aligned}$$

Note that for any $x \in (\mathbb{Z}_{2^n}, \oplus)$,

$$\begin{aligned} x \oplus x &= 0, \quad \text{so} \\ x &= -x, \end{aligned}$$

i.e., x is its own additive inverse, under bit-wise \oplus .

Connection Between $(\mathbb{Z}_2)^n$ and $(\mathbb{Z}_{2^n}, \oplus)$

$(\mathbb{Z}_{2^n}, \oplus)$ and $(\mathbb{Z}_2)^n$ are fundamentally two ways to express the same group – they are *isomorphic* in group terminology. This is symbolized by

$$(\mathbb{Z}_{2^n}, \oplus) \cong (\mathbb{Z}_2)^n ,$$

under the set and operator association

$$\begin{aligned} x = (x_{n-1}x_{n-2} \cdots x_1x_0) &\longleftrightarrow \mathbf{x} = \begin{pmatrix} x_{n-1} \\ x_{n-2} \\ \vdots \\ x_1 \\ x_0 \end{pmatrix} , \\ x \oplus y &\longleftrightarrow \mathbf{x} \oplus \mathbf{y} . \end{aligned}$$

[Exercise. For those of you fixating on the vector space aspect of $(\mathbb{Z}_2)^n$, you may as well satisfy your curiosity by writing down why this makes \mathbb{Z}_{2^n} a vector space over \mathbb{Z}_2 , one that is *isomorphic to* (effectively the same as) $(\mathbb{Z}_2)^n$.]

16.2.4 Interchangeable Notation of $\mathcal{H}_{(n)}$, $(\mathbb{Z}_2)^n$ and (\mathbb{Z}_2^n, \oplus)

In practical terms, the relationship between the above three environments allows us to use bit-vectors,

$$(1, 1, 0, 1, 0)^t = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix},$$

binary number strings,

$$11010,$$

or plain old “encoded” ints

$$26$$

interchangeably, at will. One way we’ll take advantage of this is by using plain int notation in our kets. For $n = 5$, for example, we might write any of the four equivalent expressions,

$$\begin{aligned} &|26\rangle \\ &|11010\rangle \\ &|1\rangle |1\rangle |0\rangle |1\rangle |0\rangle \\ &|1\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |0\rangle, \end{aligned}$$

usually the first or second. Also, we may add notation to designate the number of qubits under consideration, as in

$$\begin{aligned} &|26\rangle^5 \quad \text{or, to show one possible breakdown, the equivalent} \\ &|3\rangle^2 \otimes |2\rangle^3. \end{aligned}$$

Hazard

Why are these last two equivalent? We must be careful not to confuse the encoded CBS notation as if it gave coordinates of CBS kets in the natural basis – it does not. In other words, $|3\rangle^2$ expressed in natural tensor coordinates *is not*

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

It can’t be for two reasons:

- a) $|3\rangle^2$ is a 4-dimensional vector and requires four, not two, coordinates to express it, and

- b) $|3\rangle^2$ is a CBS, and any CBS ket expressed in its own (natural) basis can have only a single 1 coordinate, the balance of the column consisting of 0 coordinates.

The *correct* expression of $|3\rangle^2$ in tensor coordinates is

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

as can be seen if we construct it from its component tensor coordinates using

$$|3\rangle^2 = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Therefore, to answer this last question “why are the last two expressions equivalent?” we must first express all vectors in terms of natural coordinates. That would produce three column vectors (for $|3\rangle^2$, $|2\rangle^3$ and $|26\rangle^5$) in which all had a single 1 in its respective column, and only then could we compute the product of two of them, demonstrating that it was equal to the third.

To head off another possible source of confusion, we must understand why the tensor product dimension of the two component vectors is not $2 \times 3 = 6$ contradicting a possibly (and incorrectly) hoped-for result of 5. Well, the dimensions of these spaces are not 2, 3, 5 or even 6. Remember that “exponent” to the upper-right of the ket designates the *order* of the Hilbert space. Meanwhile, the *dimension* of each space is $(2)^{\text{order}}$, so these dimensions are actually 2^2 , 2^3 and 2^5 . Now we can see that the product space dimension, 32, equals the product of the two component dimensions, 4×8 , as it should.

Sums inside Kets

Most importantly, if x and y are two elements in \mathbb{Z}_{2^n} , we may take their mod-2 sum inside a ket,

$$|x \oplus y\rangle,$$

which means that we are first forming $x \oplus y$, as defined in $(\mathbb{Z}_{2^n}, \oplus)$, and then using that n -bit answer to signify the CBS associated with it, e.g.,

$$\begin{aligned} |1 \oplus 5\rangle &= |4\rangle, \\ |5 \oplus 11\rangle &= |14\rangle \quad \text{or, designating a qubit size,} \\ |15 \oplus 3\rangle^4 &= |12\rangle^4 \quad \text{and} \\ |21 \oplus 21\rangle^6 &= |0\rangle^6. \end{aligned}$$

Example of Different Notations Applied to Hadamard

Recall that the n th order Hadamard operator's definition, usually given in encoded binary form, is

$$H^{\otimes n} |x\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{x \odot y} |y\rangle^n,$$

where \odot is the mod-2 dot product based on the individual binary digits in the base-2 representation of x and y ,

$$x \odot y = x_{n-1} y_{n-1} \oplus x_{n-2} y_{n-2} \oplus \cdots \oplus x_1 y_1 \oplus x_0 y_0.$$

If x and y are represented as vectors in $(\mathbb{Z}_2)^n$ using the boldface \mathbf{x} and \mathbf{y} ,

$$x \leftrightarrow \mathbf{x} = \begin{pmatrix} x_{n-1} \\ x_{n-2} \\ \vdots \\ x_1 \\ x_0 \end{pmatrix}, \quad y \leftrightarrow \mathbf{y} = \begin{pmatrix} y_{n-1} \\ y_{n-2} \\ \vdots \\ y_1 \\ y_0 \end{pmatrix}.$$

the *dot product* between vector \mathbf{x} and vector \mathbf{y} is also assumed to be the mod-2 dot product,

$$\mathbf{x} \cdot \mathbf{y} = x_{n-1} y_{n-1} \oplus x_{n-2} y_{n-2} \oplus \cdots \oplus x_1 y_1 \oplus x_0 y_0,$$

giving the alternate form of the Hadamard gate,

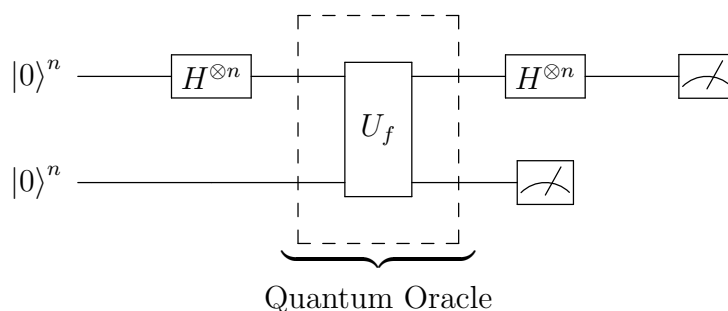
$$H^{\otimes n} |x\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{x} \cdot \mathbf{y}} |y\rangle^n.$$

This demonstrates the carefree change of notation often uncounted in this and other quantum computing presentations.

We have completed our review and study of the different notation and language used for CBS. As we move forward, we'll want to add more vocabulary that straddles these three mathematical systems, most notably, *periodicity*, but that's best deferred until we get to the algorithms which require it.

Chapter 17

Quantum Oracles



17.1 Higher Dimensional Oracles and their Time Complexity

We've seen *oracles* in our circuits for *Deutsch*, *Deutsch-Jozsa* and *Bernstein-Vazirani*, but today we will focus on the oracle itself, not a specific algorithm. Our goals will be to

- extend our *input size* to cover any dimension for each of the two oracle's channels,
- get a visual classification of the *matrix* for an oracle, and
- define *relativized* and *absolute* time complexity, two different ways of measuring a quantum algorithm's improvement over a classical algorithm.

The last item relies on an understanding of the oracle's time complexity, which is why it is included in this chapter.

We'll continue to use " U_f " to represent the oracle for a Boolean function, f . Even as we widen the input channels today, U_f will still have two general inputs, an *upper*, A or *data* register and a *lower*, B or *target* register.

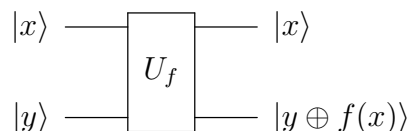
At the top of the page I've included a circuit that solves *Simon's problem* (coming soon). It reminds us how the oracle relates to the surrounding gates and contains a wider (n qubit) input to the target than we've seen up to now.

17.2 Simplest Oracle: a Boolean Function of One Bit

At the heart of our previous quantum circuits lurked the unitary transformation we've been calling the *quantum oracle* or just *the oracle*. Other gates may be wired into the circuit around the oracle, but they are usually standard “parts” that we pull off the shelf like CNOT, Hadamard and other simple gates. The oracle on the other hand is custom designed for the problem to be solved. It typically involves some function, f , that the circuit and its algorithm are meant to explore/discover/categorize.

17.2.1 Circuit and Initial Remarks

The simplest quantum oracle is one that arises from a *unary Boolean* f . We defined such a U_f in the *Deutsch* circuit. Its action on a general CBS $|x\rangle|y\rangle$ is shown in the following circuit:



In terms of the effect that the oracle has on the CBS $|x\rangle|y\rangle$, which we know to be shorthand for $|x\rangle \otimes |y\rangle$, the oracle can be described as

$$|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle.$$

There are a number of things to establish at the start, some review, others new.

Initial Remarks (Possibly Review)

1. \oplus is the mod-2 addition:

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

2. This gate can be viewed as a *reversible* operator for a typically *irreversible* boolean function, f .

3. The separable state $|x\rangle|y\rangle$ coming in from the left represents a very special and restricted input: a CBS. Whenever any gate is defined in terms of CBS, we must remember to use linearity and extend the definition to the entire Hilbert space. We do this by expanding a general ket, such as a 4-dimensional $|\psi\rangle^2$, along the CBS,

$$\begin{aligned} |\psi\rangle^2 &= c_0 |0\rangle|0\rangle + c_1 |0\rangle|1\rangle + c_2 |1\rangle|0\rangle + c_3 |1\rangle|1\rangle \\ &= c_0 |0\rangle^2 + c_1 |1\rangle^2 + c_2 |2\rangle^2 + c_3 |3\rangle^2, \end{aligned}$$

reading off the output for each of the CBS kets from our oracle description, and combining those using the complex amplitudes, c_k .

4. When a CBS is presented to the input of an oracle like U_f , the output happens to be a *separable* state (something not true for general unitary gates as we saw with the BELL operator). In this case, the separable output is $|x\rangle|y \oplus f(x)\rangle$. Considering the last bullet, we can't expect such a nice separable product when we present the oracle with some non-basis state, $|\psi\rangle^2$, at its inputs. Take care not to make the mistake of attempting to apply the above template directly on non-CBS inputs.

Initial Remarks (Probably New)

Oracles are often called *black boxes*, because we computer scientists don't care how the physicists and engineers build them or what's inside. However, when we specify an oracle using any definition (above being only one such example), we have to check that certain criteria are met.

1. The definition of U_f 's action on the CBS inputs as described above must result in unitarity. Should you come across a putative oracle with a slightly off-beat definition, a quick check of unitarity might be in order – a so called “sanity check.”
2. The above circuit is for two one-qubit inputs (that's a total of 4-dimensions for our input and output states) based on a function, f , that has *one bit in* and *one bit out*. After studying this easy case, we'll have to extend the definition and confirm unitarity for
 - multi-qubit input registers taking CBS of the form $|x\rangle^n$ and $|y\rangle^m$, and
 - an f with domain and range larger than the set $\{0, 1\}$.
3. The function that we specify needs to be *easy* to compute in the complexity sense. A quantum circuit won't likely help us if a computationally *hard* function is inside an oracle. While we may be solving *hard problems*, we need to find *easy functions* around which to build our circuits. This means the functions have to be computable in *polynomial time*.
4. Even if the function is easy, the quantum oracle still may be impractical to build in the near future of quantum computing.

17.2.2 A Two-Qubit Oracle's Action on the CBS

The nice thing about studying a one bit function, f , and its two-qubit oracle, U_f , is that we don't have to work in abstracts. There are so few options, we can compute each one directly from its definition. The results often reveal patterns that will hold in the more general cases.

Notation Reminder – If a is a single binary digit, $\bar{a} = \neg a$ is its *logical negation* (AKA the *bit-flip* or *not* operation),

$$\bar{a} \text{ (or } \neg a) \equiv \begin{cases} 0, & \text{if } a = 1, \\ 1, & \text{if } a = 0. \end{cases}$$

A short scribble should convince you that $0 \oplus a = a$ and $1 \oplus a = \bar{a}$. Therefore,

$$\begin{aligned} |x\rangle |0\rangle &\xrightarrow{U_f} |x\rangle |0 \oplus f(x)\rangle = |x\rangle |f(x)\rangle, \text{ and} \\ |x\rangle |1\rangle &\xrightarrow{U_f} |x\rangle |1 \oplus f(x)\rangle = |x\rangle |\overline{f(x)}\rangle. \end{aligned}$$

17.2.3 Case Study #1: $f(x) \equiv 1$

CBS Table for the Oracle

If $f \equiv 1$, a constant function, we can list all the possible outputs in a four-line table:

$ x\rangle y\rangle$	$U_f(x\rangle y\rangle)$
$ 0\rangle 0\rangle = 0\rangle^2$	$ 0\rangle 1\rangle = 1\rangle^2$
$ 0\rangle 1\rangle = 1\rangle^2$	$ 0\rangle 0\rangle = 0\rangle^2$
$ 1\rangle 0\rangle = 2\rangle^2$	$ 1\rangle 1\rangle = 3\rangle^2$
$ 1\rangle 1\rangle = 3\rangle^2$	$ 1\rangle 0\rangle = 2\rangle^2$

(Remember, $| \rangle^2$ does not mean “ket squared”, but is an indicator that this is an n -fold tensor product state, where $n = 2$.)

The Matrix for U_f

It's always helpful to write down the matrix for any linear operator. At the very least it will usually reveal whether or not the operator is unitary – even though we suspect without looking at the matrix that U_f is unitary since it is *real* and is *its own inverse*. However, self-invertibility does not always unitarity make, so it's safest to confirm unitarity by looking at the matrix.

This is a transformation from 4-dimensions to 4-dimensions, so we need to express the 4-dimensional basis kets in coordinate form. Let's review the connection between the four CBS kets of $\mathcal{H}_{(2)}$ and their natural basis *coordinates*:

component	$ 0\rangle 0\rangle$	$ 0\rangle 1\rangle$	$ 1\rangle 0\rangle$	$ 1\rangle 1\rangle$
encoded	$ 0\rangle^2$	$ 1\rangle^2$	$ 2\rangle^2$	$ 3\rangle^2$
coordinate	$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

To obtain the matrix, express each $U_f(|x\rangle|y\rangle)$ as a column vector for each tensor CBS, $|k\rangle^2$, $k = 0, \dots, 3$:

$$\begin{aligned}
& (U_f|0\rangle^2 \quad U_f|1\rangle^2 \quad U_f|2\rangle^2 \quad U_f|3\rangle^2) \\
&= (|1\rangle^2 \quad |0\rangle^2 \quad |3\rangle^2 \quad |2\rangle^2) \\
&= \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}
\end{aligned}$$

Aha. These rows (or columns) are clearly orthonormal, so the matrix is unitary meaning the operator is unitary.

It will be useful to express this matrix in terms of the 2×2 Pauli matrix, σ_x , associated with the X gate.

$$\begin{aligned}
U_f &= \left(\begin{array}{c|c} \sigma_x & 0 \\ \hline 0 & \sigma_x \end{array} \right) \\
&= \left(\begin{array}{c|c} \sigma_x & \\ \hline & \sigma_x \end{array} \right)
\end{aligned}$$

This form reveals a pattern that you may find useful going forward. Whenever a square matrix M can be broken down into smaller unitary matrices along its diagonal (0s assumed elsewhere), M will be unitary.

[Exercise. Prove the last statement.]

[Exercise. As amateur quantum “mechanics,” we can also see that it is Hermitian. How?]

17.2.4 Case Study #2: $f(x) \equiv x$

CBS Table for the Oracle

Now, let $f \equiv x$, the identity function. We repeat the process of the first case study and list all the possible outputs. For reference, here are the key mappings again:

$$\begin{aligned} |x\rangle |0\rangle &\xrightarrow{U_f} |x\rangle |f(x)\rangle, \\ |x\rangle |1\rangle &\xrightarrow{U_f} |x\rangle |\overline{f(x)}\rangle. \end{aligned}$$

Now, the table for $f \equiv x$:

$ x\rangle y\rangle$	$U_f(x\rangle y\rangle)$
$ 0\rangle 0\rangle = 0\rangle^2$	$ 0\rangle 0\rangle = 0\rangle^2$
$ 0\rangle 1\rangle = 1\rangle^2$	$ 0\rangle 1\rangle = 1\rangle^2$
$ 1\rangle 0\rangle = 2\rangle^2$	$ 1\rangle 1\rangle = 3\rangle^2$
$ 1\rangle 1\rangle = 3\rangle^2$	$ 1\rangle 0\rangle = 2\rangle^2$

The Matrix for U_f

We obtain the matrix by expressing each $U_f(|x\rangle |y\rangle)$ as a column vector:

$$\begin{aligned} & (U_f |0\rangle^2 \quad U_f |1\rangle^2 \quad U_f |2\rangle^2 \quad U_f |3\rangle^2) \\ &= (|0\rangle^2 \quad |1\rangle^2 \quad |3\rangle^2 \quad |2\rangle^2) \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\ &= \left(\begin{array}{c|c} \mathbb{1} & \\ \hline & \sigma_x \end{array} \right) \end{aligned}$$

This time, we've enlisted the help of the 2×2 identity matrix, $\mathbb{1}$. Again, we see how U_f acts on its inputs while confirming that it *is* unitary (and, in case anyone asks, Hermitian).

17.2.5 Remaining Cases #3 and #4

There are only two more 1-bit functions left to analyze. One of them, $f(x) \equiv 0$ was covered in our first *quantum algorithms* lesson under the topic of *Oracle for the [0]-op*.

We found its matrix to be

$$U_f = \left(\begin{array}{c|c} \mathbb{1} & \\ \hline & \mathbb{1} \end{array} \right) = \mathbb{1}.$$

The last, $f(x) \equiv \bar{x}$, has the matrix

$$U_f = \left(\begin{array}{c|c} \sigma_x & \\ \hline & \mathbb{1} \end{array} \right).$$

[**Exercise.** Derive the last matrix.]

17.3 Integers Mod-2 Review

17.3.1 The Classical f at the Heart of an Oracle

We will be building oracles based on classical Boolean functions that might have many inputs and one output,

$$f : \{0, 1\}^n \rightarrow \{0, 1\},$$

or many inputs and many outputs,

$$f : \{0, 1\}^n \rightarrow \{0, 1\}^m.$$

To facilitate this, we'll need a more compact language than $\{0, 1\}^n$, especially as we will be adding these multi-bit items both inside and outside our kets. We have such vocabulary in our quiver thanks to the previous lesson on *CBS and modular arithmetic*.

17.3.2 Mod-2 Notation for $\{0, 1\}^n$

We learned that $\{0, 1\}^n$ can be thought of any of three equivalent ways.

Encoded Integer Notation: $(\mathbb{Z}_{2^n}, \oplus)$

This is the group containing the first 2^n integers

$$\mathbb{Z}_{2^n} = \{0, 1, 2, 3, \dots, 2^n - 1\}$$

with mod-2 arithmetic, \oplus . There is always the possibility of confusion when we look at a set like \mathbb{Z}_N , for some integer $N > 0$. Usually, it means the numbers $\{0, 1, 2, \dots, N-1\}$ with the mod- N addition operation. However, when $N = 2^n$ is a power-of-2, it can often mean – and, for us, *does* – that we are using a *bitwise* mod-2 addition, \oplus . For this lesson, I'll use \mathbb{Z}_{2^n} rather than the more cumbersome $(\mathbb{Z}_{2^n}, \oplus)$ when I want to signify these “encoded” integers with mod-2 addition.

Binary Vector Notation: $(\mathbb{Z}_2)^n$

Here we mean the group consisting of the 2^n binary *vectors*,

$$(\mathbb{Z}_2)^n \equiv \left\{ \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \right\},$$

with \oplus being component-wise mod-2 addition.

The Natural Basis for $\mathcal{H}_{(n)}$

Finally, we have the 2^n basis kets for $\mathcal{H}_{(n)}$,

$$\text{CBS for } \mathcal{H}_{(n)} = \{ |x\rangle^n, \quad x = 0, 1, 2, 3, \dots, 2^n - 1 \}.$$

We can use either encoded or binary form to write these basis kets,

$$\begin{aligned} |0\rangle^n &\longleftrightarrow |0 \cdots 000\rangle, \\ |1\rangle^n &\longleftrightarrow |0 \cdots 001\rangle, \\ |2\rangle^n &\longleftrightarrow |0 \cdots 010\rangle, \\ |3\rangle^n &\longleftrightarrow |0 \cdots 011\rangle, \\ |4\rangle^n &\longleftrightarrow |0 \cdots 100\rangle, \\ &\vdots \\ |2^n - 1\rangle^n &\longleftrightarrow |1 \cdots 111\rangle. \end{aligned}$$

17.3.3 \oplus Inside the Ket

This review might be more than we need, but it will nip a few potentially confusing situations in the bud, the largest being the notation

$$|y \oplus f(x)\rangle$$

when y and $f(x)$ are more than just simple labels, 0 and 1, for the CBS of the 2-dimensional \mathcal{H} . Of course, when they *are* 0 and 1, we know what to do:

$$\begin{aligned} |1 \oplus 0\rangle &= |1\rangle \quad \text{or} \\ |1 \oplus 1\rangle &= |0\rangle. \end{aligned}$$

But when we are in a higher dimensional Hilbert space that comes about by studying a function sending \mathbb{Z}_{2^n} into \mathbb{Z}_{2^m} , we'll remember the above correspondence. For example,

$$\begin{aligned} |15 \oplus 3\rangle^4 &= |12\rangle^4 \quad \text{and} \\ |21 \oplus 21\rangle^6 &= |0\rangle^6. \end{aligned}$$

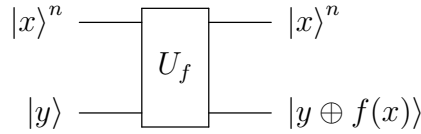
With that short review, we're ready to analyze intermediate and advanced multi-qubit oracles.

17.4 Intermediate Oracle: f is a Boolean Function of a Multiple Input Bits

17.4.1 Circuit

A Wider Data Register

Next, we study the circuit specification



which arises when the function under study is an f which maps $\{0, 1\}^n \rightarrow \{0, 1\}$. We came across this oracle in *Deutsch-Jozsa* and *Bernstein-Vazirani*. We'll review and dig deeper.

Such an f would require an n -qubit $|x\rangle^n$ input be sent to the A register, even as we maintained the single-qubit $|y\rangle$ going into the register B . This should be intuitively clear because the oracle's very definition calls for the output of the B register to be $|y \oplus f(x)\rangle$, something that can only be accomplished if both

- i) x has the right number of bits to satiate f (namely, n), and
- ii) the y to be " \oplus -ed" with $f(x)$ is commensurate with the output value of f (namely, a single binary bit).

The analysis of the simplest oracle taught us that U_f was expressible as a unitary 4×4 matrix with cute, little unitary 2×2 matrices (either $\mathbb{1}$ or σ_x) along its diagonal. It will turn out this intermediate-level oracle is nothing more than many copies of those same 2×2 s affixed to a longer diagonal, one traversing a $2^{n+1} \times 2^{n+1}$ matrix.

17.4.2 An $(n + 1)$ -Qubit Oracle's Action on the CBS

The general mapping we found so useful in the simple case continues to work for us here. Even though x is now in the larger domain, \mathbb{Z}_{2^n} , $f(x)$ is still restricted to 0 or 1, so we can reuse the familiar identities with a minor change (the first separable component gets an order n exponent),

$$\begin{aligned} |x\rangle^n |0\rangle &\xrightarrow{U_f} |x\rangle^n |f(x)\rangle, \\ |x\rangle^n |1\rangle &\xrightarrow{U_f} |x\rangle^n |\overline{f(x)}\rangle. \end{aligned}$$

17.4.3 Analyzing U_f for $x = 0$

For the moment, we won't commit to a specific f , but we *will* restrict our attention to the input $x = 0$.

$$\begin{aligned} |0\rangle^n |0\rangle &\xrightarrow{U_f} |0\rangle^n |f(0)\rangle, \\ |0\rangle^n |1\rangle &\xrightarrow{U_f} |0\rangle^n |\overline{f(0)}\rangle. \end{aligned}$$

$f(0)$ can be either 0 or 1, and y can be either 0 or 1, giving four possible combinations:

$ 0\rangle^n y\rangle$	$f(0)$	$U_f(0\rangle^n y\rangle)$
$ 0\rangle^n 0\rangle = 0\rangle^{n+1}$	0	$ 0\rangle^n f(0)\rangle = 0\rangle^n 0\rangle = 0\rangle^{n+1}$
$ 0\rangle^n 1\rangle = 1\rangle^{n+1}$	0	$ 0\rangle^n \overline{f(0)}\rangle = 0\rangle^n 1\rangle = 1\rangle^{n+1}$
$ 0\rangle^n 0\rangle = 0\rangle^{n+1}$	1	$ 0\rangle^n f(0)\rangle = 0\rangle^n 1\rangle = 1\rangle^{n+1}$
$ 0\rangle^n 1\rangle = 1\rangle^{n+1}$	1	$ 0\rangle^n \overline{f(0)}\rangle = 0\rangle^n 0\rangle = 0\rangle^{n+1}$

This is a little different from our previous table. Rather than completely determining a 4×4 matrix for a particular f , it gives us two possible 2×2 sub-matrices depending on the value of $f(0)$.

1. When $f(0) = 0$, the first two columns of the matrix for U_f will be (see upper two rows of table):

$$\begin{aligned} & (U_f |0\rangle^{n+1} \quad U_f |1\rangle^{n+1} \quad \dots) \\ &= (|0\rangle^{n+1} \quad |1\rangle^{n+1} \quad \dots) \\ &= \left(\begin{array}{cc|c} 1 & 0 & \\ 0 & 1 & \\ 0 & 0 & \\ \vdots & \vdots & \\ 0 & 0 & \end{array} \right) \\ &= \left(\begin{array}{c|c} \mathbf{1} & ? \\ \hline 0 & ? \end{array} \right) \end{aligned}$$

2. When $f(0) = 1$, the first two columns of the matrix for U_f will be (see lower

two rows of table):

$$\begin{aligned}
& (U_f |0\rangle^{n+1} \quad U_f |1\rangle^{n+1} \quad \dots) \\
&= (|1\rangle^{n+1} \quad |0\rangle^{n+1} \quad \dots) \\
&= \left(\begin{array}{cc|c} 0 & 1 & \\ 1 & 0 & \\ 0 & 0 & ? \\ \vdots & \vdots & \\ 0 & 0 & \end{array} \right) \\
&= \left(\begin{array}{c|c} \sigma_x & ? \\ \hline 0 & ? \end{array} \right)
\end{aligned}$$

The “?” represents the yet-to-be-studied portion of U_f .

So we see that regardless of the value $f(0)$, the upper-left 2×2 sub-matrix will be one of our two familiar unitary matrices, 1 or σ_x . There was nothing special about $x = 0$. We might have analyzed, $x = 1, 2, 3$, or any x up to $2^n - 1$. In fact, we’ll do that next.

17.4.4 Analyzing U_f for General x

We continue studying a non-specific f and, like the $x = 0$ case, work with a *fixed* x . However, this time $x \in \mathbb{Z}_{2^n}$ can be *any* element in that domain. If it helps, you can think of a small x , like $x = 1, 2$ or 3 . The mappings that will drive the math become

$$\begin{aligned}
|x\rangle^n |0\rangle &\xrightarrow{U_f} |x\rangle^n |f(x)\rangle, \\
|x\rangle^n |1\rangle &\xrightarrow{U_f} |x\rangle^n |\overline{f(x)}\rangle.
\end{aligned}$$

As before, there two possible values for $f(x)$ and two possible values for y , giving four possible combinations for this x :

$ x\rangle^n y\rangle$	$f(x)$	$U_f(x\rangle^n y\rangle)$
$ x\rangle^n 0\rangle$	0	$ x\rangle^n f(x)\rangle = x\rangle^n 0\rangle$
$ x\rangle^n 1\rangle$	0	$ x\rangle^n \overline{f(x)}\rangle = x\rangle^n 1\rangle$
$ x\rangle^n 0\rangle$	1	$ x\rangle^n f(x)\rangle = x\rangle^n 1\rangle$
$ x\rangle^n 1\rangle$	1	$ x\rangle^n \overline{f(x)}\rangle = x\rangle^n 0\rangle$

We computed the first two columns of the matrix for U_f before, and now we compute two columns of U_f further to the right.

[In Case You Were Wondering. Why did the single value $x = 0$ produce *two* columns in the matrix? Because there were two possible values for y , 0 and 1, which gave rise to two different basis kets $|0\rangle^n |0\rangle$ and $|0\rangle^n |1\rangle$. It was those two kets that we subjected to U_f to produce the first two columns of the matrix. Same thing here, except now the two basis kets that correspond to the fixed x under consideration are $|x\rangle^n |0\rangle$ and $|x\rangle^n |1\rangle$, and they will produce matrix columns $2x$ and $2x + 1$.]

1. When $f(x) = 0$, columns $2x$ and $2x + 1$ of the matrix for U_f will be (see upper two rows of table):

$$\begin{aligned}
 & \begin{pmatrix} \cdots & U_f(|x\rangle^n |0\rangle) & U_f(|x\rangle^n |1\rangle) & \cdots \end{pmatrix} \\
 & \begin{pmatrix} \cdots & |x\rangle^n |0\rangle & |x\rangle^n |1\rangle & \cdots \end{pmatrix} \\
 & \quad \quad \quad \begin{matrix} 2x & 2x+1 \\ \hline \end{matrix} \\
 & = \left(\begin{array}{c|c} \begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 0 \end{matrix} & \begin{matrix} 0 & 1 \end{matrix} \end{array} \right) \left. \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \right\} \begin{matrix} 2x \\ 2x+1 \end{matrix} \\
 & \quad \quad \quad \begin{matrix} 2x & 2x+1 \\ \hline \end{matrix} \\
 & = \left(\begin{array}{c|c} \begin{matrix} 0 \\ \hline 1 \\ \hline 0 \end{matrix} & \begin{matrix} 0 & 1 \end{matrix} \end{array} \right) \left. \begin{matrix} \\ \\ \end{matrix} \right\} \begin{matrix} 2x \\ 2x+1 \end{matrix}
 \end{aligned}$$

2. When $f(0) = 1$, columns $2x$ and $2x + 1$ of the matrix for U_f will be (see lower

two rows of table):

$$\begin{pmatrix} \cdots & U_f(|x\rangle^n |0\rangle) & U_f(|x\rangle^n |1\rangle) & \cdots \\ \cdots & |x\rangle^n |1\rangle & |x\rangle^n |0\rangle & \cdots \end{pmatrix}$$

$$= \left(\begin{array}{c|c|c} \overbrace{\begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 1 \\ 1 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix}}^{2x \quad 2x+1} & & \\ \hline & \sigma_x & \\ \hline & 0 & \end{array} \right) \left. \begin{array}{l} 2x \\ 2x+1 \end{array} \right\}$$

$$= \left(\begin{array}{c|c|c} \overbrace{\begin{matrix} 0 \\ \hline \sigma_x \\ \hline 0 \end{matrix}}^{2x \quad 2x+1} & & \\ \hline & & \\ \hline & & \end{array} \right) \left. \begin{array}{l} 2x \\ 2x+1 \end{array} \right\}$$

As you can see, for any x the two columns starting with column $2x$ contain all 0s *away from the diagonal* and are either the 2×2 identity $\mathbb{1}$ or the Pauli matrix σ_x *on the diagonal*, giving the matrix the overall form

$$\left(\begin{array}{cccc} [\mathbb{1} \text{ or } \sigma_x] & & & \\ & [\mathbb{1} \text{ or } \sigma_x] & & 0 \\ & & \ddots & \\ & 0 & & [\mathbb{1} \text{ or } \sigma_x] \\ & & & & [\mathbb{1} \text{ or } \sigma_x] \end{array} \right).$$

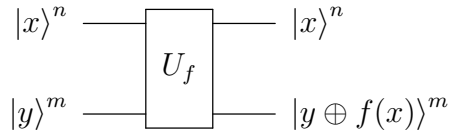
The oracle is – as we strongly suspected it would be – unitary (and let’s not forget, Hermitian), so we now have a nice visual image of what it looks like.

17.5 Advanced Oracle: f is a Multi-Valued function of a Multiple Input Bits

17.5.1 Circuit and Vocabulary

A Wider Target Register

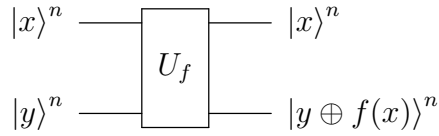
Finally, we'll need to understand the full circuit specification



which arises when the function under study is an f that maps $\mathbb{Z}_{2^n} \rightarrow \mathbb{Z}_{2^m}$. In this case it only makes sense to have an m -qubit B register, as pictured above, otherwise the sum inside the bottom right output, $|y \oplus f(x)\rangle$ would be ill-defined.

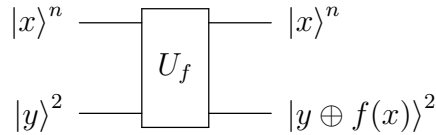
Sometimes $m = n$

Often, for multi-valued f , we can arrange things so that $m = n$ and, in that case the circuit will be



17.5.2 Smallest Advanced Oracle: $m = 2$ (Range of $f \subseteq \mathbb{Z}_{2^2}$)

In this case, $f(x) \in \{0, 1, 2, 3\}$ for each $x \in \mathbb{Z}_{2^n}$. There will be $(n+2)$ qubits going into U_f , (n qubits into the A register and 2 qubits into the B register).



This gives a total of 2^{n+2} possible input CBS going into the system, making the dimension of the overall tensor product space, $\mathcal{H}_{(n+2)}$, $= 2^{n+2}$. The matrix for U_f will, therefore, have size $(2^{n+2} \times 2^{n+2})$.

17.5.3 The 4×4 Sub-Matrix for a Fixed x

We follow the pattern set in the intermediate case and look at a non-specific f but work with a *fixed* $x \in \mathbb{Z}_{2^n}$. Because we are assuming $m = 2$, this leaves four possible images for this one x , namely $f(x) = 0, 1, 2$ or 3 . As we'll see, this leads to a 4×4

sub-matrix compared to the 2×2 sub-matrix we got when studying a fixed x in the intermediate case.

The maneuver that we have to apply here, which was not needed before, is the application of the \oplus operator between y and $f(x)$ on a bit-by-bit basis. For $m = 2$ (and using the notation $f(x)_k$ to mean the k th digit of the number $f(x)$), we get

$$\begin{aligned} |y \oplus f(x)\rangle^2 &= |y_1 y_0 \oplus f(x)_1 f(x)_0\rangle^2 \\ &= |(y_1 \oplus f(x)_1) (y_0 \oplus f(x)_0)\rangle^2, \\ &= |y_1 \oplus f(x)_1\rangle |y_0 \oplus f(x)_0\rangle^2. \end{aligned}$$

The second line is the definition of \oplus in \mathbb{Z}_{2^2} , and the $(\dots)(\dots)$ inside the ket is not multiplication but the binary expansion of the number $y_1 y_0 \oplus f(x)_1 f(x)_0$. Thus, our four combinations of the 2-bit number y with the fixed value $f(x)$ become

$$\begin{aligned} |x\rangle^n |0\rangle |0\rangle &\xrightarrow{U_f} |x\rangle^n |0 \oplus f(x)_1\rangle |0 \oplus f(x)_0\rangle, \\ |x\rangle^n |0\rangle |1\rangle &\xrightarrow{U_f} |x\rangle^n |0 \oplus f(x)_1\rangle |1 \oplus f(x)_0\rangle, \\ |x\rangle^n |1\rangle |0\rangle &\xrightarrow{U_f} |x\rangle^n |1 \oplus f(x)_1\rangle |0 \oplus f(x)_0\rangle, \text{ and} \\ |x\rangle^n |1\rangle |1\rangle &\xrightarrow{U_f} |x\rangle^n |1 \oplus f(x)_1\rangle |1 \oplus f(x)_0\rangle. \end{aligned}$$

Applying $0 \oplus a = a$ and $1 \oplus a = \bar{a}$ to the RHS of these equalities produces the identities that will drive the math,

$$\begin{aligned} |x\rangle^n |0\rangle |0\rangle &\xrightarrow{U_f} |x\rangle^n |f(x)_1\rangle |f(x)_0\rangle, \\ |x\rangle^n |0\rangle |1\rangle &\xrightarrow{U_f} |x\rangle^n |f(x)_1\rangle |\overline{f(x)_0}\rangle, \\ |x\rangle^n |1\rangle |0\rangle &\xrightarrow{U_f} |x\rangle^n |\overline{f(x)_1}\rangle |f(x)_0\rangle, \text{ and} \\ |x\rangle^n |1\rangle |1\rangle &\xrightarrow{U_f} |x\rangle^n |\overline{f(x)_1}\rangle |\overline{f(x)_0}\rangle. \end{aligned}$$

Of course, for a general f , there are now *four* possible values for $f(x)$, and we have to combine those with the four possible values for y , yielding a whopping 16 combinations for this fixed x , as the following table reveals.

$ x\rangle^n y\rangle^2$	$f(x)$	$U_f (x\rangle^n y\rangle^2)$
$ x\rangle^n 0\rangle^2$	0	$ x\rangle^n f(x)_1\rangle f(x)_0\rangle = x\rangle^n 0\rangle 0\rangle$
$ x\rangle^n 1\rangle^2$	0	$ x\rangle^n f(x)_1\rangle \overline{f(x)_0}\rangle = x\rangle^n 0\rangle 1\rangle$
$ x\rangle^n 2\rangle^2$	0	$ x\rangle^n \overline{f(x)_1}\rangle f(x)_0\rangle = x\rangle^n 1\rangle 0\rangle$
$ x\rangle^n 3\rangle^2$	0	$ x\rangle^n \overline{f(x)_1}\rangle \overline{f(x)_0}\rangle = x\rangle^n 1\rangle 1\rangle$
$ x\rangle^n 0\rangle^2$	1	$ x\rangle^n f(x)_1\rangle f(x)_0\rangle = x\rangle^n 0\rangle 1\rangle$
$ x\rangle^n 1\rangle^2$	1	$ x\rangle^n f(x)_1\rangle \overline{f(x)_0}\rangle = x\rangle^n 0\rangle 0\rangle$
$ x\rangle^n 2\rangle^2$	1	$ x\rangle^n \overline{f(x)_1}\rangle f(x)_0\rangle = x\rangle^n 1\rangle 1\rangle$
$ x\rangle^n 3\rangle^2$	1	$ x\rangle^n \overline{f(x)_1}\rangle \overline{f(x)_0}\rangle = x\rangle^n 1\rangle 0\rangle$
$ x\rangle^n 0\rangle^2$	2	$ x\rangle^n f(x)_1\rangle f(x)_0\rangle = x\rangle^n 1\rangle 0\rangle$
$ x\rangle^n 1\rangle^2$	2	$ x\rangle^n f(x)_1\rangle \overline{f(x)_0}\rangle = x\rangle^n 1\rangle 1\rangle$
$ x\rangle^n 2\rangle^2$	2	$ x\rangle^n \overline{f(x)_1}\rangle f(x)_0\rangle = x\rangle^n 0\rangle 0\rangle$
$ x\rangle^n 3\rangle^2$	2	$ x\rangle^n \overline{f(x)_1}\rangle \overline{f(x)_0}\rangle = x\rangle^n 0\rangle 1\rangle$
$ x\rangle^n 0\rangle^2$	3	$ x\rangle^n f(x)_1\rangle f(x)_0\rangle = x\rangle^n 1\rangle 1\rangle$
$ x\rangle^n 1\rangle^2$	3	$ x\rangle^n f(x)_1\rangle \overline{f(x)_0}\rangle = x\rangle^n 1\rangle 0\rangle$
$ x\rangle^n 2\rangle^2$	3	$ x\rangle^n \overline{f(x)_1}\rangle f(x)_0\rangle = x\rangle^n 0\rangle 1\rangle$
$ x\rangle^n 3\rangle^2$	3	$ x\rangle^n \overline{f(x)_1}\rangle \overline{f(x)_0}\rangle = x\rangle^n 0\rangle 0\rangle$

Building the matrix for U_f when $m = 2$ is accomplished by considering the four columns associated with the four possible $f(x)$ values. Because of the 16 possible sub-matrices of this form, we will just do a couple cases and see if a pattern emerges. (You know it will.)

1. When $f(x) = 0$, columns $4x$ through $4x + 3$ of the matrix for U_f will be (see topmost four rows of table):

$$\begin{aligned}
& \left(\cdots U_f (|x\rangle^n |0\rangle^2) \quad U_f (|x\rangle^n |1\rangle^2) \right. \\
& \quad \left. U_f (|x\rangle^n |2\rangle^2) \quad U_f (|x\rangle^n |3\rangle^2) \cdots \right) \\
& = \left(\cdots |x\rangle^n |0\rangle |0\rangle \quad |x\rangle^n |0\rangle |1\rangle \quad |x\rangle^n |1\rangle |0\rangle \quad |x\rangle^n |1\rangle |1\rangle \cdots \right),
\end{aligned}$$

which translates to the matrix

$$\left(\begin{array}{c|cccc|c} & \overbrace{0 & 0 & 0 & 0}^{4x \rightarrow 4x+3} & & \\ & \vdots & \vdots & \vdots & \vdots & \\ & 1 & 0 & 0 & 0 & \\ & 0 & 1 & 0 & 0 & \\ & 0 & 0 & 1 & 0 & \\ & 0 & 0 & 0 & 1 & \\ & \vdots & \vdots & \vdots & \vdots & \\ & 0 & 0 & 0 & 0 & \end{array} \right) \left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \right\} \begin{array}{l} 4x \\ \rightarrow \\ 4x+3 \end{array}$$

or, more concisely,

$$\left(\begin{array}{c|c|c} & \overbrace{0}^{4x \rightarrow 4x+3} & \\ \hline & \mathbb{1} & \\ \hline & 0 & \end{array} \right) \left. \begin{array}{l} \\ \\ \end{array} \right\} \begin{array}{l} 4x \rightarrow \\ 4x+3 \end{array}$$

As you'll see in a moment, it will facilitate the analysis if we break the 4×4 identity matrix into four smaller 2×2 matrices, and write this as

$$\left(\begin{array}{c|cc|c} & \overbrace{0}^{4x \rightarrow 4x+3} & & \\ \hline & 1 & 0 & \\ & 0 & 1 & \\ \hline & 0 & & \end{array} \right) \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \begin{array}{l} 4x \\ \rightarrow \\ 4x+3 \end{array}$$

This 4×4 matrix that appears in columns $4x \rightarrow 4x + 3$ is unitary by inspection.

2. Let's skip to the case $f(x) = 3$, and use the table to calculate columns $4x$ through $4x + 3$ of the matrix for U_f (see bottommost four rows of table):

$$\begin{pmatrix} \cdots & U_f(|x\rangle^n |0\rangle^2) & U_f(|x\rangle^n |1\rangle^2) & U_f(|x\rangle^n |2\rangle^2) & U_f(|x\rangle^n |3\rangle^2) & \cdots \end{pmatrix} \\ = \begin{pmatrix} \cdots & |x\rangle^n |1\rangle |1\rangle & |x\rangle^n |1\rangle |0\rangle & |x\rangle^n |0\rangle |1\rangle & |x\rangle^n |0\rangle |0\rangle & \cdots \end{pmatrix},$$

which translates to the matrix

$$\left(\begin{array}{c|cccc|} & \overbrace{0 & 0 & 0 & 0}^{4x \rightarrow 4x+3} & & & \\ & \vdots & \vdots & \vdots & \vdots & & & \\ & 0 & 0 & 0 & 1 & & & \\ & 0 & 0 & 1 & 0 & & & \\ & 0 & 1 & 0 & 0 & & & \\ & 1 & 0 & 0 & 0 & & & \\ & \vdots & \vdots & \vdots & \vdots & & & \\ & 0 & 0 & 0 & 0 & & & \end{array} \right) \left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array} \right\} \begin{array}{l} 4x \\ \longrightarrow \\ 4x+3 \end{array}$$

or, more concisely,

$$\left(\begin{array}{c|c|c|} & \overbrace{0}^{4x \rightarrow 4x+3} & & \\ \hline & 0 & \sigma_x & \\ & \sigma_x & 0 & \\ \hline & 0 & & \end{array} \right) \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} \begin{array}{l} 4x \\ \longrightarrow \\ 4x+3 \end{array}$$

This 4×4 matrix that appears in columns $4x \rightarrow 4x + 3$ is unitary by inspection.

3. **Exercise:** When $f(x) = 1$, show that columns $4x \rightarrow 4x + 3$ of U_f are

$$\left(\begin{array}{c|c|c} & \overbrace{\quad}^{4x \rightarrow 4x+3} & \\ \hline & 0 & \\ \hline & \sigma_x & 0 \\ & 0 & \sigma_x \\ \hline & 0 & \\ \hline \end{array} \right) \left. \vphantom{\begin{array}{c|c|c} & \overbrace{\quad}^{4x \rightarrow 4x+3} & \\ \hline & 0 & \\ \hline & \sigma_x & 0 \\ & 0 & \sigma_x \\ \hline & 0 & \\ \hline \end{array}} \right\} \begin{array}{l} 4x \\ \longrightarrow \\ 4x+3 \end{array}$$

This 4×4 matrix that appears in columns $4x \rightarrow 4x + 3$ is unitary by inspection.

4. **Exercise:** When $f(x) = 2$, show that columns $4x \rightarrow 4x + 3$ of U_f are

$$\left(\begin{array}{c|c|c} & \overbrace{\quad}^{4x \rightarrow 4x+3} & \\ \hline & 0 & \\ \hline & 0 & \mathbb{1} \\ & \mathbb{1} & 0 \\ \hline & 0 & \\ \hline \end{array} \right) \left. \vphantom{\begin{array}{c|c|c} & \overbrace{\quad}^{4x \rightarrow 4x+3} & \\ \hline & 0 & \\ \hline & 0 & \mathbb{1} \\ & \mathbb{1} & 0 \\ \hline & 0 & \\ \hline \end{array}} \right\} \begin{array}{l} 4x \\ \longrightarrow \\ 4x+3 \end{array}$$

This 4×4 matrix that appears in columns $4x \rightarrow 4x + 3$ is unitary by inspection.

Summary of Case $m = 2$

We've covered all four possible values for $f(x)$ and done so for arbitrary $x = 0, \dots, 2^n - 1$. What we have discovered is that

- each x controls four columns in the final matrix;
- one of four possible 4×4 unitary (sub-)matrices is positioned in these four columns, with 0s above and below that 4×4 matrix;
- the four possible sub-matrices for $f(x) = 0, 1, 2$ and 3 , respectively, are

$$\begin{array}{c|c} \mathbb{1} & \\ \hline & \mathbb{1} \end{array}, \quad \begin{array}{c|c} \sigma_x & \\ \hline & \sigma_x \end{array}, \quad \begin{array}{c|c} & \mathbb{1} \\ \hline \mathbb{1} & \end{array}, \quad \text{and} \quad \begin{array}{c|c} & \sigma_x \\ \hline \sigma_x & \end{array};$$

- since, for each x , its 4×4 unitary matrix is in rows $4x \rightarrow 4x + 3$, with zeros above and below, it follows (exercise) that there can only be zeros to the left and right on those rows.

Once again, even though we expected U_f to be unitary, its matrix can be seen to exhibit this property by direct observation, and its only non-zero elements are inside 4×4 sub-matrices which lie along the diagonal. (We may as well continue to notice the Hermiticity of these matrices. Make sure you can see that, too.) This characterizes all oracles for $f : \mathbb{Z}_{2^n} \rightarrow \mathbb{Z}_{2^2}$.

17.5.4 Easy Way to “See” Unitarity

Now that we’ve analyzed a few oracles, we can see why all U_f must be unitary. In fact, you might have noticed this even before having studied the examples. Here are the key points, and I’ll let you fill in the details as an **[Exercise]**.

- Every column is a CBS ket since it is the separable product of CBS kets.
- All CBS kets are normal vectors (all coordinates are 0 except one, which is 1).
- If two columns were the identical, U_f would map two different CBS kets to the same CBS ket.
- Any matrix that maps two different CBS kets to the same CBS ket cannot be invertible.
- Since U_f is its own inverse, it is invertible, so by last two bullets, all columns are distinct unit vectors.
- We conclude that distinct columns have their solitary 1 in different positions.
- The inner product of these columns with themselves is 1 and with other columns is 0: the matrix has orthonormal columns. QED

They’re Also Hermitian

We’re 99% of the way to confirming Hermiticity. Because

$$U_f U_f = \mathbf{1},$$

the solitary 1 in the k th row is matched by a solitary 1 in the k th column. This means

$$(U_f)^t = U_f.$$

But the matrix is real, so its transpose is also its adjoint,

$$(U_f)^\dagger = U_f,$$

in other words, it's Hermitian. It was easy to arrive at the Hermitian property from these simple relations compared with the herculean effort it took to compute the matrix.

Of course, we learn a lot by describing the *form* of the matrices, so the activities of this chapter have value beyond proving unitarity or Hermiticity.

17.5.5 The General Advanced Oracle: Any m (Range of $f \subseteq \mathbb{Z}_{2^m}$)

Extending Conclusions to Larger U_f Presents No Difficulties

This is where we should apply our intuition and extrapolate the $m = 2$ case to all m . Each time m increases by 1, the number of columns in U_f controlled by a single input value x doubles. For $m = 3$, we would have 8×8 unitary matrices along the diagonal, each built from appropriate combinations of σ_x and $\mathbb{1}$. For $m = 4$, we would have unitary 16×16 matrices along the diagonal. And when $m = n$, we would have $2^n \times 2^n$ unitary sub-matrices along the diagonal of a really large $2^{2^n} \times 2^{2^n}$ matrix for U_f . Our explicit demonstrations in the $m = 1$ and 2 cases can be extended to any m with no theoretical difficulty. We'd only be dealing with lengthier tables and larger sub-matrices. So if the number of output bits of f is m , for $m > 2$, the results are the same: U_f is unitary, Hermitian and it consists of all 0s except near the (increasingly large) diagonal where sub-matrices are built from combinations of σ_x and $\mathbb{1}$.

The Diagonal Sub-Matrices are Separable Products

However, if you are interested in one approach to a more rigorous understanding of U_f for general m , start here. In the $m = 2$ case, our four possible matrices comprising the 4×4 sub-matrices along the diagonal are actually tensor products of two matrices:

$$\begin{aligned} \left(\begin{array}{c|c} \mathbb{1} & \\ \hline & \mathbb{1} \end{array} \right) &= \mathbb{1} \otimes \mathbb{1} \\ \left(\begin{array}{c|c} \sigma_x & \\ \hline & \sigma_x \end{array} \right) &= \mathbb{1} \otimes \sigma_x \\ \left(\begin{array}{c|c} & \mathbb{1} \\ \hline \mathbb{1} & \end{array} \right) &= \sigma_x \otimes \mathbb{1} \\ \left(\begin{array}{c|c} & \sigma_x \\ \hline \sigma_x & \end{array} \right) &= \sigma_x \otimes \sigma_x \end{aligned}$$

Each component of the tensor product, $\mathbb{1}$ and σ_x , were the two possible 2×2 matrices that appeared along the diagonal of U_f for $m = 1$.

The tensor product of unitary matrices is easily shown to be unitary. And one could proceed inductively to demonstrate that the U_f for output size $m + 1$ consists of sub-matrices along the diagonal, each of which is a tensor products of the potential sub-matrices available to the U_f for size m . This will give both unitarity as well as the visual that we have already predicted.

[Caution. If we designate some function with 2^n inputs and 2^m outputs as $f_{n,m}$, we are *not* saying that $U_{f_{n,(m+1)}} = U_{f_{n,m}} \otimes U_{g_{n,m}}$ for two smaller oracles – it isn't. That statement doesn't even track, logically, since it would somehow imply we could generate an arbitrary U_f , which can be exponentially complex, out of repeated products of something very simple. Rather, we merely noted the fact that the sub-matrices along the diagonal are, individually, tensor products of the next-smaller m 's possible diagonal sub-matrices. We still need the full details of f to compute all the smaller sub-matrices, which will be different (in general) from one another.]

17.6 The Complexity of a Quantum Algorithm Relative to the Oracle

In this course, we are keeping things as simple as possible while attempting to provide the key ideas in quantum computation. To that end, we'll make only one key classification of oracles used in algorithms. You will undoubtedly explore a more rigorous and theoretical classification in your advanced studies.

Complexity of the Oracle

The above constructions all demonstrated that we can take an arbitrary function, f , and, in theory, represent it as a reversible gate associated with a unitary matrix. If you look at the construction, though, you'll see that any function which requires a full table of 2^n values to represent it (if there is no clear analytical short-cut we can use to compute it) will likewise end up with a similarly complicated oracle. The oracle would need an exponentially large number of gates (relative to the number of binary inputs, n). An example will come to us in the form of *Simon's algorithm* where we have a \mathbb{Z}_{2^n} -periodic function (notation to be defined) and seek to learn its period.

However, there are functions which we know have polynomial complexity and can be realized with a correspondingly simple oracle. An example of this kind of oracle is that which appears in Shor's *factoring* algorithm (not necessarily Shor's *period-finding* algorithm). In a factoring algorithm, we know a lot about the function that we are trying to crack and can analyze its specific form, proving it to be $O(n^3)$. Its oracle will also be $O(n^3)$.

Two Categories of a Quantum Algorithm's Complexity

Therefore, whenever presenting a quantum algorithm, one should be clear on which of the two kinds of oracles are available to the circuit. That distinction will be made using the following, somewhat informal (and unconventional) language.

- **Relativized Time Complexity** - This is the time complexity of a {circuit + algorithm} *without* knowledge of the oracle's design. If we were later given the complexity of the oracle, we would have to modify any prior analysis to account for it. Until that time we can only speak of the algorithm and circuit *around* the oracle. We can say that some algorithm is $O(n^3)$, e.g., but that doesn't mean it will be when we wire up the oracle. It is $O(n^3)$ *relative* to the oracle.
- **Absolute Time Complexity** - This is the time complexity of a {circuit + algorithm} *with* knowledge of the oracle's design. We will know and incorporate the oracle's complexity into the final {circuit + algorithm}'s complexity.

Chapter 18

Simon's Algorithm for Period-Finding

18.1 The Importance of Simon's Algorithm

Simon's algorithm represents a turning point in both the history of quantum computer science and in every student's study of the field. It is the first algorithm that we study which represents a substantial advance in relativized time complexity vs. classical computing. The problem is exponential classically, even if we allow an approximate rather than a deterministic solution. In contrast, the quantum algorithm is very fast, $O(\log^5 N)$, where N is the size of the domain of f . In fact, you will see a specific algorithm, more detailed than those usually covered, which has complexity $O(\log^3 N)$. We'll prove all this after studying the quantum circuit.

[Before going any further, make sure you didn't overlook the word "relativized" in the first paragraph. As you may recall from a past lecture, it means that we don't have knowledge, in general, about the performance of the circuit's oracle U_f . Simon's quantum algorithm gives us a *lower bound* for the relative complexity, but that would have to be revised upward if we ended up with an oracle that has a larger "*big-O*." This is not the fault of QC; the kinds of periodic functions covered in Simon's treatment are arbitrarily complex. The function that we test may have a nice $O(n)$ or $O(n^2)$ implementation, in which case we'll be in great shape. If not, it will become our bottleneck. That said, even with a polynomial-fast oracle, there is no classical algorithm that can achieve polynomial time solution, so the quantum solution is still a significant result.]

While, admittedly a toy problem in the sense that it's not particularly useful, it contains the key ingredients of the relevant algorithms that follow, most notably Shor's quantum factoring and encryption-breaking. Even better, Simon's algorithm is free of the substantial mathematics required by an algorithm like Shor's and thus embodies the essence of quantum computing (in a noise-free environment) without distracting complexities.

In the problem at hand, we are given a function and asked to find its period. However, the function is not a typical mapping of real or complex numbers, and the period is not the thing that you studied in your calculus or trigonometry classes. Therefore, a short review of periodicity, and its different meanings in distinct environments, is in order.

18.2 Periodicity

We'll first establish the meaning of periodicity in a typical mathematical context – the kind you may have seen in a past calculus or trig course. Then we'll define it in a more exotic mod-2 environment $\mathbb{Z}_{2^n} \cong (\mathbb{Z}_2)^n$.

18.2.1 Ordinary Periodicity

A function defined on the usual sets, e.g.,

$$f : \begin{Bmatrix} \mathbb{R} \\ \mathbb{C} \\ \mathbb{Z} \end{Bmatrix} \longrightarrow S \quad (S \text{ could be } \mathbb{R}, \mathbb{Z}_2, \text{ etc.}),$$

*is called **periodic** if there is a unique smallest $a > 0$ (called the **period**), with*

$$f(x + a) = f(x),$$

for all x in the domain of f .

(“Unique smallest” is redundant, but it forces us to remember the second of two separate requirements. First, there must exist *some* $a > 0$ with the property. If there does, we would call the *smallest* such a its period.) For example,

$$\sin(x + 2\pi) = \sin(x) \quad \text{for all } x,$$

and 2π is the smallest positive number with this property, so $a = 2\pi$ is its period. 4π and 12π satisfy the equality, but they're not as small as 2π , so they're not periods.

18.2.2 $(\mathbb{Z}_2)^n$ Periodicity

Let's change things up a bit. We define a different sort of periodicity which respects not ordinary addition, but mod-2 addition.

A function defined on $(\mathbb{Z}_2)^n$,

$$f : (\mathbb{Z}_2)^n \longrightarrow S \quad (S \text{ is typically } \mathbb{Z} \text{ or } (\mathbb{Z}_2)^m, m \geq n - 1),$$

is called $(\mathbb{Z}_2)^n$ **periodic** if there exists an $\mathbf{a} \in (\mathbb{Z}_2)^n$, $\mathbf{a} \neq \mathbf{0}$, such that,

$$\begin{aligned} &\text{for all } \mathbf{x} \neq \mathbf{y} \text{ in } (\mathbb{Z}_2)^n, \text{ we have} \\ &f(\mathbf{x}) = f(\mathbf{y}) \iff \mathbf{y} = \mathbf{x} \oplus \mathbf{a}. \end{aligned}$$

\mathbf{a} is called the *period* of f , with $(\mathbb{Z}_2)^n$ *periodicity* understood by context.

(Notice that in our definition, we rule out period $\mathbf{a} = \mathbf{0}$. We *could* have allowed it, in which case one-to-one functions – see definition below – are periodic with period $\mathbf{0}$, but this adds a trivial extra case to our discussion of Simon’s algorithm which has no pedagogical or historical value.)

There is a subtle but crucial message hidden in the definition. The *if-and-only-if* (\Leftrightarrow) aspect implies that if we find *even one pair* of elements, $\mathbf{x}' \neq \mathbf{y}'$ with $f(\mathbf{x}') = f(\mathbf{y}')$, then it must be true that $\mathbf{y}' = \mathbf{x}' \oplus \mathbf{a}$. Once we know that, we have the pleasant consequence that this one pair can be combined to recover the period using $\mathbf{a} = \mathbf{x}' \oplus \mathbf{y}'$. (Confirm this for yourself.) This is very different from the more common periodicity of, say, $\sin x$, wherein $\sin 0 = \sin \pi$, but π is not its period since for many other pairs (like .2 and .2 + π) $\sin x \neq \sin(x + \pi)$. It turns out that we won’t make use of this property in the quantum algorithm, but it will help when analyzing the Ω time complexity of the classical algorithm.

Due to the isomorphism between $(\mathbb{Z}_2)^n$ and $(\mathbb{Z}_{2^n}, \oplus)$, we can use the simple integer notation along with the \oplus operator to effect the same definition, as we demonstrate next.

18.2.3 \mathbb{Z}_{2^n} Periodicity

f is **\mathbb{Z}_{2^n} periodic** if there exists an $a \in \mathbb{Z}_{2^n}$, $a \neq 0$, such that,

$$\begin{aligned} &\text{for all } x \neq y \text{ in } \mathbb{Z}_{2^n}, \text{ we have} \\ &f(x) = f(y) \iff y = x \oplus a. \end{aligned}$$

a is called the *period* of f , with \mathbb{Z}_{2^n} *periodicity* understood by context.

As I mentioned, this is the same periodicity defined above and has the same implications. We will use the two periodicity definitions interchangeably going forward.

Examples of $(\mathbb{Z}_2)^n$ Periodicity

We implied that the range of f could be practically any set, S , but for the next few examples, let’s consider functions that map \mathbb{Z}_{2^n} *into* itself,

$$f : \mathbb{Z}_{2^n} \longrightarrow \mathbb{Z}_{2^n}$$

Collapsing One Bit – Periodic

A simple $(\mathbb{Z}_2)^n$ periodic function is one that preserves all the bits of x except for one, say the k th, and turns that one bit into a constant (either 0 or 1).

Let $n = 5$, $k = 2$, and the constant, 1. This is a “2nd bit collapse-to-1,” algebraically

$$f(x) \equiv \begin{pmatrix} x_4 \\ x_3 \\ 1 \\ x_1 \\ x_0 \end{pmatrix} = x_4 x_3 \ 1 \ x_1 x_0$$

If $n = 5$, $k = 4$, and the constant were 0, we’d have a “4th bit collapse-to-0,”

$$g(x) \equiv \begin{pmatrix} 0 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} = 0 \ x_3 x_1 x_1 x_0.$$

Let’s show why the first is $(\mathbb{Z}_2)^5$ periodic with period 4, and this will tell us why all the others of its ilk are periodic (with possibly a different period), as well.

Notation - Denote the *bit-flip* operation on the k th bit, \bar{x}_k , to mean

$$\bar{x}_k \equiv \begin{cases} 0, & \text{if } x_k = 1, \\ 1, & \text{if } x_k = 0. \end{cases}$$

[**Exercise.** Demonstrate that you can effect a bit-flip on the k th bit of x using $x \oplus 2^k$.]

I claim that the “2nd-bit collapse-to-1” function, f , is $(\mathbb{Z}_2)^n$ periodic with period $a = 4$. We must show

$$f(x) = f(y) \quad \Leftrightarrow \quad y = x \oplus 4.$$

- \Leftarrow : Assume $y = x \oplus 4$. That means (in vector notation)

$$\mathbf{x} = \begin{pmatrix} x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} \quad \text{and} \quad \mathbf{y} = \begin{pmatrix} y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_3 \\ \bar{x}_2 \\ x_1 \\ x_0 \end{pmatrix}.$$

Then,

$$f(\mathbf{x}) = \begin{pmatrix} x_4 \\ x_3 \\ 1 \\ x_1 \\ x_0 \end{pmatrix}, \quad \text{but, also} \quad f(\mathbf{y}) = \begin{pmatrix} x_4 \\ x_3 \\ 1 \\ x_1 \\ x_0 \end{pmatrix},$$

proving $f(x) = f(y)$.

- \Rightarrow : Assume $f(x) = f(y)$ for $x \neq y$. Since f does not modify any bits other than bit 2, we conclude $y_k = x_k$, except for (possibly) bit $k = 2$. But since $x \neq y$, *some* bit must be different between them, so it has to be bit-2. That is,

$$\mathbf{y} = \begin{pmatrix} y_4 \\ y_3 \\ y_2 \\ y_1 \\ y_0 \end{pmatrix} = \begin{pmatrix} x_4 \\ x_3 \\ \bar{x}_2 \\ x_1 \\ x_0 \end{pmatrix} = \mathbf{x} \oplus \mathbf{4},$$

showing that $y = x \oplus 4$. QED

Of course, we could have collapsed the 2nd bit to 0, or used any other bit, and gotten the same result. A single bit collapse is $(\mathbb{Z}_2)^n$ periodic with period 2^k , where k is the bit being collapsed.

Note - With single bit collapses, there are always exactly two numbers from the domain \mathbb{Z}_{2^n} that map into the same range number in \mathbb{Z}_{2^n} . With the example, f , just discussed,

$$\begin{array}{l} \left. \begin{array}{l} 00100 \\ 00000 \end{array} \right\} \mapsto 00100, \\ \left. \begin{array}{l} 10101 \\ 10001 \end{array} \right\} \mapsto 10101, \\ \left. \begin{array}{l} 11111 \\ 11011 \end{array} \right\} \mapsto 11111, \end{array}$$

and so on.

Collapsing More than One Bit – *Not* Periodic

A collapse of two or more bits can never be periodic, as we now show. For illustration, let's stick with $n = 5$, but consider a simultaneous collapse-to-1 of both the 2nd and 0th bit,

$$f(x) \equiv \begin{pmatrix} x_4 \\ x_3 \\ 1 \\ x_1 \\ 1 \end{pmatrix} = x_4 x_3 1 x_1 1$$

In this situation, for any x in the domain of f , you can find three others that map to the same $f(x)$. For example,

$$\left. \begin{array}{l} 00000 = 0 \\ 00001 = 1 \\ 00100 = 4 \\ 00101 = 5 \end{array} \right\} \mapsto 00101 = 5.$$

If there were some period, a , for this function, it would have to work for the first two listed above, meaning, $f(0) = f(1)$. For that to be true, we'd need $1 = 0 \oplus a$, which forces $a = 1$. But $a = 1$ won't work when you consider the first and *third* x , above: $f(0) = f(4)$, yet $4 \neq 0 \oplus 1$.

As you can see, this comes about because there are too many x s that get mapped to the same $f(x)$.

Let's summarize: bit collapsing gives us a periodic function only if we collapse exactly one bit (any bit) to a constant (either 0 or 1). However, a function which is a "multi-bit-collapser" (preserving the rest) can never be periodic.

So, what are the other periodic functions in the \mathbb{Z}_{2^n} milieu?

18.2.4 1-to-1, 2-to-1 and n -to-1

The answer requires a simple definition. First some more terminology.

Domain and Range

If f is a function, we'll use " $\text{dom}(f)$ " to designate the *domain* of f , and " $\text{ran}(f)$ " to designate the *range* of f . If $w \in \text{ran}(f)$, the "pre-image" of w is the set $\{x \mid f(x) = w\}$.

n -To-One

One-To-One: We say " f is 1-to-1" when $x \neq y \Rightarrow f(x) \neq f(y)$, for all x, y in $\text{dom}(f)$.

If $S \subseteq \text{dom}(f)$ then, even if f is not 1-to-1, overall, it can still be 1-to-1 on S . (Student to fill in the adjustments to the definition).

Two-To-One: We say " f is 2-to-1" when every w in $\text{ran}(f)$ has exactly two *pre-images* in $\text{dom}(f)$.

n -To-One: We say " f is n -to-1" when every w in $\text{ran}(f)$ has exactly n *pre-images* in $\text{dom}(f)$.

Lots of Periodic Functions

(In this small section we will set up a concept that you'll find useful in many future quantum algorithms: the partitioning of $\text{dom}(f)$ into subsets.)

We now characterize all \mathbb{Z}_{2^n} periodic functions.

Let's say you have one in your hands, call it f , and you even know its period, a . Buy three large trash bins at your local hardware store. Label one R , one Q and the third, S (for **S**ource pool). Dump all $x \in \text{dom}(f)$ (which is \mathbb{Z}_{2^n}) into the source pool,

S . We're going to be moving numbers from the source, S , into R and Q according to this plan:

1. Pick any $x \in S = \mathbb{Z}_{2^n}$. Call it r_0 (0 , because it's our first pick).
2. Generate r_0 's partner $q_0 = r_0 \oplus a$ (*partner*, because periodicity guarantees that f maps both r_0 and q_0 to the same image value and also that there is no other $x \in \mathbb{Z}_{2^n}$ which maps to that value.)
3. Toss r_0 into bin R and its partner, q_0 , (which you may have to dig around in S to find) into bin Q .
4. We have reduced the population of the source bin, S , by two: $S = S - \{r_0, q_0\}$. Keep going
5. Pick a new x from what's left of S . Call it r_1 (1 , because it's our second pick).
6. Generate r_1 's partner $q_1 = r_1 \oplus a$. (Again, we know that f maps both r_1 and q_1 to the same image value, and there is no other $x \in \mathbb{Z}_{2^n}$ which maps to that value.)
7. Toss r_1 into bin R and its partner, q_1 , into bin Q . S is further reduced by two and is now $S = S - \{r_0, q_0, r_1, q_1\}$.
8. Repeat this activity, each pass moving one value from bin S into bin R and its partner into bin Q until we have none of the original domain numbers left in S .
9. When we're done, half of the x s from $\text{dom}(f)$ will have ended up in R and the other half in Q . (However, since we chose the first of each pair at random, this was not a unique division of $\text{dom}(f)$, but that's not important.)

Here is the picture, when we're done with the above process:

$$\begin{aligned}\mathbb{Z}_{2^n} &= R \cup Q \\ &= \{ \dots, r_k, \dots \} \cup \{ \dots, q_k, \dots \}\end{aligned}$$

where R and Q are of equal size, and

$$f(r_k) = f(q_k)$$

but f is otherwise **one-to-one** on R and Q , individually.

From this construction, we can see a couple things:

1. Every periodic function in the \mathbb{Z}_{2^n} sense is 2-to-1. If a function is not 2-to-1, it doesn't have a prayer of being periodic.

2. We have a way to produce arbitrary \mathbb{Z}_{2^n} periodic functions. Pick a number you want to be the period and call it a . Start picking r_k numbers from \mathbb{Z}_{2^n} at random, each pick followed immediately by the computation of its partner, $q_k = r_k \oplus a$. Assign any image value you like to these two numbers. (The assigned image values don't even have to come from \mathbb{Z}_{2^n} – they can be the chickens in your yard or the contacts in your phone list. All that matters is that once you use one image value for an $r_k - q_k$ pair, you don't assign it to any future pair.) This will define a \mathbb{Z}_{2^n} periodic function with period a .

[**Exercise.** Generate two different \mathbb{Z}_{2^n} periodic functions.]

18.3 Simon's Problem

We've developed all the vocabulary and intuition necessary to understand the statement of Simon's problem, and the quantum fun can now begin.

Statement of Simon's Problem

Let $f : (\mathbb{Z}_2)^n \rightarrow (\mathbb{Z}_2)^n$ be $(\mathbb{Z}_2)^n$ periodic.
Find **a** .

(I made **a** bold because I phrased the problem in terms of the vector space $(\mathbb{Z}_2)^n$; had I used the integer notation of \mathbb{Z}_{2^n} , I would have said “find a ”, without the bold.)

It's not really necessary that $\text{ran}(f)$ be the same $(\mathbb{Z}_2)^n$ as that of its domain. In fact, the range is quite irrelevant. However, the assumption facilitates the learning process, and once we have our algorithm we can lose it.

Let's summarize some of the consequences of $(\mathbb{Z}_2)^n$ periodicity.

1. $f(\mathbf{y}) = f(\mathbf{x}) \Leftrightarrow \mathbf{y} = \mathbf{x} \oplus \mathbf{a}$
2. $f(\mathbf{y}) = f(\mathbf{x}) \Rightarrow \mathbf{a} = \mathbf{x} \oplus \mathbf{y}$
3. f is *two-to-one* on $(\mathbb{Z}_2)^n$

We can also state this in the equivalent \mathbb{Z}_{2^n} language.

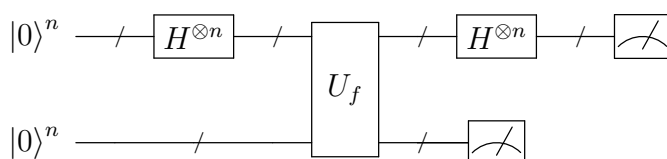
1. $f(y) = f(x) \Leftrightarrow y = x \oplus a$
2. $f(y) = f(x) \Rightarrow a = x \oplus y$
3. f is *two-to-one* on \mathbb{Z}_{2^n}

Pay particular attention to bullet 2. If we get a single pair that map to the same $f(x) = f(y)$, we will have our a . This will be used in the classical (although not quantum) analysis.

18.4 Simon's Quantum Circuit Overview and the Master Plan

18.4.1 The Circuit

A bird's eye view of the total circuit will give us an idea of what's ahead.



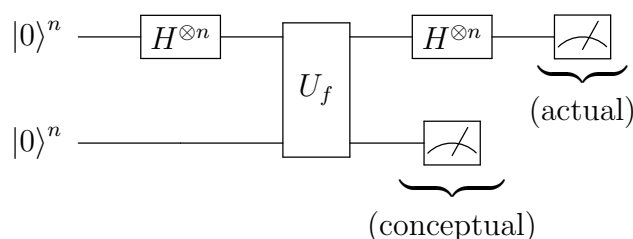
You see a familiar pattern. There are two multi-dimensional registers, the upper (which I will call the *A register*, the *data register* or even the *top line*, at my whim), and the lower (which I will call the *B register*, *target register* or *bottom line*, correspondingly.)

This is almost identical to the circuits of our recent algorithms, with the following changes:

- The target channel is “hatched,” reflecting that it has n component lines rather than one.
- We are sending a $|0\rangle^n$ into the bottom instead of a $|1\rangle$ (or even $|1\rangle^n$).
- We seem to be doing a measurement of both output registers rather than ignoring the target.

In fact, that third bullet concerning measuring the bottom register will turn out to be *conceptual* rather than *actual*. We *could* measure it, and it *would* cause the desired collapse of the upper register, however our analysis will reveal that we really don't have to. Nevertheless, we will keep it in the circuit to facilitate our understanding, label it as “conceptual,” and then abandon the measurement in the end when we are certain it has no practical value.

Here is the picture I'll be using for the remainder of the lesson.



[**Note:** I am suppressing the *hatched quantum wires* to produce a cleaner circuit. Since every channel is has n lines built-into it and we clearly see the kets and operators labeled with the “exponent” n , the hatched wires no longer serve a purpose.]

18.4.2 The Plan

We will prepare a couple CBS kets for input to our circuit, this time *both* will be $|0\rangle^n$. The data channel (top) will first encounter a multi-dimensional Hadamard gate to create a familiar superposition at the top. This sets up *quantum parallelism* which we found to be pivotal in past algorithms. The target channel's $|0\rangle^n$ will be sent directly into the oracle without pre-processing. This is the first time we will have started with a $|0\rangle$ rather than a $|1\rangle$ in this channel, a hint that we're not going to get a *phase kick-back* today. Instead, the *generalized Born rule*, (QM Trait #15") will turn out to be our best friend.

[**Preview:** When we expect to achieve our goals by applying the Born rule to a superposition, the oracle's target register should normally be fed a $|0\rangle^n$ rather than a $|1\rangle^n$.]

After the oracle, both registers will become entangled.

At that point, we conceptually test the B register output. This causes a collapse of both the top and bottom lines' states (from the *Born rule*), enabling us to know something about the A register. We'll analyze the A register's output – which resulted from this conceptual B register measurement – and discover that it has very special properties. Post processing the A register output by a second “re-organizing” Hadamard gate will seal the deal.

In the end, we may as well have measured the A register to begin with, since quantum entanglement authorizes the collapse using either line, and the A register is what we really care about.

Strategy

Our strategy will be to “load the dice” by creating a quantum circuit that spits out measured states which are “orthogonal” to the period a , i.e., $z \cdot a = 0 \bmod 2$. (This is not a true orthogonality, as we'll see, but everyone uses the term and so shall we. We'll discover that states which are orthogonal to a can often include the “vector” a , *itself*, another reason for the extra care with which we analyze our resulting “orthogonal” states.)

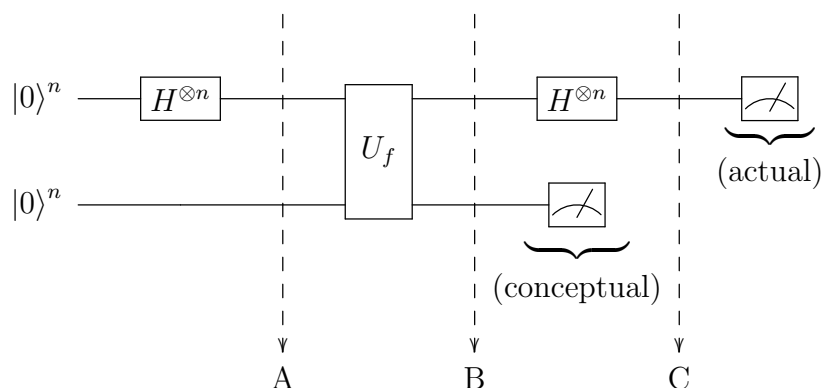
That sounds paradoxical; after all, we are looking for a , so why search for states *orthogonal* to it? Sometimes in quantum computing, it's easier to back-into the desired solution by sneaking up on it indirectly, and this turns out to be the case in Simon's problem. You can try to think of ways to get a more directly, and if you find an approach that works with better computational complexity, you may have discovered a new quantum algorithm. Let us know.

Because the states orthogonal to a are so much more likely than those that are not, we will quickly get a linearly independent set of $n-1$ equations with n unknowns, namely, $a \cdot w_k = 0$, for $k = 0, \dots, n-2$. We then augment this system instantly (using a direct classical technique) with an n th linearly independent equation, at which point we can solve the full, non-degenerate $n \times n$ system for a using fast and well-known

techniques.

18.5 The Circuit Breakdown

We need to break the circuit into sections to analyze it. Here is the segmentation:

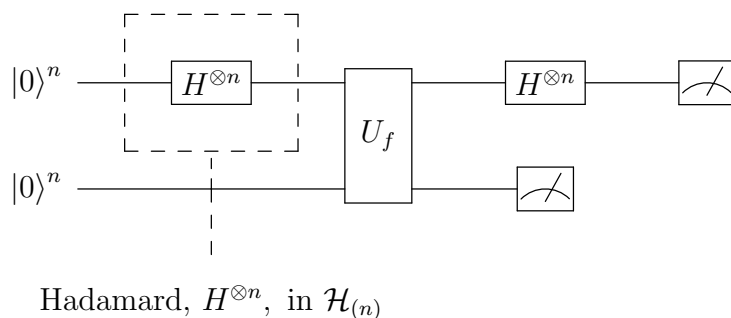


18.6 Circuit Analysis Prior to Conceptual Measurement: Point B

We now travel carefully through the circuit and analyze each component and its consequence.

18.6.1 Hadamard Preparation of the A Register

This stage of the circuit is identical in both logic and intent with the *Deutsch-Jozsa* and *Bernstein-Vazirani* circuits; it sets up *quantum parallelism* by producing a perfectly mixed entangled state, enabling the oracle to act on $f(x)$ for all possible x , simultaneously.



It never hurts to review the general definition of a gate like $H^{\otimes n}$. For any CBS $|x\rangle^n$, the 2^n -dimensional Hadamard gate is expressed in encoded form using the formula,

$$H^{\otimes n} |x\rangle^n = \left(\frac{1}{\sqrt{2}} \right)^n \sum_{y=0}^{2^n-1} (-1)^{x \odot y} |y\rangle^n,$$

where \odot is the mod-2 dot product. Today, I'll be using the alternate vector notation,

$$H^{\otimes n} |x\rangle^n = \left(\frac{1}{\sqrt{2}} \right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{x} \cdot \mathbf{y}} |y\rangle^n,$$

where the *dot product* between vector \mathbf{x} and vector \mathbf{y} is also assumed to be the mod-2 dot product. In the circuit, we have

$$|x\rangle^n \longrightarrow \boxed{H^{\otimes n}} \longrightarrow \left(\frac{1}{\sqrt{2}} \right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{x} \cdot \mathbf{y}} |y\rangle^n,$$

which, when applied to $|0\rangle^n$, reduces to

$$|0\rangle^n \longrightarrow \boxed{H^{\otimes n}} \longrightarrow \left(\frac{1}{\sqrt{2}} \right)^n \sum_{y=0}^{2^n-1} |y\rangle^n,$$

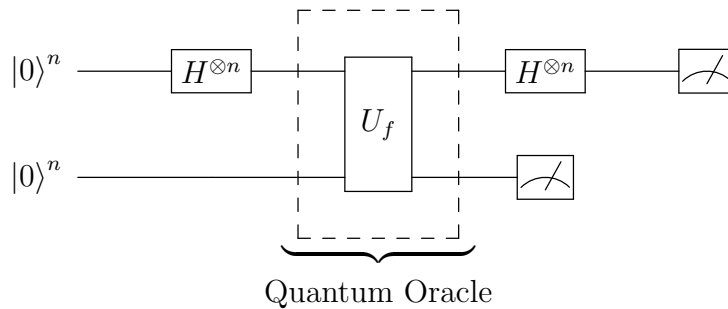
or, returning to the usual computational basis notation, $|x\rangle^n$, for the summation index is

$$|0\rangle^n \longrightarrow \boxed{H^{\otimes n}} \longrightarrow \left(\frac{1}{\sqrt{2}} \right)^n \sum_{x=0}^{2^n-1} |x\rangle^n.$$

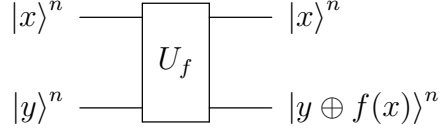
You'll recognize the output state of this Hadamard operator as the n th order x -basis CBS ket, $|0\rangle_{\pm}^n$. It reminds us that not only do Hadamard gates provide quantum parallelism but double as a $z \leftrightarrow x$ basis conversion operators.

18.6.2 The Quantum Oracle on CBS Inputs

Next, we look at this part of the circuit:

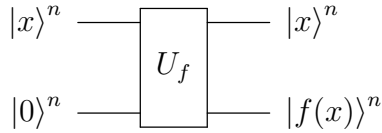


Due to the increased B channel width, we had better review the precise definition of the higher dimensional oracle. It's based on CBS kets going in,



$$|x\rangle^n |y\rangle^n \xrightarrow{U_f} |x\rangle^n |y \oplus f(x)\rangle^n ,$$

and from there we extend to general input states, linearly. We actually constructed the matrix of this oracle and proved it to be unitary in our lesson on *quantum oracles*. Today, we need only consider the case of $y = 0$:



$$|x\rangle^n |0\rangle^n \xrightarrow{U_f} |x\rangle^n |f(x)\rangle^n$$

In Words

We are

1. taking the B register CBS input $|y\rangle^n$, which is $|0\rangle^n$ in this case, and extracting the integer representation of y , namely 0,
2. applying f to the integer x (of the A register CBS $|x\rangle^n$) to form $f(x)$,
3. noting that both 0 and $f(x)$ are $\in \mathbb{Z}_{2^n}$,
4. forming the mod-2 sum of these two integers, $0 \oplus f(x)$, which, of course is $f(x)$, and
5. using the result to define the output of the oracle's B register, $|f(x)\rangle^n$.
6. Finally, we recognize the output to be a separable state of the two output registers, $|x\rangle^n |f(x)\rangle^n$.

Just to remove any lingering doubts, assume $n = 5$, $x = 18$, and $f(18) = 7$. Then the above process yields

1. $|0\rangle^5 \longrightarrow 0$
2. $|18\rangle^5 \longrightarrow 18 \xrightarrow{f(18)=7} 7$

$$3. 0 \oplus 7 = 00000 \oplus 00111 = 00111 = 7,$$

$$4. 7 \longrightarrow |7\rangle^5$$

$$5. |18\rangle^5 \otimes |0\rangle^5 \xrightarrow{U_f} |18\rangle^5 \otimes |7\rangle^5$$

18.6.3 The Quantum Oracle on Hadamard Superposition Inputs

Next, we invoke linearity to the maximally mixed superposition state $|0\rangle_{\pm}^n$ going into the oracle's top register.

Reminder. I've said it before, but it's so important in a first course such as this that I'll repeat myself. The bottom register's output is not f applied to the superposition state. f only has meaning over its domain \mathbb{Z}_{2^n} , which corresponds to the *finite* set of z -basis CBS kets $\{|x\rangle\}$. It has no meaning when applied to sums, especially weighted sums (by real or complex amplitudes) of these preferred CBSs.

By linearity, U_f distributes over all the terms in the maximally mixed input $|0\rangle_{\pm}^n$, at which point we apply the result of the last subsection, namely

$$U_f(|x\rangle^n |0\rangle^n) = |x\rangle^n |f(x)\rangle^n,$$

to the individual summands to find that

$$\left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x\rangle^n |0\rangle^n \xrightarrow{U_f} \left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x\rangle^n |f(x)\rangle^n.$$

This is a weighted sum of separable products. (The weights are the same for each separable product in the sum: $(1/\sqrt{2})^n$.) That sum is not, as a whole, separable, which makes it impossible to visualize directly on the circuit diagram unless we combine the two outputs into a single, entangled, output register. However we do have an interpretation that relates to the original circuit.

- The output state is a superposition of separable terms $|x\rangle^n \frac{|f(x)\rangle^n}{(\sqrt{2})^n}$. But this is exactly the kind of sum the *generalized Born rule* needs,

$$|\varphi\rangle^{n+m} = |0\rangle_A^n |\psi_0\rangle_B^m + |1\rangle_A^n |\psi_1\rangle_B^m + \cdots + |2^n - 1\rangle_A^n |\psi_{2^n-1}\rangle_B^m,$$

so an A -measurement of “ x ” would imply a B -state collapse to its (normalized) partner, $|f(x)\rangle^n$.

- A similar conclusion would be drawn if we chose to measure the B register first, but we'll get to that slightly more complicated alternative in a moment. (Sneak preview: there are two – not one – pre-images x for every $f(x)$ value.)

- Each of the 2^n orthogonal terms in the superposition has amplitude $(1/\sqrt{2})^n$, so the probability that a measurement by A will collapse the superposition to any one of them is $\left((1/\sqrt{2})^n\right)^2 = 1/2^n$.

[**Exercise.** Why are the terms orthogonal? **Hint:** inner product of tensors.]

[**Exercise.** Look at the sum in our specific situation: $\sum_{x=0}^{2^n-1} |x\rangle^n |f(x)\rangle^n$. QM

Trait #6 (*Probability of Outcomes*) assumes we start with an expansion along some computational basis, but this expansion only has 2^n terms, so can't be a basis for $A \otimes B$ which has $2^n \cdot 2^n = 2^{2n}$ basis vectors. Why can we claim that the scalars $(1/\sqrt{2})^n$ are amplitudes of collapse to one of these states? **Hint:** While the kets in the sum do not comprise a full basis, they are all distinct CBS kets. (Are they distinct? each $f(x)$ appears twice in the sum. But that doesn't destroy the linear independence of the tensor CBS kets in that sum, because) Therefore, we can add the missing CBS kets into the sum as long as we accompany them by 0 scalar weights. Now we can apply **Trait #6**.]

[**Exercise.** Look at the the more general sum in the *Born rule*: $\sum_{x=0}^{2^n-1} |x\rangle_A^n |\psi_x\rangle_B^m$.

QM **Trait #6** (*Probability of Outcomes*) assumes we start with an expansion along some computational basis, but this expansion only has 2^n terms, so can't be a basis for $A \otimes B$ which has $2^n \cdot 2^m = 2^{n+m}$ basis vectors. Why can we claim that the scalars $(1/\sqrt{2})^n$ are amplitudes of collapse to one of these states? **Hint:** Force this to be an expansion along the tensor CBS by expanding each $|\psi_x\rangle_B^m$ along the B -basis then distributing the $|x\rangle_A^n$ s. Now we meet the criterion of **Trait #6**.]

18.6.4 Partitioning the Domain into Cosets

It's time to use the fact that f is \mathbb{Z}_{2^n} periodic with (unknown) period a to help us rewrite the output of the Oracle's B register prior to the conceptual measurement. \mathbb{Z}_{2^n} periodicity tells us that the domain can be partitioned (in more than one way) into two disjoint sets, R and Q ,

$$\begin{aligned}\mathbb{Z}_{2^n} &= R \cup Q \\ &= \{\dots, x, \dots\} \cup \{\dots, x \oplus a, \dots\}.\end{aligned}$$

Cosets

Q can be written as $R \oplus a$, that is, we "translate" the entire subset R by adding a number to every one of its members resulting in a new subset, Q . Mathematicians say that $Q = R \oplus a$ is a *coset* of the set R . Notice that R is a coset of itself since $R = R \oplus 0$. In this case, the two cosets, R and $Q = R \oplus a$ partition the domain of f into two equal and distinct sets.

18.6.5 Rewriting the Output of the Oracle's B Register

Our original expression for the oracle's complete entangled output was

$$\left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x\rangle^n |f(x)\rangle^n,$$

but our new partition of the domain will give us a propitious way to rewrite this. Each element $x \in R$ has a unique partner in Q satisfying

$$\left. \begin{array}{c} x \\ x \oplus a \end{array} \right\}_{x \in R} \xrightarrow{f} f(x).$$

Using this fact, we only need to sum the B register output over R (half as big as \mathbb{Z}_{2^n}) and include both x and $x \oplus a$ in each term,

$$\begin{aligned} \left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x\rangle^n |f(x)\rangle^n &= \left(\frac{1}{\sqrt{2}}\right)^n \sum_{x \in R} \left(|x\rangle^n + |x \oplus a\rangle^n\right) |f(x)\rangle^n \\ &= \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{x \in R} \left(\frac{|x\rangle^n + |x \oplus a\rangle^n}{\sqrt{2}}\right) |f(x)\rangle^n \end{aligned}$$

I moved one of the factors of $1/\sqrt{2}$ into the sum so we could see

1. how the new sum consists of normalized states (length 1), and
2. the common amplitude remaining on the outside, nicely produces a normalized state overall, since now there are half as many states in the sum, but each state has twice the probability as before.

Swapping the roles of channels A and B for the *Born Rule*

The last rearrangement of the sum had a fascinating consequence. While the terms still consist of separable products from the A and B channels, now it is the B channel that has basis CBS kets, and the A channel that does not.

How can we see this? Each $|f(x)\rangle^n$ was *always* a CBS state – $f(x)$ is an integer from 0 to $2^n - 1$ and so corresponds to a CBS ket – but the original sum was plagued by its appearing twice (destroying potential linear independence of the terms), so we couldn't view the sum as an expansion along the B -basis. After consolidating all the pre-image pairs, we now have only one $|f(x)\rangle^n$ term for each x in the sum: we have factored the expansion along the B -basis. In the process, the A -factors became superpositions of A CBS kets, now mere $|\psi_x\rangle_A^n$ s in the general population. This reverses the roles of A and B and allows us to talk about measuring B along the CBS, with that measurement selecting one of the non-CBS A factors.

The upshot is that we can apply the Born rule in reverse; we'll be measuring the B register and forcing the A register to collapse into one of its “binomial” superpositions. Let's do it. But first, we should give recognition to a reusable design policy.

The Lesson: $|0\rangle^n$ into the Oracle's B Register

This all worked because we chose to send the CBS $|0\rangle^n$ into the oracle's B register. Any other CBS into that channel would not have created the nice terms $|x\rangle^n |f(x)\rangle^n$ of the oracle's entangled output. After factoring out terms that had common $|f(x)\rangle^n$ components in the B register, we were in a position to collapse along the B -basis and pick out the attached sum in the A register.

Remember this. It's a classic trick that can be tried when we want to select a small subset of A register terms from the large, perfectly mixed superposition in that register. It will typically lead to a probabilistic outcome that won't necessarily settle the algorithm in a single evaluation of the oracle, but we expect it to give a valuable result that can be combined with a few more evaluations (loop passes) of the oracle. This is the lesson we learn today that will apply next time when we study *Shor's period-finding algorithm*.

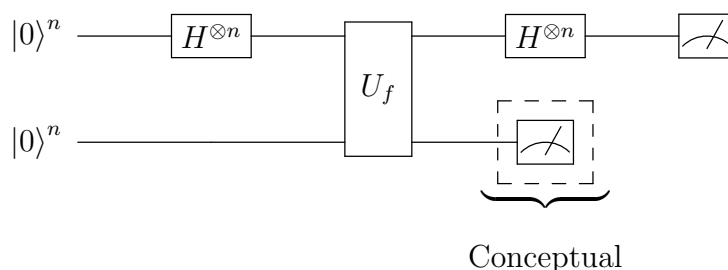
In contrast, when we were looking for a deterministic solution in algorithms like *Deutsch-Jozsa* and *Bernstein-Vazirani*, we fed a $|1\rangle$ into the B register and used the *phase kick-back* to give us an answer in a single evaluation.

$$\begin{array}{ll} |1\rangle \text{ into oracle's } B \text{ register} & \longrightarrow \text{phase kick-back ,} \\ |0\rangle^n \text{ into oracle's } B \text{ register} & \longrightarrow \text{Born rule .} \end{array}$$

18.7 Analysis of the Remainder of the Circuit: Measurements

18.7.1 Hypothetical Measurement of the B Register

Although we won't really need to do so, let's imagine what happens if we were to apply the generalized Born rule now using the rearranged sum (that turned the B channel into the CBS channel).



Each B register measurement of “ $f(x)$ ” will be attached to not one, but two, input A register states. Thus, measuring B first, while collapsing A , actually produces merely

a *superposition* in that register, not a single, unique x from the domain. It narrows things down considerably, but not completely,

$$\left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{x \in R} \left(\frac{|x\rangle^n + |x \oplus a\rangle^n}{\sqrt{2}} \right) |f(x)\rangle^n \longrightarrow \boxed{\text{Measurement}} \\ \searrow \left(\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}} \right) |f(x_0)\rangle^n \\ \text{(Here, } \searrow \text{ means } \textit{collapses to}.)$$

Well that's good, great and wonderful, but if after measuring the post-oracle B register, we were to measure line A , it would collapse to one of two states, $|x_0\rangle$ or $|x_0 + a\rangle$, but we wouldn't know which nor would we know its unsuccessful companion (the one to which the state didn't collapse). There seems to be no usable information here. As a result we *don't* measure A ... yet.

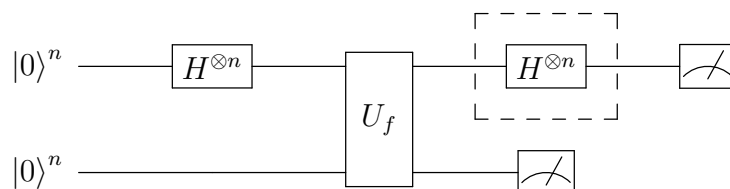
Let's name the collapsed – but unmeasured – superposition state in the A register $|\psi_{x_0}\rangle^n$, since it is determined by the measurement “ $f(x_0)$ ” of the collapsed B register,

$$|\psi_{x_0}\rangle^n \equiv \left(\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}} \right).$$

Guiding Principle: Narrow the Field. We stand back and remember this stage of the analysis for future use. Although a conceptual measurement of B does *not* produce an individual CBS ket in register A , it *does* result in a significant narrowing of the field. This is how the big remaining quantum algorithms in this course will work.

18.7.2 Effect of a Final Hadamard on A Register

In an attempt to coax the superposition ket $|\psi_{x_0}\rangle^n \in \mathcal{H}_{(n)}$ to cough up useful information, we take $H^{\otimes n} |\psi_{x_0}\rangle^n$. This requires that we place an $H^{\otimes n}$ gate at the output of the oracle's A register:



[**Apology.** I can't offer a simple reason why anyone should be able to “intuit” a Hadamard as the post-oracle operator we need. Unlike *Deutsch-Jozsa*, today we are not measuring along the x -basis, our motivation for the final $H^{\otimes n}$ back then. However, there *is* a small technical theorem about a Hadamard applied to a binomial superposition of CBSs of the form $(|x\rangle^n + |y\rangle^n) / \sqrt{2}$ which is relevant, and perhaps this inspired Simon and his compadres.]

Continuing under the assumption that we measure an $f(x_0)$ at the B register output, thus collapsing both registers, we go on to work with the resulting superposition $|\psi_{x_0}\rangle$ in the A register. Let's track its progress as we apply the Hadamard gate to it. As with all quantum gates, $H^{\otimes n}$ is linear so moves past the sum, and we get

$$\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}} \quad \xrightarrow{H^{\otimes n}} \quad \frac{H^{\otimes n}|x_0\rangle^n + H^{\otimes n}|x_0 \oplus a\rangle^n}{\sqrt{2}}$$

The Hadamard of the individual terms is

$$H^{\otimes n} |x_0\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y} \cdot \mathbf{x}_0} |y\rangle^n$$

and

$$H^{\otimes n} |x_0 \oplus a\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y} \cdot (\mathbf{x}_0 \oplus \mathbf{a})} |y\rangle^n$$

Focus on the integer coefficient

$$(-1)^{\mathbf{y} \cdot (\mathbf{x}_0 \oplus \mathbf{a})}.$$

This can be simplified by observing two facts

1. The mod-2 dot product distributes over the mod-2 sum \oplus , and
2. $(-1)^{p \oplus q} = (-1)^p (-1)^q$, for p and q in \mathbb{Z}_2^n . (**Danger:** while it's true for a base of -1, it is *not* for a general complex base c).

[**Exercise.** Prove the second identity. **One idea:** Break it into two cases, $p \oplus q$ even and $p \oplus q$ odd, then consider each one separately.]

[**Exercise.** Find a complex c , for which $(c)^{p \oplus q} \neq (c)^p (c)^q$. **Hint:** Use the simplest case, $n = 1$.]

Combining both facts, we get

$$(-1)^{\mathbf{y} \cdot (\mathbf{x}_0 \oplus \mathbf{a})} = (-1)^{\mathbf{y} \cdot \mathbf{x}_0} (-1)^{\mathbf{y} \cdot \mathbf{a}},$$

so

$$\begin{aligned}
& \frac{H^{\otimes n} |x_0\rangle^n + H^{\otimes n} |x_0 \oplus a\rangle^n}{\sqrt{2}} \\
&= \frac{\left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y} \cdot \mathbf{x}_0} \left(1 + (-1)^{\mathbf{y} \cdot \mathbf{a}}\right) |y\rangle^n}{\sqrt{2}} \\
&= \left(\frac{1}{\sqrt{2}}\right)^{n+1} \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y} \cdot \mathbf{x}_0} \left(1 + (-1)^{\mathbf{y} \cdot \mathbf{a}}\right) |y\rangle^n.
\end{aligned}$$

18.7.3 The Orthogonality of A Register Output Relative to the Unknown Period a

Here's where we are:

$$\begin{aligned}
& H^{\otimes n} \left[\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}} \right] \\
&= \left(\frac{1}{\sqrt{2}}\right)^{n+1} \sum_{y=0}^{2^n-1} (-1)^{\mathbf{y} \cdot \mathbf{x}_0} \left(1 + (-1)^{\mathbf{y} \cdot \mathbf{a}}\right) |y\rangle^n.
\end{aligned}$$

The integer expression in the parentheses is seen to be

$$1 + (-1)^{\mathbf{y} \cdot \mathbf{a}} = \begin{cases} 0, & \text{if } \mathbf{y} \cdot \mathbf{a} = 1 \pmod{2} \\ 2, & \text{if } \mathbf{y} \cdot \mathbf{a} = 0 \pmod{2} \end{cases},$$

so we can omit all those 0 terms which correspond to $\mathbf{y} \cdot \mathbf{a} = 1 \pmod{2}$, leaving

$$H^{\otimes n} \left[\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}} \right] = \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{\substack{\mathbf{y} \cdot \mathbf{a} = 0 \\ \pmod{2}}}^{2^n-1} (-1)^{\mathbf{y} \cdot \mathbf{x}_0} |y\rangle^n.$$

Note that the sum is now over only 2^{n-1} , or exactly *half*, of the original 2^n CBSs.

[Exercise. Show that, for any fixed number $a \in \mathbb{Z}_{2^n}$ (or its equivalent vector $\mathbf{a} \in (\mathbb{Z}_2)^n$) the set of all x with $x \odot a = 0$ (or \mathbf{x} with $\mathbf{x} \cdot \mathbf{a} = 0$) is exactly half the numbers (vectors) in the set.]

Mod-2 Orthogonality vs. Hilbert Space Orthogonality

Avoid confusion. Don't forget that these dot products (like $\mathbf{y} \cdot \mathbf{a}$) are *mod-2* dot products of vectors in $(\mathbb{Z}_2)^n$. This has nothing to do with the Hilbert space inner product ${}^n\langle y | a \rangle^n$, an operation on quantum states.

When we talk about *orthogonality* of vectors or numbers at this stage of the analysis, we mean the mod-2 dot product, not the state space inner product. Indeed, there is nothing new or interesting about Hilbert space orthogonality at this juncture: CBS kets *always* form an orthonormal set, so each one has inner product = 1 with itself and inner product = 0 with all the rest. However, that fundamental fact doesn't help us here. The mod-2 dot product *does*.

Pause for an Example

If $n = 4$, and

$$a = 5 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix},$$

then

$$\begin{aligned} & \{ \mathbf{y} \mid \mathbf{y} \cdot \mathbf{a} = 0 \pmod{2} \} \\ &= \left\{ \begin{pmatrix} a_3 \\ 0 \\ a_1 \\ 0 \end{pmatrix} \right\}_{a_1, a_3 \in \{0,1\}} \cup \left\{ \begin{pmatrix} a_3 \\ 1 \\ a_1 \\ 1 \end{pmatrix} \right\}_{a_1, a_3 \in \{0,1\}}, \end{aligned}$$

which consists of *eight* of the original $2^4 = \textit{sixteen}$ $(\mathbb{Z}_2)^4$ vectors associated with the CBSs. Therefore, there are $2^{n-1} = 2^3 = 8$ terms in the sum, exactly normalized by the $(1/\sqrt{2})^{n-1} = (1/\sqrt{2})^3 = (1/\sqrt{8})$ out front.

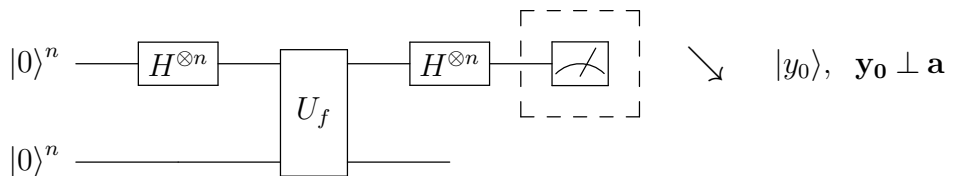
Note: In this case, a is in the set of mod-2 vectors which are orthogonal to it.

[**Exercise.** Characterize all a which are orthogonal to themselves, and explain why this includes half of all states in our total set.]

Returning to our derivation, we know that once we measure the B register and collapse the states into those associated with a specific $|f(x_0)\rangle$, the A register can be post-processed with a Hadamard gate to give

$$\frac{|x_0\rangle^n + |x_0 \oplus a\rangle^n}{\sqrt{2}} \xrightarrow{H^{\otimes n}} \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{\substack{\mathbf{y} \cdot \mathbf{a} = 0 \\ \pmod{2}}}^{|2^{n-1}|} (-1)^{\mathbf{y} \cdot \mathbf{x}_0} |y\rangle^n.$$

All of the vectors in the final A register superposition are orthogonal to a , so we can now safely measure that mixed state and get some great information:



We don't know *which* y_0 among the 2^{n-1} y s we will measure – that depends on the whimsy of the collapse, and they're all equally likely. However, we just showed that they're all orthogonal to a , including y_0 .

Warning(s): $y = 0$ Possible

This is one possible snag. We might get a 0 when we test the A register. It is a possible outcome, since $0 = \mathbf{0}$ is mod-2 orthogonal to *everything*. The probabilities are low – $1/2^n$ – and you can test for it and throw it back if that happens. We'll account for this in our probabilistic analysis, further down. While we're at it, remember that **a , itself, might get measured**, but that's okay. We won't know it's a , and the fact that it might be won't change a thing that follows.

18.7.4 Foregoing the Conceptual Measurement

We haven't yet spoken of the actual A register measurement *without* first measuring the B register. Now that you've seen the case with the simplifying B register measurement, you should be able to follow this full development that omits that step.

If we don't measure B first, then we can't say we have collapsed into any particular $f(x_0)$ state. So the oracle's output must continue to carry the full entangled summation

$$\left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{x \in R} \left(\frac{|x\rangle^n + |x \oplus a\rangle^n}{\sqrt{2}} \right) |f(x)\rangle^n$$

through the final $[H^{\otimes n} \otimes \mathbf{1}^{\otimes n}]$. This would add an extra sum $\sum_{x \in R}$ to all of our expressions.

$$\begin{aligned} & [H^{\otimes n} \otimes \mathbf{1}^{\otimes n}] \left(\frac{1}{\sqrt{2}}\right)^n \sum_{x \in R} \left(|x\rangle^n + |x \oplus a\rangle^n \right) |f(x)\rangle^n \\ &= \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{x \in R} [H^{\otimes n} \otimes \mathbf{1}^{\otimes n}] \left[\left(\frac{|x\rangle^n + |x \oplus a\rangle^n}{\sqrt{2}} \right) |f(x)\rangle^n \right] \\ &= \left(\frac{1}{\sqrt{2}}\right)^{n-1} \sum_{x \in R} \left(\frac{H^{\otimes n} |x\rangle^n + H^{\otimes n} |x \oplus a\rangle^n}{\sqrt{2}} \right) |f(x)\rangle^n . \end{aligned}$$

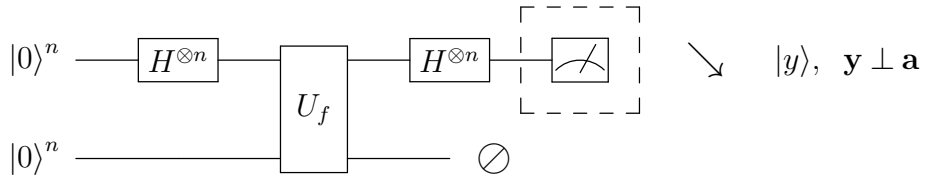
We can now cite the – still valid – result that led to expressing the Hadamard fraction

as a sum of CBS kets satisfying $\mathbf{y} \cdot \mathbf{a} = 0 \pmod{2}$, so the last expression is

$$\begin{aligned}
&= \left(\frac{1}{\sqrt{2}} \right)^{n-1} \sum_{x \in R} \left(\left(\frac{1}{\sqrt{2}} \right)^{n-1} \sum_{\substack{\mathbf{y} \cdot \mathbf{a} = 0 \\ (\text{mod } 2)}} (-1)^{\mathbf{y} \cdot \mathbf{x}} |y\rangle^n \right) |f(x)\rangle^n \\
&= \left(\frac{1}{\sqrt{2}} \right)^{2n-2} \sum_{\substack{\mathbf{y} \cdot \mathbf{a} = 0 \\ (\text{mod } 2)}} |y\rangle^n \left(\sum_{x \in R} (-1)^{\mathbf{y} \cdot \mathbf{x}} |f(x)\rangle^n \right).
\end{aligned}$$

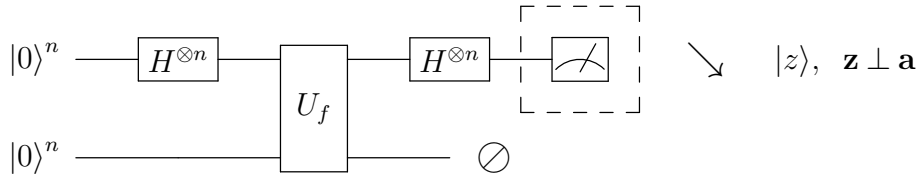
While our *double* sum has more overall terms than before, they are all confined to those y which are (mod-2) orthogonal to a . In fact, we don't have to apply the Born rule this time, because all that we're claiming is an A register collapse to one of the 2^{n-1} CBS kets $|y\rangle^n$ which we get compliments of quantum mechanics: *third postulate* + *post measurement collapse*.

Therefore, when we measure only the A register of this larger superposition, the collapsed state is still some y orthogonal to a .



A Small Change in Notation

Because I often use the variable y for general CBS states, $|y\rangle$, or summation variables, \sum_y , I'm going to switch to the variable z for the measured *orthogonal* output state, as in $|z\rangle$, $\mathbf{z} \perp \mathbf{a}$. We'll then have a mental cue for the rest of the lecture, where \mathbf{z} will always be a mod-2 vector orthogonal to \mathbf{a} . With this last tweak, our final circuit result is



18.8 Circuit Analysis Conclusion

In a single application of the circuit, we have found our first z , with $\mathbf{z} \perp \mathbf{a}$. We would like to find $n - 1$ such vectors, all linearly independent as a set, so we anticipate

sampling the circuit several more times. How long will it take us to be relatively certain we have $n - 1$ independent vectors? We explore this question next.

I will call the set of *all* vectors orthogonal to a either \mathbf{a}_\perp (if using vector notation) or a_\perp (if using \mathbb{Z}_{2^n} notation). It is pronounced “*a-perp*.”

[**Exercise.** Working with the vector space $(\mathbb{Z}_2)^n$, show that \mathbf{a}_\perp is a vector subspace. (For our purposes, it’s enough to show that it is closed under the \oplus operation).]

[**Exercise.** Show that those vectors which are *not* orthogonal to \mathbf{a} do *not* form a subspace.]

18.9 Simon’s Algorithm

18.9.1 Producing $n - 1$ Linearly Independent Vectors

[**Notational Alert.** By now, we’re all fluent in translating from the *vectors* $(\mathbf{a}, \mathbf{w}_k)$ of \mathbb{Z}_{2^n} to the *encoded decimals* (a, w_k) of $(\mathbb{Z}_2)^n$ notation, so be ready to see me switch between the two depending on the one I think will facilitate your comprehension. I’ll usually use encoded decimals and give you **notational alerts** when using vectors.]

What we showed is that we can find a vector orthogonal to a in a single application of our circuit. We need, however, to find not one, but $n - 1$, linearly-independent vectors, z , that are orthogonal to a . Does repeating the process $n - 1$ times do the trick? Doing so would certainly manufacture

$$\{z_0, z_1, z_2 \dots, z_{n-2}\}$$

with

$$z_k \perp a \quad \text{for } k = 0, \dots, n - 2,$$

however, that’s not quite good enough. Some of the z_k might be a linear combination of the others or even be repeats. For this to work, we’d need each one to not only be orthogonal to a , but linearly independent of *all the others* as well.

Pause for an Example

If $n = 4$, and

$$a = 5 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix},$$

suppose that the circuit produced the three vectors

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

after three circuit invocations. While all three are orthogonal to a , they do not form a linearly-independent set. In this case, $3 = n - 1$ was not adequate. Furthermore, a fourth or fifth might not even be enough (if, say, we got some repeats of these three).

Therefore, we must perform this process m times, $m \geq n - 1$, until we have $n - 1$ linearly-independent vectors. (We don't have to say that they must be orthogonal to a , since the circuit construction already guarantees this.) How large must m be, and can we ever be sure we will succeed?

18.9.2 The Algorithm

We provide a general algorithm in this section. In the sections that follow, we'll see that we are guaranteed to succeed in polynomial time $O(n^4)$ with probability arbitrarily close to 1. Finally, I'll tweak the algorithm just a bit and arrive at an implementation that achieves $O(n^3)$ performance.

Whether it's $O(n^4)$, $O(n^3)$ or any similar *big-O* complexity, it will be a significant relative speed-up over classical computing which has *exponential* growth, even if we accept a non-deterministic classical solution. (We'll prove that in the final section of this lesson.) The problem is *hard*, classically, but easy quantum mechanically.

Here is Simon's algorithm for \mathbb{Z}_{2^n} -period finding. To be sure that it gives a polynomial time solution, we must eventually verify that

1. we only have to run the circuit a polynomial number of times (in n) to get $n - 1$ linearly independent z s which are orthogonal to a with arbitrarily good confidence, and
2. the various classical tasks – like checking that a set of vectors is linearly independent and solving a series of n equations – are all of polynomial complexity in n , individually.

We'll do all that in the following sections, but right now let's see the algorithm:

- Select an integer, T , which reflects an acceptable failure probability of $1/2^T$.
- Initialize a set \mathcal{W} to the empty set. \mathcal{W} will eventually contain a growing number of $(\mathbb{Z}_2)^n$ vectors,
- Repeat the following loop at most $n + T$ times.
 1. Apply Simon's circuit.
 2. Measure the output of the final $H^{\otimes n}$ to get z .
 3. Use a classical algorithm to determine whether or not z is linearly dependent on the vectors in \mathcal{W} .
 - If it is independent, name it w_j , where j is the number of elements already stored in \mathcal{W} , and add w_j to \mathcal{W} .

- * if $j = n - 2$, we have $n - 1$ linearly-independent vectors in \mathcal{W} and are done; *break* the loop.
- If it is not independent (which includes special the case $z = 0$, even when \mathcal{W} is still empty), then continue to the next pass of the loop.
- If the above loop ended naturally (i.e., not from the *break*) after $n + T$ full passes, we failed.
- Otherwise, we succeeded. Add an n th vector, w_{n-1} , which is linearly independent to this set (and therefore not orthogonal to a , by a previous exercise), done easily using a simple classical observation, demonstrated below. This produces a system of n independent equations satisfying

$$w_k \cdot a = \begin{cases} 0, & k = 0, \dots, n - 2 \\ 1, & k = n - 1 \end{cases}$$

which has a unique non-zero solution.

- Use a classical algorithm to solve the systems of n equations for a .

Note: By supplying the n th vector (a fast, easy addition of cost $O(n)$), we get a full system requiring no extra quantum samples and guaranteeing that our system yields a , unequivocally.

Time and Space Complexity

We will run our circuit $n + T = O(n)$ times. There will be some classical algorithms that need to be tacked on, producing an overall growth rate of about $O(n^3)$ or $O(n^4)$, depending on the cleverness of the overall design. But let's back-up a moment.

I've only mentioned *time* complexity. What about *circuit*, a.k.a. *spatial*, complexity?

There is a circuit that must be built, and we can tell by looking at the number of inputs and outputs to the circuit ($2n$ each) that the *spatial complexity* is $O(n)$. I mentioned during our *oracle lesson* that we can ignore the internal design of the oracle because it, like its associated f , may be arbitrarily complex; all we are doing in these algorithms is getting a *relativized* speed-up. But even setting aside the oracle's internals, we can't ignore the $O(n)$ spatial input/output size leading to/from the oracle as well as the circuit as a whole.

Therefore, if I operate the circuit in an algorithmic “while loop” of $n + T = O(n)$ passes, the time complexity of the quantum circuit (not counting the classical tools to be added) is $O(n)$. Meanwhile, each pass of the circuit uses $O(n)$ wires and gates, giving a more honest $O(n^2)$ growth to the $\{ \text{algorithm} + \text{circuit} \}$ representing (only) the quantum portion of the algorithm.

So, why do I (and others) describe the complexity of the quantum portion of the algorithm to be $O(n)$ and not $O(n^2)$?

We actually touched on these reasons when describing why we tend to ignore hardware growth for these relatively simple circuits. I'll reprise the two reasons given at that time.

- Our interest is in *relative speed-up* of quantum over classical methods. Since we need $O(n)$ inputs/outputs in both the classical and quantum circuit, it is unnecessary to carry the same cost of $O(n)$ on the books – it cancels out when we compare the two regimes.
- The quantum circuits grow slowly (linearly or logarithmically) compared to more expensive classical post processing whose time complexity is often at least *quadratic* and (this is important) *in series* with the quantum circuit. These attending sub-algorithms therefore render an $O(n)$ or $O(n^2)$ quantum growth invisible.

So, be aware of the true overall complexity, but understand that it's usually unnecessary to multiply by a linear circuit complexity when doing computational accounting.

18.9.3 Strange Behavior

This section is not required but may add insight and concreteness to your understanding of the concepts.

[**Notational Alert.** I'll use vector notation $(\mathbf{a}, \mathbf{w}_k)$ rather than encoded decimal (a, w_k) to underscore the otherwise subtle concepts.]

We know that $(\mathbb{Z}_2)^n$ is not your grandparents' vector space; it has a dot-product that is not *positive definite* (e.g., $\mathbf{a} \cdot \mathbf{a}$ might = 0), which leads to screwy things. Nothing that we can't handle, but we have to be careful.

We'll be constructing a basis for $(\mathbb{Z}_2)^n$ in which the first $n-1$ vectors, $\mathbf{w}_0, \dots, \mathbf{w}_{n-2}$ are orthogonal to \mathbf{a} , while the n th vector, \mathbf{w}_{n-1} , is not. In general, this basis will not be – and cannot be – an orthonormal basis. It is *sometimes*, but not usually. To see cases where it cannot be, consider a period \mathbf{a} that is orthogonal to itself. In such a case, since $\mathbf{a} \in a_\perp$ we might have $\mathbf{a} = \mathbf{w}_k$ for some $k < n-1$, say $\mathbf{a} = \mathbf{w}_0$. Therefore, we end up with

$$\begin{aligned} \mathbf{w}_0 \cdot \mathbf{w}_0 &= \mathbf{a} \cdot \mathbf{a} = 0 & \text{and} \\ \mathbf{w}_0 \cdot \mathbf{w}_{n-1} &= \mathbf{a} \cdot \mathbf{w}_{n-1} = 1, \end{aligned}$$

either condition contradicting orthonormality.

In particular, we cannot use the *dot-with-basis trick* to compute expansion coefficients, a trick that we saw required an orthonormal basis to work. So, in this regime, even when expanding a vector \mathbf{v} along the algorithm's resulting w -basis,

$$\mathbf{v} = \sum_{k=0}^{n-1} c_k \mathbf{w}_k,$$

we would (sadly) discover that, for many of the coefficients, c_k ,

$$c_k \neq \mathbf{v} \cdot \mathbf{w}_k .$$

This is not a big deal, since we will not *need* the trick, but it's a good mental exercise to acknowledge naturally occurring vector spaces that give rise to non-*positive definite* pairings and be careful not to use those pairing as if they possessed our usual properties.

Example #1

Take $n = 4$ and

$$\mathbf{a} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} .$$

We may end up with a basis that uses the 3-dimensional subspace, a_{\perp} , generated by the three vectors

$$\left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \mathbf{a} \right\} ,$$

augmented by the n th vector – *not* orthogonal to \mathbf{a} ,

$$\mathbf{w}_{n-1} = \mathbf{w}_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} .$$

These four vectors form a possible outcome of our algorithm when applied to $(\mathbb{Z}_2)^4$ with a period of $\mathbf{a} = (0, 1, 0, 1)^t$, and you can confirm the odd claims I made above.

Example #2

If you'd like more practice with this, try the self-orthogonal

$$\mathbf{a} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

and the basis consisting of the three vectors orthogonal to \mathbf{a} ,

$$\left\{ \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \mathbf{a} \right\}$$

plus a fourth basis vector not orthogonal (to \mathbf{a})

$$\mathbf{w}_{n-1} = \mathbf{w}_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

This isn't an orthonormal basis, nor will the *dot-with-basis trick* work.

[**Exercise.** Create some of your own examples in which you do, and do not, get an orthonormal basis that results from the algorithm's desired outcome of $n - 1$ basis vectors being orthogonal to \mathbf{a} and the n th, not.]

18.10 Time Complexity of the Quantum Algorithm

18.10.1 Producing $n - 1$ Linearly-Independent \mathbf{w}_k in Polynomial Time – Argument 1

This section contains our officially sanctioned proof that one will get the desired $n - 1$ vectors, orthogonal to a , with polynomial complexity in n . It is a very straightforward argument that considers sampling the circuit $n + T$ times. While it has several steps, each one is comprehensible and contains the kind of arithmetic every quantum computer scientist should be able to reproduce.

Theorem. *If we randomly select $m + T$ samples from \mathbb{Z}_2^m , the probability that these samples will contain a linearly-independent subset of m vectors is $> 1 - (1/2)^T$.*

Notice that the constant T is independent of m , so the process of selecting the m independent vectors is $O(m + T) = O(m)$, *not counting any sub-algorithms or arithmetic we have to apply in the process* (which we'll get to). We'll be using $m = n - 1$, the dimension of a_\perp ($\cong \mathbb{Z}_{2^{n-1}}$).

18.10.2 Proof of Theorem Used by Argument 1

[**Notational Alert.** In the proof, I'll use vector notation, as in $\mathbf{z} \in (\mathbb{Z}_2)^m$ or $\mathbf{c} \in (\mathbb{Z}_2)^{m+T}$, which presents vectors in boldface.]

Pick $m + T$ vectors, $\{\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_{m+T-1}\}$, at random from $(\mathbb{Z}_2)^m$.

- **Step 1: Form a matrix and look at the column vectors.**

We begin by stacking the $m + T$ vectors atop one another to form the matrix

$$\begin{pmatrix} \mathbf{z}_0 \\ \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_{m+T-1} \end{pmatrix} = \begin{pmatrix} z_{00} & z_{01} & \cdots & z_{0(m-1)} \\ z_{10} & z_{11} & \cdots & z_{1(m-1)} \\ z_{20} & z_{21} & \cdots & z_{2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ z_{(m+T-1)0} & z_{(m+T-1)1} & \cdots & z_{(m+T-1)(m-1)} \end{pmatrix}.$$

The number of independent rows, each considered as a vector in $(\mathbb{Z}_2)^m$, is known as the *row rank* of the matrix. Likewise, *column rank* is a term used to describe the number of linearly independent columns, each viewed as a vector in the (higher dimensional) $(\mathbb{Z}_2)^{m+T}$. By elementary linear algebra, *row rank* = *column rank*. (The proof of this fact covers all matrices and vector spaces, even those based on unusual fields like \mathbb{Z}_2 .) So, let's change our perspective and think of this matrix as set of m *column* vectors, each of dimension $m + T$. Since the row vectors are linearly independent exactly when the column vectors are, if we can demonstrate that all m of the randomly selected *column* vectors

$$\begin{pmatrix} z_{00} \\ z_{10} \\ z_{20} \\ \vdots \\ z_{(m+T-1)0} \end{pmatrix}, \begin{pmatrix} z_{01} \\ z_{11} \\ z_{21} \\ \vdots \\ z_{(m+T-1)1} \end{pmatrix}, \dots, \begin{pmatrix} z_{0(m-1)} \\ z_{1(m-1)} \\ z_{2(m-1)} \\ \vdots \\ z_{(m+T-1)(m-1)} \end{pmatrix}$$

$$\begin{matrix} \uparrow & \uparrow & & \uparrow \\ \equiv & \mathbf{c}_0, & \mathbf{c}_1, & \cdots, & \mathbf{c}_{m-1} \end{matrix}$$

have a probability $> 1 - (1/2)^{T+1}$ of being *linearly independent*, then the original row vectors will also be independent with that same probability, and our proof will be settled.

[This *row rank* = *column rank* strategy has other applications in quantum computing and will be used when we study Neumark's construction for "orthogonalizing" a set of general measurements in the next course. Neumark's construction is a conceptual first step towards noisy-system analysis.]

- **Step 2: Express the probability that all m column vectors, \mathbf{c}_k , are independent as a product of m conditional probabilities.**

[**Note:** This is why we switched to column rank. The row vectors, we *know*, are not linearly independent after taking $T + m > m$ samples, but we can sample many more than m samples and continue to look at the increasingly longer column vectors, eventually making those linearly independent.]

Let

$$\mathcal{I}(j) \equiv \text{event that } \mathbf{c}_0, \mathbf{c}_1, \dots, \mathbf{c}_{j-1} \text{ are linearly-independent.}$$

Our goal is to compute the probability of $\mathcal{J}(m)$. Combining the basic identity,

$$\begin{aligned} P(\mathcal{J}(m)) &= P(\mathcal{J}(m) \wedge \mathcal{J}(m-1)) \\ &\quad + P(\mathcal{J}(m) \wedge \neg \mathcal{J}(m-1)), \end{aligned}$$

with the observation that

$$P(\mathcal{J}(m) \wedge \neg \mathcal{J}(m-1)) = 0$$

we can write

$$\begin{aligned} P(\mathcal{J}(m)) &= P(\mathcal{J}(m) \wedge \mathcal{J}(m-1)) \\ &= P(\mathcal{J}(m) \mid \mathcal{J}(m-1)) P(\mathcal{J}(m-1)), \end{aligned}$$

the last equality from Bayes' rule.

Now, apply that same process to the right-most factor, successively, and you end up with

$$\begin{aligned} P(\mathcal{J}(m)) &= P(\mathcal{J}(m) \mid \mathcal{J}(m-1)) P(\mathcal{J}(m-1) \mid \mathcal{J}(m-2)) \\ &\quad \dots P(\mathcal{J}(2) \mid \mathcal{J}(1)) P(\mathcal{J}(1)) \\ &= \prod_{j=1}^m P(\mathcal{J}(j) \mid \mathcal{J}(j-1)). \end{aligned}$$

(The $j = 1$ factor might look a little strange, because it refers to the undefined $\mathcal{J}(0)$, but it makes sense if we view the event $\mathcal{J}(0)$, i.e., that in a set of no vectors they're all linearly-independent, as vacuously *true*. Therefore, we see that $\mathcal{J}(0)$ can be said to have probability 1, so the $j = 1$ term reduces to $P(\mathcal{J}(1))$, without a conditional, in agreement with the line above.)

- **Step 3: Compute the j th factor, $P(\mathcal{J}(j) \mid \mathcal{J}(j-1))$, in this product.**

$$P(\mathcal{J}(j) \mid \mathcal{J}(j-1)) = 1 - P(\neg(\mathcal{J}(j) \mid \mathcal{J}(j-1)))$$

But what does “ $\neg(\mathcal{J}(j) \mid \mathcal{J}(j-1))$ ” mean? It means

1. we are assuming that the first $j-1$ vectors, $\{\mathbf{c}_0, \dots, \mathbf{c}_{j-2}\}$, are independent and therefore span the largest space possible for $j-1$ vectors, namely, one of size 2^{j-1} , and
2. the j th vector, \mathbf{c}_{j-1} , is \in the span of the first $j-1$ vectors $\{\mathbf{c}_0, \dots, \mathbf{c}_{j-2}\}$.

This probability is computed by counting the number of ways we can select a vector from the entire space of 2^{m+T} vectors (remember, our column vectors

have $m + T$ coordinates) that also happens to be in the span of the first $j - 1$ vectors (a subspace of size 2^{j-1}).

[Why size 2^{j-1} ? Express the first $j - 1$ column vectors in *their own* coordinates, i.e., in a basis that starts with them, then adds more basis vectors to get the full basis for $(\mathbb{Z}_2)^{m+T}$. Recall that basis vectors expanded *along themselves* always have a single 1 coordinate sitting in a column of 0s. Looked at this way, how many distinct vectors can be formed out of various sums of the original $j - 1$?

The probability we seek is, by definition of *probability*, just the ratio

$$\frac{2^{j-1}}{2^{m+T}} = \left(\frac{1}{2}\right)^{m+T-j+1},$$

so

$$P(\mathcal{J}(j) \mid \mathcal{J}(j-1)) = 1 - \left(\frac{1}{2}\right)^{m+T-j+1},$$

for $j = 1, 2, \dots, m$.

Sanity Check. Now is a good time for the computer scientist's first line of defense when coming across a messy formula: the *sanity check*.

Does this make sense for $j = 1$, the case of a single vector \mathbf{c}_0 ? The formula tells us that the chances of getting a single, linearly-independent vector is

$$P(\mathcal{J}(1)) = 1 - \left(\frac{1}{2}\right)^{m+T-1+1} = 1 - \left(\frac{1}{2}\right)^{m+T}.$$

Wait, shouldn't the first vector be 100% certain? No, we might get unlucky and pick the $\mathbf{0}$ -vector with probability $1/2^{m+T}$, which is exactly what the formula predicts.

That was too easy, so let's do one more. What about $j = 2$? The first vector, \mathbf{c}_0 , spans a space of two vectors (as do all non-zero single vectors in $(\mathbb{Z}_2)^m$). The chances of picking a second vector from this set would be $2/(\text{size of the space})$, because $2 = \text{the size of the event "pick either } \mathbf{0} \text{ or } \mathbf{c}_0\text{"}$. The size of the space is $1/2^{m+T}$, so that fraction is $2/2^{m+T} = 1/2^{m+T-1}$. Meanwhile, the formula we're testing predicts that we'll get a second *independent* vector, *not* in the span of \mathbf{c}_0 with probability

$$P(\mathcal{J}(2) \mid \mathcal{J}(1)) = 1 - \left(\frac{1}{2}\right)^{m+T-2+1} = 1 - \left(\frac{1}{2}\right)^{m+T-1},$$

exactly the complement of the probability that \mathbf{c}_1 just happened to get pulled from the set of two vectors spanned by \mathbf{c}_0 , as computed.

So, spot testing supports the correctness of the derived formula.

- **Step 4: Plug the expression for the j th factor (step 3) back into the full probability formula for all m vectors (step 2).**

In step 2, we decomposed the probability of “success”, as a product,

$$P(\mathcal{J}(m)) = \prod_{j=1}^m P(\mathcal{J}(j) \mid \mathcal{J}(j-1)).$$

In step 3, we computed the value for each term,

$$P(\mathcal{J}(j) \mid \mathcal{J}(j-1)) = 1 - \left(\frac{1}{2}\right)^{m+T-j+1}.$$

Combining the two, we get

$$P(\mathcal{J}(m)) = \prod_{j=1}^m \left(1 - \left(\frac{1}{2}\right)^{m+T-j+1}\right),$$

which can easily be re-indexed into the form

$$P(\mathcal{J}(m)) = \prod_{i=1}^m \left(1 - \left(\frac{1}{2}\right)^{T+i}\right).$$

[**Exercise.** Prove the last assertion about re-indexing.]

- **Step 5: We prove a cute mathematical lemma.** Let’s take a pleasant off-trail walk to prove a needed fact.

Lemma.

If $0 \leq \alpha_i \leq 1$

$$\text{then } \prod_{i=1}^p (1 - \alpha_i) \geq 1 - \sum_{i=1}^p \alpha_i.$$

Proof by Induction.

– Case $p = 1$:

$$1 - \alpha_1 \geq 1 - \alpha_1. \quad \checkmark$$

– Consider any $p > 1$ and assume the claim is true for $p - 1$. Then

$$\begin{aligned} \prod_{i=1}^p (1 - \alpha_i) &= (1 - \alpha_p) \prod_{i=1}^{p-1} (1 - \alpha_i) \\ &\geq (1 - \alpha_p) \left(1 - \sum_{i=1}^{p-1} \alpha_i\right) \\ &= 1 - \sum_{i=1}^{p-1} \alpha_i - \alpha_p + \sum_{i=1}^{p-1} \alpha_p \alpha_i \\ &> 1 - \sum_{i=1}^p \alpha_i. \quad \checkmark \quad QED \end{aligned}$$

[**Exercise.** Where did we use the hypothesis $\alpha_i \geq 0$? Same question, $\alpha_i \leq 1$.]

[**Exercise.** Is the stronger hypothesis $\sum \alpha_i > 0$ needed for this lemma? Why or why not?

- **Step 6: Apply the lemma to the conclusion of step 4 to finish off the proof.** Using m for the p of the lemma, we obtain

$$\begin{aligned} P(\mathcal{J}(m)) &= \prod_{i=1}^m \left(1 - \left(\frac{1}{2} \right)^{T+i} \right) \\ &\geq 1 - \sum_{i=1}^m \left(\frac{1}{2} \right)^{T+i} \\ &= 1 - \left(\frac{1}{2} \right)^T \left[\sum_{i=1}^m \left(\frac{1}{2} \right)^i \right]. \end{aligned}$$

But that big bracketed sum on the RHS is a bunch of distinct and positive powers of $1/2$, which can never add up to more than 1 (think binary floating point numbers like .101011 or .00111 or .111111), so that sum is < 1 , i.e.,

$$\begin{aligned} \left[\sum_{i=1}^m \left(\frac{1}{2} \right)^i \right] &< 1, \quad \text{so} \\ \left(\frac{1}{2} \right)^T \left[\sum_{i=1}^m \left(\frac{1}{2} \right)^i \right] &< \left(\frac{1}{2} \right)^T, \quad \text{and we get} \\ 1 - \left(\frac{1}{2} \right)^T \left[\sum_{i=1}^m \left(\frac{1}{2} \right)^i \right] &> 1 - \left(\frac{1}{2} \right)^T. \end{aligned}$$

Combining the results of the last two equation blocks we conclude

$$P(\mathcal{J}(m)) > 1 - \left(\frac{1}{2} \right)^T.$$

This proves that the column vectors, \mathbf{c}_j , are linearly-independent with probability *greater than* $1 - 1/2^T$, and therefore the row vectors, our \mathbf{z}_k also have at least m linearly independent vectors among them (*row rank = column rank*, remember?), with that same lower-bound probability. QED

18.10.3 Summary of Argument 1

We have demonstrated that by sampling $m + T$ vectors in a \mathbb{Z}_2 space of dimension m , we can be confident that the sample set contains m linearly independent vectors with probability $1 - (1/2)^T$, T independent of m .

In Simon's problem, we were looking for $n - 1$ vectors that spanned the $n - 1$ -dimensional a_\perp . By applying the above result to $m = n - 1$, we see that we can find those $n - 1$ vectors with arbitrarily high probability by running our circuit $O(T + n - 1) = O(n)$ times.

18.10.4 Producing $n - 1$ Linearly-Independent \mathbf{w}_k in Polynomial Time – Argument 2

This argument is seen frequently and is more straightforward than our preferred one, but it gives a weaker result in absolute terms. That is, it gives an unjustifiably conservative projection for the number of samples required to achieve $n - 1$ linearly independent vectors. Of course, this would not affect the performance of an actual algorithm, since all we are doing in these proofs is showing that we'll get linear independence fast. An actual quantum circuit would be indifferent to how quickly *we think* it should give us $n - 1$ independent vectors; it would reveal them in a time frame set by the laws of nature, not what we proved or didn't prove. Still, it's nice to predict the convergence to linear independence accurately, which this version doesn't do quite as well as the first. Due to its simplicity and prevalence in the literature, I include it.

Here are the steps. We'll refer back to the first proof when we need a result that was already proven there.

Theorem. *If we randomly select m samples from \mathbb{Z}_{2^m} , the probability that we have selected a complete, linearly-independent (and therefore basis) set is $> 1/4$.*

This result (once proved) will imply that the probability of getting a linearly independent set of m vectors after picking m vectors from \mathbb{Z}_{2^m} repeatedly T times is at least $1 - 1/4^T$. The number of individual times we have to sample the circuit is, then, mT . The reason we have to take the product, mT , is that the theorem only computes the probability that results when we take *exactly* m samples. We have to throw every failing set of m out and start again with another set of m samples. The theorem does not address the trickier math for overlapping sample sets or a slowly changing sample set that would come from adding one new sample and throwing away an old one. Nevertheless, this less “environmental” strategy gives an $O(m)$ time complexity.

Keep in mind that we'll be applying this theorem to $m = n - 1$, the dimension of a_\perp ($\cong \mathbb{Z}_{2^{n-1}}$).

18.10.5 Proof of Theorem Used by Argument 2

[**Notation.** As with the first proof, we'll use boldface vector notation, $\mathbf{z} \in (\mathbb{Z}_2)^m$.]

Pick m vectors, $\{\mathbf{z}_0, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_{m-1}\}$, at random from $(\mathbb{Z}_2)^m$.

- **Step 1: Express the probability that the m vectors, \mathbf{z}_k , are independent as a product of m conditional probabilities.**

Let

$$\mathcal{I}(j) \equiv \text{event that } \mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_{j-1} \text{ are linearly-independent.}$$

Our goal is to compute the probability of $\mathcal{J}(m)$.

Using the exact argument from **argument 1, step 2**, we conclude,

$$P(\mathcal{J}(m)) = \prod_{j=1}^m P(\mathcal{J}(j) \mid \mathcal{J}(j-1)).$$

(If interested, see **argument 1, step 2** to account for the fact that $\mathcal{J}(0)$ can be said to have probability 1, implying that the $j = 1$ factor reduces to $P(\mathcal{J}(1))$, without a conditional.)

- **Step 2: Compute the j th factor, $P(\mathcal{J}(j) \mid \mathcal{J}(j-1))$, in this product.**

$$P(\mathcal{J}(j) \mid \mathcal{J}(j-1)) = 1 - P(\neg(\mathcal{J}(j) \mid \mathcal{J}(j-1)))$$

But what does “ $\neg(\mathcal{J}(j) \mid \mathcal{J}(j-1))$ ” mean? It means

1. we are assuming that the first $j-1$ vectors, $\{\mathbf{z}_0, \dots, \mathbf{z}_{j-2}\}$, are independent and therefore span the largest space possible for $j-1$ vectors, namely, one of size 2^{j-1} , and
2. the j th vector, \mathbf{z}_{j-1} , is \in the span of the first $j-1$ vectors $\{\mathbf{z}_0, \dots, \mathbf{z}_{j-2}\}$.

This probability is computed by counting the number of ways we can select a vector from the entire space of 2^m vectors which happens to also be in the span of the first $j-1$ vectors, a subspace of size 2^{j-1} . But that's just the ratio

$$\frac{2^{j-1}}{2^m} = \left(\frac{1}{2}\right)^{m-j+1},$$

so

$$P(\mathcal{J}(j) \mid \mathcal{J}(j-1)) = 1 - \left(\frac{1}{2}\right)^{m-j+1},$$

for $j = 1, 2, \dots, m$.

Sanity Check. Does this make sense for $j = 1$, the case of a single vector \mathbf{z}_0 ? The formula tells us that the chances of getting a single, linearly-independent vector is

$$P(\mathcal{J}(1)) = 1 - \left(\frac{1}{2}\right)^{m-1+1} = 1 - \left(\frac{1}{2}\right)^m.$$

Wait, shouldn't the first vector be 100% certain? No, we might get unlucky and pick the $\mathbf{0}$ -vector with probability $1/2^m$, which is exactly what the formula predicts.

That was too easy, so let's do one more. What about $j = 2$? The first vector, \mathbf{z}_0 , spans a space of two vectors (as do all non-zero single vectors in $(\mathbb{Z}_2)^m$).

The chances of picking a second vector from this set would be $2/(\text{size of the space})$, which is $2/2^m = 1/(2^{m-1})$. The formula predicts that we will get a second *independent* vector, *not* in the span of \mathbf{z}_0 with probability

$$P(\mathcal{J}(2) \mid \mathcal{J}(1)) = 1 - \left(\frac{1}{2}\right)^{m-2+1} = 1 - \left(\frac{1}{2}\right)^{m-1},$$

exactly the complement of the probability that \mathbf{z}_1 just happening to get pulled from the set of two vectors spanned by \mathbf{z}_0 , as computed.

So, spot testing supports the formula's claim.

- **Step 3: Plug the expression for the j th factor (step 2) back into the full probability formula for all m vectors (step 1).**

In step 1, we decomposed the probability of “success”, as a product,

$$P(\mathcal{J}(m)) = \prod_{j=1}^m P(\mathcal{J}(j) \mid \mathcal{J}(j-1)).$$

In step 2, we computed the value for each term,

$$P(\mathcal{J}(j) \mid \mathcal{J}(j-1)) = 1 - \left(\frac{1}{2}\right)^{m-j+1}.$$

Combining the two, we get

$$P(\mathcal{J}(m)) = \prod_{j=1}^m \left(1 - \left(\frac{1}{2}\right)^{m-j+1}\right),$$

which can easily be re-indexed into the form

$$P(\mathcal{J}(m)) = \prod_{i=1}^m \left(1 - \left(\frac{1}{2}\right)^i\right).$$

[**Exercise.** Prove the last assertion about re-indexing.]

- **Step 4: Apply a result from *q-Series* to finish off the proof.**

The expression we have can be multiplied by values < 1 if we are interested in a lower bound (which we are). Therefore, we can include all the factors for $i > m$ without harm:

$$\begin{aligned} P(\mathcal{J}(m)) &= \prod_{i=1}^m \left(1 - \left(\frac{1}{2}\right)^i\right) \\ &\geq \prod_{i=1}^{\infty} \left(1 - \left(\frac{1}{2}\right)^i\right) \end{aligned}$$

From here one could just quote a result from the theory of *mathematical q-series*, namely that this infinite product is about .28879. As an alternative, there are some elementary proofs that involve taking the natural log of the product, splitting it into a finite sum plus and infinite error sum, then estimating the error. We'll accept the result without further ado, which implies

$$P(\mathcal{J}(m)) = \prod_{i=1}^m \left(1 - \left(\frac{1}{2}\right)^i\right) > .25$$

This proves that the m vectors, \mathbf{z}_j , are linearly-independent with probability $> 1/4$.

18.10.6 Summary of Argument 2

We have demonstrated that a random sample of exactly m vectors in a \mathbb{Z}_2 space of dimension m will be linearly independent with probability at least $1/4$. QED

Corollary. After T full cycles, each cycle pulling m random samples from \mathbb{Z}_{2^m} (not reusing vectors from previous cycles), the probability that none of the T sets of m -vectors is linearly independent is $< (3/4)^T$.

Proof. This follows immediately from the theorem since the chances we fail to select one set of m independent vectors $= 1 - \text{chance of success} < 3/4$, as the theorem showed. Each cycle is independent so the probability that all of them fail is the product of the probabilities that each one fails, and therefore $< (3/4)^T$. QED

In Simon's problem, we were looking for $n - 1$ vectors that spanned the $n - 1$ -dimensional a_\perp . By applying the above result to $m = n - 1$, we see that we can find those $n - 1$ vectors with arbitrarily high probability by running our circuit $O(T(n - 1)) = O(n)$ times.

18.10.7 Discussion of the Two Proofs' Complexity Estimates

Both proofs give the correct $O(n)$ complexity (of the algorithm's *quantum* processing) for finding $n - 1$ independent vectors spanning a_\perp in the context of \mathbb{Z}_{2^n} .

The second proof is appealing in its simplicity, but part of that simplicity is due to its handing off a key result to the number theorists. (I have found no sources that give all the details of the $> 1/4$ step). Also, the number of circuit samples argument 2 requires for a given level of confidence is, while still $O(n)$, many times larger than one really needs, and, in particular, many more than the first proof. This is because it does not provide probabilities for overlapping sample sets, but rather tosses out all $n - 1$ samples of any set that fails. One can adjust the argument to account for this conditional dependence, but that's a different argument; if we know that one set is not linearly independent, then reusing its vectors requires trickier math than

this proof covers. Of course, this is mainly because it is only meant to be a proof of polynomial time complexity, and not a blueprint for implementation.

For example, for $n = 10$, $T = 10$, the second proof predicts that $10 \times 9 = 90$ samples would produce at least one of the 10 sets to be linearly independent with probability greater than $1 - (3/4)^{10} \approx .943686$. In contrast, the first proof would only ask for $9 + 10 = 19$ samples to get confidence $> .999023$. That's greater confidence with fewer samples.

18.11 The Hidden Classical Algorithms and Their Cost

18.11.1 Unaccounted for Steps

As far as we have come, we can't claim victory yet, especially after having set the rather high bar of proving, not just uttering, the key facts that lead to our end result.

We have a quantum circuit with $O(n)$ gates which we activate $O(n)$ times to find the unknown period, a , with arbitrarily high probability. We've agreed to consider the time needed to operate that portion of the quantum algorithm and ignore the circuit's linear growth. Therefore, we have thus far accounted for a *while loop* which requires $O(n)$ passes to get a .

However, there are steps in the sampling process where we have implicitly used some non-trivial classical algorithms that our classical computers must execute. We need to see where they fit into the big picture and incorporate their costs. There are two general areas:

1. The test for *mod-2 linear independence* in $(\mathbb{Z}_2)^n$ that our iterative process has used throughout.
2. The cost of solving the system of n mod-2 equations,

$$w_k \cdot a = \begin{cases} 0, & k = 0, \dots, n-2 \\ 1, & k = n-1 \end{cases}.$$

For those of you who will be skipping the details in the next sections, I'll reveal the results now:

1. The test for mod-2 linear independence is handled using *mod-2 Gaussian elimination* which we will show to be $O(n^3)$.
2. Solving the system of n mod-2 equations is handled using *back substitution* which we will show to be $O(n^2)$.

3. The two classical algorithms will be applied in *series*, so we only need to count the larger of the two, $O(n^3)$. Together, they are applied once for each quantum sample, already computed to be $O(n)$, resulting in a nested count of $O(n^4)$.
4. We'll tweak the classical tools by integrating them into Simon's algorithm so that their combined cost is only $O(n^2)$, resulting in a final count of $O(n^3)$.

Conclusion. Our implementation of Simon's algorithm has a growth rate of $O(n^3)$. It is polynomial fast.

18.12 Solving Systems of Mod-2 Equations

18.12.1 Gaussian Elimination and Back Substitution

Conveniently, both remaining classical tasks are addressed by an age old and well documented technique in linear algebra for solving systems of linear equations called "Gaussian elimination with back substitution." It's a mouthful but easy to learn and, as with all the ancillary math we have been forced to cover, applicable throughout engineering. In short, a system of linear equations

$$\begin{array}{rrcrcl} 5x & + & y & + & 2z & + & w & = & 7 \\ x & - & y & + & z & + & 2w & = & 10 \\ x & + & 2y & - & 3z & + & 7w & = & -3 \end{array}$$

is symbolized by the matrix of its constant coefficients, as in

$$\begin{pmatrix} 5 & 1 & 2 & 1 \\ 1 & -1 & 1 & 2 \\ 1 & 2 & -3 & 7 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} 7 \\ 10 \\ -3 \end{pmatrix}.$$

As the example shows, there's no requirement that the system have the same number of equations as unknowns; the fewer equations, the less you will know about the solutions. (Instead of the solution being a unique vector like $(x, y, z, w)^t = (3, 0, -7, 2)^t$, it might be a *relation* between the components, like $(\alpha, 4\alpha, -.5\alpha, 3\alpha)^t$, with α free to roam over \mathbb{R}). Nevertheless, we can apply our techniques to any sized system.

We break it into the two parts,

- *Gaussian elimination*, which produces a matrix with 0s in the lower left triangle, and
- *back substitution*, which uses that matrix to solve the system of equations as best we can, meaning that if there are not enough equations to get a unique solution, we will only get relations between unknowns, not literal numbers.

18.12.2 Gaussian Elimination

Gaussian Elimination for Decimal Matrices

Gaussian Elimination seeks to change the matrix for the equation into (depending on who you read), either

- **row echelon form**, in which everything below the “diagonal” is 0, as in

$$\begin{pmatrix} 3 & 3 & 0 & 5 \\ 0 & 6 & 2 & 4 \\ 0 & 0 & 5 & -7 \end{pmatrix}, \text{ or}$$

- **reduced row echelon form**, which is echelon form with the *additional* requirement that the first non-zero element in each row be 1, e.g.,

$$\begin{pmatrix} 1 & 1 & 0 & 5/3 \\ 0 & 1 & 1/3 & 2/3 \\ 0 & 0 & 1 & -7/5 \end{pmatrix}.$$

In our case, where all the values are integers mod-2 (just 0 and 1), the two are actually equivalent: all non-zero values are 1, automatically.

Properties of Echelon Forms

Reduced or not, row echelon forms have some important properties that we will need. Let’s first list them, then have a peek at how one uses Gaussian elimination (GE), to convert any matrix to an echelon form.

- Geometrically, the diagonal, under which all the elements must be 0, is clear in a square matrix:

$$\begin{pmatrix} \bullet & * & * & \cdots & * \\ 0 & \bullet & * & \cdots & * \\ 0 & 0 & \bullet & \cdots & * \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \bullet \end{pmatrix}$$

When the the matrix is not square, the diagonal is geometrically visualized

relative to the upper left (position $(0, 0)$):

$$\begin{pmatrix} \bullet & * & * & \cdots & * \\ 0 & \bullet & * & \cdots & * \\ 0 & 0 & \bullet & \cdots & * \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \bullet \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} \bullet & * & * & \cdots & * & \cdots & * \\ 0 & \bullet & * & \cdots & * & \cdots & * \\ 0 & 0 & \bullet & \cdots & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & * \\ 0 & 0 & \cdots & 0 & \bullet & \cdots & * \end{pmatrix}$$

In any case, a diagonal element is one that sits on position (k, k) , for some k .

- The first non-zero element on row k is to the right of the first non-zero element of row $k-1$, but it might be *two or more positions* to the right. I'll use a reduced form which has the special value, 1, occupying the first non-zero element in a row to demonstrate this. Note the extra 0s, underlined, that come about as a result of some row being “pushed to the right” in this way.

$$\begin{pmatrix} 1 & * & * & * & * & * & \cdots & * \\ 0 & 1 & * & * & * & * & \cdots & * \\ 0 & 0 & \underline{0} & \underline{0} & 1 & * & \cdots & * \\ 0 & 0 & 0 & 0 & 0 & 1 & \cdots & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & [0/1/*] \end{pmatrix}$$

- Any all-zero rows necessarily appear at the bottom of the matrix.
[**Exercise.** Show this follows from the definition.]
- All non-zero row vectors in the matrix are, collectively, a linear independent set.
[**Exercise.** Prove it.]
- If we know that there are no all-zero row vectors in the echelon form, then the number of rows \leq number of columns.
[**Exercise.** Prove it.]

Including the RHS Constant Vector in the Gaussian Elimination Process

When using GE to solve systems of equations, we have to be careful that the set of equations that the reduced echelon form represents is equivalent to the original set of equations, and to that end we have to modify the *RHS column vector*, e.g., $(7, 10, -3)^t$ of our example, as we act on the matrix on the LHS. We thus start the

festivities by placing the RHS constant vector in the same “house” as the LHS matrix, but in a “room” of its own,

$$\left(\begin{array}{cccc|c} 5 & 1 & 2 & 1 & 7 \\ 1 & -1 & 1 & 2 & 10 \\ 1 & 2 & -3 & 7 & -3 \end{array} \right).$$

We will modify both the matrix and the vector at the same time, eventually resulting in the *row-echelon form*,

$$\left(\begin{array}{cccc|c} 3 & 3 & 0 & 5 & -13 \\ 0 & 6 & 2 & 4 & -\frac{832}{15} \\ 0 & 0 & 5 & -7 & -\frac{17}{5} \end{array} \right),$$

or, if we went further, in the *reduced row echelon form*

$$\left(\begin{array}{cccc|c} 1 & 1 & 0 & \frac{5}{3} & -\frac{13}{3} \\ 0 & 1 & \frac{1}{3} & \frac{2}{3} & -\frac{832}{45} \\ 0 & 0 & 1 & -\frac{7}{5} & -\frac{17}{25} \end{array} \right).$$

The Three Operations that Produce Echelon Forms

There are only three legal operations that we need to consider when performing GE,

1. swapping two rows,
2. multiplying a row by a nonzero value, and
3. adding a multiple of one row to another.

[**Exercise.** Prove that these operations produce equations that have the identical solution(s) as the original equations.]

For example,

$$\begin{aligned}
 & \left(\begin{array}{cccc|c} 5 & 1 & 2 & 1 & 7 \\ 1 & -1 & 1 & 2 & 10 \\ 1 & 2 & -3 & 7 & -3 \end{array} \right) \xrightarrow{-5 \times 2\text{nd row}} \left(\begin{array}{cccc|c} 5 & 1 & 2 & 1 & 7 \\ -5 & 5 & -5 & -10 & -50 \\ 1 & 2 & -3 & 7 & -3 \end{array} \right) \\
 & \xrightarrow{\text{add 1st to 2nd}} \left(\begin{array}{cccc|c} 5 & 1 & 2 & 1 & 7 \\ 0 & 6 & -3 & -9 & -43 \\ 1 & 2 & -3 & 7 & -3 \end{array} \right) \\
 & \xrightarrow{\text{swap 1st and 3rd}} \left(\begin{array}{cccc|c} 1 & 2 & -3 & 7 & -3 \\ 0 & 6 & -3 & -9 & -43 \\ 5 & 1 & 2 & 1 & 7 \end{array} \right) \\
 & \xrightarrow{\text{add } -5 \times 1\text{st to 3rd}} \left(\begin{array}{cccc|c} 1 & 2 & -3 & 7 & -3 \\ 0 & 6 & -3 & -9 & -43 \\ 0 & -9 & 17 & -34 & 22 \end{array} \right) \\
 & \xrightarrow{\text{add } \frac{3}{2} \times 2\text{nd to 3rd}} \left(\begin{array}{cccc|c} 1 & 2 & -3 & 7 & -3 \\ 0 & 6 & -3 & -9 & -43 \\ 0 & 0 & \frac{25}{2} & -\frac{95}{2} & -\frac{85}{2} \end{array} \right),
 \end{aligned}$$

etc. (These particular operations may not lead to the echelon forms, above; they're just illustrations of the three rules.)

The Cost of Decimal-Based Gaussian Elimination

GE is firmly established in the literature, so for those among you who are interested, I'll prescribe web search to dig up the exact sequence of operations needed to produce a row reduced echelon form. The simplest algorithms with no short-cuts use $O(n^3)$ operations, where an "operation" is either addition or multiplication, and n is the larger of the matrix's two dimensions. Some special techniques can improve that, but it is always worse than $O(n^2)$, so we'll be satisfied with the simpler $O(n^3)$.

To that result we must incorporate the cost of each *multiplication* and *addition* operation. For GE, *multiplication* could involve increasingly large numbers and, if incorporated into the full accounting, would change the complexity to slightly better than $O(n^3 (\log m)^2)$, where m is the absolute value of the largest integer involved. Addition is less costly and done in series with the multiplications so does not erode performance further.

The Cost of Mod-2 Gaussian Elimination

For mod-2 arithmetic, however, we can express the complexity without the extra variable, m . Our matrices consist of only 0s and 1s, so the "multiplication" in *operations 2* and *3* reduce to either the identity ($1 \times$ a row) or producing a row of 0s ($0 \times$ a row). Therefore, we ignore the multiplicative cost completely. Likewise, the addition

operation that is counted in the general GE algorithm is between matrix elements, each of which might be arbitrarily large. But for us, all matrix elements are either 0 or 1, so our addition is a constant time XOR between bits.

All this means that in the mod-2 milieu, each of our $\approx n^3$ GE operations requires some constant-time XORs (for the additions) and constant time *if*-statements (to account for the multiplication by 0 or 1). Evidently, the a total mod-2 GE cost is untarnished at $O(n^3)$.

With some fancy footwork in a special section, below, we'll improve it to $O(n^2)$, but we don't really care about the exact figure once we have it down this far. All we require is *polynomial time* to make Simon a success story. However, we are exercising our ability to evaluate algorithms, historical or future, so this continues to be an exercise worthy of our efforts.

18.12.3 Back Substitution

Back Substitution for Decimal Matrices

It is easiest – and for us, enough – to explain back substitution in the case when the number of linearly independent equations equals the number of unknowns, n . Let's say we begin with the system of equations in matrix form,

$$\begin{pmatrix} c_{00} & c_{01} & \cdots & c_{0(n-1)} \\ c_{10} & c_{11} & \cdots & c_{1(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ c_{(n-1)0} & c_{(n-1)1} & \cdots & c_{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix},$$

assumed to be of maximal rank, n – all rows (or columns) are linearly independent. This would result is a reduced row echelon form

$$\left(\begin{array}{cccccc|c} 1 & c'_{01} & c'_{02} & c'_{03} & \cdots & c'_{0(n-2)} & c'_{0(n-1)} & b'_0 \\ 0 & 1 & c'_{12} & c'_{13} & \cdots & c'_{1(n-2)} & c'_{1(n-1)} & b'_1 \\ 0 & 0 & 1 & c'_{23} & \cdots & c'_{2(n-2)} & c'_{2(n-1)} & b'_2 \\ 0 & 0 & & 1 & \cdots & c'_{3(n-2)} & c'_{3(n-1)} & b'_3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & c'_{(n-2)(n-1)} & b'_{n-2} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & b'_{n-1} \end{array} \right),$$

where, the c'_{kj} and b'_k are not the original constants in the equation, but the ones obtained after applying GE. In reduced echelon form, we see that the $(n-1)$ st unknown, x_{n-1} , can be read off immediately, as

$$x_{n-1} = b'_{n-1}.$$

From here, we “substitute back” into the $(n-2)$ nd equation to get,

$$x_{n-2} + b'_{n-1} c'_{(n-2)(n-1)} = b'_{n-2},$$

which can be solved for x_{n-2} (one equation, one unknown). Once solved, we substitute these numbers into the equation above, getting another equation with one unknown. This continues until all rows display the answer to its corresponding x_k , and the system is solved.

The Cost of Decimal-Based Back Substitution

The bottom row has no operations: it is already solved. The second-from-bottom has one multiplication and one addition. The third-from-bottom has two multiplications and two additions. Continuing in this manner and adding things up, we get

$$1 + 2 + 3 + \dots + (n - 1) = \frac{(n - 1)n}{2}$$

additions and the same number of multiplications, producing an overall complexity of $O(n^2)$ operations. As noted, the time complexity of the addition and multiplication algorithms would degrade by a factor of $(\log m)^2$, m being the largest number to be multiplied, making the overall “bit” complexity $O(n^2 (\log m)^2)$.

The Cost of Mod-2 Back Substitution

For mod-2 systems, we have no actual multiplication (the b'_k are either 1 and 0) and the additions are single-bit mod-2 additions and therefore, *constant time*. Thus, each of the approximately $n(n + 1)/2 (= O(n^2))$ addition operations in back substitution uses a constant time addition, leaving the total cost of back substitution un-degraded at $O(n^2)$.

18.12.4 The Total Cost of the Classical Techniques for Solving Mod-2 Systems

We have shown that Gaussian elimination and back substitution in the mod-2 environment have time complexities, $O(n^3)$ and $O(n^2)$, respectively. To solve a system of n mod-2 equation the two methods can be executed in series, so the dominant $O(n^3)$ will cover the entire expense.

However, this isn’t exactly how Simon’s algorithm uses these classical tools, so we need to count in a way that precisely fits our needs.

18.13 Applying GE and Back Substitution to Simon’s Problem

We now show how these time tested techniques can be used to evaluate the classical post processing costs in Simon’s algorithm. The eventual answer we will get is this:

They will be used in-series, so we only need to count the larger of the two, $O(n^3)$, and these algorithms are applied once for each quantum sample, already computed to be $O(n)$, resulting in a nested count of $O(n^4)$.

18.13.1 Linear Independence

Gaussian elimination – without back substitution – can be used to determine whether a vector z is linearly independent of a set $\mathcal{W} = \{w_k\}$ of m vectors which is, itself, known to be linearly independent. Therefore, we will apply GE after taking each new sample z , and if we find z to be independent of the existing \mathcal{W} , we'll add it to \mathcal{W} (increasing its size by one) and go on to get our next z . If z turns out to not be independent of \mathcal{W} , we'll throw it away and get another z without changing \mathcal{W} . This process continues until either \mathcal{W} contains the maximum $n - 1$ vectors (the most that can be orthogonal to the unknown period a) or we have exceeded the $n + T$ sampling limit.

The following process demonstrates one way to apply GE to determine linearly independence and build an increasingly larger set of independent vectors in \mathcal{W} .

1. Stack the w_k on top of one another to form an $m \times n$ matrix, which we assume to already be in *reduced echelon form* (our construction will guarantee it):

$$\begin{pmatrix} 1 & w_{01} & w_{02} & w_{03} & \dots & w_{0(n-3)} & w_{0(n-2)} & w_{0(n-1)} \\ 0 & 0 & 1 & w_{13} & \dots & w_{1(n-3)} & w_{1(n-2)} & w_{1(n-1)} \\ 0 & 0 & 0 & 1 & \dots & w_{2(n-3)} & w_{2(n-2)} & w_{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & w_{(m-1)(n-2)} & w_{(m-1)(n-1)} \end{pmatrix}.$$

2. **Observations.** Notice that $m < (n - 1)$, since the vectors in \mathcal{W} are independent, by assumption, and if m were equal to $n - 1$, we would already have a maximally independent set of vectors known to be orthogonal to a and would have stopped sampling the circuit. That means that there are at least two more columns than there are rows: the full space is n -dimensional, and we have $n - 2$ or fewer linearly independent vectors so far.

As a consequence, one or more rows (the second, in the above example) skips to the right more than one position relative to the row above it and/or the final row in the matrix has its leading 1 in column $n - 3$ or greater.

3. Put z at the bottom of the stack and re-apply GE.
 - **If z is linearly independent of \mathcal{W}** this will result in a new non-zero vector row. Replace the set \mathcal{W} with the new set whose coordinates are the rows of the new reduced-echelon matrix. These new row vectors are in the span of the old \mathcal{W} plus z added. [**Caution.** It is possible that none of the original w_k or z explicitly appear in the new rows, which come from

GE applied to those vectors. All that matters is that the span of the new rows is the same as the span of \mathcal{W} plus z , which GE ensures.] We have increased our set of linearly independent vectors by one.

- **If z is *not* linearly independent of \mathcal{W}** the last row will contain all 0s. Recover the original \mathcal{W} (or, if you like, replace it with the new reduced matrix row vectors, leaving off the final 0 row – the two sets will span the same space and be linearly independent). You are ready to grab another z based on the outer-loop inside which this linear-independence test resides.

[**Exercise.** Explain why all the claims in this step are true.]

4. Once $n - 1$ vectors populate the set $\mathcal{W} = \{w_0, \dots, w_{n-2}\}$, the process is complete. We call the associated row-reduced matrix W ,

$$W = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{n-2} \end{pmatrix},$$

and W satisfies the matrix-vector product equation

$$W \cdot \mathbf{a} = \mathbf{0},$$

where \mathbf{a} is our unknown period and $\mathbf{0}$ is the $(n - 1)$ -dimensional 0-vector in $(\mathbb{Z}_2)^{n-1}$. This condenses our system of $n - 1$ equations, $a \cdot w_k = 0$, into a single matrix relation.

Cost of Using GE to Determine Linear Independence

In summary, we have employed the classical algorithm GE to handle our linear-independence test. GE is $O(n^3)$ and is applied once for each pass of our quantum circuit. Since that circuit was $O(n)$, the new total cost (so far) for Simon's algorithm is $O(n^4)$. (We'll do better.)

18.13.2 Completing the Basis with an n th Vector Not Orthogonal to a

Once we are finished assembling the set of $n - 1$ vectors, $\mathcal{W} = \{w_0, \dots, w_{n-2}\}$, we will add a final n th vector, w_{n-1} , to the set – a vector which is *not orthogonal* to a (remember, a might not be such a vector – it will be orthogonal to itself exactly half the time in our strange mod-2 dot product). w_{n-1} can be quickly determined and inserted into the correct position of the reduced W ($\simeq \mathcal{W}$) matrix as follows.

1. Starting from the top row, w_0 , look for the last (lowest) w_k which has its leading 1 in column k (i.e., it has its leading 1 on W 's diagonal, but w_{k+1} , directly below it, has a 0 in its $(k + 1)$ st position).

- If such a w_k can be found, define

$$w_{n-1} \equiv 2^{n-2-k} \longleftrightarrow \begin{cases} (0, 0, \dots, 0, 1, 0, \dots, 0) \\ \uparrow \\ k + 1\text{st position from left} \end{cases}$$

and place this new w_{n-1} directly below w_k , pushing all the vectors in the old rows $k + 1$ and greater down to accommodate the insertion. Call the augmented matrix, W' .

Before:

$$W = \begin{pmatrix} 1 & w_{01} & w_{02} & w_{03} & w_{04} & w_{05} \\ 0 & 1 & w_{12} & w_{13} & w_{14} & w_{15} \\ 0 & 0 & 1 & w_{23} & w_{24} & w_{25} \\ 0 & 0 & 0 & 0 & 1 & w_{35} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \longleftarrow w_k$$

After:

$$W' \equiv \begin{pmatrix} 1 & w_{01} & w_{02} & w_{03} & w_{04} & w_{05} \\ 0 & 1 & w_{12} & w_{13} & w_{14} & w_{15} \\ 0 & 0 & 1 & w_{23} & w_{24} & w_{25} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & w_{35} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \longleftarrow w_{n-1}$$

- If there is no such w_k , then either
 - W has all 1s on its diagonal, or
 - W has all 0s on its diagonal.
- If W has all 1s on its diagonal, the final row of W is of the form

$$w_{n-2} = (0, 0, \dots, 0, 1, *).$$

Define

$$w_{n-1} \equiv 2^0 = 1 \longleftrightarrow (0, 0, \dots, 0, 0, 1),$$

and place this new w_{n-1} after w_{n-2} last old row, making w_{n-1} the new bottom row of W . Call the augmented matrix, W' .

Before:

$$W = \begin{pmatrix} 1 & w_{01} & w_{02} & w_{03} & w_{04} & w_{05} \\ 0 & 1 & w_{12} & w_{13} & w_{14} & w_{15} \\ 0 & 0 & 1 & w_{23} & w_{24} & w_{25} \\ 0 & 0 & 0 & 1 & w_{34} & w_{35} \\ 0 & 0 & 0 & 0 & 1 & w_{45} \end{pmatrix} \longleftarrow w_{n-2}$$

After:

$$W \equiv \begin{pmatrix} 1 & w_{01} & w_{02} & w_{03} & w_{04} & w_{05} \\ 0 & 1 & w_{12} & w_{13} & w_{14} & w_{15} \\ 0 & 0 & 1 & w_{23} & w_{24} & w_{25} \\ 0 & 0 & 0 & 1 & w_{34} & w_{35} \\ 0 & 0 & 0 & 0 & 1 & w_{45} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \longleftarrow w_{n-1}$$

- **Exercise.** If W has all 0s on its diagonal, explain what to do, showing example matrices.

[**Exercise.** Prove all outstanding claims in step 1.]

2. Into the $(n - 1)$ -dimensional $\mathbf{0}$ vector comprising the RHS of the equation

$$W \cdot \mathbf{a} = \mathbf{0},$$

insert a 1 into the position corresponding to the new row in W . Push any 0s down, as needed, to accommodate this 1. It will now be an n -dimensional vector corresponding to 2^k for some k .

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

That will produce a full set of n linearly independent vectors for \mathbb{Z}_{2^n} in reduced echelon form, which we call W' .

Cost of Completing the Basis

The above process consists of a small number of $O(n)$ operations, each occurring in series with each other and with the loops that come before. Therefore, its $O(n)$ adds nothing to the previous complexity, which now stands at $O(n^4)$.

18.13.3 Using Back-Substitution to Close the Deal

We are, metaphorically, 99% of the way to finding the period of f . We want to solve

$$W \cdot \mathbf{a} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix},$$

but W is already in *reduced-echelon form*. We need only apply *mod-2 back-substitution* to extract the solution vector, \mathbf{a} .

Cost of Back Substitution to the Algorithm

This is an $O(n^2)$ activity done *in series* with the loops that come before. Therefore, its $O(n^2)$ adds nothing to the previous complexity, which still stands at $O(n^4)$.

18.13.4 The Full Cost of the Hidden Classical Algorithms

We have accounted for the classical cost of testing the linear independence and solving the system of equations. In the process, we have demonstrated that it increases the complexity by a factor n^3 , making the full algorithm $\in O(n^4)$, not counting the oracle, whose complexity is unknown to us.

We will do better, though, by leveraging mod-2 shortcuts that are integrated into Simon's algorithm. You'll see.

The melody that keeps repeating in our head, though, is the footnote that this entire analysis is *relative* to the quantum oracle U_f , the reversible operator associated with the \mathbb{Z}_{2^n} periodic function, f . Its complexity is that of the black box for f , itself. We do not generally know that complexity, and it may well be very bad. Fortunately, in some special cases of great interest, we know enough about the function to be able to state that it has polynomial time complexity, often of a low polynomial order.

18.14 Adjusted Algorithm

18.14.1 New Linear Independence Step

We now integrate Gaussian elimination into Simon's algorithm during the test for linear independence. Here is the step, as originally presented:

3. Use a classical algorithm to determine whether or not z is linearly dependent on the vectors in \mathcal{W} .
 - If it is independent, name it w_j , where j is the number of elements already stored in \mathcal{W} , and add w_j to \mathcal{W} .
 - if $j = n - 2$, we have $n - 1$ linearly-independent vectors in \mathcal{W} and are done; break the loop.
 - If it is not independent (which includes the special case $z = 0$, even when \mathcal{W} is still empty), then continue to the next pass of the loop.

The entire block can be replaced by the following, which not only tests for linear independence, but keeps \mathcal{W} 's associated matrix, W , in reduced echelon form and does so leveraging the special simplicity afforded by mod-2 arithmetic.

The algorithm is more readable if we carry an example along. Assume that, to date, \mathcal{W} is represented by

$$\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

and we just pulled

$$(0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1)$$

from our circuit.

New Step 3

3. **Loop (new *inner* loop) until either z has been added to $W (\simeq \mathcal{W})$ or $z = 0$ produced.**
 - (a) **$m \leftarrow$ most significant bit (MSB) position of z that contains a 1.**
 $(O(n))$
 [In our example, $m =$ bit #5 (don't forget that we count from the right, not the left)]
 - (b) **Search W for a w_k row with the same non-0 MSB position, m .**
 $(O(n))$
 [In our example, w_1 is the (only) row with an MSB of 1 $\neq 0$ in bit #5]
 - **If a row vector w_k with same non-0 MSB is *not found*, insert z between the two rows of W that will guarantee the preservation of its reduced echelon form, effectively adding it to \mathcal{W} . End this loop.**
 [In our example, this case does not occur.]

- If a row vector w_k with same non-0 MSB *is found* replace

$$z \leftarrow z \oplus w_k,$$

effectively replacing its old MSB with a 0. Continue to next pass of this loop. ($O(n)$ for \oplus)

[In our example,

$$z \leftarrow z \oplus w_1 = (000001101)]$$

[**Exercise.** Finish this example, repeating the loop as many times as needed, and decide whether the original z produces an augmented \mathcal{W} or turns z into 0.]

[**Exercise.** Try the loop with $z = 000001100$.]

[**Exercise.** Try the loop with $z = 010001100$.]

Why this Works

In the proposed loop, we are considering whether or not to add some z to our set, \mathcal{W} .

- If we **do not** find a w_k whose MSB matches the MSB of z , then adding z to \mathcal{W} in the prescribed fashion produces an additional row for our matrix, while preserving its *echelon form*. Therefore, the rows are still *linearly independent*. (This was an exercise from the *Gaussian elimination* section.) We have increased \mathcal{W} by one vector, and thereby broadened its span to a subspace with twice as many vectors, all are orthogonal to a .

[**Exercise.** Fill in the details.]

- Say **we do** find a w_k whose MSB matches the MSB of z . $z \oplus w_k$ is, by definition, in the span of $\mathcal{W} \cup \{z\}$. Since z is *orthogonal* to a , so, too is the new $z = z \oplus w_k$, but the new z has more 0s, bringing us one loop pass closer to either adding it \mathcal{W} (the above case) or arriving at a termination condition in which we ultimately produce $z = 0$, which is never independent of any set, so neither was the previous z s that got us to this point (including the original z produced by the circuit.)

[**Exercise.** Fill in the details, making sure to explain why z and $z \oplus w_k$ are either a) both linearly independent of \mathcal{W} or b) both not.]

Cost of New Step 3

This step has a new inner-loop, relative to the outer quantum sampling loop, which has, at most, $n - 1$ passes (we move the MSB to the right each time). Inside the loop we apply some $O(n)$ operations in series with each other. Therefore, the total cost of the Step 3 loop is $O(n^2)$, which includes all arithmetic.

Summary

We are not applying GE all at once to a single matrix. Rather, we are doing an $O(n^2)$ operation after each quantum sample that keeps the accumulated W set in eternal echelon-reduced form. So, it's a custom $O(n^2)$ algorithm nested within the quantum $O(n)$ loop, giving an outer complexity of $O(n^3)$.

18.14.2 New Solution of System Step

Finally, we integrate back-substitution into the final step of the original algorithm.

The original algorithm's final step was

- Otherwise, we succeeded. Add an n th vector, w_{n-1} , which is linearly independent to this set (and therefore not orthogonal to a , by a previous exercise), done easily using a simple classical observation, demonstrated below. This produces a system of n independent equations satisfying

$$w_k \cdot a = \begin{cases} 0, & k = 0, \dots, n-2 \\ 1, & k = n-1 \end{cases}$$

which has a unique non-zero solution.

Replace this with the new final step,

- Otherwise, we succeeded and W is an $(n-1) \times n$ matrix in reduced echelon form. Add an n th row vector, w_{n-1} , which is linearly independent to W 's rows (and therefore not orthogonal to a), using the process described in *Solving the Final Set of Linear Equations*, above. That was an $O(n)$ process that produced an $n \times n$ W , also in reduced echelon form. We now have a system of n independent equations satisfying

$$w_k \cdot a = \begin{cases} 0, & k = 0, \dots, n-2 \\ 1, & k = n-1 \end{cases}$$

which is already in reduced echelon form. Solve it using only back-substitution, which is $O(n^2)$.

The take-away is that we have already produced the echelon form as part of the linear-independence tests, so we are positioned to solve the system using only back-substitution, $O(n^2)$.

18.14.3 Cost of Adjusted Implementation of Simon's Algorithm

We detailed two adjustments to the original algorithm. The first was the test for linear independence using a mod-2 step that simultaneously resulted in GE at the end of all the quantum sampling. The second was the solution of the system of equations.

Cost of the mod-2 Test for Independence

As already noted, the step-3 loop is repeated at most $n - 1$ times, since we push the MSB of z to the right at least one position each pass. Inside this loop, we have $O(n)$ operations, all applied in series. The nesting in step 3, therefore, produces an $O(n^2)$ complexity. Step 3 is within the outer quantum loop, $O(n)$, which brings the outer-most complexity to $O(n^3)$, so far

Cost of Solving the System

We saw that we could solve the system using only back-substitution, $O(n^2)$. This is done outside the entire quantum loop.

The total cost, quantum + classical, using our fancy footwork is, therefore, $O(n^3)$, relativized to the oracle.

18.15 Classical Complexity of Simon's Problem

Classically, this problem is hard, that is, deterministically, we certainly need a number of trials that increases exponentially in n to get the period, and even if we are satisfied with a small error, we would *still* need to take an exponential number of samples to achieve that (and not just any exponential number of samples, but a really big one). Let's demonstrate all this.

18.15.1 Classical Deterministic Cost

Recall that the domain can be partitioned (in more than one way) into two disjoint sets, R and Q ,

$$\begin{aligned}\mathbb{Z}_{2^n} &= R \cup Q \\ &= \{ \dots, x, \dots \} \cup \{ \dots, x \oplus a, \dots \},\end{aligned}$$

with f one-to-one on R and Q , individually. We pick x s at random (avoiding duplicates) and plug each one into f – or a classical *oracle* of f if you like,

$$x \text{ --- } \boxed{\text{Classical } f} \text{ --- } f(x).$$

If we sample f any fewer than (*half the domain size*) + 1, that is, $(2^n/2) + 1 = 2^{n-1} + 1$, times, we may be unlucky enough that all of our outputs, $f(x)$, are images of $x \in R$ (or all $\in Q$), which would produce all distinct functional values. There is no way to determine what a is if we don't get a duplicate output, $f(x') = f(x'')$ for some distinct $x', x'' \in \text{dom}(f)$. (Once we do, of course, $a = x' \oplus x''$, but until that time, no dice.)

Therefore, we have to sample $2^{n-1} + 1$ times, exponential in n , to be sure we get at least one x from R and one from Q , thus producing a duplicate.

18.15.2 Classical Probabilistic Cost

However, what if we were satisfied to find a with a pre-determined probability of error, ε ? This means that we could choose an ε as small as we like and know that if we took $m = m(\varepsilon, n)$ samples, we would get the right answer, a , with probability

$$P(\text{figure out } a, \text{ correctly}) \geq 1 - \varepsilon.$$

The functional dependence $m = m(\varepsilon, n)$ is just a way to express that we are allowed to let the number of samples, m , depend on both how small an error we want and also how big the domain of f is. For example, if we could show that $m = 2^{1/\varepsilon}$ worked, then since that is not dependent on n the complexity would be constant time. On the other hand, if we could only prove that an $m = n^4 2^{1/\varepsilon}$ worked, then the algorithm would be $O(n^4)$.

The function dependence we care about does not involve ε , only n , so really, we are interested in $m = m(n)$.

What we will show is that even if we let m be a particular function that grows exponentially in n , we won't succeed. That's not to say *every* exponentially increasing sample size which is a function of n would fail – we already know that if we chose $2^{n-1} + 1$ we will succeed with certainty. But we'll see that some smaller exponential function of n , specifically $m = 2^{n/4}$, will not work, and if that won't work then no polynomial dependence on n , which necessarily grows more slowly than $m = 2^{n/4}$, has a chance, either.

An Upper Bound for Getting Repeats in m Samples

For the moment, we won't concern ourselves with whether or not m is some function of ε or n . Instead, let's compute an upper bound on the probability of getting a repeat when sampling a classical oracle m times. That is, we'll get the probability as a function of m , alone. Afterwards, we can stand back and see what kind of dependence m would require on n in order that the deck be stacked in our favor.

We are looking for the probability that at least two samples $f(x_i)$, $f(x_j)$ are *equal* when choosing m distinct inputs, $\{x_0, x_1, \dots, x_{m-1}\}$. We'll call that event \mathcal{E} . The more specific event that some pair of inputs, x_i and x_j , yield *equal* $f(x)$ s, will be referred to as \mathcal{E}_{ij} . Since \mathcal{E}_{ij} and \mathcal{E}_{ji} are the same event, we only have to list it once, so we only consider the cases where $i < j$. Clearly,

$$\mathcal{E} = \bigcup_{\substack{i, j=0 \\ i < j}}^{m-1} \mathcal{E}_{ij}$$

The probability of a union is the sum of the probabilities of the individual events

minus the probabilities of the various intersections of the individual events,

$$\begin{aligned} P(\mathcal{E}) &= \sum_{\substack{i,j=0 \\ i < j}}^{m-1} P(\mathcal{E}_{ij}) - \sum_{\substack{\text{various} \\ \text{combinations}}} P(\mathcal{E}_{ij} \wedge \mathcal{E}_{kl} \dots) \\ &\leq \sum_{\substack{i,j=0 \\ i < j}}^{m-1} P(\mathcal{E}_{ij}) . \end{aligned}$$

The number of unordered pairs, $\{i, j\}$, $i \neq j$, when taken from m things is (look up “ n choose k ” if you have never seen this)

$$\binom{m}{2} = \frac{m!}{2!(m-2)!} = \frac{m(m-1)}{2} .$$

This is exactly the number of events \mathcal{E}_{ij} that we are counting since our condition, $0 \leq i < j \leq m-1$, is in 1-to-1 correspondence with the set of unordered pairs $\{i, j\}$, i and j between 0 and $m-1$, inclusive and $i \neq j$.

Meanwhile, the probability that an individual pair produces the same f value is just the probability that we choose the second one, x_j , in such a way that it happens to be exactly $x_i \oplus a$. Since we’re intentionally not going to pick x_i a second time this leaves $2^n - 1$ choices, of which only one is $x_i \oplus a$, so that gives

$$P(\mathcal{E}_{ij}) = \frac{1}{2^n - 1} .$$

Therefore, we’ve computed the number of elements in the sum, $m(m-1)/2$, as well as the probability of each element in the sum, $1/(2^n - 1)$, so we plug back into our inequality to get

$$P(\mathcal{E}) \leq \frac{m(m-1)}{2} \cdot \frac{1}{2^n - 1} .$$

[**Exercise.** We know that when we sample $m = 2^{n-1} + 1$ times, we are certain to get a duplicate. As a *sanity check*, make sure that plugging this value of m into the derived inequality gives an upper bound that is no less than one. Any value ≥ 1 will be consistent with our result.]

This is the first formula we sought. We now go on to see what this implies about how m would need to depend on n to give a decent chance of obtaining a .

What the Estimate Tells Us about $m = m(n)$

To get our feet wet, let’s imagine the pipe dream that we can use an m that is independent of n . The bound we proved,

$$P(\mathcal{E}) \leq \frac{m(m-1)}{2} \cdot \frac{1}{2^n - 1} ,$$

tells us that any such m (say an integer $m > 1/\varepsilon^{1000000}$) is going to have an exponentially *small* probability as $n \rightarrow \infty$. So that settles that, at least.

An Upper Bound for Getting Repeats in $m = 2^{n/4}$ Samples

But, what about allowing m to be a really large proportion of our domain, like

$$m = 2^{n/4} ?$$

That's an exponential function of n , so it really grows fast. Now we're at least being realistic in our hopes. Still, they are dashed:

$$\begin{aligned}
 P(\mathcal{E}) &\leq \frac{m(m-1)}{2} \cdot \frac{1}{2^n - 1} = \frac{2^{n/4}(2^{n/4} - 1)}{2} \cdot \frac{1}{2^n - 1} \\
 &= \frac{2^{n/2} - 2^{n/4}}{2} \cdot \frac{1}{2^n - 1} < \frac{2^{n/2}}{2} \cdot \frac{1}{2^n - 1} \\
 &= \frac{1}{2} \cdot \frac{2^{n/2}}{2^n - 1} < \frac{1}{2} \cdot \frac{2^{n/2}}{2^n - 2^{n/2}} \\
 &= \frac{1}{2} \cdot \frac{2^{n/2}}{2^{n/2}(2^{n/2} - 1)} = \frac{1}{2} \cdot \frac{1}{2^{n/2} - 1} \\
 &< \frac{1}{2^{n/2} - 1},
 \end{aligned}$$

a probability that still approaches 0 (exponentially fast – unnecessary salt in the wound) as $n \rightarrow \infty$. Although we are allowing ourselves to sample the function, f , at $m = m(n)$ distinct inputs, where $m(n)$ grows exponentially in n , we cannot guarantee a reasonable probability of getting a repeat $f(x)$ for all n . The probability always shrinks to 0 when n gets large. This is exponential time complexity.

This problem is hard, even probabilistically, in classical terms. Simon's algorithm gives a (relativized) exponential speed-up over classical methods.

Chapter 19

Real and Complex Fourier Series

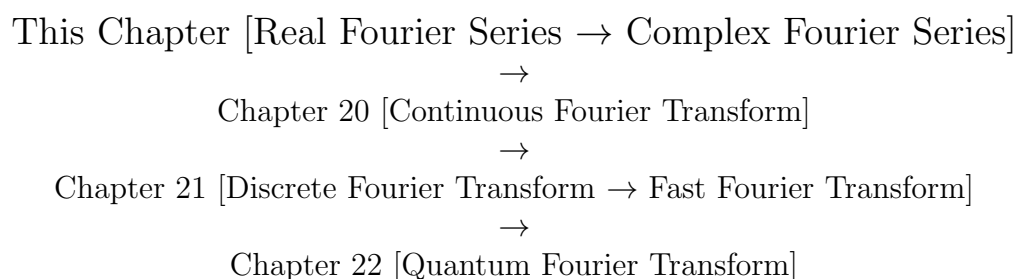
19.1 The Classical Path to Quantum Fourier Transforms

Our previous quantum algorithms made propitious use of the n th order Hadamard transform, $H^{\otimes n}$, but our next algorithm will require something a little higher octane. The fundamental rules apply: a gate is still a gate and must be unitary. As such, it can be viewed as a *basis change* at one moment and a tool to turn a separable input state like $|0\rangle^n$ into a *superposition*, the next. We'll have occasion to look at it in both lights.

Our objective in the next four lessons is to study the *quantum Fourier transform* a.k.a. the QFT . This is done in three classical chapters and one quantum chapter. Reading and studying all three classical chapters will best prepare you for the fourth QFT chapter, but you can cherry pick, skim, or even skip one or more of the three depending on your interest and prior background.

An aggressive short cut might be to try starting with the third, the *Discrete and Fast Fourier Transforms*, read that for general comprehension, then see if it gives you enough of a foundation to get through the fourth QFT chapter.

If you have time and want to learn (or review) the classic math that leads to QFT , the full path will take you through some beautiful topics:



If you do decide to invest time in the early material, you will be incidentally adding knowledge applicable to many fields besides quantum computing. You will also have

some additional insight into the QFT .

In this first lesson we introduce the concepts *frequency* and *period* and show how a periodic function can be built using only sines and cosines (*real Fourier series*) or exponentials (*complex Fourier series*). The basic concepts here make learning the *Fourier transform* (next chapter) very intuitive.

19.2 Periodic Functions and Their Friends

Fourier Series apply to functions that are either periodic or are only defined on a bounded interval. We'll first define the terms, *periodicity*, *bounded domain* and *compact support*, then we can get on with Fourier Series.

19.2.1 Periodic Functions over \mathbb{R}

A function f whose domain is (nearly) all the real numbers, \mathbb{R} , is said to be *periodic* if there is a unique smallest positive real number T , such that

$$f(x + T) = f(x), \quad \text{for all } x.$$

The number, T , is called the *period* of f .

(“Unique smallest” is a little redundant, but it condenses two ideas. Officially, we'd say that there must exist *some* $T > 0$ with the stated property and, if there did, we would call the *smallest* such T (technically the *GLB* or *greatest lower bound*) its *period*.)

The simplest examples we can conjure up instantly are the functions sine and cosine. Take sine, for example:

$$\sin(x + 2\pi) = \sin(x), \quad \text{for all } x,$$

and 2π is the smallest positive number with this property, so $a = 2\pi$ is its period. 4π and 12π satisfy the equality, but they're not as small as 2π , so they're not periods.

Its graph manifests this periodicity in the form of repetition (Figure 19.1).

I said that the domain could be “nearly” all real numbers, because it's fine if the function blows-up or is undefined on some isolated points. A good example is the tangent function, $y = \tan x$, whose period is half that of $\sin x$ but is undefined for $\pm\pi/2$, $\pm3\pi/2$, etc. (Figure 19.2).

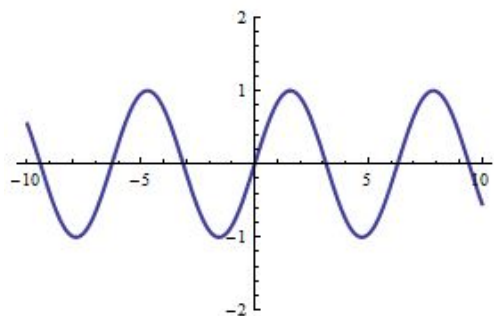


Figure 19.1: Graph of the quintessential periodic function $y = \sin x$
(period = 2π)

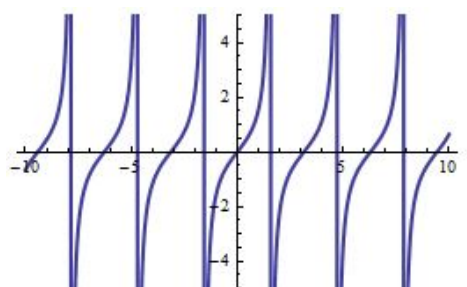


Figure 19.2: The function $y = \tan x$ blows-up at isolated points but is still periodic
(with period π)

19.2.2 Functions with Bounded Domain or Compact Support

A function which is defined only over a *bounded interval* of the real numbers (like $[0, 100]$ or $[-\pi, \pi]$) is said to have *bounded domain*. An example is:

$$f(x) = \begin{cases} x^2, & \text{if } x \in [-1, 3) \\ \text{undefined,} & \text{otherwise} \end{cases}$$

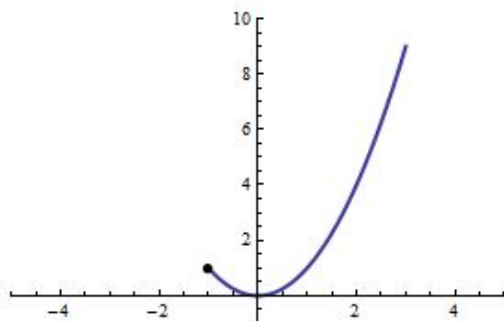


Figure 19.3: Graph of a function defined only for $x \in [-1, 3)$

[Note the notation $[-1, 3)$ for the “half-open” interval that contains the left endpoint, -1 , but not the right endpoint, 3 . We could have included 3 and not included

-1 , included both or neither. It all depends on our particular goal and function. Half-open intervals, closed on the left and open on the right, are the most useful to us.]

A subtly different concept is that of *compact support*. A function might be defined on a relatively large set, say *all* (or most) real numbers, but happen to be zero outside a bounded interval. In this case we prefer to say that it has *compact support*.

The previous example, extended to all \mathbb{R} , but set to zero outside $[-1, 3]$ is

$$f(x) = \begin{cases} x^2, & \text{if } x \in [-1, 3) \\ 0, & \text{otherwise} \end{cases}$$

(See Figure 19.4)

Terminology

The *support* of the function is the *closure* of the domain where $f \neq 0$. In the last function, although f is non-zero only for $[-1, 0) \cup (0, 3)$, we include the two points 0 and 3 in its *support* since they are part of the *closure* of the set where f is non-zero. (I realize that I have not defined *closure*, and I won't do so rigorously. For us, *closure* means adding back any points which are “right next” to places where f is non-zero, like 0 and 3 in the last example.)

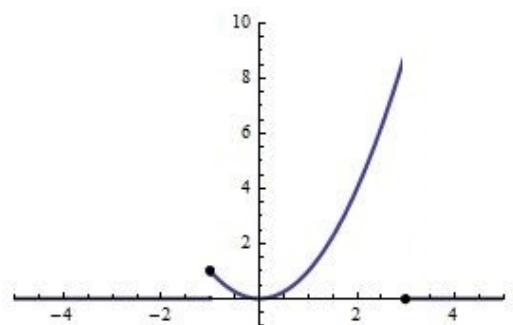


Figure 19.4: Graph of a function defined everywhere, but whose support is $[-1, 3]$, the closure of $[-1, 0) \cup (0, 3)$

19.2.3 The Connection Between Periodicity and Bounded Domain

If a function is periodic, with period T , once you know it on any half-open interval of length T , say $[0, T)$ or $[-T/2, T/2)$, you automatically know it for all x , complements of $f(x) = f(x + T)$. So we could restrict our attention to the interval $[-T/2, T/2)$ – imagining, if it suited us, that the function was *undefined* off that interval. Our understanding of the function on this interval would tell us everything there is to

know about the function elsewhere, since the rest of the graph of the function is just a repeated (or, better, “translated”) clone of what we see on this small part.

Likewise, if we had a (non-periodic) function with bounded domain, say $[a, b]$, we could throw away b to make it a half-open interval $[a, b)$ (we don’t care about f at one point, anyway). We then convert that to an *induced periodic function* by insisting that $f(x) = f(x + T)$, for $T \equiv b - a$. This defines f everywhere *off* that interval, and the expanded function agrees with f on its original domain but is now periodic with period $T = b - a$.

As a result of this duality between periodic functions and functions with bounded domain, I will be interchanging the terms *periodic* and *bounded domain* at will over the next few sections, choosing whichever one best fits the context at hand.

19.3 The Real Fourier Series of a Real Period Function

19.3.1 Definitions

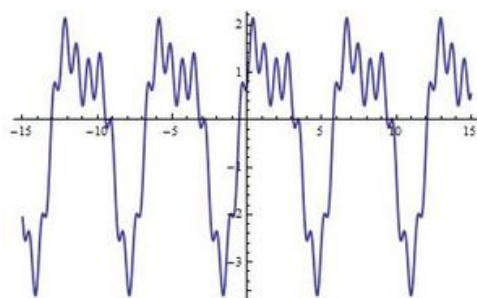


Figure 19.5: A periodic function that can be expressed as a Fourier series
(period = 2π)

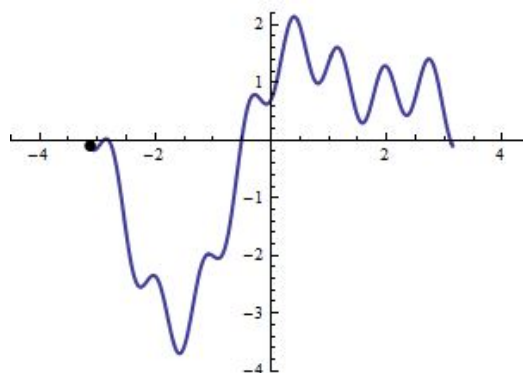


Figure 19.6: A function with bounded domain that can be expressed as a Fourier series (support width = 2π)

Until further notice we confine ourselves to functions with *domain* $\subseteq \mathbb{R}$ and *range* $\subseteq \mathbb{R}$.

Any “well-behaved” function of the real numbers that is either *periodic* (See Figure 19.5), or has *bounded domain* (See Figure 19.6), can be expressed as a sum of sines and cosines. This is true for any period or support width, T , but we normally simplify things by taking $T = 2\pi$.

The Real Fourier Series. *The real Fourier series of f , a well-behaved periodic function with period 2π , is the sum on the RHS of the equation*

$$f(x) = a_0 \frac{1}{2} + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx.$$

The sum on the RHS of this equation is called the *Fourier Series of the function f* . The functions of x (that is, $\{\sin nx\}_n$, $\{\cos nx\}_n$ and the constant function $1/2$) that appear in the sum are sometimes called the *Fourier basis functions*.

Study this carefully for a moment. There is a constant term out front ($a_0/2$), which simply shifts the function’s graph up or down, vertically. Then, there are two infinite sums involving $\cos nx$ and $\sin nx$. Each term in those sums has a coefficient – some real number a_n or b_n – in front of it. In thirty seconds we’ll see what all that means.

[The term *well-behaved* could take us all week to explore, but every function that you are likely to think of, that we will need, or that comes up in physics and engineering, is almost certainly going to be well-behaved.]

19.3.2 Interpretation of the Fourier Series

Each sinusoid in the Fourier series has a certain *frequency* associated with it: the n in $\sin nx$ or $\cos nx$. The larger the n , the higher the frequency of that sine or cosine. (See Figure 19.7)

A given f has different amounts (generally) of each of its frequencies. That’s where the coefficients in front of the sines and cosines come into play. The way we think about the collection of coefficients, $\{a_n, b_n\}$, in the Fourier expansion is summarized in the bulleted list, below.

- When *small- n* coefficients like a_0, a_1, b_1 or b_2 are large in magnitude, the function possesses significant *low frequency characteristics* (visible by slowly changing, large curvature in the graph of f).
- When the *higher- n* coefficients like a_{50}, b_{90} or b_{1000} are large in magnitude, the function has lots of *high frequency characteristics* (busy squiggling) going on.

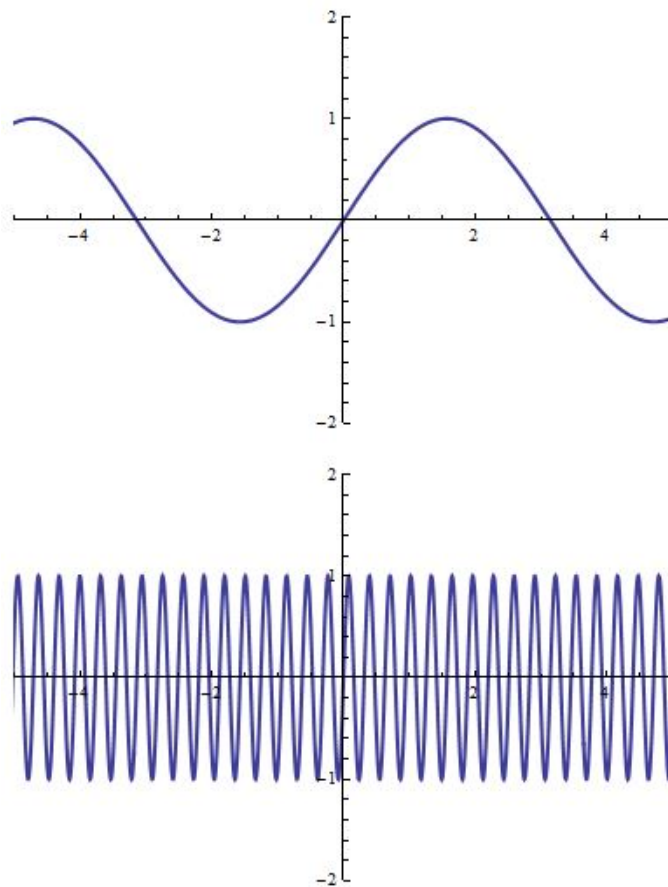


Figure 19.7: A low frequency ($n = 1 : \sin x$) and high frequency ($n = 20 : \sin 20x$) basis function in the Fourier series

- The coefficients, $\{a_n\}$ and $\{b_n\}$ are often called the *weights* or *amplitudes* of their respective basis functions (in front of which they stand). If $|a_n|$ is large, there's a “lot of” $\cos nx$ needed in the recipe to prepare a meal of $f(x)$ (same goes for $|b_n|$ and $\sin nx$). Each coefficient adds just the right amount, or weight, of its corresponding sinusoid to make the desired function f .
- As mentioned, the functions $\{\sin nx\}$ and $\{\cos nx\}$ are sometimes called the *Fourier basis functions*, at other times the *normal modes*, and in some contexts the *Fourier eigenfunctions*. Whatever we call them, they represent the individual ingredients used to build the original f out of trigonometric objects, and the weights instruct the chef how much of each function to add to the recipe: “a pinch of $\cos 3x$, a quart of $\sin 5x$, three tablespoons of $\sin 17x$ ”, etc.
- **Caution:** A sharp turn (f' blows up or there is a jump discontinuity) at even a single domain point is a kind of extreme wiggleness, so the function may appear smooth except for one or two angled or cornered points, but those points require lots of high frequencies in order to be modeled by the Fourier series.

19.3.3 Example of a Fourier Series

To bring all this into focus, we look at the Fourier series of a function that is about as simple as you can imagine,

$$f(x) = x, \quad x \in [-\pi, \pi).$$

f 's graph on the fundamental interval $[-\pi, \pi)$ is shown in Figure 19.8. When viewed as a periodic function, f will appear as shown in Figure 19.9. Either way, we represent it as the following Fourier series

$$x = \sum_{n=1}^{\infty} (-1)^{n+1} \left(\frac{2}{n} \right) \sin nx.$$

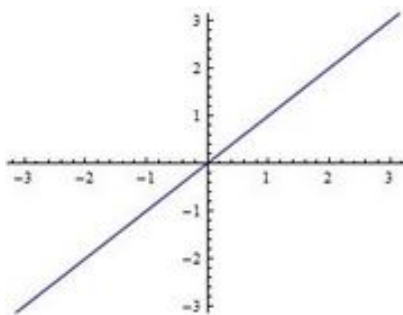


Figure 19.8: $f(x) = x$, defined only on bounded domain $[-\pi, \pi)$

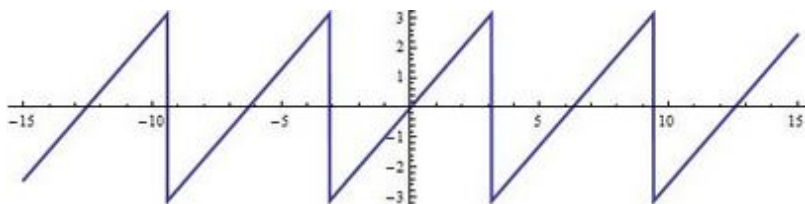


Figure 19.9: $f(x) = x$ as a periodic function with fundamental interval $[-\pi, \pi)$:

The expression of such a simple function, x , as the infinite sum of sines should strike you as somewhat odd. Why do we even bother? There are times when we need the Fourier expansion of even a simple function like $f(x) = x$ in order to fully analyze it. Or perhaps we wish to build a circuit to generate a signal like $f(x) = x$ electronically in a signal generator. Circuits naturally have sinusoids at their disposal, constructed by squeezing, stretching and amplifying the 60 Hz signal coming from the AC outlet in the wall. However, they don't have an $f(x) = x$ signal, so that one must built it from sinusoids, and the Fourier series provides the blueprint for the circuit.

Why is the Fourier series of $f(x) = x$ so complicated? First, f , has a jump discontinuity – a sharp edge – so it takes a lot of high frequency components to shape it. If you look at the periodic sawtooth shape of $f(x) = x$, you'll see these sharp points. Second, f 's graph is linear (but not constant). Crafting a non-horizontal straight line from curvy sines and cosines requires considerable tinkering.

Finite Approximations

While the Fourier sum is exact (for well-behaved f) the vagaries of hardware require that we merely approximate it by taking only a partial sum that ends at some finite $n = N < \infty$. For our f under consideration, the first 25 coefficients of the sines are shown in Figure 19.10 and graphed in Figure 19.11.

$$\left\{ 2, -1, \frac{2}{3}, -\frac{1}{2}, \frac{2}{5}, -\frac{1}{3}, \frac{2}{7}, -\frac{1}{4}, \frac{2}{9}, -\frac{1}{5}, \frac{2}{11}, -\frac{1}{6}, \frac{2}{13}, -\frac{1}{7}, \frac{2}{15}, -\frac{1}{8}, \frac{2}{17}, -\frac{1}{9}, \frac{2}{19}, -\frac{1}{10}, \frac{2}{21}, -\frac{1}{11}, \frac{2}{23}, -\frac{1}{12}, \frac{2}{25} \right\}$$

Figure 19.10: First 25 Fourier coefficients of $f(x) = x$

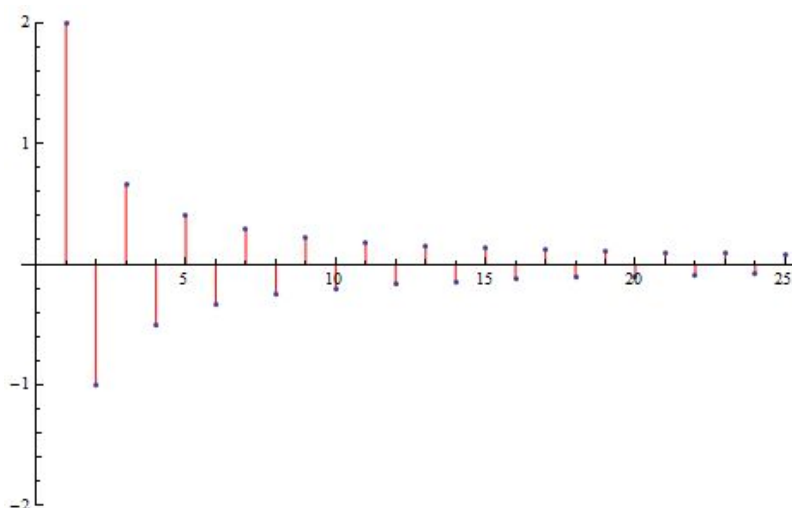


Figure 19.11: Graph of the Fourier coefficients of $f(x) = x$

The Spectrum

Collectively, the Fourier coefficients (or their graph) is called the *spectrum* of f . It is a possibly infinite list (or graph) of the weights of the various frequencies contained in f .

Viewed in this way, the coefficients, themselves, represent a new function, $F(n)$.

The Fourier Series as an Operator Mapping Functions to Functions

The Fourier mechanism is a kind of operator, \mathcal{FS} , applied to $f(x)$ to get a new function, $F(n)$, which is *also* called the *spectrum*.

$$\begin{array}{c}
F(n) = \mathcal{FS}[f(x)] \\
\updownarrow \\
\{a_n, b_n\}
\end{array}$$

The catch is, this new function, F , is only defined on the *non-negative integers*. In fact, if you look closely, it's really two separate functions of integers, $a(n) = a_n$ and $b(n) = b_n$. But that's okay – we only want to get comfortable with the idea that the Fourier “operator” takes one function, $f(x)$, domain \mathbb{R} , and produces another function, its *spectrum* $F(n)$, domain $\mathbb{Z}_{\geq 0}$.

$$\begin{array}{lcl}
f & : & \mathbb{R} \longrightarrow \mathbb{R} \\
F & : & \mathbb{Z}_{\geq 0} \longrightarrow \mathbb{R} \\
f & \xrightarrow{FS} & F
\end{array}$$

F contains every ounce of information of the original f , only expressed in a different form.

Computing Fourier Coefficients

The way to produce the Fourier coefficients, $\{a_n, b_n\}$ of a function, f , is through these easy formulas (that I won't derive),

$$\begin{aligned}
a_0 &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) dx, \quad n = 0, \\
a_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx dx, \quad n > 0, \quad \text{and} \\
b_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx dx, \quad n > 0.
\end{aligned}$$

They work for functions which have period 2π or bounded domain $[-\pi, \pi)$. For some other period T , we would need to multiply or divide $T/(2\pi)$ in the right places (check on-line or see if you can derive the general formula).

Using these formulas, you can do lots of exercises computing the Fourier series of various functions restricted to the interval $[-\pi, \pi)$.

In practice, we can't build circuits or algorithms that will generate an infinite sum of frequencies, but it's easy enough to stop after any finite number of terms. Figure 19.12 shows what we get if we stop the sum after three terms.

It's not very impressive, but remember, we are using only three sines/cosines to approximate a diagonal line. Not bad, when you think of it that way. Let's take the first 50 terms and see what we get (Figure 19.13).

Now we understand how Fourier series work. We can see the close approximation to the straight line near the middle of the domain and also recognize the high frequency

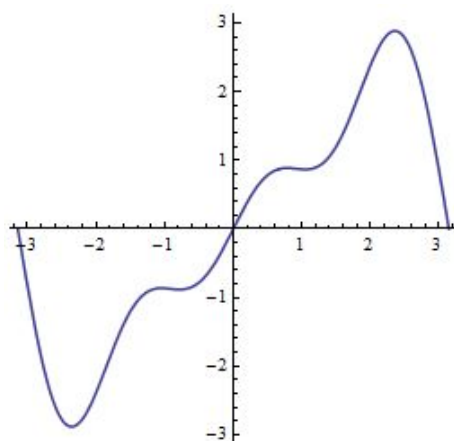


Figure 19.12: Fourier partial sum of $f(x) = x$ to $n = 3$

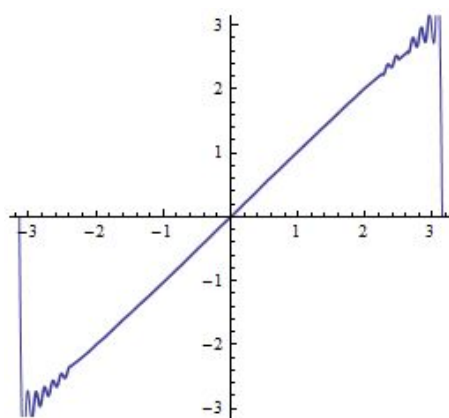


Figure 19.13: Fourier partial sum of $f(x) = x$ to $n = 50$

effort to get at the sharp “teeth” at each end. Taking it out to 1000 terms, Figure [19.14](#) shows a stage which we might find acceptable in some applications.

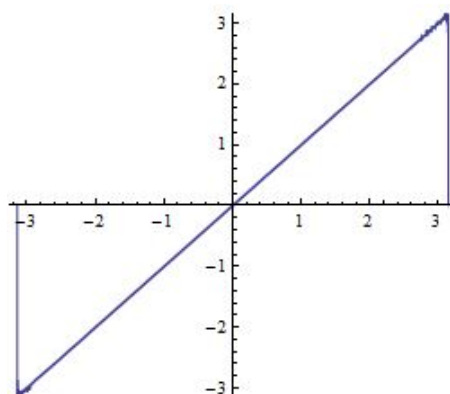


Figure 19.14: Fourier partial sum of $f(x) = x$ to $n = 1000$

Note: The Fourier series

$$f(x) = a_0 \frac{1}{2} + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx$$

always produces a function of x which is periodic on the *entire real line*, even if we started with (and only care about) a function, $f(x)$ with bounded domain. The RHS of this “equation” matches the original f over its original domain, but the domain of the RHS may be larger. To illustrate this, if we were modeling the function $f(x) = x^2$, restricted to $[-\pi, \pi)$, the Fourier series would converge on the entire real line, \mathbb{R} , beyond the original domain (See Figure 19.15).

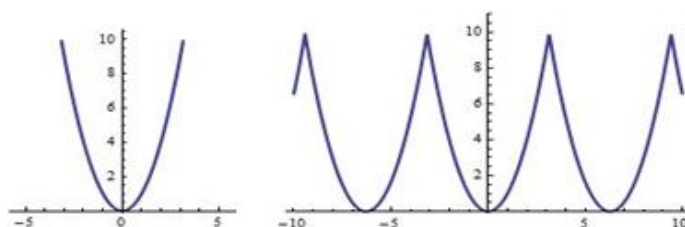


Figure 19.15: $f(x)$ has bounded domain, but its Fourier expansion is periodic.

The way to think about and deal with this is to simply ignore the infinite number of period-shifted copies of the original magnanimously afforded by the Fourier series expression (as a function of x) and only take the part of the graph that lies above f ’s original, bounded domain.

Compact Support, Alone, is Not Enough

In contrast, if we defined a function which is defined over all \mathbb{R} but had compact support, $[-\pi, \pi]$, it would not have a Fourier series; no single weighted sum of sinusoids could build this function, because we can’t reconstruct the flat $f(x) = 0$ regions on the left and right with a single (even infinite) sum. We *can* break it up into three regions, and deal with each separately, but that’s a different story.

19.4 The Complex Fourier Series of a Periodic Function

19.4.1 Definitions

We continue to study *real* functions of a *real* variable, $f(x)$, which are either *periodic* or have a *bounded domain*. We still want to express them as a weighted sum of special “pure” *frequency functions*. I remind you, also, that we are restricting our attention to functions with period (or domain length) 2π , but our results will apply to functions having any period T if we tweak them using factors of T or $1/T$ in the right places.

To convert from sines and cosines to complex numbers, one formula should come to mind: *Euler’s formula*,

$$e^{i\theta} = \cos \theta + i \sin \theta .$$

Solving this for cosine and sine, we find:

$$\begin{aligned} \cos \theta &= \frac{e^{i\theta} + e^{-i\theta}}{2} \\ \sin \theta &= \frac{e^{i\theta} - e^{-i\theta}}{2i} \end{aligned}$$

While I won’t show the four or five steps, explicitly, we can apply these equivalences to the real Fourier expression for f ,

$$f(x) = a_0 \frac{1}{2} + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx,$$

to get an equivalent expression,

$$f(x) = a_0 \frac{1}{2} + \sum_{n=1}^{\infty} \frac{1}{2} (a_n - ib_n) e^{inx} + \sum_{n=1}^{\infty} \frac{1}{2} (a_n + ib_n) e^{-inx}.$$

Now, let n run negative to form our *Complex Fourier Series* of the (same) function f .

The Complex Fourier Series. *The **complex Fourier series** of f , a well-behaved periodic function with period 2π , is the sum on the RHS of the equation*

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx},$$

where

$$c_n \equiv \begin{cases} \frac{1}{2} (a_n - ib_n) , & n > 0 \\ \frac{1}{2} (a_{-n} + ib_{-n}) , & n < 0 \\ \frac{1}{2} a_0 , & n = 0 \end{cases} .$$

The complex Fourier series is a cleaner sum than the real Fourier expansion which uses sinusoids, and it allows us to deal with all the coefficients at once when doing computations. The price we pay is that the coefficients, c_n , are generally *complex* (not to mention the exponentials, themselves). However, even when they are all complex, *the sum is still real*. We have been – and continue to be – interested in real-valued functions of a real variable. The fact that we are using complex functions and coefficients to construct a real-valued function does not change our calculus.

19.4.2 Computing The Complex Fourier Coefficients

I expressed the c_n of the complex Fourier series in terms of the a_n and b_n of the real Fourier series. That was to demonstrate that this new form existed, not to encourage you to first compute the real Fourier series and, from that, compute the c_n of the complex form. The formula for computing the complex spectrum, $\{c_n\}$ is

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-inx} f(x) dx .$$

We can learn a lot by placing the complex Fourier expansion of our periodic f on the same line as the (new, explicit) expression for the c_n .

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}, \quad \text{where } c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-inx} f(x) dx$$

Make a mental note of the following observations by confirming, visually, that they're true.

- We are expressing f as a weighted-sum (weights c_n) of *complex basis functions* e^{inx} . But the integral is also a kind of sum, so we are simultaneously expressing the c_n as a “weighted-sum” (weights $f(x)$) of complex basis functions e^{-inx}
- Under the *first* sum, the x in the n th “basis function,” e^{inx} , is fixed (that's the number at which we are evaluating f), but n is a summation variable; under the *second* integration, it is the n of e^{-inx} which is fixed (that's the index at which we are evaluating the coefficient, c_n), and x is the integration variable. So the roles of x and n are reversed.
- The sequence of complex weights, $\{c_n\}$, is nothing more than a function $c(n)$ on the set of all integers, \mathbb{Z} . This emphasizes that not only is $f()$ a function of x , but $c()$ is a function of n . This way of thinking makes the above expression look even more symmetric

$$f(x) = \sum_{n=-\infty}^{\infty} c(n) e^{inx},$$

while,

$$c(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} dx .$$

- If we know a function, $f(x)$, we compute a specific spectrum value, $c(n)$, by multiplying each $f(x)$ by the complex basis function e^{-inx} and integrating x from $-\pi$ to π . If we know a spectrum, $\{c(n)\}$, we compute a specific functional value, $f(x)$, by multiplying each $c(n)$ by the complex basis functions e^{inx} and summing n from $-\infty$ to ∞ .
- The correspondence between functions over \mathbb{R} and \mathbb{Z} ,

$$f(x) \longleftrightarrow c(n)$$

is, conceptually, its own inverse. You do (very roughly) the same thing to get the *spectrum* from the *function* as you do to build the *function* from its *spectrum*.

Example

Let's expand $f(x) = x$ along the complex Fourier basis.

Our goal is to find the complex coefficients, c_n , that make the following true:

$$x = \sum_{n=-\infty}^{\infty} c_n e^{inx}.$$

Always begin with the easy one:

$$c_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} x e^0 dx = \frac{1}{2\pi} \int_{-\pi}^{\pi} x dx = 0.$$

The others, both positive and negative, are done in a single breath,

$$\begin{aligned} c_n &= \frac{1}{2\pi} \int_{-\pi}^{\pi} x e^{-inx} dx \\ &= \frac{1}{2\pi} \frac{e^{-inx}}{n^2} (inx + 1) \Big|_{-\pi}^{\pi} \\ &= \frac{1}{2\pi} \frac{1}{n^2} [in\pi (e^{-in\pi} + e^{in\pi}) + (e^{-in\pi} - e^{in\pi})] \\ &= \frac{1}{2\pi} \frac{1}{n^2} [in\pi (2 \cos n\pi) - (2i \sin n\pi)] \\ &= \frac{1}{2\pi} \frac{1}{n^2} 2i [n\pi (\cos n\pi) - 0] \\ &= \frac{i}{n} \cos n\pi \\ &= (-1)^n \frac{i}{n}. \end{aligned}$$

Putting it all together, we get

$$x = \sum_{\substack{n=-\infty \\ n \neq 0}}^{\infty} (-1)^n \frac{i}{n} e^{inx}.$$

19.5 Periods and Frequencies

19.5.1 The Frequency of any Periodic Function

This short section will be very useful in motivating the approach to Shor's period-finding algorithm. In order to use it, I'll temporarily need the letter f to mean *frequency* (in keeping with the classical scientific literature), so we're going to call our periodic function under study $g(x)$.

We've been studying periodic functions, $g(x)$ of real x which have periods $T = 2\pi$, and we have shown how to express them as a sum of either real sines and cosines,

$$g(x) = a_0 \frac{1}{2} + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx,$$

or complex exponentials,

$$g(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}.$$

Each term in these sums has a certain frequency: the n . You may have gotten the impression that the term "frequency" only applies to functions of the form $\sin(nx)$, $\cos(nx)$ or e^{inx} . If so, I'd like to disabuse you of that notion (for which my presentation was partly responsible). In fact any periodic function has a frequency, even those which are somewhat arbitrary looking.

We'll relax the requirement that our periodic functions have period $T = 2\pi$. That was merely a convenience to make its Fourier sum take on a standard form. For the moment, we don't care about Fourier series.

We will define frequency twice, first in the usual way and then using a common alternative.

19.5.2 Ordinary Frequency

We know what the *period*, T , of a periodic $g(x)$ means. The *frequency*, f , of a periodic $g(x)$ is just the *reciprocal of the period*,

$$f \equiv \frac{1}{T}.$$

The interpretation is what's important.

- The frequency f tells you how many *periods* of $g(x)$ fit into any *unit interval* (like $[0, 1)$ or $[-1/2, 1/2)$).
- When the period is on the small side, like $1/10$, the frequency will be large (10). In this case we'd see 10 repeated patterns if we graphed $g(x)$ between $-1/2$ and $1/2$. (See Figure 19.16)

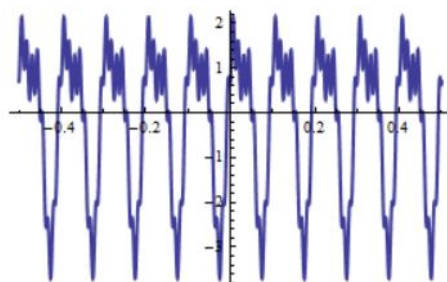


Figure 19.16: $f = 10$ produces ten copies of the period in $[-.5, .5)$

- When the period is larger, like 10, the frequency will be small ($1/10$). In this case we'd see only a small portion of the graph of $g(x)$ between $-1/2$ and $1/2$, missing the full view of its curves and gyrations, only apparent if we were to graph it over much larger interval containing at least one full period, say -5 to $+5$ or 0 to 10 . (See Figure 19.17)

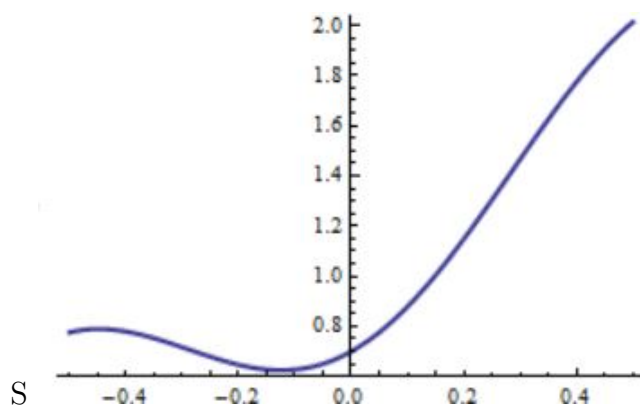


Figure 19.17: $f = .1$ only reveals one tenth of period in $[-.5, .5)$

The take-away here is that

$$f \cdot T = 1$$

and if you know g 's *period*, you know its *frequency* (and vice versa).

19.5.3 Angular Frequency

When we ask the question “How many periods fit an interval of length l ?” there's nothing forcing us to choose $l = 1$. That's a common choice in physics only because it produces answers *per second*, *per unit* or *per radian* of some cyclic phenomenon. If, instead, we wanted to express things *per cycle* or *per revolution* we would choose $l = 2\pi$. It's a slightly larger interval, so the same function would squeeze 6+ times as many periods into it; if you were to repeat something 10 times in the space (or time) of one unit or radian, you would get 62.8 repetitions in the span of a full revolution of 2π units or radians.

Angular frequency, usually denoted by the letter ω , is therefore defined to be

$$\omega \equiv \frac{2\pi}{T}.$$

The relationship between *angular frequency* and *ordinary frequency* is

$$\omega = 2\pi f.$$

The relationship between *period* and *angular frequency* has the same interpretation and form as that for ordinary frequency with the qualitatively unimportant change that the number 1 now becomes 2π . In particular, if you know the function's *angular frequency*, you know its *period*, courtesy of

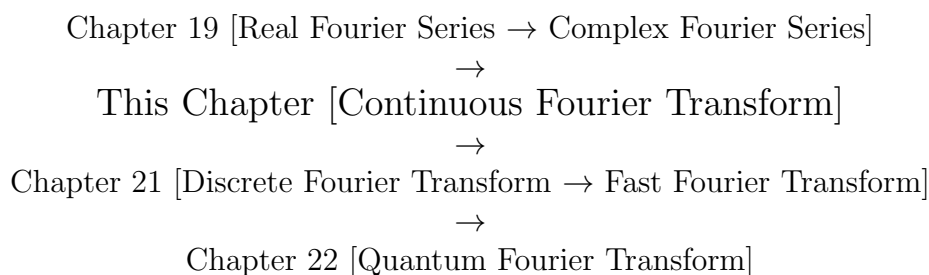
$$\omega \cdot T = 2\pi.$$

Chapter 20

The Continuous Fourier Transform

20.1 From Series Transform

This is the second of three classical chapters in Fourier theory meant to prepare you for the *quantum Fourier transform* or QFT . It fits into the full path according to:



In this lesson we learn how any reasonable function (not just periodic ones covered in the previous chapter) can be built using exponential functions, each having a specific frequency. The sums used in Fourier series turn into integrals this time around.

Ironically, we abandon the integrals after today and go back to sums since the QFT is a “finite” entity. However, the *continuous Fourier transform* of this lesson is correct foundation for the QFT , so if you can devote the time, it is good reading.

20.2 Motivation and Definitions

20.2.1 Non-Periodic Functions

Fourier series assumed (and required) that a function, f , was either *periodic* or restricted to a *bounded domain* before we could claim it was expressible as a weighted-sum of frequencies. A natural question to ask is whether we can find such weighted-sum expansions for non-periodic functions defined over *all* \mathbb{R} , with or without compact

support. Can *any* (well-enough-behaved-but-non-periodic) function be expressed as a sum of basis functions like e^{inx} ? There are two reasons to be hopeful.

1. Even a non-periodic function over \mathbb{R} can be considered to have bounded domain (and therefore be periodic) if we throw away the very distant “wings” (say $x < -10^{10000}$ and $x > 10^{10000}$). Therefore, restricted to this extremely large interval, at least, it is made up of a weighted-sum of basis functions, either sinusoids or e^{inx} . All we need to do is fix up those pesky two wings that are so far away.
2. The wild, fast wiggles or the slow, smooth curves of a graph are not exclusive to periodic functions; they are part of any function. We should be able to build them out of sines and cosines (or exponentials) whether they appear in periodic functions, functions with compact support or general functions over the real line.

The answer to this question is the *Fourier Transform* (\mathcal{FT}).

20.2.2 Main Result of Complex Fourier Series

We begin with a periodic f and restate the duality between it and its (complex) Fourier series,

$$\begin{aligned} f(x) &= \sum_{n=-\infty}^{\infty} c(n) e^{inx}, \\ \updownarrow \\ c(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-inx} dx. \end{aligned}$$

20.2.3 Tweaking the Complex Fourier Series

The price we’ll have to pay in order to express non-periodic functions as a weighted sum of frequencies is that the Fourier basis will no longer be a discrete set of functions, $\{e^{inx}\}_{n \in \mathbb{Z}}$, indexed by an integer, $n \in \mathbb{Z}$, and neither will their corresponding weights, $\{c_n\}_{n \in \mathbb{Z}}$ be discretely indexable. Instead, n will have to be replaced by a real number, s . This means $c(n)$ is going to be a full-fledged function of the real numbers, $c(s)$, $-\infty < s < \infty$.

The above formulas have to be modified in the following ways:

- The integer n must become a real number s .
- The sum will have to turn into an integral.
- The limits of integration have to be changed from $\pm\pi$ to $\pm\infty$.

- While not required, we make the formulas symmetric by replacing the normalization constant, $1/(2\pi)$ by $\sqrt{1/(2\pi)}$, and thus spreading that constant over both expressions.

These considerations suggest that if we want to express a non-periodic but well-behaved function f as a “sum” of frequencies, we would do so using

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} c(s) e^{isx} ds .$$

Since the sum is really an integral, the weights become a function over \mathbb{R} , $c(s)$, where each real s represents a frequency, and $c(s)$ is “how much” of that frequency the function f requires.

20.2.4 Definition of the Fourier Transform

This weighting function, $c(s)$, is computable from $f(x)$ using the companion formula,

$$c(s) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-isx} dx .$$

It is this last function of s that we call the *Fourier Transform*, or \mathcal{FT} , of the function $f(x)$, and it is usually denoted by the capital letter of the function we are transforming, in this case $F(s)$,

$$F(s) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-isx} dx .$$

Notation

We denote the Fourier transform operator using “ \mathcal{FT} ,” as in

$$\begin{aligned} F &= \mathcal{FT}(f), \\ f &\xrightarrow{\mathcal{FT}} F \end{aligned}$$

or, using the script notation \mathcal{F} ,

$$\begin{aligned} F &= \mathcal{F}(f), \\ f &\xrightarrow{\mathcal{F}} F . \end{aligned}$$

20.2.5 The Inverse Fourier Transform

We also consider the *inverse of the Fourier transform*, which allows us to recover f from F

$$\begin{aligned} f &= \mathcal{FT}^{-1}(F), \quad \text{or} \\ f &= \mathcal{F}^{-1}(F). \end{aligned}$$

If we look at our original motivation for the \mathcal{FT} , we see that $\mathcal{FT}^{-1}(F)$ is actually built-into that formula. Substitute $F(s)$ in for what we originally called it, $c(s)$, and we have the explicit form of the inverse Fourier transform,

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(s) e^{isx} ds .$$

The noteworthy items are:

- The $\mathcal{FT}^{-1}(F)$ and \mathcal{FT} differ only by a sign in the exponent.
- Alternate Definitions
 - We have defined the \mathcal{FT} to provide maximum symmetry. Many authors omit the factor of $\sqrt{1/(2\pi)}$ in the \mathcal{FT} at the cost of having the full factor $1/(2\pi)$ in the inverse.
 - There is a second common variant that puts a 2π into the exponent to make the result of certain computations look cleaner.
 - In a third variation, the exponent in the “forward” \mathcal{FT} is positive and the exponent in the inverse is negative.
 - Be prepared to see the alternate definitions of \mathcal{FT} and the correspondingly modified results they will yield, usually in the form of a constant factor change in our results.
- Sometimes, instead of using a capital letter F to represent $\mathcal{F}(f)$, authors will use the caret or tilde for that purpose,

$$\begin{aligned} \hat{f} &= \mathcal{F}(f), & \text{or} \\ \tilde{f} &= \mathcal{F}(f). \end{aligned}$$

20.2.6 Real vs. Complex, Even vs. Odd

Although we start with a real-valued function, f , there is no guarantee that F will be real-valued. After all, we are multiplying f by a complex valued function e^{-isx} prior to integration. However, there are situations in which F will be real, i.e., all the imaginary parts will cancel out during the integration. We’ll need a couple definitions.

Even Function. A function, f , is said to be **even** if it has the property

$$f(-x) = f(x).$$

A famously even function is $\cos x$.

Odd Function. A function, f , is said to be **odd** if it has the property

$$f(-x) = -f(x).$$

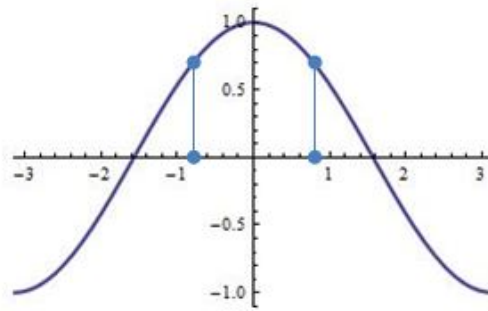


Figure 20.1: $\cos x$ is even

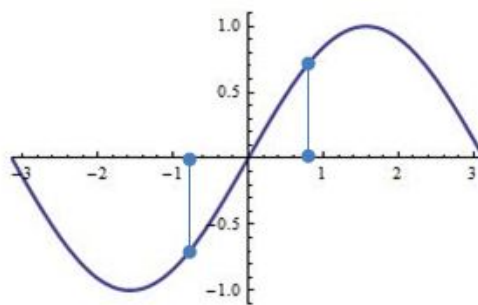


Figure 20.2: $\sin x$ is odd

A famously odd function is $\sin x$.

Theorem. *If f is **even**, F will be **real-valued**. If f is **odd**, F will be **purely imaginary-valued**. In all other cases, F will take on values that have both real and imaginary parts.*

The *domains* of both f and F are, in our context, always assumed to be real (x and s are both real). It is only the *values* of the functions f and F that we were concerned about.

Finally, you might notice that there was nothing in any of our definitions that required the original f be real-valued. In fact, as long as the domain is \mathbb{R} , the range can be $\subseteq \mathbb{C}$. We'll be using this fact implicitly as we consider the symmetric aspects of the spatial (or time) "domain" (i.e., where f lives) and the frequency "domain" (where F lives). In other words, we can start out with a complex-valued f and end up with another complex-valued F , or perhaps even a real-valued F . All combinations are welcome.

20.2.7 Conditions for a function to Possess a Fourier Transform

Unlike Fourier series, where practically any reasonable periodic function possessed a Fourier series, the same cannot be said of Fourier transforms. Since our function is

now “free-range” over all of \mathbb{R} , like a hyper-chicken in a billion acre ranch, it might go anywhere. We have to be circumspect when forming the integral, prepared that it might not even *converge*. One oft cited sufficient condition is that $f(x)$ be *absolutely integrable*, i.e.,

$$\int_{-\infty}^{\infty} |f(x)| \, dx < \infty.$$

As you can see, simple functions. like $f(x) = x$ or $f(x) = x^2 - x^3 + 1$ don’t pass the absolute-integrability test. We need functions that tend to zero strongly at both $\pm\infty$, like a pulse or wavelet that peters-out at both sides. Some pictures to help you visualize the graphs of square integrable functions are seen in figure 20.3. The main characteristic is that they peter-out towards $\pm\infty$.

(The functions that I’m claiming possess a Fourier transform are the $f_k(x) = \psi_k^2(x)$. While the (unsquared) ψ_k , seen on the left, might not be absolutely integrable, their squares, whose absolute values are pictured on the right, are.)

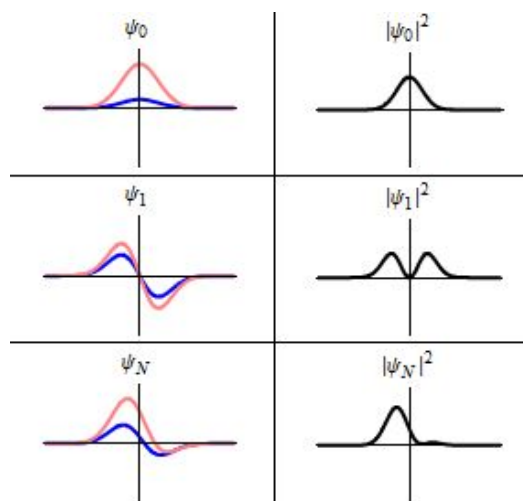


Figure 20.3: Square-integrable wavefunctions from Wikipedia StationaryStatesAnimation.gif, leading to the absolutely integrable $\psi_k^2(x)$

20.3 Learning to Compute

Let’s do a couple even functions, since they give real-valued Fourier transforms which are easy to graph.

20.3.1 Example 1: Rectangular Pulse

Compute the Fourier transform of

$$f(x) \equiv \begin{cases} 1, & |x| \leq .5 \\ 0, & \text{everywhere else} \end{cases}.$$

We plug it directly into the definition:

$$\begin{aligned} F(s) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-isx} dx \\ &= \frac{1}{\sqrt{2\pi}} \int_{-.5}^{+.5} e^{-isx} dx \\ &= \frac{1}{-is\sqrt{2\pi}} e^{-isx} \Big|_{-.5}^{+.5} = \frac{1}{-is\sqrt{2\pi}} (e^{-i(.5s)} - e^{i(.5s)}) \\ &= \sqrt{\frac{2}{\pi}} \cdot \frac{1}{s} \left(\frac{e^{i(.5s)} - e^{-i(.5s)}}{2i} \right) = \sqrt{\frac{2}{\pi}} \left(\frac{\sin .5s}{s} \right). \end{aligned}$$

That wasn't so bad. You can see both functions' graphs in Figure 20.4. They demonstrate that while f is restricted to a compact support, F requires the entire real line for its full definition. We'll see that this is no accident, and has profound consequences.

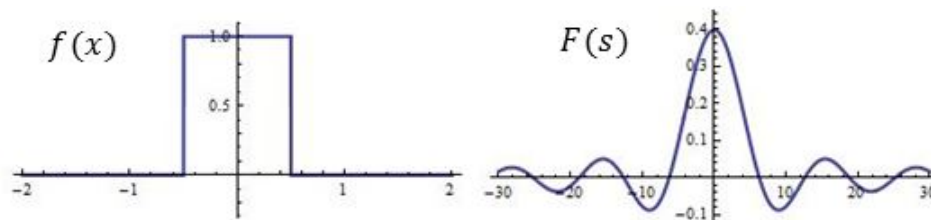


Figure 20.4: A simple function and its Fourier transform

20.3.2 Example 2: Gaussian

A *Gaussian* function, centered at the origin has the form

$$f(x) = N e^{-x^2/(2\sigma^2)}.$$

N is the height of its peak at the origin, and σ is called the *standard deviation* which conveys what percentage of the total area under f falls between $\pm k\sigma$, for $k = 1, 2, 3$, or any multiple we like. When $k = 3$, 99.7% of the area is covered. (Figure 20.5 demonstrates this.)

Its Fourier transform is

$$F(s) = \frac{N}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-x^2/(2\sigma^2)} e^{isx} dx.$$

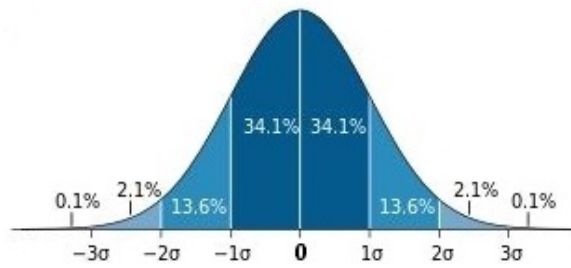


Figure 20.5: Interpretation of σ from Wikipedia Standard_deviation_diagram)

This integrand has the form

$$e^{-\alpha x^2 + \beta x},$$

which can be computed by completing the square and using a polar coordinate trick (look it up – it’s fun). The result is

$$F(s) = N\sigma e^{-s^2(\frac{\sigma^2}{2})},$$

which, if you look carefully, is another Gaussian, but now with a different height and standard deviation. The standard deviation is now $1/\sigma$, rather than σ . Loosely speaking, if the “spread” of f is wide, the “spread” of F is narrow, and vice versa.

20.4 Interlude: The Delta Function

20.4.1 Characterization of the Delta Function

An invaluable computational tool in the theory of Fourier transforms is the “impulse”, $\delta(x)$, a construct that (check your scrutiny at the door, please) behaves like a function of the real numbers that is zero everywhere except at the origin, where its value is ∞ . It also has the property that its integral over $-\varepsilon$ to $+\varepsilon$ (for any positive ε) is 1. In symbols,

$$\delta(x) = \begin{cases} \infty, & \text{if } x = 0 \\ 0, & \text{otherwise} \end{cases}$$

and

$$\int_{-\infty}^{\infty} \delta(x) dx = 1.$$

(Since δ is 0 away from the origin, the limits of integration can be chosen to be any interval that contains it.)

The *delta function* is also known as the “**Dirac delta function**,” after the physicist, Paul Dirac, who introduced it into the literature. It can’t be graphed, exactly, since it requires information not visible on a page, but it has a graphic representation as shown in figure 20.6.

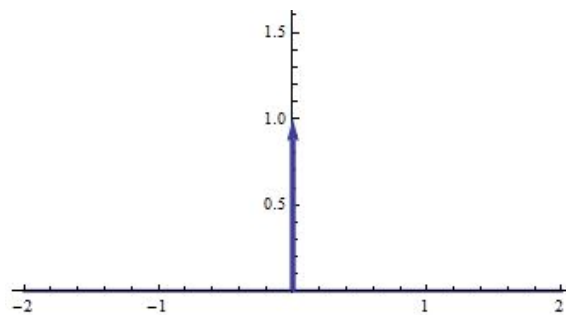


Figure 20.6: Graph of the Dirac delta function

20.4.2 The Delta Function as a Limit of Rectangles

There are many ways to make this notation rigorous, the simplest of which is to visualize $\delta(x)$ as the limit of a sequence of functions, $\{\delta_n(x)\}$, the n th “box” function defined by

$$\delta_n(x) = \begin{cases} n, & \text{if } x \in \left[-\frac{1}{2n}, \frac{1}{2n}\right] \\ 0, & \text{otherwise} \end{cases}.$$

Each $\delta_n(x)$ satisfies the integration requirement, and as $n \rightarrow \infty$, $\delta_n(x)$ becomes arbitrarily narrow and tall, maintaining its unit area all the while. (See Figure 20.7)

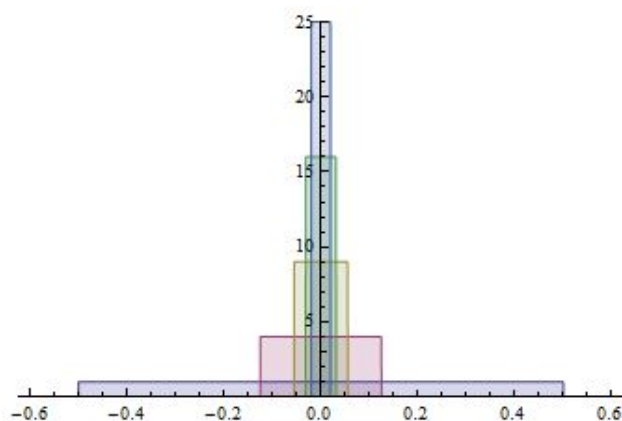


Figure 20.7: Sequence of box functions that approximate the delta function with increasing accuracy

And if we’re not too bothered by the imprecision of informality, we accept the definition

$$\delta(x) = \lim_{n \rightarrow \infty} \delta_n(x).$$

In fact, the converging family of functions $\{\delta_n(x)\}$ serves a dual purpose. In a computer we would select an N large enough to provide some desired level of accuracy and use $\delta_N(x)$ as an approximation for $\delta(x)$, thus creating a true function (no infinities involved) which can be used for computations, yet still has the properties we require.

20.4.3 The Delta Function as a Limit of Exponentials

While simple, the previous definition lacks “smoothness.” Any function that has unit area and is essentially zero away from the origin will work. A smoother family of Gaussian functions, indexed by a real parameter α ,

$$\delta_\alpha(x) = \sqrt{\frac{\alpha}{\pi}} e^{-\alpha x^2},$$

does the job nicely:

$$\delta(x) = \lim_{\alpha \rightarrow \infty} \delta_\alpha(x).$$

(See Figure 20.8)

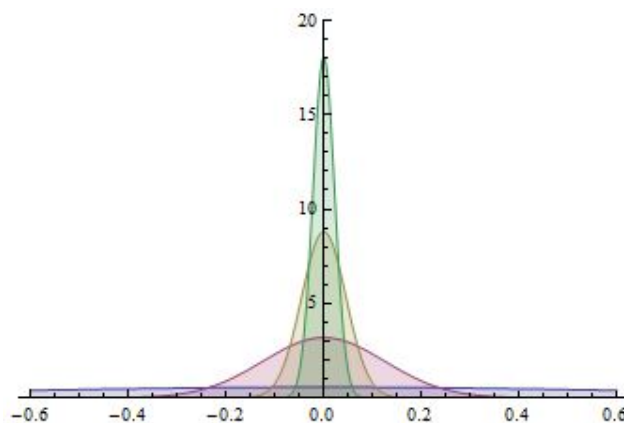


Figure 20.8: Sequence of smooth functions that approximate the delta function with increasing accuracy

Using the integration tricks mentioned earlier, you can confirm that this has the integration properties needed.

20.4.4 Sifting Property of the Delta Function

Among the many properties of the delta function is its ability to pick out (*sift*) an individual $f(x_0)$ of any function f at the domain point x_0 (no matter how mis-behaved f is),

$$f(x_0) = \int_{-\infty}^{\infty} f(x) \delta(x - x_0) dx ,$$

or (equivalently),

$$= \int_{x_0 - \varepsilon}^{x_0 + \varepsilon} f(x) \delta(x - x_0) dx .$$

You can prove this by doing the integration on the approximating sequence $\{\delta_n(x)\}$ and take the limit.

This sifting property is useful in its own right, but it also gives another way to express the delta function,

$$\delta(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{isx} ds .$$

It looks weird, I know, but you have all the tools to prove it. Here are the steps:

1. Compute $\mathcal{F}(\delta(x - x_0))$ with the help of the sifting property.
2. Take \mathcal{F}^{-1} of both sides.
3. Set $x_0 = 0$.

[**Exercise.** Show these steps explicitly.]

20.5 Fourier Transforms Involving $\delta(x)$

It turns out to be convenient to take Fourier transforms of functions that are *not absolutely integrable*. However, not only are such functions not guaranteed to have converging \mathcal{FT} integrals, the ones we want to transform in fact *do not* have converging \mathcal{FT} integrals. That's not going to stop us, though, because we just introduced a *non-function* function, $\delta(x)$, which will be at the "receiving end" when we start with a not-so-well-behaved f .

20.5.1 Example 3: A Constant

Consider the function $f(x) = 1$.

$$\left[\mathcal{F}[f(x) = 1] \right](s) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} 1 \cdot e^{-isx} dx = ?$$

The integral looks very much like an expression of the *delta function*. (Compare it to the last of our many definitions of $\delta(x)$, about half-a-page up). If the exponent did not have that minus sign, the integral would be *exactly* $2\pi\delta(s)$, making $? = \sqrt{2\pi}\delta(s)$. That suggests that we use integration by substitution, setting $x' = -x$ and wind up with an integral that *does* match this last expression of the delta function. That would work:

[**Exercise.** Try it.]

For an interesting alternative, let's simply *guess* that the minus sign in the exponent doesn't matter, implying the answer would still be $\sqrt{2\pi}\delta(s)$. We then test

our hypothesis by taking $\mathcal{F}^{-1} [\sqrt{2\pi} \delta(s)]$ and confirm that it gives us back $f(x) = 1$. Watch.

$$\begin{aligned} \left[\mathcal{F}^{-1} [\sqrt{2\pi} \delta(s)] \right] (s) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \sqrt{2\pi} \delta(s) e^{isx} ds \\ &= \int_{-\infty}^{\infty} \delta(s) e^{isx} ds \\ &= \int_{-\infty}^{\infty} \delta(s-0) e^{isx} ds = e^{i \cdot 0 \cdot x} = 1. \quad \checkmark \end{aligned}$$

The last line comes about by applying the sifting property of $\delta(s)$ to the function $f(s) = e^{isx}$ at the point $x_0 = 0$.

We have a Fourier transform pair

$$1 \xleftrightarrow{\mathcal{F}} \sqrt{2\pi} \delta(s),$$

and we could have started with a $\delta(x)$ in the spatial domain which would have given a constant in the frequency domain. (The *delta function* apparently has *equal amounts of all frequencies*.)

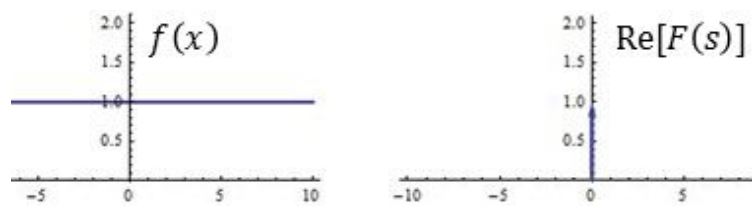


Figure 20.9: $\mathcal{F}[1]$ is a 0-centered delta function

Also, our intuition turned out to be correct: ignoring the minus sign in the exponent of e did not change the resulting integral, even when we left the integration boundaries alone; the delta function can be expressed with a negative sign in the exponent,

$$\delta(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-isx} ds.$$

20.5.2 Example 4: A Cosine

Next we try $f(x) = \cos x$. This is done by first solving *Euler's formula* for cosine, then using the definition of $\delta(x)$,

$$\begin{aligned}
 \left[\mathcal{F}[\cos x] \right](s) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \cos(x) e^{-isx} dx \\
 &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \left(\frac{e^{ix} + e^{-ix}}{2} \right) e^{-isx} dx \\
 &= \frac{1}{2\sqrt{2\pi}} \left(\int_{-\infty}^{\infty} e^{ix(1-s)} dx + \int_{-\infty}^{\infty} e^{-ix(1+s)} dx \right) \\
 &= \frac{1}{2\sqrt{2\pi}} \left(2\pi \delta(1-s) + 2\pi \delta(1+s) \right) \\
 &= \sqrt{\frac{\pi}{2}} \left(\delta(1-s) + \delta(1+s) \right).
 \end{aligned}$$

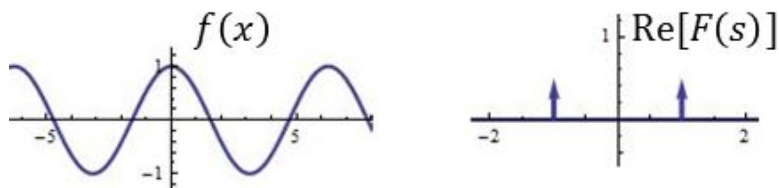


Figure 20.10: $\mathcal{F}[\cos x]$ is a pair of real-valued delta functions

20.5.3 Example 5: A Sine

$f(x) = \sin x$ can be derived exactly the same way that we did the cosine, only here we solve *Euler's formula* for sine, giving

$$\left[\mathcal{F}[\sin x] \right](s) = i \sqrt{\frac{\pi}{2}} \left(\delta(1+s) - \delta(1-s) \right).$$

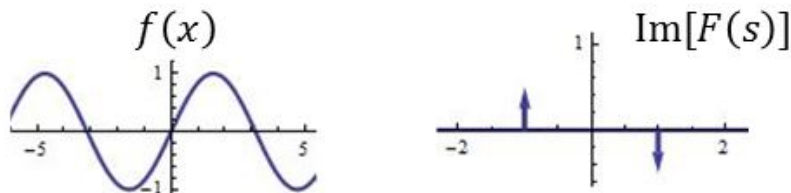


Figure 20.11: $\mathcal{F}[\sin x]$ is a pair of imaginary delta functions

Notice something interesting: this is the first time we see a Fourier transform that is not real (never mind that $\delta(s)$ isn't even a function ... we're inured to that by

now). What do you remember from the last few sections that would have predicted this result?

Does the \mathcal{FT} of $\sin x$ (or $\cos x$) make sense? It should. The spectrum of $\sin x$ needs only one frequency to represent it: $|s| = 1$. (We get two, of course, because there's an impulse at $s = -\pm 1$, but there's only one *magnitude*.) If we had done $\sin ax$ instead, we would have seen the impulses appear at $s = \pm a$, instead of ± 1 , which agrees with our intuition that $\sin ax$ requires only one frequency to represent it. The Fourier coefficients (weights) are zero everywhere except for $s = \pm a$.

(Use this reasoning to explain why a *constant function* has an $\mathcal{FT} =$ one impulse at 0.)

As an exercise, you can throw constants into any of the functions whose \mathcal{FT} s we computed, above. For example, try doing $A \sin(2\pi nx)$.

20.6 Properties of the Fourier Transform

There are some oft cited facts about the Fourier Transform that we present in this section. The first is one we'll need in this course, and the others are results that you'll probably use if you take courses in physics or engineering.

20.6.1 Translation Invariance

One aspect of the \mathcal{FT} we will find useful is its *shift-property*. For *real* α (the only kind of number that makes sense when f happens to be a function of \mathbb{R}),

$$f(x - \alpha) \xrightarrow{\mathcal{F}} e^{-i\alpha s} F(s).$$

This can be stated in various equivalent forms, two of which are

$$f(x + \alpha) \xrightarrow{\mathcal{F}} e^{i\alpha s} F(s), \quad \text{and} \\ F(s - \gamma) \xrightarrow{\mathcal{F}^{-1}} e^{i\gamma x} f(x),$$

where in the last version we *do* need to state that γ is real, since F generally is a function of \mathbb{C} .

If you “translate” (move five feet or delay by two seconds) the function in the spatial or time domain, it causes a benign *phase-shift* (by α) in the frequency domain. Seen in reverse, if you translate all the frequencies by a constant, this only multiplies the spatial or time signal by a unit vector in \mathbb{C} , again, something that can usually be ignored (although it may not make sense if you are only considering real f). This means that the translation in one domain has no measurable effect on the *magnitude* of the signal in the other.

This is a kind of *invariance*, because we don't care as much about $e^{ias} F(s)$ as we do its absolute-value-squared, and in that case

$$|e^{ias} F(s)|^2 = |e^{ias}|^2 |F(s)|^2 = |F(s)|^2.$$

Since we'll be calculating probabilities of quantum states and probabilities are the amplitudes' absolute-values-squared, this says that both $f(x)$ and $f(x + \alpha)$ have Fourier transforms which possess the same absolute-values and therefore the same probabilities. We'll use this in Shor's quantum period-finding algorithm.

20.6.2 Plancherel's Theorem

There's an interesting and crucial connection between f and F : the area between their graphs and their domain axes are equal. This result is called *Plancherel's Theorem*.

Plancherel's Theorem. *For any Fourier transform pair, F and f , we have*

$$\int_{-\infty}^{\infty} |f(x)|^2 dx = \int_{-\infty}^{\infty} |F(s)|^2 ds .$$

A picture demonstrating Plancherel's Theorem appears in Figure 20.12

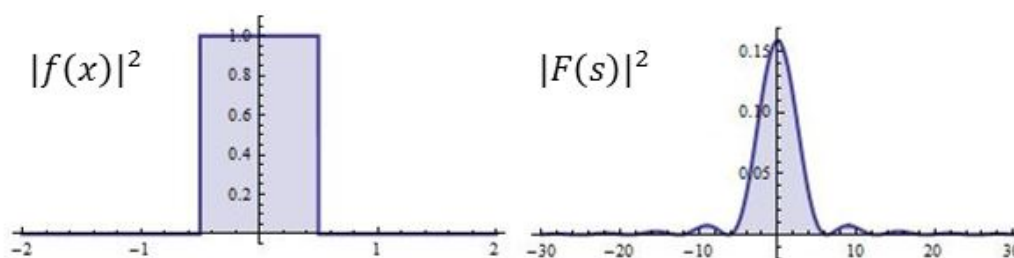


Figure 20.12: Area under $|f|^2$ and $|F|^2$ are equal (Plancherel's Theorem)

The reason this theorem is so important in quantum mechanics is that our functions are *amplitudes* of quantum states, and their squared absolute values are the *probability densities* of those states. Since we want any state to sum (integrate) to 1 over all space (“the particle must be somewhere”), we need to know that the Fourier transform does not disturb that property. Plancherel's theorem assures us of this fact.

20.6.3 Convolution

One last property that is heavily used in engineering, signal processing, math and physics is the *convolution theorem*.

A *convolution* is a binary operator on two functions that produces a third function. Say we have an *input signal* (maybe an image), f , and a *filter function* g that we want to *apply* to f . Think of g as anything you want to “do” to the signal. Do you want to reduce the salt-and-pepper noise in the image? There's a g for that. Do you want to make the image high contrast? There's a g for that. How about looking at only vertical edges (which a robot would care about when slewing its arms). There's

another g for that. The filter g is applied to the signal f to get the output signal which we denote $f * g$ and call the *convolution of f and g* : **The convolution of f and g , written $f * g$, is the function defined by**

$$[f * g](x) \equiv \int_{-\infty}^{\infty} f(\xi) g(x - \xi) d\xi.$$

The simplest filter to imagine is one that smooths out the rough edges (“noise”). This is done by replacing $f(x)$ with a function $h(x)$ which, at each x , is the *average* over some interval containing x , say ± 2 from x . With this idea, $h(10)$ would be

$$h(10) = K \int_8^{12} f(\xi) d\xi,$$

while $h(10.1)$ would be

$$h(10.1) = K \int_{8.1}^{12.1} f(\xi) d\xi.$$

(Here K is some normalizing constant like $1/(\text{sample interval})$.) If f consistently produced widely differing values between any x to its close neighbors (perhaps due to signal noise), $|f(10) - f(10.1)|$ could be quite large. But $|h(10) - h(10.1)|$ will be small since the two numbers are integrals over almost the same interval around $x = 10$. This is sometimes called a *running average* and is used to track financial markets by filtering out the moment-to-moment or day-to-day noise. Well this is nothing more than a convolution of f and g , where $g(x) = K$ for $|x| \leq \#DaysToAvg$, and 0, everywhere else (K often chosen to be $1/\#DaysToAvg$).

20.6.4 The Convolution Theorem

The *convolution theorem* tells us that rather than apply a convolution to two functions directly, we can get it by taking the ordinary point-by-point multiplication of their Fourier transforms, something that is actually easier and faster due to fast algorithms to compute transforms.

The Convolution Theorem.

$$f * g = \sqrt{2\pi} \mathcal{F}^{-1}[\mathcal{F}(f) \cdot \mathcal{F}(g)]$$

The constant $\sqrt{2\pi}$ would be replaced by a different constant if we were using one of the alternate definitions of the Fourier transform.

20.7 Period and Frequency in Fourier Transforms

Let’s take a moment to review the relationship between the *period* and *frequency* of the particularly pure periodic function $\sin nx$ for some integer n . For

$$\sin nx$$

the *period* is $T = 2\pi/n$ making the *angular frequency*

$$\omega = \frac{2\pi}{T} = n.$$

Well this is how we first introduced the idea of frequency informally at the start of this lecture: I just declared the n in $\sin nx$ to be a frequency and showed pictures suggesting that sines with large n squiggled more than those with small n .

The period-frequency relationship is demonstrated when we look at the *Fourier transform* of two sinusoids, one with period 2π and another with period $2\pi/3$. First, $\sin x$ and its spectrum are shown in figure 20.13 where it is clear that the frequency domain has only two non-zero frequencies at ± 1 .

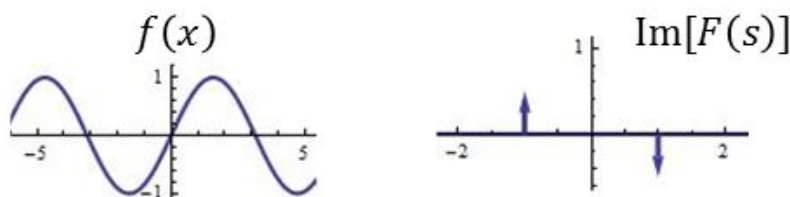


Figure 20.13: $\sin(x)$ and its spectrum

Next, $\sin 3x$ and its spectrum appear in figure 20.14 where the spectrum still has only two non-zero frequencies, but this time they appear at ± 3 .

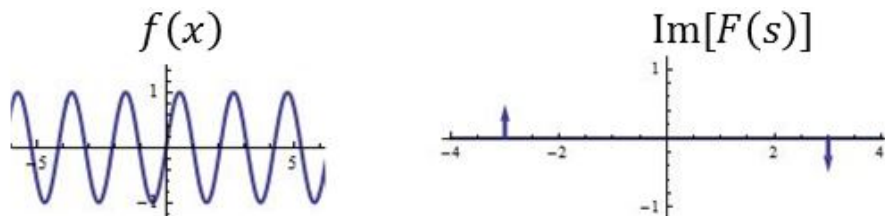


Figure 20.14: $\sin(3x)$ and its spectrum

Notice that both cases, when we consider ω to be the *absolute value* of the delta spike's position, we confirm the formula,

$$\omega T = 2\pi,$$

reminding us that if we know the period we know the (angular) frequency, and vice versa.

20.8 Applications

20.8.1 What's the \mathcal{FT} Used For?

The list of applications of the \mathcal{FT} is impressive. It ranges from cleaning up noisy audio signals to applying special filters in digital photography. It's used in communications,

circuit-building and numerical approximation. In wave and quantum physics, the Fourier transform is an alternate way to view a quantum state of a system.

In one example scenario, we seek to design an algorithm that will target certain frequencies of a picture, audio or signal, f . Rather than work directly on f , where it is unclear where the frequencies actually are, we take $F = \mathcal{F}(f)$. Now, we can isolate, remove, enhance the exact frequencies of $F(s)$ that interest us. After modifying F to our satisfaction, we recover $f = \mathcal{F}^{-1}(f)$.

A second application, dearer to quantum physicists, is the representation of states of a quantum system. We might model a system such as an electron zipping through a known magnetic field (but it could be a laser scattering off a crystal, the spin states of two entangled particles, or some other physical or theoretical apparatus). We prepare the system in a specific state – the electron is given a particular energy as it enters the field at time $t = 0$. This state is expressed by a wavefunction Ψ . If we did a good job modeling the system, Ψ tells us everything we could possibly know about the system at a specific time.

Now Ψ is essentially a vector in a Hilbert space, and like all vectors, it can be expressed in different bases depending on our interest.

- Position Basis - $\psi(\mathbf{x})$

When expressed in the *position basis*, it has one form, $\psi(\mathbf{x})$. This form reveals the amplitudes for the *position* of the electron at some moment in time; we take its magnitude-squared, $|\psi(\mathbf{x})|^2$, and integrate that over a region of \mathbf{x} -space to learn the likelihood that the electron's position, if measured, would be in that region.

- Momentum Basis - $\varphi(\mathbf{p})$

When expressed in the *momentum basis*, it has a different form, $\varphi(\mathbf{p})$. This form reveals the amplitudes for the *momentum* of the electron: we take its magnitude-squared, $|\varphi(\mathbf{p})|^2$, and integrate it over a region of \mathbf{p} -space to learn the likelihood that the electron's momentum, if measured, would fall in that region.

Both $\psi(\mathbf{x})$ and $\varphi(\mathbf{p})$ tell the exact same story, but from different points of view. Again, like any vector's coordinates, we can transform the coordinates to a different basis using a simple linear transformation. It so happens that the way one transforms the position representation to the momentum representation in quantum mechanics is by using the Fourier transform

$$\varphi(\mathbf{p}) = \mathcal{F}(\psi(\mathbf{x})).$$

In other words, moving between position space and momentum space is accomplished by applying the Fourier transform or its inverse.

20.8.2 The Uncertainty Principle

In quantum mechanics, every student studies a Gaussian *wave-packet*, which has the same form as that of our example, but with an interpretation: $f = \psi$ is a particle's position-state, with $\psi(x)$ being an *amplitude*. We have seen that the magnitude squared of the amplitudes tell us the probability that the particle has a certain position. So, if ψ represents a wave-packet in *position space*, then $|\psi(x)|^2$ reveals relative likelihoods that the particle is at position x . Meanwhile $\varphi = \mathcal{F}(\psi)$, as we learned a moment ago, is the same state in terms of *momentum*. If we want the probabilities for momentum, we would graph $\varphi(s)$.

(Figures 20.15 and 20.16 show two different wave-packet Gaussians after taking their absolute value-squared and likewise for their Fourier transforms.)

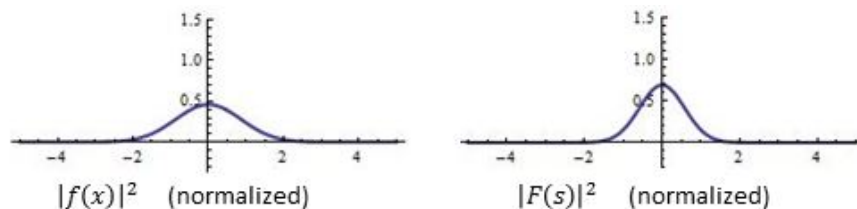


Figure 20.15: A normalized Gaussian with $\sigma^2 = 3$ and its Fourier transform

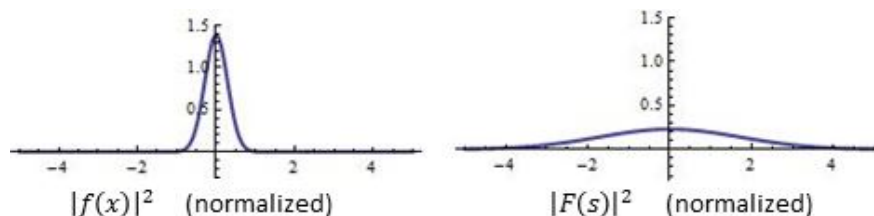


Figure 20.16: A more localized Gaussian with $\sigma^2 = 1/7$ and its Fourier transform

The second pair represents an initial narrower spread of its position state compared to the first. The uncertainty of its position has become smaller. But notice what happened to its Fourier transform, which is the probability density for momentum. Its uncertainty has become larger (a wider spread). As we set up the experiment to pin down its position, any measurement of its momentum will be less certain. You are looking at the actual mathematical expression of the Heisenberg uncertainty principle in the specific case where the two observables are position and momentum.

20.9 Summary

This is a mere fraction of the important techniques and properties of the Fourier transform, and I'd like to dig deeper, but we're on a mission. If you're interested, I recommend researching the *sampling theorem* and *Nyquist frequency*.

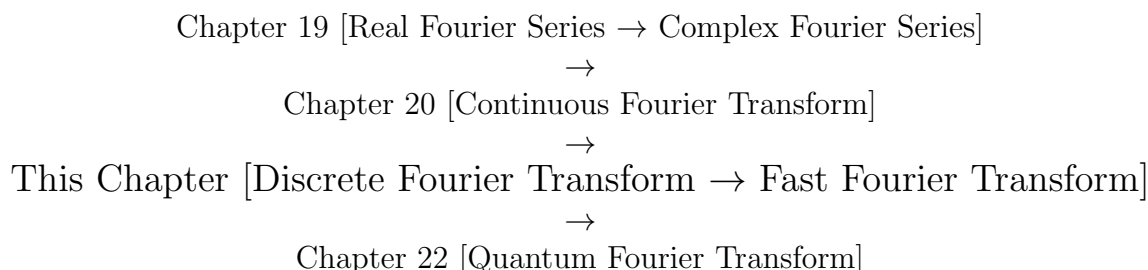
And with that, we wrap up our overview of the *classical Fourier series* and *Fourier transform*. Our next step in the ladder to \mathcal{QFT} is the *discrete Fourier transform*.

Chapter 21

The Discrete and Fast Fourier Transforms

21.1 From Continuous to Discrete

This is the last of three classical chapters in *Fourier theory* meant to prepare you for the *quantum Fourier transform*. It fits into the full path according to:



In this this lesson we turn the integrals of the last chapter back into nice finite sums. This is allowed because the functions we really care about will be defined on a finite set of integers, a property shared by the important functions that are the target of Shor's quantum period-finding and factoring algorithms.

After defining the *discrete Fourier transform*, or \mathcal{DFT} , we'll study a computational improvement called the *fast Fourier transform*, or \mathcal{FFT} . Even though the \mathcal{DFT} is what we really need in order to understand the *use of* the \mathcal{QFT} in our algorithms, the development of the recursion relations in the \mathcal{FFT} will be needed in the next chapter when we *design the quantum circuitry* that implements the \mathcal{QFT} .

21.2 Motivation and Definitions

21.2.1 Functions Mapping $\mathbb{Z}_N \rightarrow \mathbb{C}$

Sometimes Continuous Domains Don't Work

Instead of functions, $f : \mathbb{R} \rightarrow \mathbb{C}$, we turn to functions which are only defined on the finite set of integers $\mathbb{Z}_N = \{0, 1, 2, \dots, N-1\}$. This is motivated by two independent needs:

1. Computers, having a finite capacity, can't store data for continuous functions like $0.5x^3 - 1$, $\cos Nx$ or $Ae^{2\pi i n x}$. Instead, such functions must be sampled at discrete points, producing arrays of numbers that only approximate the original function. We are obliged to process these arrays, not the original functions.

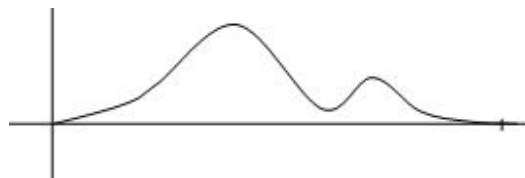


Figure 21.1: Continuous functions on a continuous domain

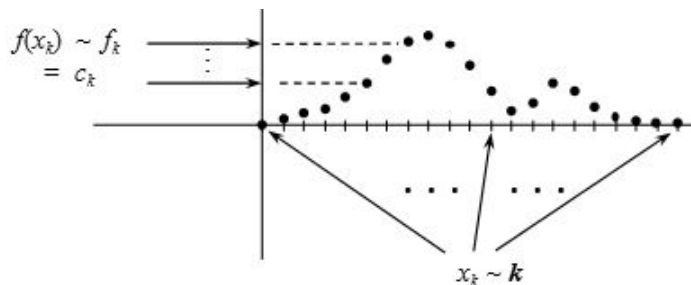


Figure 21.2: Sampling a continuous function at finitely many points

2. Most of the data we analyze don't *have* a function or formula behind them. They originate as discrete arrays of numbers, making a function of a continuous variable inapplicable.

Functions as Vectors

Although defined only on a finite set of integers, such function may still take real or complex values,

$$f : \mathbb{Z}_N \rightarrow \begin{cases} \mathbb{R} & \text{or} \\ \mathbb{C} \end{cases},$$

and we can convey all the information about a particular function using an array or vector,

$$f \longleftrightarrow \begin{pmatrix} f(0) \\ f(1) \\ \vdots \\ f(N-1) \end{pmatrix} \longleftrightarrow \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{pmatrix}.$$

In other words, the function, f can be viewed as a vector in \mathbb{C}^N (or possibly \mathbb{R}^N). You will see me switch freely between functional notation, $f(k)$, and vector notation, f_k or c_k . The vectors can be 2, 3, or N -dimensional. They may model a 2-D security cam image, a 3-D printer job or an N -dimensional quantum system.

Applicability of Complex Vectors

Since N -dimensional quantum systems are the stuff of this course, we'll need the vectors to be complex: *Quantum mechanics requires complex scalars in order to accurately model physical systems and create relative phase differences of superposition states.* Although our initial vector coordinates might be real (in fact, they may come from the tiny set $\{0, 1\}$), we'll still want to think of them as living in \mathbb{C} . Indeed, our operators and gates will convert such coordinates into complex numbers.

21.2.2 Defining the \mathcal{DFT}

The definitions and results of the classical \mathcal{FT} carry over nicely to the \mathcal{DFT} .

Primitive N th Roots are Central

We start by reviving our notation for the complex primitive N th root of unity,

$$\omega_N \equiv e^{2\pi i/N}.$$

(I say “the,” primitive N th root because in this course I only consider this one number to hold that title. It removes ambiguity and simplifies the discussion.) When clear from the context, we'll suppress the subscript N and simply use ω ,

$$\omega \equiv \omega_N.$$

From the primitive N th root, we generate all N of the roots (including 1, itself):

$$\begin{aligned} &\{1, \omega, \omega^2, \omega^3, \dots, \omega^{N-1}\} \\ &\text{or} \\ &\{1, e^{2\pi i/N}, e^{4\pi i/N}, e^{6\pi i/N}, \dots, e^{(N-1)2\pi i/N}\} \end{aligned}$$

These roots will be central to our definition of \mathcal{DFT} .

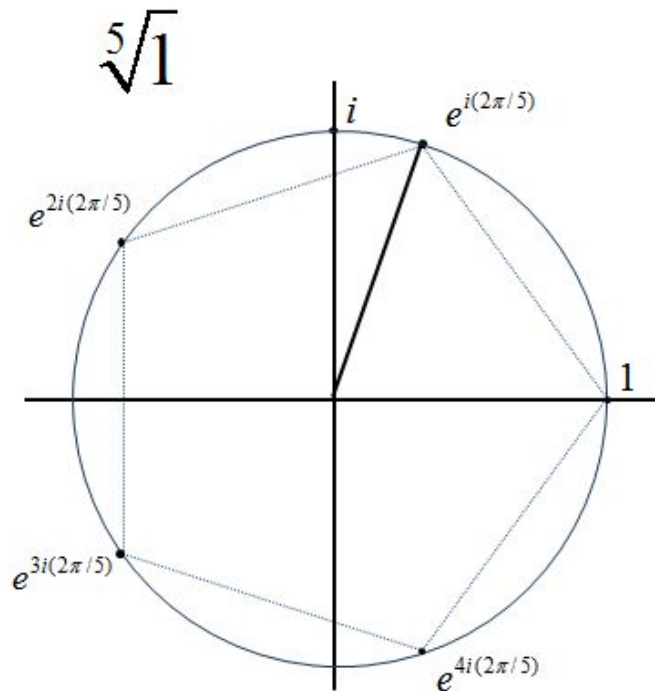


Figure 21.3: Primitive 5th root of 1 (the thick radius) and the four other 5th roots it generates

Recap of the Continuous Fourier Transform

Let's look again at the way the \mathcal{FT} maps mostly continuous functions to other mostly continuous functions. The \mathcal{FT} , also written \mathcal{F} , was defined as a map between functions,

$$\begin{aligned} F &= \mathcal{F}(f) \\ f &\xrightarrow{\mathcal{F}} F, \end{aligned}$$

which produced F from f using the formula

$$F(s) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-isx} dx.$$

It's easy to forget what the \mathcal{FT} is and think of it merely as the above formula, so I'll pester you by emphasizing the reason for this definition: we wanted to express $f(x)$ as a weighted-“sum” of frequencies, s , the weights being $F(s)$,

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} F(s) e^{isx} ds.$$

Adapting the Definition to the \mathcal{DFT}

To define the \mathcal{DFT} , we start with the \mathcal{FT} and make the following adjustments.

- The *integrals* will become *sums* from 0 to $N - 1$. Rather than evaluating f at real continuous x , we will be taking the N discrete values of the vector (f_k) and producing a complex spectrum vector (F_j) that also has N components.
- The factor of $1/\sqrt{2\pi}$ will be replaced by a normalizing factor $1/\sqrt{N}$. This is a choice, just like the specific flavor of \mathcal{FT} we used (e.g., the factor of $1/\sqrt{2\pi}$ which some authors omit). In quantum computing the choice is driven by our need for all vectors to live on the *projective sphere* in Hilbert space and therefore be normalized.
- We will use N th roots of unity, ω^{jk} , in place of the general exponential, e^{isx} . [You can skip the remainder of this bullet, without loss of crucial information, but for those wishing a more complete explanation, read on.]

Consider e^{-isx} to be a function of x which treats s as a constant frequency – a *continuous parameter* that nails down the function,

$$\varphi_s(x) = e^{-isx}.$$

Let's rewrite the spectrum, F , using the symbolism of this s -parameter family, $\varphi_s(x)$, in place of e^{-isx} ,

$$F(s) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) \varphi_s(x) dx.$$

In the discrete case, we want functions of the *index* k , each of whose constant frequency is parametrized by the *discrete parameter* j . The N th roots of unity provide the ideal surrogate for the continuously parametrized φ_s . To make the analogy to the \mathcal{FT} as true as possible, I will take the *negative* of all the roots' exponents (which produces the same N roots of unity, but in a different order), and define

$$\phi_j(k) = \omega_N^{-jk} = \omega^{-jk}.$$

The N functions $\{\phi_j\}_{j=0}^{N-1}$ replace the infinite $\{\varphi_s\}_{s \in \mathbb{R}}$. In other words, the continuous basis functions of x , e^{-isx} parametrized by s , become N vectors,

$$\mathbf{v}_j = \begin{pmatrix} v_{j0} \\ v_{j1} \\ \vdots \\ v_{jk} \\ \vdots \\ v_{j(N-1)} \end{pmatrix} = \begin{pmatrix} \omega^{-j0} \\ \omega^{-j1} \\ \vdots \\ \omega^{-jk} \\ \vdots \\ v_{-j(N-1)} \end{pmatrix}, \quad j = 0, 1, \dots, N-1$$

where k is the coordinate index and j is the parameter that labels each vector.

Official Definition of \mathcal{DFT}

And so we find ourselves compelled to define the *discrete Fourier transform*, or \mathcal{DFT} , as follows.

Definition of \mathcal{DFT} . An N th order \mathcal{DFT} is a function of an N -component vector $f = (f_k)$ that produces a new N -component vector $F = (F_j)$ described by the formula giving the output vector's N coordinates,

$$F_j \equiv \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \omega^{-jk}, \quad \text{for } j = 0, 1, \dots, N-1,$$

where $\omega = \omega_N$ is the primitive N th root of 1.

If we need to emphasize the order of the \mathcal{DFT} , we could use a parenthesized exponent, $\mathcal{DFT}^{(N)}$. However, when the common dimension of the input and output vectors is understood from the context, we usually omit exponent N as well as the phrase “ N th order” and just refer to it as the \mathcal{DFT} .

Notation

Common notational options that we'll use for the \mathcal{DFT} include

$$F = \mathcal{DFT}(f) = \mathcal{DFT}[f].$$

The j th coordinate of the output can be also expressed in various equivalent ways,

$$F_j = [\mathcal{DFT}(f)]_j = \mathcal{DFT}(f)_j = \mathcal{DFT}[f]_j.$$

The last two lack surrounding parentheses or brackets, but the subscript j still applies to the entire output vector, F .

Note on Alternate Definitions

Just as with the \mathcal{FT} , the \mathcal{DFT} has many variants all essentially equivalent but producing slightly different constants or minus signs in the results. In one case the forward \mathcal{DFT} has no factor of $1/\sqrt{2\pi}$, but the reverse \mathcal{DFT} contains a full $1/(2\pi)$. In another, the exponential is positive. To make things more confusing, a third version has a positive exponent of ω , but ω is defined to have the minus sign built-into it, so the overall definition is actually the same as ours. Be ready to see deviations as you perambulate the literature.

Inverse \mathcal{DFT}

Our expectation is that $\{F_j\}$ so defined will provide the *weighting factors* needed to make an expansion of f , as a weighted sum of the frequencies ω^j ,

$$f_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} F_j \omega^{kj},$$

valid. If the \mathcal{DFT} is to truly represent a discrete rendition of the continuous \mathcal{FT} , not only should this equality hold, but it should provide the definition of the *inverse transform*, \mathcal{DFT}^{-1} . Therefore, we must verify the formula, or stated differently, consider it to be the *definition* of \mathcal{DFT}^{-1} and prove that it is consistent with the \mathcal{DFT} in the sense that $\mathcal{DFT}^{-1} \circ \mathcal{DFT} = \mathbb{1}$.

(The “ -1 ” in \mathcal{DFT}^{-1} means *inverse* not *order*, the latter always appearing in parentheses. Of course, “order (-1) ” doesn’t make sense, anyway.)

Proof of Consistency of the Definitions. We substitute the definition of \mathcal{DFT} of a vector (f_k) into the expression of f as a weighted sum of F_j in the definition of \mathcal{DFT}^{-1} to see if we recover the original vector (f_k) . Let’s have some fun.

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} F_j \omega^{kj}$$

(replace F_j by its definition, being careful not to re-use the reserved index, k)

$$\begin{aligned} &= \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \left(\frac{1}{\sqrt{N}} \sum_{m=0}^{N-1} f_m \omega^{-jm} \right) \omega^{kj} \\ &= \frac{1}{N} \sum_{m=0}^{N-1} f_m \left(\sum_{j=0}^{N-1} \omega^{(k-m)j} \right) \end{aligned}$$

From exercise (d) of the section *Roots of Unity* (at the end of our *complex arithmetic lecture*) the sum in parentheses collapses to $N\delta_{km}$, so the double sum becomes

$$\frac{1}{N} \sum_{m=0}^{N-1} f_m (N\delta_{km}) = \frac{N}{N} f_k = f_k. \quad \text{QED}$$

21.2.3 $\mathcal{DFT}^{(N)}$ Evaluated Outside \mathbb{Z}_N

The definition of $\mathcal{DFT}^{(N)}$ tells us how to convert any input N -vector, f , to an output N -vector, F (f ’s “spectrum.”). We’ve talked about how such vectors are equivalent to functions of the N integers, $0, 1, \dots, N-1$. However, even though f ’s domain is limited to \mathbb{Z}_N , its spectrum, $F = \mathcal{DFT}^{(N)}[f]$, can actually be considered a function over all of \mathbb{Z} .

“We’ll that’s no big shocker,” you say? True, we already knew that any function on a bounded interval could be extended to all the reals, \mathbb{R} (if it’s continuous), or all

the integers, \mathbb{Z} (if it's discrete), just by declaring that its values outside the original bounded domain be induced by the periodicity relation

$$F(j + N) \equiv F(j).$$

Well, yes, but I mean something less preachy. Without invoking periodicity, we can show that this relation for a j outside the original N -vector has a natural meaning. In fact, just substitute a $j \geq N$ (or < 0) into the definition of $\mathcal{DFT}^{(N)}$ and we'll get the periodicity relation without having to force it on anyone. For simplicity, consider a j in the interval, $[N, 2N - 1]$, specifically, $j = N + p$, where $0 \leq p < N$.

$$\begin{aligned} F_j &= F_{N+p} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \omega^{-(N+p)k} \\ &= \frac{1}{\sqrt{N}} \omega^{-Nk} \sum_{k=0}^{N-1} f_k \omega^{-pk} \\ &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \omega^{-pk} = F_p. \end{aligned}$$

So we get this large-scale N -periodicity for free in the frequency domain, even though the original vector, f , is only defined on a finite set in the spatial (time) domain.

[Exercise. Where, in Fourier theory, have we seen this idea before?]

Another way to say this is that, for any j , we can evaluate the $F(j) = \mathcal{DFT}^{(N)}[f](j)$ by taking its value at (the non-negative) integer $(j \bmod N)$,

$$F_j = F_{(j \bmod N)},$$

which is the $(j \bmod N)$ th coordinate of the N -vector F . I only ask you to keep in mind that this is not a *definition* of F_j but a *consequence* of the definition of the \mathcal{DFT} .

21.3 Matrix Representation of the \mathcal{DFT}

If we let $\zeta \equiv \omega^{-1}$ (and remember, we are using the shorthand $\omega = \omega_N = e^{2\pi i/N}$), the following matrix, W , encodes the \mathcal{DFT} in a convenient way. Define W as

$$W = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \zeta & \zeta^2 & \zeta^3 & \cdots & \zeta^{N-1} \\ 1 & \zeta^2 & \zeta^4 & \zeta^6 & \cdots & \zeta^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \zeta^j & \zeta^{2j} & \zeta^{3j} & \cdots & \zeta^{(N-1)j} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \zeta^{N-1} & \zeta^{2(N-1)} & \zeta^{3(N-1)} & \cdots & \zeta^{(N-1)(N-1)} \end{pmatrix}.$$

Now, we can express \mathcal{DFT} of the vector (f_k) as

$$\mathcal{DFT}[f] = W \cdot (f_k).$$

To see this, just write it out long hand.

$$\frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \zeta & \zeta^2 & \zeta^3 & \cdots & \zeta^{N-1} \\ 1 & \zeta^2 & \zeta^4 & \zeta^6 & \cdots & \zeta^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \zeta^j & \zeta^{2j} & \zeta^{3j} & \cdots & \zeta^{(N-1)j} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & \zeta^{N-1} & \zeta^{2(N-1)} & \zeta^{3(N-1)} & \cdots & \zeta^{(N-1)(N-1)} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_k \\ \vdots \\ f_{N-1} \end{pmatrix}.$$

The j th component of this product is the sum

$$\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \zeta^{jk} = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \omega^{-jk},$$

which is just $\mathcal{DFT}[f]_j$.

21.4 Properties of \mathcal{DFT}

All the results for the continuous \mathcal{FT} carry over to the \mathcal{DFT} . (If you skipped the continuous \mathcal{FT} chapter, take these as definitions without worry.)

21.4.1 Convolution of Two Vectors

In vector form, *convolution* of two discrete functions, f and g , becomes

$$[f * g]_k \equiv \sum_{l=0}^{N-1} f_l g_{k-l}.$$

The *convolution theorem* for continuous \mathcal{FT} holds in the discrete case, where we still have a way to compute the (this time discrete) convolution by using \mathcal{DFT} :

$$f * g = \sqrt{2\pi} \mathcal{DFT}^{-1}[\mathcal{DFT}(f) \cdot \mathcal{DFT}(g)]$$

21.4.2 Translation Invariance (Shift Property) for Vectors

Translation invariance a.k.a. the *shift property* holds in the discrete case in the form of

$$f_{k-l} \xrightarrow{\mathcal{DFT}} \omega^{-lk} F_k.$$

As you can see, it's the same idea: a spatial or time *translation* in one domain (shifting the index by $-l$) corresponds to a *phase shift* (multiplication by a root of unity) in the other domain.

21.4.3 Computational Complexity of \mathcal{DFT}

The formula

$$[\mathcal{DFT}(f)]_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \omega^{-jk}$$

tells us that we have $\sim N$ complex terms to add and multiply for each resulting component F_j in the spectrum. Since the spectrum consists of N coordinates, that's N^2 complex operations, each of which consists of two or three real operations, so it's still $O(N^2)$ in terms of real operations. Furthermore, we often use fixed point (fixed precision) arithmetic in computers, making the real sums and products independent of the size, N , of the vector. Thus, the \mathcal{DFT} is $O(N^2)$, period.

You might argue that as N increases, so will the precision needed by each floating point multiplication or addition, which would require that we incorporate the number of digits of precision, m , into the growth estimate, causing it to be approximately $O(N^2 m^2)$. However, we'll stick with $O(N^2)$ and call this the time complexity *relative to the arithmetic operations*, i.e., “above” them. This is simpler, often correct and will give us a fair basis for comparison with the up-coming *fast Fourier transform* and *quantum Fourier transform*.

21.5 Period and Frequency in Discrete Fourier Transforms

A \mathcal{DFT} applies only to functions defined on the bounded integral domain of integers $\mathbb{Z}_N = \{0, 1, 2, \dots, N-1\}$. Within that interval of N domain numbers, if the function repeats itself many times – that is, if it has a period T , where $T \ll N$ (T is *much less* than N) – then the function would exhibit *periodicity* relative to the domain. That, in turn, implies it has an associated frequency, f . In the continuous cases, T and f are related by one of two common relationships, either

$$Tf = 1$$

or

$$Tf = 2\pi.$$

One could actually make the constant on the RHS different from 1 or 2π , although it's rarely done. But in the discrete case we *do* use a different constant, namely, N . We still have the periodicity condition that

$$f(k+T) = f(k),$$

for all k (when both k and $k+T$ are in the domain), but now the frequency is defined by

$$Tf = N.$$

If you were to compute the \mathcal{DFT} of a somewhat random vector like

```
( .1, .5, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .1, 0, .25, .3, 0, .6,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .3, 0, 0, .1, .35, .3, 0, .1,
  .1, .3, 0, .5, .45, .3, 0, .1,
  0, 0, 0, .25, .15, .4, .6, .6,
  0, 0, 0, .2, .15, .3, .6, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .4, 0, 1, .6, .15, .1, 0, .1,
  .3, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 2, .25, .75, .2, 0, .1,
  .1, 0, 0, .25, .05, .1, 0, .6,
  .2, 0, 0, .25, .0, .3, 0, .1,
  .1, .25, 0, .25, .15, .0, .6, .1,
  .1, 0, .5, .25, .0, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1 ) ,
```

you would see no dominant frequency, which agrees with the fact that the function is not periodic. (See figure 21.4)

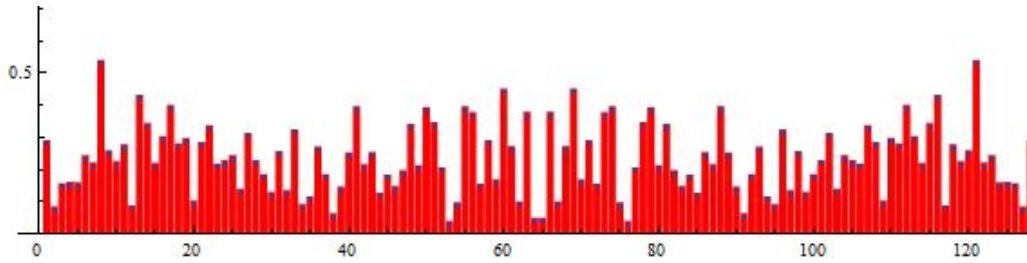


Figure 21.4: The \mathcal{DFT} of a non-periodic vector

On the other hand, a periodic vector, like

```
( .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1,
  .1, 0, 0, .25, .15, .3, 0, .1 )
```

exhibits a very strongly selective \mathcal{DFT} , as figure 21.5 shows.

A very *pure* periodic vector akin to the pure “basis functions” of the continuous cases, like $\sin nx$, $\cos nx$ and e^{ins} would be one like

```
( 0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0 )
```

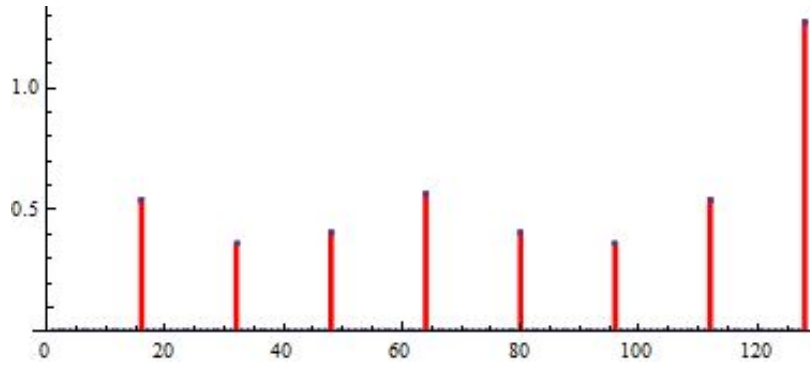


Figure 21.5: The spectrum of a periodic vector

```
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0,
0, 0, 0, .25, 0, 0, 0, 0 ) ,
```

and this one has a \mathcal{DFT} in which all the non-zero frequencies in the spectrum have the same amplitudes, as seen in figure 21.6.

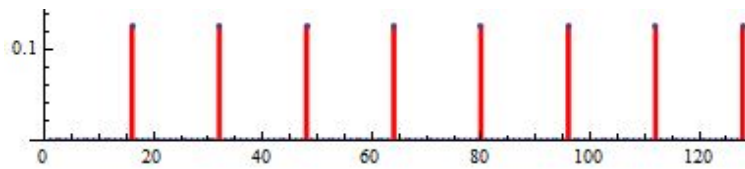


Figure 21.6: The spectrum of a very pure periodic vector

For all these examples I used $N = 128$, and for the two periodic cases, the period was $T = 8$ (look at the vectors), which would make the frequency $f = N/8 = 128/8 = 16$. You can see that all of the non-zero amplitudes in the *spectrum* (i.e., the \mathcal{DFT}) are multiples of 16: 16, 32, 48, etc. (There is a “phantom spike” at 128, but that’s beyond the end of the vector and merely an artifact of the graphing software, not really part of the spectrum.)

The thing to note here is that by looking at the spectrum we can see multiples of the frequency in the form cf , for $c = 1, 2, 3, \dots, 7$ and from that, deduce f and from f , get T .

21.6 A Cost-Benefit Preview of the \mathcal{FFT}

21.6.1 Benefit

Although merely a computational technique to speed up the \mathcal{DFT} , the *fast Fourier transform*, or \mathcal{FFT} , is well worthy of our attention. Among its accolades,

1. it is short and easy to derive,
2. it has wide application to circuit and algorithm design,
3. it improves the \mathcal{DFT} 's $O(N^2)$ to $O(N \log N)$, a speed up that changes the computational time of some large arrays from over an hour to a fraction of a second, and
4. the recursive nature of the solution sets the stage for studying the *quantum Fourier transform* (\mathcal{QFT}) circuit.

21.6.2 Cost

A slightly sticky requirement is that the \mathcal{FFT} only operates on vectors which have *exactly* $N = 2^n$ components, but there are easy work-arounds. One is to simply pad a deficient f with enough 0s to bring it up to the next power-of-two. For example, we might upgrade a 5-vector to an 8-vector like so:

$$\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \end{pmatrix} \longrightarrow \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_4 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Therefore, until further notice, we assume that $N = 2^n$, for some $n \geq 0$.

21.7 Recursive Equation for \mathcal{FFT}

The development of a truly fast \mathcal{FFT} proceeds in two stages. In this section, we develop a recursive relation for the algorithm. In the next section, we show how to turn that into a non-recursive, i.e., iterative, solution which is where the actual speed-up happens.

21.7.1 Splitting f into f^{even} and f^{odd}

Given a vector $f = (f_k)$ of dimension $N = 2^n$, divide it notationally into two vectors of size $N/2$ called f^{even} and f^{odd} , defined by

$$f^{\text{even}} \equiv \begin{pmatrix} f_0 \\ f_2 \\ f_4 \\ \vdots \\ f_{N/2} \end{pmatrix} \quad \text{and} \quad f^{\text{odd}} \equiv \begin{pmatrix} f_1 \\ f_3 \\ f_5 \\ \vdots \\ f_{N/2+1} \end{pmatrix}.$$

In terms of coordinates,

$$\begin{aligned} f_k^{\text{even}} &= f_{2k} \\ \text{and} \\ f_k^{\text{odd}} &= f_{2k+1}, \end{aligned}$$

for $k = 0, 1, \dots, N/2 - 1$. This leads to a string of equalities.

$$\begin{aligned} \mathcal{DFT}[f]_j &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} f_k \omega^{-jk} \\ &= \frac{1}{\sqrt{N}} \left(\sum_{k=0}^{\frac{N}{2}-1} f_k^{\text{even}} \omega^{-j(2k)} + \sum_{k=0}^{\frac{N}{2}-1} f_k^{\text{odd}} \omega^{-j(2k+1)} \right) \\ &= \frac{1}{\sqrt{N}} \left(\sum_{k=0}^{\frac{N}{2}-1} f_k^{\text{even}} \omega^{-j(2k)} + \omega^{-j} \sum_{k=0}^{\frac{N}{2}-1} f_k^{\text{odd}} \omega^{-j(2k)} \right) \\ &= \frac{1}{\sqrt{N}} \left(\sum_{k=0}^{\frac{N}{2}-1} f_k^{\text{even}} (\omega^2)^{-jk} + \omega^{-j} \sum_{k=0}^{\frac{N}{2}-1} f_k^{\text{odd}} (\omega^2)^{-jk} \right). \end{aligned}$$

We'll rewrite this to our advantage, next.

21.7.2 N th Order \mathcal{DFT} in Terms of Two $(N/2)$ th Order \mathcal{DFT} s

Now, $\omega = \omega_N$ was our primitive N th root of unity, so ω^2 is an $(N/2)$ th root of unity (e.g., $(\sqrt[8]{a})^2 = \sqrt[4]{a}$). If we rewrite the final sum using ω' for ω^2 , N' for $N/2$ and labeling the ω outside the odd sum as an N th root, things look very interesting:

$$\mathcal{DFT}[f]_j = \frac{1}{\sqrt{2}} \left(\frac{1}{\sqrt{N'}} \sum_{k=0}^{N'-1} f_k^{\text{even}} (\omega')^{-jk} + \omega_N^{-j} \frac{1}{\sqrt{N'}} \sum_{k=0}^{N'-1} f_k^{\text{odd}} (\omega')^{-jk} \right)$$

We recognize each sum on the RHS as an order $N/2$ \mathcal{DFT} , so let's go ahead and label them as such, using an exponent label to help identify the orders. We get

$$\mathcal{DFT}^{(N)}[f]_j = \frac{1}{\sqrt{2}} \left(\mathcal{DFT}^{(N/2)}[f^{\text{even}}]_j + \omega_N^{-j} \cdot \mathcal{DFT}^{(N/2)}[f^{\text{odd}}]_j \right).$$

Since the j on the LHS can go from $0 \rightarrow (N-1)$, while both \mathcal{DFT} s on the RHS are only size $N/2$, we have to remind ourselves that a \mathcal{DFT} applied to any vector is a function whose domain can be extended to all integers, \mathbb{Z} , if we wish. We saw that one way to get $\mathcal{DFT}^{(N/2)}[f]$'s value at those “outer” j s is to first move j back into the interval $[0, N/2-1]$ using *modulo* arithmetic. We'll add that detail for utter clarity:

$$\begin{aligned} \mathcal{DFT}^{(N)}[f]_j = \frac{1}{\sqrt{2}} \left(\mathcal{DFT}^{(N/2)}[f^{\text{even}}]_{(j \bmod N/2)} \right. \\ \left. + \omega_N^{-j} \cdot \mathcal{DFT}^{(N/2)}[f^{\text{odd}}]_{(j \bmod N/2)} \right) \end{aligned}$$

Finally, let's clear the smoke using shorthand like $F^{(N)} = \mathcal{DFT}^{(N)}(f)$, $F_E = F^{\text{even}}$ and $F_O = F^{\text{odd}}$. The final form is due to Danielson and Lanczos and dates back to 1942.

21.7.3 Danielson-Lanczos Recursion Relation

$$[F^{(N)}]_j = \frac{1}{\sqrt{2}} \left([F_E^{(N/2)}]_{(j \bmod N/2)} + \omega_N^{-j} \cdot [F_O^{(N/2)}]_{(j \bmod N/2)} \right)$$

We have reduced the computation of a size N \mathcal{DFT} to that of two size $(N/2)$ \mathcal{DFT} s (and a *constant time* multiplication and addition). Because we can do this all the way down to a size 1 \mathcal{DFT} (which is just the identity operation – check it out), we are able to compute F_j in $\log N$ iterations, each one a small, fixed number of complex additions and multiplications.

We're Not Quite There Yet

This is promising: We have to compute N output coordinates, and Danielson-Lanczos tells us that we can get each one using, what appears to be $\log N$ operations, so it seems like we have an $O(N \log N)$ algorithm.

Not so fast (literally and figuratively).

1. The cost of partitioning f into f^{even} and f^{odd} , unfortunately, does require that we run through the full array at each recursion level, so that's a deal breaker.
2. We can fix the above by passing the array “in-place” and just adding a couple parameters, START and GAP, down each recursion level. This obviates the need

to partition the arrays, but, each time we compute a single output value, we still end up accessing and adding all N of the original elements (do a little example to compute F_3 for an 8-element f).

3. Recursion has its costs, as any computer science student well knows, and there are many internal expenses that can ruin our efficiency even if we manage to fix the above two items yet refuse to abandon recursion.

In fact, it took some 20 years before someone (Tukey and Cooley get the credit) figured out how to leverage this recursion relation to break the $O(N^2)$ barrier.

21.7.4 Code Samples: Recursive Algorithm

Before we present a non-recursive algorithm based on Tukey and Cooley's work, we should confirm that the recursive algorithm actually works and has expected time complexity, N^2 .

We'll use an object-oriented (OOP) solution written in C++ whose main utility class will be `FftUtil`. A single `FftUtil` object will have its own \mathcal{FFT} size, provided either at instantiation (by a *constructor*) or during mid-life (by an *instance method* `setSize()`). Once a size is established all the roots of unity are stored inside the object so that subsequent computations using that same object do not require root calculations. We'll have the obvious accessors, mutators and computational methods in the class, as demonstrated by a simple client, shown next. Because many of the aspects of the class are independent of the details of the computational algorithm, we can use this class for the up-coming non-recursive, $N \log N$, solution.

Not all the capabilities of the class will be demonstrated, but you'll get the idea.

First, the client's view.

A Client's View of Class `FftUtil` which Utilizes the Recursive \mathcal{FFT}

```
// client's use of the classes FftUtil and Complex
// to compute a recursive-style FFT

// client -----
int main()
{
    const int FFT_SIZE = 8;
    int k;
    bool const TIMING = false;
    Complex *a = new Complex[FFT_SIZE];

    // load input array a
    for ( k = 0; k < FFT_SIZE; k++ )
        a[k] = k * .1;

    // instantiate an fft object
    FftUtil fft(FFT_SIZE);

    // load the object's input signal with a[]
    fft.setInSig(a, FFT_SIZE);
```

```

// confirm that the object has the right input
cout << "IN SIGNAL (recursive) ----- " << endl;
cout << fft.toStringIn() << endl;

// instruct object to compute its FFT using recursion
fft.calcFftRecursive();

// display the object's output
cout << "FFT OUT (recursive) ----- " << endl;
cout << fft.toStringOut() << endl;

// release dynamic memory
delete[] a;
}

```

The client is invoking `setInSig()` and `toStringOut()` to transfer the signals between it and object and also calling `calcFftRecursive()` to do the actual *DFT* computation. The client also uses a simple eight-element input signal for testing,

$$\{f_k\}_{k=0}^7 \equiv \{0, .1, .2, .3, .4, .5, .6, .7\}.$$

Although the class proves to be only a slow $O(N^2)$ solution, we should not shrug-off its details; as computer scientists, we need to have reliable benchmarks for future comparison and proof-of-correctness coding runs. Thus, we want to look inside class `FftUtil` and then test it.

The publicly exposed `calcFftRecursive()` called by the client leans on a private helper not seen by the client: `calcFftRecursive()`. First the definition of the public member method:

```

// public method that client uses to request FFT computation
// assumes signal is loaded into private array inSig[]

bool FftUtil::calcFftRecursive()
{
    int k;

    // check for fatal allocation errors
    if (inSig == NULL || outSig == NULL)
        return false;

    // calculate FFT(k) for each k using recursive helper method
    for ( k = 0; k < fftSize; k++ )
        outSig[k] = (1/sqrt(fftSize))
            * calcFftRecWorker(inSig, k,  fftSize,  1, 1 );

    return true;
}

```

Here's the private recursive helper which you should compare with the Danielson-Lanczos relation. It is a direct implementation which emerges naturally from that formula.

```

// private recursive method that does the work
// a:      array start location
// gap:     interval between consec a[] elements (so we can pass array in-place)
// size:    size of a[] in current iteration
// rootPos: index in the (member) roots[] array where current omega stored
// j:       output index FFT we are computing
// reverse: true if doing an inverse FFT

```

```

Complex fftFHC(int n, Complex a[], Complex roots[],
               int j, int size, int rootPos, int gap, bool reverse )
{
    Complex retVal, even, odd;
    int arrayPos, sizeNxtLevel, rootPosNxtLevel, gapNxtLev;

    // DFT of single element is itself
    if ( size == 1 )
        return a[j * gap]; // since using orig arraym need gap

    // locals used for clarity
    sizeNxtLevel = size >> 1;
    rootPosNxtLevel = rootPos << 1;
    gapNxtLev = gap << 1;

    // two recursive calls
    even = fftFHC( n, a, roots, j % sizeNxtLevel, sizeNxtLevel,
                  rootPosNxtLevel, gapNxtLev, reverse );
    odd = fftFHC( n, a + gap, roots, j % sizeNxtLevel, sizeNxtLevel,
                  rootPosNxtLevel, gapNxtLev, reverse );

    // put the even and odd results together to produce return for current call
    // (inverse FFT wants positive exponent, forward wants negative)
    if (reverse)
        arrayPos = (j * rootPos) % n ; // j * omega
    else
        arrayPos = (j * (n - rootPos)) % n ; // -j * omega

    retVal = even + roots[arrayPos] * odd;

    return retVal;
}

```

I'll let you analyze the code to show that it does predict $O(N^2)$ *big-O* timing, but we will verify it using benchmarks in the next few lines.

The output confirms that we are getting the correct values.

```

/* ===== sample run =====

IN SIGNAL (recursive) -----
0 0.1 0.2 0.3
0.4 0.5 0.6 0.7

FFT OUT (recursive) -----
0.989949 -0.141421+0.341421i -0.141421+0.141421i -0.141421+0.0585786i
-0.141421 -0.141421-0.0585786i -0.141421-0.141421i -0.141421-0.341421i

Press any key to continue . . .

===== */

```

And we loop through different input-array sizes to see how the time complexity shakes out:

```

FFT size 1024 Recursive FFT: 0.015 seconds.
FFT size 2048 Recursive FFT: 0.066 seconds.
FFT size 4096 Recursive FFT: 0.259 seconds.
FFT size 8192 Recursive FFT: 1.014 seconds.
FFT size 16384 Recursive FFT: 4.076 seconds.

```

The pattern is unmistakable: doubling the array size causes the time to grow four-fold. This is classic N^2 time complexity. Besides the growth rate, the absolute times

(four seconds for a modest sized signal) are unacceptable.

21.8 A Non-Recursive, $N \log N$ Solution that Defines the \mathcal{FFT}

Strictly speaking the above code is not an \mathcal{FFT} since it is, frankly, just *not that fast*. A true \mathcal{FFT} is an algorithm that produces $N \log N$ performance based on non-recursive techniques.

We now study the improved \mathcal{FFT} algorithm which consists of two main phases, *bit-reversal* and *iterative array-building*. We'll gain insight into these two sub-algorithms by previewing the very short high-level \mathcal{FFT} code that invokes them.

21.8.1 The High-Level \mathcal{FFT} Method

To class `FftUtil`, we add *public* and *private* instance methods that will compute the \mathcal{DFT} using a non-recursive algorithm.

[**Coding Note.** My OOP students would expect me to call the above class `DftUtil` and derive an `FftUtil` subclass from it (rather than insert new methods directly into the original class). However, we'll avoid the extra notation demanded by *derived classes* for this example.]

The highest level public method will be the (forward) \mathcal{FFT} , called `calcFft()`. It consists of three method calls:

1. **Bit Reversal.** The first implements bit-reversal by creating a new “indexing window” into our data. That is, it generates a *utility array* of indexes that will help us reorder the input signal. The new array is independent of the input signal, but it does depend on the \mathcal{FFT} size, N . The effect is the virtual reordering the input signal.
2. **Iterative Array Building.** The second applies an iterative algorithm that builds up from the newly ordered input array and, ultimately, replaces it by the output \mathcal{DFT} .
3. **Normalization.** Finally, we multiply the result by a normalizing factor.

Here's a bird's eye view of the method.

```
// public method that client uses to request FFT computation
// assumes signal is loaded into private array inSig[]
bool FftUtil::calcFft()
{
    // throwing exception or single up-front test has slight appeal, but
    // following is safer for future changes to constituent methods

    if ( !copyInSigToAuxSigBitRev() )
        return false;
```

```

    if ( !combineEvenOdd(false) )    // false = forward FFT
        return false;

    if ( !normalize() )
        return false;

    return true;
}

```

21.8.2 Bit-Reversal

The big break in our quest for speed comes when we recognize that the recursive algorithm leads – at its deepest nested level – to many tiny *order-one* arrays, and that happens after $\log N$ method calls. This is the end of the recursion at which point we compute each of these order-one \mathcal{DFT} s, manually. It’s the infamous “escape valve” of recursion. But computing the \mathcal{DFT} of those size-one arrays is trivial:

$$\mathcal{DFT}^{(1)}(\{c\}) = \{c\},$$

that is, the \mathcal{DFT} of any *single element* array is itself (apply the definition). So we don’t really have to go all the way down to that level - there’s nothing to do there. (The *size one* \mathcal{DFT} s are already done, even if they are in a mixed up order in our input signal.) Instead we can halt recursion when we have *size two* arrays, at which point we compute the *order two* \mathcal{DFT} s explicitly. Take a look:

$$\left[F_{EEOE...OE}^{(2)} \right]_j = \frac{1}{\sqrt{2}} (f_p + (-1)^{-j} \cdot f_q),$$

for some p and q , gives us the j th component of the *size two* \mathcal{DFT} s. $F_{EEOE...OE}^{(2)}$ represents one of the many *order two* \mathcal{DFT} s that result from the recursion relation by taking increasingly smaller *even* and *odd* sub-arrays in our recursive descent from size N down to size two.

The Plan: Knowing that the first order \mathcal{DFT} s are just the original input signal’s array elements (whose exact positions are unclear at the moment), our plan is work *not* from the top down, recursively, but from the bottom with the original array values, and build-up from there. In other words, instead of *recursing down* from size N to size one, we *iterate up* from size one to size N . To do that we need to get the original signal, $\{f_k\}$, in the right order in preparation for this rebuilding process.

So our first task is to re-order the input array so that every pair f_p and f_q that we want to combine to get a *size two* \mathcal{DFT} end up next to one another. While at it, we’ll make sure that once they are computed, all *size two* pairs which need to be combined to get the fourth order \mathcal{DFT} s will *also* be adjacent, and so on. This reordering is called *bit-reversal*. The reason for that name will be apparent shortly.

Let’s start with an input signal of size $8 = 2^3$ that we wish to transform and define it such that it’ll be easy to track:

$$\{f_k\}_{k=0}^7 \equiv \{0, .1, .2, .3, .4, .5, .6, .7\}.$$

We start at the top and, using the Danielson-Lanczos recursion relation, see how the original f decomposes into two four-element even-odd sets, then four two-element even-odd sets, and finally eight singleton sets. Figure 21.7 shows how we separate the original eight-element $f^{(8)}$ into $f_E^{(4)}$ and $f_O^{(4)}$, each of length 4.

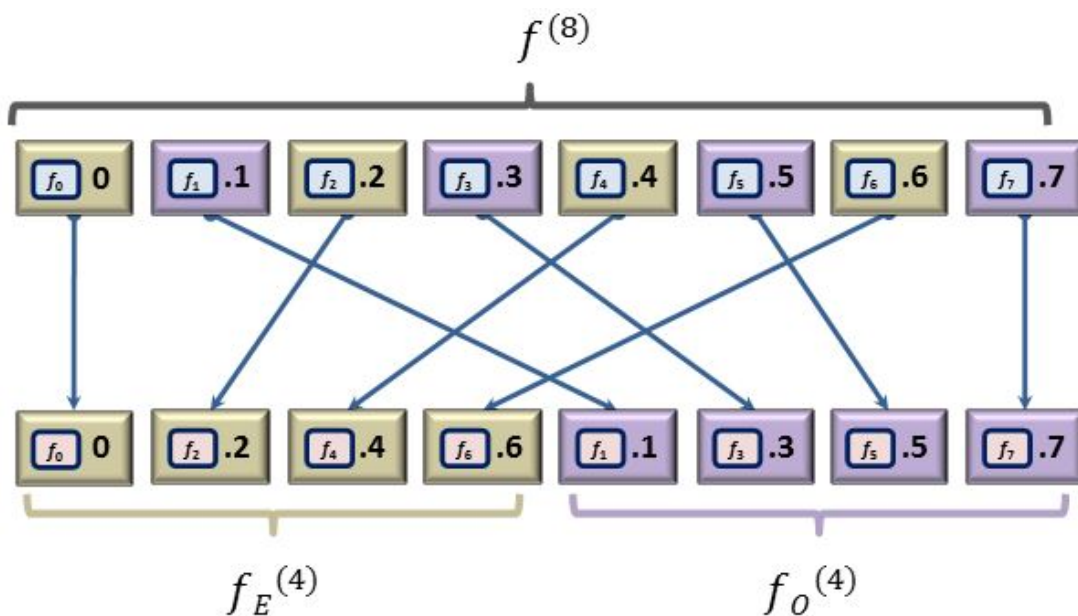


Figure 21.7: Going from an 8-element array to two 4-element arrays (one even and one odd)

I'll select one of these, $f_E^{(4)}$, for further processing (figure 21.8).

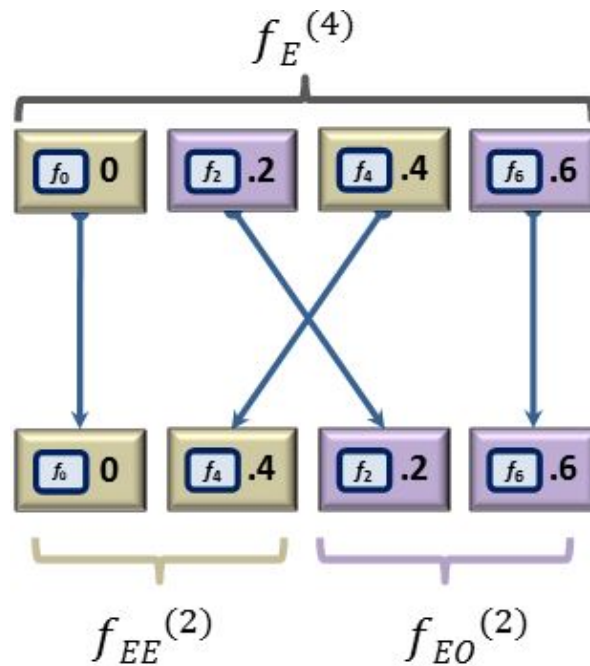


Figure 21.8: Decomposing the even 4-element array to two 2-element arrays (one even and one odd)

This time, for variety, we'll recurse on the odd sub-array, $f_{EO}^{(2)}$ (figure 21.9).

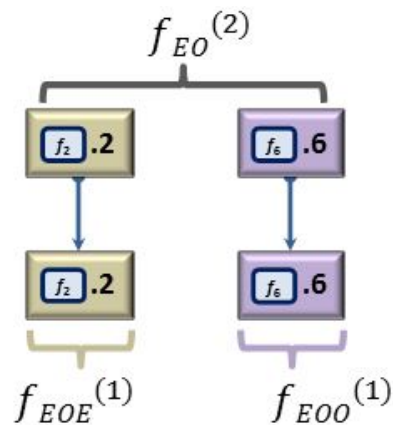


Figure 21.9: Breaking a two-element array into two singletons

Remember, our goal is to re-order the input array to reposition pairs such as these next to one-another. The last picture told us that we want to see f_2 repositioned so it is next to f_6 (figure 21.10).



Figure 21.10: Final positions of f_2 should be next to f_4 .

Once this happens, we are at the bottom of recursion. These one-element arrays are *their own* \mathcal{DFT} s, and as such, are neither even nor odd. Each singleton is some F_0 (more accurately, $[F_{EOE}]_0$ and $[F_{EOO}]_0$).

Aside: Recall that we needed to take $j \pmod{N/2}$ at each level, but when $N/2$ is 1, anything mod 1 is 0, which is why we are correct in calling these F_0 for different one-element F arrays (figure 21.11).



Figure 21.11: Singletons are their own \mathcal{DFT} s

If we had taken a different path, we'd have gotten a different pair. Doing it for all possibilities would reveal that we would want the original eight values paired as follows:

$$\begin{aligned} f_2 &\longleftrightarrow f_6 \\ f_3 &\longleftrightarrow f_7 \\ f_0 &\longleftrightarrow f_4 \\ f_1 &\longleftrightarrow f_5 \end{aligned}$$

Now, we want more than just adjacent pairs; we'd like the two-element \mathcal{DFT} s that they generate to also be next to one another. Each of these pairs has to be positioned properly with respect to the rest. Now is the time for us to stand on the shoulders of the giants who came before and write down the full ordering we seek. This is shown in figure 21.12. (Confirm that the above pairs are adjacent).

What you are looking at in figure 21.12 is the bit-reversal arrangement. It's so named because in order to get f_6 , say, into its correct position for transform building, we “reverse the bits” of the integer index 6 (relative to the size of the overall transform, 8, which is three bits). It's easier to see than say: $6 = 110$ reversed is $011 = 3$, and indeed you will find the original $f_6 = .6$ ends up in position 3 of the bit-reversed,

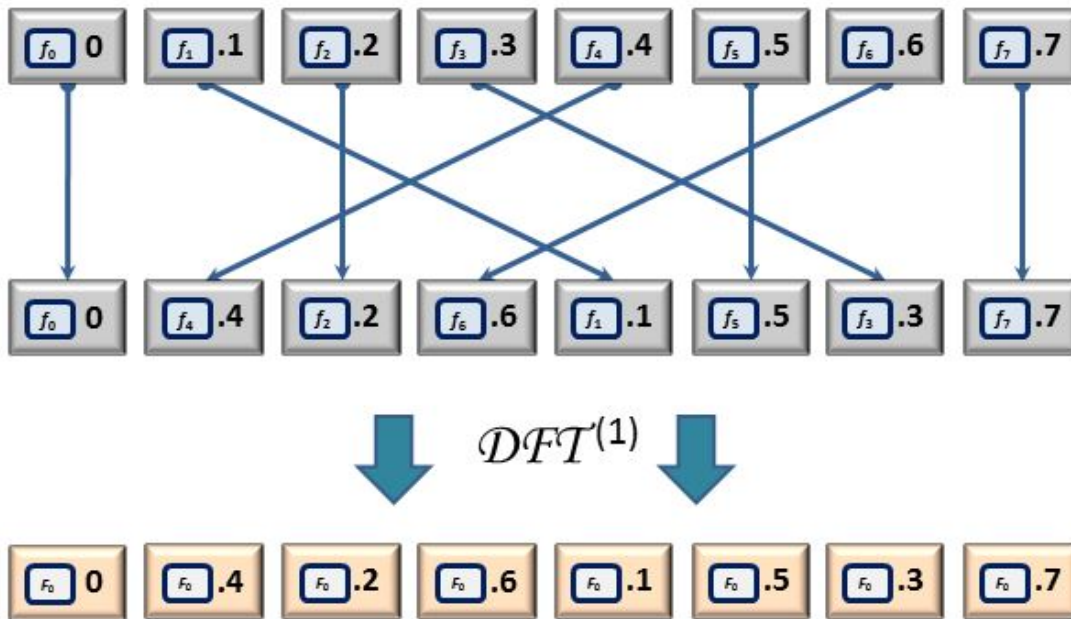


Figure 21.12: Bit-reversal reorders an eight-element array

array. On the other hand, $5 = 101$ is its own bit reversed index, so it should – and does – not change array positions.

Bit Reversal Mapping for 3-bit Integers

0	=	000	\mapsto	000	=	0
1	=	001	\mapsto	100	=	4
2	=	010	\mapsto	010	=	2
3	=	011	\mapsto	110	=	6
4	=	100	\mapsto	001	=	1
5	=	101	\mapsto	101	=	5
6	=	110	\mapsto	011	=	3
7	=	111	\mapsto	111	=	7

Bit Reversal Code

The Code. The bit reversal procedure uses a managing method, `allocateBitRev()` which calls a helper `reverseOneInt()`. First, a look at the methods:

```
// called once for a given FFT size and not repeated for that size
// produces array bitRev[] used to load input signal
void FftUtil::allocateBitRev()
{
    int k;

    if ( bitRev != NULL )
        delete bitRev;

    bitRev = new int[fftSize];
    for ( k = 0; k < fftSize; k++ )
        bitRev[k] = reverseOneInt(k);
}
```

```

    return;
}

int FftUtil::reverseOneInt(int inVal)
{
    int retVal, logSize, inValSave;
    inValSave = inVal;

    // inVal and retVal are array locations, and size of the array is fftSize
    retVal = 0;
    for (logSize = fftSize >> 2; logSize > 0; logSize >>= 1)
    {
        retVal |= (inVal & 1);
        retVal <<= 1;
        inVal >>= 1;
    }

    // adjusts for off-by-one last half of array
    if (inValSave >= (fftSize>>1))
        retVal++;

    return retVal;
}

```

Time Complexity

The driver method has a simple loop of size N making it $O(N)$. In that loop, it calls the helper, which careful inspection reveals to be $O(\log N)$: the loop in that helper is managed by the statement `logSize >>= 1`, which *halves the size* of array each pass, an action that always means log growth. Since this is a nesting of two loops, the full complexity is $O(N \log N)$.

Maybe Constant Time? This is done *in series* with the second phase of \mathcal{FFT} rebuilding, so this complexity and that of the next phase do not multiply; we will take the slower of the two. But the story gets better. Bit reversal need only be done *once* for any size N and can be skipped when new \mathcal{FFT} s of the same size are computed. It prepares a static array that is independent of the input signal. In that sense, it is really a *constant time* operation for a given \mathcal{FFT} or order N .

Either way you look at it, this preparation code won't affect the final complexity since we're about to see that the next phase is also $O(N \log N)$ and the normalization phase is only $O(N)$ making the full algorithm $O(N \log N)$ whether or not we count bit reversal.

21.8.3 Rebuilding from the Bit-Reversed Array

We continue to use the Danielson-Lanczos recursion relation to help guide our next steps. Once the input array is bit-reversed, we need to build-up from there. Figure 21.13 shows how we do this at the first level, using the singleton values to build the order-two \mathcal{DFT} s.

Note that since we are building a two-element \mathcal{DFT} , we use the 2nd root of unity, a.k.a. “-1.” After we do this for all the pairs, thus replacing all the singletons with

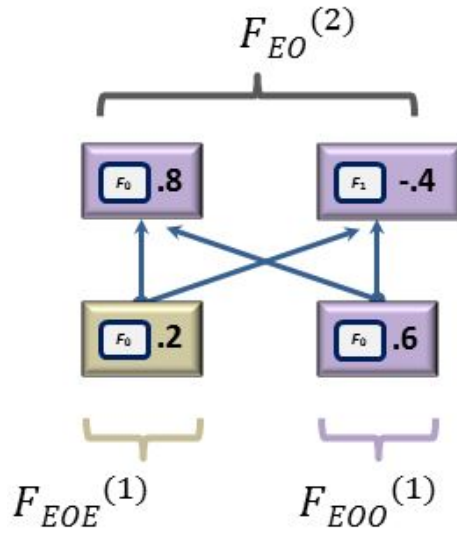


Figure 21.13: $\left[F_{EO}^{(2)}\right]_j = \left[F_{EOE}^{(1)}\right]_{j \pmod{1}} + (-1)^{-j} \left[F_{EOO}^{(1)}\right]_{j \pmod{1}}$

two-element \mathcal{DFT} s, we repeat the process at the next level: we build the four-element arrays from these two-element arrays. Figure 21.14 shows the process on one of the two four-element arrays, $F_E^{(4)}$. This time, we are using the 4th root of unity, i , as the multiplier of the odd term.

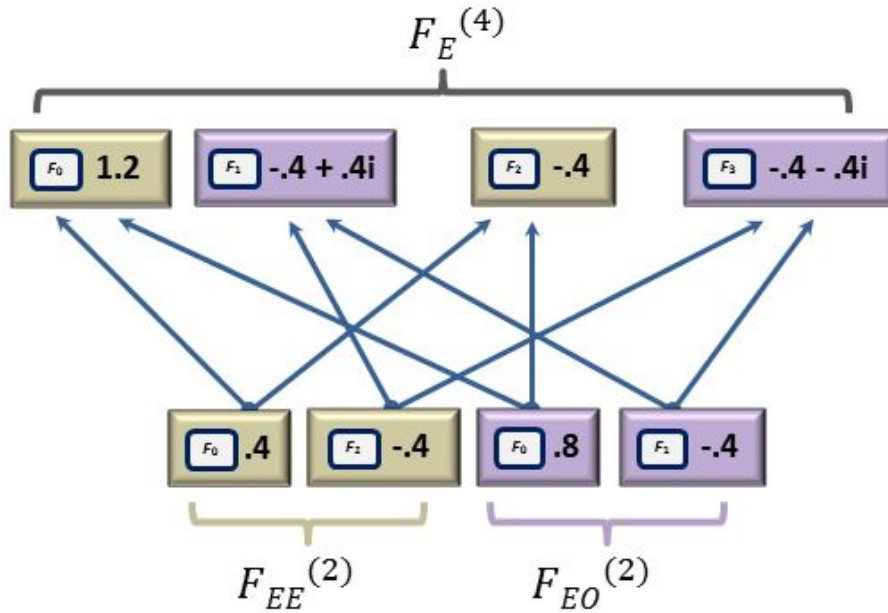


Figure 21.14: $\left[F_E^{(4)}\right]_j = \left[F_{EE}^{(2)}\right]_{j \pmod{2}} + (i)^{-j} \left[F_{EO}^{(2)}\right]_{j \pmod{2}}$

After the four-element $F_E^{(4)}$ we compute its companion four-element \mathcal{DFT} , $F_O^{(4)}$, then go to the next and final level, the computation of the full output array $F^{(8)}$. If the original signal were length 16, 32 or greater, we'd need to keep iterating this outer loop, building higher levels until we reached the full size of the input (and output) array.

Bottom-Up Array-Building Code

The implementation will contain loops that make it easier to “count” the *big-O* of this part, so let's first take a casual look at the class methods.

The Code. The private method, `combineEvenOdd()`, which does all the hard work, uses an iterative approach that mirrors the diagrams. Before looking at the full definition, it helps to first imagine processing the pairs, only, which turns those singleton values into order-two \mathcal{DFT} s. Recall, that for that first effort, we use -1 as the root of unity and either add or subtract,

$$\left[F_{EO}^{(2)}\right]_j = \left[F_{EOE}^{(1)}\right]_{j \pmod{1}} + (-1)^j \left[F_{EOO}^{(1)}\right]_{j \pmod{1}}$$

for $j = 0, 1$.

The effect is to replace those eight one-element \mathcal{DFT} s with four two-element \mathcal{DFT} s. The code to do that isn't too bad.

Two-Element \mathcal{DFT} s from Singletons:

```
// this computes the DFT of length 2 from the DFTs of length 1 using
// F0 = f0 + f1
```

```

// F1 = f0 - f1
// and does so, pairwise (after bit-reversal re-ordering).
// It represents the first iteration of the loop, but has the concepts
// of the recursive FFT formula.
// the roots[0], ... , roots[fftSize-1] are the nth roots in normal order
// which implies that -1 would be found at position fftSize/2

rootPos = fftSize/2; // identifies location of the -1 = omega in first pass
for (base = 0; base < fftSize; base +=2)
{
    for(j = 0; j < 2; j++)
    {
        arrayPos = (j * (fftSize - rootPos)) % fftSize ; // -j * omega
        outSig[base + j] = inSig[base] + roots[arrayPos] * inSig[base + 1];
    }
}

```

If we were to apply only this code, we would be replacing the adjacent pairs (after bit-reversal) by their order-two \mathcal{DFT} s. For an input signal

$$\{f_k\}_{k=0}^7 \equiv \{0, .1, .2, .3, .4, .5, .6, .7\},$$

the bit reversal would produce

$$\{0, .4, .2, .6, .1, .5, .3, .7\},$$

and the the above loop would replace these by the order-two \mathcal{DFT} s to produce

$$\{.4, -.4, .8, -.4, .6, -.4, 1, -.4\}.$$

(We are not normalizing by $1/\sqrt{N}$ until later.)

In the second generation of the loop we want to combine the order two \mathcal{DFT} s to produce *size four* \mathcal{DFT} s. This involves the following adjustments to the code:

- `base +=2` → `base +=4`
- `rootPos = fftSize/2` → `rootPos = fftSize/4` (locates i rather than -1)
- `inSig[base]` → `inSig[base + (j % 2)]` (original `inSig[base]` was equivalent to `inSig[base + (j % 1)]`), since $j \% 1 = 0$, *always*).
- `inSig[base + 1]` → `inSig[base + 2 + (j % 2)]`
- `j < 2` → `j < 4`

I'll let you write out the second iteration of the code that will combine \mathcal{DFT} s of *length two* and produce \mathcal{DFT} s of *length four*. After doing that exercise, one can see that the literals 1, 2, 4, etc. should be turned into a variable, `groupSize`, over which we loop (by surrounding the above code in an outer `groupSize`-loop). The result would be the final method.

Private Workhorse method combineEvenOdd():

```

// private non-recursive method builds FFT from 2-elements up
// reverse: true if doing an inverse FFT

bool FftUtil::combineEvenOdd(bool reverse)
{
// instance members of class -----
// outSig[], uxSig[]: working member arrays
// roots[]: precomputed roots of unity
// fftSize: size of current FFT

// key local variables
// rootPos: index in the roots[] array where current omega stored
// j: output index FFT being computed
// groupSize: power-of-2 size whose sub-FFT we are computing.
// increases from 1 to fftSize, doubling each time

int base, j, rootPos, arrayPos, groupSize, nextGroupStart;
Complex omegaToJ;

// check for fatal allocation problems
if (outSig == NULL || auxSig == NULL || roots == NULL)
    return false;

// process entire array in groups of 1, 2, 4, ... up to fftSize
for (groupSize = 1; groupSize < fftSize; groupSize <= 1)
{
    nextGroupStart = groupSize << 1;
    rootPos = fftSize / nextGroupStart;

    for (base = 0; base < fftSize; base += nextGroupStart)
    {
        for(j = 0; j < nextGroupStart; j++)
        {
            // allow forward and inverse fft
            if (reverse)
                arrayPos = (j * rootPos) % fftSize ; // j * omega
            else
                arrayPos = (j * (fftSize - rootPos)) % fftSize ; // -j * omega
            omegaToJ = roots[arrayPos];

            // Danielson and Lanczos formula
            outSig[base + j] = auxSig[ base + (j % groupSize) ]
                + omegaToJ * auxSig[ base + groupSize + (j % groupSize) ];
        }
    }

    // rather than be clever, copy array back to aux-input for next pass
    if ( (groupSize << 1) < fftSize )
        copyOutSigToAuxSig();
}

return true;
}

```

Time Complexity

At first glance, it might look like a triple nested loop, leading to some horrific cubic performance. However, on closer examination, we are relieved to find that

- the outer loop is a doubling of `groupsize` until it reaches N , essentially a $\log N$ proposition, and

- the two inner loops together constitute only N loop passes.

Thus, *array building* is $O(N \log N)$, as we had hoped.

21.8.4 Normalization

To complete the trio, we have to write the `normalize()` method. It's a simple linear complexity loop taken in series with slower complexity code, so it doesn't affect growth of the algorithm.

The Normalize Method

```
bool FftUtil::normalize()
{
    double factor;
    int k;

    if (outSig == NULL)
        return false;

    factor = 1. / sqrt(mSize);
    for (k = 0; k < mSize; k++)
        outSig[k] = outSig[k] * factor;
    return true;
}
```

21.8.5 Overall Complexity

We have three methods in series, the most costly of which is $O(N \log N)$, making the entire *FFT* $O(N \log N)$.

21.8.6 Software Testing

The only thing left to do is time this and compare with the recursive approach. Here's the output:

```
FFT size 1024   Non-recursive FFT: 0 seconds.
FFT size 2048   Non-recursive FFT: 0 seconds.
FFT size 4096   Non-recursive FFT: 0.001 seconds.
FFT size 8192   Non-recursive FFT: 0.001 seconds.
FFT size 16384   Non-recursive FFT: 0.003 seconds.
FFT size 32768   Non-recursive FFT: 0.007 seconds.
FFT size 65536   Non-recursive FFT: 0.016 seconds.
```

It is slightly slower than *linear*, the difference being the expected factor of $\log N$ (although we couldn't tell that detail from the above times). Not only is this evidence of the $N \log N$ time complexity, but it is orders of magnitude faster than the recursive algorithm (4.076 seconds vs. .003 seconds for a 16k array). We finally have our true *FFT*.

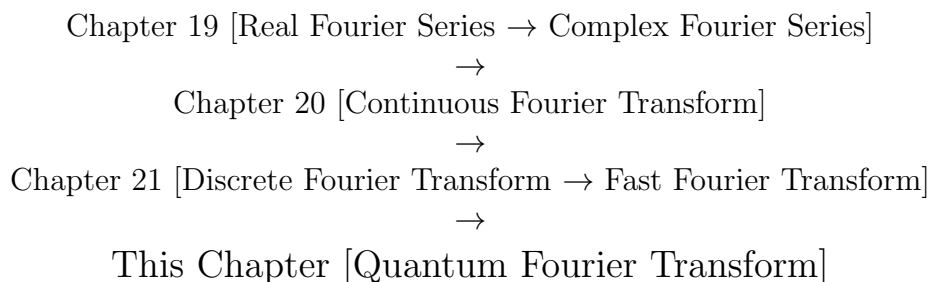
This gives us more (far more) than enough classical background to tackle the *quantum Fourier transform*.

Chapter 22

The Quantum Fourier Transform

22.1 From Classical Fourier Theory to the QFT

Today you'll meet the capstone of our four chapter sequence, the *quantum Fourier transform*, or QFT .



If you skipped the three earlier lessons, you can always refer to them on an as-needed basis. They contain every detail and definition about classical Fourier theory that will be cited today, all developed from scratch. Search the table of contents for any topic you want to review if you hit a snag.

22.2 Definitions

22.2.1 From \mathbb{C}^{2^n} to $\mathcal{H}_{(n)}$

We know that the *discrete Fourier transform of order N* , or \mathcal{DFT} (its *order* usually implied by context), is a special operator that takes an N -dimensional complex vector $f = (f_k)$ to another complex vector $\tilde{f} = (\tilde{f}_j)$. In symbols,

$$\begin{aligned} \mathcal{DFT} : \mathbb{C}^N &\longrightarrow \mathbb{C}^N, \\ \mathcal{DFT}(f) &\longmapsto \tilde{f}, \end{aligned}$$

or, if we want to emphasize the coordinates,

$$\mathcal{DFT}[(f_k)] \mapsto (\tilde{f}_j).$$

When we moved to the *fast Fourier transform* the only change (other than the implementation details) was that we required $N = 2^n$ be a power-of-2. Symbolically,

$$\begin{aligned}\mathcal{FFT} : \mathbb{C}^{2^n} &\longrightarrow \mathbb{C}^{2^n}, \\ \mathcal{FFT}[(f_k)] &\mapsto (\tilde{f}_j).\end{aligned}$$

We maintain this restriction on N and continue to take n to be $\log N$ (base 2 always implied).

We'll build the definition of the \mathcal{QFT} atop our firm foundation of the \mathcal{DFT} , so we start with

$$\mathcal{DFT} : \mathbb{C}^{2^n} \longrightarrow \mathbb{C}^{2^n},$$

a 2^n th order mapping of \mathbb{C}^{2^n} to itself and use that to define the 2^n th order \mathcal{QFT} acting on an n th order Hilbert space,

$$\mathcal{QFT} : \mathcal{H}_{(n)} \longrightarrow \mathcal{H}_{(n)}.$$

[**Order.** The word “order,” when describing the $\mathcal{DFT} = \mathcal{DFT}^{(N)}$ means N , the *dimension* of the underlying space, while the same word, “order,” when applied to a *tensor product* space $\mathcal{H}_{(n)}$ is n , the number of component, single qubit, spaces in the product. The two “orders” are not the same; they are related, though, by $N = 2^n$ or, equivalently, $n = \log N$.]

22.2.2 Approaches to Operator Definition

We know that there are various ways to define a unitary operator, U ,

$$U : \mathcal{H}_{(n)} \longrightarrow \mathcal{H}_{(n)}.$$

Among them, three are prominent.

1. Describe the action of U on the $N = 2^n$ CBS kets, $\{|x\rangle^n\} = \{|0\rangle^n, \dots, |N-1\rangle^n\}$, and *extend linearly*.
2. Describe the action of U on an arbitrary $|\psi\rangle^n \in \mathcal{H}_{(n)}$, by expressing $U|\psi\rangle^n$ in terms of its $N = 2^n$ complex coefficients c_0, \dots, c_{N-1} .
3. Give the *matrix* for U , in the standard basis, used to convert $|\psi\rangle^n$'s coefficients to $U|\psi\rangle^n$'s coefficients through matrix multiplication.

Whichever method we choose, we have to confirm that U is both *linear* and *unitary* (although in methods 1 and 3 we get linearity for free, since it's built-into those definitions). Also, we only have to use one technique, since the other two can be surmised using linear algebra. However, if we use a different method than someone else to define the same operator, we had better check that our definition agrees with theirs.

22.2.3 Review of the Hadamard Operator

Let's make sure we understand these concepts before defining the \mathcal{QFT} by reprising an example from our past, the *Hadamard operator*.

First Order Hadamard

We used method 1 to define $H = H^{\otimes 1}$, by

$$\begin{aligned} H|0\rangle &\equiv \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \\ H|1\rangle &\equiv \frac{|0\rangle - |1\rangle}{\sqrt{2}}, \end{aligned}$$

and extended, linearly. We might have used method 2, this way:

$$\begin{aligned} \text{If } |\psi\rangle &= \alpha|0\rangle + \beta|1\rangle, \\ \text{then } H|\psi\rangle &\equiv \left(\frac{\alpha + \beta}{\sqrt{2}}\right)|0\rangle + \left(\frac{\alpha - \beta}{\sqrt{2}}\right)|1\rangle \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}. \end{aligned}$$

Finally, some authors use method 3 to define the matrix for H ,

$$M_H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

which can then be used to produce the formulas for either method 1 or method 2.

N th Order Hadamard

Meanwhile, for the n -fold Hadamard, we didn't need to define it: we just derived it using the laws of tensor products. We found (using method 1), that

$$H^{\otimes n} |x\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle^n.$$

(Remember that $x \cdot y = x \odot y$ is the mod-2 pairing – a *pseudo* inner product – of the two bit strings.)

Because it is so much easier to express operators on the CBS, $\{|x\rangle^n\}$, than to say what it does on the complex amplitudes, we normally do it that way, just as we did with $H^{\otimes n}$. Indeed, this is about how half of the authors define the \mathcal{QFT} .

Despite the sales pitch for method 1, I think it's more instructive to use method 2 for the \mathcal{QFT} , then afterward show how it can be done using method 1, being careful to show the equivalence of the two approaches.

22.2.4 Defining the QFT

As long as we're careful to check that our definition provides a *linear* and *unitary* transformation, we are free to use a state's *coordinates* to define it. Consider a general state $|\psi\rangle^n$ and its preferred basis *amplitudes* (a.k.a., *coordinates* or *coefficients*),

$$|\psi\rangle^n \longleftrightarrow (c_x)_{x=0}^{N-1} = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{pmatrix}, \quad N = 2^n.$$

We describe how the order- N $QFT = QFT^{(N)}$ acts on the 2^n coefficients, and this will define the QFT for any qubit in $\mathcal{H}_{(n)}$.

Concept Definition of the Order- N QFT

$$\begin{aligned} \text{If } |\psi\rangle^n &\longleftrightarrow (c_x)_{x=0}^{N-1}, \\ \text{then } QFT^{(N)} |\psi\rangle^n &\longleftrightarrow (\tilde{c}_y)_{y=0}^{N-1}. \end{aligned}$$

In words, starting from $|\psi\rangle^n$, we form the vector of its amplitudes, (c_x) ; we treat (c_x) like an ordinary complex vector of size 2^n ; we take its $DFT^{(N)}$ to get another vector (\tilde{c}_y) ; we declare the coefficients $\{\tilde{c}_y\}$ to be the amplitudes of our desired output state, $QFT^{(N)} |\psi\rangle^n$. The end.

Expressing the Order of the QFT

If we need it, we'll display the QFT 's *order* in the superscript with the notation $QFT^{(N)}$. Quantum computer scientists don't usually specify the order in diagrams, so we'll often go with a plain " QFT " and remember that it operates on an order- n Hilbert space having dimension $N = 2^n$.

Explicit Definition of the Order- N QFT

The concept definition expressed formulaically, says that

$$\begin{aligned} \text{if } |\psi\rangle^n &= \sum_{x=0}^{N-1} c_x |x\rangle^n, \\ \text{then } QFT |\psi\rangle^n &\equiv \sum_{y=0}^{N-1} \tilde{c}_y |y\rangle^n. \end{aligned}$$

We can really feel the Fourier transform concept when we view the states as complex vectors $|\psi\rangle^n = \mathbf{c} = (c_x)$ of size 2^n to which we subject the standard \mathcal{DFT} ,

$$\mathcal{QFT} |\psi\rangle^n = \sum_{y=0}^{N-1} [\mathcal{DFT}(\mathbf{c})]_y |y\rangle^n.$$

The definition assumes the reader can compute a \mathcal{DFT} , so we'd better unwind our definition by expressing the \mathcal{QFT} explicitly. The y th coordinate of the output \mathcal{QFT} is produced using

$$[\mathcal{QFT} |\psi\rangle^n]_y = \tilde{c}_y = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} c_x \omega^{yx},$$

where $\omega = \omega_N$ is the primitive N th root of unity. The y th coordinate can also be obtained using the *the dot-with-the-basis-vector trick*,

$$[\mathcal{QFT} |\psi\rangle^n]_y = {}^n\langle y | \mathcal{QFT} |\psi\rangle^n,$$

so you might see the RHS notation, rather than the LHS y -subscript form, by physics-oriented authors. For example, it could appear in the definition – or even computation – of the y th coordinate of the \mathcal{QFT} ,

$${}^n\langle y | \mathcal{QFT} |\psi\rangle^n = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} c_x \omega^{yx}.$$

We'll stick with the subscript notation, $[\mathcal{QFT} |\psi\rangle^n]_y$, for now.

Notation and Convention

*Hold the phone. Doesn't the \mathcal{DFT} have a **negative** root exponent, ω^{-jk} ?* The way I defined it, yes. As I already said, there are two schools of thought regarding forward vs. reverse Fourier transforms and \mathcal{DFT} s. I really prefer the negative exponent for the forward transform because it arises naturally when decomposing a function into its frequencies. But there is only one school when defining the \mathcal{QFT} , and it has a *positive* root exponent in the forward direction (and negative in the reverse). Therefore, I have to switch conventions.

[If you need more specifics, here are three options. (i) You can go back and define the forward \mathcal{DFT} using positive exponent from the start ... OR ... (ii) You can consider the appearance of “ \mathcal{DFT} ” in the above expressions as motivational but rely only on the explicit formulas for the formal definition of the \mathcal{QFT} without anxiety about the exponent's sign difference ... OR ... (iii) You can preserve our original \mathcal{DFT} but modify the definition of \mathcal{QFT} by replacing “ \mathcal{DFT} ” with “ \mathcal{DFT}^{-1} ” everywhere in this section.]

Anyway, we won't be referring to the \mathcal{DFT} , only the \mathcal{QFT} – effective *immediately* – so the discrepancy starts and ends now.

Full Definition of QFT using Method 2

You should also be ready to see the full output vector, $QFT |\psi\rangle^n$, expanded along the CBS,

$$\begin{aligned} \text{if } |\psi\rangle^n &= \sum_{x=0}^{N-1} c_x |x\rangle^n \\ \text{then } QFT |\psi\rangle^n &\equiv \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} c_x \omega^{yx} |y\rangle^n. \end{aligned}$$

[**Exercise.** Show how this is a consequence of the earlier definition.]

Typical Definition QFT Using Definition Method 1

It's more common in quantum computing to take the CBS route, and in that case the definition of the QFT would be

$$QFT |x\rangle^n \equiv \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{yx} |y\rangle^n,$$

from which we would get the definition for arbitrary states by applying linearity to their CBS expansion. Our task at hand is to make sure

1. the definitions are equivalent, and
2. they produce a linear, unitary operator.

Vetting a putative operator like this represents a necessary step in demonstrating the viability of our ideas, so it's more than just an academic exercise. You may be doing this on your own operators some day. Imagine future students studying “the *[your name here]* transform”. It can happen.

Equivalence of the Definitions

Step 1) Agreement on CBS. We'll show that the coefficient definition, QFT_{ours} , agrees with the typical CBS definition, QFT_{cbs} on the CBS. Consider the CBS $|x\rangle^n$.

It is a tall 2^n component vector with a 1 in position x :

$$\begin{aligned}
 |x\rangle^n &= \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \leftarrow x\text{th coefficient} \\
 &= \sum_{k=0}^{N-1} c_k |k\rangle^n = \sum_{k=0}^{N-1} \delta_{kx} |k\rangle^n .
 \end{aligned}$$

Now apply our definition to this ket and see where it leads:

$$\begin{aligned}
 \mathcal{QFT}_{\text{ours}} |x\rangle^n &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \sum_{k=0}^{N-1} c_k \omega^{yk} |y\rangle^n \\
 &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \sum_{k=0}^{N-1} \delta_{kx} \omega^{yk} |y\rangle^n \\
 &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{yx} |y\rangle^n \\
 &= \mathcal{QFT}_{\text{cbs}} |x\rangle^n . \quad \text{QED}
 \end{aligned}$$

Step 2) Linearity. We must also show that $\mathcal{QFT}_{\text{ours}}$ is linear. Once we do that we'll know that both it and $\mathcal{QFT}_{\text{cbs}}$ (linear by construction) not only agree on the CBS but are *both* linear, forcing the two to be equivalent over the entire $\mathcal{H}_{(n)}$.

The definition of $\mathcal{QFT}_{\text{ours}}$ in its expanded form is

$$\mathcal{QFT} |\psi\rangle^n \equiv \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \sum_{x=0}^{N-1} c_x \omega^{yx} |y\rangle ,$$

which, on close inspection, we recognize as matrix multiplication of the vector (c_x) by the matrix (ω^{xy}) (the factor of $1/\sqrt{N}$, notwithstanding). If this is not obvious, here's a "slo-mo" version of that statement.

$$|\psi\rangle^n = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \end{pmatrix} ,$$

so

$$\begin{aligned}
\mathcal{QFT}_{\text{ours}} |\psi\rangle^n &= \begin{pmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \tilde{c}_2 \\ \vdots \end{pmatrix} \\
&= \frac{1}{\sqrt{N}} \begin{pmatrix} \sum_x c_x \omega^{0x} \\ \sum_x c_x \omega^{1x} \\ \sum_x c_x \omega^{2x} \\ \vdots \end{pmatrix} \\
&= \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots \\ 1 & \omega & \omega^2 & \cdots \\ 1 & \omega^2 & \omega^4 & \cdots \\ 1 & \omega^3 & \omega^6 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \end{pmatrix}.
\end{aligned}$$

Whenever an operator's coordinates are transformed by matrix multiplication, the operator is linear. QED

Step 3) Unitarity. Now that we have the matrix for the \mathcal{QFT} ,

$$M_{\mathcal{QFT}} \equiv \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots \\ 1 & \omega & \omega^2 & \cdots \\ 1 & \omega^2 & \omega^4 & \cdots \\ 1 & \omega^3 & \omega^6 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix},$$

we can use $M_{\mathcal{QFT}}$ to confirm that \mathcal{QFT} is unitary: if the matrix is unitary, so is the operator. (**Caution.** This is only true when the basis in which the matrix is expressed is orthonormal, which $\{|x\rangle^n\}$ is.) We need to show that

$$(M_{\mathcal{QFT}})^\dagger M_{\mathcal{QFT}} = \mathbb{1},$$

so let's take the dot product of row x of $(M_{\mathcal{QFT}})^\dagger$ with column y of $M_{\mathcal{QFT}}$:

$$\begin{aligned}
\frac{1}{\sqrt{N}} (1, (\omega^x)^*, (\omega^{2x})^*, \dots) \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ \omega^y \\ \omega^{2y} \\ \vdots \end{pmatrix} &= \frac{1}{N} (1, \omega^{-x}, \omega^{-2x}, \dots) \begin{pmatrix} 1 \\ \omega^y \\ \omega^{2y} \\ \vdots \end{pmatrix} \\
&= \frac{1}{N} \sum_{k=0}^{N-1} \omega^{k(y-x)} = \frac{1}{N} (\delta_{xy} N) = \delta_{xy}. \quad \text{QED}
\end{aligned}$$

(The second-from-last identity is from **Exercise D** (roots-of-unity section) in the early lesson *complex arithmetic*.)

22.3 Features of the QFT

22.3.1 Shift Property

The shift property of the DFT ,

$$f_{k-l} \xrightarrow{DFT} \omega^{-lk} F_k,$$

when plugged into the definition of QFT results in a quantum *translation invariance*

$$QFT |x - z\rangle^n = \omega^{zx} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{yx} |y\rangle^n = \omega^{zx} QFT |x\rangle^n, \quad z \leq x.$$

We lose the minus exponent of ω because the QFT uses a positive exponent in its forward direction.

[**Caution.** CBS kets contain encoded ints in the range 0 to $2^n - 1$. But the subtraction on the LHS is ordinary, not mod-2, subtraction, so there's no guarantee that it stays in that range unless we force it to, either by taking subtraction mod- N or the simpler condition $z \leq x$, which is enough for our purposes.]

22.3.2 A Comparison between QFT and H

You may find it enlightening to see some similarities and differences between the Hadamard operator and the QFT .

We are working in an $N = 2^n$ -dimensional Hilbert space, $\mathcal{H}_{(n)}$. We always signify this on the Hadamard operator using a superscript, as in $H^{\otimes n}$. On the other hand, when we specify the order of the QFT , we do so using the superscript (N) , as in “ $QFT^{(N)}$.” In what follows, I'll continue to omit the superscript for the QFT initially and only bring it in when needed.

How do these two operators compare on the CBS?

n th Order Hadamard

$$H^{\otimes n} |x\rangle = \left(\frac{1}{\sqrt{2}} \right)^n \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle,$$

where $x \cdot y = x \odot y$ is the mod-2 dot product based on the individual binary digits in the base-2 representation of x and y . Now $-1 = \omega_2 (e^{2\pi i/2} = e^{\pi i} = -1, \checkmark)$, so let's

replace the -1 , above, with its symbol as the *square root of unity*,

$$H^{\otimes n} |x\rangle^n = \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} \omega_2^{x \cdot y} |y\rangle^n.$$

N -Dimensional QFT

The definition of the N th order QFT for $N = 2^n$ is

$$QFT^{(N)} |x\rangle^n \equiv \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} \omega^{yx} |y\rangle^n.$$

The differences between the two operators are now apparent.

- The QFT uses a primitive 2^n th root of unity, while $H^{\otimes n}$ uses a square root of unity.
- The exponent of the root-of-unity is an ordinary integer product for QFT , but is a mod-2 dot product for $H^{\otimes n}$.

QFT Equals H for $N = 2$

A useful factoid is that when $N = 2$ ($n = 1$) both operators are the same. Look:

$$\begin{aligned} QFT^{(2)} |x\rangle &= \left(\frac{1}{\sqrt{2}}\right)^1 \sum_{y=0}^1 (-1)^{yx} |y\rangle \\ &= \frac{1}{\sqrt{2}} \left((-1)^{0 \cdot x} |0\rangle + (-1)^{1 \cdot x} |1\rangle \right) \\ &= \begin{cases} \frac{|0\rangle + |1\rangle}{\sqrt{2}}, & x = 0 \\ \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & x = 1 \end{cases} \end{aligned}$$

22.3.3 The Quantum Fourier Basis

Any quantum gate is unitary by necessity (see the lecture on *single qubits*), and a unitary operator acting on an orthonormal basis produces another orthonormal basis. I'll restate the statute (from that lecture that provided the rigor for all this).

Theorem (Basis Conversion Property). *If U is a unitary operator and \mathcal{A} is an orthonormal basis, then $U(\mathcal{A})$, i.e., the image of vectors \mathcal{A} under U , is another orthonormal basis, \mathcal{B} .*

Applying $QFT^{(2^n)}$ to the preferred z -basis in $\mathcal{H}_{(n)}$ will give us another basis for $\mathcal{H}_{(n)}$. We'll write it as $\{ |\tilde{x}\rangle^n \}$, where

$$|\tilde{x}\rangle^n \equiv QFT^{(2^n)} |x\rangle^n .$$

We refer to this as the *quantum Fourier basis* or, once we are firmly back in purely quantum territory, simply as the *Fourier basis* or *frequency basis*. It will be needed in Shor's period-finding algorithm.

The Take-Away

Where we used $H^{\otimes n}$ to our advantage for Simon's $(\mathbb{Z}_2)^n$ - periodicity, we anticipate using the QFT to achieve a similar effect when we work on Shor's ordinary integer periodicity.

22.4 The QFT Circuit

Going from an operator definition to a quantum circuit that has an efficient (polynomial growth complexity) circuit is always a challenge. We will approach the problem in bite-sized pieces.

22.4.1 Notation

We'll review and establish some notation applicable to an $N = 2^n$ -dimensional $\mathcal{H}_{(n)}$.

The CBS kets in an N -dimensional $\mathcal{H}_{(n)}$ are officially tensor products of the individual CBSs of the 2-dimensional qubit spaces,

$$|x\rangle^n = |x_{n-1}\rangle \otimes |x_{n-2}\rangle \otimes |x_{n-3}\rangle \otimes \dots \otimes |x_0\rangle = \bigotimes_{k=0}^{n-1} |x_k\rangle ,$$

where $|x_k\rangle$ is either $|0\rangle$ or $|1\rangle$ of the k th qubit space.

We index in decreasing order from x_{n-1} to x_0 because we'll want the right-most bit to correspond to the least significant bit of the binary number $x_{n-1} \dots x_1 x_0$.

One shorthand we have used in the past for this CBS is

$$|x_{n-1}\rangle |x_{n-2}\rangle \dots |x_1\rangle |x_0\rangle ,$$

but we also know two other common variations, the encoded decimal integer x ,

$$|x\rangle^n , \quad x \in \{0, 1, 2, 3, \dots, 2^n - 1\} ,$$

and its binary representation,

$$|x_{n-1} x_{n-2} \dots x_3 x_2 x_1 x_0\rangle , \quad x_k \in \{0, 1\} .$$

For example, for $n = 3$,

$$\begin{aligned} |0\rangle^3 &\longleftrightarrow |000\rangle, \\ |1\rangle^3 &\longleftrightarrow |001\rangle, \\ |2\rangle^3 &\longleftrightarrow |010\rangle, \\ |3\rangle^3 &\longleftrightarrow |011\rangle, \\ |4\rangle^3 &\longleftrightarrow |100\rangle, \end{aligned}$$

and, in general,

$$|x\rangle^3 \longleftrightarrow |x_2 x_1 x_0\rangle.$$

We'll also use the natural consequence of this notation,

$$x = \sum_{k=0}^{n-1} x_k 2^k.$$

22.4.2 The Math that Leads to the Circuit: Factoring the QFT

We begin with the definition of the QFT 's action on a CBS and gradually work towards the expression that will reveal the circuit. It will take a few screens, so here we go. (I'm going to move the constant $1/\sqrt{N}$ to LHS to reduce syntax.)

$$\begin{aligned} (\sqrt{N}) \mathcal{QFT}^{(N)} |x\rangle^n &= \sum_{y=0}^{N-1} \omega^{xy} |y\rangle^n \\ &= \sum_{y=0}^{N-1} \omega^{x \sum_{k=0}^{n-1} y_k 2^k} |y_{n-1} \dots y_1 y_0\rangle \\ &= \sum_{y=0}^{N-1} \left(\prod_{k=0}^{n-1} \omega^{x y_k 2^k} \right) |y_{n-1} \dots y_1 y_0\rangle \end{aligned}$$

(I displayed the order, N , explicitly in the LHS's " $\mathcal{QFT}^{(N)}$," something that will come in handy in about two screens.) To keep the equations from overwhelming us, let's symbolize the inside product by

$$\pi_{xy} \equiv \prod_{k=0}^{n-1} \omega^{x y_k 2^k},$$

providing a less intimidating

$$(\sqrt{N}) \mathcal{QFT} |x\rangle^n = \sum_{y=0}^{N-1} \pi_{xy} |y_{n-1} \dots y_1 y_0\rangle.$$

To make this concrete, we pause to see what it looks like for $n = 3$.

$$\begin{aligned}
& \left(\sqrt{8} \right) \mathcal{QFT} |x\rangle^3 \\
&= \pi_{x \cdot 0} |000\rangle + \pi_{x \cdot 1} |001\rangle + \pi_{x \cdot 2} |010\rangle + \pi_{x \cdot 3} |011\rangle \\
&+ \pi_{x \cdot 4} |100\rangle + \pi_{x \cdot 5} |101\rangle + \pi_{x \cdot 6} |110\rangle + \pi_{x \cdot 7} |111\rangle
\end{aligned}$$

Separating a \mathcal{DFT} into even and odd sub-arrays led to the \mathcal{FFT} algorithm, and we try that here in the hope of a similar profit.

$$\begin{aligned}
& \left(\sqrt{8} \right) \mathcal{QFT} |x\rangle^3 \\
&= \pi_{x \cdot 0} |000\rangle + \pi_{x \cdot 2} |010\rangle + \pi_{x \cdot 4} |100\rangle + \pi_{x \cdot 6} |110\rangle \\
&+ \pi_{x \cdot 1} |001\rangle + \pi_{x \cdot 3} |011\rangle + \pi_{x \cdot 5} |101\rangle + \pi_{x \cdot 7} |111\rangle \\
&\equiv \quad y\text{-even group } \sum \\
&\quad + \quad y\text{-odd group } \sum.
\end{aligned}$$

Analysis of y -even Group

The least significant bit, y_0 , of all terms in the y -even group is always 0, so for terms in this group,

$$\begin{aligned}
y_0 &= 0 \\
\omega^{xy_0 2^0} &= \omega^{x \cdot 0 \cdot 1} = 1, \quad \text{so for even } y \text{ we get} \\
\pi_{xy} &= \prod_{k=0}^2 \omega^{xy_k 2^k} = \prod_{k=1}^2 \omega^{xy_k 2^k}.
\end{aligned}$$

Evidently, the π_{xy} in the y -even group can start the product at $k = 1$ rather than $k = 0$ since the $k = 0$ factor is 1. We rewrite the even sum with this new knowledge:

$$\begin{aligned}
& y\text{-even group } \sum \\
&= \pi_{x \cdot 0} |000\rangle + \pi_{x \cdot 2} |010\rangle + \pi_{x \cdot 4} |100\rangle + \pi_{x \cdot 6} |110\rangle \\
&= \left(\pi_{x \cdot 0} |00\rangle + \pi_{x \cdot 2} |01\rangle + \pi_{x \cdot 4} |10\rangle + \pi_{x \cdot 6} |11\rangle \right) |0\rangle \\
&= \left(\sum_{\substack{y=0 \\ y \text{ even}}}^7 \left(\prod_{k=1}^2 \omega^{xy_k 2^k} \right) |y_2 y_1\rangle \right) |0\rangle
\end{aligned}$$

Now that $|y_0\rangle = |0\rangle$ has been factored from the sum we can run through the even y more efficiently by

- halving the y -sum from $\sum_{\text{even}}^7 \rightarrow \sum_{\text{all}}^3$,
- replacing $|y_2 y_1\rangle \rightarrow |y_1 y_0\rangle$,
- shifting the k -product down-by-1 so $\prod_1^2 \rightarrow \prod_0^1$, and
- replacing $2^k \rightarrow 2^{k+1}$.

(Take a little time to see why these adjustments make sense.) Applying the bullets gives

$$\begin{aligned}
 & y\text{-even group } \sum \\
 &= \left(\sum_{y=0}^3 \left(\prod_{k=0}^1 \omega^{x y_k 2^{k+1}} \right) |y_1 y_0\rangle \right) |0\rangle \\
 &= \left(\sum_{y=0}^3 \left(\prod_{k=0}^1 (\omega^2)^{x y_k 2^k} \right) |y_1 y_0\rangle \right) |0\rangle .
 \end{aligned}$$

There's one final reduction to be made. While we successfully halved the size of y inside the kets from its original $0 \rightarrow 7$ range to the smaller interval $0 \rightarrow 3$, the x in the exponent still roams free in the original set. How do we get it to live in the same smaller world as y ? The key lurks here:

$$(\omega^2)^{x y_k 2^k}$$

The even sub-array rearrangement precipitated a 4th root of unity ω^2 rather than the original 8th root ω . This enables us to replace any $x > 3$ with $x - 4$, bringing it back into the $0 \rightarrow 3$ range without affecting the computed values. To see why, do the following short exercise.

[Exercise. For $4 \leq x < 7$ write $x = 4 + p$, where $0 \leq p < 3$. Plug $4 + p$ in for x in the above exponent and simplify, leveraging the fact that $\omega = \sqrt[8]{1}$.]

The bottom line is that we can replace x with $(x \bmod 4)$ and the equality still holds true,

$$y\text{-even group } \sum = \left(\sum_{y=0}^3 \left(\prod_{k=0}^1 (\omega^2)^{(x \bmod 4) y_k 2^k} \right) |y_1 y_0\rangle \right) |0\rangle .$$

Using the same logic on general n , we get

$$\begin{aligned}
 & y\text{-even group } \sum \\
 &= \left(\sum_{y=0}^{N/2-1} \left(\prod_{k=0}^{n-2} (\omega^2)^{(x \bmod N/2) y_k 2^k} \right) |y_{n-2} y_{n-3} \dots y_0\rangle \right) |0\rangle .
 \end{aligned}$$

Compare with our latest retooling of the \mathcal{QFT} ,

$$\left(\sqrt{N}\right) \mathcal{QFT}^{(N)} |x\rangle^n = \sum_{y=0}^{N-1} \left(\prod_{k=0}^{n-1} \omega^{xy_k 2^k} \right) |y_{n-1} \dots y_1 y_0\rangle ,$$

and we are encouraged to see the order- $N/2$ \mathcal{QFT} staring us in the face,

$$y\text{-even group } \sum = \left(\sqrt{\frac{N}{2}} \mathcal{QFT}^{(N/2)} |x \bmod (N/2)\rangle^{(n-1)} \right) |0\rangle .$$

Let's make this as concise as possible by using \tilde{x} to mean $x \bmod (N/2)$.

$$y\text{-even group } \sum = \left(\sqrt{\frac{N}{2}} \mathcal{QFT}^{(N/2)} |\tilde{x}\rangle^{(n-1)} \right) |0\rangle .$$

We've expressed the even group as a \mathcal{QFT} whose order is $N/2$, half the original N . The aroma of recursion (and success) is in the air. Now, let's take a stab at the odd group.

Analysis of y -odd Group

The least significant bit, y_0 , of the all terms in the y -odd group is always 1, so for terms in this group,

$$\begin{aligned} y_0 &= 1 \\ \omega^{xy_0 2^0} &= \omega^{x \cdot 1 \cdot 1} = \omega^x, \quad \text{so for odd } y \text{ we get} \\ \pi_{xy} &= \prod_{k=0}^2 \omega^{xy_k 2^k} = \omega^x \prod_{k=1}^2 \omega^{xy_k 2^k}. \end{aligned}$$

We separated the factor ω^x from the rest so that we could start the product at $k = 1$ to align our analysis with the y -even group, above. We rewrite the odd sum using this adjustment:

$$\begin{aligned} y\text{-odd group } \sum &= \pi_{x \cdot 1} |001\rangle + \pi_{x \cdot 3} |011\rangle + \pi_{x \cdot 5} |101\rangle + \pi_{x \cdot 7} |111\rangle \\ &= \left(\pi_{x \cdot 1} |00\rangle + \pi_{x \cdot 3} |01\rangle + \pi_{x \cdot 5} |10\rangle + \pi_{x \cdot 7} |11\rangle \right) |1\rangle \\ &= \omega^x \left(\sum_{\substack{y=0 \\ y \text{ odd}}}^7 \left(\prod_{k=1}^2 \omega^{xy_k 2^k} \right) |y_2 y_1\rangle \right) |1\rangle \end{aligned}$$

Now that $|y_0\rangle = |1\rangle$ and ω^x have both been factored from the sum, we run through the odd y by

- halving the y -sum from $\sum_{\text{odd}}^7 \rightarrow \sum_{\text{all}}^3$,

- replacing $|y_2 y_1\rangle \longrightarrow |y_1 y_0\rangle$,
- shifting the k -product down-by-1 so $\prod_1^2 \longrightarrow \prod_0^1$, and
- replacing $2^k \longrightarrow 2^{k+1}$.

These bullets give us

$$\begin{aligned} & y\text{-odd group } \sum \\ &= \omega^x \left(\sum_{y=0}^3 \left(\prod_{k=0}^1 (\omega^2)^{x y_k 2^k} \right) |y_1 y_0\rangle \right) |1\rangle , \end{aligned}$$

and we follow it by the same replacement, $(x \bmod 4) \rightarrow x$ that worked for the y -even group (and works here, too):

$$\begin{aligned} & y\text{-odd group } \sum \\ &= \omega^x \left(\sum_{y=0}^3 \left(\prod_{k=0}^1 (\omega^2)^{(x \bmod 4) y_k 2^k} \right) |y_1 y_0\rangle \right) |1\rangle . \end{aligned}$$

Using the same logic on general n , we get

$$\begin{aligned} & y\text{-odd group } \sum \\ &= \omega^x \left(\sum_{y=0}^{N/2-1} \left(\prod_{k=0}^{n-2} (\omega^2)^{(x \bmod N/2) y_k 2^k} \right) |y_{n-2} y_{n-3} \dots y_0\rangle \right) |1\rangle . \end{aligned}$$

Once again, we are thrilled to see an $(N/2)$ -Order \mathcal{QFT} emerge from the fray,

$$y\text{-odd group } \sum = \omega^x \left(\sqrt{\frac{N}{2}} \mathcal{QFT}^{(N/2)} |\tilde{x}\rangle^{(n-1)} \right) |1\rangle .$$

The Recursion Relation

Combine the y -even group \sum and y -odd group \sum to get

$$\begin{aligned} \left(\sqrt{N} \right) \mathcal{QFT} |x\rangle^n &= y\text{-even group } \sum + y\text{-odd group } \sum \\ &= \left(\sqrt{\frac{N}{2}} \mathcal{QFT}^{(N/2)} |\tilde{x}\rangle^{(n-1)} \right) |0\rangle \\ &\quad + \omega^x \left(\sqrt{\frac{N}{2}} \mathcal{QFT}^{(N/2)} |\tilde{x}\rangle^{(n-1)} \right) |1\rangle \\ &= \left(\sqrt{\frac{N}{2}} \mathcal{QFT}^{(N/2)} |\tilde{x}\rangle^{(n-1)} \right) (|0\rangle + \omega^x |1\rangle) \end{aligned}$$

The binomial $|0\rangle + \omega^x |1\rangle$ had to end up on the *right* of the sum because we were peeling off the *least*-significant $|0\rangle$ and $|1\rangle$ in the even-odd analysis; *tensor products*

are not commutative. This detail leads to a slightly annoying but easily handled wrinkle in the end. You'll see.

Dividing out the \sqrt{N} , using 2^n for N and rearranging, we get an even clearer picture.

$$\mathcal{QFT}^{(2^n)} |x\rangle^n = \mathcal{QFT}^{(2^{n-1})} |\tilde{x}\rangle^{n-1} \left(\frac{|0\rangle + \omega_{2^n}^x |1\rangle}{\sqrt{2}} \right)$$

Compare this with the Danielson-Lanczos Recursion Relation for the \mathcal{DFT} which we turned into an \mathcal{FFT} by unwinding recursion. In our current context it's even easier, because we have only one, not two, recursive calls to unwind.

If we apply the same math to the lower-order \mathcal{QFT} on the RHS and plug the result into last equation, using $\tilde{\tilde{x}}$ for the \tilde{x} mod $(N/2)$, we find

$$\mathcal{QFT}^{(2^n)} |x\rangle^n = \mathcal{QFT}^{(2^{n-2})} |\tilde{\tilde{x}}\rangle^{n-2} \left(\frac{|0\rangle + \omega_{2^{n-1}}^x |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + \omega_{2^n}^x |1\rangle}{\sqrt{2}} \right).$$

Now let recursion off its leash. Each iteration pulls a factor of $(|0\rangle + \omega_{2^k}^x |1\rangle)$ out (and to the right) of the lower-dimensional $\mathcal{QFT}^{(2^k)}$ until we get to $\mathcal{QFT}^{(2)}$, which would be the final factor on the left,

$$\mathcal{QFT}^{(2^n)} |x\rangle^n = \prod_{k=1}^n \left(\frac{|0\rangle + \omega_{2^k}^x |1\rangle}{\sqrt{2}} \right).$$

First, admire the disappearance of those pesky \tilde{x} factors, so any anxiety about x mod N/k is now lifted. Next, note that the RHS is written in terms of different roots-of-unity, $\omega_2, \omega_4, \dots, \omega_{2^n} = \omega$. However, they can all be written as powers of $\omega = \omega_N$,

$$\omega_{2^k} = \omega^{2^{n-k}}.$$

For example, when $k = 1$ we have $\omega_2 = (-1)$, which is $\omega^{2^{n-1}}$. Using this, we write the N th order \mathcal{QFT} in terms of the N th root-of-unity ω ,

$$\mathcal{QFT}^{(2^n)} |x\rangle^n = \prod_{k=1}^n \left(\frac{|0\rangle + \omega^{2^{n-k}x} |1\rangle}{\sqrt{2}} \right).$$

Interesting Observation. When $N = 2$ ($n = 1$), we know $\mathcal{QFT} = H$, so the first factor – the last one that got peeled-off on the far left – is literally $H|x_0\rangle$ (by that time we would be taking x mod 2 which is the least significant bit of x, x_0). This final, far left factor is

$$H|x_0\rangle = \frac{1}{\sqrt{2}} \left((-1)^{0 \cdot x_0} |0\rangle + (-1)^{1 \cdot x_0} |1\rangle \right).$$

and explains why the $|\tilde{x}\rangle, |\tilde{\tilde{x}}\rangle$, etc. terms disappeared from the RHS; each factor gobbled up another qubit from the tensor product, and the final, smallest, $\mathcal{QFT}^{(2)}$ turns the remaining single qubit, $|x_0\rangle$, into the Hadamard superposition.

Notation and Discussion

Products between kets are *tensor products*. Each factor in the final product expressing \mathcal{QFT}^{2^n} is a single superposition qubit consisting of a mixture of $|0\rangle$ and $|1\rangle$ in its own component space. Meanwhile, there are n of those small superpositions creating the final tensor product. It is sometimes written

$$\mathcal{QFT}^{(2^n)} |x\rangle^n = \bigotimes_{k=0}^{n-1} \left(\frac{|0\rangle + \frac{\omega_{2^{n-k}}^x |1\rangle}{\sqrt{2}}}{\sqrt{2}} \right).$$

I won't use this notation, since it scares people, and when you multiply kets by kets everyone knows that tensors are implied. So, I'll use the \prod notation for all products, and you can infuse the tensor interpretation mentally when you see that the components are all qubits.

However, it's still worth remarking that this *is* a tensor product of kets from the individual 2-dimensional \mathcal{H} spaces (of which there are n) and as such results in a *separable* state in the N -dimensional $\mathcal{H}_{(n)}$. This is a special way – different from the expansion along the CBS – to express a state in this high dimensional Hilbert space. But you should not be left with the impression that we were entitled to find a factored representation. Most states in $\mathcal{H}_{(n)}$ cannot be factored – they're not separable. The result we derived is that when taking the \mathcal{QFT} of a CBS we happily end up with a separable state.

The factored representation and the CBS expansion each give different information about the output state, and it may not always be obvious how the coefficients or factors of the two relate (without doing the math).

A simple example is the equivalence of a factored representation and the CBS expansion of the following $|\psi\rangle^2$ in a two qubit system ($n = 2, N = 4$):

$$|\psi\rangle^2 = \frac{|0\rangle|0\rangle}{\sqrt{2}} + \frac{|0\rangle|1\rangle}{\sqrt{2}} = |0\rangle \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right)$$

Here we have *both* the CBS definition of $\mathcal{QFT} |x\rangle^2$ and the separable view.

In the N -dimensional case, the two different forms can be shown side-by-side,

$$\mathcal{QFT}^{(2^n)} |x\rangle^n = \prod_{k=1}^n \left(\frac{|0\rangle + \frac{\omega_{2^{n-k}}^x |1\rangle}{\sqrt{2}}}{\sqrt{2}} \right) = \sum_{y=0}^{2^n-1} \omega^{xy} |y\rangle.$$

Of course, there can only be (at most) n factors in the separable factorization, while there will be up to 2^n terms in the CBS expansion.

The reason I bring this up is that the (separable) factorization is more relevant to the \mathcal{QFT} than it was to the \mathcal{FFT} , because we are basing our quantum work on the supposition that there will be quantum gates in the near future. These gates are unitary operators applied to the input CBS qubit-by-qubit, which is essentially a tensor product construction.

Let's see how we can construct an actual \mathcal{QFT} circuit from such unitary operators.

22.4.3 The QFT Circuit from the Math: $n = 3$ Case Study

In the case of $n = 3$, the factored QFT is

$$\begin{aligned} QFT^{(8)} |x\rangle^3 \\ = \left(\frac{|0\rangle + \omega^{4x} |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + \omega^{2x} |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + \omega^x |1\rangle}{\sqrt{2}} \right). \end{aligned}$$

Good things come to those who calmly examine each factor, separately. Work from left (most-significant output qubit) to right (least-significant output qubit).

The First (Most-Significant) Output Factor

We already know that this is $H|x_0\rangle$, but we'll want to re-derive that fact in a way that can be used as a template for the other two factors.

ω is an 8th root of unity, so the coefficient of $|1\rangle$ in the numerator of the output fraction, ω^{4x} , can be simplified to

$$\begin{aligned} \omega^{4x} &= \omega^{4(4x_2 + 2x_1 + x_0)} = \omega^{16x_2} \omega^{8x_1} \omega^{4x_0} \\ &= 1 \cdot 1 \cdot (\omega^4)^{x_0} = (-1)^{x_0}, \end{aligned}$$

which means

$$\begin{aligned} \left(\frac{|0\rangle + \omega^{4x} |1\rangle}{\sqrt{2}} \right) &= \left(\frac{|0\rangle + (-1)^{x_0} |1\rangle}{\sqrt{2}} \right) \\ &= \begin{cases} \frac{|0\rangle + |1\rangle}{\sqrt{2}}, & x_0 = 0 \\ \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & x_0 = 1 \end{cases} \\ &= H|x_0\rangle. \end{aligned}$$

This was the most-significant qubit factor of output ket (the one on the far left of the product). Let's refer to the output ket as $|\tilde{x}\rangle$ and its most significant separable factor (the one at the far left of our product) as $|\tilde{x}_2\rangle$. We can then rewrite the last equation as

$$|\tilde{x}_2\rangle = H|x_0\rangle.$$

[Don't be lulled into thinking this is a computational basis element, though. Unlike the input state, $|x\rangle = |x_2\rangle |x_1\rangle |x_0\rangle$, which *is* a product of CBS, and therefore, itself a tensor CBS, the output, $|\tilde{x}\rangle$ while a product of states, to be sure, *is not* comprised of factors which are CBS in their 2-D homes. Therefore, the product, $|\tilde{x}\rangle$, is not a CBS in the 2^n -dimensional product space.]

Summary: By expressing x as powers-of-2 in the most-significant output factor, $|\tilde{x}_2\rangle$, we were able to watch the higher-powers dissolve because they turned ω into 1. That left only the lowest power of ω , namely $\omega^4 = (-1)$, which, in turn, produced a

“Hadamard effect” on the least significant bit of the input ket, $|x_0\rangle$. We’ll do this for the other two factors with the sober acceptance that each time, fewer high powers will disappear.

But first, let’s stand back and admire our handiwork. We have an actual circuit element that generates the most significant separable factor for $QFT^{(8)}|x\rangle$:

$$|x_0\rangle \text{ ————— } \boxed{H} \text{ ————— } |\tilde{x}_2\rangle \quad \checkmark$$

First, wow. Second, do you remember that I said pulling the least-significant kets toward the right during factorization would introduce a small wrinkle? You’re looking at it. The output state’s *most*-significant factor, $|\tilde{x}_2\rangle$, is derived from the input state’s *least* significant ket, $|x_0\rangle$. Make a mental note that this will necessitate a reversal of the kets once we have the output of the circuit; following the input line for qubit $|x_0\rangle$ leads *not* to the output ket’s 0th factor, as we might have hoped, but rather to the output ket’s $(n - 1)$ st factor.

The Middle Factor

Again, we remain aware that ω is an 8th root of unity. Now the coefficient of $|1\rangle$ in the numerator of the output fraction, ω^{2x} , can be simplified, but not as much as before. It reduces as follows:

$$\begin{aligned} \omega^{2x} &= \omega^{2(4x_2 + 2x_1 + x_0)} = \omega^{8x_2} \omega^{4x_1} \omega^{2x_0} \\ 1 \cdot (\omega^4)^{x_1} (\omega^2)^{x_0} &= (-1)^{x_1} (i)^{x_0} \end{aligned}$$

which means

$$\begin{aligned} \left(\frac{|0\rangle + \omega^{2x}|1\rangle}{\sqrt{2}} \right) &= \left(\frac{|0\rangle + (-1)^{x_1}(i)^{x_0}|1\rangle}{\sqrt{2}} \right) \\ &= \begin{cases} H|x_1\rangle, & x_0 = 0 \\ \left(\frac{|0\rangle + (-1)^{x_1}(i)^{x_0}|1\rangle}{\sqrt{2}} \right), & x_0 = 1 \end{cases} \end{aligned}$$

- The good news is that if $x_0 = 0$, then the factor i^{x_0} becomes 1 and we are left with a Hadamard operator applied to the middle input ket, $|x_1\rangle$.
- The bad news is that if $x_0 = 1$, we see no obvious improvement in the formula.

Fixing the bad news is where I need you to focus all your attention and patience, as it is the key to everything and takes only a few more neurons. Let’s go ahead and take $H|x_1\rangle$, regardless of whether x_0 is 0 or 1. If x_0 was 0, we guessed right, but if it was 1, what do we have to do to patch things up? Not much, it turns out.

Let’s compare the actual state we computed (wrong, if x_0 was 1) with the one we wanted (right, no matter what) and see how they differ. Writing them in coordinate

form will do a world of good.

What we got when applying H to $|x_1\rangle \dots$:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1^{x_1} \end{pmatrix}$$

... but if $x_0 = 1$, we really wanted:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1^{x_1} \cdot i \end{pmatrix}$$

How do we transform

$$\begin{pmatrix} 1 \\ -1^{x_1} \end{pmatrix} \mapsto \begin{pmatrix} 1 \\ -1^{x_1} \cdot i \end{pmatrix} ?$$

Answer: multiply by

$$\begin{aligned} R_1 &\equiv \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} : \\ \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \begin{pmatrix} 1 \\ -1^{x_1} \end{pmatrix} &= \begin{pmatrix} 1 \\ -1^{x_1} \cdot i \end{pmatrix}. \end{aligned}$$

Now we have the more pleasant formula for the second factor,

$$\left(\frac{|0\rangle + \omega^{2x} |1\rangle}{\sqrt{2}} \right) = \begin{cases} H |x_1\rangle, & x_0 = 0 \\ R_1 H |x_1\rangle, & x_0 = 1 \end{cases}$$

A Piece of the Circuit

We found that the two most-significant factors of the (happily separable) $QFT|x\rangle$ could be computed using the formulas

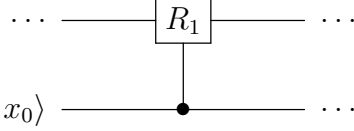
$$\begin{aligned} |\tilde{x}_2\rangle &= H |x_0\rangle, \quad \text{and} \\ |\tilde{x}_1\rangle &= \begin{cases} H |x_1\rangle, & x_0 = 0 \\ R_1 H |x_1\rangle, & x_0 = 1. \end{cases} \end{aligned}$$

In words:

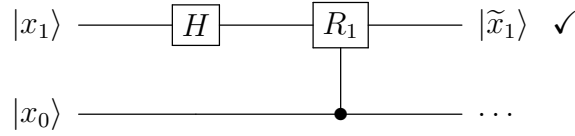
1. We apply H to the two *least* significant kets, $|x_0\rangle$ and $|x_1\rangle$, unconditionally, since they will always be used in the computation of the final two most-significant factors of $QFT|x\rangle$.
2. We *conditionally* apply another operator, R_1 , to the result of $H|x_1\rangle$ in the eventuality that $x_0 = 1$.
3. Although we apply all this to the two least significant input kets, $|x_1\rangle|x_0\rangle$, what we get is the *most*-significant portion of the output state's factorization, $|\tilde{x}_2\rangle|\tilde{x}_1\rangle$ (not the least-significant, so we must be prepared to do some swapping before the day is done).

Item 2 suggests using a *controlled- R_1* gate, where bit x_0 is the control. If $x_0 = 0$, the operator being controlled is not applied, but if $x_0 = 1$, it is. Here's the schematic for that piece:

This leads to the following circuit element:

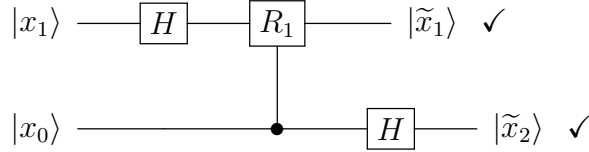


Let's add the remaining components one-at-a-time. First, we want to apply the unconditional Hadamard gate to $|x_1\rangle$. As our formulas indicate, this done *before* R_1 , ($R_1 H |x_1\rangle$ is applied right-to-left). Adding this element, we get:



This computes our final value for the $|\tilde{x}_1\rangle$ factor.

We have yet to add back in the Hadamard applied to $|x_0\rangle$. Here the order is important. We have to make sure we do this *after* x_0 is used to control x_1 's R_1 gate. Were we to apply H to x_0 before using it to control R_1 , x_0 would no longer be there – it would have been replaced by the Hadamard superposition. So we place the H -gate to the right of the control vertex:



That completes the circuit element for the two most-significant separable output factors. We can now get back to analyzing the logic of our instructional $n = 3$ case and see how we can incorporate the last of our three factors.

The Last (Least-Significant) Factor

The rightmost output factor, $|\tilde{x}_0\rangle$, has an ω in its numerator with the exponent x . Not $2x$. Not $4x$, just plain x . So we're not going to be able to simplify any of its factors away:

$$\begin{aligned}\omega^x &= \omega^{(4x_2 + 2x_1 + x_0)} = \omega^{4x_2} \omega^{2x_1} \omega^{x_0} \\ (\omega^4)^{x_2} (\omega^2)^{x_1} (\omega)^{x_0} &= (-1)^{x_2} (i)^{x_1} (\omega)^{x_0}\end{aligned}$$

which means

$$\begin{aligned}\left(\frac{|0\rangle + \omega^x |1\rangle}{\sqrt{2}}\right) &= \left(\frac{|0\rangle + (-1)^{x_2} (i)^{x_1} (\omega)^{x_0} |1\rangle}{\sqrt{2}}\right) \\ &= \begin{cases} \left(\frac{|0\rangle + (-1)^{x_2} (i)^{x_1} |1\rangle}{\sqrt{2}}\right), & x_0 = 0 \\ \left(\frac{|0\rangle + (-1)^{x_2} (i)^{x_1} (\omega)^{x_0} |1\rangle}{\sqrt{2}}\right), & x_0 = 1 \end{cases}\end{aligned}$$

This time, while the output factor does not reduce to something as simple as a $H|x_2\rangle$ in any of the cases, when $x_0 = 0$, it *does* look like the expression we had for the middle factor, except applied here to $|x_2\rangle|x_1\rangle$ rather than the $|x_1\rangle|x_0\rangle$ of the middle factor. In other words, when $x_0 = 0$ this least significant factor reduces to

$$\left(\frac{|0\rangle + (-1)^{x_2}(i)^{x_1}|1\rangle}{\sqrt{2}} \right),$$

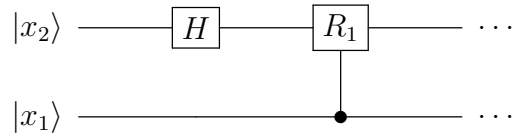
while the middle factor was *in its entirety*

$$\left(\frac{|0\rangle + (-1)^{x_1}(i)^{x_0}|1\rangle}{\sqrt{2}} \right).$$

This suggests that, if $x_0 = 0$, we apply the *same exact* logic to $|x_2\rangle|x_1\rangle$ that we used for $|x_1\rangle|x_0\rangle$ in the middle case. That logic would be (if $x_0 = 0$)

$$\left(\frac{|0\rangle + \omega^x|1\rangle}{\sqrt{2}} \right) = \begin{cases} H|x_2\rangle, & x_1 = 0 \\ R_1 H|x_2\rangle, & x_1 = 1 \end{cases}$$

Therefore, *in the special case where* $x_0 = 0$, the circuit that works for $|\tilde{x}_0\rangle$ looks like the one that worked for $|\tilde{x}_1\rangle$, applied, this time, to qubits 1 and 2:



To patch this up, we have to adjust for the case in which $x_0 = 1$. The state we just generated with this circuit was

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1^{x_2} \cdot (i)^{x_1} \end{pmatrix}$$

... but if $x_0 = 1$, we really wanted:

$$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1^{x_2} \cdot (i)^{x_1} \cdot (\omega)^{x_0} \end{pmatrix}$$

How do we transform

$$\begin{pmatrix} 1 \\ -1^{x_2} \cdot (i)^{x_1} \end{pmatrix} \mapsto \begin{pmatrix} 1 \\ -1^{x_2} \cdot (i)^{x_1} \cdot (\omega)^{x_0} \end{pmatrix}?$$

Answer: multiply by

$$R_2 \equiv \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} :$$

$$\begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} \begin{pmatrix} 1 \\ -1^{x_2} \cdot (i)^{x_1} \end{pmatrix} = \begin{pmatrix} 1 \\ -1^{x_2} \cdot (i)^{x_1} \cdot (\omega)^{x_0} \end{pmatrix}.$$

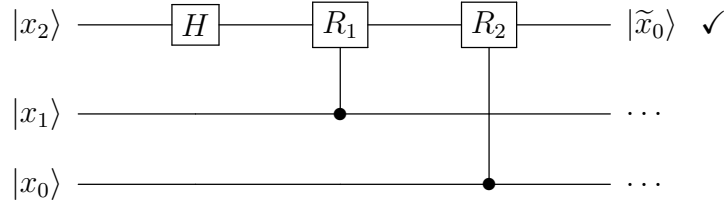
(Remember, this is in the case when $x_0 = 1$, so $\omega = \omega^{x_0}$.) This gives us the complete formula for the rightmost output factor, $|\tilde{x}_0\rangle$,

$$\left(\frac{|0\rangle + \omega^x |1\rangle}{\sqrt{2}} \right) = \begin{cases} H|x_2\rangle, & x_0 = 0, x_1 = 0 \\ R_1 H|x_2\rangle, & x_0 = 0, x_1 = 1 \\ \text{-----} \\ R_2 H|x_2\rangle, & x_0 = 1, x_1 = 0 \\ R_2 R_1 H|x_2\rangle, & x_0 = 1, x_1 = 1 \end{cases}$$

In words, we took the tentative circuit that we designed for $|\tilde{x}_0\rangle$ under the assumption that $x_0 = 0$ but tagged on a correction if $x_0 = 1$. That amounts to our newly introduced operator R_2 controlled by x_0 , so that the result would be further multiplied by R_2 .

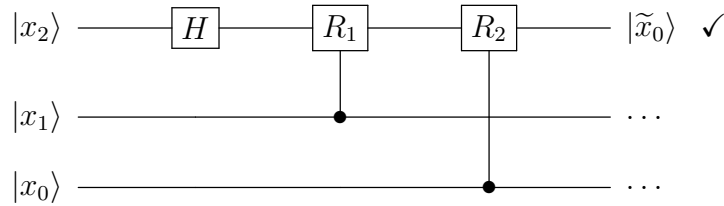
The Circuit Element for $|\tilde{x}_0\rangle$, in Isolation

As long as we only consider $|\tilde{x}_0\rangle$, we can easily use this new information to patch up the most recent $x_0 = 0$ case. We simply add an $|x_0\rangle$ at the bottom of the picture, and use it to control an R_2 - gate, applied to the end of the $|x_2\rangle \rightarrow |\tilde{x}_0\rangle$ assembly line.

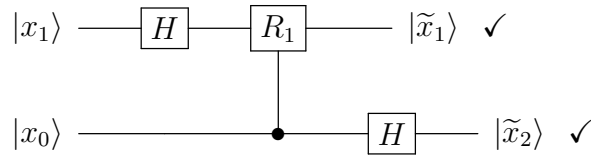


The Full Circuit for $N = 8$ ($n = 3$)

In the previous section, we obtained the exact result for the least-significant output factor, $|\tilde{x}_0\rangle$.

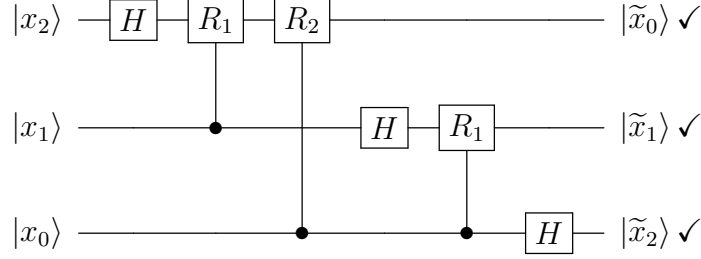


In the section prior, we derived the circuit for output two most significant factors, $|\tilde{x}_1\rangle$ and $|\tilde{x}_2\rangle$

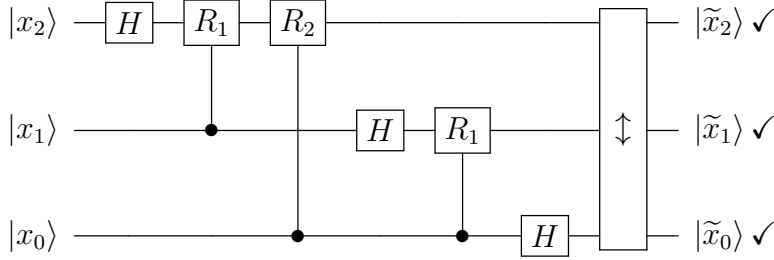


All that's left to do is combine them. The precaution we take is to defer applying any operator to an input ket until *after* that ket has been used to control any R -gates

needed by its siblings. That suggests that we place the $|\tilde{x}_1\rangle |\tilde{x}_2\rangle$ circuit elements to the right of the $|\tilde{x}_0\rangle$ circuit element, and so we do.



Prior to celebration, we have to symbolize the somewhat trivial circuitry for re-ordering the output. While trivial, it has a linear (in $n = \log N$) cost, but it adds nothing to the time complexity, as we'll see.



You are looking at the complete QFT circuit for a 3-qubit system.

Before we leave this case, let's make one notational observation. We defined R_1 to be the matrix that “patched-up” the $|\tilde{x}_1\rangle$ factor, and R_2 to be the matrix that “patched-up” the $|\tilde{x}_0\rangle$ factor. Let's look at those gates along with the only other gate we needed, H :

$$R_2 \equiv \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} \quad R_1 \equiv \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

$$H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

The lower right-hand element of each matrix is a root-of-unity, so let's see all three matrices again, this time with that lower right element expressed as a power of ω :

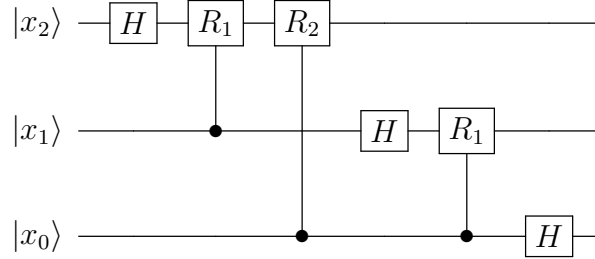
$$R_2 \equiv \begin{pmatrix} 1 & 0 \\ 0 & \omega \end{pmatrix} \quad R_1 \equiv \begin{pmatrix} 1 & 0 \\ 0 & \omega^2 \end{pmatrix}$$

$$H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & \omega^4 \end{pmatrix}$$

This paves the way to generalizing to a QFT of any size.

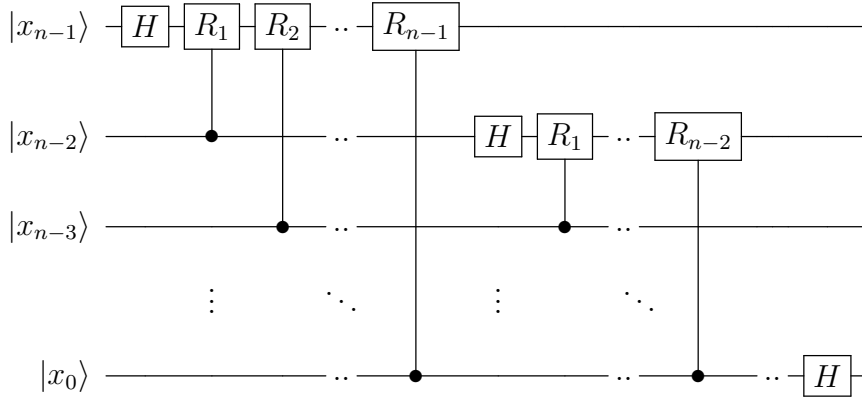
22.4.4 The QFT Circuit from the Math: General Case

Minus the re-ordering component at the far right, here's the QFT circuit we designed for $n = 3$:



[**Exercise.** Go through the steps that got us this circuit, but add a fourth qubit to get the $n = 4$ ($QFT^{(16)}$) circuit.]

It doesn't take too much imagination to guess what the circuit would be for any n :



That's good and wonderful, but we have not defined R_k for $k > 2$ yet. However, the final observation of the $n = 3$ case study suggested that it should be

$$R_k \equiv \begin{pmatrix} 1 & 0 \\ 0 & \omega^{2^{n-k-1}} \end{pmatrix}.$$

You can verify this by analyzing it formally, but it's easiest to just look at the extreme cases. No matter what n is, we want R_1 's lower-right element $= i$, and R_{n-1} 's to be ω (compare our $n = 3$ case study directly above), and you can verify that for $k = 1$ and $k = n - 1$, that's indeed what we get.

Well, we have defined and designed the QFT circuit out of small, unitary gates. Since you'll be using QFT in a number of circuit designs, you need to be able to cite its computational complexity, which we do now.

22.5 Computational Complexity of QFT

This is the easy part because we have the circuit diagrams to lean on.

Each gate is a single unitary operator. Some are 2-qubit gates (the controlled- R_k gates) and some are single qubit gates (the H gates). But they are all constant time and constant size, so we just add them up.

The topmost input line starting at $|x_{n-1}\rangle$ has n gates (count the *two-qubit* controlled- R_k s as single gates in this top line, but you don't have to count their control nodes when you get to them in the lines, below). As we move down to the lower lines observe that each line has one-fewer gates than the one above until we get to the final line, starting at $|x_0\rangle$, which has only one gate. That's

$$\sum_{k=1}^n k = \frac{n(n+1)}{2} \text{ gates.}$$

The circuit complexity for this is $O(n^2)$. Adding on a circuit that reverses the order can only add an additional $O(n)$ gates, but in series, not nested, so that does not affect the circuit complexity. Therefore, we are left with a computational factor of

$$O(n^2) = O(\log^2 N).$$

You might be tempted to compare this with the $O(N \log N)$ performance of the \mathcal{FFT} , but that's not really an apples-to-apples comparison, for several reasons:

1. Our circuit computes $\mathcal{QFT}|x\rangle^n$ for only *one* of the $N = 2^n$ basis states. We'd have to account for the algorithm time required to repeat N passes through the circuit which (simplistically) brings it to $O(N \log^2 N)$. While this can be improved, the point remains: our result above would have to be multiplied by *something* to account for all N output basis states.
2. If we were thinking of using the \mathcal{QFT} to compute the \mathcal{DFT} , we'd need to calculate the N complex amplitudes, $\{\tilde{c}_k\}$ from the inputs $\{c_k\}$. They don't appear in our analysis because we implicitly considered the special N amplitudes $\{\delta_{xy}\}_{y=0}^{N-1}$ that define the CBS. Fixing this is feels like $O(N)$ proposition.
3. Even if we could repair the above with clever redesign, we have the biggest obstacle: the output coefficients – which hold the \mathcal{DFT} information – are amplitudes. Measuring the output state collapses them destroying their quantum superposition.

Although we cannot (yet) use the \mathcal{QFT} to directly compute a \mathcal{DFT} with growth smaller than the \mathcal{FFT} , we can still use it to our advantage in quantum circuits, as we will soon discover.

[Accounting for Precision.] You may worry that increasingly precise matrix multiplies will be needed in the 2×2 unitary matrices as n increases. This is a valid concern. Fixed precision will only get us so far until our ability to generate and compute with increasingly higher roots-of-unity will be tapped out. So the *constant time* unitary gates are relative to, or “above,” the primitive complex (or real) multiplications and additions. We would have to make some design choices to either limit n

to a maximum useable size or else account for these primitive arithmetic operations in the circuit complexity. We'll take the first option: our n will remain below some maximum n_0 , build our circuitry and algorithm to be able to handle adequate precision for that n_0 and all $n \leq n_0$. This isn't so hard to do in our current problem since n never gets too big: $n = \log_2 N$, where N is the true size of our problem, so we won't be needing arbitrarily large n s in practice. If that doesn't work for us in some future problem, we can toss in the extra complexity factors and they will usually still produce satisfactory polynomial *big-O*s for problems that are classically exponential.]

Further Improvements

I'll finish by mentioning, without protracted analysis, a couple ways the above circuits can be simplified and/or accelerated:

- If we are willing to destroy the quantum states and measure the output qubits immediately after performing the QFT (something we *are* willing to do in most of our algorithms), then the two-qubit (controlled- R_k) gates can be replaced with 1-qubit gates. This is based on the idea that, rather than construct a controlled- R_k gate, we instead measure the controlling qubit first and then apply R_k based on the outcome of that measurement. This sounds suspicious, I know: we're still doing a conditional application of a gate. However, a controlled- R_k does not destroy the controlling qubit and contains all the conditional logic inside the quantum gate, whereas measuring a qubit and then applying a 1-qubit gate based on its outcome, moves the controlling aspect from inside the quantum gate to the outer classical logic. It is much easier to build stable conditioned *one-qubit* gates than *two-qubit* controlled gates. Do note, however, that this does not improve the computational complexity.
- If m -bit accuracy is enough, where $m < n$, then we get improved complexity. In this case, we just ignore the least-significant $(n - m)$ output qubits. That amounts to tossing out the top $(n - m)$ lines, leaving only m channels to compute. The new complexity is now $O(m^2)$ rather than $O(n^2)$.

Chapter 23

Shor's Algorithm

23.1 The Role of Shor's Algorithms in Computing

23.1.1 Context for The Algorithms

Shor's algorithms are the crown jewels of elementary quantum information theory. They demonstrate that a quantum computer, once realized, will be able handle some practical applications that are beyond the reach of the fastest existing super computers, the most dramatic being the factoring of large numbers.

As you may know from news sources or academic reports, the inability of computers to factor astronomically large integers on a human timescale is the key to RSA encryption, and RSA encryption secures the Internet. Shor's *quantum factoring algorithm* should be able to solve the problem in minutes or seconds and would be a disruptive technology should a quantum computer be designed and programmed to implement it. Meanwhile, quantum encryption, an advanced topic that we study in the next course, offers a possible alternative to Internet security that could replace RSA when the time comes.

23.1.2 Period Finding and Factoring

There are many ways to present Shor's results, and it is easy to become confused about what they all say. We'll try to make things understandable by dividing our study into two parts adumbrated by two observations.

1. Shor's algorithm for *period-finding* is a *relativized* (read "not absolute") exponential speed-up over a classical counterpart. Like Simon's algorithm, there are periodic functions that do not have polynomial-fast oracles. In those cases the polynomial complexity of the {circuit + algorithm} *around* the oracle will not help.
2. Shor's algorithm for *factoring* not only makes use of the period-finding algo-

rithm, but also provides an oracle for a specific function that *is* polynomial-time, making the entire {circuit + algorithm} an *absolute* exponential speed-up over the classical version.

23.1.3 The Period Finding Problem and its Key Idea

Informal Description of Problem

Shor’s period-finding algorithm seeks to find the *period*, a , of a *periodic function* $f(k)$ whose domain is a finite set of integers, i.e., $k \in \mathbb{Z}_M = \{0, 1, 2, \dots, M-1\}$. Typically M can be very large; we can even consider the domain to be all of \mathbb{Z} , although the \mathbb{Z}_M case will be shown to be an equivalent finitization that makes the problem tractable. I’ll state this explicitly in a couple screens. Here, we want to get a general picture of the problem and the plan.

First, for this to even make sense a has to be less than M so that such periodicity is “visible” by looking at the M domain points available to us.

Second, this kind of periodicity is more “traditional” than Simon’s, since here we use ordinary addition to describe the period a via $f(x) = f(x + a)$ rather than the exotic *mod-2* periodicity of Simon, in which $f(x) = f(x \oplus a)$.

Shor’s Circuit Compared to Simon’s Circuit

We’ll see that the circuit and analysis have the same general framework as that of Simon’s algorithm with the main difference being a post-oracle *QFT* gate rather than a post-oracle *Hadamard* gate. What’s the big idea behind this change? At the highest level of analysis it is quite simple even if the details are thorny.

If we put a maximally mixed superposition into the oracle as we have done for all previous algorithms (and will do here) then a post-oracle measurement in the standard basis should give all possible values of f with equal probabilities. That won’t do. We have to measure along a basis that will be biased toward giving information. For Simon, that happened to be the x -basis, thus the use of a final Hadamard operator. What is the right basis in which to measure when we are hoping to discover an integer function’s *period*?

Why the *QFT* Will Help

Our lectures on *classical Fourier series and transforms* had many ideas and consequences of which I’d like to revive two.

1. *Period*, T , and *frequency*, f (not the function, $f(x)$, but its frequency) are related by

$$T \cdot f = \text{constant},$$

where the constant is usually 1 or 2π for continuous functions and the vector size, M , for discrete functions.

2. If a discrete function is periodic, its *spectrum* $\mathcal{DFT}(f)$ will have values which are mostly small or zero except at domain points that are multiples of the frequency (See Figure 23.1).

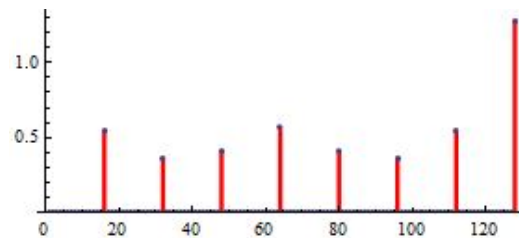


Figure 23.1: The spectrum of a vector with period 8 and frequency $16 = 128/8$

In very broad – and slightly inaccurate – terms, this suggests we query the spectrum of our function, $f(x)$, ascertain its fundamental frequency, m , (the first non-zero spike) and from it get the period, $a = M/m$.

But what does it mean to “query the frequency?” That’s code for “take a post-oracle measurement in the *Fourier basis*.” We learned that measuring along a non-preferred basis is actually applying the operator that converts the preferred basis to the alternate basis, and for frequencies of periodic functions, that gate is none other than the \mathcal{QFT} .

Well this sounds easy, and while it may motivate the use of the \mathcal{QFT} , figuring out how to use it and what to test – that will consume the next two weeks.

How We’ll Set-Up the State Prior to Applying the \mathcal{QFT}

Another thing we saw in our Fourier lectures was that we get the cleanest, easiest-to-analyze spectrum of a periodic function when we start with a *pure* periodic function in the spatial (or time) domain, and then apply the transform. In the continuous case that was a sinusoid or exponential, e.g., $\sin 3x$ (Figure 23.2).

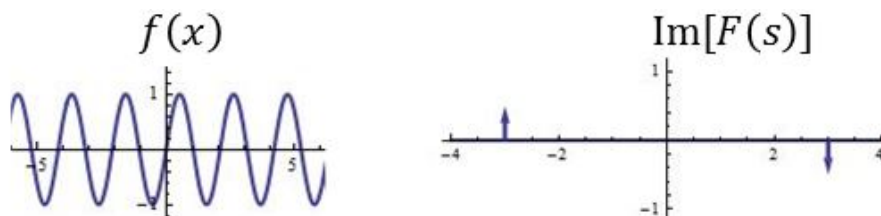


Figure 23.2: $\sin(3x)$ and its spectrum

In the discrete case it was a function that was zero everywhere except for a single $k < a$ and its multiples, like

```
( 0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0,
  0, 0, 0, .25, 0, 0, 0, 0 ) .
```

Such overtly periodic vectors have \mathcal{DFT} s in which all the non-zero frequencies in the spectrum have the same amplitudes as shown in Figure 23.3.

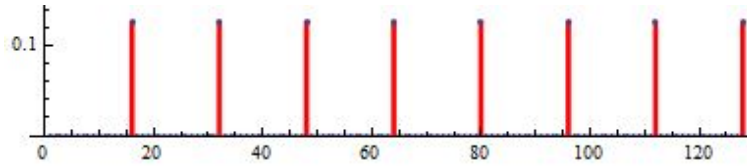


Figure 23.3: The spectrum of a purely periodic vector with period 8 and frequency $16 = 128/8$

We will process our original function so that it produces a *purely periodic* cousin with the same period by

1. putting a maximally mixed state into the oracle's A register to enable *quantum parallelism*, and
2. “conceptually” collapsing the superposition at the output of the oracle by taking a B register measurement and applying the *generalized Born rule*.

This will leave a “pure” periodic vector in the A register which we can send through a post processing \mathcal{QFT} gate and, finally, measure. We may have to do this more than once, thereby producing several measurements. To extract m , and thus a , from the measurements we will apply some beautiful mathematics.

The Final Approach

After first defining the kind of periodicity that Shor's work addresses, we begin the final leg of our journey that will take us through some quantum and classical terrain. When we're done, you will have completed the first phase in your study of quantum computation and will be ready to move on to more advanced topics.

The math that accompanies Shor's algorithms is significant; it spans areas as diverse as Fourier analysis, number theory, complex arithmetic and trigonometry.

We have covered each of these subjects completely. If you find yourself stuck on some detail, please search the table of contents in this volume for a pointer to the relevant section.

23.2 Injective Periodicity

23.2.1 Functions of the Integers

The kind of periodicity Shor’s algorithm addresses can be expressed in terms of functions defined over *all* the integers, \mathbb{Z} , or over just a finite group of integers like \mathbb{Z}_M . The two definitions are equivalent, but it helps to define periodicity both ways so we can speak freely in either dialect. Here we’ll consider all integers, and in the next subsection we’ll deal with \mathbb{Z}_M .

We’ve discussed ordinary periodicity, like that of a function $\sin x$, as well as $(\mathbb{Z}_2)^n$ periodicity, studied in Simon’s algorithm. Shor’s periodicity is much closer to ordinary periodicity, but has one twist that gives it a pinch of Simon’s more exotic variety.

A function defined on \mathbb{Z} ,

$$f: \mathbb{Z} \longrightarrow S, \quad S \subset \mathbb{Z},$$

*is called **periodic injective** if there exists an integer $a > 0$ (called the **period**), such that*

for all $x \neq y$ in \mathbb{Z} , we have

$$f(x) = f(y) \iff y = x + ka, \text{ some integer } k.$$

The term “injective” will be discussed shortly. Because of the *if and only if* (\Leftrightarrow) in the definition, we don’t need to say “smallest” or “unique” for a . Those conditions follow naturally.

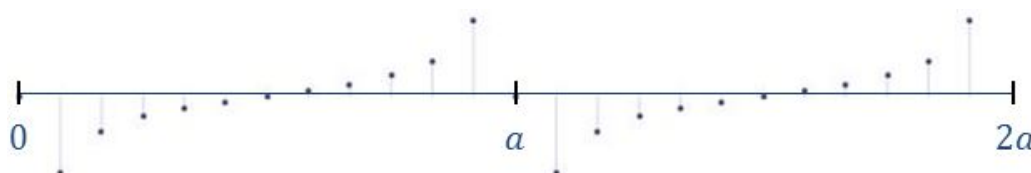


Figure 23.4: Graph of two periods of a *periodic injective* function

Caution: Some authors might call f “ a -periodic” to make the period visible, omitting the reference to injectivity. Others, might just call f “periodic” and let you fend for yourselves.

In theory, S , the range of f , can be any set: integers, sheep or neutrinos. It’s the periodicity that matters, not what the functional values are. Still, we will typically consider the range to be a subset of \mathbb{Z} , most notably, \mathbb{Z}_{2^r} (for some positive integer r). This will make our circuit analysis clearer.

23.2.2 Functions of the Group \mathbb{Z}_M

A little consideration of the previous definition should convince you that any periodic injective function with period a can be confined to a finite subset of \mathbb{Z} which contains the interval $[0, a)$. To “feel” f ’s periodicity, though, we’d want M to contain at least a few “copies of a ” inside it, i.e., we would like $M > 3a$ or $M > 1000a$. It helps if we assume that we do know such an M , even if we don’t know the period, a , and let f be defined on \mathbb{Z}_M , rather than the larger \mathbb{Z} . The definition of *periodic injective* in this setting would be as follows.

A function defined on \mathbb{Z}_M ,

$$f: \mathbb{Z}_M \longrightarrow S, \quad S \subset \mathbb{Z}_M,$$

is called **periodic injective** if there exists an integer $a \in \mathbb{Z}_M$ (called the **period**), such that

for all $x \neq y$ in \mathbb{Z}_M , we have

$$f(x) = f(y) \iff y = x + ka, \text{ some integer } k.$$

As before, S can be any set, but we’ll be using \mathbb{Z}_{2^r} .

The phrase

$$“y = x + ka”$$

uses ordinary addition, not mod- M addition. We are not saying that we can let $x + ka$ wrap around M back to $0, 1, 2, \dots$ and find more numbers that are part of the $f(x) = f(y)$ club. For example, the function

$$f(x) = x \% 8$$

is *periodic injective* with period 8 according to either definition. However, when we look at the second definition and take, say, $M = 20$ as the integer that we proclaim to be larger than a defining the domain \mathbb{Z}_{20} , then only the numbers $\{3, 11, 19\}$ which are of the form $3 + k8$ should be considered pre-images of 3. Were we were to allow $3 + k8 \pmod{20}$, then the members of this club would grow (illogically) to $\{3, 11, 19, 7, 15\}$. This does not track; while $f(3) = f(11) = f(19) = 3$, it is *not* true of $f(7)$, or $f(15)$, both of which are 7.

23.2.3 Discussion of Injective Periodicity

The term *injective* is how mathematicians say “1-to-1”. Also, *periodic injective* is seen in the quantum computing literature, so I think its worth using (rather than the rag-tag 1-to-1-periodicity, which is a hyphen extravaganza). But *what*, exactly, are we claiming to be *injective*? A periodic function is patently non-injective, mapping

multiple domain points to the same image value. Where is there 1-to-1-ness? It derives from the following fact which is a direct consequence of the definition:

The *if-and-only-if* (\Leftrightarrow) condition in the definition of *periodic injective* implies that, when restricted to the set $[0, a) = \{0, 1, 2, \dots, a-1\}$, f is 1-to-1. The same is true of any set of $\leq a$ consecutive integers in the domain.

[Exercise. Prove it.]

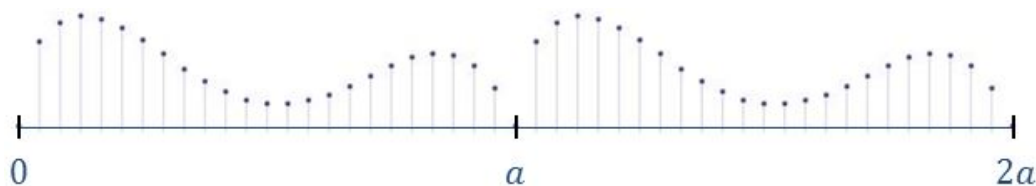


Figure 23.5: Example of a periodic function that is **not** periodic *injective*

This is the twist I promised had some kinship to Simon’s periodic functions. Recall that they were also 1-to-1 on each of their two disjoint cosets R and Q in that conversation. The property is required in our treatment of Shor’s period-finding as well; we must be able to partition f ’s domain into disjoint sets on which f is 1-to-1. This is not to say that the property is necessary in order for Shor’s algorithm to work (there may be more general results that work for vanilla flavored periodicity). However, the majority of historical quantum period-finding proofs make use of injective periodicity, whether they call it that, or not. For factoring and encryption-breaking, that condition is always met.

While we’re comparing Simon to Shor, let’s discuss a difference. In Simon’s case, if we found *even one pair* of elements, $x' \neq y'$ with $f(x') = f(y')$, then we knew a . However, the same cannot be said of Shor’s problem. All we would know in the current case is that x' and y' differ by a *multiple* of a , but we would know neither a nor the multiple.

23.3 Shor’s Periodicity Problem

Statement of Shor’s Periodicity Problem

Let $f : \mathbb{Z}_M \longrightarrow \mathbb{Z}$ be injective periodic.
Find a .

The M in the problem statement is not about the periodicity (a describes that) as much as it is about how big the problem is; it gives us a bound on a . It is M which is used to measure the computational complexity of the algorithm; how does the {algorithm + circuit} grow as M gets larger?

Relativized vs. Absolute Speed-Up

I feel compelled to say this again before we hit-the-road. Our goal is to produce a quantum algorithm that completes in polynomial time when its classical counterpart has exponential complexity. The time complexity of Shor's *period-finding algorithm* will end up being limited by *both* the circuits/algorithms we build around our oracle U_f , *as well as* f , itself. So if f cannot be computed in polynomial time, at least by some quantum circuit, we won't end up with an *easy* algorithm. We will prove all the circuits and algorithms *around* U_f to be of polynomial complexity, specifically $O(\log^3 M)$. We will also show (in a future lesson) that the f needed for RSA encryption-breaking is $O(\log^4 M)$, so the *factoring problem* is *quantum-easy* compared with a *hard* problem using classical circuits/algorithms.

The point is that there are two problems: *period-finding*, which has quantum relativized speed-up and *factoring*, which has quantum absolute speed-up. Our job is to make sure that the quantum machinery around U_f is “polynomial,” so that when f , itself, is polynomial (as is the case in factoring) we end up with a problem that is *absolutely easy* in the quantum realm.

One Convenient Assumptions

In the development of the algorithm it will also help to add the very weak assumption

$$a < M/2,$$

i.e., the periodicity cycles at least twice in the interval $[0, M - 1]$.

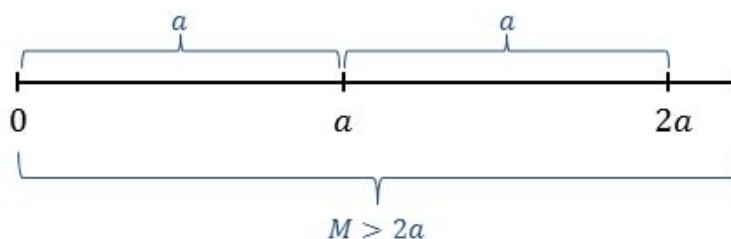


Figure 23.6: We add the weak assumption that 2(+) a -intervals fit into $[0, M)$

In fact, when we apply Shor's periodicity algorithm to RSA encryption-breaking or the factoring problem, this added assumption will automatically be satisfied. Usually, we have an even stronger assumption, namely that $a \ll M$.

In the general case, we'll see that even if we are only guaranteed that $a \leq .9999M$, we still end up proving that Shor's problem is solvable in polynomial time by a quantum circuit. But there's no reason to be so conservative. Even $a < M/2$ is overdoing it for practical purposes, yet it makes all of our estimates precise and easy to prove without any hand-waving. Besides, it costs us nothing algorithmically, as we shall see.

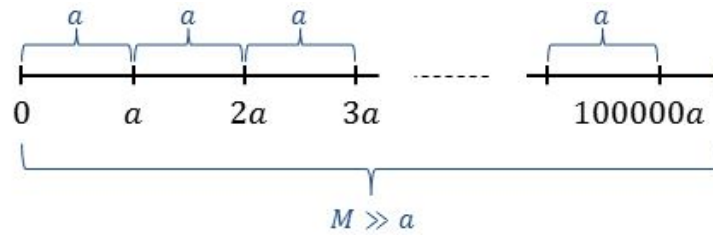


Figure 23.7: Typical application provides many a -intervals in $[0, M)$

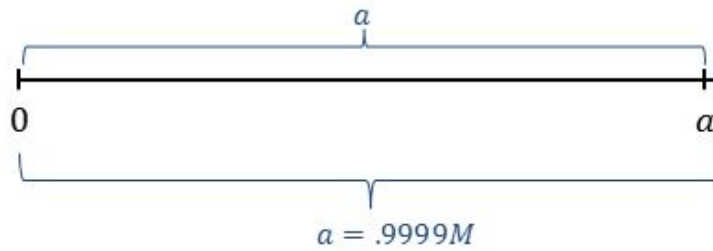


Figure 23.8: Our proof will also work for only one a interval in $[0, M)$

23.3.1 Definitions and Recasting the Problem

As stated, the problem has relatively few moving parts: an unknown period, a , a known upper bound for a , M , and the injective periodicity property. To facilitate the circuit and algorithm, we'll have to add a few more letters: n , N and r . Here are their definitions.

A Power of 2, 2^n , for the Domain Size

Let n be the exponent that establishes smallest power-of-2 equal to or above M^2 ,

$$2^{n-1} < M^2 \leq 2^n.$$

We'll use the integer interval $[0, 1, \dots, 2^n - 1]$ as our official domain for f , and we'll let N be the actual power-of-2,

$$N \equiv 2^n.$$

Since $M > 2a$ we are guaranteed that $[0, N-1]$ will contain at least as many intervals of size a within it.

[You're worried about how to define f beyond the original domain limit M ? Stop worrying. It's not our job to define f , just discover its period. We know that f is periodic with period a , even though we don't know a yet. That means its definition can be extended to all of \mathbb{Z} . So we can take any size domain we want. Stated another way, we assume our oracle can compute $f(x)$ for any x .]

The reason for bracketing M^2 like this only becomes apparent as the plot unfolds. Don't be intimidated into believing anyone could predict we would need these exact

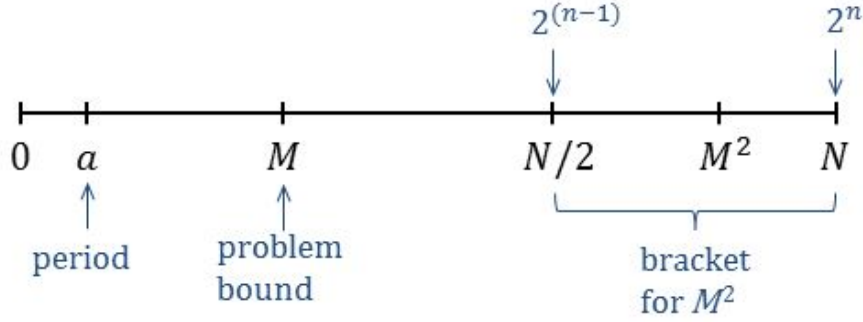


Figure 23.9: $N = 2^n$ chosen so $(N/2, N]$ bracket M^2

bounds so early in the game. Even the pioneers certainly got to the end of the derivation and noticed that these limits would be needed, then came back and added them up front. That's exactly how you will do it when you write up the quantum algorithms that you discover.

The bracketing of M^2 , when written in terms of N , looks like

$$\frac{N}{2} < M^2 \leq N.$$

A Power of 2, 2^r , for the Range Size

Also, without loss of generality, we assume that the range of f is a subset of \mathbb{Z}_{2^r} for some sufficiently large $r > 0$, i.e.,

$$\begin{aligned} \text{ra}(f) &\subseteq [0, 2^r - 1], \quad \text{also written} \\ 0 &\leq f(x) < 2^r. \end{aligned}$$

Summarizing all this,

$$\begin{aligned} f : [0, 2^n - 1] &\longrightarrow [0, 2^r - 1], \quad \text{also written} \\ f : [0, N - 1] &\longrightarrow [0, 2^r - 1], \quad N \equiv 2^n. \end{aligned}$$

The Equivalence of N and M in Time Complexity

Let's be certain that using N for the time complexity is the same as using M . Taking the log (base 2) of the inequality that brackets M^2 , we get

$$\begin{aligned} \log \frac{N}{2} &< \log M^2 \leq \log N, \quad \text{or} \\ \log N - \log 2 &< 2 \log M \leq \log N. \end{aligned}$$

The *big-O* of every expression in this equation will kill the constants and weaken $<$ to \leq , producing

$$O(\log N) \leq O(\log M) \leq O(\log N),$$

and since the far left and far right are equal, they both equal middle, i.e.

$$O(\log N) = O(\log M).$$

Therefore, a growth rate of any polynomial function of these two will also be equal,

$$O(\log^p N) = O(\log^p M).$$

Thus, bracketing M^2 between $N/2$ and N allows us to use N to compute the complexity and later replace it with M . Specifically, we'll eventually compute a *big-O* of $\log^3 N$ for Shor's algorithm, implying a complexity of $\log^3 M$.

23.3.2 The $\mathbb{Z}_N - (\mathbb{Z}_2)^n - \text{CBS Connection}$

We're all full of energy and eager to start wiring this baby up, but there is one final precaution we should take lest we find ourselves adrift in a sea of math. On the one hand, the problem lives in the world of ordinary integer arithmetic. We are dealing with simple functions and sums like,

$$\begin{aligned} f(18) &= 6, \quad \text{and} \\ 6 + 7 &= 13. \end{aligned}$$

On the other hand, we will be working with a quantum circuit which relies on mod-2 arithmetic, most notably the oracle's B register output,

$$|y \oplus f(x)\rangle^r,$$

or, potentially more confusing, ordinary arithmetic inside a ket, as in the expression

$$|x + ja\rangle^n.$$

There's no need to panic. The simple rule is that when you see $+$, use ordinary addition and when you see \oplus use *mod-2*.

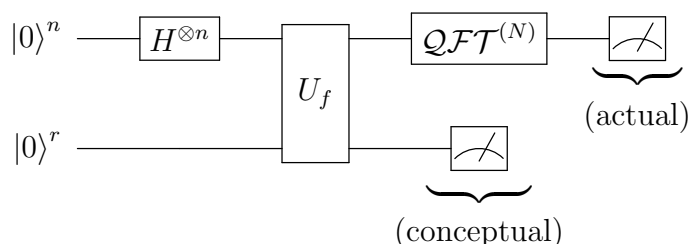
- $|y \oplus f(x)\rangle^r$. We're familiar with the *mod-2 sum* and its use inside a ket, especially when we are expressing U_f 's target register. The only very minor adjustment we'll need to make arises from the oracle's B channel being r qubits wide (where 2^r is f 's *range* size) instead of the same n qubits of the oracle's A channel (where 2^n is f 's *domain* size). We'll be careful when we come to that.
- $|x + ja\rangle^n$. As for ordinary addition inside the kets, this will come about when we partition the *domain* into mutually exclusive "cosets," a process that I'll describe shortly. The main thing to be aware of is that the sum must not extend beyond the dimension of the Hilbert space in which the ket lives, namely 2^n . That's necessary since an integer x inside a ket $|x\rangle^n$ represents a CBS state, and there are only 2^n of those, $|0\rangle^n, \dots, |2^n - 1\rangle^n$. We'll be sure to obey that rule, too.

Okay, I've burdened you with eye protection, seat belts and other safety equipment, and I know you're bursting to start building something. Let's begin.

23.4 Shor's Quantum Circuit Overview and the Master Plan

23.4.1 The Circuit

The total circuit looks very much like Simon's.



[**Note:** As with Simon, I suppressed the hatching of the quantum wires so as to produce a cleaner looking circuit. The A channel has n lines, and the B channel has r lines, as evinced by the kets and operators which are labeled with the “exponents” n , N and r .]

There are two multi-dimensional registers, the upper A register, and the lower B register. A side-by-side comparison of Shor's and Simon's circuits reveals two differences:

1. The post-oracle's A register is processed by a *quantum Fourier transform* instead of a *multi-order Hadamard gate*.
2. Less significant is the size of the B register. Rather than it being an n -fold tensor space of the same dimension, 2^n , as the A register, it is an r -fold space, with a smaller dimension, 2^r . This reflects the fact that we know f to be periodic with period a , forcing the number of distinct image values of f to be *exactly* a because of *injective* periodicity. Well, $a < M < M^2 \leq 2^n$, so there you have it. These images can be reassigned to fit into a vector space of dimension, smaller, usually much smaller, than A 's 2^n . (Remember that we don't care what the actual images are – sheep, neutrinos – so they may as well be $0, 1, \dots, 2^{r-1}$.) An exact value for r may be somewhat unclear at this point – all we know is that it need never be more than n , and we'd like to reserve the right to give it a different value by using a distinct variable name.

23.4.2 The Plan

Initial State Preparation

We prepare the separable state $|0\rangle^n \otimes |0\rangle^r$ as input.

Data Channel. The top line uses a multi-dimensional *Hadamard* to turn its $|0\rangle^n$ into a maximally mixed (and perfectly balanced) superposition that it passes on to the oracle's top input, thus setting up *quantum parallelism*.

Target Channel. The bottom line forwards its $|0\rangle^r$ directly on to the quantum oracle’s B register, a move that (we saw with Simon) anticipates an application of the *generalized Born rule*.

At that point, we conceptually test the B register output, causing a collapse of both registers (*Born rule*). We’ll analyze what’s left in the collapsed A register’s output, (with the help of a “re-organizing,” *QFT* gate). We’ll find that only a very small and special set of measurement results are likely. And like Simon’s algorithm, we may need more than one sampling of the circuit to get an adequate collection of useful outputs on the A -line, but it’ll come very quickly due to the probabilities.

Strategy

Up to now, I’ve been comparing Shor to Simon. There’s an irony, though, when we come to trying to understand the application of the final post-oracle, pre-measurement gate. It was quite difficult to give a simple reason why a final *Hadamard* gate did the trick for Simon’s algorithm (I only alluded to a technical lemma back then.) But the need for a final *QFT*, as we have already seen, is quite easy to understand: we want the *period*, so we measure in the *Fourier basis* to get the fundamental *frequency*, m . Measuring in the *Fourier basis* means applying a $[z \text{ basis}]\text{-to-}[\text{Fourier basis}]$ transformation, i.e., the *QFT*. m gets us a and we go home early.

One wrinkle rears its head when we look at the *spectrum* of a periodic function, even one that is *pure* in the sense described above. While the likely (or in some cases *only*) measurement possibilities may be limited to a small subset $\{cm\}_{c=0}^{a-1}$ where $m = N/a$ is the *frequency* associated with the *period* a , we don’t know *which* cm we will measure; there are a of them and they are all about equally likely. You’ll see why we should expect to get lucky.

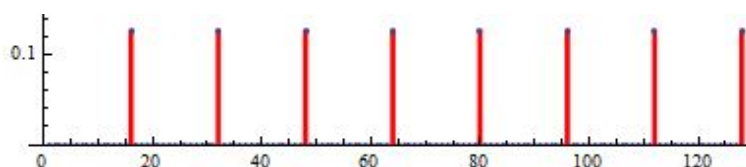


Figure 23.10: Eight highly probable measurement results, cm , for $N = 128$ and $a = 8$

So, while we’ll know we have measured a multiple cm of the frequency, we won’t know *which* multiple. As it happens, if we are lucky enough to get a multiple c that has the bonus feature of being relatively prime (coprime) to a , we be able to use it to find a .

A second wrinkle is that despite what I’ve led you to believe through my pictures, the likely measurements aren’t exact multiples, cm , of the fundamental frequency, m . Instead they will be a values y_c , for $c = 0, 1, \dots, (a - 1)$ which are *very close* to some cm . We’ll have to find out how to lock-on to the nearby cm associated with our measured, y_c . Still, when we do, a c coprime to a will be the the most desirable multiple that will lead to a .

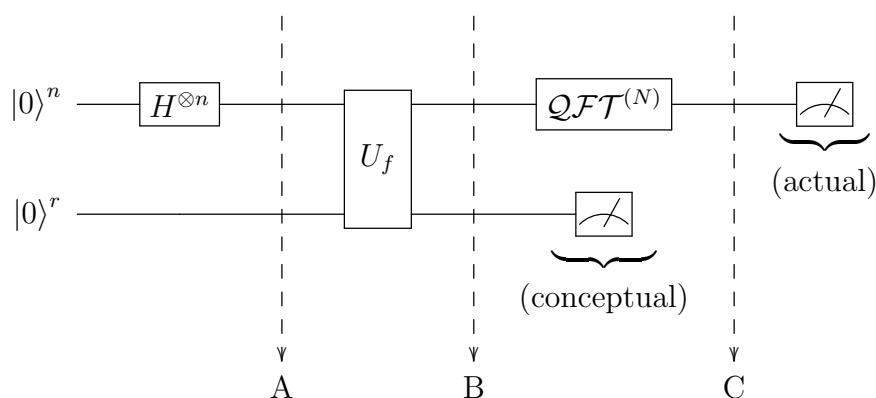
Two Forks

As we proceed, we'll get to a fork in the road. If we take the right fork, we'll find an *easy option*. However the left fork will require much more detailed math. That's the *hard option*. In the *easy case* the spectrum measurement will yield an exact multiple of the fundamental frequency, cm . The harder, *general case*, will give us only a y_c *close to* cm . Then we'll have to earn our money and use some math to hop from the y_c on which we landed to the nearby cm that we really want.

That's the plan. It may not be a perfect plan, but I think that's what I like about it.

23.5 The Circuit Breakdown

We'll segment the circuit into the familiar sub-sections.



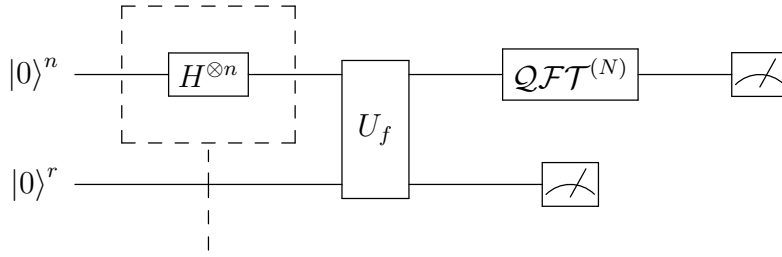
Since many of the sections are identical to what we've done earlier, the analysis is also the same. However, I'll repeat the discussion of those common parts to keep this lecture somewhat self-contained.

23.6 Circuit Analysis Prior to Conceptual Measurement: Point B

23.6.1 The Hadamard Preparation of the A register

I suppose we would be well advised to make certain we know what the state looks like at *access point A* before we tackle point B, and that stage of the circuit is identical to Simon's; it sets up *quantum parallelism* by producing a perfectly mixed entangled

state, enabling the oracle to act on $f(x)$ for all possible x , simultaneously.



Hadamard, $H^{\otimes n}$, in $\mathcal{H}_{(n)}$

Even though we are only going to apply the 2^n -dimensional Hadamard gate to the simple input $|0\rangle^n$, let's review the effect it has on *any* CBS $|x\rangle^n$.

$$|x\rangle^n \longrightarrow H^{\otimes n} \longrightarrow \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} (-1)^{\mathbf{x} \cdot \mathbf{y}} |y\rangle^n ,$$

where the *dot product* between vector \mathbf{x} and vector \mathbf{y} is the mod-2 dot product. When applied to $|0\rangle^n$, reduces to

$$|0\rangle^n \longrightarrow H^{\otimes n} \longrightarrow \left(\frac{1}{\sqrt{2}}\right)^n \sum_{y=0}^{2^n-1} |y\rangle^n ,$$

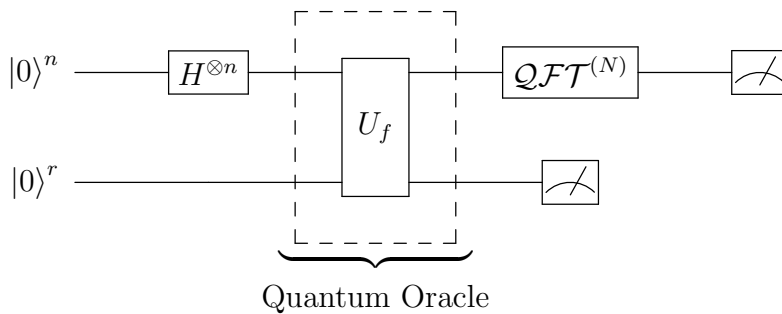
or, returning to the usual computational basis notation $|x\rangle^n$ for the summation,

$$|0\rangle^n \longrightarrow H^{\otimes n} \longrightarrow \left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x\rangle^n .$$

The output state of this Hadamard operator is the n th order x -basis CBS ket, $|+\rangle^n = |0\rangle_{\pm}^n$, reminding us that Hadamard gates provide both quantum parallelism as well as a $z \leftrightarrow x$ basis conversion operator.

23.6.2 The Quantum Oracle

Next, we consider the oracle:



The only difference between this and Simon’s oracle is the width of the oracle’s B register. Today, it is r qubits wide, where r will be (typically) smaller than the n of the A register.

$$\begin{array}{ccc}
 |x\rangle^n & \text{---} & \boxed{U_f} & \text{---} & |x\rangle^n \\
 |0\rangle^r & \text{---} & & \text{---} & |0 \oplus f(x)\rangle^r = |f(x)\rangle^r
 \end{array}$$

$$|x\rangle^n |0\rangle^r \xrightarrow{U_f} |x\rangle^n |f(x)\rangle^r$$

23.6.3 The Quantum Oracle on Hadamard Superposition Inputs

Next, we invoke linearity.

We remarked in Simon’s circuit that the oracle’s B register output is not simply f applied to the superposition state, a nonsensical interpretation since f only has a meaning over its domain, \mathbb{Z}_N . Instead, we apply linearity to the maximally mixed superposition $|0\rangle_\pm^n$ going into the oracle’s top register and find that

$$\left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x\rangle^n |0\rangle^r \xrightarrow{U_f} \left(\frac{1}{\sqrt{2}}\right)^n \sum_{x=0}^{2^n-1} |x\rangle^n |f(x)\rangle^r,$$

which we see is a weighted sum of separable products, all weights being equal to $(1/\sqrt{2})^n$. The headlines are these:

- The output is a superposition of separable terms $|x\rangle^n \frac{|f(x)\rangle^r}{(\sqrt{2})^n}$, the kind of sum the *generalized Born rule* needs,

$$|\varphi\rangle^{n+r} = |0\rangle_A^n |\psi_0\rangle_B^r + |1\rangle_A^n |\psi_1\rangle_B^r + \cdots + |2^n - 1\rangle_A^n |\psi_{2^n-1}\rangle_B^r.$$

An A -measurement of “ x ” would imply a B -state collapse to its (normalized) partner, $|f(x)\rangle^n$.

- If we chose to measure the B register first, a similar collapse to its entangled A partner would result, but in that scenario there would be (approximately) $N/a = m$, not one, pre-images x for every $f(x)$ value. We will study those details shortly.

23.7 Fork-in-the Road: An Instructional Case Followed by the General Case

At this point, we split the discussion into two parallel analyses:

1. an *instructional/easy case* that has some unrealistic constraints on the period, a , and
2. the *general/difficult case*.

The *easy case* will give us the general framework that we'll re-use in the general analysis. Don't skip it. The general case requires that you have your mind already primed with the key steps from the *easy case*.

23.8 Intermezzo – Notation for GCD and Coprime

We'll be using two classical concepts heavily when we justify Shor's quantum algorithm, and we'll also need these result for time complexity estimation.

Basic Notation

We express the fact that one integer, c , divides another integer, a , evenly (i.e. with remainder 0) using the notation

$$c \mid a.$$

Also, we will symbolize the non-negative integers using the notation

$$\mathbb{Z}_{\geq 0}.$$

Now, assume we have two distinct non-negative integers, a and b , i.e.,

$$a, b \in \mathbb{Z}_{\geq 0}, \quad a > b,$$

The following definitions are fundamental in number theory.

23.8.1 Greatest Common Divisor

$$\gcd(a, b) \equiv \text{largest integer, } c, \text{ with } c \mid a \text{ and } c \mid b.$$

23.8.2 Coprime (Relatively Prime)

If $\gcd(a, b) = 1$ we say that a is *coprime* to b (or a and b are *coprime*).

$a \not\mid b$ is my shorthand for a is *coprime* to b .

$\neg a \not\mid b$ is my shorthand for a is *not coprime* to b .

23.9 First Fork: *Easy Case* ($a|N$)

We now consider the special case in which a is a divisor of $N = 2^n$ (in symbols, $a|N$). This implies $a = 2^l$. Immediately, we recognize that there's really no need for

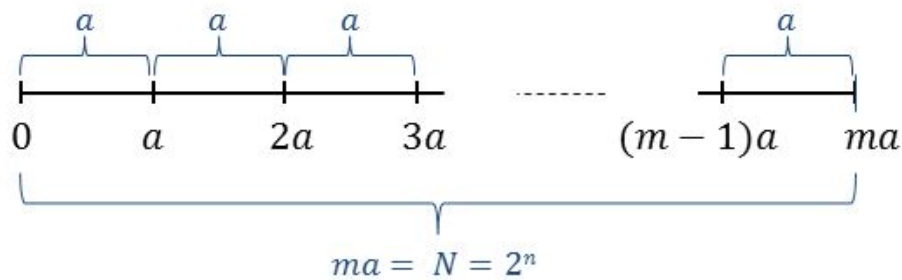


Figure 23.11: *Easy case* covers $a|N$, exactly

a quantum algorithm in this situation because we can test for periodicity using classical means by simply trying 2^l for $l = 1, 2, 3, \dots, (n-1)$, which constitutes $O(\log N)$ trials. In the case of factoring, to which we'll apply period-finding in a later lecture, we'll see that each trial requires a computation of $f(x) = y^x \pmod{N} \in O(\log^4 N)$, y some constant (to be revealed later). So the classical approach is $O(\log N)$ *relative* to the oracle, and $O(\log^5 N)$ *absolute* including the oracle, all without the help of a quantum circuit. However, the quantum algorithm in this *easy case* lays the foundation for the *difficult case* that follows, so we will develop it now and confirm that QC leads to at least $O(\log^5 N)$ classical complexity (we'll do a little better).

23.9.1 Partitioning the Domain into Cosets

It's time to use the fact that f is *periodic injective* with (unknown) period a to help rewrite the output of the Oracle's B register prior to the conceptual measurement. Injective-periodicity tells us that the domain can be partitioned (in more than one way) into many disjoint *cosets* of size a , each of which provides a 1-to-1 sub-domain for f . Furthermore, because we are in the *easy case*, these cosets fit exactly into the big interval $[0, N)$. Here's how.

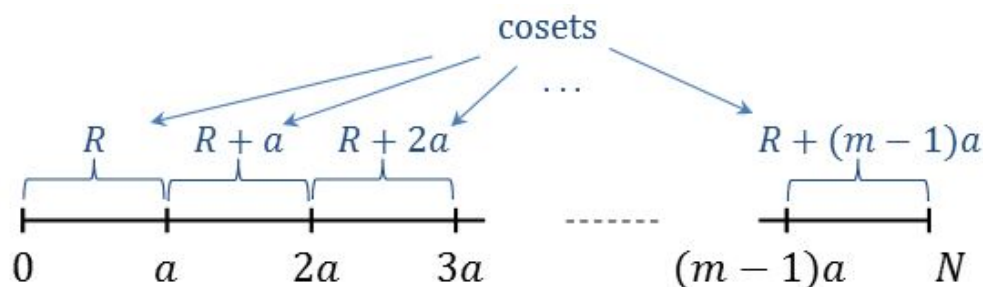


Figure 23.12: $[0, N)$ is the union of distinct cosets of size a

$$\begin{aligned}
[0, N-1] &= [0, ma-1] \\
&= [0, a-1] \cup [a, 2a-1] \cup [2a, 3a-1] \cdots \cup [(m-1)a, ma-1] \\
&= R \cup R+a \cup R+2a \cdots \cup R+(m-1)a,
\end{aligned}$$

where

$$\begin{aligned}
R &\equiv [0, a-1] = \{0, 1, 2, \dots, a-1\}, \\
a &= \text{period of } f, \text{ and} \\
m &= N/a \text{ is the number of times } a \text{ divides } N = 2^n.
\end{aligned}$$

Definition of Coset. $R + ja$ is called the *j*th **coset** of R .

We rewrite this decomposition relative to a typical element, x , in the base coset R .

$$\begin{aligned}
[0, N-1] &= \{0, 1, \dots, x, \dots, a-1\} \cup \{a, 1+a, \dots, x+a, \dots, 2a-1\} \\
&\cup \{2a, 1+2a, \dots, x+2a, \dots, 3a-1\} \cup \cdots \\
&\cdots \cup \{(m-1)a, 1+(m-1)a, \dots, x+(m-1)a, \dots, ma-1\} \\
&= \bigcup_{j=0}^{m-1} \left\{ x + ja \mid x \in [0, a) \right\} \\
&= \bigcup_{j=0}^{m-1} \left\{ x + ja \right\}_{x=0}^{a-1}.
\end{aligned}$$

23.9.2 Rewriting the Output of the Oracle

It seems like a long time since we saw the original expression for oracle's output, so let's write it down again. It was

$$\left(\frac{1}{\sqrt{2}} \right)^n \sum_{x=0}^{2^n-1} |x\rangle^n |f(x)\rangle^r.$$

Our new partition of the domain gives us a nice way to express this. Each element $x \in R$ has a unique partner in each of the cosets, $R + ja$, satisfying

$$\left. \begin{array}{c} x \\ x+a \\ x+2a \\ \vdots \\ x+ja \\ \vdots \\ x+(m-1)a \end{array} \right\}_{x \in R} \xrightarrow{f} f(x).$$

Using this fact (and keeping in mind that $N = 2^n$), we need only sum over the a elements in R and include all the $x + ja$ siblings in each term's A register factor,

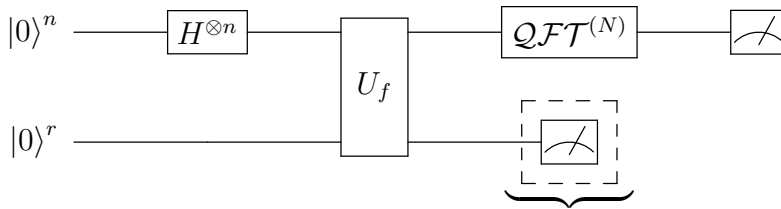
$$\begin{aligned}
& \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle^n |f(x)\rangle^r \\
&= \frac{1}{\sqrt{N}} \sum_{x=0}^{a-1} \left(|x\rangle^n + |x+a\rangle^n + |x+2a\rangle^n + \cdots + |x+(m-1)a\rangle^n \right) |f(x)\rangle^r \\
&= \frac{1}{\sqrt{N}} \sum_{x=0}^{a-1} \left(\sum_{j=0}^{m-1} |x+ja\rangle^n \right) |f(x)\rangle^r \\
&= \sqrt{\frac{m}{N}} \sum_{x=0}^{a-1} \left(\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x+ja\rangle^n \right) |f(x)\rangle^r .
\end{aligned}$$

I moved a factor of $1/\sqrt{m}$ to the right of the outer sum so we could see that

1. each term in that outer sum is a normalized state (there are m CBS terms in the inner sum, and each inner term has an amplitude of $1/\sqrt{m}$) and
2. the common amplitude remaining on the outside produces a normalized state overall (there are a normalized terms in the outer sum, and each term has an amplitude of $\sqrt{m/N} = 1/\sqrt{a}$).

23.9.3 Implication of a Hypothetical Measurement of the B register Output

Although we won't really need to do so, let's imagine what happens if we were to apply the generalized Born rule now using the rearranged sum in which the B channel now plays the role of Born's CBS factors and A channel holds the general factors.



Conceptual

As the last sum demonstrated, each B register measurement of $f(x)$ will be attached to not one, but m , input A register states. Thus, measuring B first, while collapsing A , merely produces a *superposition* of m states in that register, not a single, unique x from the domain. It narrows things down, but not enough to measure,

$$\begin{aligned}
\sqrt{\frac{m}{N}} \sum_{x=0}^{a-1} \left(\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x + ja\rangle^n \right) |f(x)\rangle^r &\longrightarrow \boxed{\text{Measurement}} \\
&\searrow \left(\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x_0 + ja\rangle^n \right) |f(x_0)\rangle^r \\
&\text{(Here, } \searrow \text{ means } \textit{collapses to}.)
\end{aligned}$$

If after measuring the post-oracle B register we were to go on to measure the A register, its resulting collapse would give us one of the m values, $x_0 + ja$, but we would have no way to extract a from that measurement; there is no real information for us here, so we *don't* measure A yet.

Let's name the collapsed – but unmeasured – superposition state in the A register $|\psi_{x_0}\rangle^n$, since it is determined by the measurement “ $f(x_0)$ ” of the collapsed B register,

$$|\psi_{x_0}\rangle^n \equiv \left(\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x_0 + ja\rangle^n \right).$$

Motivation for Next Step

Stand back and you'll see that we've accomplished one of the goals of our introductory “key idea” section. The conceptual measurement of the B register leaves an overall state in the A register in which all of the amplitudes are zero except for m that have equal amplitude $\frac{1}{\sqrt{m}}$. Furthermore, those non-zero terms are spaced at intervals of a in the N -dimensional vector: *this is a “pure” periodic vector with the same period a as our function f .* We have produced a vector whose \mathcal{DFT} is akin to that shown in Figure 23.13. All non-zero amplitudes of the \mathcal{DFT} are multiples of the frequency m , i.e., of the form cm , $c = 0, 1, \dots, (a-1)$. (Due to an artifact of the graphing software, the 0 frequency appears after the array at phantom position $N = 128$.)

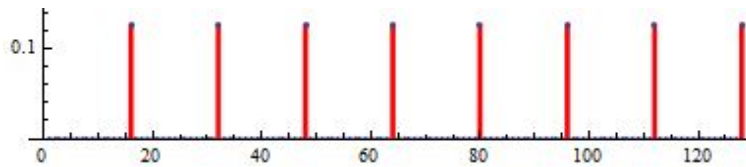


Figure 23.13: The spectrum of a purely periodic vector with period 8 and frequency $16 = 128/8$

The Big Picture

This strongly suggests that we apply the \mathcal{QFT} to the A register in order to produce a state that looks like Figure 23.13.

The Details

We'll get our ideal result if we can produce an A register measurement, cm , where c is coprime to a . The following two thoughts will guide us.

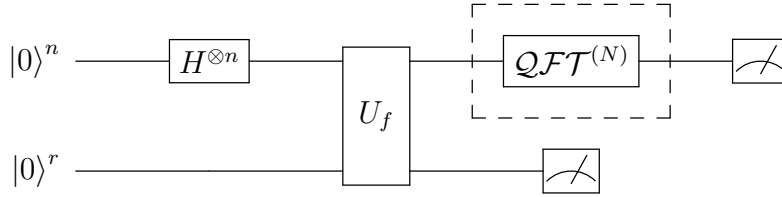
- The *shift property* of the QFT will turn the sum $x + ja$ into a product involving a root-of-unity,

$$QFT^{(N)} |x - x_0\rangle^n = \omega^{x_0 x} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{yx} |y\rangle^n.$$

- Sums of roots-of-unity are famous for canceling themselves out to produce a “whole lotta 0.”

23.9.4 Effect of a Final QFT on the A Register

The much advertised QFT is applied to the conceptually semi-collapsed $|\psi_{x_0}\rangle^n$ at the output of the oracle's A register:



Being a linear operator, it distributes over sums so passes right through the Σ ,

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x_0 + ja\rangle^n \quad \xrightarrow{QFT^{(N)}} \quad \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} QFT^{(N)} |x_0 + ja\rangle^n$$

The QFT of each individual term is

$$\begin{aligned} QFT^{(N)} |x_0 + ja\rangle^n &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{(x_0 + ja)y} |y\rangle^n \\ &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x_0 y} \omega^{jay} |y\rangle^n, \end{aligned}$$

so the QFT of the entire collapsed superposition is

$$\begin{aligned}
QFT^{(N)} \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x_0 + ja\rangle^n \\
&= \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x_0 y} \omega^{j a y} |y\rangle^n \\
&= \frac{1}{\sqrt{mN}} \sum_{y=0}^{N-1} \sum_{j=0}^{m-1} \omega^{x_0 y} \omega^{j a y} |y\rangle^n \\
&= \frac{1}{\sqrt{mN}} \sum_{y=0}^{N-1} \omega^{x_0 y} \sum_{j=0}^{m-1} \omega^{j a y} |y\rangle^n
\end{aligned}$$

[**Factoid.** Rather than invoking QFT 's shift property, we actually re-derived it in-place.]

Summary. This, then, is the superposition state – and our preferred organization of that expression – just prior to sampling the final A register output:

$$\frac{1}{\sqrt{mN}} \sum_{y=0}^{N-1} \omega^{x_0 y} \left(\sum_{j=0}^{m-1} \omega^{j a y} \right) |y\rangle^n$$

We can measure it at any time, and we next look at what the probabilities say we will see when we do.

Foregoing the B Register Measurement. Although we analyzed this under the assumption of a B measurement, an A channel measurement really doesn't care about a “conceptual” B channel measurement. The reasoning is the same as in Simon's algorithm. If we don't measure B first, the oracle's output must continue to carry the full entangled summation

$$\sqrt{\frac{m}{N}} \sum_{x=0}^{a-1} \left(\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x + ja\rangle^n \right) |f(x)\rangle^r .$$

through the final $[QFT^{(N)} \otimes \mathbb{1}^{\otimes r}]$. This would add an extra outer-nested sum

$$\frac{1}{\sqrt{a}} \sum_{x \in [0, a)} \left(\right) |f(x)\rangle^r$$

to our “Summary” expression, above, making it the full oracle output, not just that of the A register. Even leaving B is unmeasured, the algebraic simplification we get below will still take place inside the big parentheses above for each x , and the

probabilities won't be affected. (Also, note that an A register collapse to one specific $|x_0 + ja\rangle^n$ will implicitly select a unique $|f(x_0)\rangle$ in the B register.) With this overview, try carrying this complete sum through the next section if you'd like to see its (non) effect on the outcome.

23.9.5 Computation of Final Measurement Probabilities (*Easy Case*)

We are now in an excellent position to analyze this final A register superposition and see much of it disappear as a result of some of the properties of roots-of-unities that we covered in a past lecture. After that, we can analyze the probabilities which will lead to the algorithm. We proceed in five steps that will

1. identify a special set of a elements, $\mathcal{C} = \{y_c = cm\}_{c=0}^{a-1}$ of certain measurement likelihood,
2. observe that each of $y_c = cm$ will be measured with equal likelihood,
3. prove that a random selection from $[0, a - 1]$ will be coprime-to- a 50% of the time,
4. observe that a $y = cm$ associated with c coprime-to- a will be measured with probability $1/2$, and
5. measure $y = cm$ associated with c coprime-to- a with arbitrarily high confidence in *constant time complexity*.

23.9.6 STEP I: Identify a Special Set of a Elements, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ of Certain Measurement Likelihood

After (conceptual) measurement/collapse of the B register to state $|f(x_0)\rangle^r$, the post- QFT A register was left in the state:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x_0 + ja\rangle^n \quad \text{---} \boxed{QFT^{(N)}} \text{---} \quad \frac{1}{\sqrt{mN}} \sum_{y=0}^{N-1} \omega^{x_0 y} \left(\sum_{j=0}^{m-1} \omega^{jay} \right) |y\rangle^n$$

We look at the inner sum in parentheses in a moment. First, let's recap some facts about ω .

$$\omega \equiv \omega_N$$

was our primitive N th root of unity, so

$$\omega^N = 1.$$

Because m is the number of times a divides (evenly) into N ,

$$m = N/a,$$

we conclude

$$1 = \omega^N = \omega^{am} = (\omega^a)^m.$$

In other words, we have shown that

$$\omega^a = \omega_m$$

is the primitive m th root of unity. Using ω_m in place of ω^a in the above sum produces a form that we have seen before (lecture *Complex Arithmetic for Quantum Computing*, section *Roots of Unity*, exercise (d)),

$$\sum_{j=0}^{m-1} \omega^{jay} = \sum_{j=0}^{m-1} \omega_m^{jy} = \begin{cases} m, & \text{if } y \equiv 0 \pmod{m} \\ 0, & \text{if } y \not\equiv 0 \pmod{m} \end{cases}.$$

This causes a vast quantity of the terms in the QFT output (the double sum) to disappear: only 1-in- m survives,

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x_0 + ja\rangle^n \quad \xrightarrow{QFT^{(N)}} \quad \sqrt{\frac{m}{N}} \sum_{\substack{y \equiv 0 \\ \pmod{m}}} \omega^{x_0 y} |y\rangle^n.$$

Let's think about how to quantify the property that y is 0 mod- m :

$$\begin{aligned} y &\equiv 0 \pmod{m} \\ &\Leftrightarrow \\ y &= cm, \quad \text{for some } c = 0, 1, 2, \dots, a-1. \end{aligned}$$

This defines the special set of size a which will be certain to contain our measured y :

$$\mathcal{C} = \{cm\}_{c=0}^{a-1}.$$

23.9.7 Step II: Observe that Each cm Will be Measured with Equal Likelihood

Since the *easy case* assumes $N = am$, we know that $\sqrt{m/N} = \sqrt{1/a}$, so the last equation becomes

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |x_0 + ja\rangle^n \quad \xrightarrow{QFT^{(N)}} \quad \frac{1}{\sqrt{a}} \sum_{c=0}^{a-1} \omega^{x_0 cm} |cm\rangle^n,$$

and we see that each cm is measured with probability $1/a$.

Consider a function that has *period* $8 = 2^3$ defined on a domain of size $128 = 2^7$. Our problem variables for this function become

$$\begin{aligned} n &= 7, \\ N &= 2^n = 128, \\ a &= 2^3 = 8 \quad \text{and} \\ m &= \frac{N}{a} = \frac{2^7}{2^3} = 16. \end{aligned}$$

$$\begin{aligned} & \frac{1}{\sqrt{128}} \sum_{x=0}^{127} |x\rangle^7 |f(x)\rangle^r \\ &= \frac{1}{\sqrt{128}} \sum_{x=0}^7 \left(|x\rangle^7 + |x+8\rangle^7 + |x+16\rangle^7 + \cdots + |x+120\rangle^7 \right) |f(x)\rangle^r \end{aligned}$$
$$|\psi_3\rangle^7 |f(3)\rangle^r = \frac{1}{\sqrt{16}} \left(|3\rangle^7 + |11\rangle^7 + |19\rangle^7 + \cdots + |123\rangle^7 \right) |f(3)\rangle^r,$$
[illegible]
$$Q\mathcal{FT}^{(128)}|\psi_3\rangle^7 = \sqrt{\frac{m}{N}} \sum_{\substack{y \equiv 0 \\ (\text{mod } m)}} \omega^{x_0 y} |y\rangle^n = \sqrt{\frac{1}{8}} \sum_{\substack{y \equiv 0 \\ (\text{mod } 16)}} \omega^{x_0 y} |y\rangle^7,$$

each corresponding to a y which is a multiple of $m = 16$ having an amplitude-squared (i.e., *probability of collapse*) $= 1/8 = .125$, all other probabilities being zero. Graphing the square amplitudes of the QFT confirms this nicely. (See Figure 23.14.) So we do

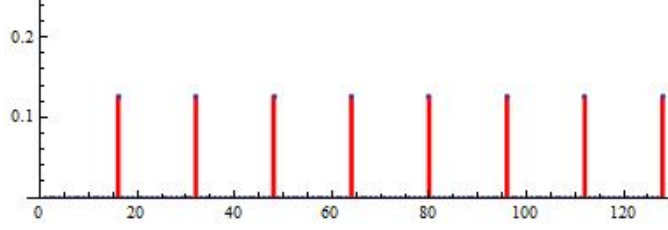


Figure 23.14: Eight probabilities, .125, of measuring a multiple of $m = 16$

find that the only possible measurements *in the frequency domain* lie in the special set

$$\mathcal{C} = \{ 0, 16, 2(16), 3(16), \dots, 7(16) \}.$$

Match this with the graph to verify the spikes are at positions 16, 32, 48, etc. (Due to an artifact of the graphing software, the 0 frequency appears after the array at phantom position 128.) If we can use this set to glean the frequency $m = 16$, we will be able to determine the period $a = N/m = 128/16 = 8$.

Measuring along the frequency basis means applying the basis-transforming QFT after the oracle and explains its presence in the circuit.

23.9.8 Step III: Prove that a Random Selection from $[0, a - 1]$ will be Coprime-to- a 50% of the Time

Specifically, we prove that the probability of a randomly selected $c \in [0, a - 1]$ being coprime to a is $1/2$ (in the *easy case*).

Recall that, in this *easy case*, $N = 2^n$, and $a|N$, so, $a = 2^r$ must also be a power-of-two.

$$\begin{aligned} P(c \not\sim a) &= \frac{\# \text{ coprimes to } a \in [0, a - 1]}{a}, \\ &= \frac{\# \text{ odds} < 2^r}{2^r}, \quad \text{since } a = 2^r \\ &= \frac{1}{2}. \quad \text{QED} \end{aligned}$$

23.9.9 Step IV: Observe that a $y = cm$ Associated with c Coprime-to- a Will be Measured with Probability $1/2$

Our next goal will be to demonstrate that we not only measure *some* value in the set $\mathcal{C} = \{cm\}$ in constant time, but that we can expect to get a *special* subset $\mathcal{B} \subseteq \mathcal{C}$

in constant time, namely those cm corresponding to $c \nmid a$ (i.e., c coprime to a). This will enable us to find m (details shortly), and once we have m we get the period a instantly from $a = N/m$.

In **Step I**, we proved that the likelihood of measuring one of the special $\mathcal{C} = \{y_c\} = \{cm\}$ was 100%. Then, in **Steps II** and **III**, we demonstrated that

- each of the cm will be measured with equal likelihood, and
- the probability of selecting a number that is coprime to a at random from the numbers between 0 and $a - 1$ is (in this special *easy case*) $1/2$, i.e., a constant, independent of a .

We combine all this with the help of a little probability theory. The derivation may seem overly formal given the simplicity of the probabilities in this easy case, but it sets the stage for the difficult case where we will certainly need the formalism.

First we reprise some notation and introduce some new.

$$\begin{aligned}\mathcal{C} &\equiv \{cm\}_{c=0}^{a-1} \\ \mathcal{B} &\equiv \{y_b \mid y_b = bm \in \mathcal{C} \text{ and } b \nmid a\}\end{aligned}$$

(Note: $\mathcal{B} \subseteq \mathcal{C}$.)

$$\begin{aligned}P(\mathcal{C}) &\equiv P(\text{we measure some } y \in \mathcal{C}) \\ P(\mathcal{B}) &\equiv P(\text{we measure some } y \in \mathcal{B}) \\ P(\mathcal{B}|\mathcal{C}) &\equiv P(\text{we measure some } y \in \mathcal{B} \\ &\quad \text{given that} \\ &\quad \text{the measured } y \text{ is known to be } \in \mathcal{C})\end{aligned}$$

We would like a lower bound (specifically, $1/2$) on the probability of measuring a $y_c = cm$ which also has the property that its associated c is coprime to a . In symbols, we would like to show:

Claim. $P(\mathcal{B}) \geq 1/2$, independent of a, M, N , etc.

Proof. We know that we will measure a $y \in \mathcal{C}$ with 100% certainty, so $P(\mathcal{C}) = 1$. We also showed that each cm was equally likely to be measured, so this is equivalent to randomly selecting a $c < a$. Finally, we saw that randomly selecting a $c < a$ would also produce one that was *coprime* to a 50% of the time. So,

$$P(\mathcal{B}) = P(\mathcal{B}|\mathcal{C}) P(\mathcal{C}) = \left(\frac{1}{2}\right) \times 1 = \frac{1}{2}. \quad \text{QED}$$

We proved, in fact, $P(\mathcal{B}) = 1/2$, exactly. Of course this means that

$$P(\neg c \nmid a) = 1 - P(c \nmid a) = \frac{1}{2}.$$

23.9.10 Step V: Observe that y_c Associated with c Coprime to a Will be Measured in Constant Time

Result from the CTC Theorem for Looping Algorithms

Step V follows immediately from a past result.

We have a circuit that we plan on using in an algorithm. Furthermore, our algorithm is going to loop, querying the circuit each pass of the loop, obtaining a statistically independent result each time. It produces a number c , each pass, and for the sake of terminology we'll consider $c \not\equiv a$ to be a *success* (and a c which is *not* coprime to a to be a *failure*). In the last subsection we proved that $P(\text{success}) = 1/2$, independent of the size, N .

These are the exact conditions of the *CTC theorem for looping algorithms* that we covered at the end of our *probability lesson*. That theorem tells us that the algorithm will “succeed,” i.e., yield a $c \not\equiv a$, in constant time, T , and the discussion that followed told us what T would be,

$$T = \left\lfloor \frac{\log(\varepsilon)}{\log(1 - \frac{1}{2})} \right\rfloor + 1,$$

where ε is our desired (small) error tolerance and $\lfloor x \rfloor$ is notation for the *floor* of x , or the greatest integer $\leq x$.

Result from First Principles

We can reproduce the effect of the CTC theorem without invoking it, and for practice with probability theory let's go ahead and do it that way, too.

If we run the circuit T times, obtaining T samples,

$$mc_1, mc_2, \dots, mc_T,$$

the probability that *none* of the associated c were coprime to a is

$$\begin{aligned} P \left(\left(\neg c_1 \not\equiv a \right) \wedge \left(\neg c_2 \not\equiv a \right) \wedge \dots \wedge \left(\neg c_T \not\equiv a \right) \right) \\ = \left(\frac{1}{2} \right)^T, \quad \text{constant time, } T, \text{ independent of } N. \end{aligned}$$

This means that the probability of *at least one* mc having a $c \not\equiv a$ is:

$$P(\text{at least one } c \not\equiv a) = 1 - \left(\frac{1}{2} \right)^T.$$

Conclusion

After an adequate number of measurements (independent of a, M), which produce $y_{c_1}, y_{c_2}, \dots, y_{c_T}$, we can expect at least one of the $y_{c_k} = cm$ to correspond to c ,

with $c \nmid a$ – that is, $y_{c_k} \in \mathcal{B}$ – with high probability. In a moment, we’ll see why it’s so important to achieve this coprime condition.

Example: If we measure the output of our quantum circuit repeatedly, insisting on $y_c \leftrightarrow c \nmid a$ with $P > .999999$, or error tolerance $\varepsilon = 10^{-6}$, we would need

$$T = \left\lceil \frac{\log(10^{-6})}{\log(.5)} \right\rceil + 1 = \left\lceil \frac{-6}{-.30103} \right\rceil + 1 = 19 + 1 = 20$$

measurements.

We can instruct our algorithm to cycle 100 times, to get an even better confidence, but if the data determined that only eight measurements were needed to find the desired cm , then it would return with a successful cm after eight passes. Our hyper-conservative estimate costs us nothing.

23.9.11 Algorithm and Complexity Analysis (*Easy Case*)

A Classical Tool: The Euclidean Algorithm

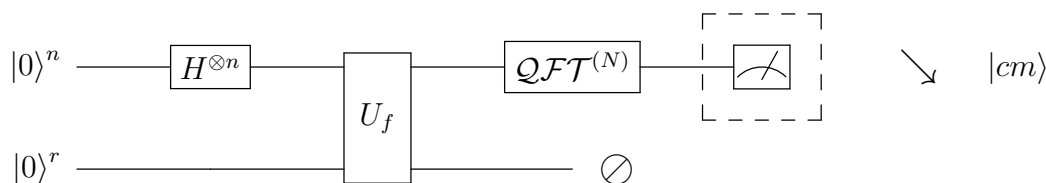
The *Euclidean Algorithm* (EA) is a famous technique from number theory takes two input integers P, Q with $P > Q$ in $\mathbb{Z}_{\geq 0}$ and produces the greatest common divisor of P and Q , also represented by $\gcd(P, Q)$. This is the largest integer that divides both P and Q , evenly (without remainder). $\text{EA}(P, Q)$ will produce $\gcd(P, Q)$ in $O(\log^3 P)$ time. We will be applying EA to N and cm , to get $m' = \gcd(N, cm)$ so the algorithm will produce m' with time complexity $O(\log^3 N) = O(\log^3 M)$.

We’ll assume that $\text{EA}(N, cm)$ delivers as promised and go into its inner workings in a later chapter. EA is a crucial classical step in Shor’s algorithm as we see next.

Why Finding $y_c \in \mathcal{B}$ Solves Shor’s Problem

We proved that we will measure $y_c \in \mathcal{B}$ with near certainty after some predetermined number, T of measurements. Why does such a y_c solve Shor’s period-finding problem (in the *easy case*)?

Here is our circuit, after measuring cm :



After the measurement, this is what we know and don’t know:

Known	Unknown
N	c
cm	m
	a

So the first thing we do is use EA to produce $m' = \gcd(N, cm)$. Now, if $cm \in \mathcal{B}$ then $c \nmid a$, and

$$c \nmid a \Rightarrow m' \equiv \gcd(N, cm) = \gcd(am, cm) = m$$

That gives us the unknown m and from that we can get a :

$$a = \frac{N}{m}.$$

But this only happens if we measure $y \in \mathcal{B}$, which is why finding such a y will give us our period and therefore solve Shor's problem.

How do we know whether we measured an $cm \in \mathcal{B}$? We don't, but we try m' , anyway. We manufacture an a' using m' , by

$$a' = \frac{N}{m'},$$

hoping $a' = a$, which only happens when $m' = m$ (and, after all, finding a , not m , is our goal). We test this by asking whether $f(x + a) = f(x)$ for *any* x and $x = 1$ will do. a is the only number that will produce an equality here by our requirement of *injective* periodicity. If it does, we're done. If not, we try $T - 1$ more times because, with .999999 probability, we'll succeed after $T = 20$ times – no matter how big M (or N) is.

As for the cost of the test $f(x + a) = f(x)$, that depends on f . We are only asserting *relativized* polynomial complexity for period-finding but know (by a future chapter) that for *factoring*, f will be polynomial fast ($O(\log^4(M))$), and maybe better).

We can now present the algorithm.

Shor-like Algorithm (*Easy Case*)

- Select an integer, T , that reflects an acceptable failure rate based on any known aspects of the period. E.g., for a failure tolerance of .000001, We might choose $T = 20$
- Repeat the following loop at most T times.
 1. Apply Shor's circuit.
 2. Measure output of \mathcal{QFT} and get cm .
 3. Compute $m' = \text{EA}(N, cm)$, and set $a' \equiv N/m'$.
 4. Test a' : If $f(1 + a') = f(1)$ then $a' = a$, (success) break from loop.
 5. Otherwise continue to the next pass of the loop.
- If the above loop ended naturally (i.e., not from the *break*) after T full passes, we failed. Otherwise, we have found a .

Computational Complexity (*Easy Case*)

We have already seen that $O(\log(N)) = O(\log(M))$ as well as any powers of the logs, so I will use M in the following.

- The Hadamard gates are $O(\log(M))$.
- The QFT is $O(\log^2(M))$.
- Complexity of U_f is same as that of f , which could be anything. We don't count that here, but we will show in a separate lecture that for integer factoring, it is certainly at least $O(\log^4 M)$.
- The outer loop is $O(T) = O(1)$, since T is constant.
- The classical EA sub-algorithm is $O(\log^3 M)$.
- The four non-oracle components, above, are done in series, not in nested loops, so the overall *relativized* complexity will be the worst case among them, $O(\log^3(M))$.
- In the case of factoring needed for RSA encryption breaking (order-finding) the actual oracle is $O(\log^4(M))$ or better, so the absolute complexity in that case will be $O(\log^4(M))$

So the entire Shor circuit for an $f \in O(\log^4(M))$ (true for RSA/order-finding) would have an absolute complexity of $O(\log^4(M))$. Notice that, while not an exponential speed-up over the $O(\log^5(M))$ *easy case* classical algorithm presented, it *is* “faster” by a factor of $\log N$. That's due to the fact that the quantum circuit requires a constant time sampling, while the classical function must be sampled $O(\log N)$ times. To be fair, though, the classical algorithm was deterministic, and the quantum algorithm, probabilistic. You can debate whether or not this counts in your forums.

This completes the *easy case* fork-in-the-road. It contains all the way points that we'll need for the general case without the subtle and tricky math. Checking our fear of the upcoming technicalities at the door, we forge onward.

23.10 Second Fork: General Case (We do not Assume $a|N$)

Now, a need not be a power-of-2, so the classical approach no longer admits an $O(\log^5 N)$ algorithm. Also, if we take the integer quotient, $m = \lfloor N/a \rfloor$, there will usually be a remainder representing those “excess” integers that don't fit into a final, full period interval.

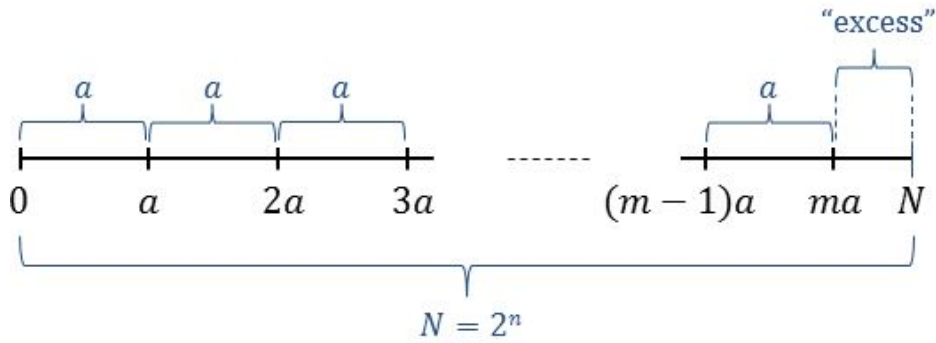


Figure 23.15: There is (possibly) a remainder for N/a , called the “excess”

23.10.1 Partitioning the Domain into Cosets

Like the *easy case*, f 's *injective periodicity* helps us rewrite the output of the oracle's B register prior to the conceptual measurement. The domain can still be partitioned into many disjoint *cosets* of size a , each of which provides a 1-to-1 sub-domain for f , but now there is a final coset which may not be complete:

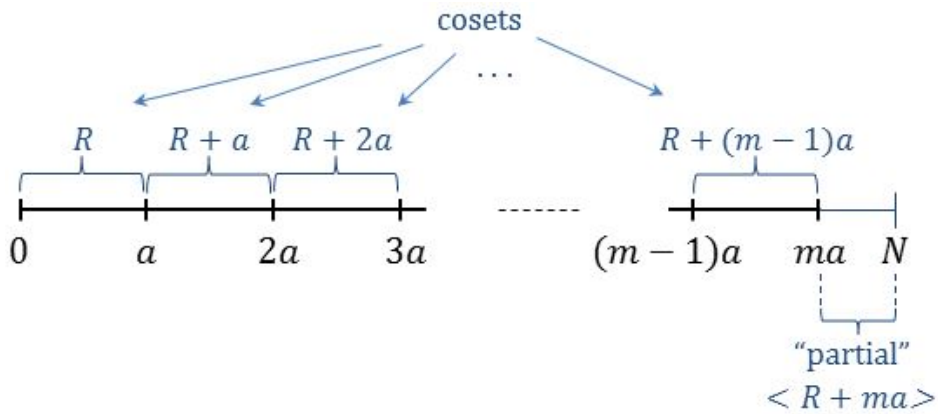


Figure 23.16: $[0, N)$ is the union of distinct cosets of size a , except for last

Express $[0, N - 1]$ as the union of a union,

$$[0, N - 1]$$

$$= [0, a - 1] \cup [a, 2a - 1] \cdots \cup [(m - 1)a, ma - 1] \cup \overbrace{[ma, N - 1]}^{\text{excess}}$$

$$= R \cup R + a \cdots \cup R + (m - 1)a, \cup \overbrace{\langle R + ma \rangle}^{\text{partial}}$$

where

$$R \equiv [0, a - 1] = \{0, 1, 2, \dots, a - 1\},$$

$$a = \text{period of } f, \text{ and}$$

$$m = \lfloor N/a \rfloor, \text{ the number of times } a \text{ divides (unevenly) into } N.$$

Cosets. As before, $R + ja$ is called the j th *coset* of R , but now we have

$$\langle R + ma \rangle \subseteq [R + ma],$$

the *partial* m th coset of R from ma to $N - 1$.

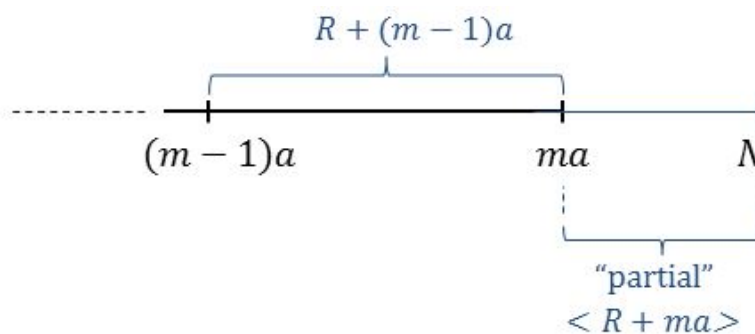


Figure 23.17: The final coset may have size $< a$

We express the decomposition of our entire domain relative to a typical element x in the base coset R ,

$$[0, N - 1] = \bigcup_{j=0}^{m-1} \left\{ x + ja \right\}_{x=0}^{a-1} \cup \left\{ x + ma \right\}_{x=0}^{N-ma-1},$$

but now we had to “slap on” the partial coset, $\{x + ma\}$, to account for the possible overflow.

Notation to Deal with the “Partial Coset”

We have to be careful about counting the family members of each element $x \in R$, i.e., those $x + ja$ who map to the same $f(x)$ by periodicity. We sometimes have a member in the last, partial, coset, and sometimes not. If x is among the first few integers of R , i.e., $\in [0, N - ma)$, then there will be $m + 1$ partners (including x) akin to it.

However, if x is among the latter integers of R , i.e., $\in [N - ma, a)$, then there will be only m partners (including x) in its kin.

We’ll use \tilde{m} to be either m or $m + 1$ depending on x ,

$$\tilde{m} = \begin{cases} m + 1, & \text{for the “first few” } x \text{ in } [0, a - 1] \\ m, & \text{for the “remaining” } x \text{ in } [0, a - 1] \end{cases}$$

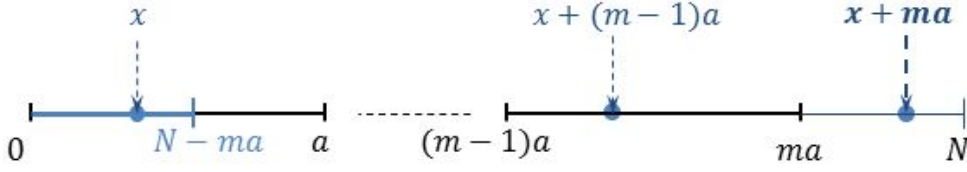


Figure 23.18: If $0 \leq x < N - ma$, a full $m + 1$ numbers in \mathbb{Z}_N map to $f(x)$

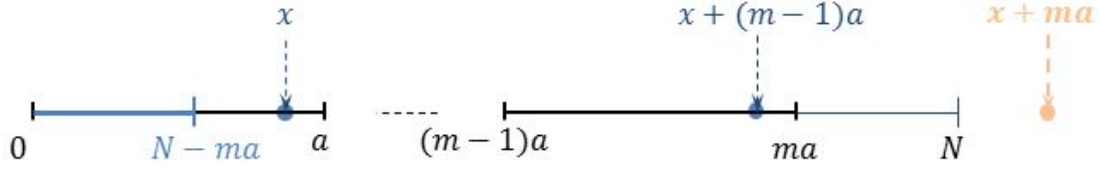


Figure 23.19: If $N - ma \leq x < a$, only m numbers in \mathbb{Z}_N map to $f(x)$

23.10.2 Rewriting the Output of the Oracle's output

The original expression for oracle's B register output was

$$\left(\frac{1}{\sqrt{2}} \right)^n \sum_{x=0}^{2^n-1} |x\rangle^n |f(x)\rangle^r ,$$

and our new partition of the domain gives us a nice way to rewrite this. First, note that

$$\left(\begin{array}{c} x \\ x+a \\ x+2a \\ \vdots \\ x+ja \\ \vdots \\ x+(m-1)a \\ \text{(and sometimes } \dots x+ma) \end{array} \right)_{x \in R} \xrightarrow{f} f(x) .$$

Now we make use of our flexible notation, \tilde{m} , to keep the expressions neat without sacrificing precision of logic:

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle^n |f(x)\rangle^r = \left(\approx \sqrt{\frac{\tilde{m}}{N}} \right) \sum_{x=0}^{a-1} \left(\frac{1}{\sqrt{\tilde{m}}} \sum_{j=0}^{\tilde{m}-1} |x+ja\rangle^n \right) |f(x)\rangle^r ,$$

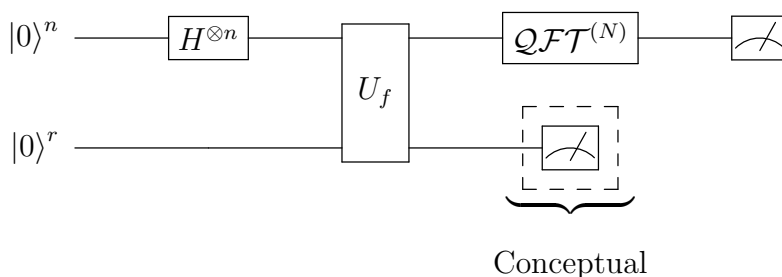
where (to repeat for emphasis),

$$\tilde{m} = \tilde{m}(x) = \begin{cases} m+1, & x \in [0, N-ma) \\ m, & x \in [N-ma, a) \end{cases} .$$

The factor of $1/\sqrt{\tilde{m}}$ inside the sum normalizes each term in the outer sum. However, the common amplitude remaining on the outside is harder to symbolize in a formula, which is why I used “ \approx ” to describe it. (\tilde{m} doesn’t even make good sense outside the sum, but it gives us an idea of what the normalization factor is.) It turns out that we don’t care about its exact value. It will be some number between $\sqrt{\frac{m}{N}}$ and $\sqrt{\frac{m+1}{N}}$, the precise value being whatever is needed to normalize the overall state.

23.10.3 Implication of a Hypothetical Measurement of the B register Output

We’ve seen that it helps to imagine a measurement of the oracle’s B register’s output.



Each B register measurement of $f(x)$ will be attached to not one, but \tilde{m} , input A register states. The generalized Born rule tells us that measuring B will cause the collapse of A into a superposition of \tilde{m} CBS states, narrowing things down considerably.

$$\begin{aligned} \sqrt{\frac{\tilde{m}}{N}} \sum_{x=0}^{a-1} \left(\frac{1}{\sqrt{\tilde{m}}} \sum_{j=0}^{\tilde{m}-1} |x + ja\rangle^n \right) |f(x)\rangle^r &\longrightarrow \text{Measurement} \\ \searrow & \left(\frac{1}{\sqrt{\tilde{m}}} \sum_{j=0}^{\tilde{m}-1} |x_0 + ja\rangle^n \right) |f(x_0)\rangle^r \\ \text{(Here, } \searrow \text{ means } \textit{collapses to}.) \end{aligned}$$

If after measuring the post-oracle B register we were to go on to measure the A register, it would collapse, giving us a reading of one of the \tilde{m} values, $x_0 + ja$, but that value would not get us any closer to knowing a . Therefore, just like the *easy case*, we *don’t* measure A yet. Instead, we name the collapsed – but unmeasured – superposition state in the A register $|\psi_{x_0}\rangle^n$, since it is determined by the measurement “ $f(x_0)$ ” of the collapsed B register,

$$|\psi_{x_0}\rangle^n \equiv \left(\frac{1}{\sqrt{\tilde{m}}} \sum_{j=0}^{\tilde{m}-1} |x_0 + ja\rangle^n \right)$$

Foregoing the B Register Measurement. By now we have twice seen why we can avoid this conceptual measurement yet still analyze the A register as if it had

occurred. There is no harm in pretending we measured and collapsed to a $|f(x_0)\rangle^r$ first. (See the **Easy Case** or our lesson *Simon's Algorithm* for longer explanations of the same argument.)

Motivation for Next Step

The conceptual measurement of the B register leaves an overall state in the A register in which all the amplitudes are zero except for \tilde{m} of them which have amplitude $\frac{1}{\sqrt{m}}$. Furthermore, those non-zero terms are spaced at intervals of a in the N -dimensional vector: *this is a “pure” periodic vector with the same period a as our function f .*

In contrast to the *easy case*, however, \tilde{m} is sometimes m and sometimes $m + 1$, never mind that a is not a perfect power-of-2 or that it doesn't divide into N evenly. All this imperfection destroys the arguments made in the easy case.

A look at the \mathcal{DFT} of the vector whose components match the amplitudes of a typical collapsed state left in the A register (see Figure 23.20) confirms that a frequency domain measurement of $\mathcal{QFT}^{(N)} |\psi_{x_0}\rangle^n$ no longer assures us of seeing one of the a numbers cm' , $c = 0, \dots, a-1$, with m' the true – and typically *non-integer* – frequency N/a . As for our integer quotient $m \equiv \lfloor N/a \rfloor$ close to the actual frequency N/a , it's possible that *none* of the frequency domain points cm have high amplitudes.

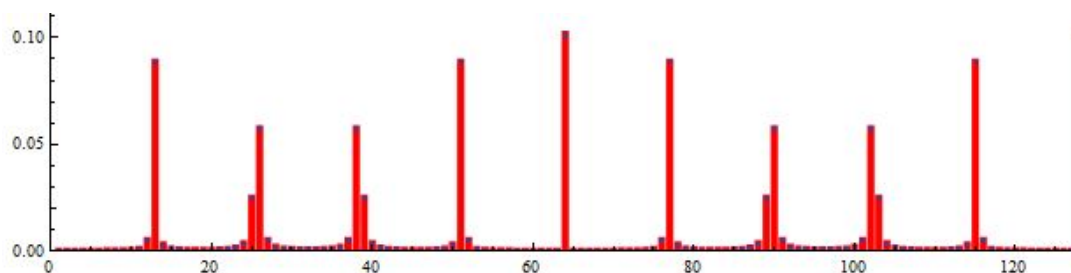


Figure 23.20: The spectrum of a purely periodic vector with period 10 and frequency $12.8 = 128/10$

The situation appears grim.

Let's look at the bright side, though. This picture of a typical \mathcal{DFT} applied to an N dimensional vector, 0 except for amplitudes $\frac{1}{\sqrt{m}}$ at \tilde{m} time domain points, suggests that there *are* still only a frequencies, y_0, y_1, \dots, y_{a-1} which have large magnitudes. And there's even more reason for optimism. Those likely y_k appear to be at least *close* to multiples of the “integer-ized frequency” m , i.e., they are *near* frequency domain points of the form cm . (Due to an artifact of the graphing software, the 0 frequency appears after the array at phantom position $N = 128$.)

The Big Picture

It still seems to be a good idea to apply a QFT to the A register in order to produce a state that looks like Figure 23.20.

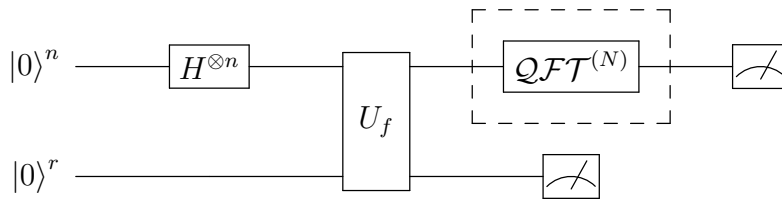
The Key Steps

We'll show that there are only a likely A register measurements, y_c , $c = 0, 1, \dots, (a-1)$. And even if a given $y_c \neq cm$ exactly, it will at least *lead* us to a cm . In fact, we can expect the resulting c to be relatively prime to a with good probability, a very desirable outcome. Here are the general steps.

- We will identify a special set of a elements, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ of high measurement likelihood.
- We will show that each y_c is very close to – in fact, uniquely selects – a point of the form cm . We'll describe a fast (polynomial time) algorithm that takes any of the likely measured y_c to its unique partner cm .
- We'll prove that we can expect to get a y_c (and thus cm) with c coprime to the period a in constant time. Such a c will unlock the near-frequency, m , and therefore the period, a .

23.10.4 Effect of a Final QFT on the A Register

The QFT is applied to the conceptually semi-collapsed $|\psi_{x_0}\rangle^n$ at the output of the oracle's A register:



The linear QFT passes through the Σ ,

$$\frac{1}{\sqrt{\tilde{m}}} \sum_{j=0}^{\tilde{m}-1} |x_0 + ja\rangle^n \quad \text{---} \quad \boxed{QFT^{(N)}} \quad \text{---} \quad \frac{1}{\sqrt{\tilde{m}}} \sum_{j=0}^{\tilde{m}-1} QFT^{(N)} |x_0 + ja\rangle^n$$

Note that it doesn't matter that \tilde{m} is sometimes m and sometimes $m+1$. Once it collapses, it becomes one of the two depending on which x_0 is selected by the measurement, and the formulas all still hold true. The QFT of each individual term

is

$$\begin{aligned} \mathcal{QFT}^{(N)} |x_0 + ja\rangle^n &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{(x_0 + ja)y} |y\rangle^n \\ &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x_0 y} \omega^{jay} |y\rangle^n, \end{aligned}$$

so the \mathcal{QFT} of the entire collapsed superposition is

$$\begin{aligned} \mathcal{QFT}^{(N)} \frac{1}{\sqrt{\tilde{m}}} \sum_{j=0}^{\tilde{m}-1} |x_0 + ja\rangle^n &= \frac{1}{\sqrt{\tilde{m}}} \sum_{j=0}^{\tilde{m}-1} \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{x_0 y} \omega^{jay} |y\rangle^n \\ &= \frac{1}{\sqrt{\tilde{m}N}} \sum_{y=0}^{N-1} \sum_{j=0}^{\tilde{m}-1} \omega^{x_0 y} \omega^{jay} |y\rangle^n \\ &= \frac{1}{\sqrt{\tilde{m}N}} \sum_{y=0}^{N-1} \omega^{x_0 y} \sum_{j=0}^{\tilde{m}-1} \omega^{jay} |y\rangle^n \end{aligned}$$

In this expression, the normalizing factor $\frac{1}{\sqrt{\tilde{m}N}}$ is *precise*. That's in contrast to the pre-collapsed state in which we had an *approximate* factor outside the full sum. The B register measurement “picked out” one specific x_0 , which had a definite \tilde{m} associated with it. Whether it was m or $m + 1$ doesn't matter. It is one of the two, and that value is used throughout this expression.

Summary. This is the preferred organization of our superposition state prior to sampling the final A register output:

$$\frac{1}{\sqrt{\tilde{m}N}} \sum_{y=0}^{N-1} \omega^{x_0 y} \left(\sum_{j=0}^{\tilde{m}-1} \omega^{jay} \right) |y\rangle^n.$$

The next several sections explore what the probabilities say we will see when we measure this state. And while we analyzed it under the assumption of a prior B measurement, the upper (A) channel measurement won't care about that conceptual measurement, as we'll see. We continue the analysis as if we had measured the B channel.

23.10.5 Computation of Final Measurement Probabilities (General Case)

This general case, which I scared you into thinking would be a mathematical horror story, has been a relative cakewalk so far. About all we had to do was replace the firm m with the slippery \tilde{m} , and everything went through without incident. That's about to change.

In the *easy case*, we were able to make the majority terms in our sum vanish (all but 1-in- m). Let's review how we did that. We noted that $\omega \equiv \omega_N$, the primitive N th root, so

$$\omega^N = 1.$$

Then we replaced N with ma , to get

$$\omega^{ma} = 1$$

and realized that this implied that ω^a was a primitive m th root of unity. From there we were able to get massive cancellation due to the facts we developed about sums of roots-of-unity.

The problem, now, is that we *cannot* replace N with ma . We have an \tilde{m} , but even resolving that to m or $m+1$ won't work, because neither one divides N evenly (by the general case hypothesis). So we'll never be able to manufacture an m th root of unity at this point and cannot watch those big sums dissolve before our eyes. So sad.

We can still get what we need, though, and have fun with math, so let's rise to the challenge.

As with the *easy case* our job is to analyze the final (post *QFT*) A register superposition. While none of the terms will politely disappear the way they did in the *easy case*, we will find that certain y states will be much more likely than others, and this will be our savior.

Computing the final measurement probabilities will require the following five steps.

1. identify (*without proof*) a special set of a elements, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ of high measurement likelihood,
2. *prove* that the values in, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ have high measurement likelihood,
3. associate $\{y_c\}_{c=0}^{a-1}$ with $\{c/a\}_{c=0}^{a-1}$,
4. describe an $O(\log^3 N)$ algorithm that will produce c/a from y_c , and
5. observe that y_c associated with c coprime to a will be measured in constant time.

23.10.6 STEP I: Identify (*Without Proof*) a Special Set of a Elements, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ of High Measurement Likelihood

In this step, we will merely describe the subset of y that we want to measure. In the next step, we'll provide the proof.

In the *easy case* we measured y , which had the special form

$$y = cm, \quad c = 0, 1, 2, \dots, a-1,$$

with 100% certainly in *single* measurement. From there we tested whether the c was coprime to a , (which it was with high probability), and so on. This time we can't be 100% sure of *anything*, even after post-processing with the *QFT*, but that's normal for quantum algorithms – we often have to “work the numbers” and be satisfied to get what we want in constant or polynomial time. I claim that in the general case we will measure an equally small subset of y , again a in all, that we label

$$y = y_c, \quad c = 0, 1, 2, \dots, a-1$$

with probability $P = 1 - \varepsilon$ in $O(1)$ measurements. (Whenever you see me use a ε , it represents a positive number which is as small as we want, say .000000001, for example.) The following construction will lead us to these special a values.

Consider the (very) long line $[0, aN - 1]$.



Figure 23.21: A very long line consisting of a copies of $N = 2^n$

Around the a integral points on this line,

$$0, N, 2N, \dots, cN, \dots, (a-1)N,$$

place (relatively small) half-open intervals of width a .

$$\begin{aligned} & \left[-\frac{a}{2}, +\frac{a}{2}\right), \quad \left[N - \frac{a}{2}, N + \frac{a}{2}\right), \\ & \dots \quad \left[cN - \frac{a}{2}, cN + \frac{a}{2}\right) \quad \dots \\ & \dots \quad \left[(a-1)N - \frac{a}{2}, (a-1)N + \frac{a}{2}\right) \end{aligned}$$

Each interval contains *exactly one* integral multiple, ya , of a in it. We'll label the multiplier that gets us into the c th interval y_c . (y_0 is easily seen to be 0.)

$$\begin{aligned} 0a \in \left[-\frac{a}{2}, +\frac{a}{2}\right), \quad \dots, \quad y_ca \in \left[cN - \frac{a}{2}, cN + \frac{a}{2}\right), \\ \dots, \quad y_{a-1}a \in \left[(a-1)N - \frac{a}{2}, (a-1)N + \frac{a}{2}\right) \end{aligned}$$

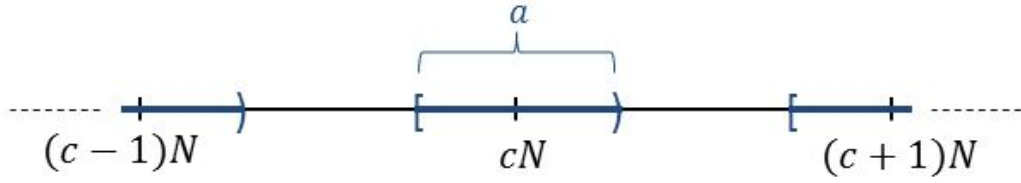


Figure 23.22: Half-open intervals of width a around each point cN

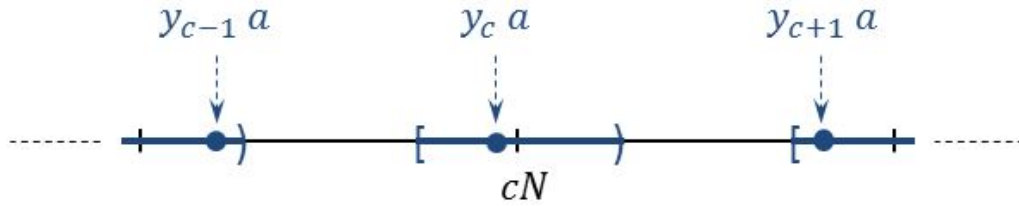


Figure 23.23: Exactly one integral multiple of a falls in each interval

Since $y_c a < aN$, for all $c = 0, 1, \dots, (a-1)$, we are assured $y_c < N$, so it is a possible result of our measurement after the QFT .

Claim. The $\{y_c\}$ just described are the a values most likely to be measured. We'll see that with probability $P = 1 - \varepsilon$, arbitrarily close to 1, we will measure one of these y_c in constant time, T , independent of N . (The proof appears in the next step. In this step we only want to establish our vocabulary and goals.)

Example 1

In this first of two examples, we illustrate the selection of y_c for an actual periodic function defined on integers $[0, N-1]$, where $N = 2^5 = 32$, and the period $a = 3$. (Remember, it doesn't matter what the values $f(k)$ are, so long as we're told it is periodic.)

- $c = 0$: The center of the interval is $0N = 0 \cdot 32 = 0$. We seek y_0 such that

$$\begin{aligned} 3y_0 &\in [-1.5, 1.5) \\ \Rightarrow y_0 &= 0, \quad \text{since} \\ 3 \cdot 0 &= 0 \in [-1.5, 1.5). \end{aligned}$$

- $c = 1$: The center of the interval is $1N = 1 \cdot 32 = 32$. We seek y_1 such that

$$\begin{aligned} 3y_1 &\in [30.5, 33.5) \\ \Rightarrow y_1 &= 11, \quad \text{since} \\ 3 \cdot 11 &= 33 \in [30.5, 33.5). \end{aligned}$$

- $c = 2$: The center of the interval is $2N = 2 \cdot 32 = 64$. We seek y_2 such that

$$\begin{aligned} 3y_2 &\in [62.5, 65.5) \\ \Rightarrow y_2 &= 21, \quad \text{since} \\ 3 \cdot 21 &= 63 \in [62.5, 65.5). \end{aligned}$$

- Etc.

Example 2

In this second example, we test the claim that the y_c so described really do have high relative measurement likelihood for an actual periodic function.

Consider a function that has *period* 10 defined on a domain of size $128 = 2^7$. Our problem variables for this function become

$$\begin{aligned} n &= 7, \\ N &= 2^n = 128, \\ a &= 10 \quad \text{and} \\ m &= \left\lfloor \frac{N}{a} \right\rfloor = \left\lfloor \frac{128}{10} \right\rfloor = 12. \end{aligned}$$

Let's say that we measured the B register and got the value $f(x_0)$, corresponding to $x_0 = 3$. For this x ,

$$\tilde{m} = m + 1 = 13,$$

since

$$x_0 + ma = 3 + 12 \cdot 10 = 123 < 128.$$

According to the above analysis, the full pretested superposition,

$$\begin{aligned} &\frac{1}{\sqrt{128}} \sum_{x=0}^{127} |x\rangle^7 |f(x)\rangle^r \\ &= \frac{1}{\sqrt{128}} \sum_{x=0}^7 \left(|x\rangle^7 + |x+10\rangle^7 + |x+20\rangle^7 + \cdots + \overbrace{|x+120\rangle^7}^{\text{sometimes}} \right) |f(x)\rangle^r \end{aligned}$$

when subjected to a B register measurement will collapse to

$$|\psi_3\rangle^7 |f(3)\rangle^r = \frac{1}{\sqrt{13}} \left(|3\rangle^7 + |13\rangle^7 + |23\rangle^7 + \cdots + |123\rangle^7 \right) |f(3)\rangle^r,$$

leaving the following vector in the A register (which I have “stacked” so as to align the 13 non-zero coordinates):

```

( 0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0, 0,
  0, 0, 0, .27735, 0, 0, 0, 0, 0 ) .

```

We are interested in learning f 's *period* and, like the *easy case*, the way to get at it is by looking at this state vector's *spectrum*, so we take the QFT . Now, unlike the *easy case*, this vector's QFT will *not* create lots of 0 amplitudes; generally all $N = 128$ of them will be non-zero. That's because the resulting sum

$$QFT^{(128)} |\psi_3\rangle^7 = \frac{1}{\sqrt{\tilde{m}N}} \sum_{y=0}^{N-1} \omega^{x_0 y} \left(\sum_{j=0}^{\tilde{m}-1} \omega^{jay} \right) |y\rangle^n.$$

did not admit any cancellations or simplification. Instead, the above claim – which we will prove in the next section – is that for $x_0 = 3$ only $\tilde{m} = 13$ of them will be *likely*, the special $\{y_c\}$, for $c = 0, 1, 2, \dots, 12$ we described in our last analysis. Let's put our money where our mouth is and at least show this to be true for the one function under consideration.

We take three y_c as examples: y_4 , y_5 and y_6 . We'll do this in two stages. First we identify the three y_c values. Next, we graph the probabilities of $QFT^{(128)} |\psi_3\rangle^7$ around those three values to see how they compare with nearby y values.

Stage 1. Compute y_4 , y_5 and y_6

- $c = 4$: The center of the interval is $4N = 4 \cdot 128 = 512$. We seek y_4 such that

$$\begin{aligned} 10y_4 &\in [507, 517) \\ \Rightarrow y_4 &= 51, \quad \text{since} \\ 10 \cdot 51 &= 510 \in [507, 517). \end{aligned}$$

- $c = 5$: The center of the interval is $5N = 5 \cdot 128 = 640$. We seek y_5 such that

$$\begin{aligned} 10y_5 &\in [635, 645) \\ \Rightarrow y_5 &= 64, \quad \text{since} \\ 10 \cdot 64 &= 640 \in [635, 645). \end{aligned}$$

- $c = 6$: The center of the interval is $6N = 6 \cdot 128 = 768$. We seek y_6 such that

$$\begin{aligned} 10y_6 &\in [763, 773) \\ \Rightarrow y_6 &= 77, \quad \text{since} \\ 10 \cdot 77 &= 770 \in [763, 773). \end{aligned}$$

Stage 2. Look at the Graph of $\left\| Q\mathcal{FT}^{(128)} |\psi_3\rangle^7 \right\|^2$ Around These Three y

Here is a portion of the graph of the $Q\mathcal{FT}$'s absolute value squared showing the probabilities of measuring over 35 different y value in the frequency domain. It exhibits in a dramatic way how much more likely the y_c are to be detected than are the non- y_c values. (See Figure 23.24.) Even though the non- y_c have non-zero

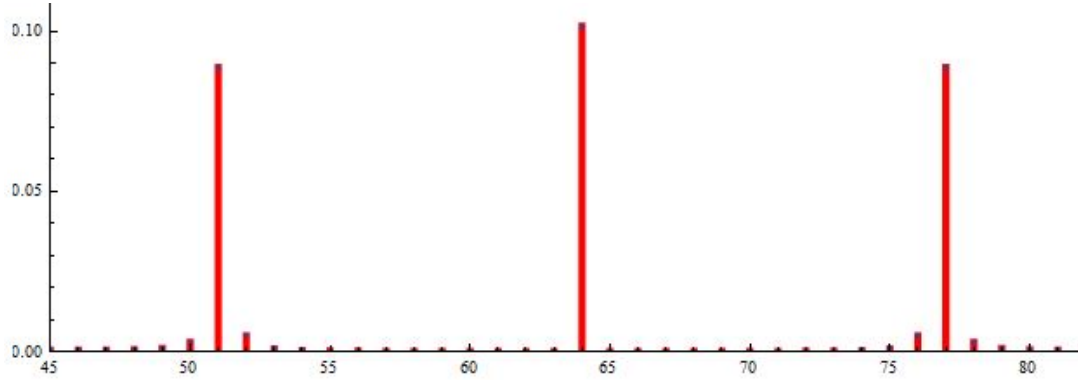


Figure 23.24: Probabilities of measuring $y_4 = 51$, $y_5 = 64$ or $y_6 = 77$ are dominant.

probabilities of measurement, they barely register except for a few.

If we apply the math to all 10 y_c for this x_0 (which, remember is 3), we find that the measurements *in the frequency domain* will usually yield values that lie in the special set

$$\mathcal{C} = \{ 0, 13, 26, 38, 51, 64, 77, 90, 102, 115 \}$$

[**Exercise.** Show this for the y_c not computed in our example.]

Measuring along the frequency basis means applying the basis-transforming $Q\mathcal{FT}$ after the oracle and explains its presence in the circuit.

Introducing the \hat{y}_c

To assist our analysis we'll define a set of \hat{y}_c associated with the desired y_c such that the \hat{y}_c all live in the interval $\left[-\frac{a}{2}, +\frac{a}{2}\right)$ (Figure 23.25).

We do this by subtracting cN from each $y_c a$ to “bring it” into the base interval $\left[-\frac{a}{2}, +\frac{a}{2}\right)$. More precisely,

$$\hat{y}_c \equiv y_c a - cN \in \left[-\frac{a}{2}, +\frac{a}{2}\right), \quad c = 0, 1, 2, \dots, a-1,$$

which can be stated using modular arithmetic by

$$\hat{y}_c = y_c a \pmod{N}.$$

While we will be looking to measure the y_c , it will be important to remember that they correspond ($\equiv \pmod{N}$) to these $\hat{y}_c \in \left[-\frac{a}{2}, +\frac{a}{2}\right)$.

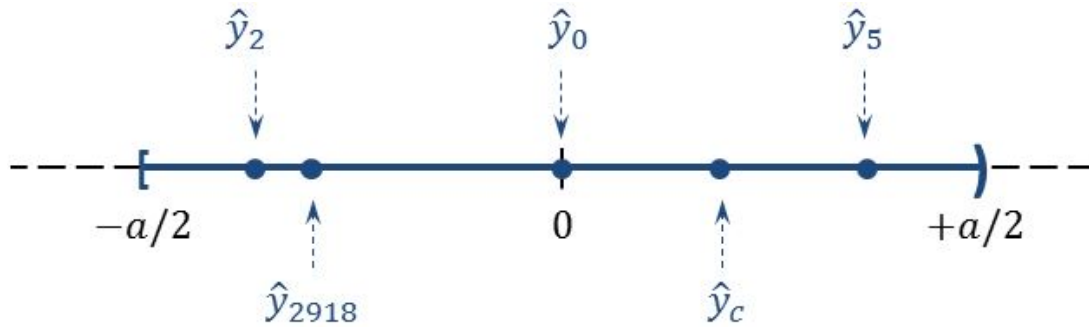


Figure 23.25: \hat{y}_c all fall in the interval $[-a/2, a/2]$

This is the vocabulary we will need. Next, we'll take a lengthy – but leisurely – mathematical cruise to prove our claim of high probability measurement for the set \mathcal{C} .

23.10.7 STEP II: *Prove* that the Values in, $\mathcal{C} = \{y_c\}_{c=0}^{a-1}$ Have High Measurement Likelihood

We presented anecdotal evidence that the a distinct frequency domain values $\{y_c\}$ constructed in the last section do produce highly likely measurement results, but we didn't even try to prove it. It's time to do that now. This is the messiest math of the lecture. I'll try to make it clear by offering pictures and gradual steps.

When we last checked, the (conceptual) measurement/collapse of B register to state $|f(x_0)\rangle$ left the post- \mathcal{QFT} A register in the state

$$\frac{1}{\sqrt{\tilde{m}N}} \sum_{y=0}^{N-1} \omega^{x_0 y} \left(\sum_{j=0}^{\tilde{m}-1} \omega^{j a y} \right) |y\rangle^n.$$

The probability of measuring the state $|y\rangle$ is the amplitude's magnitude squared:

$$\begin{aligned} P(\text{measurement yields } y) &= \frac{1}{\tilde{m}N} |\omega^{x_0 y}|^2 \left| \sum_{j=0}^{\tilde{m}-1} \omega^{j a y} \right|^2 \\ &= \frac{1}{\tilde{m}N} \left| \sum_{j=0}^{\tilde{m}-1} \omega^{j a y} \right|^2 \end{aligned}$$

Letting

$$\mu \equiv \omega^{a y},$$

the summation factor on the right (prior to magnitude-squaring) is

$$\begin{aligned}\sum_{j=0}^{\tilde{m}-1} \mu^j &= 1 + \mu + \mu^2 + \cdots + \mu^{\tilde{m}-1} \\ &= \frac{\mu^{\tilde{m}} - 1}{\mu - 1}.\end{aligned}$$

Having served its purpose, μ can now be jettisoned, and

$$\begin{aligned}\frac{\mu^{\tilde{m}} - 1}{\mu - 1} &= \frac{\omega^{ay\tilde{m}} - 1}{\omega^{ay} - 1} = \frac{e^{\frac{2\pi i}{N}ay\tilde{m}} - 1}{e^{\frac{2\pi i}{N}ay} - 1} \\ &= \frac{e^{i\theta_y\tilde{m}} - 1}{e^{i\theta_y} - 1},\end{aligned}$$

where we are defining the “angle”

$$\theta_y \equiv \frac{2\pi ay}{N}.$$

In other words, the probability of measuring any y is

$$P(\text{measurement yields } y) = \frac{1}{\tilde{m}N} \left| \frac{e^{i\theta_y\tilde{m}} - 1}{e^{i\theta_y} - 1} \right|^2$$

The next several screens are filled with the math to estimate the magnitude of the fraction,

$$\left| \frac{e^{i\theta_y\tilde{m}} - 1}{e^{i\theta_y} - 1} \right|.$$

We will do it by bounding numerator and denominator, separately. It’s a protracted side-trip because I go through each step slowly, so don’t be intimidated by the length of the derivation.

A General Bound for $|e^{i\phi} - 1|$

It will help to bracket the expression:

$$? \leq |e^{i\phi} - 1| \leq ?$$

An Upper Bound for $|e^{i\phi} - 1|$

In the complex plane, $|e^{i\phi} - 1|$ is the length of the chord from 1 to $e^{i\phi}$, and this is always \leq the arc length from 1, counterclockwise, to $e^{i\phi}$, all on the unit circle. But the arc length is, by definition, the (absolute value of the) angle, ϕ , itself, so:

$$|e^{i\phi} - 1| \leq |\phi|.$$

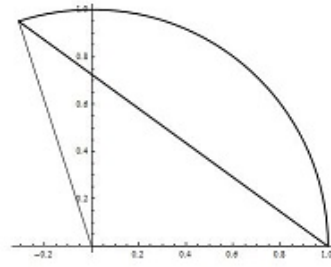


Figure 23.26: The chord is shorter than the arc length

A Lower Bound for $|e^{i\phi} - 1|$

By the Euler identity

$$1 - e^{i\phi} = 1 - (\cos \phi + i \sin \phi).$$

By the addition laws for sine and cosine applied to $\phi = \frac{\phi}{2} + \frac{\phi}{2}$,

$$\begin{aligned} \cos \phi &= \cos^2 \left(\frac{\phi}{2} \right) - \sin^2 \left(\frac{\phi}{2} \right) \\ \sin \phi &= 2 \cos \left(\frac{\phi}{2} \right) \sin \left(\frac{\phi}{2} \right) \end{aligned}$$

So

$$\begin{aligned} 1 - e^{i\phi} &= 1 - \left(\left[\cos^2 \left(\frac{\phi}{2} \right) - \sin^2 \left(\frac{\phi}{2} \right) \right] \right. \\ &\quad \left. + i \left[2 \cos \left(\frac{\phi}{2} \right) \sin \left(\frac{\phi}{2} \right) \right] \right) \\ &= 1 - \cos^2 \left(\frac{\phi}{2} \right) + \sin^2 \left(\frac{\phi}{2} \right) \\ &\quad - i 2 \cos \left(\frac{\phi}{2} \right) \sin \left(\frac{\phi}{2} \right) \\ &= 2 \sin^2 \left(\frac{\phi}{2} \right) - i 2 \cos \left(\frac{\phi}{2} \right) \sin \left(\frac{\phi}{2} \right) \\ &= 2 \sin \left(\frac{\phi}{2} \right) \left(\sin \left(\frac{\phi}{2} \right) - i \cos \left(\frac{\phi}{2} \right) \right) \end{aligned}$$

Noting that

$$\left| \sin \left(\frac{\phi}{2} \right) - i \cos \left(\frac{\phi}{2} \right) \right|^2 = 1,$$

we have shown that

$$|e^{i\phi} - 1| = 2 \left| \sin \left(\frac{\phi}{2} \right) \right|.$$

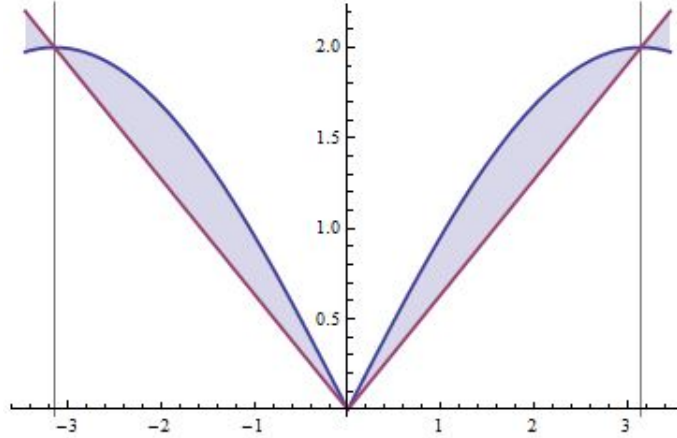


Figure 23.27: $2|\sin(x/2)|$ lies above $2|x/\pi|$ in the interval $(-\pi, \pi)$

By simple calculus and solving $2\phi/\pi = 2\sin(\phi/2)$ for ϕ , or noting where the graph of the sine curve and the line intersect, we conclude that when ϕ is in the interval $[-\pi, \pi]$, we can bound $2|\sin(\phi/2)|$ from below,

$$\frac{2|\phi|}{\pi} \leq 2\left|\sin\left(\frac{\phi}{2}\right)\right|, \quad \text{for } \phi \in [-\pi, \pi].$$

This gives us a lower bound for $|e^{i\phi} - 1|$:

$$\frac{2|\phi|}{\pi} \leq |e^{i\phi} - 1|, \quad \text{for } \phi \in [-\pi, \pi].$$

Combining both bounds when ϕ is in the interval $[-\pi, \pi]$, we have bracketed the expression under study,

$$\frac{2|\phi|}{\pi} \leq |e^{i\phi} - 1| \leq |\phi|, \quad \text{for } \phi \in [-\pi, \pi].$$

Applying Both Bounds to the Fraction

Our goal was to estimate

$$\left| \frac{e^{i\theta_y \tilde{m}} - 1}{e^{i\theta_y} - 1} \right|,$$

especially when $y = y_c$. We prefer to work with the \hat{y}_c which are in an a -sized interval around 0:

$$\hat{y}_c = y_c a - cN \in \left[-\frac{a}{2}, +\frac{a}{2} \right), \quad c = 0, 1, 2, \dots, a-1$$

So, let's first convert the exponentials in the numerator and denominator using the above relation between those special $\{y_c\}$ (which we hope to measure with high probability) and their corresponding mod N equivalents, the $\{\widehat{y}_c\}$, close to the origin.

$$\begin{aligned} e^{i\theta_{y_c}} &= e^{i(2\pi a y_c/N)} = e^{i(2\pi [\widehat{y}_c + cN]/N)} = e^{i(2\pi \widehat{y}_c/N)} e^{i2\pi c} \\ &= e^{i(2\pi \widehat{y}_c/N)} = e^{i\theta_{\widehat{y}_c}/a} \end{aligned}$$

This allows us to rewrite the absolute value of the amplitudes we wish to estimate as

$$\left| \frac{e^{i\theta_y \widetilde{m}} - 1}{e^{i\theta_y} - 1} \right| = \left| \frac{e^{i\theta_{\widehat{y}_c} \widetilde{m}/a} - 1}{e^{i\theta_{\widehat{y}_c}/a} - 1} \right|.$$

We want a *lower bound* for this magnitude. This way we can see that the likelihood of measuring these relatively few y s is high. To that end, we get

- a lower bound for the numerator, and
- an upper bound for the denominator.

Upper Bound for Denominator

The denominator is easy. We derived an upper bound for all angles, ϕ , so:

$$|e^{i\theta_{\widehat{y}_c}/a} - 1| \leq \left| \frac{\theta_{\widehat{y}_c}}{a} \right|.$$

Lower Bound for Numerator

The numerator is

$$|e^{i\theta_{\widehat{y}_c} \widetilde{m}/a} - 1| = |e^{i(2\pi \widehat{y}_c/N) \widetilde{m}} - 1|.$$

Remember that \widetilde{m} is sometimes m and sometimes $m + 1$.

Sub-Case 1: \widetilde{m} is m

When \widetilde{m} is m , we have things easy: $2\pi m \widehat{y}_c / N \in (-\pi, \pi)$ because

$$-\frac{a}{2} \leq \widehat{y}_c < \frac{a}{2},$$

or

$$-\pi \frac{am}{N} \leq \frac{2\pi m}{N} \widehat{y}_c < \pi \frac{am}{N}.$$

And, since

$$\frac{am}{N} < 1,$$

we get

$$-\pi < \frac{2\pi m}{N} \hat{y}_c < \pi.$$

This allows us to invoke the lower bound we derived when $\phi \in [\pi, \pi]$, namely

$$\left| e^{i 2\pi m \hat{y}_c / N} - 1 \right| \geq \frac{2 \left| 2\pi m \hat{y}_c / N \right|}{\pi}.$$

Re-applying the notation

$$\theta_y \equiv \frac{2\pi a y}{N},$$

we write it as

$$\left| e^{i 2\pi m y_c / N} - 1 \right| \geq \frac{2 \left| \frac{m}{a} \theta_{\hat{y}_c} \right|}{\pi}.$$

Combining the bounds for the numerator and denominator (in the case where $\tilde{m} = m$), we end up with

$$\left| \frac{e^{i \theta_{\hat{y}_c} \tilde{m}/a} - 1}{e^{i \theta_{\hat{y}_c}/a} - 1} \right| \geq 2 \frac{\left| \frac{m}{a} \theta_{\hat{y}_c} \right|}{\pi} \bigg/ \left| \frac{\theta_{\hat{y}_c}}{a} \right| = \frac{2m}{\pi}$$

Sub-Case 2: \tilde{m} is $m + 1$ (Deferred)

That only worked for $\tilde{m} = m$; the bounds argument we presented won't hold up when $\tilde{m} = m + 1$, but we'll deal with that kink in a moment. Let's pretend the above bound works for *all* \tilde{m} , both m and $m + 1$, and finish computing the probabilities under this white lie. Then we'll come back and repair the argument to also include $\tilde{m} = m + 1$.

Simulating the Probabilities Using the $\tilde{m} = m$ Bounds

If the above bounds worked for *all* \tilde{m} , we could show that this leads to a desired lower bound for our probabilities of getting a desired $y \in \mathcal{C}$ in constant time using the following argument.

For any y , we said

$$P(\text{measurement yields } y) = \frac{1}{\tilde{m}N} \left| \frac{e^{i \theta_y \tilde{m}} - 1}{e^{i \theta_y} - 1} \right|^2,$$

but for $y = y_c$, one of the (a) special y s that lie in the neighborhoods of the cN s, we can substitute our new-found lower bound for the magnitude of the fraction. (Remember, we are allowing that the bound holds for all \tilde{m} , even though we only proved it for $\tilde{m} = m$), so

$$P(\text{measurement yields } y_c) \geq \frac{1}{\tilde{m}N} \frac{4\tilde{m}^2}{\pi^2} \geq \frac{m}{N} \frac{4}{\pi^2}$$

The last inequality holds because some of the \tilde{m} are $m + 1$ and some are m . Now, there are a such y_c , namely, y_0, y_1, \dots, y_{a-1} , each describing a distinct, independent, measurement (we can't get *both* y_{32} and y_{81}). Thus, the probability of getting *any one of them* is the sum of the individual probabilities. Since those individual probabilities are all bounded below by the same constant, we can multiply it by a to get the collective lower bound,

$$P(\text{measurement yields one of the } y_c) \geq \frac{am}{N} \frac{4}{\pi^2}$$

[**Exercise.** Make this last statement precise.]

In this hard case, allowing $\neg a|N$, we defined m to be the unique integer satisfying $ma \leq N < (m + 1)a$, or quoting only the latter inequality,

$$(m + 1)a > N.$$

It's now time to harvest that "weak additional assumption" requiring at least two periods of a to fit into M ,

$$a < \frac{M}{2},$$

which also implies

$$a < \frac{N}{2}.$$

We combine those to conclude

$$\frac{am}{N} > \frac{1}{2}.$$

[**Exercise.** Do it.]

We can plug that result into our probability estimate to get

$$P(\text{measurement yields one of the } y_c) > \frac{2}{\pi^2}.$$

Note-to-file. If $a \ll M < N$, as is often the case, then

$$\frac{am}{N} \approx 1,$$

and the above lower bound improves to

$$P(\text{measurement yields one of the } y_c) \geq \frac{4}{\pi^2} - \varepsilon,$$

where $\varepsilon \rightarrow 0$ as $m \rightarrow \infty$

[**Exercise.** Compute the lower bound when $m = 100$.]

Our last lower bound for the “ p of success,” $2/\pi^2 > 0$, was independent of M (or N or a), so by repeating the random measurement a fixed number, T , times, we can assure we measure one of those y_c with any desired level of confidence. This follows from the *CTC theorem for looping algorithms* (end of probability lesson), but for a derivation that does not rely on any theorems, compute directly,

$$\begin{aligned}
& P(\text{we do not get a } y_c \text{ after } T \text{ measurements}) \\
&= P\left(\bigwedge_{k=1}^T \neg \text{measurement } T_k \text{ yields a } y_c\right) \\
&= \prod_{k=1}^T P(\neg \text{measurement } T_k \text{ yields a } y_c) \\
&= \prod_{k=1}^T \left(1 - P(\text{measurement } T_k \text{ yields a } y_c)\right) \\
&\leq \prod_{k=1}^T \left(1 - \frac{2}{\pi^2}\right).
\end{aligned}$$

The last product can be made arbitrarily small by making T large enough, independent of N , M , a , etc. This would prove the claim of Step 2 if we could only use the bound we got for $\tilde{m} = m$. But we can't. So we must soldier on ...

Repeating the Estimates when $\tilde{m} = m + 1$ Bounds

We now repair the convenient-but-incorrect assumption that \tilde{m} is always m . To do so, let's repeat the estimates, but do so for $\tilde{m} = m + 1$. This is a little harder sub-case. When we're done, we'll combine both cases.

Remember where \tilde{m} came from, and why it could be either m or $m + 1$. In our general (hard) case, a does not divide N evenly; the first few $x \in R = [0, a - 1]$ will generate $m + 1 \bmod a$ relatives within $[0, N - 1]$ that map to the same $f(x)$, while the last several $x \in R = [0, a - 1]$ will only produce m such $\bmod a$ relatives within $[0, N - 1]$. \tilde{m} represented however many $\bmod a$ relatives of x fit into the $[0, N - 1]$ interval: m for some x , and $m + 1$ for others.

We retrace our steps.

The probability of measuring the state $|y\rangle$ is the amplitude's magnitude squared,

$$\begin{aligned}
P(\text{measurement yields } y) &= \frac{1}{\tilde{m}N} \left| \sum_{j=0}^{\tilde{m}-1} \omega^{jay} \right|^2 \\
&= \frac{1}{\tilde{m}N} \left| \frac{e^{i\theta_y \tilde{m}} - 1}{e^{i\theta_y} - 1} \right|^2.
\end{aligned}$$

So far, we're okay; we had not yet made any assumption about the particular choice of \tilde{m} . To bound this probability, we went on to get an estimate for the fraction

$$\left| \frac{e^{i\theta_{\hat{y}_c} \tilde{m}/a} - 1}{e^{i\theta_{\hat{y}_c}/a} - 1} \right|.$$

The denominator's upper bound worked for *any* θ , so no change needed there. But the numerator's lower bound has to be recomputed, this time, under the harsher assumption that $\tilde{m} = m + 1$.

Earlier, we showed that $2\pi m \hat{y}_c / N$ was confined to the interval $(-\pi, \pi)$, which gave us our desired result. Now, however, we replace m with $m + 1$, and we'll see that $2\pi(m + 1) \hat{y}_c / N$ won't be restricted to $(-\pi, \pi)$. What, exactly, *are* its limits? Start, as before,

$$-\frac{a}{2} \leq \hat{y}_c < \frac{a}{2},$$

but we need to multiply by a different constant this time,

$$-\pi \frac{a(m + 1)}{N} \leq \frac{2\pi(m + 1)}{N} \hat{y}_c < \pi \frac{a(m + 1)}{N}.$$

By our working assumption, $a/N < 2$ (which continues to reap benefits) we can assert that

$$\frac{a(m + 1)}{N} = \frac{am}{N} + \frac{a}{N} < 1 + \frac{1}{2},$$

giving us the new absolute bounds

$$-\frac{3}{2}\pi < \frac{2\pi(m + 1)}{N} \hat{y}_c < \frac{3}{2}\pi.$$

The next step would have been to apply the general result we found for any $\phi \in [-\pi, \pi]$, namely that

$$\frac{2|\phi|}{\pi} \leq 2 \left| \sin \left(\frac{\phi}{2} \right) \right|, \quad \text{for } \phi \in [-\pi, \pi].$$

But now our " ϕ " = $2\pi \hat{y}_c(m + 1)/N$ lives in the enlarged interval $[-(3/2)\pi, (3/2)\pi]$, so that general result no longer applies. Sigh. We have to go back and get new general result for this larger interval. It turns out that old bound merely needs to be multiplied by a constant:

$$K \frac{2|\phi|}{\pi} \leq 2 \left| \sin \left(\frac{\phi}{2} \right) \right|, \quad \text{for } \phi \in \left[-\frac{3}{2}\pi, \frac{3}{2}\pi \right],$$

where K is the constant

$$K = \frac{2 \sin \frac{3\pi}{4}}{3} \approx .4714.$$

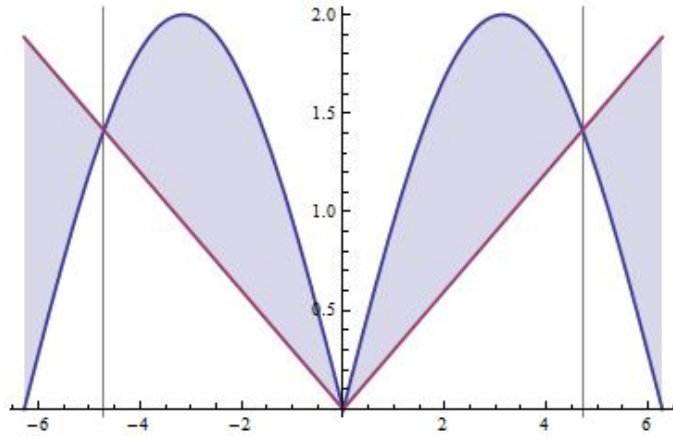


Figure 23.28: $2|\sin(x/2)|$ lies above $2|Kx/\pi|$ in the interval $(-1.5\pi, 1.5\pi)$

Again, this can be done using calculus and solving $2K\phi/\pi = 2\sin(\phi/2)$ for K . Visually, we can see where the graphs of the sine and the line intersect, which confirms that assertion. Summarizing the general result in the expanded interval,

$$|e^{i\phi} - 1| = 2 \left| \sin\left(\frac{\phi}{2}\right) \right| \geq \frac{2K|\phi|}{\pi}, \quad \text{for } \phi \in \left[-\frac{3}{2}\pi, \frac{3}{2}\pi\right].$$

This gives us the actual bound we seek in the case $\tilde{m} = m + 1$, namely

$$|e^{i2\pi(m+1)\hat{y}_c/N} - 1| \geq \frac{2K|2\pi(m+1)\hat{y}_c/N|}{\pi}.$$

Re-applying the notation

$$\theta_y \equiv \frac{2\pi ay}{N},$$

we get

$$|e^{i2\pi(m+1)y_c/N} - 1| \geq \frac{2K \left| \frac{m+1}{a} \theta_{\hat{y}_c} \right|}{\pi}.$$

Combining the bounds for the numerator and denominator (in the case where $\tilde{m} = m + 1$),

$$\begin{aligned} \left| \frac{e^{i\theta_{\hat{y}_c}\tilde{m}/a} - 1}{e^{i\theta_{\hat{y}_c}/a} - 1} \right| &\geq \frac{2K \left| \frac{m+1}{a} \theta_{\hat{y}_c} \right|}{\pi} \bigg/ \left| \frac{\theta_{\hat{y}_c}}{a} \right| \\ &= \frac{2K(m+1)}{\pi} = \frac{2K\tilde{m}}{\pi}. \end{aligned}$$

Finishing the Probability Estimates by Combining Both Cases $\tilde{m} = m$ and $\tilde{m} = m + 1$

Remember that when $\tilde{m} = m$, we had the stronger bound:

$$\left| \frac{e^{i\theta_{\hat{y}_c}\tilde{m}/a} - 1}{e^{i\theta_{\hat{y}_c}/a} - 1} \right| \geq \frac{2m}{\pi} = \frac{2\tilde{m}}{\pi},$$

so we can use the new, weaker, bound to cover both cases. For *all* \tilde{m} , both m and $m + 1$, we have

$$\left| \frac{e^{i\theta_{\hat{y}_c} \tilde{m}/a} - 1}{e^{i\theta_{\hat{y}_c}/a} - 1} \right| \geq \frac{2K\tilde{m}}{\pi},$$

for

$$K = \frac{2 \sin \frac{3\pi}{4}}{3} \approx .4714.$$

We can reproduce the final part of the $\tilde{m} = m$ argument, incorporating K into the inequalities

$$P(\text{measurement yields } y_c) \geq \frac{1}{\tilde{m}N} \frac{4K^2\tilde{m}^2}{\pi^2} \geq \frac{m}{N} \frac{4K^2}{\pi^2}.$$

The last inequality still acknowledges that some of the \tilde{m} are $m + 1$ and some are m . Again, there are a such y_c , namely, y_0, y_1, \dots, y_{a-1} , so the probability of getting any one of them is (\geq) a times this number,

$$\begin{aligned} P(\text{measurement yields one of the } y_c) &\geq \frac{am}{N} \frac{4K^2}{\pi^2} \\ &> \frac{1}{2} \frac{4K^2}{\pi^2} = \frac{2K^2}{\pi^2} \approx .04503. \end{aligned}$$

[**Exercise.**] Compute K when $m = 100$.

Note-to-file. If $a \ll M < N$ (i.e., m gets very large), as is often the case, two things happen. First,

$$\frac{(m+1)}{N} \approx \frac{m}{N},$$

so we can use our first estimate, $\tilde{m} = m$, for measuring a y_c , even in the general case. That estimate gave us

$$P(\text{measurement yields one of the } y_c) > \frac{2}{\pi^2}.$$

Second, our earlier “Note-to-file” suggested that, for $\tilde{m} = m$,

$$\begin{aligned} P(\text{measurement yields one of the } y_c) &> \frac{4}{\pi^2} - \varepsilon, \\ \text{where } \varepsilon &\rightarrow 0 \text{ as } m \rightarrow \infty. \end{aligned}$$

Putting these two observations together, we conclude that when $a \ll N$, the lower bound for *any* measurement is $\approx 4/\pi^2$. (“*any*” means it doesn’t matter whether the state to which the second register collapsed, $|f(x_0)\rangle$, is associated with an x_0 for which there were m or $m + 1 \bmod a$ equivalents in $[0, N - 1]$).

So we have both a hard bound assuming worst case scenarios (the period, a , cycles no more than twice in M) and the more likely scenario, $a \ll M$. Symbolically,

$$P(\text{measurement yields one of the } y_c) > \begin{cases} .04503, & \text{worst case: } 3a > M \\ .40528, & \text{typically} \end{cases}.$$

In the worst case, the smaller lower bound doesn't change a thing; it's still a constant probability bounded away from zero, independent of N, M, a , etc., and it still gives us a constant time, T , for detecting one of our special y_c with arbitrarily high confidence. This, again, follows from the *CTC Theorem for looping algorithms*, or you can simply apply probability theory directly.

In practice, $a \ll M$, so we can use .40528 as our constant “ p of success” bounded away from 0. If we are satisfied with an $\varepsilon = 10^{-6}$ of failure, the CTC theorem tells us that the number of passes of our circuit would be

$$T = \left\lceil \frac{\log(10^{-6})}{\log(.59472)} \right\rceil + 1 = \left\lceil \frac{-6}{-.2256875} \right\rceil + 1 = 26 + 1 = 27.$$

If we sampled y 27 times our chances of *not* measuring at least one y_c is less than of one in a million.

This completes the proof of the fact that we will measure one of the a y_c in constant time. Our next task is to demonstrate that by measuring relatively few of these y_c we will be able to determine the period. This will be broken into small, bite-sized steps.

23.10.8 STEP III: Associate $\{y_c\}_{c=0}^{a-1}$ with $\{c/a\}_{c=0}^{a-1}$

We now know that we'll measure one of those y_c with very good probability if we sample enough times ($O(1)$ complexity). But, what's our real goal here? We'd like emulate the *easy case* by finding a number, c , that's *coprime* to a in constant time and used it compute a . In the general case, however, c is a mere *index* of the y_c , not a *multiplier* that produced the *cm* of the *easy case*. What can we hope to know about the c which just indexes the set $\{y_c\}$? You'd be surprised.

In this step, we demonstrate that each of these special, likely-measured y_c values is bound tightly to the fraction c/a in a special way: y_c/N will turn out to be extremely (and uniquely) close to c/a . This, in itself, should feel a little like magic: somehow the index of the likely-measured set of y s shows up in the numerator of a fraction that is close to y_c/N . Let's pull back the curtain.

Do you remember those (relatively small) half-open intervals of width a around

the points cN ,

$$\begin{aligned} & \left[-\frac{a}{2}, +\frac{a}{2} \right), \quad \left[N - \frac{a}{2}, N + \frac{a}{2} \right), \\ & \quad \cdots \quad \left[cN - \frac{a}{2}, cN + \frac{a}{2} \right) \quad \cdots \\ & \quad \quad \cdots \quad \left[(a-1)N - \frac{a}{2}, (a-1)N + \frac{a}{2} \right), \end{aligned}$$

each of which contained *exactly one* integral multiple, $y_c a$, of a in it?

$$\begin{aligned} 0a \in \left[-\frac{a}{2}, +\frac{a}{2} \right), \quad \cdots, \quad y_c a \in \left[cN - \frac{a}{2}, cN + \frac{a}{2} \right), \\ \cdots, \quad y_{a-1} a \in \left[(a-1)N - \frac{a}{2}, (a-1)N + \frac{a}{2} \right) \end{aligned}$$

Let's use the relationship of those y_c with their host intervals. Start with the obvious,

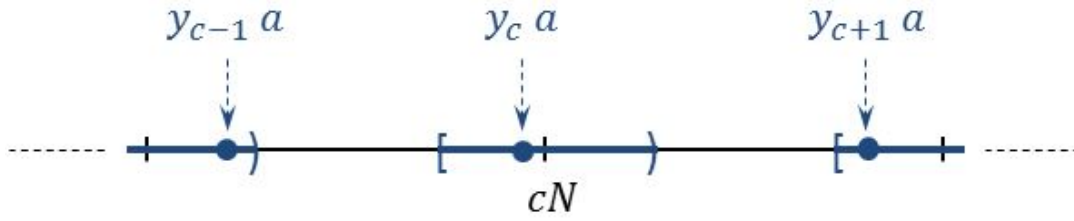


Figure 23.29: Exactly one integral multiple of a falls in each interval

$$cN - \frac{a}{2} \leq y_c a < cN + \frac{a}{2},$$

then divide by N ,

$$\frac{c}{a} - \frac{1}{2N} \leq \frac{y_c}{N} < \frac{c}{a} + \frac{1}{2N}.$$

In other words, y_c/N is within $1/(2N)$ of c/a or, symbolically,

$$\left| \frac{c}{a} - \frac{y_c}{N} \right| \leq \frac{1}{2N}.$$

Remember, the relationship between M and N was defined by

$$\frac{N}{2} < M^2 \leq N,$$

so $N \geq M^2$. Applying that to the above inequality involving y_c/N , we get,

$$\left| \frac{c}{a} - \frac{y_c}{N} \right| \leq \frac{1}{2M^2}.$$

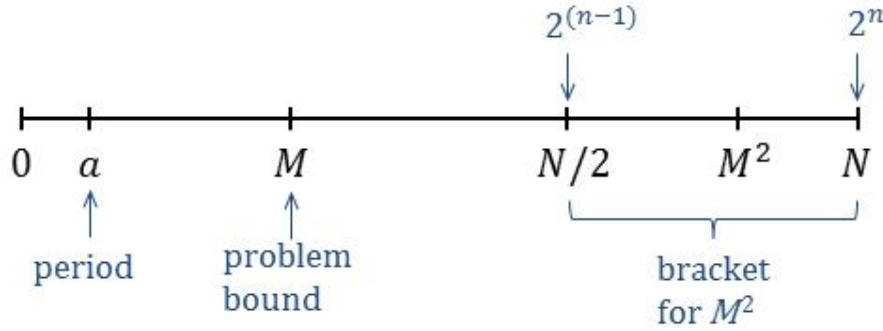


Figure 23.30: $N = 2^n$ chosen so $(N/2, N]$ bracket M^2

Compare that with the distances between consecutive c/a values (which I will prove in a moment),

$$\left| \frac{c}{a} - \frac{c+1}{a} \right| \geq \frac{1}{M^2},$$

and we can conclude that each of our special y_c divided by N is closer to c/a than it is to any other fraction with denominator a . Thus, when we measure one of these y_c (which we already showed we can do in constant time with arbitrarily good probability) we will be picking out the rational number c/a . In fact, we will show something stronger, but first, we fill in the gap and demonstrate why consecutive fractions c/a and $(c+1)/a$ differ by, at least, $1/M^2$.

Lemma. *If two distinct integers satisfy $0 < p, q \leq M$, then for any distinct rationals $\frac{l}{p}$ and $\frac{k}{q}$, we are guaranteed that*

$$\left| \frac{k}{p} - \frac{l}{q} \right| \geq \frac{1}{M^2}.$$

Use this with $p = q \leftarrow a$, $k \leftarrow c$ and $l \leftarrow c+1$, and you get the aforementioned lower bound for $|c/a - (c+1)/a|$.

Proof. Assume $k/p > l/q$. (Reverse the following if not).

$$\frac{k}{p} - \frac{l}{q} = \frac{kq - lp}{pq} \geq \frac{kq - lp}{M^2} \geq \frac{1}{M^2}. \quad \text{QED}$$

This lemma tells us that c/a is not only the best fractional approximation to y_c with denominator a , but it's best among *all* fractions having denominators $\leq M$. For letting n/d be any fraction with denominator $d \leq M$. The lemma says

$$\left| \frac{c}{a} - \frac{n}{d} \right| \geq \frac{1}{M^2},$$

which places n/d squarely outside the $1/(2M^2)$ neighborhood that contains both c/a and y_c/N .

Conclusion: Of all fractions n/d with denominator $d \leq M$, c/a is the only one that lies in the neighborhood of “radius” $1/(2M^2)$ around y_c/N . Thus y_c/N strongly selects c/a and vice versa.

[**Interesting Observation.** We showed that

$$\frac{y_c}{N} \text{ is uniquely close to } \frac{c}{a}.$$

But multiply both sides by N to get to the equivalent conclusion,

$$y_c \text{ is uniquely close to } c \left(\frac{N}{a} \right).$$

What is N/a ? It is *freq*, the *exact frequency* of our function f relative to the interval $[0, N - 1]$. (In general, it’s a real number between m and $m + 1$.) In other words, each likely-measured y_c is uniquely close to an integer multiple of the function’s exact frequency,

$$y_c \text{ is uniquely close to } c \cdot \text{freq}.$$

Our math doesn’t require that we recognize this fact, but it does provide a nice parallel with the *easy case*, in which our measured $\{y_c\}$ were exact multiples, $\{cm\}$, of the true integer frequency, $\text{freq} = m$.]

23.10.9 STEP IV: Describe an $O(\log^3 N)$ Algorithm that Will Produce c/a from y_c

Highlights of Continued Fractions

The *continued fractions algorithm* (CFA) will be developed in an optional lesson next week. In short, it takes a real number x as input and produces a sequence of (reduced) fractions, $\{n_k/d_k\}$ that approach (get closer and closer to) x . We will be applying CFA to $x = y_c/N$, a rational number itself, but we still want these other fractional approximations because among them we’ll find one, n/d , which is identical to our sought-after c/a .

Caution: There is no guarantee that the $c \leftrightarrow y_c$ satisfies $c \nmid a$. So when CFA tells us that it has found the *reduced* fraction $n/d = c/a$, we will not be able to conclude that $n = c$ and $d = a$. We will deal with that wrinkle in **Step V**.

The plan is to list a few results about CFA, then use those facts to show that it will produce (*return*) the unique a -denominator fraction, c/a , closest to y_c/N , in $O(\log^3 M)$ operations. (Reminder: M is the known upper-bound we were given for a).

Here are the properties of CFA that we’ll need. Some of them are only true when x is rational, but for us, that’s the case.

1. During execution, CFA consists of a loop that produces a fraction in reduced form, n_k/d_k , at the end of the k th iteration. n_k/d_k is called the k th *convergent* for x .
2. For any real number, x , the convergents approach x (thus justifying their name), that is

$$\lim_{k \rightarrow \infty} \left(\frac{n_k}{d_k} \right) = x.$$

(n stands for *numerator* here, as in **n**umerator/**d**enomintor, not the n in the exponent of Shor's $N = 2^n$.)

3. For rational x , the above limit is finite, i.e., there will be a $K < \infty$, with $n_K/d_K = x$, exactly, and no more fractions are produced for $k > K$.
4. In our version of CFA, we will supply a requested degree of accuracy ε , and ask CFA to return n/d , the first fraction it generates which is within ε of x .
Depending on ε and x , CFA either returns $n/d = n_K/d_K = x$, exactly, as its final convergent, or returns an ε -approximation $n/d \neq x$, but within ε of it.
5. The denominators $\{d_k\}$ are strictly increasing and, for rational x , all \leq the denominator of x (whether or not x was given to us in reduced form).
6. If a fraction n/d differs from x by less than $1/(2d^2)$ then n/d will appear in the list of convergents for x . Symbolically, if

$$\left| \frac{n}{d} - x \right| < \frac{1}{2d^2},$$

then

$$\frac{n}{d} = \frac{n_{k_0}}{d_{k_0}} \in \left\{ \frac{n_k}{d_k} \right\}_{k=0}^K \quad (\text{the convergents for } x).$$

7. When $x = p/q$ is a rational number, CFA will complete in $O(\log^3 q)$. (Sharper bounds exist, but this is enough for our purposes and is easy to explain.)

Procedure for Using CFA to Produce c/a

We apply CFA to $x = y_c/N$ and $\varepsilon = 1/(2M^2)$.

Claim. CFA will produce and return the unique c/a within $1/(2M^2)$ of y_c/N .

Proof. $a < M$, so

$$\frac{1}{2M^2} < \frac{1}{2a^2},$$

but in step III we showed

$$\left| \frac{c}{a} - \frac{y_c}{N} \right| < \frac{1}{2M^2},$$

so

$$\left| \frac{c}{a} - \frac{y_c}{N} \right| < \frac{1}{2a^2}.$$

As this is the hypothesis of **bullet 6**, c/a must appear among the convergents of y_c/N . Since it is within $1/(2M^2)$ of y_c/N , we know that CFA will terminate when it reaches c/a , if not before.

We now show that CFA cannot terminate before its loop produces c/a . If CFA returned a convergent n_k/d_k that preceded c/a , we would have

$$\left| x - \frac{n_k}{d_k} \right| \leq \frac{1}{2M^2}$$

by **bullet 4**. But since the d_k are strictly increasing (**bullet 5**), and we are saying that the algorithm terminated *before* getting to c/a , then

$$d_k < a.$$

That would give us a *second* fraction, n_k/d_k with denominator $d_k < M$ within $1/(2M^2)$ of y_c/N , a title uniquely held by c/a (from Step III). Therefore, when we give CFA the inputs $x = y_c/N$ and $\varepsilon = 1/(2M^2)$, it must produce and return c/a .
QED

CFA is $O(\log^3 M)$

By **Bullet 7** CFA has time complexity $O(\log^3 N)$, which we have already established to be equivalent to $O(\log^3 M)$.

23.10.10 Step V: Measure y_c Associated with c Coprime to a in Constant Time

In **Steps I and II** we proved that the likelihood of measuring one of the special $\{y_c\}_{c=0}^{a-1}$ was always bounded below by a constant, independent of N . (The constant may depend on how many periods, a , fit into M , but for RSA encryption-breaking, we will see that the special case we require will assure us at least two periods, and normally, many more).

In the *easy case*, the $\{y_c\}$ were *all equally likely*, and we also knew that there were *exactly* $a/2$ *coprimes* $< a$. We combined those two facts to put the proof to bed. Here, neither condition holds, so we have to work harder (which is why this isn't called the "*easy case*").

What we *can* say is that, from **Steps III and IV**, each of the measured $\{y_c\}$ s leads – in *constant time*, with any predetermined confidence, ε – to a partner fraction in $\{c/a\}$ with the help of some $O(\log^3 M)$ logic provided by CFA.

In this step, we demonstrate that not only do we measure *some* y_c (constant time) and get a partner, $\{c/a\}$ ($O(\log^3 M)$), but that we can even expect to get a special subset $\mathcal{B} \subseteq \{y_c\} \leftrightarrow \{c/a\}$ in constant time, namely those y_c corresponding to $c \not\equiv a$ (i.e., c coprime to a). This will enable us to extract the period a from c/a .

We do it all in three steps, the first two of which correspond to the missing conditions we enjoyed in the *easy case*:

- Stated loosely, in our quantum circuit, the “difference” (really ratio) between measuring the least likely y_c and the most likely y_c is bounded by a fixed constant independent of a , M , etc. This means, that while the y_c have different likelihoods of being measured, they are within a fixed “probability interval.” (This corresponds to the equi-probabilities of the *easy case*.)
- The probability of selecting a number, c , which is coprime to a , at random, from the numbers between 0 and $a - 1$ is constant, independent of a . (This corresponds to the 50% likelihood of the *easy case*.)
- We combine the first two bullets to show that the probability of getting a $c \not\equiv a$ in any single pass of the circuit is bounded away from 0 by a constant independent of the size of the algorithm. That’s the requirement of the *CTC theorem for looping algorithms* and therefore guarantees we obtain such a c with small error tolerance ε in *constant time*, i.e., after a fixed number of loop passes independent of N .

Proof of First Bullet

This can be demonstrated by retooling an analysis analysis we already did. We established that the amplitude squared of a general $|y\rangle$ in our post- QFT A register was

$$P(\text{measurement yields } y) = \frac{1}{\tilde{m}N} \left| \frac{e^{i\theta_y \tilde{m}} - 1}{e^{i\theta_y} - 1} \right|^2.$$

There are clearly two measurement-dependent parameters that will affect this probability: \tilde{m} , which is either m or $m + 1$, depending on the collapsed state of the second register, and θ_y , which depends on the measured y . When $a \ll M$ the probabilities are very close to being uniform, but to avoid hand-waving, let’s go with our worst-case scenario – the assumption that we added to Shor’s hypothesis in order to get hard estimates for all our bounds: $M > 2a$, but not necessarily any larger.

When we computed our *lower bound* on the probability of getting a y_c , we used an inequality that contained an angle ϕ under the assumption that ϕ was restricted to

an interval *wider* than $[-\pi, \pi]$, specifcally $[-(3/2)\pi, (3/2)\pi]$. The inequality we found applicable was

$$K \frac{2|\phi|}{\pi} \leq 2 \left| \sin \left(\frac{\phi}{2} \right) \right|, \quad \text{for } \phi \in \left[-\frac{3}{2}\pi, \frac{3}{2}\pi \right],$$

where K is the constant

$$K = \frac{2 \sin \frac{3\pi}{4}}{3} \approx .4714.$$

The key to our current predicament is to get an *upper bound* on measuring any (even the most likely) y_c . This will lead us to get an inequality for general ϕ restricted to a *narrower* range, namely, $[-\pi/2, \pi/2]$. This can be easily determined using the same graphing or calculus techniques, and is

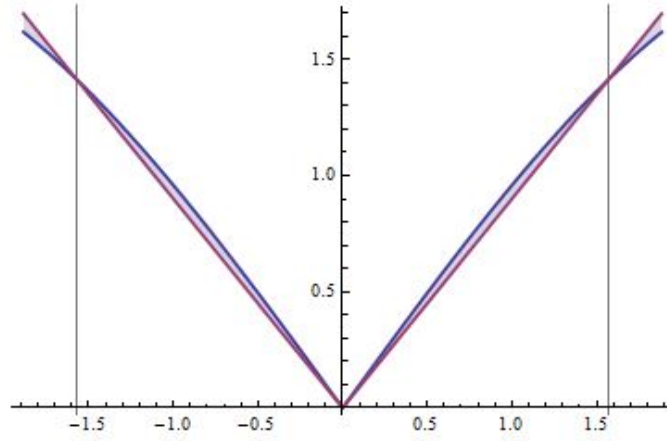


Figure 23.31: $2|\sin(x/2)|$ lies above $2|Lx/\pi|$ in the interval $(-.4714\pi, .4714\pi)$

$$L \frac{2|\phi|}{\pi} \leq 2 \left| \sin \left(\frac{\phi}{2} \right) \right|, \quad \text{for } \phi \in \left[-\frac{\pi}{2}, \frac{\pi}{2} \right],$$

where L is the constant

$$L = 2 \sin \frac{\pi}{4} \approx 1.4142.$$

Let's see how this gives us the upper bound we seek.

The probability we wish to bound, this time from above, comes from the amplitude whose absolute value is

$$\frac{1}{\sqrt{\tilde{m}N}} \left| \frac{e^{i\theta_y \tilde{m}} - 1}{e^{i\theta_y} - 1} \right| = \frac{1}{\sqrt{\tilde{m}N}} \left| \frac{e^{i\theta_{\hat{y}_c} \tilde{m}/a} - 1}{e^{i\theta_{\hat{y}_c}/a} - 1} \right|.$$

We want an *upper bound* for the magnitude of the fractional term. To that end, we get an upper bound for the numerator and a lower bound for the denominator.

Upper Bound for Numerator

The numerator is easy. We derived an upper bound for all angles, ϕ , so:

$$\left| e^{i\theta_{\hat{y}_c} \tilde{m}/a} - 1 \right| \leq \left| \frac{\theta_{\hat{y}_c} \tilde{m}}{a} \right|.$$

Lower Bound for Denominator

The denominator is

$$\left| e^{i\theta_{\hat{y}_c}/a} - 1 \right| = \left| e^{i(2\pi\hat{y}_c/N)} - 1 \right|,$$

but $2\pi\hat{y}_c/N \in (-\pi/2, \pi/2)$ because

$$-\frac{a}{2} \leq \hat{y}_c < \frac{a}{2}$$

or

$$-\pi \frac{a}{N} \leq \frac{2\pi}{N} \hat{y}_c < \pi \frac{a}{N}.$$

And since

$$\frac{a}{N} < 1/2,$$

we get

$$-\frac{\pi}{2} < \frac{2\pi}{N} \hat{y}_c < \frac{\pi}{2}.$$

This allows us to invoke the latest lower bound just mentioned for $\phi \in [\pi/2, \pi/2]$:

$$\left| e^{i2\pi\hat{y}_c/N} - 1 \right| \geq L \frac{2|2\pi\hat{y}_c/N|}{\pi}, \quad L = 2 \sin \frac{\pi}{4} \approx 1.4142.$$

Re-applying the notation

$$\theta_y \equiv \frac{2\pi ay}{N}$$

we can write

$$\left| e^{i2\pi y_c/N} - 1 \right| \geq L \frac{2|\theta_{\hat{y}_c}/a|}{\pi}.$$

Applying Both Bounds

Combining the bounds for the numerator and denominator,

$$\left| \frac{e^{i\theta_{\hat{y}_c} \tilde{m}/a} - 1}{e^{i\theta_{\hat{y}_c}/a} - 1} \right| \leq \left| \frac{\theta_{\hat{y}_c} \tilde{m}}{a} \right| \bigg/ L \frac{2 |\theta_{\hat{y}_c}/a|}{\pi} = \frac{2\tilde{m}}{L\pi}.$$

Finally,

$$P(\text{measurement yields } y_c) \leq \frac{1}{\tilde{m}N} \frac{4L^2 \tilde{m}^2}{\pi^2} \leq \frac{m+1}{N} \frac{4L^2}{\pi^2}.$$

The last inequality still acknowledges that some of the \tilde{m} are $m+1$ and some are m . This time, we are only interested in the probabilities for each individual y_c , not getting one of the set $\{y_c\}$, so instead of summing all a of the y_c s, we combine this, as is, with the earlier one, which I repeat here,

$$P(\text{measurement yields } y_c) \geq \frac{1}{\tilde{m}N} \frac{4K^2 \tilde{m}^2}{\pi^2} \geq \frac{m}{N} \frac{4K^2}{\pi^2},$$

to get

$$\frac{P(\text{least-likely } y_c)}{P(\text{most-likely } y_c)} \geq \frac{mK^2}{(m+1)L^2}.$$

Our standing assumption has been that $a \leq M/2$, so m (the integer quotient N/a) is ≥ 2 (usually much greater). This, and the estimates for $L \approx 1.4142$ and $K \approx .4714$, result in a ratio – independent of a , M or N – between the probability of measuring the least likely y_c and that of measuring the most likely y_c .

$$\frac{P(\text{least-likely } y_c)}{P(\text{most-likely } y_c)} \geq \frac{mK^2}{(m+1)L^2} \geq \frac{2K^2}{3L^2} \approx .072.$$

If you prefer, you can flip the fraction to see an equivalent way of bracketing the diversity of the probabilities,

$$\frac{P(\text{most-likely } y_c)}{P(\text{least-likely } y_c)} \leq \approx \frac{1}{.072} \approx 13.9. \quad \text{QED}$$

This proves the bullet about the ratio of the least likely and the most likely y_c . It should be thought of as a guarantee that the probabilities of the various y_c never deviate by an increasingly large relative amount. They’re “sort of all the same” in this regard, and it’s good enough to use instead of the stronger *equi-probable* $y = cm$ outcomes of the *easy case* (as we’re about to see).

Note-to-file. if $a \ll M < N$ (i.e., m gets very large), as is often the case, Both K and L will be close to 1 (review the derivations and previous notes-to-file). This means all y_c are roughly equi-probable. Also $m/(m+1) \approx 1$. Taken together, the ratio of the least likely to most likely is approximately 1.

Summarizing, we have a hard minimum for the worst case scenario (only two intervals of size a fit into $[0, M)$) as well as an expected minimum for the more realistic one ($a \ll M$). That is,

$$\frac{P(\text{least-likely } y_c)}{P(\text{most-likely } y_c)} \geq \begin{cases} .072, & \text{worst case: } 3a > M \\ 1 - \varepsilon, & \text{typically} \end{cases}.$$

Proof of Second Bullet

We will show that the probability of randomly selecting a number $c \nmid a$ is $> 60\%$.

First, we express this as a product of terms, each of which is the probability that a *specific* prime, $p|c$ (p divides evenly into c).

Note that, a is an unknown: no one a is favored over any other a , a priori. Also, c is selected at random, by hypothesis of this bullet. So both a and c are considered selected at random with respect to any particular prime, p . We use this fact in the derivation.

$$\begin{aligned} P(c \nmid a) &= P(\text{There is no prime that divides both } c \text{ and } a) \\ &= \\ P(\neg(2|c \wedge 2|a) \wedge \neg(3|c \wedge 3|a) \wedge \neg(5|c \wedge 5|a) \wedge \dots \\ &\quad \dots \wedge \neg(p_k|c \wedge p_k|a) \wedge \dots), \text{ where } p_k = k\text{th prime.} \\ P(c \nmid a) &= P\left(\bigwedge_{p \text{ prime}}^{\text{finite}} \neg(p|c \wedge p|a)\right) = \prod_{p \text{ prime}}^{\text{finite}} P(\neg(p|c \wedge p|a)) \end{aligned}$$

Since $\neg(p|c \wedge p|a)$ is true for $p > a$ or $p > c$, the probabilities for those higher primes, p , are all 1, which is why the product is finite.

Next, we compute these individual probabilities. For a fixed prime, p , the probability that it divides an arbitrary non-negative c chosen randomly from *all* non-negative integers is actually independent of c ,

$$P(p|c) = \frac{1}{p}.$$

[**Exercise.** Justify this. **Hint:** Graph all the c that p divides evenly and count the integers between each one.] This is also true for $p|a$, so,

$$\begin{aligned} P(p|c \wedge p|a) &= \frac{1}{p^2} \implies \\ P(\neg(p|c \wedge p|a)) &= 1 - \frac{1}{p^2}. \end{aligned}$$

Remember, both a and c can be considered randomly selected relative to a fixed prime, p . If, instead, we restrict ourselves to non-negative values chosen randomly from a *finite* set, such as $c < a$ (or $a < M$), then the probability that $p|c$ is actually $< 1/p$. To see this, consider the worst cases, say $a < p$ ($P = 0$) or $p \leq a < 2p$ ($P = 1/a < 1/p$). So the above equalities become bounds which work even better for us,

$$\begin{aligned} P(p|c) &< \frac{1}{p}, \\ P(p|c \wedge p|a) &< \frac{1}{p^2} \implies \\ P(\neg(p|c \wedge p|a)) &\geq 1 - \frac{1}{p^2}. \end{aligned}$$

Finally, we plug this result back into the full product, to get

$$\begin{aligned} P(c \nmid a) &= \prod_{p \text{ prime}}^{\text{finite}} P(\neg(p|c \wedge p|a)) \geq \prod_{p \text{ prime}}^{\infty} \left(1 - \frac{1}{p^2}\right) \\ &= \zeta(2), \end{aligned}$$

where $\zeta(s)$ is the most famous function you never heard of, the *Riemann zeta function*, defined by

$$\zeta(s) = \prod_{p \text{ prime}} \left(1 - \frac{1}{p^s}\right)$$

in Euler product form. The value of $\zeta(2)$ has to be handed-off to the mathematical annals, and we'll simply quote the result,

$$\zeta(2) \approx .607.$$

so we have shown that

$$P(c \nmid a) \geq \zeta(2) \approx .607. \quad \text{QED}$$

That proves our second bullet, which is all we will need, but notice what it implies. Since

$$P(\neg c \nmid a) < .393,$$

after T random integer selections c_1, c_2, \dots, c_T in the interval $[0, a-1]$,

$$\begin{aligned} P\left(\left(\neg c_1 \nmid a\right) \wedge \left(\neg c_2 \nmid a\right) \wedge \dots \wedge \left(\neg c_T \nmid a\right)\right) \\ < .393^T, \quad \text{constant time, } T, \text{ independent of } N. \end{aligned}$$

This can be made arbitrarily small by choosing a large enough T , so we can expect to get a $c \nmid a$ with arbitrarily high probability after T random samples from the set $[0, a-1]$.

Combine Both Bullets to Complete Step V

To control the syntax, it'll help to introduce some notation.

$$\begin{aligned}\mathcal{C} &\equiv \{y_c\}_{c=0}^{a-1} \\ \mathcal{B} &\equiv \{y_b \mid y_b \in \mathcal{C} \text{ and } b \not\mid a\}\end{aligned}$$

(Note: $\mathcal{B} \subseteq \mathcal{C}$.)

$$\begin{aligned}|\mathcal{B}| &\equiv \text{the size (number of elements) in set } \mathcal{B} \\ |\mathcal{C}| &\equiv \text{(same)} = a \\ q &\equiv \text{the ratio } \frac{|\mathcal{B}|}{|\mathcal{C}|} \\ P(y_c) &\equiv P(\text{we measure a specific } y_c \in \mathcal{C})\end{aligned}$$

(Note: In the last definition, y_c may or may not also be $\in \mathcal{B}$.)

$$\begin{aligned}P(\mathcal{C}) &\equiv P(\text{we measure some } y \in \mathcal{C}) \\ P(\mathcal{B}) &\equiv P(\text{we measure some } y \in \mathcal{B}) \\ P(\mathcal{B}|\mathcal{C}) &\equiv P(\text{we measure some } y \in \mathcal{B} \\ &\quad \text{given that} \\ &\quad \text{the measured } y \text{ is known to be } \in \mathcal{C})\end{aligned}$$

We would like a lower bound on the probability of measuring a y_c which also has the property that its associated c is coprime to a . In symbols, we would like to show:

Claim: $P(\mathcal{B}) \geq$ some constant independent of a, M, N , etc.

Proof:

$$\begin{aligned}P(\mathcal{B}) &= P(\mathcal{B}|\mathcal{C}) P(\mathcal{C}) \\ &= \left(\sum_{b \in \mathcal{B}} P(y_b) \right) / \left(\sum_{c \in \mathcal{C}} P(y_c) \right) P(\mathcal{C})\end{aligned}$$

Let

$$\begin{aligned}y_{B\min} &\equiv y \in \mathcal{B} \text{ with } P(y_{B\min}) \text{ minimum over } \mathcal{B} \\ y_{B\max} &\equiv y \in \mathcal{B} \text{ with } P(y_{B\max}) \text{ maximum over } \mathcal{B} \\ y_{C\min}, y_{C\max} &\text{ same, except over all of } \mathcal{C}\end{aligned}$$

If there is more than one y that produce minimum or maximum probabilities, choose

any one. Then,

$$\begin{aligned} \left(\sum_{b \in \mathcal{B}} P(y_b) \right) / \left(\sum_{c \in \mathcal{C}} P(y_c) \right) &\geq \frac{|\mathcal{B}| P(y_{B\min})}{|\mathcal{C}| P(y_{C\max})} \\ &\geq \frac{|\mathcal{B}| P(y_{C\min})}{|\mathcal{C}| P(y_{C\max})} \\ &\geq q \times .072, \end{aligned}$$

so

$$P(\mathcal{B}) \geq (q \times .072) P(\mathcal{C}).$$

From the proof of the second bullet of this step, $q > .607$, and from **Step II** $P(\mathcal{C}) > .04503$, so

$$P(\mathcal{B}) \geq .607 \times .072 \times .04503 \approx .002.$$

This is independent of a, M, N , etc. and allows us to apply the *CTC theorem for looping algorithms* to aver that, after a fixed number, T , of applications of the quantum circuit, we will produce a y_c with $c \not\vdash a$ with any desired error tolerance. We'll compute T in a moment.

Remember that we've been using a worst case bounds (only two a intervals fit in $[0, M)$). As we demonstrated, normally the ratio we are getting, $.072$, is actually much larger: very close to 1. Also, typically $P(\mathcal{C}) > .40528$, so we can expect a better constant lower bound:

$$P(\mathcal{B}) \geq .607 \times 1 \times .40528 \approx .266.$$

Conclusion of Step V

After an adequate number of measurements (independent of a, M), which produce $y_{c_1}, y_{c_2}, \dots, y_{c_T}$, we can expect at least one of the $y_{c_k} = y_c$ to correspond to c/a , with $c \not\vdash a$ with high probability.

Examples that Use Different Assumptions about a

How many passes, T , do we need to get an error tolerance of, say $\varepsilon = 10^{-6}$ (one in a million)? It depends on the number of times our period, a , fits into the interval $[0, N)$. Under the worst case assumption that we formally required – only two – we would need a much larger number of whacks as the circuit than in a typical problem that fits hundreds of periods in the interval. Let's see the difference.

The “ p of success” bounded away from 0 for a single pass of our algorithm's loop in the *CTC theorem*, along with the error tolerance ε , gives us the number of required passes, the formula provided by the theorem,

$$T = \left\lceil \frac{\log(\varepsilon)}{\log(1-p)} \right\rceil + 1.$$

Worst Case ($P(\mathcal{B}) \approx .002$): We solved this near the end of the probability lesson and we found

$$T = \left\lceil \frac{\log(10^{-6})}{\log(.998)} \right\rceil + 1 = 6901,$$

or, more briefly and conservatively, 7000 loop passes.

Typical Case ($P(\mathcal{B}) \approx .266$): This was also an example in the earlier chapter,

$$T = \left\lceil \frac{\log(10^{-6})}{\log(.734)} \right\rceil + 1 = 45,$$

or, rounding up, 50 loop passes.

It's important to remember that **the data's actual probability doesn't care about *us***. We can instruct our algorithm to cycle 7000 times, but if the data determined that only 15 loop passes were needed to find the desired c/a , then it would return with a successful c/a after 15 passes. Our hyper-conservative estimate costs us nothing.

On the other hand, if we're worried that $a < N/2$ is too risky, and want to allow for only *one* period of a fitting into M or N , the math *still* works. For example, we could require merely $a < .999N$ and still get constant time bounds for all probabilities. Just repeat the analysis replacing $1/2$ with $.999$ to find the more conservative bounds. You would still find that proofs all worked, albeit with $P(\mathcal{B}) >$ a constant much smaller than $.002$.

23.10.11 Algorithm and Complexity Analysis (General Case)

Why Finding $y_c \in \mathcal{B}$ Solves Shor's Problem

In the final step of the previous section we proved that we will measure $y_c \in \mathcal{B}$ with near certainty after some predetermined number, T of measurements. Why does such a y_c solve Shor's period-finding problem? CFA returns $n/d = c/a$ close to y_c/N , but the actual numerators and denominators do not have to match: n/d is reduced, but c/a may not be for general c . However, we are safe when $y_c \in \mathcal{B}$, for then, $c \nmid a$, and we are assured that c/a is a reduced fraction. In that case, we *can* assert that the numerators and denominators match; for the denominators, that means $d = a$ and we are done.

This happens if we measure $y \in \mathcal{B}$, and this is why finding such a y will give us our period and therefore solve Shor's problem. (Wait. Might we end up getting $d = a$ even if we happened to measure some $y \notin \mathcal{B}$? In other words, is the *if* in the last sentence *if and only if* or just *if*? Well, there are two ways to deal with this question. You could chase down the logic to see whether a $y \notin \mathcal{B}$ used to send $x = y/N$ to the CFA could possibly give us $d = a$, but why bother? I prefer the second method to dispatch the question: Our algorithm will test $d = a$ after every measurement, so if (possible or not) we found a for the wrong reason, we still found a .)

Shor's Period-Finding Algorithm, General Case

After some hard work we have assembled all the results. Two things need to go right to be certain we measured a $y \in \mathcal{B}$:

1. We measure $y \in \mathcal{C} = \{y_c\}$.
2. the associated c satisfies $c \nmid a$.

We could fail in either or both, but the test of overall success is that the fraction, n/d , returned, by $\text{CFA}(y_c/N, 1/(2M^2))$ has the property that $d = a$. And we can test *that* in a single evaluation of our given function f . (Remember f ? It was given to us in Shor's hypothesis, and we used it to design/control U_f).

Testing whether, say, $f(1+d) = f(1)$ is enough to know whether d is the period, a .

The complexity of this test may or may not be polynomial time, depending on f . For RSA encryption-breaking the f in question is $O(\log^4(M))$, as we shall show, so for that application we will have absolute speed-up over classical solutions. In general, this step confines our quantum period-finding to only *relativized* speed-up.

So, the short description of the algorithm is this:

- Run the circuit producing y_c s, use those to get (n/d) s, stop when we confirm $f(1+d) = f(1)$, report that $d = a$ (our period) and declare victory.
- In the unlikely event that we run out of time (exceed our established T), we admit defeat.

The full version follows.

- Select an integer T that reflects an acceptable failure rate based on any known aspects of the period. (E.g., for a failure tolerance of .000001, we might choose $T = 7000$ if we expect only 2 periods, a , to fit into $[0, M - 1)$ or $T = 45$ if we know $a \ll M$. If we are happy with failure at .001, then these values would be adjusted downward.)
- Repeat the following loop at most T times.
 1. Apply Shor's circuit.
 2. Measure the output of \mathcal{QFT} to get a y .
 3. Apply CFA to y/N and $1/(2M^2)$ to produce n/d .
 4. Test d : If $f(1+d) = f(1)$ then $d = a$, (success); break from the loop.
 5. Otherwise continue to the next pass of the loop.
- If the above loop ended naturally (i.e., not from the *break*) after T full passes, we failed. Otherwise, we found a .

Computational Complexity (General Case)

(This is nearly word-for-word identical to the *easy case*.)

We have already seen that $O(\log(N)) = O(\log(M))$ as well as any powers of the logs, so I will use M in the following.

- The Hadamard gates are $O(\log(M))$.
- The \mathcal{QFT} is $O(\log^2(M))$.
- Complexity of U_f is same as that of f , which could be anything. We don't count that here, but we will show in a separate lecture that for integer factoring, it is certainly at least $O(\log^4 M)$.
- The outer loop is $O(T) = O(1)$, since T is constant.
- The classical CFA sub-algorithm is $O(\log^3(M))$.
- The four non-oracle components, above, are done in series, not in nested loops, so the overall *relativized* complexity will be the worst among them, $O(\log^3(M))$.
- In the case of factoring needed for RSA encryption breaking (order-finding) the actual oracle is $O(\log^4(M))$ or better, yielding an absolute complexity of $O(\log^4(M))$

So the entire Shor circuit for an $f \in O(\log^4(M))$ (true for RSA/order-finding) would have an absolute complexity of $O(\log^4(M))$.

23.10.12 Epilogue on Shor's Period-Finding Algorithm

Other than a discussion of Euclid's algorithm for computing the greatest common divisor and some facts about continued fractions (both covered in this volume) you have completed a rather in-depth development of Shor's periodicity algorithm for quantum computers.

You studied the circuit, the algorithm and computational complexity for quantum period-finding.

There remains the question of how quantum period-finding can be applied to RSA encryption-breaking, which is a form of "order-finding."

Before closing out this first course, let's cover those loose ends.

Chapter 24

Euclidean Algorithm and Continued Fractions

24.1 Ancient Algorithms for Quantum Computing

The *Euclidean algorithm* is a method for computing the *greatest common divisor* of two integers. The technique was described (although probably not invented) by the Greek mathematician Euclid in about 300 B.C. Two thousand years later it continues to find application in many computational tasks, one of which is Shor's quantum period finding algorithm. In fact, we apply it twice in this context, once directly, then again indirectly when we apply a second, centuries old technique called *continued fractions*.

In this lesson, we will study both of these tools and compute their time complexities.

24.2 The Euclidean Algorithm

24.2.1 Greatest Common Divisor

The *Euclidean Algorithm* (EA) takes two positive integers, P and Q , and returns the largest integer that divides both P and Q evenly (i.e., without remainder). Its output is called *the greatest common divisor of P and Q* and is written as a function of its two input integers, $\gcd(P, Q)$. We'll be particularly interested in the fact that

- $\text{EA}(P, Q) \in O(\log^3 X)$, where X is the larger of the two integers passed to it (although sharper/subtler bounds exist), and
- EA will be used as a basis for our *Continued Fractions Algorithm*, CFA.

24.2.2 Long Division

A long division algorithm (LDA) for integers A and B , both > 0 , produces $A \div B$ in the form of *quotient*, q and *remainder*, r satisfying

$$A = qB + r.$$

The *big-O* of $\text{LDA}(A, B)$ in its simplest form is $O(\log^2 X)$, where X is the larger of $\{A, B\}$. If you research this, you'll find it given as $O(N^2)$, where N is the *number of digits* in the larger of $\{A, B\}$, but that makes $N = \log_{10} X$ (and \log_{10} has the same complexity as \log_2 , which for us is designated just “log”).

24.2.3 The Euclidean Algorithm, $\text{EA}(P, Q)$

Although the final result is independent of which number is sent to first parameter, we always use letter P in that position; this will simplify the analysis.

I'll describe the algorithm without proof. If you're interested, you can check online for the many easy-to-follow demonstrations.

General Idea

To produce $\text{EA}(P, Q)$, for $P, Q > 0$ (initially we don't care which is larger), we start by applying long division to P, Q . $\text{LDA}(P, Q)$ returns a quotient, q , and remainder, r , satisfying

$$P = qQ + r.$$

Notice that either $r = 0$, in which case $Q|P$ and we are done ($\text{gcd}(P, Q) = Q$), or else we call LDA again, this time passing in the two integers Q and r . (From this point on, we know that $Q > r$, as will be true of all the qs and rs to follow). The second application of LDA gives us $Q \div r$ as well as the new remainder (call it r'),

$$Q = q'r + r'.$$

Like before, if $r' = 0$ we are done ($\text{gcd}(P, Q) = r$), and if not, we keep going. This continues until we get a remainder, $\tilde{r} = 0$, at which point the gcd is the integer “standing next to” (being multiplied by) its corresponding quotient, \tilde{q} , in the most recent long division. For example, if our first try produced $\tilde{r} = r = 0$, we would have returned Q “standing next to” q . If it happened in the second try, i.e., $\tilde{r} = r' = 0$, we would have returned r “standing next to” q' .

Let's add some indexing to our “general idea” before we give the complete algorithm. Define

$$\begin{aligned} r_0 &\equiv P, \\ r_1 &\equiv Q, \end{aligned}$$

and

$$\begin{aligned} q_0 &\equiv \text{the first quotient of } P \div Q, \\ r_2 &\equiv \text{the first remainder of } P \div Q. \end{aligned}$$

Using this notation, the initial division becomes

$$r_0 = \mathbf{q_0} \cdot r_1 + \mathbf{r_2}.$$

If $r_2 = 0$, return $\text{gcd} = q_0$, otherwise, $r_1 > r_2 > 0$, and we go on to compute $r_1 \div r_2$,

$$r_1 = \mathbf{q_1} \cdot r_2 + \mathbf{r_3}.$$

Continue until $r_k = 0$, at which point return $\text{gcd} = r_{k-1}$.

24.2.4 The Algorithm

EA(P, Q):

- Initialize

$$\begin{aligned} r_0 &= P \\ r_1 &= Q \end{aligned}$$

- Loop over k , starting from $k = 0$

– Compute $r_k \div r_{k+1}$ using LDA(r_k, r_{k+1}) to compute q_k and r_{k+2}

$$r_k = \mathbf{q_k} \cdot r_{k+1} + \mathbf{r_{k+2}}.$$

– until $r_{k+2} = 0$

- Return $\text{gcd} = r_{k+1}$

Example

We use EA to compute $\gcd(285, 126)$

$$\begin{aligned}r_0 &= 285 \\r_1 &= 126 \\r_0 &= \mathbf{q_0} \cdot r_1 + \mathbf{r_2} \\285 &= \mathbf{2} \cdot 126 + \mathbf{33} \\r_2 &\neq 0, \text{ so compute } r_1 \div r_2 \\r_1 &= \mathbf{q_1} \cdot r_2 + \mathbf{r_3} \\126 &= \mathbf{3} \cdot 33 + \mathbf{27} \\r_2 &\neq 0, \text{ so compute } r_2 \div r_3 \\r_2 &= \mathbf{q_2} \cdot r_3 + \mathbf{r_4} \\33 &= \mathbf{1} \cdot 27 + \mathbf{6} \\r_3 &\neq 0, \text{ so compute } r_3 \div r_4 \\r_3 &= \mathbf{q_3} \cdot r_4 + \mathbf{r_5} \\27 &= \mathbf{4} \cdot 6 + \mathbf{3} \\r_4 &\neq 0, \text{ so compute } r_4 \div r_5 \\r_4 &= \mathbf{q_4} \cdot r_5 + \mathbf{r_6} \\6 &= \mathbf{2} \cdot 3 + \mathbf{0} \\r_5 &= 0, \text{ so return } \gcd = r_4 \\ \gcd &= 3\end{aligned}$$

24.2.5 Time Complexity of EA

This is easiest seen by looking at the very first division,

$$P = qQ + r.$$

Notice that

$$r < \frac{P}{2}.$$

To see this, break it into two cases.

- case i) $Q \leq P/2$. In this case, since $r < Q$, we have $r < P/2$.
- case ii) $Q > P/2$. In this case $q = 0$ and $r = P - Q < P - P/2 = P/2$.

This same logic, applied to the iterative division

$$r_k = \mathbf{q_k} \cdot r_{k+1} + \mathbf{r_{k+2}},$$

gives

$$r_{k+2} < \frac{r_k}{2}.$$

In other words, every two divisions, we have reduced r_k by half, forcing the even-indexed r s to become 0 in at most $2 \log P$ iterations. Therefore, the number of steps is $O(2 \log P) = O(\log P)$.

(Incidentally, a similar argument works for $Q = r_1$, spawning the odd-indexed r s. So, whichever is smaller, P or Q , gives a tighter bound on the number of steps. However, we don't need that level of subtlety.)

We have shown that EA's main loop has complexity $O(\log X)$, where X is the larger (or either) of P, Q .

Next, we note that each loop iteration calls the $\text{LDA}(r_k, r_{k+1})$ which is $O(\log^2 r_k)$, since $r_k > r_{k+1}$. For all k , $r_k < X$ (larger of P and Q), so $\text{LDA}(r_k, r_{k+1})$ is also in the more conservative class, $O(\log X)$.

Combining the last two observations, we conclude that the overall complexity of $\text{EA}(P, Q)$ is the product of complexities of loop, $O(\log X)$, and the LDA within the loop, $O(\log^2 X)$, X being the larger of P and Q . Symbolically,

$$\begin{aligned} \text{EA}(P, Q) &\in O(\log^3 X), \\ X &= \text{larger of } \{P, Q\}. \end{aligned}$$

24.3 Convergents and the Continued Fraction Algorithm

24.3.1 Continued Fractions

Although we won't deal with such dizzying entities directly, you need to see an example of at least one *continued fraction* so we can define the more useful derivative (*convergents*) and present the *CFA algorithm*. A *continued fraction* is a nested construct (possibly infinite) of sums and quotients of integers in the form

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4 + \ddots}}}}.$$

We will only consider $a_0 \geq 0$ and a_k strictly positive for all $k > 0$.

Notice that the sequence of integers, $\{a_0, a_1, a_2, \dots\}$, completely defines this object because of its special form.

I said that some continued fractions are finite. When that happens there is a final a_K alone in its denominator, terminating the continued fraction.

The headlines are:

- Any real number, x , including both rationals ($29/48, 7, 8765/1234$, etc.) and irrationals ($\pi, \sqrt{2}, e^{2\pi}$, etc.), can be expressed as a continued fraction. (We only care about positive x .)
- When x is rational the continued fraction is finite. When x is irrational the continued fraction is infinite (in which case it “pencils out” to the irrational x but takes an infinite amount of graphite to write).

A Rational Example

An example of a *rational* number expressed as a continued fraction is (check it yourself):

$$x = \frac{285}{126} = 2 + \frac{1}{3 + \frac{1}{1 + \frac{1}{4 + \frac{1}{2}}}}.$$

You might ask why we bother with a continued fraction of an (already) rational number, x . The special form of a continued fraction leads to important approximations to x that are crucial in our proof of the quantum period-finding algorithm.

An Irrational Example

A famously simple *irrational* continued fraction is

$$x = \sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \ddots}}}}.$$

I will not prove any the above claims or most of what comes below. The derivations are widely and clearly published in elementary number theory texts and short web-pages on continued fractions. But I will organize the key facts that are important to Shor’s Algorithm.

24.3.2 Computing the CFA a_k s Using the EA if x is Rational

Before we abandon the ugly nested construct above, let’s look at one easy way to get the a_k if x is rational. This will tell us something about the computational complexity of our upcoming *continued fractions algorithm* (CFA).

CFA Using the EA. *If*

$$x = \frac{P}{Q},$$

the $\{a_k\}$ in its continued fraction expansion are exactly the unique $\{q_k\}$ of the Euclidean Algorithm, $EA(P, Q)$, for finding the $\gcd(P, Q)$.

In other words, we already have an algorithm for expanding a rational number as a continued fraction. It's called the *Euclidean algorithm* and we already presented it. We just grab q_k from EA and we're done.

(That's why I made a distinction between the first parameter, P and the second, Q , and also why I labeled the individual q_k in the EA, which we didn't seem to need at the time.)

Time Complexity for Computing a Rational (Finite) Continued Fraction

Theorem. *The time complexity for computing all the a_k for a rational x is $O(\log^3 X)$, where X is the larger of $\{P, Q\}$.*

Proof: Since the $\{a_k\}$ of continued fractions are just the $\{q_k\}$ of the EA, and we have proved that the $EA \in O(\log^3 X)$, where X is the larger of P and Q , the conclusion follows. QED

24.3.3 A Computer Science Method for Computing the a_k s of Continued Fractions

An iterative algorithm is typically used when programming CFA. We use the notation

$$\lfloor x \rfloor \equiv \text{greatest integer } \leq x.$$

The CF Method

- Specify a termination condition (e.g., a maximum number of loop passes or a continued fraction within a specified ε of the target x , etc.).
- Loop over k starting at $k = 0$ and incrementing until we reach the maximum number of loop passes or we break sooner due to a “hit” detected in the loop body.

1. $a_k \leftarrow \lfloor x \rfloor$
2. $frac \leftarrow x - \lfloor x \rfloor$
3. If $frac = 0$, break from loop; we've found x .
4. $x \leftarrow 1/frac$

- Return the sequence $\{a_k\}$.

24.3.4 Convergents of a Continued Fraction

Algorithmic Definition of Convergents

Rather than using the unwieldy continued fractions directly, we'll work with the much more tractable “simple fractions” which derive from them.

The k th Convergent of x . *If we forcibly stop the loop in the **CF method** after k iterations, we produce (and return) a number which is only an approximation of x . This approximation is called the **k th convergent of x** .*

Of course if x is irrational, we can only *ever* get an approximation, so this is not big news. But if x is rational, and we halt the process prematurely, before $a_k = a_K$ (where I'm using a_K to mean the final denominator we would have gotten had we been patient enough to go all the way to x), then we only have an approximation of x . Surprisingly, these approximations are sometimes more useful than the whole enchilada. Certainly, for our purposes, they will turn out to be.

When we fully evaluate the k th convergent it will, of course, resolve to a fraction in its own right, and we always express *that* fraction in reduced form.

Algebraic Definition of Convergents

We could have defined the k th convergent of x when we first introduced continued fractions. They are just the partially completed constructs that we see when writing out (or attempting to write out) the full continued fraction. Explicitly, the first few convergents are

$$\begin{aligned}\frac{n_0}{d_0} &= a_0, \\ \frac{n_1}{d_1} &= a_0 + \frac{1}{a_1}, \\ \frac{n_2}{d_2} &= a_0 + \frac{1}{a_1 + \frac{1}{a_2}},\end{aligned}$$

and generally,

$$\frac{n_k}{d_k} = a_0 + \frac{1}{a_1 + \frac{1}{\ddots + \frac{1}{a_{k-1} + \frac{1}{a_k}}}}.$$

Example

For the rational $x = 285/126$, whose continued fraction and $\{a_k\}$ we computed earlier, you can verify that the convergents are

$$\begin{aligned}\frac{n_0}{d_0} &= \frac{2}{1}, \\ \frac{n_1}{d_1} &= \frac{7}{3}, \\ \frac{n_2}{d_2} &= \frac{9}{4}, \\ \frac{n_3}{d_3} &= \frac{43}{19}, \quad \text{and} \\ \frac{n_4}{d_4} &= \frac{95}{42}.\end{aligned}$$

Notice that the final convergent is our original x in reduced form. This is very important for our purposes. Figures 24.1 and 24.2 show two graphs of these convergents at different zoom levels.



Figure 24.1: A wide shot of the of the convergents starting at $2/1$

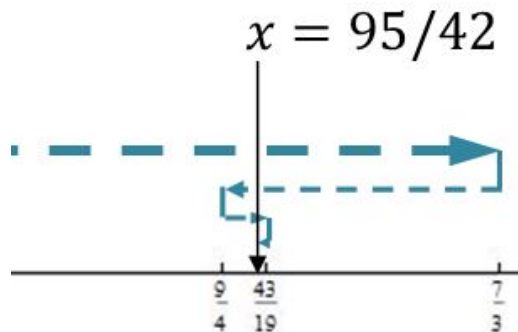


Figure 24.2: A close-up showing some of the latter convergents.

A couple features might pop:

1. The convergents “converge” very fast – every two convergents are much closer together than the previous two.
2. The convergents bounce back-and-forth around the target, x , alternating less-than- x values and greater-than- x values.

Before making this precise, let’s see if it holds for a second example.

Example

For the rational $x = 11490/16384$ the convergents are:

$$\begin{aligned}\frac{n_0}{d_0} &= \frac{0}{1}, \\ \frac{n_1}{d_1} &= \frac{1}{1}, \\ \frac{n_2}{d_2} &= \frac{2}{3}, \\ \frac{n_3}{d_3} &= \frac{5}{7}, \\ \frac{n_4}{d_4} &= \frac{7}{10}, \\ \frac{n_5}{d_5} &= \frac{54}{77}, \\ \frac{n_6}{d_6} &= \frac{1897}{2705}, \quad \text{and} \\ \frac{n_7}{d_7} &= \frac{5745}{8192} = \frac{11490}{16384}.\end{aligned}$$

This time, we’ll need more zoom levels because the later convergents are so close to x they are impossible to see. Figures 24.3 through 24.7 show various stages of the convergents.

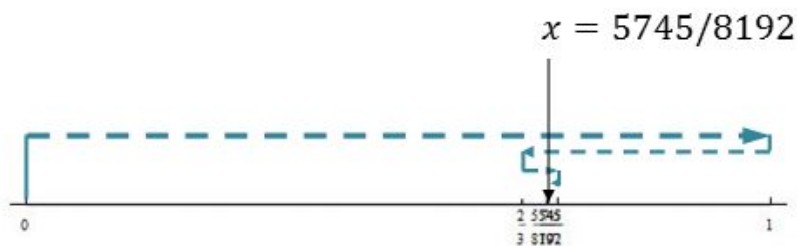


Figure 24.3: First view of convergents

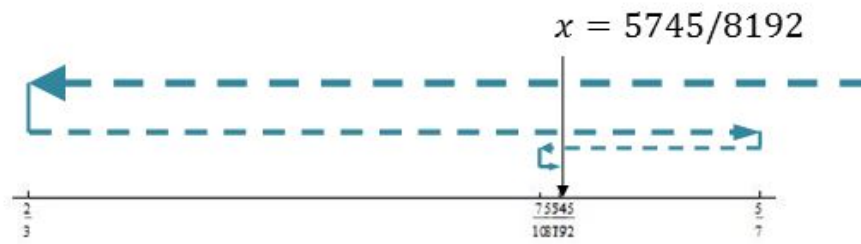


Figure 24.4: Second view of convergents

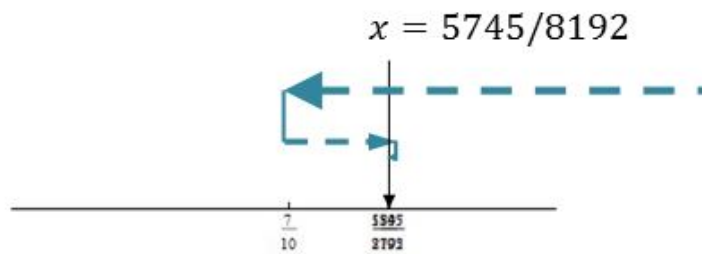


Figure 24.5: Third view of convergents

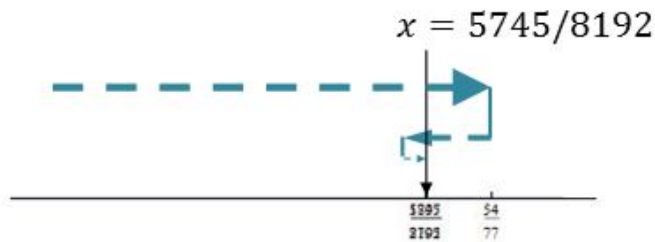


Figure 24.6: Fourth view of convergents

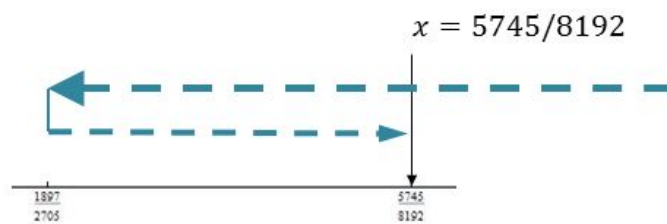


Figure 24.7: Fifth view of convergents

24.3.5 An Algorithm for Computing the Convergents $\{n_k/d_k\}$

- Specify a termination condition (e.g., a maximum number of loop passes or a convergent within a specified ε of the target x , etc.).
- Invoke the *CF method*, given earlier, that produces the entire sequence $\{a_k\}$ to the desired accuracy, ε , but let it complete – we don't stop after the maximum loop passes set in the first bullet. (Note: The *CF method* can be merged into this algorithm by combining loops carefully.)
- Set the first two convergents, manually:

$$\begin{aligned} n_0 &\leftarrow a_0, & d_0 &\leftarrow 1 \\ n_1 &\leftarrow a_1 a_0 + 1, & d_1 &\leftarrow a_1 \end{aligned}$$

- Loop over k starting at $k = 2$ and iterating until $k = K$, the final index of the sequence $\{a_k\}$ returned by the *CF method*.

$$\begin{aligned} 1. \quad n_k &\leftarrow a_k n_{k-1} + n_{k-2} \\ 2. \quad d_k &\leftarrow a_k d_{k-1} + d_{k-2} \end{aligned}$$

- Return the sequence $\{n_k/d_k\}$.

24.3.6 Easy Properties of the Convergents

The convergents $\{n_k/d_k\}$ have the following properties which are derived in most beginning tutorials in very few steps.

1. For any real number, x , the convergents approach x ,

$$\lim_{x \rightarrow \infty} \left(\frac{n_k}{d_k} \right) = x.$$

2. For rational x , the above limit is finite, i.e., there will be a $K < \infty$, with $n_K/d_K = x$ exactly, and no more fractions are produced for $k > K$.
3. They alternate above and below x ,

$$\frac{n_k}{d_k} = \begin{cases} > x, & \text{if } k \text{ is odd} \\ < x, & \text{if } k \text{ is even} \end{cases} \quad (\text{assuming } \{n_k/d_k\} \neq x).$$

4. Each n_k/d_k (if not the final convergent which is exactly x) differs from x by no more than $1/(d_k d_{k+1})$,

$$\left| x - \frac{n_k}{d_k} \right| \leq \frac{1}{d_k d_{k+1}}.$$

5. For $k > 0$, n_k/d_k is the best approximation to x of all fractions with denominator $\leq d_k$,

$$\left| x - \frac{n_k}{d_k} \right| \leq \left| x - \frac{n}{d} \right|, \text{ for all } d \leq d_k.$$

6. Consecutive convergents differ from *each other* by *exactly* $1/(d_k d_{k+1})$,

$$\left| \frac{n_{k-1}}{d_{k-1}} - \frac{n_k}{d_k} \right| = \frac{1}{d_k d_{k+1}}.$$

7. The denominators $\{d_k\}$ are strictly increasing and, if x is rational, are all \leq the denominator of x (whether or not x was given to us in reduced form).
8. When $x = P/Q$ is a rational number, computation of the all convergents is $\in O(\log^3 X)$, where X is the larger of $\{P, Q\}$. (This follows from the fact that the *convergent* algorithms above are based on EA as well as the details of the loops used.)

There is one not-so-easy-to-prove fact that we will need. It can be found in *An Introduction to the Theory of Numbers* by Hardy and Wright (Oxford U. Press) as Theorem 184. The proof is rather involved.

- If a fraction n/d differs from x by less than $1/(2d^2)$ then n/d will appear in the list of convergents for x . Symbolically:

If

$$\left| \frac{n}{d} - x \right| < \frac{1}{2d^2}$$

then

$$\frac{n}{d} = \frac{n_{k_0}}{d_{k_0}} \in \left\{ \frac{n_k}{d_k} \right\}_{k=0}^K \quad (\text{the convergents for } x) .$$

24.3.7 CFA: Our Special Brand of Continued Fraction Algorithm

Our version of a convergent-generating algorithm, which I call CFA, will take as input parameters a *rational* target x and requested degree of accuracy ε . $\text{CFA}(x, \varepsilon)$ will return n/d , the first convergent (i.e., that with the smallest index k) to x within ε of x . It simply wraps the previous algorithm into an envelope that returns a single fraction (rather than all the convergents).

1. To our previous algorithm for generating the convergents, pass x along with the terminating condition that it stop looping when it detects that $|x - n_k/d_k| \leq \varepsilon$.

2. Return n_k/d_k .

That's all there is to it.

Depending on ε and x , $\text{CFA}(x, \varepsilon)$ either returns $n/d = n_K/d_K = x$, exactly, as its final convergent or an ε -approximation $n/d \neq x$, but within ε of it.

Chapter 25

From Period-Finding to Factoring

25.1 Period Finding to Factoring to RSA Encryption

RSA encryption relies on an interesting computational and mathematical fact. Given large enough positive integer, $M = pq$, which is the product of two primes, it would take the world's fastest super computer billions of years to compute the two unknown factors p and q . How large is *large enough*? An M with around 28 decimal digits would work.

You'd have to take a course in classical cryptography – or find a tutorial on the Web – to learn why the inability to factor large numbers leads to encryption and how the RSA works. Our task today is to learn how a quantum computer can solve the same factoring problem for such a large number efficiently (potentially in minutes or seconds). We have, in our quill, a very powerful arrow: Shor's quantum algorithm for period-finding. Let's see how a hypothetical quantum computer could leverage it to overcome the computational limitation to which a classical super computer is subject.

25.2 The Problem and Two *Classically Easy* Cases

The Factoring Problem. *Given some $M \in \mathbb{Z}_{>0}$ that we know is not prime, find a $q \in \mathbb{Z}_{>0}$ that satisfies $q|M$.*

[**Variable Name** – Most number theory publications use the letter N for the large number to be factored, but I've been using it to be the power-of-2 larger than M^2 , where M is our bound on the period a (and the number we will want to factor). Therefore, I'll deviate from the usual practice and let M be the arbitrary integer we want to factor.]

To narrow down the kind of M that we're willing to look at, let's dispose of two cases that can be tested and factored by ordinary computers efficiently,

1. M even, and
2. $M = p^k$, $k > 1$, is a power of some prime.

Why the Two Cases Are “Easy”

The test “ M even?” entails a simple examination of the least significant bit (0 or 1), trivially fast. Meanwhile, there are easy classical methods that determine whether $M = p^k$ for some prime p and produce such a p in the process (thus providing a divisor of M).

In fact, we can dispose of a *larger* class of M : those for which $M = q^k$, $k > 1$ for *any* integer q (prime or not) less than M . We can even produce such a q in the process. And we’ll do it all, using classical machinery. If we detected that case, it would provide a factor q , do so without requiring Shor’s quantum circuit, and cover the more restrictive *condition 2*, in which q is a prime.

So why does the second condition only seek to eliminate the case in which M is a power of some *prime* p before embarking on our quantum algorithm rather than using classical methods to test and bypass the larger class of M that are powers of *any integer*, q ? First, eliminating only those M that are powers of a single prime is all that the quantum algorithm actually requires. So once we have disposed of that possibility, we are authorized to move on to Shor’s quantum algorithm. Second, knowing we can move on after confirming $M \neq p^k$, for a p prime, gives us options.

- We can ask the number theorists to provide a very fast answer to the question “is $M = p^k$, p prime,” and let the quantum algorithm scoop up the remaining cases (which include $M = q^k$, q not prime).
- Alternatively, we can apply a fast classical method to search for a q (prime or not) that satisfies $M = q^k$, thus avoiding the quantum algorithm in a larger class of M .

One of the above two paths may be faster than the other in any particular {hardware + software} implementation, so knowing that we can go either way gives us choices.

Now let’s outline why either of the two tests

$$M = q^k, \quad k > 1 \quad \text{or}$$

$$M = p^k, \quad k > 1, \quad p \text{ prime}$$

can be dispatched classically.

Classical Algorithm for Larger Class: $M = q^k$, $k > 1$

$$M = q^k, \quad k > 1 \quad \text{or}$$

$$M = p^k, \quad k > 1, \quad p \text{ prime}$$

Any such power k would have to satisfy

$$k = \log_q M \leq \log_2 M ,$$

because q has to be ≥ 2 . (In fact, we've eliminated M even, so the above \leq can be strengthened to $<$, but no matter.) Now, for each $k \leq \log_2 M$ we compute the integer

$$q = \left\lfloor \sqrt[k]{M} \right\rfloor$$

(for which fast algorithms exist), then test whether $q^k = M$. If it does, q is our divisor and we have covered the case $M = q^k$, $k > 1$, for *any* integer q without resorting to quantum computation. The time complexity is polynomial fast because

- the outer loop has only $\log_2 M$ passes (one for each k),
- even a slow, brute force method to compute $q = \left\lfloor \sqrt[k]{M} \right\rfloor$ has a polynomial *big-O*, and
- taking the power q^k is also polynomial fast. (Moreover, the implementation of previous bullet can be designed to absorb this computation, obviating it.)

[**Exercise.** Design an algorithm that implements these bullets and derive its *big-O*.]

Classical Algorithm for Smaller Class: $M = p^k$, $k > 1$, p Prime

But if one wanted to also know whether the produced q in the above process were prime, one could use an algorithm like AKS (Google it) which has been shown to be *log polynomial*, better than $O(\log^8(\# \text{digits in } M)) = O(\log^8(\log M))$, in M .

This approach was based on first finding a q with $M = q^k$, then going on to determine whether q was prime. That's not efficient, and I presented it only because the components are easy, off-the-shelf results that can be combined to prove the classical solution is polynomial fast. In practice we would seek a solution that tests whether M is a power of a prime *directly*, using an approach that would be faster than testing whether it is a power of a general integer, q .

Why We Eliminate the Two Cases

The reason we quickly dispose of these two cases is that the reduction of factoring to period-finding, described next, will not work for either one. However, we now understand why we can be comfortable assuming M is neither even nor a power of a single prime and can proceed based on that supposition.

25.3 A Sufficient Condition

The next step is to change the the factoring problem into a proposition with which we can work more easily.

Claim. *We will obtain a $q|M$ if we can find an x with the property that*

$$\begin{aligned} x^2 &= 1 \pmod{M}, \text{ for} \\ x &\not\equiv \pm 1 \pmod{M}. \end{aligned}$$

Proof

$$\begin{aligned} x^2 &= 1 \pmod{M} \\ &\Rightarrow \\ x^2 - 1 &= 0 \pmod{M} \\ &\Leftrightarrow \\ M &\mid (x^2 - 1) \\ &\Leftrightarrow \\ M &\mid (x - 1)(x + 1). \end{aligned}$$

That can only happen if M has a factor, $p > 1$, in common with one or both of $(x - 1)$ and $(x + 1)$, i.e.,

$$\begin{aligned} p|M \text{ and } p|(x - 1) \text{ or} \\ p|M \text{ and } p|(x + 1). \end{aligned}$$

That factor, p , *cannot* be M , itself, for if it did, either

$$\begin{aligned} M|(x - 1), \quad \text{contrary to } x &\not\equiv +1 \pmod{M} \text{ or} \\ M|(x + 1), \quad \text{contrary to } x &\not\equiv -1 \pmod{M}, \end{aligned}$$

both, outlawed by the hypothesis of the “claim”. Define

$$q \equiv \begin{cases} \gcd(M, x - 1), & \text{if common factor } p|(x - 1) \text{ or} \\ \gcd(M, x + 1), & \text{if common factor } p|(x + 1). \end{cases}$$

Whichever of the above two cases is true (and we just proved at least one must be), we have produced a q with $q|M$. **QED**

The time complexity of $\gcd(M, k)$, $M \geq k$ is shown in another lecture to be $O(\log^3 M)$, so once we have x , getting q is “fast.”

Before we find x , we take a short diversion to describe something called *order – finding*.

25.4 A Third Easy Case and Order-Finding in \mathbb{Z}_M

Pick a y at random from $\mathbb{Z}_M - \{0, 1\} = \{2, 3, 4, \dots, M-1\}$. Either y will be coprime to M ($y \nmid M$) or it won't.

- If $\neg(y \nmid M)$ we're done, because $q \equiv \gcd(M, y)$ will be our desired factor of M (and we don't even need to look for an x). An $O(M^3)$ application of GCD will reveal this fact by either returning a $q > 1$ (we're done) or result in 1 (we continue).
- If $y \nmid M$ we go on to find the "order" of y , defined next and which leads to the rest of the algorithm.

The New "Easy Case"

Because of the first bullet, we have a third easy condition that leads to a fast classical solution to factoring: We pick a y between 2 and $M-1$ at random, apply the GCD algorithm, and if it returns a value > 1 , we've got our factor, q , of M .

Defining Order

We therefore assume that the randomly selected y satisfies $y \nmid M$. While it may not be obvious, finding the "order" of y in \mathbb{Z}_M will be the key. In this section we define "order" and learn how we compute it with the help of Shor's quantum period-finding; the final section will explain how doing so factors M .

Order. When $y \nmid M$, we define the **order** of $y \in \mathbb{Z}_M$, also called the **order of $y \pmod{M}$** , to be the smallest positive integer, $b > 1$ such that

$$y^b \equiv 1 \pmod{M}.$$

Why are we so sure such an $b > 1$ exists? Consider all powers, y^k , $k = 1, 2, 3, \dots$. Since \mathbb{Z}_M is finite, so is

$$\{y^k \pmod{M}\}_{k=1}^{\infty} \quad (\text{being a } \subseteq \mathbb{Z}_M),$$

and there must be many (infinitely many) distinct pairs k, k' with

$$y^k \equiv y^{k'} \pmod{M}.$$

For each pair, take k' to be the larger of the two, and write this last equality as

$$y^k \equiv y^{k+b} \pmod{M}, \quad b \equiv k' - k > 0.$$

We just argued that there are infinitely many $k > 1$ (with potentially a different $b > 0$ for each k) for which the above holds. There must be a *smallest* b that satisfies this

among all the pairs. (Once you find *one* pair, take the k and b for that pair. Keep looking for other pairs with different k s and smaller b s. You can't do this indefinitely, since eventually you'd reach $b = 0$, which we know isn't possible because all such pairs were chosen so that $k \neq k'$. It doesn't matter how long this takes – we only need the *existence* of such a b , not to produce it, physically.) Assume this last equality represents that smallest $b > 0$ for any k which makes it true. That means, there exists a pair with

$$y^k - y^{k+b} = 0 \pmod{M}, \quad b \text{ minimal over all } k.$$

Factoring,

$$\begin{aligned} y^k (1 - y^b) &= 0 \pmod{M} \\ \Rightarrow \\ M &\mid y^k (1 - y^b) \\ \Rightarrow \\ M &\mid (1 - y^b). \end{aligned}$$

The last step holds because we are in the case (also required by the definition of “Order”) $y \not\equiv 1 \pmod{M}$. We're done (proving that an order, b , of y exists) because the final equality means

$$y^b \equiv 1 \pmod{M}.$$

From Orders to Periods

If we set

$$a \equiv b + 1$$

and substitute $a - 1$ for b in the last result, we find

$$y^a \equiv y \pmod{M}, \quad a \text{ the smallest int with this property.}$$

That means the function

$$f(x) = y^x \pmod{M},$$

built upon our randomly selected y , is *periodic with period a* . Furthermore, it is \mathbb{Z}_M -periodic, which implies the extra condition

$$f(x') \neq f(x) \quad \text{whenever} \quad x - x' < a.$$

(Review the definition of b and its minimality to verify this extra condition.)

It might be intuitively obvious that $a < M$ (a condition that we need for Shor), but we should still give a solid argument for it. One of the more common is an observation from elementary number theory. I have to define a famous function $\phi(M) \equiv \#$ of elements $< M$ that are coprime to M , i.e.,

Euler ϕ Function. For any positive integer M , define $\phi(M)$ to be

$$\phi(M) \equiv \left| \{ y \mid y \not\equiv a, \quad \text{for } 1 \leq y \leq M-1 \} \right|.$$

The basic result (I'll let you look it up – it's not too difficult to follow) is that the order, b of any y (necessarily coprime to M) in \mathbb{Z}_M , divides evenly, into $\phi(M)$, i.e.,

$$b \mid \phi(M).$$

Since $\phi(M) < M$, this implies $b < M$. Therefore, $a \leq M$.

Enter Quantum Computing

After first disposing of a few easy cases, we are left with (i) M , an odd integer which we wish to factor that (ii) is not an exact power of some smaller integer, $q > 1$ and (iii) through random selection have produced a $y \not\equiv M$ that gives us a \mathbb{Z}_M -periodic function, $f(x) = y^x \pmod{M}$, with unknown period, $a < M$. Where do we go from here?

Phrase (iii) is the hypothesis of Shor's quantum algorithm which we have already proved can be applied in $\log^3 M$ time. This is exactly where we would use our quantum computer in the course of factoring M . So we proceed to apply our circuit and algorithm to get the period a , which also gives us y 's order, $b = a - 1$. The next, and final, step is to demonstrate how we use the period, a , of f and/or the order, b , of y to factor M .

25.5 Using the Order of y to Find the x of our “Sufficient Condition”

We've already established that if we can find an x that satisfies

$$\begin{aligned} x^2 &= 1 \pmod{M}, \\ x &\not\equiv \pm 1 \pmod{M}, \end{aligned}$$

and do so efficiently (in polynomial time), we will get our factor q of M . We did manage to leverage quantum period-finding to efficiently get the period of a randomly selected $y \in \mathbb{Z}_M$, so our job is to use that order, a , without using excessive further computation, to factor M . First, we work with the order of $y \in \mathbb{Z}_M$, namely, $b = a - 1$. There are the three cases to consider:

1. b is even, and $y^{b/2} \not\equiv -1 \pmod{M}$.
2. b is even, and $y^{b/2} \equiv -1 \pmod{M}$.
3. b is odd.

Case 1: Even b , with $y^{b/2} \not\equiv -1 \pmod{M}$

Claim.

$$x \equiv y^{b/2}$$

satisfies our sufficient condition.

Proof

$$\begin{aligned} x^2 &= y^b, \quad \text{where } b = \text{order of } y \pmod{M}, \text{ so} \\ x^2 &\equiv 1 \pmod{M}. \end{aligned}$$

That's half the sufficient condition. The other half is $x \not\equiv \pm 1 \pmod{M}$. Our assumption in this case is that

$$x \not\equiv -1 \pmod{M},$$

so we only need to show that

$$x \not\equiv +1 \pmod{M},$$

and we'll have shown that this x satisfies our sufficient condition. Proceed by contradiction. What would happen if

$$x \equiv +1 \pmod{M}?$$

Then

$$\begin{aligned} y^{b/2} &\equiv +1 \pmod{M}, \quad \text{i.e.,} \\ y^{(b/2)+1} &\equiv y \pmod{M}. \end{aligned}$$

If $b = 2$, then

$$\begin{aligned} y^2 &\equiv y \pmod{M}, \quad \text{i.e.,} \\ y &\equiv 1 \pmod{M}, \end{aligned}$$

which contradicts that $y \in \mathbb{Z}_M - \{0, 1\}$ (1 is not in that set). So we are forced to conclude that $b > 2$, which gives

$$\begin{aligned} \frac{b}{2} &> 1, \text{ so} \\ \frac{b}{2} + 1 &< \frac{b}{2} + \frac{b}{2} = b. \end{aligned}$$

But now we have an $b' = (b/2) + 1 < b$ that satisfies

$$y^{b'} = y^{(b/2)+1} \equiv y \pmod{M}.$$

That contradicts that b is the order of $y \pmod{M}$, because the *order* is the *smallest* integer with that property, by construction. QED

That dispatches the first case; we have found an x which satisfies the sufficient condition needed to find a factor, q , of M .

Cases 2 and 3: Even b , with $y^{b/2} = -1 \pmod{M}$ or Odd b

I combine these two cases because, while they are both possible, we rely on results from number theory (one being the Chinese Remainder Theorem) which tell us that the probability of both cases, taken together is never more than .5, i.e.,

$$P(\text{case 2} \vee \text{case 3}) \leq \frac{1}{2}.$$

This result is independent of M . That means that if we repeatedly pick y at random, T times, the chances that we are unlucky enough to get case 2 or case 3 in *all* T trials is

$$P\left(\bigwedge_{k=1}^T \text{case 2} \vee \text{case 3}\right) \leq \prod_{k=1}^T \frac{1}{2} = \frac{1}{2^T}.$$

Therefore, we end up in case 1 at least once in T trials with probability $1 - 1/(2^T)$, independent of M , i.e., in constant time. So we efficiently (constant time) get into case 1, and once we are, we use Shor's period-finding algorithm on a quantum computer, which is $O(\log^3(M))$, to compute the period and produce x . After we have x , another $O(\log^3(M))$ algorithm, GCD (on an ordinary computer), gets us M .

Note. This constant time T is different from the constant time T we needed inside Shor's algorithm. If we wanted a specific ε for our algorithm, we'd want to make sure both T s are defined using their respective and distinct, formulas involving the same ε .

25.6 The Time Complexity of $f(x) = y^x \pmod{M}$

There is one final detail we have not addressed. Shor's quantum period-finding was only as fast as its weakest link. Since the algorithm is $\log^3 M$ only counting the logic *exterior* to the quantum oracle, U_f , then any f which has a larger growth rate would erode Shor's performance accordingly. In other words, it has *relativized* exponential speed up. If it is to have *absolute* speed-up over the classical case we must show that the oracle, itself, is polynomial time in M . I'm happy to inform you that in the case of the factoring problem, the function $f(x) = y^x \pmod{M}$ is, indeed, $\log^3 M$.

25.7 The Complexity Analysis

$y \in \mathbb{Z}_M$ and we also saw that $a < M$, so we only need to consider computing y^x for both $y, x < M$. Since $M < M^2 \leq N = 2^n$, we can express both x and y as a sum of powers-of-2 with, at most, $n = \log N$ terms. Let's do that for x :

$$x = \sum_{k=0}^{n-1} x_k 2^k,$$

where the x_k are x 's base-2 digits. So (all products taken mod- M)

$$y^x = y^{\sum_k x_k 2^k} = \prod_{k=0}^{n-1} y^{x_k 2^k}.$$

However long it takes to compute the general factor, $y^{x_k 2^k}$, we need to repeat it n times, and when those n factors are computed, multiply them together, burning $(n-1)$ multiplications. There are two parts, taken in series, to this computation:

1. $n \times [\text{complexity of } y^{x_k 2^k} \pmod{M}]$
2. $(n-1) \times [\text{complexity of a multiplication } \pmod{M}]$

The slower (not the product) of the two will determine the overall complexity of $y^x \pmod{M}$.

Preliminary Note. The computational complexity of integer multiplication is $O(\log^2 X)$, where X is the larger of the two numbers. For us, each product is bounded by $N > M$, so integer multiplication costs us, at most, $O(\log^2 N)$.

25.7.1 Complexity of Step 1

$$y^{x_k 2^k} = \left(y^{2^k}\right)^{x_k},$$

and in the process of computing y^{2^k} we would end up computing $y^{2^{k-1}}$, so, in order to avoid repeated calculations from factor-to-factor, we first compute an array of the n factors

$$\left\{y^{2^k}\right\}_{k=0}^{n-1} = \{1, y, y^2, y^4, y^8, \dots, y^{n-1}\}, \quad \text{all mod-}M.$$

Starting with the second element, y , each element in this array is the square of the one before. That's a total of $n-2$ multiplications (we get 1 and y for free). Thus, to produce the entire array it costs $O(\log N)$ multiplications, with each multiplication (by above note) $\in O(\log^2 N)$. That's a total complexity of $O(\log^3 N)$. This is done once for each factor in the product

$$\prod_{k=0}^{n-1} \left(y^{2^k}\right)^{x_k}.$$

To complete the computation k factors, we raise each of the pre-computed y^{2^k} (in our array) to the x_k power. That's x_k multiplications for each factor. Wait a minute – x_k is a binary digit, either 0 or 1, so this is nothing other than a *choice*; for each n we tag on an *if-statement* to finish off the computation for that factor. Therefore, the computation of each factor remains $O(\log^3 N)$.

There are n factors to compute, so this tags on another $\log N$ magnitude to the bunch, producing a final cost for step 1 of $O(\log^4 N)$. However, we haven't tackled the big Π product yet

25.7.2 Complexity of Step 2

Each binary product in the big Π is $O(\log^2 N)$. The big Π has $n - 1$ such products, so the final product (after computing the factors) is $O((n - 1) \log^2 N) = O(\log^3 N)$.

Combining Both Results

The evaluation of all n factors ($O(\log^4 N)$) is done *in series* with (not nested within) the final product ($O(\log^3 N)$), so the slower of the two, $O(\log^4 N)$, determines the full complexity of the oracle. Note that this was a lazy and coarse computation, utilizing simple multiplication algorithms and a straightforward build of the $f(x) = y^x \pmod{M}$, function, and one can certainly do a little better.

As we demonstrated when covering Shor's algorithm, the relationship between M and N ($N/2 < M^2 \leq N$) implies that this is the equal to $O(\log^4 M)$

25.7.3 Absolute Complexity of Shor's Factoring

(This is also stated in the lecture on Shor's quantum period-finding.)

We have now shown the the oracle can be built with an efficiency that is at least $O(\log^4 M)$. Shor's period finding had a *relativized complexity* of $O(\log^3 M)$, due to a *constant time* circuit sampling that contained four computational components in series,

- $H^{\otimes n} : O(\log M)$,
- the oracle: $O(\log^4 M)$,
- $QFT : O(\log^2 M)$, and
- classical post-measurement processing (EA/CF): $O(\log^3 M)$.

The bottleneck is the oracle, at $O(\log^4 M)$, which we will use as our polynomial time complexity, proving the *absolute speed-up* of quantum factoring. We acknowledge the existence of faster oracles than the construction provided above, improving overall algorithm, accordingly.

List of Figures

1	After measuring one location, we know them all	23
2	After measuring one location, we don't know much	23
1.1	a few numbers plotted in the complex plane	33
1.2	visualization of complex addition	35
1.3	the connection between cartesian and polar coordinates of a complex number	37
1.4	conjugation as reflection	38
1.5	modulus of a complex number	39
1.6	$e^{i\theta}$ after θ seconds, @ 1 rad/sec	41
1.7	multiplication – moduli multiply and args add	44
1.8	the fifth roots-of-unity	47
1.9	Three of the fourth roots-of-unity (find the fourth)	48
2.1	A vector in \mathbb{R}^2	54
2.2	Vector addition in \mathbb{R}^2	55
2.3	Scalar multiplication in \mathbb{R}^2	56
2.4	Orthogonal vectors	57
2.5	A vector in \mathbb{R}^3	59
2.6	A vector expressed as linear combination of $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$	62
2.7	A vector expressed as linear combination of \mathbf{b}_0 and \mathbf{b}_1	65
2.8	A vector expressed as linear combination of \mathbf{c}_0 and \mathbf{c}_1	66
3.1	Dot-product of the first row and first column yields element 1-1	76
3.2	Dot-product of the second row and second column yields element 2-2 . .	76
3.3	Minor of a matrix element	81
3.4	The numerator of Cramer's fraction	87
4.1	The Cauchy sequence $\{1 - \frac{1}{k}\}_{k=2}^{\infty}$ has its limit in $[0, 1]$	101

4.2	The Cauchy sequence $\{1 - \frac{1}{k}\}_{k=2}^{\infty}$ does not have its limit in $(0, 1)$. . .	101
4.3	Triangle inequality in a metric space.svg from Wikipedia	103
4.4	A “3-D” quantum state is a ray in its underlying $\mathcal{H} = \mathbb{C}^3$	104
4.5	Dividing a vector by its norm yields a unit vector on the same ray . . .	105
5.1	T mapping a vector \mathbf{v} in \mathbb{R}^3 to a \mathbf{w} in \mathbb{R}^3	111
5.2	A scaling transformation	112
5.3	Projection onto the direction $\hat{\mathbf{z}}$, a.k.a. $\hat{\mathbf{x}}_3$	113
5.4	Projection onto an arbitrary direction $\hat{\mathbf{n}}$	113
5.5	Rotation of $\hat{\mathbf{x}}$ counter-clockwise by $\pi/2$	115
5.6	Rotation of $\hat{\mathbf{y}}$ counter-clockwise by $\pi/2$	116
6.1	Classical angular momentum	138
6.2	A classical idea for spin: A 3-D direction and a scalar magnitude . . .	139
6.3	Polar and azimuthal angles for the (unit) spin direction	139
6.4	A soup of electrons with randomly oriented spins	140
6.5	The z -projection of one electron’s spin	141
6.6	The Classical range of z -projection of spin	141
6.7	The measurements force S_z to “snap” into one of two values.	142
6.8	Near “vertical” spin measurements give illegally accurate knowledge of S_x , S_y and S_z	143
6.9	Plans for a follow-up to experiment # 1	144
6.10	Results of follow-up measurements of S_z on the $+z$ ($ +\rangle$) and $-z$ ($ -\rangle$) groups.	144
6.11	Experiment #2: $ +\rangle_z$ electrons enter an S_x measurement apparatus. . .	145
6.12	The input state for experiment # 2	146
6.13	The x -projection of spin	146
6.14	Viewed from top left, the classical range of x -projection of spin. . . .	146
6.15	A guess about the states of two groups after experiment #2	147
6.16	$ -\rangle$ electrons emerging from a group of $ +\rangle$ electrons	148
6.17	The destruction of $ +\rangle S_z$ information after measuring S_x	149
6.18	A spin direction with polar angle θ from $+z$, represented by $ \psi\rangle$. . .	151
6.19	The prepared state for experiment #3, prior to measurement	152
6.20	Classical, or semi-classical expectation of experiment #3’s measurement	152
6.21	Probabilities of measuring $ \pm\rangle$ from starting state $ \psi\rangle$, θ from $+z$. .	153
19.1	Graph of the quintessential periodic function $y = \sin x$	550

19.2	The function $y = \tan x$ blows-up at isolated points but is still periodic (with period π)	550
19.3	Graph of a function defined only for $x \in [-1, 3)$	550
19.4	Graph of a function defined everywhere, but whose support is $[-1, 3]$, the closure of $[-1, 0) \cup (0, 3)$	551
19.5	A periodic function that can be expressed as a Fourier series	552
19.6	A function with bounded domain that can be expressed as a Fourier series (support width = 2π)	552
19.7	A low frequency ($n = 1 : \sin x$) and high frequency ($n = 20 : \sin 20x$) basis function in the Fourier series	554
19.8	$f(x) = x$, defined only on bounded domain $[-\pi, \pi)$	555
19.9	$f(x) = x$ as a periodic function with fundamental interval $[-\pi, \pi)$:	555
19.10	First 25 Fourier coefficients of $f(x) = x$	556
19.11	Graph of the Fourier coefficients of $f(x) = x$	556
19.12	Fourier partial sum of $f(x) = x$ to $n = 3$	558
19.13	Fourier partial sum of $f(x) = x$ to $n = 50$	558
19.14	Fourier partial sum of $f(x) = x$ to $n = 1000$	559
19.15	$f(x)$ has bounded domain, but its Fourier expansion is periodic.	559
19.16	$f = 10$ produces ten copies of the period in $[-.5, .5)$	564
19.17	$f = .1$ only reveals one tenth of period in $[-.5, .5)$	564
20.1	$\cos x$ is even	570
20.2	$\sin x$ is odd	570
20.3	Square-integrable wavefunctions from Wikipedia StationaryStatesAnimation.gif, leading to the absolutely integrable $\psi_k^2(x)$	571
20.4	A simple function and its Fourier transform	572
20.5	Interpretation of σ from Wikipedia Standard_deviation_diagram)	573
20.6	Graph of the Dirac delta function	574
20.7	Sequence of box functions that approximate the delta function with increasing accuracy	574
20.8	Sequence of smooth functions that approximate the delta function with increasing accuracy	575
20.9	$\mathcal{F}[1]$ is a 0-centered delta function	577
20.10	$\mathcal{F}[\cos x]$ is a pair of real-valued delta functions	578
20.11	$\mathcal{F}[\sin x]$ is a pair of imaginary delta functions	578
20.12	Area under $ f ^2$ and $ F ^2$ are equal (Plancherel's Theorem)	580
20.13	$\sin(x)$ and its spectrum	582

20.14	$\sin(3x)$ and its spectrum	582
20.15	A normalized Gaussian with $\sigma^2 = 3$ and its Fourier transform	584
20.16	A more localized Gaussian with $\sigma^2 = 1/7$ and its Fourier transform	584
21.1	Continuous functions on a continuous domain	587
21.2	Sampling a continuous function at finitely many points	587
21.3	Primitive 5th root of 1 (the thick radius) and the four other 5th roots it generates	589
21.4	The \mathcal{DFT} of a non-periodic vector	596
21.5	The spectrum of a periodic vector	597
21.6	The spectrum of a very pure periodic vector	597
21.7	Going from an 8-element array to two 4-element arrays (one even and one odd)	606
21.8	Decomposing the even 4-element array to two 2-element arrays (one even and one odd)	607
21.9	Breaking a two-element array into two singletons	607
21.10	Final positions of f_2 should be next to f_4	608
21.11	Singletons are their own \mathcal{DFT} s	608
21.12	Bit-reversal reorders an eight-element array	609
21.13	$\left[F_{EO}^{(2)}\right]_j = \left[F_{EOE}^{(1)}\right]_{j \pmod{1}} + (-1)^{-j} \left[F_{EOO}^{(1)}\right]_{j \pmod{1}}$	611
21.14	$\left[F_E^{(4)}\right]_j = \left[F_{EE}^{(2)}\right]_{j \pmod{2}} + (i)^{-j} \left[F_{EO}^{(2)}\right]_{j \pmod{2}}$	612
23.1	The spectrum of a vector with period 8 and frequency $16 = 128/8$	646
23.2	$\sin(3x)$ and its spectrum	646
23.3	The spectrum of a purely periodic vector with period 8 and frequency $16 = 128/8$	647
23.4	Graph of two periods of a <i>periodic injective</i> function	648
23.5	Example of a periodic function that is not <i>periodic injective</i>	650
23.6	We add the weak assumption that $2(+)$ a -intervals fit into $[0, M)$	651
23.7	Typical application provides many a -intervals in $[0, M)$	652
23.8	Our proof will also work for only one a interval in $[0, M)$	652
23.9	$N = 2^n$ chosen so $(N/2, N]$ bracket M^2	653
23.10	Eight highly probable measurement results, cm , for $N = 128$ and $a = 8$	656
23.11	<i>Easy case</i> covers $a N$, exactly	661
23.12	$[0, N)$ is the union of distinct cosets of size a	661

23.13	The spectrum of a purely periodic vector with period 8 and frequency 16 = 128/8	664
23.14	Eight probabilities, .125, of measuring a multiple of $m = 16$	670
23.15	There is (possibly) a remainder for N/a , called the “excess”	676
23.16	$[0, N)$ is the union of distinct cosets of size a , except for last	676
23.17	The final coset may have size $< a$	677
23.18	If $0 \leq x < N - ma$, a full $m + 1$ numbers in \mathbb{Z}_N map to $f(x)$	678
23.19	If $N - ma \leq x < a$, only m numbers in \mathbb{Z}_N map to $f(x)$	678
23.20	The spectrum of a purely periodic vector with period 10 and frequency 12.8 = 128/10	680
23.21	A very long line consisting of a copies of $N = 2^n$	684
23.22	Half-open intervals of width a around each point cN	685
23.23	Exactly one integral multiple of a falls in each interval	685
23.24	Probabilities of measuring $y_4 = 51$, $y_5 = 64$ or $y_6 = 77$ are dominant.	688
23.25	\hat{y}_c all fall in the interval $[-a/2, a/2)$	689
23.26	The chord is shorter than the arc length	691
23.27	$2 \sin(x/2) $ lies above $2 x/\pi $ in the interval $(-\pi, \pi)$	692
23.28	$2 \sin(x/2) $ lies above $2 Kx/\pi $ in the interval $(-1.5\pi, 1.5\pi)$	698
23.29	Exactly one integral multiple of a falls in each interval	701
23.30	$N = 2^n$ chosen so $(N/2, N]$ bracket M^2	702
23.31	$2 \sin(x/2) $ lies above $2 Lx/\pi $ in the interval $(-.4714\pi, .4714\pi)$	707
24.1	A wide shot of the of the convergents starting at 2/1	725
24.2	A close-up showing some of the latter convergents.	725
24.3	First view of convergents	726
24.4	Second view of convergents	727
24.5	Third view of convergents	727
24.6	Fourth view of convergents	727
24.7	Fifth view of convergents	727

List of Tables