LECTURE NOTES ON QUANTUM COMPUTATION

from Mermin

Second pass on quantum computation

where we rely on mathematical ideas and arguments for many of the choices to we made earlier.

I note that if we did not make the first pass using QM first, many of the mathematical choices we make in pass #2 would not be obvious!

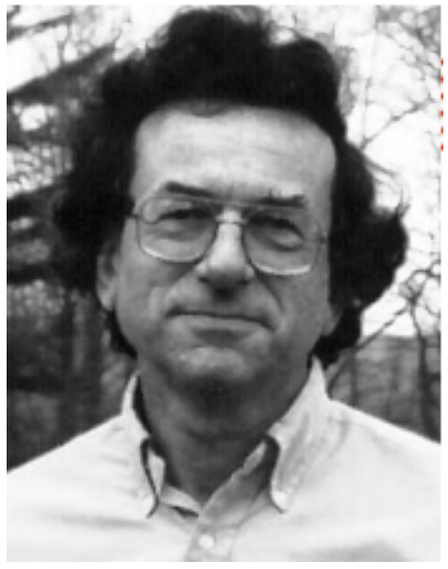
I hope this effort will fill all the gaps you feel you have in your mathematical knowledge, quantum mechanics knowledge and, of course, in quantum computation.

Now is the time to make sure you understand all this stuff!

I think you will after Mermin's Approach.

I chose to put it second so that you have a first pass before seeing the way Mermin approaches the same topics.

I note that Mermin joined the Cornell University faculty in 1964 (while I was still in graduate school. We became good friends and he was a great influence on my thinking (not quite as much as Hans Bethe and Ken Wilson) so I found his approach to quantum computing following the same path that I would have chosen! He is truly a great physicist and also a wonderful person! He often came to visit at Swarthmore when I was there - it was always enjoyable and fun for me and my students, no matter what class I happened to be teaching at that time! If you want to understand why I think about physics the way I do - look up the works of these guys!





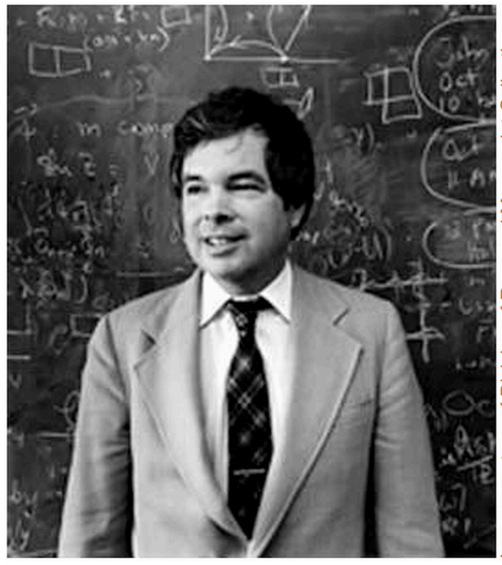


Image courtesy of Division of Rare and Manuscript Collections, Cornell University Library.

To those who do not know mathematics it is difficult to get across a real feeling as to the beauty, the deepest beauty, of nature. If you want to learn about nature, to appreciate nature, it is necessary to understand the language that she speaks in.

- R. Feynman

Part I. Fundamental Properties of Cbits and Qbits

It is tempting to say that a quantum computer is one whose operation is governed by the laws of quantum mechanics.

But since the laws of quantum mechanics govern the behavior of all physical phenomena, this temptation must be resisted.

Your laptop operates under the laws of quantum mechanics, but it is not a quantum computer.

A quantum computer is one whose operation exploits certain very special transformations of its internal state.

The laws of quantum mechanics allow these peculiar transformations under very carefully controlled conditions.

For a computer to be a quantum computer the physical systems that encode the individual bits must have no physical interactions whatever that are not under the complete control of the program.

All other interactions, however irrelevant they might be in an ordinary computer — which we shall call "classical" when we wish to contrast it to a quantum computer — introduce potentially catastrophic disruptions into the operation of a quantum computer.

Such disastrous interactions can include interactions with the external environment — air molecules bouncing off the physical systems that represent bits, or those systems absorbing a minute amount of ambient radiant thermal energy.

There can even be disruptive interactions between the computationally relevant degrees of freedom of the physical systems that represent bits with other degrees of freedom of those same systems, associated with computationally irrelevant features of their internal structure.

All such interactions between what is computationally relevant and what is not are said to result in "decoherence", which is **fatal** to a quantum computation.

To avoid decoherence individual bits cannot be encoded in physical systems of macroscopic size, because such systems *cannot* be isolated from their own irrelevant internal degrees of freedom.

The bits must be encoded in a very small number of states of a system of atomic size, where *extra internal degrees of freedom do not come into play* because they do not exist, or because they require unavailable large amounts of energy to excite.

Such atomic-scale systems must also be decoupled from their surroundings except for the completely controlled interactions that are associated with the computational process itself.

Two things keep the situation from being hopeless.

First, because the separation between the discrete energy levels of a system on the atomic scale can be enormously larger than the separation between the levels of a large system, the dynamical isolation of an atomic system is easier to achieve.

It can take a substantial kick to knock an atom out of its ground state.

The second reason for hope is the discovery that errors induced by extraneous interactions can actually be corrected, provided they occur at a sufficiently low rate.

While error correction is routine for bits represented by classical systems, quantum error correction is constrained by the formidable requirement that it be done in the *absence of any knowledge* of what either the original or the corrupted state of the bits might actually be.

Remarkably, this turns out to be possible.

But although the situation is not hopeless, the practical difficulties in the way of achieving useful quantum computation are enormous.

Only a rash person would declare that there will be no useful quantum computers by the year 2050, but only a rash person would predict that there will be.

Never mind.

Whether or not it will ever become a practical technology, there is a beauty to the theory of quantum computation that gives it a powerful appeal (a) as a lovely branch of abstract mathematics, (b) as a generalization of the paradigm of classical computer science that had completely escaped the attention of computer scientists until the 1980's, demonstrating that in a very deep sense the theory of computation cannot be divorced from the physics of the devices that carry out the computation, and (c) as a source of new examples that illustrate and illuminate the surprising and often mysterious kinds of phenomena that the quantum behavior of matter can give rise to.

You may or may not find point (a) compelling.

Many physicists (not including me) are immune to the songs of this particular siren.

The most striking manifestation of point (b) is that for certain special computational tasks of practical interest, a quantum computer can be vastly more efficient than anything ever imagined in the classical theory of computational complexity, in that the time it takes the quantum computer to accomplish the task scales up much more slowly with the size of the input than it can in any classical computer.

Much of what follows will be devoted to examining the most celebrated examples of this speed-up.

Item (c) brings us to our first topic.

Many of my theoretical physicist friends would be aghast that I might spend the first four or five lectures of a course in quantum computation giving an introduction to quantum mechanics for mathematically literate people who knew nothing about quantum mechanics and quite possibly little if anything about physics.

Their response would probably be that any application of quantum mechanics that can be taught after only a four hour introduction to the subject, cannot have serious intellectual content.

After all, they would say, it takes any physicist many years to develop a feeling for quantum mechanics.

It's a good point.

That is why I chose to make the first part of these lectures very dependent on quantum mechanics as you will see.

Nevertheless computer scientists and mathematicians with no background in physics have been able quickly to learn enough quantum mechanics to understand and contribute importantly to the theory of quantum computation and possibly have gotten a better understanding then those insisting on a rigorous quantum mechanics approach!!

I believe there are two main reasons for this.

First of all, a quantum computer — or, more accurately, the abstract quantum computer that one hopes some day to be able to embody in actual hardware — is an extremely simple example of a physical system.

It is discrete, not continuous.

It is made up out of a finite number of units, each of which is the simplest possible kind of quantum mechanical system, a so-called two-state system, whose possible behavior, as we shall see, is highly constrained and easily analyzed.

Much of the analytical complexity of learning quantum mechanics is connected to mastering the description of continuous (infinite-state) systems.

By restricting attention to collections of two-state (or even d-state systems for finite d) one can avoid much suffering.

Of course one also loses much wisdom, but hardly any of it — at least at this stage of the art — is relevant to the theory of quantum computation.

Second, and just as important, the most difficult part of learning quantum mechanics is to get a good feeling for how the abstract formalism can be applied to actual phenomena.

This almost invariably involves formulating oversimplified abstract models of the real phenomena, to which the quantum formalism can then be applied.

The best physicists have an extraordinary intuition for what features of the phenomena are essential and must be represented in an abstract model, and what features are inessential and can be ignored.

It takes years to develop such intuition.

Some never do!

The theory of quantum computation, however, is entirely concerned with the abstract model — the easy part of the problem.

To understand how to build a quantum computer, or even to study what physical systems are promising candidates for realizing such a device, you must indeed have many years of experience in quantum mechanics and its applications under your belt.

But if you only want to know what such a device is capable of doing in principle, then there is no reason to get involved in the really difficult physics of the subject.

The same thing holds for ordinary classical computers.

One can be a masterful practitioner of computer science without having the foggiest notion of what a transistor is, not to mention how it works.

So while you should be warned that the subset of quantum mechanics we will use here is extremely focused and quite limited in its scope, you can also be assured that it is neither oversimplified nor incomplete, when applied to the special task for which it is intended.

It might also be noted that another impediment to developing a good intuition for quantum physics is that in some ways, the behavior implied by quantum mechanics is highly counterintuitive, if not downright weird.

Glimpses of such strange behavior sometimes show up at the level of quantum computation.

One of the major appeals of quantum computation, at least for many, is that it affords a new conceptual arena for trying to come to a better understanding of quantum weirdness.

When opportunities arise I will try to call attention to some of this strange behavior, rather than letting it pass by unremarked upon and unnoticed.

We begin our survey of quantum computation with a minimalist introduction to quantum mechanics, designed to give you, as quickly as possible, the conceptual tools you need to delve into the theories of quantum computation and quantum information processing. **I hope you realize you know this stuff!!**

I do this by restating the fundamentals of quantum mechanics, not as the remarkable revision of classical Newtonian mechanics required to account for the behavior of matter at the atomic and subatomic levels, but as a curious way to generalize the behavior of an ordinary classical digital computer.

By focusing on the special topic of how quantum mechanics enlarges the possibilities for the physical manipulation of digital information, it is possible to characterize how the quantum theory works in an elementary and quite concise way, which is nevertheless complete for this restricted area of application.

While this part of the lectures assumes no prior familiarity with quantum mechanics(on purpose), I do assume that you are well acquainted with linear algebra in finite-dimensions using complex numbers.

Hopefully, the first part of the lectures will make sure of that truth!

So, let us proceed with this different approach and I hope the combination of the two approaches will produce and much better overall understanding of quantum computation.

A. Cbits and their states.

We begin with a minimalist statement of what an ordinary classical computer does.

I shall frame the elementary — indeed banal — remarks that follow in a language which, though it may look artificial and cumbersome, is designed to accommodate the richer variety of things that a computer can do if it is designed to take full advantage of the possibilities made available by the quantum mechanical behavior of its constituent parts.

Introducing and applying the unfamiliar nomenclature and notation of quantum mechanics in a familiar classical setting should make its subsequent extension to the broader quantum context look a little less peculiar.

A classical computer operates on strings of 0's and 1's, such as 1100101110110000, converting them into other such strings.

Each position in such a string is called a bit, and it contains either a 0 or a 1.

To represent such collections of bits a computer must contain a corresponding collection of physical systems, each of which can exist in two unambiguously distinguishable physical states, associated with the value (0 or 1) of the abstract bit that the physical system represents.

Such a physical system could be, for example, a switch which could be open (0) or shut (1), or a magnet whose magnetization could be oriented in two different directions, “up” (0) or “down” (1).

It is a common practice in quantum computer science to use the same term “bit” to describe the two-state classical system that represents the value of the abstract bit.

But this use of a single term to characterize both the abstract bit (0 or 1) and the physical system whose two states represent the two values is a *potential source of confusion*.

To avoid such confusion, I shall use the term Cbit (“C” for “classical”) to describe the two-state classical physical system and Qbit to describe its quantum generalization.

This terminology is inspired by Paul Dirac’s early use of c-numbers and q-numbers to describe classical quantities and their quantum-mechanical generalizations.

“Cbit” and “Qbit” are preferable to “c-bit” and “q-bit” because the terms themselves often appear in hyphenated constructions.

Unfortunately the preposterous spelling “qubit” (*that we have been using*) currently holds sway for the quantum system.

Although “qubit” honors the English (German, Italian,...) rule that q should be followed by u, it ignores the equally powerful requirement that qu should be followed by a vowel!

My guess is that “ qubit” has gained acceptance because it visually resembles an ancient English unit of distance, the “homophone” *cubit*.

To see its ungainliness with fresh eyes, it suffices to imagine that Dirac had written *qunumber* instead of q-number, or that one erased transparencies and cleaned one’s ears with *Qutips*.

Because clear distinctions between bits, Cbits, and Qbits are crucial in the introduction to quantum computation that follows, I shall use this unfashionable but superior terminology.

To prepare for the extension from Cbits to Qbits I introduce what may well strike you as a degree of notational overkill in the discussion of Cbits that follows.

We shall represent the state of each Cbit as a kind of box, depicted by the symbol $|\ \rangle$, into which we place the value, 0 or 1, represented by that state.

Thus the two distinguishable states of a Cbit are represented by the symbols $|0\rangle$ and $|1\rangle$.

It is the common practice to call the symbol $|0\rangle$ or $|1\rangle$ itself the state of the Cbit, thereby using the same term to refer to both the physical condition of the Cbit and the abstract symbol that represents that physical.

There is nothing unusual in this.

For example one commonly uses the term “position” to refer to the symbol x that represents the physical position of an object.

I call this common, if little noted, practice to your attention only because in the quantum case “state” refers *only* to the symbol, there being no internal property of the Qbit that the symbol represents.

The rather subtle relation between Qbits and their state symbol will emerge later in this part

Along the same lines, we shall characterize the states of the 5 Cbits representing 11001, for example, by the symbol

$$|1\rangle|1\rangle|0\rangle|0\rangle|1\rangle, \tag{1.1}$$

and refer to this object as the *state* of all five Cbits.

Thus a pair of Cbits can have (or "be in") any of the four possible states,

$$|0\rangle|0\rangle,\ |0\rangle|1\rangle,\ |1\rangle|0\rangle,\ \text{or}\ |1\rangle|1\rangle, \tag{1.2}$$

three Cbits can be in any of the eight possible states,

$$|0\rangle|0\rangle|0\rangle,\ |0\rangle|0\rangle|1\rangle,\ |0\rangle|1\rangle|0\rangle,\ |0\rangle|1\rangle|1\rangle,\ |1\rangle|0\rangle|0\rangle,\ |1\rangle|0\rangle|1\rangle,\ |1\rangle|1\rangle|0\rangle,\ \text{or}\ |1\rangle|1\rangle|1\rangle, \tag{1.3}$$

and so on.

As (1.3) already makes evident, when there are many Cbits such products are often much easier to read if one encloses the whole string of 0's and 1's in a single bigger box of the form $|\ \rangle$ rather than having a separate box for each Cbit:

$$|000\rangle,\ |001\rangle,\ |010\rangle,\ |011\rangle,\ |100\rangle,\ |101\rangle,\ |110\rangle,\ \text{or}\ |111\rangle. \tag{1.4}$$

We shall freely move between these two equivalent ways of expressing the state of several Cbits that represent a string of bits, boxing the whole string, or boxing each individual bit.

Whether the form (1.3) or (1.4) is to be preferred depends on the context.

There is also a third form, which is useful when we regard the 0's and 1's as constituting the binary expansion of an integer.

We can then replace the representations of the 3-Cbit states (1.4) by the even shorter forms:

$$|0\rangle,\ |1\rangle,\ |2\rangle,\ |3\rangle,\ |4\rangle,\ |5\rangle,\ |6\rangle,\ \text{or}\ |7\rangle. \tag{1.5}$$

Note, though, that unlike the forms (1.3) and (1.4), the form (1.5) is **ambiguous**, unless we are told that these symbols express the state of 3 Cbits.

If we are not told, then there is no way of telling, for example, whether $|3\rangle$ represents the 2-Cbit state $|11\rangle$ or the 3-Cbit state $|011\rangle$ or the 4-Cbit state $|0011\rangle$, etc.

This ambiguity can be removed, when necessary, by adding a subscript making the number of Cbits explicit:

$$|0\rangle_3,\ |1\rangle_3,\ |2\rangle_3,\ |3\rangle_3,\ |4\rangle_3,\ |5\rangle_3,\ |6\rangle_3,\ \text{or}\ |7\rangle_3. \tag{1.6}$$

Be warned, however, that when there is no ambiguity about how many Cbits $|x\rangle$ represents, it can be useful to use such subscripts for other purposes.

If, for example, Alice and Bob (Alice and Bob, long the heroine and hero of diverse cryptographic adventures, play major roles in the exposition of quantum computation and information processing) each possess a single Cbit it can be convenient to describe the state of Alice's Cbit (if it has the value 1) by $|1\rangle_a$, Bob's (if it has the value 0) by $|0\rangle_b$, and the joint state of the two by $|1\rangle_a|0\rangle_b$ or $|10\rangle_{ab}$.

Dirac introduced the $|\,\rangle$ notation (known as Dirac notation) in the early days of the quantum theory, as a useful way to write and manipulate vectors - once you understand it —> a new language for QM(Part II).

For silly reasons —see Part F later — he called such vectors kets, a terminology that has survived to this day.

Class 9

In Dirac notation you can put into the box $|\,\rangle$ anything that serves to specify what the vector is.

If, for example, we were talking about displacement vectors in ordinary 3-dimensional space, we could have a vector

$$|5 \text{ horizontal centimeters northeast}\rangle. \tag{1.7}$$

In using Dirac notation to express the state of a Cbit, or a collection of Cbits, I'm suggesting that there might be some utility in thinking of the states as vectors.

Is there?

Well in the case of Cbits, not very much, but maybe a little.

We shall briefly explore what one can do with Cbits when one takes the two states $|0\rangle$ and $|1\rangle$ of a single Cbit to be represented by *two orthogonal unit vectors in a 2-dimensional space*.

While this is little more than a curious and unnecessarily elaborate way of describing Cbits, it is fundamental and unavoidable in dealing with Qbits.

Playing unfamiliar and somewhat silly games with Cbits will, I hope, provide a painless way of becoming introduced to some of the quantum mechanical formalism in a familiar setting.

If you prefer your vectors to be expressed in components, note that we can represent the two orthogonal states of a single Cbit, $|0\rangle$ and $|1\rangle$, as column vectors

$$|0\rangle \longleftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle \longleftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1.8}$$

In the case of two Cbits the vector space is 4-dimensional, with an orthonormal basis

$$|00\rangle,\ |01\rangle,\ |10\rangle,\ |11\rangle. \tag{1.9}$$

The alternative notation for this basis,

$$|0\rangle|0\rangle,\ |0\rangle|1\rangle,\ |1\rangle|0\rangle,\ |1\rangle|1\rangle, \tag{1.10}$$

is deliberately designed to suggest multiplication, since it is, in fact, a short-hand notation for the tensor products of the two single-Cbit 2-vectors, written in more formal mathematical notation as

$$|0\rangle \otimes |0\rangle,\quad |0\rangle \otimes |1\rangle,\quad |1\rangle \otimes |0\rangle,\quad |1\rangle \otimes |1\rangle. \tag{1.11}$$

FYI: the tensor product a⊗b of an M-component vector a with components a_μ and an N-component vector b with components b_ν is the (MN)-component vector with components indexed by all the MN possible pairs of indices (μ,ν), whose $(\mu,\nu)^{th}$ component is just the product $a_\mu b_\nu$.

I shall freely move back and forth between these various ways of writing the tensor product, trying in each case to choose the form that makes the content easiest to read.

Before proceeding with this discussion, let me digress to hopefully make you feel better about tensor products.

The Tensor Product, Demystified

A recurring theme throughout mathematics is making new things from old things.

Mathematicians do this all the time:

- When you have two integers, you can find their greatest common divisor or least common multiple.
- When you have some sets, you can form their Cartesian product or their union.
- When you have two groups, you can construct their direct sum or their free product.
- When you have a topological space, you can look for a subspace or a quotient space.
- When you have some vector spaces, you can ask for their direct sum or their intersection.
- The list goes on!

To help our understanding in this class, I'd like to focus on a particular way to build a new vector space from old vector spaces: **the tensor product**.

This construction often come across as scary and mysterious, but I hope to help shine a little light and dispel some of the fear.

In particular, we won't talk about axioms or universal properties.

Instead, we'll take an elementary, concrete look:

Given two vectors **v** and **w**, we can build a new vector, called the *tensor product* **v**⊗**w**.

But what is that vector, really?

Likewise, given two vector spaces V and W , we can build a new vector space, also called their tensor product V ⊗ W .

But what is that vector space, really?

Making new vectors from old

In this discussion, we'll assume V and W are finite dimensional vector spaces.

That means we can think of V as R^n and W as R^m for some positive integers n and m.

So a vector **v** in R^n is really just a list of n numbers, while a vector **w** in R^m is just a list of m numbers.

$$\vec{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

a vector in $\mathbb{R}^3$

$$\vec{w} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

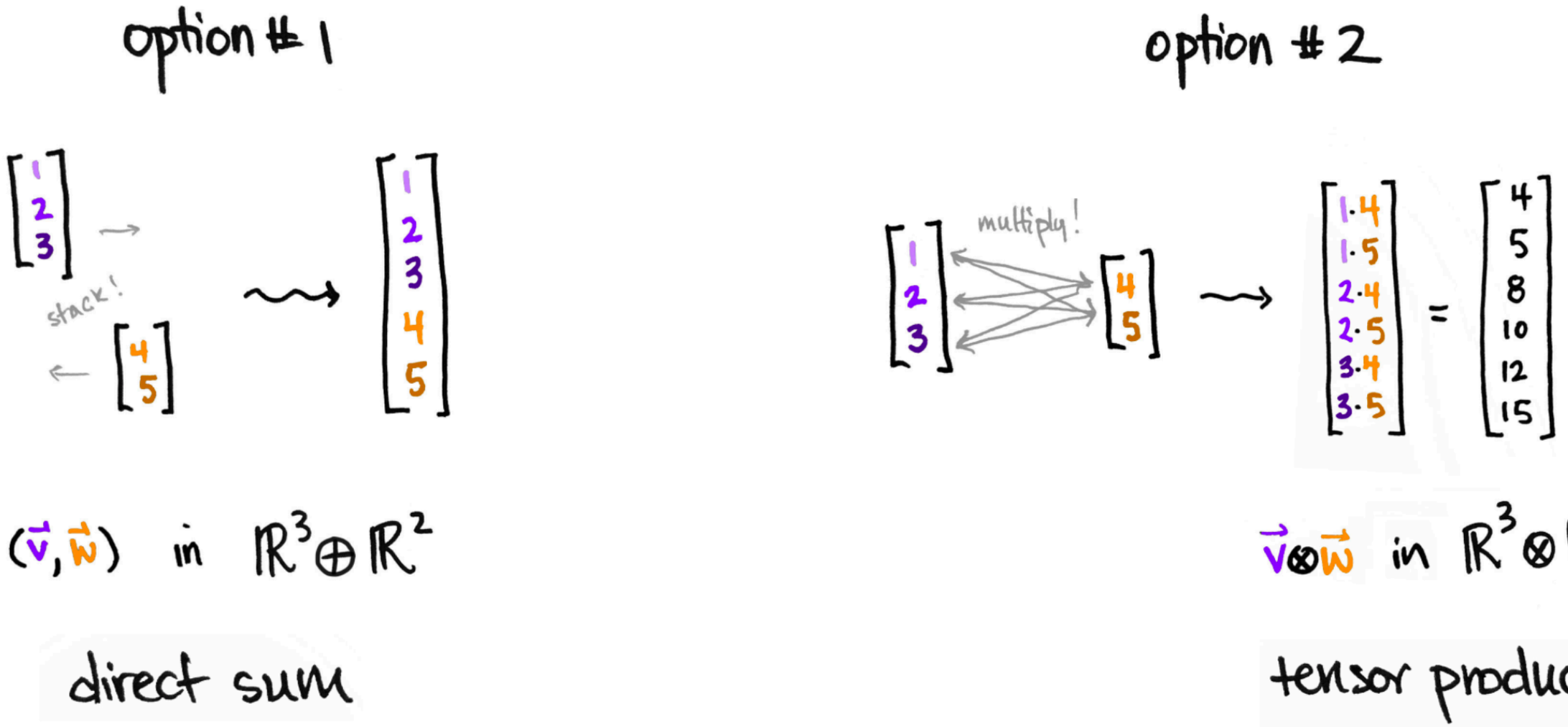
a vector in $\mathbb{R}^2$

Let's try to make new, third vector out of **v** and **w**.

But how?

Here are two ideas:

We can stack them on top of each other, or we can first multiply the numbers together and then stack them on top of each other.

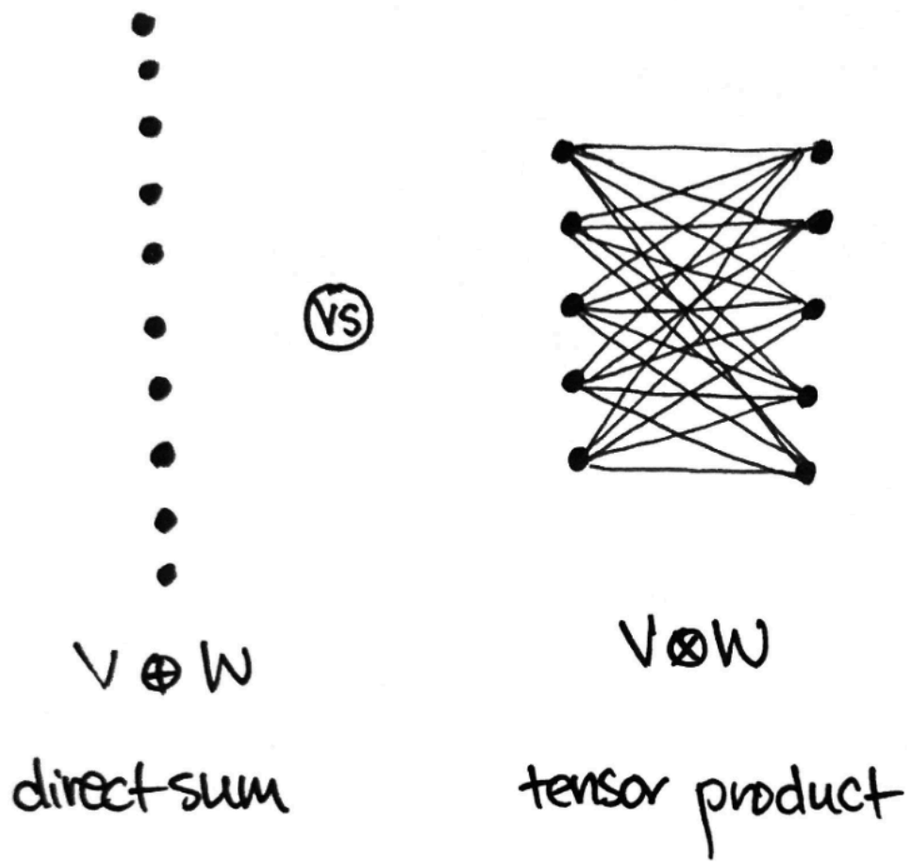


The first option gives a new list of n + m numbers, while the second option gives a new list of nm numbers.

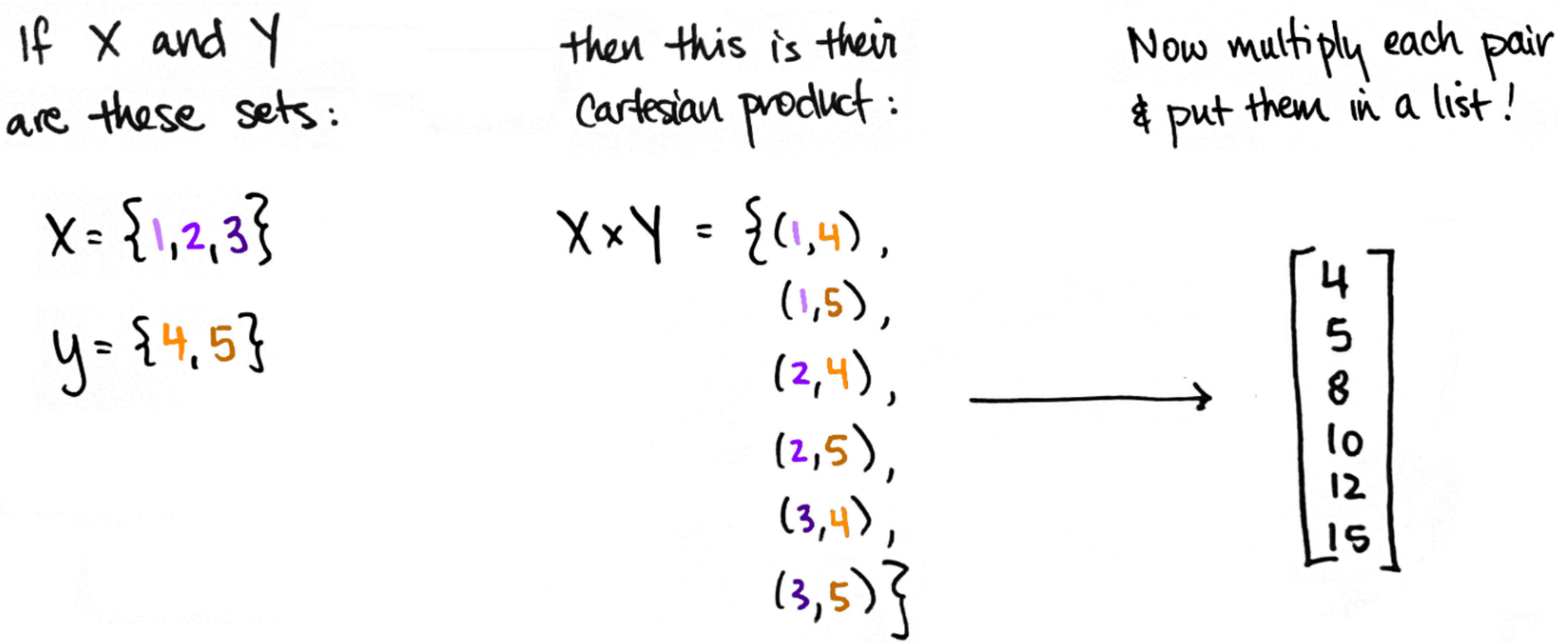
The first gives a way to build a new space where the dimensions add; the second gives a way to build a new space where the dimensions multiply.

The first is a vector (**v**,**w**) in the *direct sum* V⊕W (this is the same as their direct product V×W); the second is a vector **v**⊗**w** in the *tensor product* V⊗W.

And that's it!



Forming the tensor product $\mathbf{v}\otimes\mathbf{w}$ of two vectors is a lot like forming the Cartesian product of two sets X x Y. In fact, that's exactly what we're doing if we think of X as the set whose elements are the entries of **v** and similarly for Y and **w**.



So a tensor product is like a grown-up version of multiplication.

It's what happens when you systematically multiply a bunch of numbers together, then organize the results into a list.

It's multi-multiplication, if you will.

There's a little more to the story.

Does every vector in $V \otimes W$ look like $\mathbf{v} \otimes \mathbf{w}$ for some $\mathbf{v} \in V$ and $\mathbf{w} \in W$?

Not quite.

Remember, a vector in a vector space can be written as a weighted sum of basis vectors, which are like the space's building blocks.

This is another instance of making new things from existing ones: we get a new vector by taking a weighted sum of some special vectors!

If V has basis $\{v_1, v_2, v_3\}$ then

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = 1v_1 + 2v_2 + 3v_3 .$$

So a typical vector in $V \otimes W$ is a weighted sum of basis vectors.

What are those basis vectors?

Well, there must be exactly nm of them, since the dimension of $V \otimes W$ is nm.

Moreover, we'd expect them to be built up from the basis of V and the basis of W.

This brings us again to the "How can we construct new things from old things?" question.

Asked explicitly: If we have n bases $\mathbf{v}_1, \ldots, \mathbf{v}_n$ for V and if we have m bases $\mathbf{w}_1, \ldots, \mathbf{w}_m$ for W then

can we combine them to get a new set of nm vectors?

This is totally analogous to the construction we saw above: given a list of n things and a list of m things, we can obtain a list of nm things by multiplying them all together.

So we'll do the same thing here!

We'll simply multiply the $\mathbf{v}_i$ together with the $\mathbf{w}_j$ in all possible combinations, except “multiply $\mathbf{v}_i$ and $\mathbf{w}_j$ " now means "take the tensor product of $\mathbf{v}_i$ and $\mathbf{w}_j$.”

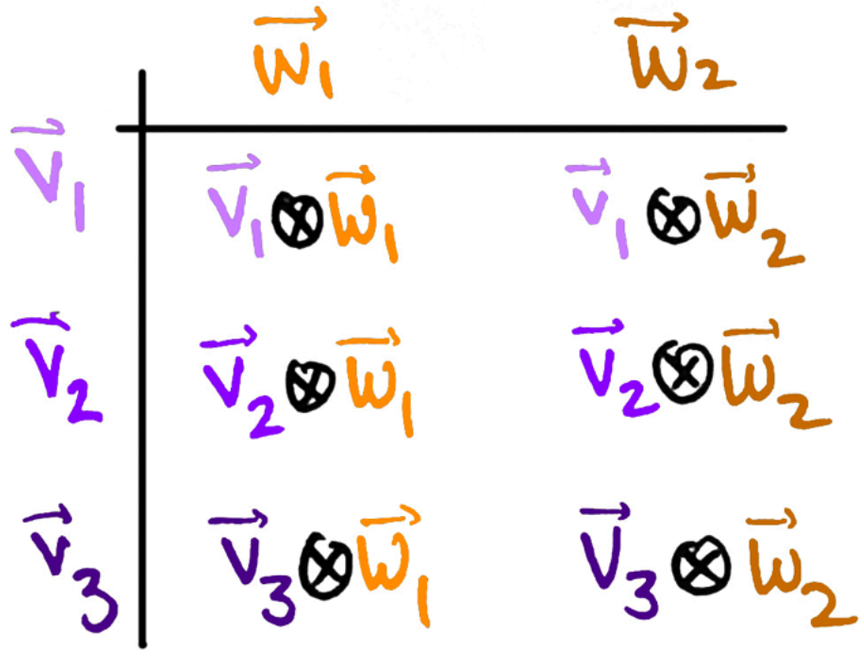
Concretely, a basis for $V \otimes W$ is the set of all vectors of the form $\mathbf{v}_i \otimes wj$ where i ranges from 1 to n and j ranges from 1 to m.

As an example, suppose n = 3 and m = 2 as before.

Then we can find the six basis vectors for $V \otimes W$ by forming a 'multiplication chart.'

The sophisticated way to say this is:

” V⊗W is the free vector space on A × B, where A is a set of generators for V and B is a set of generators for W”.



So V⊗W is the six-dimensional space with basis

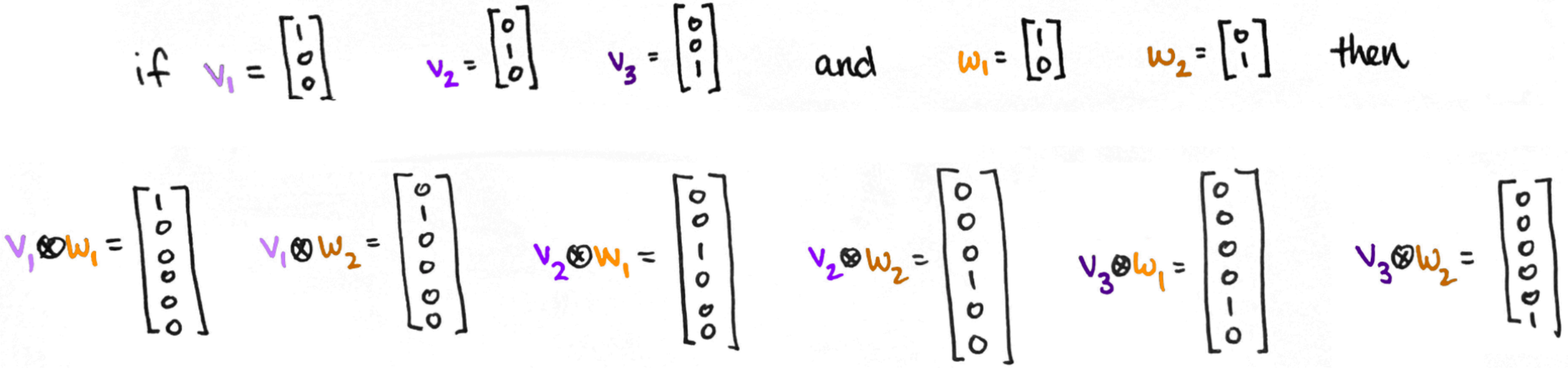
$$\{\mathbf{v}_1 \otimes \mathbf{w}_1,\ \mathbf{v}_1 \otimes \mathbf{w}_2,\ \mathbf{v}_2 \otimes \mathbf{w}_1,\ \mathbf{v}_2 \otimes \mathbf{w}_2,\ \mathbf{v}_3 \otimes \mathbf{w}_1,\ \mathbf{v}_3 \otimes \mathbf{w}_2\}$$

This might feel a little abstract with all the ⊗ symbols littered everywhere.

But don't forget—we know exactly what each $\mathbf{v}_i \otimes \mathbf{w}_j$ looks like—it's just a list of numbers!

Which list of numbers?

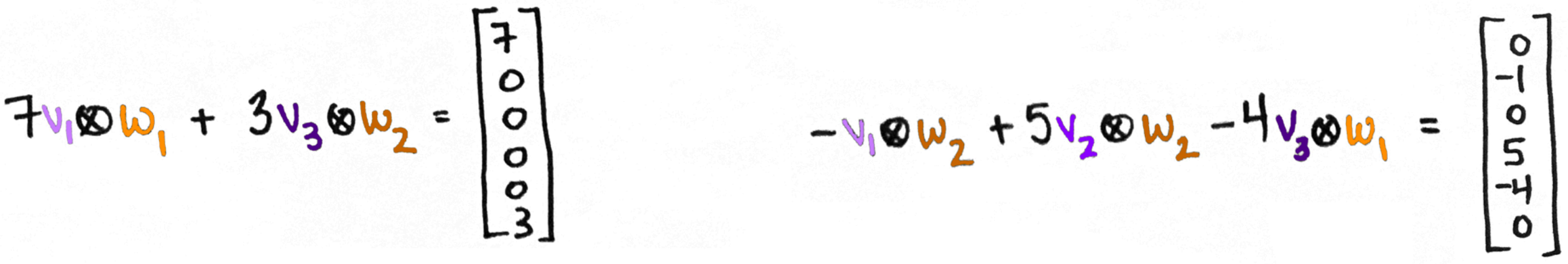
Well,



So what is V⊗W?

It's the vector space whose vectors are linear combinations of the $\mathbf{v}_i \otimes \mathbf{w}_j$.

For example, here are a couple of vectors in this space:



Well, technically...

Technically, $\mathbf{v} \otimes \mathbf{w}$ is called the *outer product* of $\mathbf{v}$ and $\mathbf{w}$ and is defined by

$$\mathbf{v} \otimes \mathbf{w} := \mathbf{v}\mathbf{w}^\top$$

where $\mathbf{w}^\top$ is the same as $\mathbf{w}$ but written as a row vector. (And if the entries of $\mathbf{w}$ are complex numbers, then we also replace each entry by its complex conjugate.)

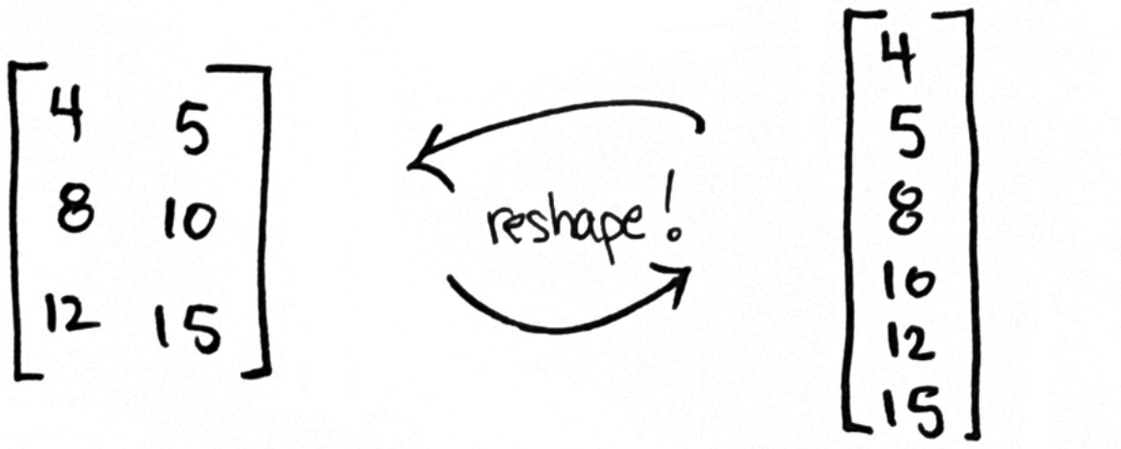
So technically the tensor product of vectors is matrix:

$$\vec{v} \otimes \vec{w} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 4 & 5 \end{bmatrix} = \begin{bmatrix} 1 \cdot 4 & 1 \cdot 5 \\ 2 \cdot 4 & 2 \cdot 5 \\ 3 \cdot 4 & 3 \cdot 5 \end{bmatrix} = \begin{bmatrix} 4 & 5 \\ 8 & 10 \\ 12 & 15 \end{bmatrix}$$

This may seem to be in conflict with what we did above, but it's not!

The two go hand-in-hand.

Any m x n matrix can be reshaped into a nm x 1 column vector and vice versa.



It's a little like a *process-state duality*.

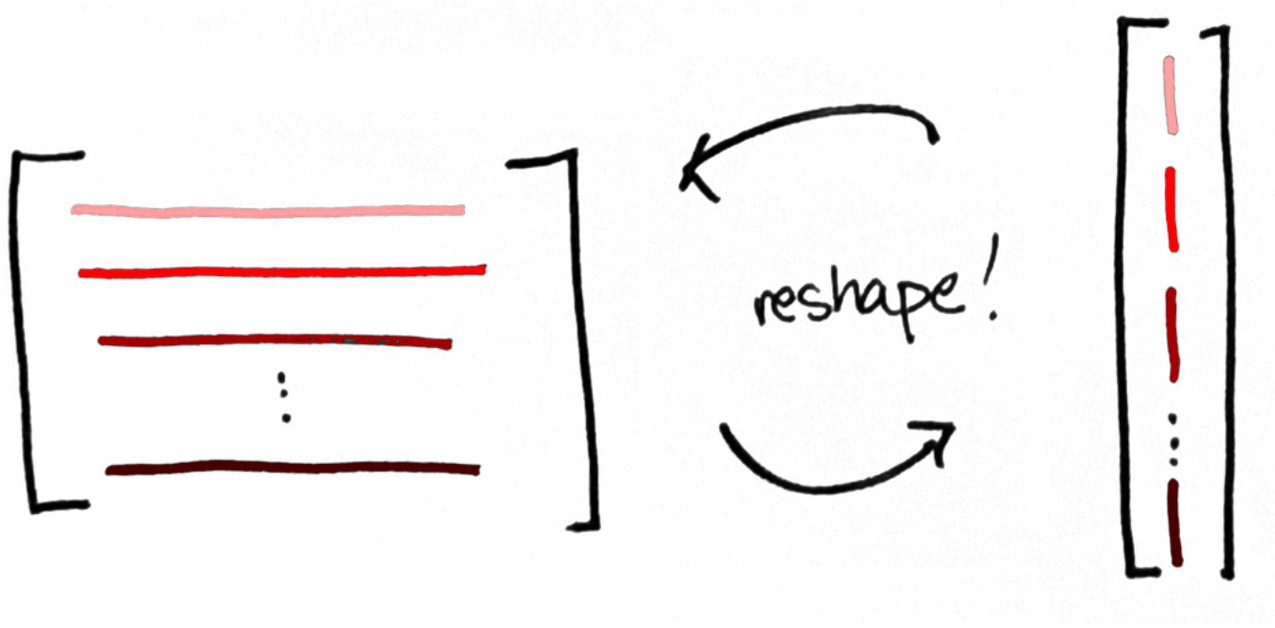
On the one hand, a matrix $\mathbf{v} \otimes \mathbf{w}$ is a *process*—it's a concrete representation of a (linear) *transformation*.

On the other hand, $\mathbf{v} \otimes \mathbf{w}$ is, abstractly speaking, a vector.

And a *vector* is the mathematical gadget that physicists use to describe the *state* of a quantum system.

So matrices *encode* processes; vectors *encode* states.

The upshot is that a vector in a tensor product V ⊗ W can be viewed in *either way* simply by reshaping the numbers as a list or as a rectangle.



In quantum physics

One application of tensor products is related to the brief statement I made above:

"A vector is the mathematical gadget that physicists use to describe the state of a quantum system."

To elaborate: if you have a quantum particle, perhaps you'd like to know what it's doing.

Or what it's capable of doing.

Or the probability that it'll be doing something.

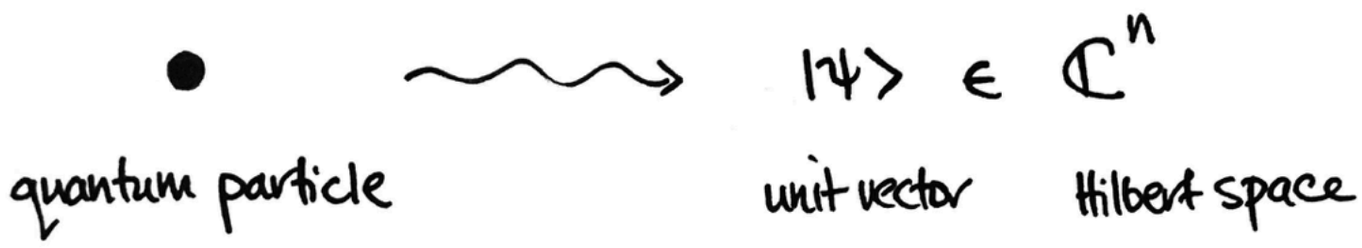
In essence, you're asking:

What's its status? What's its state?

The answer to this question— provided by a postulate of quantum mechanics—is given by a unit vector in a vector space.

Really, a Hilbert space, say $\mathbb{C}^n$.

That unit vector encodes information about that particle.

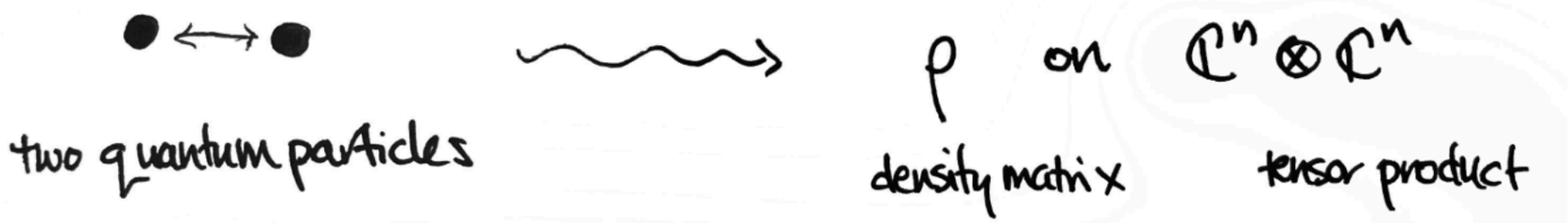


The dimension n is, loosely speaking, the number of different things you could observe after making a measurement on the particle.

But what if we have two quantum particles?

The state of that two-particle system can be described by something called a *density matrix* ρ on the tensor product of their respective spaces $\mathbb{C}^n \otimes \mathbb{C}^n$ —-> a lot of new stuff……

A density matrix is a generalization of a unit vector—it accounts for interactions between the two particles.



The same story holds for N particles—the state of an N-particle system can be described by a density matrix on an N -fold tensor product.

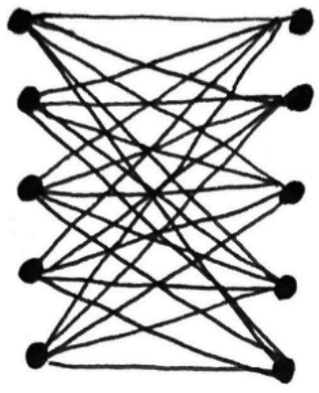


But why the *tensor product*?

Why is it that this construction—out of all things—describes the interactions within a quantum system so well, so naturally?

I don’t know the answer, but perhaps the appropriateness of tensor products shouldn't be too surprising.

The tensor product itself captures all ways that basic things can "interact" with each other!



$V \otimes W$

tensor product

Of course, there's lots more to be said about tensor products.

I've only shared a snippet of basic arithmetic.

Now back to our discussion.

Once one agrees to regard the two 1-Cbit states as orthogonal unit vectors, the tensor product is indeed the natural way to represent multi-Cbit states, since it leads to the obvious multi-Cbit generalization of the representation (1.8) of 1-Cbit states as column vectors.

If we express the states $|0\rangle$ and $|1\rangle$ of each single Cbit as column vectors, then we can get the column-vector describing a multi-Cbit state by applying the standard rule for the components of the tensor product of several two-dimensional vectors, illustrated here for a three-fold tensor product:

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \otimes \begin{pmatrix} z_0 \\ z_1 \end{pmatrix} \longleftrightarrow \begin{pmatrix} x_0y_0z_0 \\ x_0y_0z_1 \\ x_0y_1z_0 \\ x_0y_1z_1 \\ x_1y_0z_0 \\ x_1y_0z_1 \\ x_1y_1z_0 \\ x_1y_1z_1 \end{pmatrix}. \qquad (1.12)$$

This follows from two successive applications of the rule enunciated in the FYI statement.

Applying this, for example, to the case $|5\rangle_3$ we have

$$|5\rangle_3 = |101\rangle = |1\rangle|0\rangle|1\rangle \longleftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}. \qquad (1.13)$$

If we label the vertical components of the 8-vector on the right 0, 1,…, 7, from the top down, then the single non-zero component is the 1 in the position, 5, specified by the binary number 101 that the three-bit state vector specifies in its original form on the left of (1.13) - make sure you see it!!!!

This is indeed the obvious multi-Cbit generalization of the column-vector form (1.8) for 1-Cbit states.

This is quite general: the tensor product structure of multi-Cbit states is just what one needs in order for the 2^n-dimensional column vector representing a particular one of the 2^n possible states of n Cbits, to have all its entries zero except for a single 1 in the position down from the top specified by the number that those n Cbits are representing in binary.

Indeed, one can turn this development upside down, taking as one's starting point the simple rule that an integer x in the range $0 \leq x < N$ is represented by one of N orthonormal vectors in an N-dimensional space.

One can then pick a basis so that 0 is represented by an N-component column vector $|0\rangle$ which has 0 in every position except for a 1 in the top position, and x is to be represented by an N-component column vector $|x\rangle$ which has 0 in every position except for a 1 in the position x down from the top.

It then follows from the nature of the tensor product that when $N = 2^n$ and x has the binary expansion $x = \sum_{j=0}^{n-1} x_j 2^j$,, then the column vector $|x\rangle$ is the **tensor** product of the n 2-component column vectors $|x_j\rangle$:

$$|x\rangle = |x_{n-1}\rangle \otimes \cdots \otimes |x_0\rangle. \qquad (1.14)$$

Go over it again and again until your brain agrees!!!!!!

This relation between tensor products of vectors and positional notation for integers is not confined to the binary system.

If, for example, one represents a decimal digit x = 0, 1,…, as a 10-component column vector $\mathbf{v}^{(x)}$ with all components 0 except for a 1, x positions down from the top, then if the n-digit decimal number $X = \sum_{j=0}^{n-1} x_j 10^j$ is represented by the tensor product $\mathbf{V} = \mathbf{v}^{(x_{n-1})} \otimes \mathbf{v}^{(x_{n-2})} \otimes \cdots \otimes \mathbf{v}^{(1)} \otimes \mathbf{v}^{(0)}$, then **V** will be a 10^n-component column vector with all components 0 except for a 1, X positions down from the top.

$$X = \sum_{j=0}^{n-1} x_j 10^j$$

n = 4 , $x_0 = 2, x_1 = 3, x_2 = 7, x_3 = 4$

$x = 2x10^0 + 3x10^1 + 7x10^2 + 4x10^3 = 4732$

$$v^{(2)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad v^{(3)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{V} = \mathbf{v}^{(x_{n-1})} \otimes \mathbf{v}^{(x_{n-2})} \otimes \cdots \otimes \mathbf{v}^{(1)} \otimes \mathbf{v}^{(0)}$$

etc, etc, etc…

Although the representation of Cbit states by column vectors helps to motivate the use of the tensor product to describe multi-Cbit states, I emphasize that for almost all other purposes it is much better and simpler to forget about column vectors and components, and deal directly with the state vectors in their various abstract forms (1.3)-(1.6).

B. Reversible Operations on Cbits.

Quantum computers do an important part of their magic through *reversible* operations, which transform the initial state of the Qbits into its final form using only processes whose action can be inverted.

There is only one *irreversible* part of the operation of a quantum computer.

It is called *measurement*, and is the *only way to extract* useful information from the Qbits after their state has acquired its final form.

Although measurement is a nontrivial and crucial part of the quantum computational process, in a classical computer the extraction of information from the state of Cbits is so straightforward a procedure that it is rarely even described as part of the computational process, though it is, of course, a nontrivial concern for those who design digital displays or printers.

But all computationally relevant operations in a classical computer correspond to reversible operations on a quantum computer, and therefore only reversible operations on Cbits will be of interest to us here.

In a reversible operation every final state arises from a unique initial state.

An example of an irreversible operation is ERASE which forces a Cbit into the state $|0\rangle$ regardless of whether its initial state is $|0\rangle$or $|1\rangle$.

ERASE is irreversible in the sense that given only the final state and the fact that it was the output of the operation ERASE, there is no way to recover the initial state.

The only non-trivial reversible operation we can apply to a single Cbit is the NOT (or flip) operation, denoted by the symbol **X**, which interchanges (or flips) the two states $|0\rangle$and $|1\rangle$:

$$\mathbf{X} : |x\rangle \rightarrow |\overline{x}\rangle; \quad \overline{1} = 0, \quad \overline{0} = 1. \tag{1.15}$$

It is reversible because it has an inverse: applying NOT a second time brings the state of the Cbit back to its original form.

If we represent the two orthogonal states of the Cbit by the column vectors (1.8), then we can express NOT by a linear operator **X** on the 2-dimensional vector space, whose action on the column vectors is given by the matrix

$$\mathbf{X} \longleftrightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{1.16}$$

So the two reversible things you can do to a single bit — leaving it alone or flipping its state — correspond to the two linear operators **X** and **1**,

$$\mathbf{1} \longleftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{1.17}$$

on its two-dimensional vector space.

FYI: A pedantic point: Since multiplication by the scalar 1 and action by the unit operator **1** achieve the same result, I shall sometimes follow the possibly irritating practice of physicists and not distinguish notationally between them. The same applies to the scalar 0, the zero vector 0, and the zero operator **0**.

It can be useful to introduce a *number operator* **n** for a single Cbit, defined by

$$\mathbf{n}|x\rangle = x|x\rangle, \quad x = 0 \text{ or } 1; \tag{1.18}$$

i.e. $|0\rangle$ and $|1\rangle$are eigenvectors of **n** with eigenvalues 0 and 1.

While multiplying the state of a Cbit by 1 (i.e. leaving it alone) makes sense, you may well ask what it means to multiply its state by 0. (It *does not mean* removing the Cbit from the computer!)

The answer is that we shall never use $\mathbf{n}$ by itself but only in combination with other operations whose combined effect on the states of Cbits does have a straightforward physical meaning.

It is also convenient to define the *complementary* operator,

$$\overline{\mathbf{n}} = \mathbf{1} - \mathbf{n}, \tag{1.19}$$

so that $|0\rangle$ and $|1\rangle$are eigenvectors of $\mathbf{n}$ with eigenvalues 1 and 0.

These two operators have the matrix representations

$$\mathbf{n} \longleftrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad \overline{\mathbf{n}} \longleftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{1.20}$$

It follows directly from their definitions that

Identity

$$\mathbf{n}^2 = \mathbf{n}, \quad \overline{\mathbf{n}}^2 = \overline{\mathbf{n}}, \quad \mathbf{n}\overline{\mathbf{n}} = \overline{\mathbf{n}}\mathbf{n} = \mathbf{0}, \quad \mathbf{n} + \overline{\mathbf{n}} = \mathbf{1}. \tag{1.21}$$

We also have

$$\mathbf{nX} = \mathbf{X}\bar{\mathbf{n}}, \quad \bar{\mathbf{n}}\mathbf{X} = \mathbf{X}\mathbf{n}, \tag{1.22}$$

since flipping the state of a Cbit and then acting on it with $\bar{\mathbf{n}}(\mathbf{n})$ is the same as acting on the state with $\bar{\mathbf{n}}(\mathbf{n})$ and then flipping it — try it!

To complete this collection of trivial identities, note that

$$\mathbf{X}^2 = \mathbf{1}: \tag{1.23}$$

the NOT operator is its own inverse.

All the simple relations (1.21)-(1.23) also follow, as they must, from the matrix representations (1.16) and (1.20) for $\mathbf{X}$, $\mathbf{n}$, and $\bar{\mathbf{n}}$.

The possibilities for reversible operations get richer when we go from a single Cbit to a pair of Cbits.

One important operation you can perform on a pair of Cbits is the swap (or exchange) operation **S**, which simply interchanges the states of the two.

Since **S** acts as the identity if the states of the Cbits are the same, and it flips both Cbits if their states are different, it can be written as

$$\mathbf{S} = \mathbf{n} \otimes \mathbf{n} + \overline{\mathbf{n}} \otimes \overline{\mathbf{n}} + (\mathbf{X} \otimes \mathbf{X})(\mathbf{n} \otimes \overline{\mathbf{n}}) + (\mathbf{X} \otimes \mathbf{X})(\overline{\mathbf{n}} \otimes \mathbf{n}). \tag{1.24}$$

As you can see, I am moving back and forth without comment between two levels of description: (1) physical operations on physical Cbits, and (2) linear operators acting on the vectors that represent the states of those Cbits.

In doing so I am also blurring the distinction between states (of Cbits) and the vectors that represent those states and between operations (on Cbits) and operators acting on the vectors that represent their states.

Indeed, in many contexts "state" and "vector" are taken to be synonymous terms, a practice we shall sometimes follow.

Here the tensor product $\otimes$ of two 1-bit operators is the 2-bit operator that acts on the left Cbit with the operator on the left of $\otimes$ and the right Cbit with the operator on the right; i.e.

$$(\mathbf{a} \otimes \mathbf{b})\ |x\rangle \otimes |y\rangle = \mathbf{a}|x\rangle \otimes \mathbf{b}|y\rangle, \tag{1.25}$$

Note that we call operations that act jointly on n Cbits n-bit operators, rather than n-Cbit operators, because we shall be extending such operators to act on Qbits as well as Cbits.

It then follows that

$$(\mathbf{a} \otimes \mathbf{b})(\mathbf{c} \otimes \mathbf{d}) = (\mathbf{ac}) \otimes (\mathbf{bd}). \tag{1.26}$$

FYI: While I assume you are familiar with the basic concepts of linear algebra, I shall occasionally (but unsystematically) remind you of some of them, partly because the notation or terminology you have learned may differ from that used in quantum computing.

$$\mathbf{S} = \mathbf{n} \otimes \mathbf{n} + \overline{\mathbf{n}} \otimes \overline{\mathbf{n}} + (\mathbf{X} \otimes \mathbf{X})(\mathbf{n} \otimes \overline{\mathbf{n}}) + (\mathbf{X} \otimes \mathbf{X})(\overline{\mathbf{n}} \otimes \mathbf{n}). \tag{1.24}$$

At the risk of belaboring the obvious, I note that (1.24) acts as the swap operator because if both Cbits are in the state $|1\rangle$ (so swapping their states does nothing) then only the first term in the sum acts (and multiplies the state by 1 i.e., each of the other three terms gives 0), if both Cbits are in the state $|0\rangle$ only the second term acts (and again multiplies the state by 1), if the left Cbit is in the state $|1\rangle$ and the right Cbit is in the state $|0\rangle$ only the third term acts and the effect of flipping both Cbits is to swap their states, and if the left Cbit is in the state $|0\rangle$ and the right Cbit is in the state $|1\rangle$, only the fourth term acts and the effect of the two $\mathbf{X}$'s is again to swap their states.

The tensor product notation for operators can become quite horrible when one is dealing with a large number of Cbits and want to write a 2-bit operator that affects only a particular pair of Cbits.

$$(\mathbf{a} \otimes \mathbf{b})\ |x\rangle \otimes |y\rangle = \mathbf{a}|x\rangle \otimes \mathbf{b}|y\rangle, \tag{1.25}$$

If, for example, the 2-bit operator in (1.25) acts only on the second and fourth Cbits from the right in a 6-Cbit state, then the operator on the 6-Cbit state has to be written as

$$\mathbf{1} \otimes \mathbf{1} \otimes \mathbf{a} \otimes \mathbf{1} \otimes \mathbf{b} \otimes \mathbf{1}. \tag{1.27}$$

To avoid such typographical monstrosities we simplify (1.27) to

$$\mathbf{1} \otimes \mathbf{1} \otimes \mathbf{a} \otimes \mathbf{1} \otimes \mathbf{b} \otimes \mathbf{1} = \mathbf{a}_3\mathbf{b}_1 = \mathbf{b}_1\mathbf{a}_3, \tag{1.28}$$

where the subscript indicates which Cbit the 1-bit operator acts on, and it is understood that those Cbits whose subscripts do not appear remain unmodified — i.e., they are acted on by the unit operator.

We label each Cbit by the power of 2 it would represent if the Cbits were representing an integer; i.e. the Cbit on the extreme right is labeled 0, the one to its left, 1, etc.

Since the order in which a and b are written is clearly immaterial if their subscripts specify different Cbits, the order in which one writes them in (1.28) doesn't matter: 1-bit operators that act on different Cbits commute.

So with this convention, if the swap operator **S** acts on Cbits i and j, we can rewrite (1.24) as

$$\mathbf{S} = \mathbf{n}\otimes\mathbf{n} + \bar{\mathbf{n}}\otimes\bar{\mathbf{n}} + (\mathbf{X}\otimes\mathbf{X})(\mathbf{n}\otimes\bar{\mathbf{n}}) + (\mathbf{X}\otimes\mathbf{X})(\bar{\mathbf{n}}\otimes\mathbf{n}). \qquad (1.24)$$

$$\mathbf{S}_{ij} = \mathbf{n}_i\mathbf{n}_j + \bar{\mathbf{n}}_i\bar{\mathbf{n}}_j + (\mathbf{X}_i\mathbf{X}_j)(\mathbf{n}_i\bar{\mathbf{n}}_j + \bar{\mathbf{n}}_i\mathbf{n}_j). \qquad (1.29)$$

Sometimes we deal with 1-bit operators that have subscripts in their names; under such conditions it is more convenient to indicate which Cbit the operator acts on by a superscript, which I shall enclose in parentheses to avoid confusion with an exponent: thus $\mathbf{X}^{(2)}$ represents the 1-bit operator that flips the third (2,1,0) Cbit from the right, but $\mathbf{X}^2$ represents the square of the flip operator (i.e. the unit operator) without reference to which Cbit it acts on.

As a second illustration of this way of expressing operations on Cbits, consider the *controlled-not* or cNOT operation, $\mathbf{C}_{ij}$, which turns out to be the great workhorse of two-bit operations in quantum computation.

If the value represented by the i-th Cbit (the control bit) is 0, $\mathbf{C}_{ij}$ leaves the value represented by the j-th Cbit (the target bit) unchanged, but if the control bit is 1, $\mathbf{C}_{ij}$ flips the target bit.

In either case the control bit is left unchanged.

We can summarize this compactly by writing

$$\mathbf{C}_{10}|x\rangle|y\rangle = |x\rangle|y \oplus x\rangle, \tag{1.30}$$

where $\oplus$ denotes addition modulo 2:

$$y \oplus 0 = y, \quad y \oplus 1 = \overline{y} = 1 - y. \tag{1.31}$$

Computer scientists like to call x⊕y the "exclusive OR" (or XOR) of x and y.

You can build up a swap out of three cNOT operations:

$$\mathbf{S}_{ij} = \mathbf{C}_{ij}\mathbf{C}_{j\,i}\mathbf{C}_{ij}. \tag{1.32}$$

This can easily be verified by repeated applications of (1.30).

It can also be demonstrated using a more algebraic approach.

Note first that **C** can be expressed in terms of **n**'s and **X**'s by

$$\mathbf{C}_{ij} = \bar{\mathbf{n}}_i + \mathbf{X}_j \mathbf{n}_i, \tag{1.33}$$

since if the state of Cbit *i* is $|0\rangle$ only the first term acts which leaves the states of both Cbits unchanged, but if the state of Cbit *i* is $|1\rangle$only the second term acts, which leaves the state of Cbit 1 unchanged, while $\mathbf{X}_j$ flips Cbit j.

If you substitute expressions of the form (1.33) for each of the three terms in (1.32), then with the help of (1.21)-(1.23) (and the fact that operators with different subscripts commute) you can show that four of the eight terms into which the products expand vanish and the remaining four can be rearranged to give the swap operator in the form (1.29).

A curious symmetry of the cNOT operator is revealed if we define the operator

$$\mathbf{Z} = \bar{\mathbf{n}} - \mathbf{n} \longleftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.34}$$

This is a rather peculiar operator, since applied to $|1\rangle$ it produces $-|1\rangle$and it is not clear what it means to multiply the vector representing the state of a Cbit by -1 (or any number other than 1).

Do not worry ……

We shall see that in quantum computation **Z** plays as important a role as the NOT operator **X**, but for now we are *only going to use it* combined with other operators to produce perfectly *sensible* transformations on Cbits.

It follows from (1.22) (or from the matrix representations (1.16) and (1.34)) that the NOT operator **X** *anticommutes* with **Z**:

$$\mathbf{ZX} = -\mathbf{XZ}. \tag{1.35}$$

Since $\bar{\mathbf{n}} + \mathbf{n} = \mathbf{1}$ we can use (1.34) to express $\bar{\mathbf{n}}$ and **n** in terms of **1** and **Z**:

$$\mathbf{n} = \tfrac{1}{2}(\mathbf{1} - \mathbf{Z}), \quad \bar{\mathbf{n}} = \tfrac{1}{2}(\mathbf{1} + \mathbf{Z}). \tag{1.36}$$

Using this we can then rewrite the cNOT operator (1.33) in terms of **X** and **Z** operators:

$$\begin{aligned} \mathbf{C}_{ij} &= \tfrac{1}{2}(\mathbf{1} + \mathbf{Z}_i) + \tfrac{1}{2}\mathbf{X}_j(\mathbf{1} - \mathbf{Z}_i) \\ &= \tfrac{1}{2}(\mathbf{1} + \mathbf{X}_j) + \tfrac{1}{2}\mathbf{Z}_i(\mathbf{1} - \mathbf{X}_j). \end{aligned} \tag{1.37}$$

The second form follows from the fact that $\mathbf{X}_j$ and $\mathbf{Z}_i$ commute when $i \neq j$.

Note that if we were to interchange **X** and **Z** in the second line of (1.37) we would get back the expression directly above it except for the interchange of i and j.

So interchanging the **X** and **Z** operators has the effect of switching which bit is the control and which is the target, changing $\mathbf{C}_{ij}$ into $\mathbf{C}_{ji}$.

An operator that can produce just this effect is the *Hadamard transformation* (also sometimes called the *Walsh-Hadamard transformation*.),

$$\mathbf{H} = \tfrac{1}{\sqrt{2}}(\mathbf{X}+\mathbf{Z}) \longleftrightarrow \tfrac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{1.38}$$

This is another great work-horse of quantum computation.

Physicists should note here an unfortunate clash between the notations of computer science and physics.

Quantum physicists invariably use *H* to denote the Hamiltonian function (in classical mechanics) or Hamiltonian operator (in quantum mechanics).

Fortunately Hamiltonian operators, although of crucial importance in the design of quantum computers, play a very limited role in the general theory of quantum computation, being overshadowed by the unitary transformations that they generate.

So we can go along with the computer-science notation without getting into serious trouble.

Since $\mathbf{X}^2 = \mathbf{Z}^2 = \mathbf{1}$ and $\mathbf{XZ} = -\mathbf{ZX}$ one easily shows from the definition (1.38) of $\mathbf{H}$ in terms of $\mathbf{X}$ and $\mathbf{Z}$ that

$$\mathbf{H}^2 = \mathbf{1} \tag{1.39}$$

and that

$$\mathbf{HXH} = \mathbf{Z}, \quad \mathbf{HZH} = \mathbf{X}. \tag{1.40}$$

This also follows, of course, from the matrix representation (1.38) of $\mathbf{H}$.

This shows how **H** can be used to interchange the **X** and **Z** operators in $\mathbf{C}_{ji}$: it follows from (1.40), together with (1.37) and (1.39), that

$$\mathbf{C}_{ji} = (\mathbf{H}_i\mathbf{H}_j)\mathbf{C}_{ij}(\mathbf{H}_i\mathbf{H}_j). \tag{1.41}$$

This simple relation can be put to some quite remarkable uses in a quantum computer, as we shall see.

Of course the action of **H** on the state of a Cbit that follows from (1.38),

$$\mathbf{H}|0\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad \mathbf{H}|1\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \tag{1.42}$$

makes no sense at all.

Nevertheless, when combined with other operations, as on the right side of (1.41), the Hadamard operations result in the perfectly sensible operation given on the left side.

In a quantum computer the action of **H** on 1-Qbit states turns out to be not only meaningful but also easily implemented, and the possibility of interchanging control and target Qbits in the manner shown in (1.41) turns out to have important consequences.

The most general reversible operation on two Cbits is any permutation of their 4 possible states.

There are 4! = 24 such operations.

There are $(2^n)!$ distinct reversible operations on n Cbits, given by all possible permutations **P** of their 2^n states.

If you would like a matrix(I would NOT) for the cNOT operation in the 4-dimensional 2-Cbit subspace note that if the control bit is given by the entry on left then cNOT leaves $|00\rangle = |0\rangle_2$ and $|01\rangle = |1\rangle_2$ fixed and exchanges $|10\rangle = |2\rangle_2$ and $|11\rangle = |3\rangle_2$.

Therefore the $4 \otimes 4$ matrix is just

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \qquad (1.43)$$

If the control bit is shown on the right then $|01\rangle = |1\rangle_2$ and $|11\rangle = |3\rangle_2$ are interchanged and the matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{1.44}$$

By the same token, since the swap operator **S** interchanges $|01\rangle = |1\rangle_2$ and $|10\rangle = |2\rangle_2$, leaving $|00\rangle = |0\rangle_2$ and $|11\rangle = |3\rangle_2$ fixed, its matrix is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1.45}$$

Note that the construction (1.32) of **S** out of cNOT operators follows from (1.43)-(1.45) using matrix multiplication.

Messy and confusing I hope you thought!!!

As a practical matter, however, it is almost always more efficient to establish operator identities by dealing with them directly as operators (as in (1.32) in two different ways), avoiding matrix representations.

Before turning to the quantum generalization of Cbits, let me show you an alternative form for the swap operator (1.29), as a further exercise in treating operations on classical bits as linear operations on vectors.

This new form has a rather more elegant structure than (1.29), though you may well regard it as an utterly perverse way to treat Cbits.

If we use (1.36) to reexpress each $\mathbf{n}$ and $\bar{\mathbf{n}}$ appearing in the swap operator (1.29) in terms of $\mathbf{Z}$, we find that

$$\mathbf{S}_{ij} = \tfrac{1}{2}(\mathbf{1} + \mathbf{Z}_i\mathbf{Z}_j) + \tfrac{1}{2}(\mathbf{X}_i\mathbf{X}_j)(\mathbf{1} - \mathbf{Z}_i\mathbf{Z}_j). \tag{1.46}$$

If we define

$$\mathbf{Y} = \mathbf{ZX} = -\mathbf{XZ} \longleftrightarrow \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \tag{1.47}$$

we get the more compact form

$$\mathbf{S}_{ij} = \tfrac{1}{2}(\mathbf{1} + \mathbf{X}_i\mathbf{X}_j - \mathbf{Y}_i\mathbf{Y}_j + \mathbf{Z}_i\mathbf{Z}_j). \tag{1.48}$$

We can get rid of the irritating minus sign that mars the elegance of (1.48) by replacing $\mathbf{Y}$ with $-i\mathbf{Y}$.

If you wonder what it means to multiply the vector representing a Cbit by $i = \sqrt{-1}$, I merely remark at this point that it is no more or less mysterious than multiplying it by -1.

And like -1, the i drops out of the swap operator itself.

We also give $\mathbf{X}$, $\mathbf{Z}$, and $-i\mathbf{Y}$ new names, adopting a more cumbersome notation much beloved by physicists:

$$\sigma_x = \mathbf{X} \longleftrightarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \ \sigma_y = -i\mathbf{Y} \longleftrightarrow \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \ \sigma_z = \mathbf{Z} \longleftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \quad (1.49)$$

The swap operator then becomes

$$\mathbf{S}_{ij} = \tfrac{1}{2}(1 + \sigma_x^{(i)}\sigma_x^{(j)} + \sigma_y^{(i)}\sigma_y^{(j)} + \sigma_z^{(i)}\sigma_z^{(j)}). \quad (1.50)$$

Physicists might be amused at the simplicity of this “computational” derivation of the form of the exchange operator, compared with the conventional quantum mechanical derivation, which invokes the full apparatus of angular momentum theory.

The pleasing symmetry brought about by introducing a factor of i in the definition of σ_y is not limited to (1.50).

In addition the three σ matrices all square to unity,

$$\sigma_x^2 = \sigma_y^2 = \sigma_z^2 = 1, \tag{1.51}$$

they all anticommute in pairs, and the product of any two of them is simply related to the third:

$$\begin{aligned}
\sigma_x\sigma_y &= -\sigma_y\sigma_x = i\sigma_z, \\
\sigma_y\sigma_z &= -\sigma_z\sigma_y = i\sigma_x, \\
\sigma_z\sigma_x &= -\sigma_x\sigma_z = i\sigma_y.
\end{aligned} \tag{1.52}$$

Note that the three relations (1.52) differ only by cyclic permutations of x, y, and z.

All the relations (1.51) and (1.52) can be summarized in a single compact identity.

Let **a** and **b** be two three-dimensional vectors,

$$\mathbf{a} = (a_x, a_y, a_z), \quad \mathbf{b} = (b_x, b_y, b_z), \tag{1.53}$$

and let σ be a formal vector whose three components are the operators σ_x, σ_y, and σ_z,

$$\sigma = (\sigma_x, \sigma_y, \sigma_z), \tag{1.54}$$

so that, for example by **a**·σ we mean the ordinary inner product (or "dot product") of the two vectors **a** and σ:

$$\mathbf{a} \cdot \sigma = a_x \sigma_x + a_y \sigma_y + a_z \sigma_z. \tag{1.55}$$

Then all the relations (1.51) and (1.52) imply and are implied by the single identity

$$(\mathbf{a} \cdot \sigma)(\mathbf{b} \cdot \sigma) = \mathbf{a} \cdot \mathbf{b} + i(\mathbf{a} \times \mathbf{b}) \cdot \sigma, \tag{1.56}$$

where **a** × **b** denotes the vector product (or "cross product") of **a** and **b**.

By building the i into the definition of σ_y we have also made it hermitian, like σ_x and σ_z.

Recall that the elements of a hermitian matrix A satisfy $A_{ji} = A^*_{ij}$, where $*$ denotes complex conjugation.

Together with the unit matrix **1**, the matrices σ_x, σ_y and σ_z form a *basis* for the 4-dimensional algebra of 2-dimensional matrices of complex numbers: any such matrix is a unique linear combination of these four with complex coefficients.

Because the four are all hermitian, any 2-dimensional hermitian matrix A of complex numbers must be a *real* linear combination of the four, and therefore of the form.

$$A = a_0 + \mathbf{a} \cdot \sigma, \tag{1.57}$$

where a_0 and the components of the vector $\mathbf{a}$ are all real numbers.

Unless it is important to emphasize the operator structure, when a_0 is an scalar we shall write expressions such as $a_0\mathbf{1}$ as simply a_0.

The matrices σ_x, σ_y, and σ_z were introduced in the early days of quantum mechanics by Wolfgang Pauli, to describe the angular momentum associated with the spin of an electron.

They have many other useful purposes to deal efficiently with the composition of three-dimensional rotations.

It is pleasing (if somewhat perverse) to find them here, buried in the interior of the operator that simply swaps two classical bits.

We shall have extensive occasion to use Pauli's one-bit operators when we come to the subject of quantum error correction, and some of their properties, developed later further in sections A2 and A3 in this part, can be useful in treating other features of Qbits, to which we now turn.

C. Qbits and their states.

The state of a Cbit is a pretty miserable specimen of a two-dimensional vector, since the only vectors with any classical meaning in the whole two-dimensional vector space are the two orthonormal vectors $|0\rangle$ and $|1\rangle$.

Putting it another way, each Cbit can have only two states: $|0\rangle$ and $|1\rangle$.

Happily, nature has provided us with physical systems, Qbits, described by states that do not suffer from this limitation.

The state $|\psi\rangle$ associated with a Qbit can be any unit vector in the two-dimensional space spanned by $|0\rangle$ and $|1\rangle$.

Since we have already seen a hint of the elegance the use of $i = \sqrt{-1}$ can introduce even into the tightly constrained world of Cbits, you will, I hope, be pleased to learn that the scalars in the two-dimensional vector space containing the states of a Qbit are complex numbers.

The general state of a Qbit is

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \longleftrightarrow \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}, \tag{1.58}$$

where α_0 and α_1 are two complex numbers constrained only by the requirement that $|\psi\rangle$, like $|0\rangle$and $|1\rangle$, should be a unit vector in the complex vector space — i.e., only by the normalization condition

$$|\alpha_0|^2 + |\alpha_1|^2 = 1. \tag{1.59}$$

One says that the state $|\psi\rangle$ is a *superposition* of the states $|0\rangle$ and $|1\rangle$ with amplitudes α_0 and α_1.

If one of α_0 and α_1 is 0 and the other is 1 — i.e., in the special case where the state of the Qbit is one of the two classical states $|0\rangle$ or $|1\rangle$— it is often convenient to retain the language appropriate to Cbits, speaking of the Qbit "having the value" 0 or 1.

More *correctly*, however, one is only entitled to say that the state of the Qbit is $|0\rangle$ or $|1\rangle$.

Qbits, in contrast to Cbits, *cannot* be said to "have values".

They only have — or, more correctly, are described by— *states*.

We shall often sacrifice correctness for ease of expression.

Some reasons for this apparently pedantic terminological hair splitting will emerge below.

Just as the general state of a single Qbit is an arbitrary normalized superposition (1.58) of the two possible classical states, the general state $|\Psi\rangle$ that nature allows us to associate with two Qbits is an arbitrary normalized superposition of the four orthogonal classical states,

$$|\Psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \longleftrightarrow \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}, \tag{1.60}$$

with the complex amplitudes being constrained only by the normalization condition

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1. \tag{1.61}$$

This generalizes in the obvious way to n Qbits, whose general state can be a superposition of the 2^n different classical states, with amplitudes whose squared magnitudes sum to unity

$$|\Psi\rangle = \sum_{0 \le x < 2^n} \alpha_x |x\rangle_n, \tag{1.62}$$

$$\sum_{0 \le x < 2^n} |\alpha_x|^2 = 1. \tag{1.63}$$

In the context of quantum computation, the 2^n classical states — the set of all possible products of individual Qbit states $|0\rangle$ and $|1\rangle$ — is called the *computational basis*.

For many purposes classical basis would be a more appropriate term, and I shall use the two terms interchangeably.

The states that characterize n Cbits — the classical-basis states — are an extremely limited subset of the states of n Qbits, which can be any (normalized) superposition with complex coefficients of these classical-basis states.

If we have two Qbits, one in the state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ and the other in the state $|\phi\rangle = \beta_0|0\rangle + \beta_1|1\rangle$, then the state of the pair, in straightforward generalization of the rule for multi-Cbit states, is taken to be the *tensor* product of the individual states,

$$|\Psi\rangle = |\psi\rangle \otimes |\phi\rangle = \big(\alpha_0|0\rangle + \alpha_1|1\rangle\big) \otimes \big(\beta_0|0\rangle + \beta_1|1\rangle\big)$$

$$= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle \longleftrightarrow \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}. \qquad (1.64)$$

Note that a *general* 2-Qbit state (1.60) is of the special form (1.64) if and only if $\alpha_{00}\alpha_{11} = \alpha_{01}\alpha_{10}$. $\begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}$

Since the four amplitudes in (1.60) are constrained only by the normalization condition (1.61), this relation need not hold, and the general 2-Qbit state, unlike the general state of two Cbits, is not a product (1.64) of two 1-Qbit states.

The same is true for states of n Qbits: unlike Cbits, whose general state can only be one of the 2^n products of $|0\rangle$'s and $|1\rangle$'s, a general state of n Qbits is a superposition of these 2^n product states which will not, in general, be any product of any set of 1-Qbit states.

Individual Qbits making up a multi-Qbit system, in contrast to individual Cbits, cannot always be characterized as having individual states of their own.

More precisely they do not always have what are called pure states of their own.

It is often convenient to give a statistical characterization of an individual Qbit in terms of what is called a *density matrix* or *mixed state*.

If one wishes to emphasize that one is not talking about a mixed state, one uses the term "pure state".

In this exposition of quantum computation the term "state" can always be taken to mean "pure state" unless explicitly noted otherwise.

Such nonproduct states of two or more Qbits are called *entangled* states(*a superposition of correlations*).

Same as I showed in QM class

When the state of several Qbits is entangled, they can sometimes behave in some very strange ways.

An example of such peculiar behavior is discussed later in this part.

D. Reversible Operations on Qbits.

The only nontrivial reversible operation a classical computer can perform on a single Cbit is the NOT operation **X**.

Nature has been far more versatile in what it allows us to do to a Qbit.

The reversible operations that a quantum computer can perform upon a single Qbit are represented by the action on the state of the Qbit of *any linear* transformation that takes unit vectors into unit vectors.

Such transformations **u**, of which **X** is a special case, are called *unitary* and satisfy the condition

$$\mathbf{u}\mathbf{u}^\dagger = \mathbf{u}^\dagger\mathbf{u} = 1. \qquad (1.65)$$

Since any unitary transformation has a unitary *inverse*, such actions of a quantum computer on a Qbit are *reversible*.

The importance of reversibility for the effective functioning of a quantum computer will emerge in Part 2.

Such unitary operations on a Qbit are called **1-Qbit gates**.

Their action is represented schematically as in Figure 1.1.

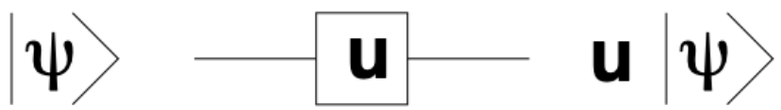


Figure 1.1. Circuit diagram representing the action on a single Qbit of the 1-Qbit gate **u**. Initially the Qbit is described by the input state $|\psi\rangle$ on the left. The thin line (wire) represents the subsequent history of the Qbit. After emerging from the box representing **u** the Qbit is described on the right by the final state $\mathbf{u}|\psi\rangle$.

The most general reversible n-Cbit operation in a classical computer is a permutation of the $(2^n)!$ different classical-basis states.

The most general reversible operation that a quantum computer can perform upon n-Qbits is represented by the action on the state of those Qbits of any linear transformation that takes unit vectors into unit vectors — i.e. any 2^n-dimensional unitary transformation **U**, satisfying

$$\mathbf{U}\mathbf{U}^\dagger = \mathbf{U}^\dagger\mathbf{U} = 1. \qquad (1.66)$$

Such transformations are called n-Qbit gates.

The action of an n-Qbit unitary gate on n Qbits is represented schematically by the kind of diagram shown in Figure 1.2.



Figure 1.2. Circuit diagram representing the action on n Qbits of the n-Qbit gate **U**. Initially the Qbits ares described by the input state $|\Psi\rangle$ on the left. The thick line (bar) represents the subsequent history of the Qbits. After emerging from the box representing **U** the Qbits are described on the right by the final state $\mathbf{U}|\Psi\rangle$

Figure 1.3 shows the schematic representation of two successive unitary transformations: first **V** is applied to the Qbits, and then **U**.

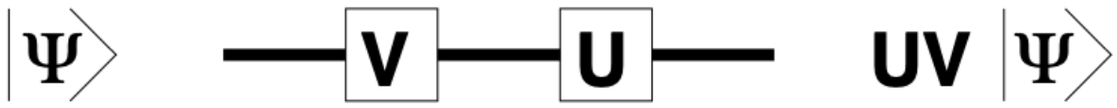


Figure 1.3. Circuit diagram representing the action on n Qbits of two n-Qbit gates. Initially the Qbits ares described by the input state $|\Psi\rangle$ on the left. The thick line (bar) represents the subsequent history of the Qbits. First the are acted upon by the gate **V**, and then, by the gate **U**, emerging on the right in the final state $\mathbf{UV}|\Psi\rangle$. Note that the order in which the Qbits encounter unitary gates in the figure is opposite to the order in which the corresponding unitary symbols are written in the symbol for the final state on the right.

It is the practice of computer scientists to read such circuit diagrams from left to right, like ordinary prose in European languages, so the the operator **V** occurs to the left of **U** because it acts on the Qbits before **U** acts.

The final state, however, is conventionally written as $\mathbf{U(V)}|\Psi\rangle = \mathbf{UV}|\Psi\rangle$, it being the practice of physicists to write the operators to the left of the state symbol.

Thus, as an unfortunate result of this intermingling of cultures, the left-to-right order in which operators appear in circuit diagrams is reversed from the left-to-right order in which they appear in the mathematical expression for the state that diagram gives rise to.

Any operation **P** that acts on n Cbits, can be associated with a corresponding unitary operation **U** on n Qbits.

One defines the action of **U** on the classical-basis states to be identical to the operation of **P** on the corresponding classical states.

Since the classical basis *is* a basis, **U** can then be extended to arbitrary n-Qbit states by *linearity*.

But since the action of **U** on the classical-basis states is merely to permute them, its effect on any superposition of such states is merely to permute the amplitudes.

So it trivially takes unit vectors (for which the sums of the moduli of the squared amplitudes is unity) into unit vectors for arbitrary quantum states.

Being norm-preserving and linear, it is indeed unitary.

Many important unitary operations on Qbits we shall be examining below are defined in this way, as linear extensions to Qbits of classical operations on Cbits.

But the available unitary transformations on Qbits are, of course, much more general than straightforward extensions of classical operations.

In the actual design of quantum algorithms the class of allowed unitary transformations is almost always restricted to ones that can be built up out of products of unitary transformations that act on only one Qbit at a time (*1-Qbit gates*) or a pair of Qbits (*2-Qbit gates*).

This is because the technical problems of making higher order quantum gates are even more formidable than the (already difficult) problems of constructing reliable 1-and 2-Qbit gates.

It turns out that this is not a fundamental limitation, since any unitary transformations can be approximated to an arbitrary degree of precision by sufficiently many 1- and 2-Qbit gates.

We shall not prove this general result, because all of the quantum algorithms we shall be developing will be explicitly built up entirely from 1- and 2-Qbit gates.

One very important illustration of the sufficiency of 1- and 2-Qbit gates will emerge in Part 2: *In a reversible classical computer it turns out that one needs 3-Cbit gates to build up general logical operations*.

But in a quantum computer we shall see that the quantum extensions of such 3-Cbit gates can be constructed out of a small number of 1- and 2-Qbit gates.

While unitarity is generally taken to be the hallmark of the transformations nature allows us to perform on quantum states, it is worth emphasizing that what is really remarkable about these transformations is their *linearity*, which is, of course, one aspect of their unitarity.

It is, for example, easy to dream up straightforward classical models for a Qbit, particularly if one restricts its superpositions to real linear combinations of the two classical states.

And it is not hard to invent classical models for NOT and even Hadamard 1-Qbit gates, for example, that act linearly on all the 1-Qbit states, while acting as NOT or Hadamard on the classical basis states.

But it is not at all easy to dream up a straightforward classical model for a cNOT gate, for example, that acts linearly on all the extended 2-Qbit states, while acting as cNOT on the four computational basis states.

It is remarkable and highly nontrivial that nature allows us, with much ingenuity and hard work, to fabricate such gates.

end 9

E. The measurement of Qbits.

Begin Class 10

To specify the state of a single Cbit you need only one bit of information: whether the state of the Cbit is $|0\rangle$ or $|1\rangle$.

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \longleftrightarrow \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}$$

But to specify the state (1.58) of a single Qbit to an arbitrarily high degree of precision you have to specify arbitrarily many bits of information, since you must specify two complex numbers α and β subject only to a normalization constraint (1.59).

Because Qbits not only have a much richer set of states than Cbits, but also can be acted on by a correspondingly richer set of transformations, it might appear obvious that a quantum computer would be vastly more powerful than a classical computer.

True. But there is a major catch!

The catch is this: if you have n Cbits, each represents either 0 or 1, and you can find out the state of each just by looking.

There is nothing problematic about learning the state of a Cbit, and hence learning the result of any calculation you may have built up out of operations on those Cbits.

Furthermore — and this is taken completely for granted in any discussion of a classical computer — the state of Cbits is *not* altered by the process of reading them.

The act of acquiring the information from Cbits is *not* disruptive.

You can read the Cbits at any stage of a computation without disrupting subsequent stages.

In stark contrast, if you have n Qbits in a superposition (1.62) of computational basis states, there is nothing whatever you can do to to them to extract from those Qbits the vast amount of information contained in the amplitudes α_x.

$$|\Psi\rangle = \sum_{0 \le x < 2^n} \alpha_x |x\rangle_n,$$

You cannot read out all those values and therefore you cannot find out what the state is.

The state of n Qbits is not associated with any determinable property of those Qbits, as it is for Cbits.

There is only **one way** to extract information from n Qbits in a given state: it is called *making a measurement*.

Physicists will note — others need pay no attention to this remark — that what follows is more accurately characterized as "making a (von Neumann) measurement in the computational (classical) basis."

There are other ways to make such a measurement, but they *can all be reduced* to measurements in the computational basis *provided* an appropriate unitary transformation is applied to the n-Qbit state of the computer *just before* carrying out the measurement.

So when I use the term "measurement", I shall always mean measurement in the computational basis.

Measurements in other bases will always be treated as measurements in the computational basis *preceded* by suitable unitary transformations to convert to the computational basis.

Making a measurement consists of performing a certain test on each Qbit, the outcome of which is either 0 or 1.

The particular collection of 0's and 1's produced by the test is not in general determined by the state $|\Psi\rangle$ of the Qbits; the state determines only the *probability* of the possible outcomes, according to the *following rule*:

> The probability of getting a particular result — say 01100, if you have 5 Qbits — is given by the squared magnitude of the amplitude of the state $|01100\rangle$ in the expansion of the state $|\Psi\rangle$ of the Qbits in the 2^5 computational basis states.

More generally, if the state of n Qbits is

$$|\Psi\rangle_n = \sum_{0 \le x < 2^n} \alpha_x |x\rangle_n, \tag{1.67}$$

then the probability that the 0's and 1's resulting from measurements of all the Qbits will give the *binary expansion of the integer x* is

$$p(x) = |\alpha_x|^2. \tag{1.68}$$

This basic rule for how information can be extracted from a quantum state was first enunciated by Max Born(in a paper footnote!), and is known as the *Born rule* (all of modern QM follows from Born rule).

It provides the link between amplitudes and the numbers you can actually read out when you test — i.e. measure — the Qbits.

The squared magnitudes of the amplitudes give the probabilities of outcomes of measurements.

Normalization conditions like (1.61) are just the requirements that the probabilities for all of the 2^n mutually exclusive outcomes *add* up to 1.

This process of measurement is carried out by a piece of hardware with a *digital* display known as an *n-Qbit measurement gate*.

Such an n-Qbit “measurement gate” is depicted schematically in Figure 1.4.

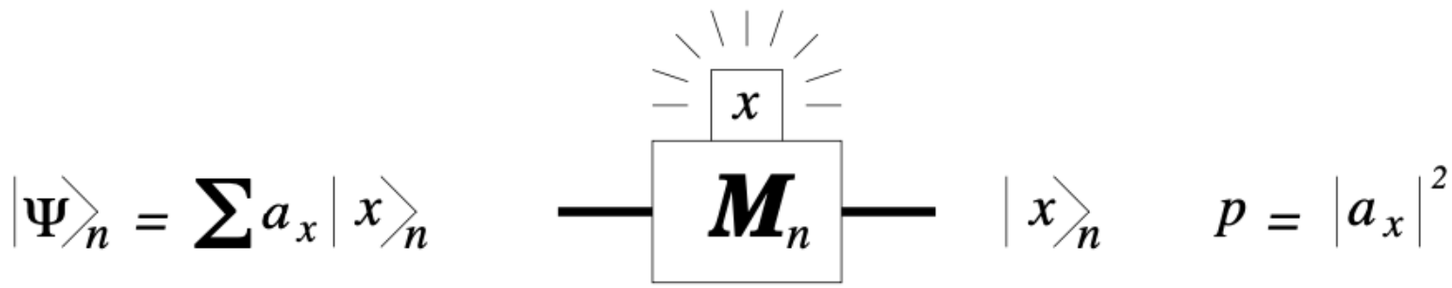


Figure 1.4. Circuit diagram representing an n-Qbit measurement gate. The Qbits are initially described by the n-Qbit state

$$|\Psi\rangle_n = \sum_{0 \le x < 2^n} a_x |x\rangle_n,$$

on the left. After the measurement gate $\mathbf{M}_n$ has acted, with probably $p = |a_x|^2$ it indicates an integer x, $0 \le x < 2^n$, and the Qbits are subsequently described by the state $|x\rangle_n$ on the right.

In contrast to unitary gates, which have a unique output state for each input state, the state of the Qbits emerging from a measurement gate is only *statistically* determined by the state of the input Qbits.

In further contrast to unitary gates, the action of a measurement gate *cannot* be undone: given the final state $|x\rangle$ there is no way of reconstructing the initial state $|\Psi\rangle$.

Nor is the action of a measurement gate in any sense linear.

To the extent that it suggests that some preexisting property is being revealed, "measurement" is a highly *misleading* term, but it is hallowed by three quarters of a century of use by quantum physicists, and impossible to avoid in treatments of quantum computation.

One should avoid being misled by such spurious connotations of "measurement" (though it confused many physicists in the early days of quantum mechanics).

In quantum computation "measurement" means nothing more or less than applying and reading the display of an appropriate measurement gate, whose action is fully specified by the Born rule, as described above (and expanded on below).

While measurement in quantum mechanics is not at all like measuring somebody's weight, it is a bit more like measuring somebody's IQ, which (I would say) reveals no pre-existing property of people, but only what happens when they are subject to the IQ test.

The simplest statement of the Born rule is for a single Qbit.

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \longleftrightarrow \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}$$

If the state of the Qbit is the superposition (1.58) of the states $|0\rangle$ and $|1\rangle$with amplitudes α_0 and α_1 then the result of the measurement is 0 with probability $|\alpha_0|^2$ and 1 with probability $|\alpha_1|^2$.

The measurement is carried out by a 1-Qbit measurement gate, as illustrated in Figure 1.5.

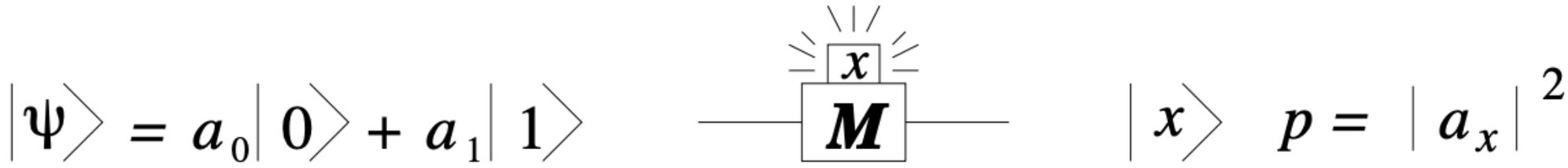


Figure 1.5. Special case of Figure 1.4: A 1-Qbit measurement gate. The reading x of the gate is either 0 or 1.

We shall see that n-Qbit measurement gates can be realized by applying 1-Qbit measurement gates to *each* of the nQbits - will see in later circuits.

The process of measurement can thus be *reduced* to applications of multiple copies of a single elementary piece of hardware: the 1-Qbit measurement gate.

In addition to displaying an n-bit integer with probabilities determined by the amplitudes, there is a second very important aspect of the action of measurement gates: If n Qbits, initially described by a state $|\Psi\rangle$, are sent through an n-Qbit measurement gate, and the display of the measurement gate indicates the integer x, then one must associate with the Qbits emerging from that measurement gate the classical-basis state $|x\rangle_n$ (as already shown in Figures 1.4 and 1.5). reduction

Among other things, this means that all traces of the amplitudes α_x characterizing the input state have vanished from the output state.

The only role they have played in the measurement is to determine the probability of a particular output.

If the state of the input Qbits is one of the classical-basis states $|x\rangle_n$, then according to the Born rule the probability is 1 that the measurement gate will read x and the output state will remain $|x\rangle_n$.

$$|\Psi\rangle_n = \sum_{0 \le x < 2^n} \alpha_x |x\rangle_n,$$

But for superpositions (1.67) with more than a single non-zero amplitude α_x, the output state is not determined, and, being a single one of the classical basis states $|x\rangle_n$, no longer carries any information about the amplitudes characterizing the initial state, other than certifying that the particular amplitude α_x was not zero, and, in all likelihood, was not exceedingly small.

So once you send n Qbits through an n-Qbit measurement gate, you *remove* the possibility of extracting any further information about their original state $|\Psi\rangle$.

After such a measurement of 5 Qbits, if the result is 01100, then the post-measurement state associated with the Qbits is no longer $|\Psi\rangle$, but $|01100\rangle$.

The original state $|\Psi\rangle$, and all the rich information potentially available in its amplitudes is *irretrievably* lost.

Qbits emerging from a measurement gate that indicates the outcome x are characterized by the state $|x\rangle$, *regardless* of what their pre-measurement state may have been.

This change of state attendant upon a measurement is often referred to as a *reduction* or *collapse* of the state.

One says that as a consequence of the measurement the pre-measurement state reduces or collapses to the post-measurement state.

This *should not* be taken to imply (though, alas, it often is) that the Qbits themselves suffer a catastrophic “reduction” or “collapse”.

It is important to keep in mind, in this context, that the state of n Qbits is *nothing more than an abstract* symbol, used, via the Born rule, to calculate probabilities of measurement outcomes.

As noted earlier, there is *no internal property* of the Qbits that corresponds to their state.

You might well wonder how one can learn anything at all of computational interest under these wretched conditions.

The artistry of quantum computation consists in producing, through a cunningly constructed unitary transformation, a superposition in which most of the amplitudes α_x are zero or extremely close to zero, with useful information being carried by any of the values of x that have an appreciable probability of being indicated by the measurement.

It is thus important to be seeking information which, once possessed, can easily be confirmed (e.g. the factors of a large number) so that one is not misled by rare and irrelevant low probability outcomes.

How this is actually accomplished in various cases of interest will be one of our major preoccupations.

There is a stronger version of the Born rule, which plays a very important role in quantum computation, even though it is rarely explicitly mentioned in most standard quantum mechanics texts.

We shall call it the *generalized* Born rule.

This stronger form applies when one measures only a single one of n Qbits, using a standard 1-Qbit measurement gate.

The generalized Born rule is based on the fact that any state $|\Psi\rangle$ of all n Qbits can be represented in the form

$$|\Psi\rangle = a_0|0\rangle|\Phi_0\rangle + a_1|1\rangle|\Phi_1\rangle, \qquad |a_0|^2 + |a_1|^2 = 1, \tag{1.69}$$

where the Qbit to be measured appears on the left, and where $|\Phi_0\rangle$ and $|\Phi_1\rangle$ are normalized (but not necessarily orthogonal) states of the n−1 *unmeasured* Qbits.

This follows straightforwardly from the fact that the general form (1.67) of $|\Psi\rangle$ is indeed of the form (1.69) with $|\Phi_0\rangle$ and $|\Phi_1\rangle$ given, to within normalization constants, by $|\Psi\rangle_n = \sum_{0 \le x < 2^n} \alpha_x |x\rangle_n,$

$$|\Phi_0\rangle \propto \sum_{x=0}^{2^{n-1}-1} \alpha_x |x\rangle_{n-1}, \quad |\Phi_1\rangle \propto \sum_{x=0}^{2^{n-1}-1} \alpha_{x+2^{n-1}} |x\rangle_{n-1}. \qquad (1.70)$$

(1 bit less)

The generalized Born rule asserts that if only the single Qbit associated with the one-Qbit state on the left of (1.69) is measured, then the 1-Qbit measurement gate will indicate x (0 or 1) with probability $|a_x|^2$, after which the n-Qbit state will then be the product state $|x\rangle|\Phi_x\rangle$.

$|\Psi\rangle = a_0|0\rangle|\Phi_0\rangle + a_1|1\rangle|\Phi_1\rangle$

This action of a 1-Qbit measurement gate on an n-Qbit state is depicted schematically in Figure 1.6.

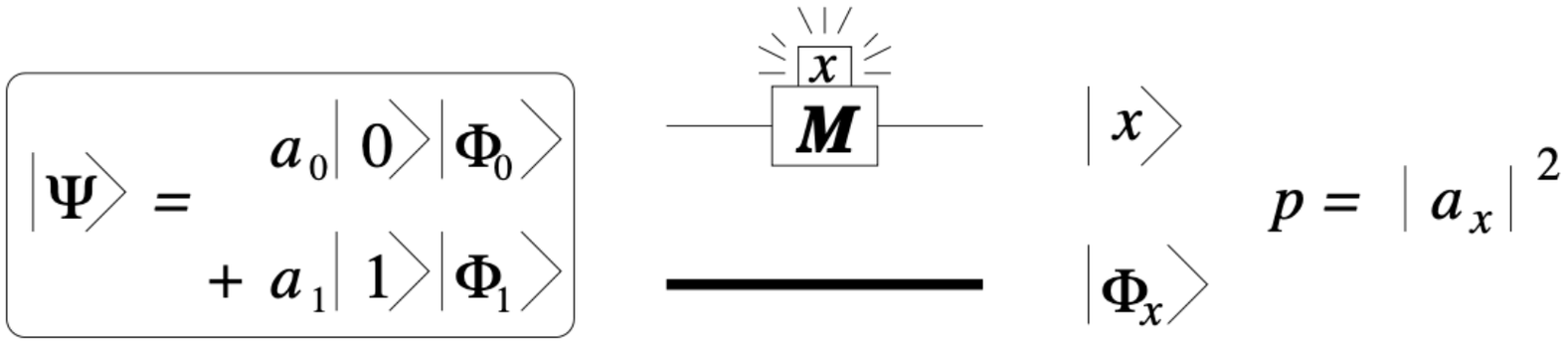


Figure 1.6. Action of a 1-Qbit measurement gate on a single one of n Qbits, according to the generalized Born rule. The initial state (on the left) is $|\Psi\rangle_n = a_0|0\rangle|\Phi_0\rangle_{n-1} + a_1|1\rangle|\Phi_1\rangle_{n-1}$.

Figure 1.7 (and its caption) demonstrates that if the Qbit on which the 1-Qbit gate acts is *initially disentangled* from the remaining n−1 Qbits, then the action of the gate on the measured Qbit is just that specified by the ordinary Born rule, and the unmeasured Qbits play *no role at all*, remaining throughout the process in their original state.

Only the single Qbit on the left is subject to a measurement gate.

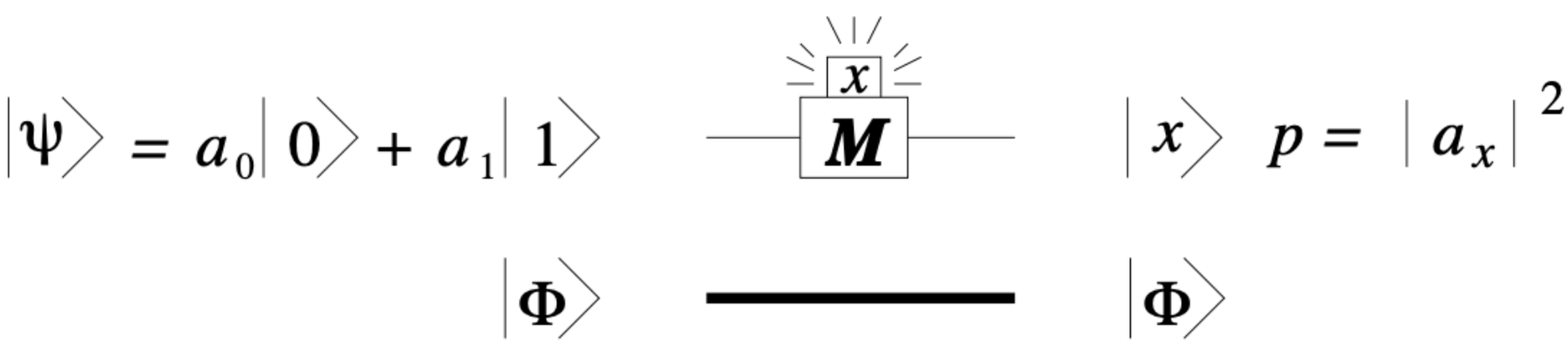


Figure 1.7. Simplification of Figure 1.6 when $|\Phi_0\rangle = |\Phi_1\rangle = |\Phi\rangle$. In this case the initial state on the right is just the product state $|\Psi\rangle_n = (a_o|0\rangle + a_1|1\rangle)|\Phi\rangle$, and the final state of the unmeasured Qbits continues to be $|\Phi\rangle$regardless of the value of x indicated by the 1-Qbit measurement gate. Thus the unmeasured Qbits play no role at all: they are unentangled with the measured Qbit and described by the state Φ throughout the process. The 1-Qbit measurement gate acts on the measured Qbit exactly as it does in Figure 1.5 when no other Qbits are present, thereby demonstrating that the generalized Born rule of Figure 1.6 reduces to the ordinary Born rule, when the measured Qbit is unentangled with the others.

This is also evident from the preceding paragraph, if one notes that it corresponds to the case in which the two not necessarily orthogonal states $|\Phi\rangle_0$ and $|\Phi\rangle_1$ are identical.

If one applies the generalized Born rule n times to successive one-Qbit measurements of each of n Qbits, initially in the general n-Qbit state (1.67), one can show (by a straight-forward but irritatingly awkward argument) that the final state of the n-Qbits is x with probability $|\alpha_x|^2$, where x is the *n-bit integer* whose bits are given by the readings on the n one-Qbit measurement gates.

This is nothing but the ordinary Born rule, with the n 1-Qbit measurement gates playing the role of the single n-Qbit measurement gate.

There is thus, as remarked upon above, *only one primitive piece* of measurement hardware: the 1-Qbit measurement gate.

This construction of an n-Qbit measurement gate out of n 1-Qbit measurement gates is depicted schematically in Figure 1.8.

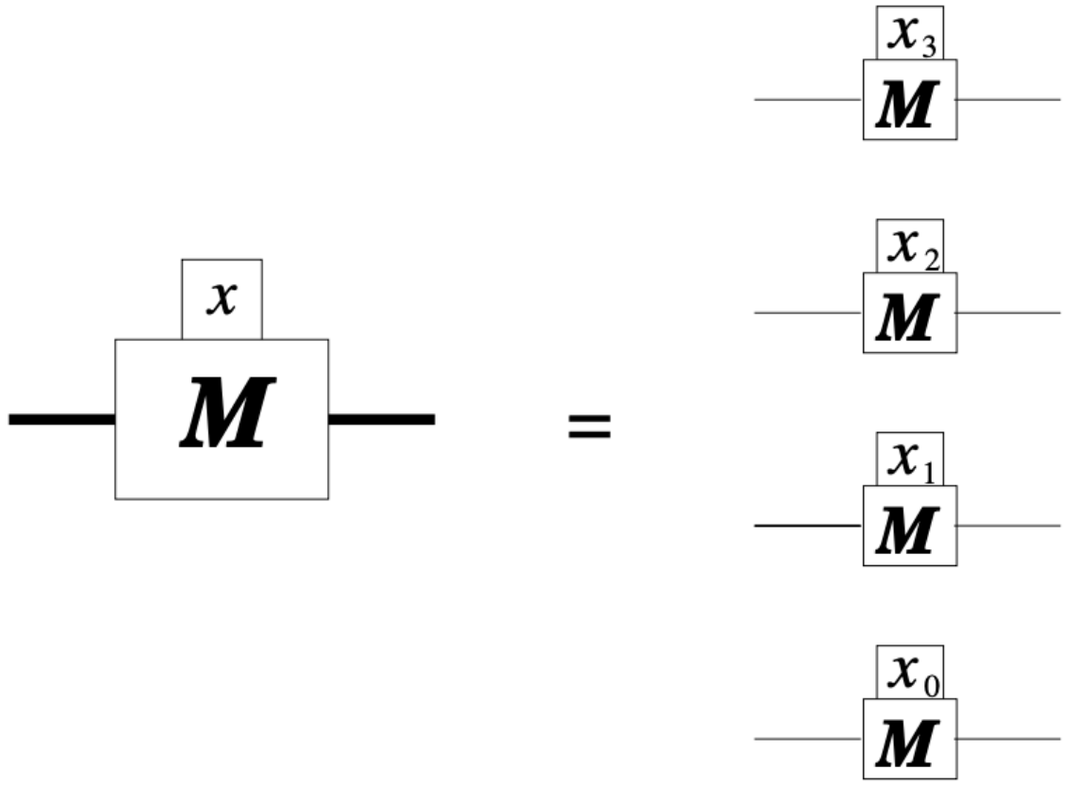


Figure 1.8. Constructing a 4-Qbit measurement gate out of four 1-Qbit measurement gates. The integer x has the binary expansion $x_3x_2x_1x_0$.

An even more *general* version of the Born rule follows from the generalized Born rule as stated above.

The general state of m + n Qbits can be written as

$$|\Psi\rangle_{m+n} = \sum_x \alpha_x |x\rangle_m |\Phi_x\rangle_n \qquad (1.71)$$

where $\sum_x |\alpha_x|^2 = 1$ and the states $|\Phi_x\rangle_n$ are normalized, but not necessarily orthogonal.

By applying the generalized Born rule m times to m Qbits in an m + n Qbit state, one establishes the rule that if only the m Qbits on the left of (1.71) are measured, then with probability $|\alpha_x|^2$ the result will be x, and after the measurement the state of all m + n Qbits, will be the product state

$$|x\rangle_m |\Phi_x\rangle_n \qquad (1.72)$$

in which the m measured Qbits are in the state $|x\rangle_m$ and the n unmeasured ones are in the state $|\Phi_x\rangle_n$.

It is important to note and immediately reject a possible misunderstanding of the Born rule.

One might be tempted to infer from that rule that for a Qbit to be in a superposition like the state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$ *means nothing more than* that the "actual state" of the Qbit is *either* $|0\rangle$with probability $|\alpha_0|^2$ *or* $|1\rangle$with probability $|\alpha_1|^2$.

Such an assertion goes *beyond* the rule, of course, which merely asserts that if one subjects a Qbit in the state $|\psi\rangle$ to an appropriate test — a measurement — then the outcome of the test will be 0 or 1 with those probabilities and the post-measurement state of the Qbit can correspondingly be taken to be $|0\rangle$ or $|1\rangle$.

This *does not imply that prior* to the test the Qbit already had the value — i.e. already was described by the classical-basis state — revealed by the test, since, among other possibilities, the action of the test itself might well play a role in bringing forth the outcome.

In fact it is easy to produce examples that demonstrate that the Qbit, prior to the test, could not have been in either one or the other of the states $|0\rangle$ or $|1\rangle$.

We can see this with the help of the Hadamard transformation (1.38).

We have defined the action of the 1-Qbit operators **H**, **X**, and **Z** only on the computational basis states $|0\rangle$ and $|1\rangle$, but we can extend their action to arbitrary linear combination of these states by *requiring* the extensions to be linear operators.

Since the states $|0\rangle$ and $|1\rangle$ form a basis, this determines the action of **H**, **X**, and **Z** on any 1-Qbit state.

Because it is hermitian and its own inverse, **H** is unitary, and is therefore the kind of operation a quantum computer can apply to the state of a Qbit: a 1-Qbit gate.

The result of the operation of a Hadamard gate is to change the state $|\phi\rangle$ of a Qbit to $\mathbf{H}|\phi\rangle$.

Suppose, now, that we apply **H** to a Qbit that is initially in the state

$$|\phi\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{1.73}$$

It follows from (1.42) that the result is just

$$\mathbf{H}|0\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad \mathbf{H}|1\rangle = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle), \tag{1.42}$$

$$\mathbf{H}|\phi\rangle = |0\rangle. \tag{1.74}$$

So according to the Born rule if we measure a Qbit described by the state $\mathbf{H}|\phi\rangle$the result will be 0 with probability 1.

But suppose that a Qbit in the state $|\phi\rangle$were indeed either in the state $|0\rangle$ with probability 1/2 *or* in the state $|1\rangle$ with probability 1/2.

In either case, according to (1.42), the subsequent action of **H** would produce a state — either $1/\sqrt{2}(|0\rangle + |1\rangle)$ or $1/\sqrt{2}(|0\rangle - |1\rangle)$ — that under measurement yielded 0 or 1 with equal probability.

This contradicts the fact just extracted directly from (1.74) that the result of making a measurement on a Qbit in the state $H|\phi\rangle$ is invariably 0.

So a Qbit in a quantum superposition of $|0\rangle$ and $|1\rangle$ *cannot be viewed* as being either in the state $|0\rangle$ or in the state $|1\rangle$ with certain probabilities.

Such a state represents something quite different from what either of these possibilities represent.

Although the Qbit only reveals a 0 or a 1 when you query it by means of a measurement, prior to such a query its state is not in general either $|0\rangle$ or $|1\rangle$, but a superposition of the form (1.58).

Such a superposition is as natural and irreducible a description of a Qbit as $|0\rangle$ and $|1\rangle$ are.

Shortly we will have further development of this point.

If the state (1.67) is one of the 2n computational-basis states $|x\rangle_n$ so that $\alpha_x = 1$ and $\alpha_y = 0$ when $y \neq x$, then the Born rule (1.68) reduces to the assertion that the outcome of the measurement is x with probability 1, and the post-measurement state is $|x\rangle$ — i.e. it is unchanged from the pre-measurement state.

$$|\Psi\rangle_n = \sum_{0 \leq x < 2^n} \alpha_x |x\rangle_n,$$

So if the states of n Qbits are restricted to computational basis states then the process of measurement is just like the classical process of "learning the value" of x without altering the state.

A quantum computer can be made to simulate a reversible classical computer by allowing only computational basis states as input, and only allowing unitary transformations that take computational basis states into computational basis states.

In addition to providing an output at the end of a computation, measurement gates also play a crucial role (not often sufficiently emphasized) at the beginning.

If there is no way to determine the state of a given collection of Qbits — indeed, in general such a collection might be entangled with other Qbits and therefore not even have a state of its own — then *how* can one produce a set of Qbits in a definite state for the gates of a quantum computer to transform into another computationally useful state?

The answer is by measurement.

If one takes n Qbits off the shelf, and subjects them to a measurement gate that registers x, then one can be *sure* that the Qbits emerging from that gate are in the classical-basis state $|x\rangle_n$.

If one then applies the 1-Qbit operation **X** to each Qbit that registered a 1 in the measurement, doing nothing to the Qbits that registered 0, the resulting set of Qbits will be described by the state $|0\rangle_n$, and it is this state that most quantum-computational algorithms take as their input.

Such a use of a measurement gate to provide a Qbit described by the state $|0\rangle$ is shown in Figure 1.9.

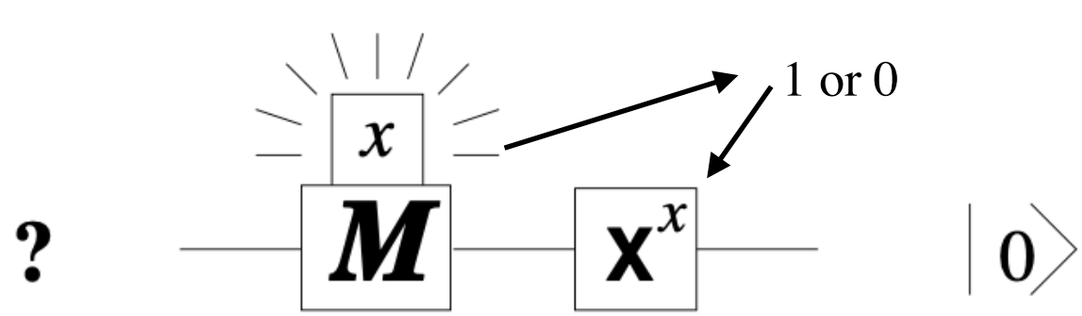


Figure 1.9. Using a 1-Qbit measurement gate to prepare an off-the-shelf Qbit so its associated state is $|0\rangle$. The input on the left is a Qbit in an unknown condition — i.e. nothing is known of its past history. After the measurement gate is applied, X is or is not applied, depending on whether the measurement gate indicates 1 or 0. The Qbit emerges (on the right) in the state $|0\rangle$.

Measurement gates therefore play *two* roles in a quantum computation.

They get the Qbits ready for the subsequent action of the computer, and after the computer has acted they extract from the Qbits a digital output.

The initial action of the measurement gates is often called “state preparation,” since the Qbits emerging from the process can be characterized by a definite state.

The association of unitary operators with the gates that subsequently act on the Qbits, permits one to update that state assignment into the corresponding unitary transformation of the initial state, thereby making it possible to calculate, using the Born rule, the probabilities of the outcomes of the final measurement gates.

Table: Cbits vs. Qbits

I conclude this offbeat introduction to quantum mechanics with a table that summarizes the relevant physical features of Qbits by contrasting them to the analogous features of Cbits.

In the table I introduce the term “Bit”, with an upper-case B, to mean “Qbit or Cbit” (as opposed to “bit”, with a lower-case b, which means “0 or 1”).

Alice (5th line) is anybody who knows the relevant history of the Qbits: the initial “state preparation” and the unitary gates that have subsequently acted.

CLASSICAL vs. QUANTUM BITS	**Cbits**	**Qbits**
States of n Bits	$\|x\rangle_n, \quad 0 \leq x < 2^n$	$\sum \alpha_x \|x\rangle_n, \quad \sum \|\alpha_x\|^2 = 1$
Subsets of n Bits	Always have states	Generally have no states
Reversible operations on states	Permutations	Unitary transformations
Can state be learnt from Bits?	Yes	No
To learn state of Bits	Examine them	Go ask Alice
To get information from Bits	Just look at them	Measure them
Information acquired	x	x with probability $\|\alpha_x\|^2$
State after information acquired	Same: still $\|x\rangle$	Different: now $\|x\rangle$

I now give some further mathematical developments of a slightly more technical nature, that we will make only occasional use of.

It also lets us practice using the formalism we have introduced.

Now we digress a bit to learn more mathematics and quantum mechanics before continuing!

Further Features of Dirac Notation (learning more mathematics)

The features of Dirac notation developed previously are the only ones needed to understand the manipulation of and extraction of information out of Qbits, and to understand what follows in the class.

There are, however, several mathematical embellishments of the notation which can be useful, which you will encounter if you delve into the literature on quantum computation and quantum information, and which will come up occasionally in this class.

There is an inner product between two n-Qbit states $|\phi\rangle$and $|\psi\rangle$, written in the form $\langle\phi|\psi\rangle$.

Dirac "braket"

In more conventional vector-space notation one would talk of an inner product between vectors ϕ and ψ, written in the form (ϕ,ψ).

Some mathematicians find it ludicrous that quantum physicists insist on drawing extra lines "|" and "$\rangle$" around all their vectors.

It is, however, no sillier than the once widespread practice of drawing little arrows ($\rightarrow$) above all vectors.

The application to Qbits illustrates the power of the notation, since what comes between the "|" and the "⟩" need not look like a vector at all, as in |01101⟩.

The inner product(braket) is a number satisfying the usual rules for inner products in a vector space with complex scalars:

$$\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*,$$

$$\text{if } |\chi\rangle = \alpha|\psi\rangle + \beta|\lambda\rangle \text{ then } \langle\phi|\chi\rangle = \alpha\langle\phi|\psi\rangle + \beta\langle\phi|\lambda\rangle,$$

$$\langle\phi|\phi\rangle > 0, \quad |\phi\rangle \neq 0. \tag{1.75}$$

Since the two 1-Qbit states |0⟩ and |1⟩ are orthogonal unit vectors, we have

$$\langle 0|0\rangle = \langle 1|1\rangle = 1, \quad \langle 1|0\rangle = \langle 0|1\rangle = 0. \tag{1.76}$$

For the n-Qbit computational basis states |x⟩ ($0 \leq x < 2^n$) we have

$$\langle x|y\rangle = 0, \ x \neq y; \quad = 1, \ x = y, \tag{1.77}$$

since the inner product of two such n-fold tensor product is just the ordinary product of the n single-Qbit inner products.

It is often useful to think of the inner product $\langle\phi|\psi\rangle$ as a linear *functional* associated with the vector $|\phi\rangle$ that takes vectors $|\psi\rangle$ into complex numbers by operating on it.

Dirac gave the functional associated with $|\phi\rangle$ the name "bra" vector

$$\big(|\phi\rangle\big)^\dagger \text{ or } \langle\phi|, \tag{1.78}$$

so that the inner product $\langle\phi|\psi\rangle$ could actually be viewed as a compact expression of the functional $\langle\phi|$ acting on the vector $|\psi\rangle$:

$$\langle\phi|\psi\rangle = \langle\phi|\big(|\psi\rangle\big). \tag{1.79}$$

It is a standard result of linear algebra that the set of all such linear functionals on the original vector space is itself a vector space of the same dimension, known as the dual space.

As see, Dirac introduced the cloying but universal (among physicists) nomenclature of calling vectors like $|\psi\rangle$ in the original space *ket-vectors* or *kets*, and vectors like $\langle\phi|$ in the dual space *bra-vectors* or *bras*.

This terminology was inspired (I'm not making this up) by the fact that the notation for the inner product (1.79) encloses the names of the two vectors between the bra[c]kets $\langle\ \rangle$.

The duals $|x\rangle^\dagger = \langle x|$ of the computational basis vectors are defined by linearity and their action (1.77) on the computational basis.

To make the rules (1.75) for the inner product come out right we must have

$$\big(\alpha|\psi\rangle + \beta|\phi\rangle\big)^\dagger = \alpha^*\langle\psi| + \beta^*\langle\phi|, \tag{1.80}$$

which defines the dual for any *superposition* of computational basis vectors.

If you feel more comfortable with components, note that if we represent a general ket

$$|\psi\rangle = \sum_{x=0}^{2^{n-1}} \alpha_x |x\rangle_n \tag{1.81}$$

by the *column* vector

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \end{pmatrix} \tag{1.82}$$

then the associated bra $\langle\psi|$ is represented by the *row* vector

$$\left(\alpha_0^* \; \alpha_1^* \; \alpha_2^* \ldots\right), \tag{1.83}$$

so that the inner product of $|\psi\rangle$ with

$$|\phi\rangle = \sum_{x=0}^{2^{n-1}} \beta_x |x\rangle_n \tag{1.84}$$

is just the ordinary matrix product:

$$\langle\phi|\psi\rangle = \left(\beta_0^* \; \beta_1^* \; \beta_2^* \ldots\right) \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \end{pmatrix} = \sum \beta_x^* \alpha_x. \tag{1.85}$$

If **A** is a general linear operator on kets, one also defines its action on bras by

$$\langle\psi|\mathbf{A} = \left(\mathbf{A}^\dagger|\psi\rangle\right)^\dagger. \tag{1.86}$$

It is a useful, if somewhat irritating, exercise in Dirac notation to prove that as so defined **A** is indeed linear when acting on bras.

Note that this definition extends the associative law to the three-fold product $\langle\psi|\mathbf{A}|\phi\rangle$ which can be interpreted, in more conventional notation, as being either $(\psi, \mathbf{A}\phi)$ or $(\mathbf{A}^\dagger\psi, \phi)$.

As a famous teacher of mine once put it, "you can think of **A** as acting either to the right or to the left in $\langle\psi|\mathbf{A}|\phi\rangle$."

Relations like these permit us to extend to combinations of states and operators the general rule for operators that the adjoint of the adjoint is the original object, and the adjoint of a product is the product of the adjoints in the opposite order.

Since the adjoint of a pure number (i.e. the operator which just multiplies any state by that number) is just the complex conjugate of the number, one has

$$\langle\psi|\mathbf{A}|\phi\rangle^* = \langle\psi|\mathbf{A}|\phi\rangle^\dagger = \langle\phi|\mathbf{A}^\dagger|\psi\rangle, \tag{1.87}$$

which gives us back the matrix definition of the adjoint as the complex conjugate of the transposed matrix.

One can also associate an *outer product* $|\psi\rangle\langle\phi|$ with a bra and a ket, defined as a linear operator on kets $|\chi\rangle$ satisfying

$$\big(|\psi\rangle\langle\phi|\big)|\chi\rangle = |\psi\rangle\big(\langle\phi|\chi\rangle\big). \tag{1.88}$$

If $|\psi\rangle$ and $|\phi\rangle$ have expansions (1.81) and (1.84) then the matrix expressing their outer product in the computational basis is given by the ordinary matrix product

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \end{pmatrix} (\beta_0^* \ \beta_1^* \ \beta_2^* \ldots), \tag{1.89}$$

whose (x-y)th element is $M_{xy} = \alpha_x \beta_y^*$.

Note that here (and almost everywhere else) Dirac notation eliminates the need for playing games with matrices, since the value of the matrix elements fall directly out of the notation.

For example, $\langle x|\psi\rangle\langle\phi|y\rangle$ can be interpreted either as the matrix element

$$M_{xy} = \langle x|\big(|\psi\rangle\langle\phi|\big)|y\rangle \tag{1.90}$$

of the outer-product operator $M = |\psi\rangle\langle\phi|$ or, equally well, as the product of two complex numbers

$$\big(\langle x|\psi\rangle\big)\big(\langle\phi|y\rangle\big). \tag{1.91}$$

The latter form (and the expansions (1.81) and (1.84) of $|\psi\rangle$ and $|\phi\rangle$ in the computational basis) gives us directly $\alpha_x \beta_y^*$.

This is a good example of the primary point of Dirac notation: it has many built-in ambiguities, but it is designed so that any way you chose to resolve those ambiguities is correct.

In this way elementary little theorems become consequences of the notation.

Mathematicians tend to loathe Dirac notation, because it prevents them from making distinctions they consider important.

Physicists love Dirac notation, because they are always forgetting that such distinctions exist and the notation liberates them from having to remember.

An important special case of the outer product is the operator

$$\mathbf{P}_\psi = |\psi\rangle\langle\psi| \tag{1.92}$$

(not to be confused with the classical permutation operators **P**).

This is just the projection operator on the state $|\psi\rangle$.

If a set of vectors $|\psi_i\rangle$, i = 1… n constitute a complete orthonormal set, then the sum of the projections on each of them is just the unit operator:

$$\mathbf{1} = \sum_{i=1}^{n} |\psi_i\rangle\langle\psi_i|. \qquad (1.93)$$

This trivial identity can be surprisingly useful.

Letting both sides act on an arbitrary vector $|\phi\rangle$, for example, it tells us that the amplitudes we need to expand $|\phi\rangle$ in the basis given by the $|\psi\rangle_i$ are $\langle\psi_i|\phi\rangle$:

$$|\phi\rangle = \sum_{i=1}^{n} |\psi_i\rangle\langle\psi_i|\phi\rangle. \qquad (1.94)$$

Sandwiching an arbitrary operator **A** between two such expansions of the identity, tells us how to expand **A** in terms of its matrix elements $A_{ij} = \langle\psi_i|A|\psi_j\rangle$ and the n^2 dimensional operator basis $|\psi_i\rangle\langle\psi_j|$ for the whole algebra of operators:

$$\mathbf{A} = \mathbf{1A1} = \sum_{i,j=1}^{n} |\psi_i\rangle\langle\psi_i|A|\psi_j\rangle\langle\psi_j| = \sum_{i,j=1}^{n} \big(\langle\psi_i|A|\psi_j\rangle\big)|\psi_i\rangle\langle\psi_j|\,. \qquad (1.95)$$

The Born rule, relating the amplitudes in the expansion (1.58) of a general 1-Qbit state $|\psi\rangle$ to the probabilities of measuring 0 or 1 is often stated in terms of inner products: If a Qbit is in a state $|\psi\rangle$ then the probabilities of a measurement of the Qbit giving 0 or 1 are given by

$$p_\psi(0) = |\langle 0|\psi\rangle|^2, \quad p_\psi(1) = |\langle 1|\psi\rangle|^2. \tag{1.96}$$

This can also be written in terms of the projection operator $\mathbf{P}_\psi = |\psi\rangle\langle\psi|$ as

$$p_\psi(0) = \langle 0|\mathbf{P}_\psi|0\rangle, \quad p_\psi(1) = \langle 1|\mathbf{P}_\psi|1\rangle. \tag{1.97}$$

The rule can also be stated in terms of the projection operators $\mathbf{P}_0 = |0\rangle\langle 0|$ and $\mathbf{P}_1 = |1\rangle\langle 1|$ as

$$p_\psi(0) = \langle\psi|\mathbf{P}_0|\psi\rangle, \quad p_\psi(1) = \langle\psi|\mathbf{P}_1|\psi\rangle. \tag{1.98}$$

More generally, if $|\Psi\rangle$ is the state of n Qbits, then the probability of a measurement giving the result x ($0 \le x < 2^n$) is

$$p_\Psi(x) = |\langle x|\Psi\rangle|^2 = \langle x|\mathbf{P}_\Psi|x\rangle = \langle\Psi|\mathbf{P}_x|\Psi\rangle, \tag{1.99}$$

where $\mathbf{P}_\Psi = |\Psi\rangle\langle\Psi|$ and $\mathbf{P}_x = |x\rangle\langle x|$.

Structure of the general 1-Qbit unitary transformation (learn some interesting QM stuff!)

I describe here some relations between Pauli matrices, unitary transformations, and real-space rotations.

They are of fundamental importance in many applications of quantum mechanics, and are an essential part of the intellectual equipment of anybody wanting a deep understanding of 3-dimensional rotations.

Occasionally they can also be useful in applications to quantum computation, as we will see.

The unit operator **1** and the three Pauli matrices (1.49) form a *basis* for the four dimensional algebra of two dimensional matrices: any two-dimensional matrix has a unique expansion of the form

$$\mathbf{u} = u_0 + \mathbf{u} \cdot \sigma \qquad (1.100)$$

for some complex number u_0 and 3-vector **u** with complex components u_x,u_y, and u_z.

Here σ, unadorned by any subscript, represents the "3-vector" whose components are the Pauli matrices σ_x, σ_y, and σ_z given in (1.49), so in expanded form (1.100) reads

$$\mathbf{u} = u_0\mathbf{1} + u_x\sigma_x + u_y\sigma_y + u_z\sigma_z = \begin{pmatrix} u_0 + u_z & u_x - iu_y \\ u_x + iu_y & u_0 - u_z \end{pmatrix}. \qquad (1.101)$$

Because there are only a finite number of letters, etc available you need to be careful.

Please take care to distinguish, for example, between the various use sometimes of a boldface **u** as a 3-component vector and other times as an operator on two-dimensional vectors.

Some writers will use slightly different fonts, but that is often difficult to see.

In general, which one is being used is *always* evident from the *context* - one just has to *get used* to this!

While in most of these notes we follow the computer science notation, calling σ_x,σ_y, and σ_z by their computer-science aliases, **X**, i**Y**, and **Z**, the asymmetry introduced by the non-hermitian matrix **Y** makes the analysis that follows awkward and unattractive, so we stick here with the physicists' notation.

Note also the physicists' practice — a generalization of their habit of not distinguishing notationally between the unit operator **1** and the scalar 1 — of omitting the explicit occurrence of **1** from $u_0\mathbf{1}$.

It is quite analogous to regarding the ordinary complex numbers as an extension of the real numbers, rather than as real linear combinations of the two-dimensional matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

$$\mathbf{u} = u_0 + \mathbf{u} \cdot \sigma$$

As what follows demonstrates, however, it is invariably simpler to use the form (1.100) together with the multiplication rule (1.56), rather than dealing explicitly with 2-dimensional matrices.

$$\mathbf{u} = u_0 + \mathbf{u} \cdot \sigma$$

We now impose on (1.100) the condition (1.65) that **u** be unitary.

Since any unitary matrix remains unitary if it is multiplied by an overall multiplicative phase factor $e^{i\theta}$ with θ real, we can require u_0 to be real and arrive at a form which is general *except* for such an overall phase factor.

Since the Pauli matrices are hermitian, we then have

$$\mathbf{u}^\dagger = u_0 + \mathbf{u}^* \cdot \sigma. \qquad (1.102)$$

The rule (1.56) now tells us that for **u** to be unitary we must have

$$0 = 1 - \mathbf{u}^\dagger \mathbf{u} = 1 - u_0^2 - \mathbf{u}^* \cdot \mathbf{u} - \big(u_0(\mathbf{u} + \mathbf{u}^*) + i\mathbf{u}^* \times \mathbf{u}\big) \cdot \sigma. \qquad (1.103)$$

Since **1**, σ_x, σ_y, and σ_z, are linearly independent in the 4-dimensional algebra of 1-Qbit operators, the coefficients of all four in (1.103) must vanish and we have

$$1 = u_0^2 + \mathbf{u}^* \cdot \mathbf{u}, \quad 0 = u_0(\mathbf{u} + \mathbf{u}^*) + i\mathbf{u}^* \times \mathbf{u}. \tag{1.104}$$

The second of these requires the real and imaginary parts of the vector **u** to satisfy

$$u_0 \mathrm{Re}\,\mathbf{u} = \mathrm{Re}\,\mathbf{u} \times \mathrm{Im}\,\mathbf{u}. \tag{1.105}$$

If $u_0 \neq 0$, it follows from (1.105) that Re**u**·Re**u** = 0, so Re**u** = 0, and the vector **u** must be i times a real vector **v**.

On the other hand if $u_0 = 0$ then (1.105) requires the real and imaginary parts of **u** to be parallel vectors, so that **u** itself is just a complex multiple of a real vector.

But if $u_0 = 0$ we retain the freedom to pick the overall phase of the operator **u**, which we can chose to make the vector **u** purely imaginary.

So whether or not $u_0 = 0$, the general form for a two-dimensional unitary **u** is, to within an overall phase factor,

$$\mathbf{u} = u_0 + i\mathbf{v} \cdot \sigma, \tag{1.106}$$

where u_0 is a real number, **v** is a real vector, and, from the first of (1.104),

$$u_0^2 + \mathbf{v} \cdot \mathbf{v} = 1. \tag{1.107}$$

The identity (1.107) is ensured by parametrizing u_0 and **v** in terms of a real unit vector **n** parallel to **v**, and a real angle γ so that

$$\mathbf{u} = \cos\gamma + i(\mathbf{n} \cdot \sigma)\sin\gamma. \tag{1.108}$$

Reiterating the warning above, do not confuse boldface **n**, the real unit 3-vector, with boldface **n**, the 1-qubit number operator.

An alternative way of writing (1.108) is

$$\mathbf{u} = e^{i\gamma\mathbf{n}\cdot\sigma}. \tag{1.109}$$

This follows from the forms of the power series expansions of the exponential, sine, and cosine, together with the fact that $(\mathbf{n}\cdot\sigma)^2 = 1$ for any unit vector **n** as a special case of (1.56).

The argument is the same as the argument that $e^{i\phi} = \cos\phi + i\sin\phi$ for any real number ϕ.

A remarkable connection between these 2-dimensional unitary matrices and ordinary three dimensional rotations emerges from the fact that each of the three Pauli matrices (1.49) has zero trace, and the fact that the operator unitary transformation

$$\mathbf{A} \rightarrow \mathbf{u}\mathbf{A}\mathbf{u}^{\dagger} \tag{1.110}$$

preserves the trace of **A**.

Note first that if **a** is a real vector then $\mathbf{u}(\mathbf{a}\cdot\sigma)\mathbf{u}^{\dagger}$ is hermitian and can therefore be expressed as a linear combination of **1** and the three Pauli matrices with real coefficients.

Since σ_x, σ_y, and σ_z all have zero trace, so does $\mathbf{a}\cdot\sigma$ and therefore so does $\mathbf{u}(\mathbf{a}\cdot\sigma)\mathbf{u}^{\dagger}$.

Its expansion as a linear combination of **1** and the three Pauli matrices must therefore be of the form $\bar{\mathbf{a}}\cdot\sigma$ for some real vector $\bar{\mathbf{a}}$ (since **1** alone among the four has non-zero trace):

$$\mathbf{u}(\mathbf{a}\cdot\sigma)\mathbf{u}^{\dagger} = \bar{\mathbf{a}}\cdot\sigma. \tag{1.111}$$

It follows that

$$\mathbf{u}(\mathbf{a}\cdot\sigma)(\mathbf{b}\cdot\sigma)\mathbf{u}^{\dagger} = \big(\mathbf{u}(\mathbf{a}\cdot\sigma)\mathbf{u}^{\dagger}\big)\big(\mathbf{u}(\mathbf{b}\cdot\sigma)\mathbf{u}^{\dagger}\big) = \big(\bar{\mathbf{a}}\cdot\sigma\big)\big(\bar{\mathbf{b}}\cdot\sigma\big). \tag{1.112}$$

Since unitary transformations preserve the trace,

$$\mathrm{Tr}(\mathbf{a}\cdot\sigma)(\mathbf{b}\cdot\sigma) = \mathrm{Tr}\big(\overline{\mathbf{a}}\cdot\sigma\big)\big(\overline{\mathbf{b}}\cdot\sigma\big). \tag{1.113}$$

Hence, from (1.56),

$$\overline{\mathbf{a}}\cdot\overline{\mathbf{b}} = \mathbf{a}\cdot\mathbf{b}. \tag{1.114}$$

But the most general real, linear inner-product-preserving transformation on real 3-vectors is a *rotation*.

Consequently the transformation from real 3-vectors $\mathbf{a}$ to real 3-vectors $\overline{\mathbf{a}}$ induced by any 2-dimensional unitary $\mathbf{u}$ through (1.111) is a rotation:

$$\overline{\mathbf{a}} = \mathbf{R_u}\,\mathbf{a}. \tag{1.115}$$

Furthermore, by applying the (unitary) product $\mathbf{uv}$ of two unitary transformations in two steps,

$$(\mathbf{uv})(\mathbf{a}\cdot\sigma)(\mathbf{uv})^\dagger = \mathbf{u}\big(\mathbf{v}(\mathbf{a}\cdot\sigma)\mathbf{v}^\dagger\big)\mathbf{u}^\dagger = \mathbf{u}\big(R_\mathbf{v}\mathbf{a}\cdot\sigma\big)\mathbf{u}^\dagger = \mathbf{R_u}\mathbf{R_v}\mathbf{a}\cdot\sigma, \tag{1.116}$$

we deduce that

$$\mathbf{R_{uv}} = \mathbf{R_u}\mathbf{R_v}. \tag{1.117}$$

Thus the association of three-dimensional rotations with two-dimensional unitary matrices preserves the multiplicative structure of the rotation group: the rotation associated with the product of two unitary transformations is the product of the two associated rotations.

Which rotation is associated with which unitary transformation?

To answer this note first that when the vector **a** in (1.111) is taken to be the vector **n** appearing in **u** (in (1.108) or (1.109)) then $\bar{\mathbf{n}} = \mathbf{n}$, since **u** then commutes with $\mathbf{n}\cdot\sigma$.

Therefore **n** is along the axis of the rotation associated with $\mathbf{u} = e^{i\gamma \mathbf{n}\cdot\sigma}$.

To determine the angle θ of that rotation let **m** be any unit vector perpendicular to the axis **n**, so that

$$\cos\theta = \mathbf{m}\cdot\overline{\mathbf{m}}. \tag{1.118}$$

We then have

$$\begin{aligned}
\cos\theta &= \tfrac{1}{2}\mathrm{Tr}\big((\mathbf{m}\cdot\sigma)(\overline{\mathbf{m}}\cdot\sigma)\big) \\
&= \tfrac{1}{2}\mathrm{Tr}\big((\mathbf{m}\cdot\sigma)(\cos\gamma + i\sin\gamma\ \mathbf{n}\cdot\sigma)(\mathbf{m}\cdot\sigma)(\cos\gamma - i\sin\gamma\ \mathbf{n}\cdot\sigma)\big) \\
&= \tfrac{1}{2}\mathrm{Tr}\big((\cos\gamma\mathbf{m} - \sin\gamma\mathbf{m}\times\mathbf{n})\cdot\sigma)(\cos\gamma\mathbf{m} + \sin\gamma\mathbf{m}\times\mathbf{n})\cdot\sigma)\big) \\
&= \cos^2\gamma - \sin^2\gamma = \cos(2\gamma),
\end{aligned} \tag{1.119}$$

where we have made repeated use of (1.56) and the fact that $\mathbf{m}\cdot\mathbf{n} = 0$.

So the unitary matrix (1.109) or (1.108) is associated with a rotation about the axis **n** through the angle 2γ.

Since the identity rotation is associated both with $\mathbf{u} = \mathbf{1}$ and $\mathbf{u} = -\mathbf{1}$, the correspondence between these unitary matrices and three dimensional proper rotations is a 2-to-1 *homomorphism* - a *transformation* of one set into another that preserves in the second set the operations between the members of the first set.

It is useful to introduce the notation $\mathbf{u}(\mathbf{n},\theta)$ for the 1-Qbit unitary transformation associated with the rotation $\mathbf{R}(\mathbf{n},\theta)$ about the axis **n** through the angle θ:

$$\mathbf{u}(\mathbf{n},\theta) = e^{i(\theta/2)(\mathbf{n}\cdot\sigma)} = \cos\tfrac{1}{2}\theta + i(\mathbf{n}\cdot\sigma)\sin\tfrac{1}{2}\theta. \qquad (1.120)$$

The 3-dimensional rotations arrived at in this way are all proper (i.e. they preserve rather than invert handedness) because they can all be continuously connected to the identity.

Any proper rotation can be associated with a **u**, and in just two different ways (**u** and **–u** clearly being associated with the same rotation).

The choice of phase leading to the general form (1.106) with real u_0 can be imposed by requiring that the determinant of **u** must be 1.

Although all this may strike you as tediously abstract formalism, it is actually extremely useful at a very practical level, in reducing some highly nontrivial 3-dimensional geometry to elementary algebra, just as Euler's relation $e^{i\phi} = \cos\phi + i \sin\phi$ reduces some nontrivial 2-dimensional trigonometry to simple algebra.

Suppose, for example, you combine a rotation through an angle α about an axis given by the unit vector **a** with a rotation through β about **b**.

The result, of course, is a single rotation.

For combined rotation, what is its angle γ and axis **c**?

Answering this question can be a *nasty* exercise in three-dimensional geometry.

But to answer it using the Pauli matrices you only have to note that $\mathbf{u}(\mathbf{c},\gamma) = \mathbf{u}(\mathbf{a},\alpha)\mathbf{u}(\mathbf{b},\beta)$, i.e.

$$\cos\tfrac{1}{2}\gamma + i(\mathbf{c}\cdot\sigma)\sin\tfrac{1}{2}\gamma = \left(\cos\tfrac{1}{2}\alpha + i(\mathbf{a}\cdot\sigma)\sin\tfrac{1}{2}\alpha\right)\left(\cos\tfrac{1}{2}\beta + i(\mathbf{b}\cdot\sigma)\sin\tfrac{1}{2}\beta\right). \qquad (1.121)$$

Now multiply out the right side of (1.121), using (1.56).

To get the angle γ take the trace of both sides (or identify the coefficients of unity) to find

$$\cos\tfrac{1}{2}\gamma = \cos\tfrac{1}{2}\alpha\cos\tfrac{1}{2}\beta - (\mathbf{a}\cdot\mathbf{b})\sin\tfrac{1}{2}\alpha\sin\tfrac{1}{2}\beta. \qquad (1.122)$$

To get the axis **c**, identify the vectors of coefficients of the Pauli matrices:

$$\sin\tfrac{1}{2}\gamma\ \mathbf{c} = \sin\tfrac{1}{2}\beta\cos\tfrac{1}{2}\alpha\ \mathbf{b} + \sin\tfrac{1}{2}\alpha\cos\tfrac{1}{2}\beta\ \mathbf{a} - \sin\tfrac{1}{2}\alpha\sin\tfrac{1}{2}\beta\,(\mathbf{a}\times\mathbf{b}). \qquad (1.123)$$

Note that (1.122) and (1.123) are trivially correct when **a** and **b** are parallel.

A little geometrical thought reveals that they are also correct when α and β are both 180◦.

To try to see geometrically why they are correct more generally is to acquire a deep appreciation for the remarkable power of the representation of 3-dimensional rotations in terms of 2-dimensional Pauli matrices.

Structure of the general 1-Qbit state (learning more mathematics)

Let $|\phi\rangle$ be any 1-Qbit state, and let $|\psi\rangle$ be the orthogonal state (unique to within an overall phase), satisfying $\langle\psi|\phi\rangle = 0$.

There is a *unique* unitary **u** taking the computational basis states $|0\rangle$ and $|1\rangle$ into $|\phi\rangle$ and $|\psi\rangle$.

Now the computational basis states are eigenstates of the 1-Qbit number operator:

$$\mathbf{n}|x\rangle = x|x\rangle, \quad x = 0 \text{ or } 1, \tag{1.124}$$

where

$$\mathbf{n} = \tfrac{1}{2}(1 - \mathbf{Z}) = \tfrac{1}{2}(1 - \sigma_z) = \tfrac{1}{2}(1 - \mathbf{z}\cdot\sigma). \tag{1.125}$$

Since

$$|\phi\rangle = \mathbf{u}|0\rangle, \quad |\psi\rangle = \mathbf{u}|1\rangle, \tag{1.126}$$

the operator $\mathbf{n}' = \mathbf{unu}^\dagger$ acts as a Qbit number operator on $|\phi\rangle$ and $|\psi\rangle$:

$$\mathbf{n}'|\phi\rangle = 0, \quad \mathbf{n}'|\psi\rangle = |\psi\rangle. \tag{1.127}$$

Since any 1-Qbit unitary transformation **u** is associated with a rotation **R**(**m**,θ), we have

$$\mathbf{n}' = \mathbf{u}\,\mathbf{n}\,\mathbf{u}^\dagger = \tfrac{1}{2}\big(1 - \mathbf{u}(\mathbf{z}\cdot\sigma)\mathbf{u}^\dagger\big) = \tfrac{1}{2}(1 - \mathbf{z}'\cdot\sigma), \tag{1.128}$$

where $\mathbf{z}' = \mathbf{R}(\mathbf{m},\theta)\mathbf{z}$.

Thus **n**′, which functions as a number operator for the states $|\phi\rangle = \mathbf{u}(\mathbf{m},\theta)|0\rangle = |0'\rangle$ and $|\psi\rangle = \mathbf{u}(\mathbf{m},\theta)|1\rangle = |1'\rangle$ is constructed out of the component of the vector of operators σ along the direction $\mathbf{z}' = \mathbf{R}(\mathbf{m},\theta)\mathbf{z}$ in exactly the same way that **n**, the number operator for the computational basis states $|0\rangle$ and $|1\rangle$is constructed out of the component along z.

This *suggests* that that there might be nothing special about the *choice* of $|0\rangle$ and $|1\rangle$ to form the computational basis states for each Qbit — that *any pair* of orthogonal states, $|0'\rangle = \mathbf{u}|0\rangle$and $|1'\rangle = \mathbf{u}|1\rangle$ could serve as well.

Furthermore it is at least a consistent possibility that to get the apparatus to measure the Qbits in this new basis we need do nothing more than than apply the rotation **R** associated with **u** to the apparatus that served to measure them in the original basis. Neat!!!

This physical possibility is realized by some, but by no means all, of the physical systems that have been proposed as possible embodiments of Qbits.

It is realized for certain atomic magnets — also called spins— which have the property that when the magnetization of such a spin is measured along any given direction the magnet is found to be either maximally aligned along that direction, or maximally aligned opposite to that direction.

These two possible outcomes for a particular direction — conventionally taken to be **z**— are associated with the values 0 and 1 for the Qbit.

After such a measurement the spin is left in the state $|0\rangle$ or $|1\rangle$.

Any other state $|\phi\rangle$ and its orthogonal partner $|\psi\rangle$ specify an alternative direction, along which the magnetization might have been measured, associated with an alternative scheme for reading out values for the Qbits.

From this point of view the immensely greater set of possible states available to a Qbit than is available to a Cbit reflects the continuum of different ways one can read a Qbit (measuring its magnetization along any direction) as opposed to the single option available for reading a Cbit (finding out what value it actually has).

For Qbits that are not spins, the richness lies in the possibility of applying an arbitrary unitary transformation to each Qbit, before measuring it in the computational basis.

What makes spins special is that applying the unitary transformation to the Qbits (which is not always that easy to arrange) can be *replaced* by straightforwardly applying the corresponding rotation to every one-Qbit measurement gate.

We are now going to apply some of the quantum formalism developed in this part to illustrate a famous and striking peculiarity of Qbits called “Spooky Action at a Distance.”

Although the subject is of broad cultural interest, it is not directly relevant to quantum computation or quantum information processing.

I have included it primarily to give you some further illustrations of how to use the quantum formalism developed in the part.

An application of the formalism: "Spooky action at a distance"

It is well worth you time and effort to do this stuff in detail on your own!!)

Suppose Alice and Bob *each* has one member of a pair of Qbits, which have been *prepared* in the 2-Qbit state

$$|\Phi\rangle = \tfrac{1}{\sqrt{12}}\big(3|00\rangle + |01\rangle + |10\rangle - |11\rangle\big). \qquad (1.129)$$

This state can also be written in terms of 1-Qbit Hadamard operators (1.42) as

$$|\Phi\rangle = \tfrac{1}{\sqrt{3}}\big(2|00\rangle - \mathbf{H}\otimes\mathbf{H}|11\rangle\big) = \tfrac{1}{\sqrt{3}}\big(2|00\rangle - \mathbf{H}_a\mathbf{H}_b|11\rangle\big), \qquad (1.130)$$

where we take $\mathbf{H}_a$ to act on the left (Alice's) Qbit and $\mathbf{H}_b$ to act on the right (Bob's) Qbit.

It follows from (1.129) that If Alice and Bob each measures their Qbit, then there is a non-zero probability (1/12) that each will get the value 1.

The following additional features of the state $|\Phi\rangle$ *lead* to a rather peculiar state of affairs:

Follow very closely!!

(i) If Alice and Bob each applies a Hadamard to their Qbit then, since $\mathbf{H}^2 = 1$, the state (1.130) of the Qbits becomes

$$\mathbf{H}_a\mathbf{H}_b|\Phi\rangle = \tfrac{1}{\sqrt{3}}\big(2\mathbf{H}_a\mathbf{H}_b|00\rangle - |11\rangle\big) = \tfrac{1}{\sqrt{3}}\big(|00\rangle + |01\rangle + |10\rangle\big), \tag{1.131}$$

so if they measure their Qbits after each has applied a Hadamard, then there the probability that both will get the value 1 is zero.

(ii) If only Alice applies a Hadamard to her Qbit, then the state (1.130) of the Qbits becomes

$$\mathbf{H}_a|\Phi\rangle = \tfrac{1}{\sqrt{3}}\big(2\mathbf{H}_a|00\rangle - \mathbf{H}_b|11\rangle\big). \tag{1.132}$$

Since $\mathbf{H}_a|00\rangle = \big(\mathbf{H}|0\rangle\big) \otimes |0\rangle$, it is a linear combination of $|00\rangle$ and $|10\rangle$, and since $\mathbf{H}_b|11\rangle = |1\rangle \otimes \big(\mathbf{H}|1\rangle\big)$, it is a linear combination of $|10\rangle$ and $|11\rangle$. Thus the state $|01\rangle$ does not appear in the expansion of $\mathbf{H}_a|\Phi\rangle$ in computational basis states, so when the Qbits are subsequently measured, if Bob gets the value 1, there is zero probability that Alice will get the value 0.

(iii) If only Bob applies a Hadamard to his Qbit, then by exactly the same reasoning (except for the interchange of Alice and Bob), when the Qbits are subsequently measured if Alice gets the value 1, there is zero probability that Bob will get the value 0.

Taken all together, these probabilities for certain outcomes, depending on whether each does or does not apply a Hadamard gate before measuring, *are quite strange*.

For suppose that Alice and Bob each *independently* decide whether to apply a Hadamard to their qubit, *before* sending it through a measurement gate, by *tossing* a coin.

There is then a small but non-zero probability ($1/4 \times 1/12 = 1/48$) that neither applies a Hadamard and both measurement gates show 1.

In the *one time* in 48 that this happens, it is *tempting* to conclude that each Qbit was, even *before* the coins were tossed, *capable* of producing a 1 when directly subject to a measurement gate because, after all, each Qbit *did* produce a 1 when directly subject to a measurement gate.

But if Bob's Qbit had such a capability, then, in the *absence* of spooky interactions between Alice's Hadamard and Bob's Qbit, his Qbit surely would have *retained* that capability, *even* if Alice applied a pre-measurement Hadamard to her own Qbit.

But if Bob's Qbit *was* indeed still capable of registering a 1 when measured directly, then Alice's Qbit must have been *incapable* of registering a 0 if measured after a Hadamard, since, as noted in (ii) above, when Alice applies a Hadamard before her measurement and Bob does not it is *impossible* for Alice's measurement to give 0 while Bob's gives 1.

By the *same* reasoning (interchanging Alice and Bob and referring to (iii) above) we *conclude* that Bob's Qbit must also have been *incapable* of registering a 0 when measured *after* a Hadamard.

So in each of the slightly more than 2% of the cases in which *neither* Bob *nor* Alice apply Hadamards and *both* their measurement gates register 1, we conclude that if the tosses of *both* coins had come out the *other* way and *both* had applied Hadamards *before* measuring, then *neither* Qbit *could* have registered 0 when *measured*: *both* would have had to register 1.

But *according* to (i) above, the probability of *both* registering 1 after a Hadamard is *zero*.

Over the years *passions* have run high on the *significance* of this.

Some claim that it shows that the value Alice or Bob finds upon measuring *does* depend on *whether* or not the other, who could be far away, *applies* a Hadamard to his or her own Qbit *before* measuring.

They call this "quantum nonlocality" or "spooky action at a distance".

This is a *translation* of Einstein's disparaging term spukhafte Fernwirkungen.

Although this particular argument was discovered by Lucien Hardy in the early 1990's, similar situations (where one has to work harder to unearth a paradox) have been known since a famous paper by John Bell appeared in 1964.

My *own* take on it is rather different.

With any given pair of Qbits, each of Alice and Bob either does or does not apply a Hadamard prior to their measurement.

Only one of the *four* possible cases is actually *realized*.

The *other* three cases do *not* happen.

In a *deterministic* world it can make sense to talk about what *would* have happened if things had been other than the way they actually were, *since* the hypothetical situation can entail unique subsequent behavior.

But in the intrinsically *nondeterministic* world that we actually inhabit, it makes *no sense* to infer, from what Bob's Qbit actually *did*, that it *has* a "capability" to do *what* it actually did, which it *retains* even in a hypothetical situation that *did not*, in fact, take place.

To *characterize* the possible behavior of Bob's Qbit in the fictitious world requires more than just the *irrelevance* of Alice's decision whether or not to apply a Hadamard.

It *also* requires that whatever it is that actually is *relevant* to Bob's outcome *remains* the same in both worlds and *plays* the same role in bringing about that outcome.

But in the quantum world there is an *irreducible* randomness to such outcomes: *nothing* need play a role in bringing them about.

The real *lesson* is that *if* one has a single pair of Qbits and various choices of gates to apply to them before sending them through a measurement gate, *then* it makes no sense to infer *from* the actual outcome of the measurement for the actual choice of gates, *additional* constraints (*beyond* those implied by the initial state of the Qbits) on the *hypothetical* outcomes of measurements in the fictional case in which one made a *different* choice of gates. (you have to read that many times!)

One can, however, *let* Alice and Bob repeatedly play this game with many different pairs of Qbits, always preparing the Qbits in the same initial 2-Qbit state (1.129).

It is then entirely *sensible* to ask whether the *statistics* of the values Bob finds upon measuring his Qbit *depend* on *whether* Alice applied a Hadamard transform to her Qbit.

For Alice and Bob can then *accumulate* a mass of data, and directly *compare* the statistics Bob got when Alice applied the Hadamard with those he got when she did not.

If Bob got a different statistical distribution of readings *depending* on whether Alice did or did not apply a Hadamard to her faraway Qbit before she measured it, this would *permit* nonspooky action at a distance which could actually be used to *send* messages.

So it is important to *note* that Bob's statistics *do not*, in fact, depend on whether or not Alice applies a Hadamard.

We can *show* this under quite general conditions.

Suppose n Qbits are *divided* into two subsets, each of which may be *independently* manipulated (i.e. subject to unitary transformations) *prior* to a measurement.

Let the n_a Qbits on the left constitute *one* such group and the $n_b = n - n_a$ on the right, the *other*.

Think of the first group as under the *control* of Alice and the second as *belonging* to Bob.

If the n Qbits are *always* prepared in the state $|\Psi\rangle$, then *if* Alice and Bob separately measure their Qbits, the Born rule *tells* us that the joint probability p(x,y) of Alice getting x and Bob, y, is

$$p_\Psi(x,y) = \langle\Psi|\mathbf{P}_x^a\mathbf{P}_y^b|\Psi\rangle, \tag{1.133}$$

where the projection operator $\mathbf{P}^a$ acts only on Alice's Qbits (i.e. it acts as the identity on Bob's) and $\mathbf{P}^b$, only on Bob's.

Suppose, now, that Alice acts on her Qbits with the unitary transformation $\mathbf{U}_a$ *before* making her measurement and Bob acts on his with **U**b.

Then the state $|\Psi\rangle$ is *changed* into

$$|\Phi\rangle = \mathbf{U}_a\mathbf{U}_b|\Psi\rangle. \tag{1.134}$$

Now the *probability* of their measurements giving x and y, conditioned on their choices of unitary transformation, is

$$p_\Psi(x,y|\mathbf{U}_a,\mathbf{U}_b)=\langle\Phi|\mathbf{P}_x^a\mathbf{P}_y^b|\Phi\rangle=\langle\Psi|\mathbf{U}_b^\dagger\mathbf{U}_a^\dagger\big(\mathbf{P}_x^a\mathbf{P}_y^b\big)\mathbf{U}_a\mathbf{U}_b|\Psi\rangle$$

$$=\langle\Psi|(\mathbf{U}_a^\dagger\mathbf{P}_x^a\mathbf{U}_a)(\mathbf{U}_b^\dagger\mathbf{P}_y^b\mathbf{U}_b)|\Psi\rangle, \qquad (1.135)$$

where we have used the fact that all operators that act only on Alice's Qbits *commute* with all operators that act only on Bob's.

It *follows* from the fact that

$$\sum_x\mathbf{U}_a^\dagger\mathbf{P}_x^a\mathbf{U}_a=\mathbf{U}_a^\dagger\Big(\sum_x\mathbf{P}_x^a\Big)\mathbf{U}_a=\mathbf{U}_a^\dagger\mathbf{1}\mathbf{U}_a=1, \qquad (1.136)$$

that Bob's *marginal* statistics *do not depend* on what Alice chose to do to her own Qbits:

$$p_\Psi(y|\mathbf{U}_a\mathbf{U}_b)=\sum_x p_\Psi(x,y|\mathbf{U}_a,\mathbf{U}_b)=\langle\Psi|(\mathbf{U}_b^\dagger\mathbf{P}_y^b\mathbf{U}_b)|\Psi\rangle=p_\Psi(y|\mathbf{U}_b), \qquad (1.137)$$

which *does not depend* on the particular unitary transformation $\mathbf{U}_a$ chosen by Alice.

Therefore the statistics of the measurement outcomes for any group of Qbits, are *not* altered by anything done to other Qbits (*provided*, of course, the other Qbits do not subsequently interact with those in the original group, for example by the application of appropriate 2-Qbit gates). WOW!!

Conscience requires Mermin to report here the *existence* of a small but vocal deviant subculture of physicists, known as *Bohmians*, who *maintain* that there is a deterministic sub-structure, unfortunately *inaccessible* to us, that *underlies* quantum phenomena.

Needless to say, all Bohmians are *enthusiastic believers* in real instantaneous action at a distance.

Personally, I think the Bohmians and the String Theorists are as close as we can get in physics to MAGA believers - they just will not accept any arguments that do not go their way!

possible end 10 - see how much time left…..

Part 2. Quantum Computation: General features and some simple examples

A. The general computational process

We would like a suitably programmed quantum computer to act on a number x to produce another number $f(x)$ for some specified function f.

Appropriately interpreted, with an accuracy that increases with increasing k, we can treat all such numbers as non-negative integers less than 2^k.

Each integer is represented in the quantum computer by the corresponding computational-basis state of k Qbits.

If we specify x as an n-bit integer and $f(x)$ as an m-bit integer, then we shall need at least n+m Qbits: a set of n-Qbits, called the *input register*, to represent x, and another set of m-Qbits, called the *output register*, to represent $f(x)$.

Qbits being a scarce commodity, you might wonder *why* we need separate registers for input and output.

One important reason is that if $f(x)$ assigns the same value to different values of x, as many interesting functions do, then the computation *cannot* be inverted if its only effect is to transform the contents of a single register from x to $f(x)$.

Having separate registers for input and output is standard practice in the classical theory of reversible computation.

Since (as remarked in earlier and expand on later in this part) quantum computers must operate reversibly (except for measurement gates) to perform their magic, quantum computers are generally designed to operate with both input and output registers.

We shall find that this dual-register procedure can be exploited by a quantum computer in some strikingly nonclassical ways.

The computational process will generally require many Qbits besides the $n + m$ in the input and output registers, but we shall ignore these additional Qbits for the moment, viewing a computation of f as doing nothing more than applying a unitary transformation, $\mathbf{U}_f$ to the $n + m$ Qbits of the input and output registers.

We shall return to the fundamental question of why the additional Qbits can be ignored later, only noting for now that it will be the reversibility of the computation that makes it possible for them to be ignored.

To define unitary transformations $\mathbf{U}$ it is enough to define their action on any basis, since any other state $|\Psi\rangle$ must be a linear superposition of basis states, and therefore, since unitary transformations are linear, their action on $|\Psi\rangle$ is entirely determined by their action on the basis.

In most cases of interest one takes the basis in which **U** is defined to be the so-called computational (classical) basis.

In many cases (such as the definition below of $\mathbf{U}_f$) the action of **U** on each computational basis state is *simply* to produce another computational basis state.

Since **U** must have an inverse, it therefore acts on the entire computational basis as a *permutation*, just as a classical transformation does.

Since linear transformations that take orthonormal bases into other orthonormal bases are necessarily unitary, and since a permutation of an orthonormal basis trivially remains an orthonormal basis, any invertible linear transformation that is defined to take computational basis states into other computational basis states, is *automatically* unitary.

Putting the point another way, *any classically meaningful reversible transformation on Cbits, has a linear extension to a unitary transformation on Qbits that acts as the classical transformation when restricted to states of the computational basis*.

I shall use this fact repeatedly in defining unitary transformations on Qbits, without explicitly reminding you each time of this justification for it.

The standard quantum computational protocol defines the action of $\mathbf{U}_f$ on every computational basis state $|x\rangle_n|y\rangle_m$ of the $n + m$ Qbits making up the input and output registers as follows:

$$\mathbf{U}_f\big(\,|x\rangle_n|y\rangle_m\big) = |x\rangle_n|y \oplus f(x)\rangle_m, \tag{2.1}$$

where $\oplus$ indicates modulo-2 bitwise addition (without carrying), a straightforward generalization of the single-bit $\oplus$ defined in Part 1.

If x and y are m-bit integers whose j-th bits are x_j and y_j, then $x \oplus y$ is the m-bit integer whose j-th bit is $x_j \oplus y_j$.

Thus $1101 \oplus 0111 = (1 \oplus 0)\,(1 \oplus 1)\,(0 \oplus 1)\,(1 \oplus 1) = 1010.$

If the initial value represented by the output register is $y = 0$ then we have from (2.1)

$$\mathbf{U}_f\big(\,|x\rangle_n|0\rangle_m\big) = |x\rangle_n|f(x)\rangle_m \tag{2.2}$$

and we do indeed end up with $f(x)$ in the output register.

Regardless of the initial value of y the input register *ends up* back in its initial state $|x\rangle_n$.

The transformation (2.1) is clearly invertible.

Indeed $\mathbf{U}_f$ is its own inverse:

$$\mathbf{U}_f\mathbf{U}_f\big(|x\rangle|y\rangle\big) = \mathbf{U}_f\big(|x\rangle|y \oplus f(x)\rangle\big) = |x\rangle|y \oplus f(x) \oplus f(x)\rangle = |x\rangle|y\rangle, \qquad (2.3)$$

since $z \oplus z = 0$ for any z.

From now on I shall keep the subscripts reminding you of the numbers of Qbits only when it is important to emphasize what those numbers are.

Since **U** is invertible and takes computational basis states into other computational basis states, its linear extension to all Qbit states is indeed unitary.

$$\mathbf{U}_f\big(|x\rangle_n|0\rangle_m\big) = |x\rangle_n|f(x)\rangle_m \qquad (2.2)$$

The form (2.2) inspires one of the most important tricks of the quantum computational repertoire.

If we apply to each Qbit in the two-Qbit state $|0\rangle|0\rangle$ the 1-Qbit Hadamard transformation **H** (introduced earlier), then we get

$$(\mathbf{H}\otimes\mathbf{H})(|0\rangle|0\rangle) = (\mathbf{H}|0\rangle)(\mathbf{H}|0\rangle) = \tfrac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\tfrac{1}{\sqrt{2}}(|0\rangle+|1\rangle) =$$

$$\tfrac{1}{2}(|0\rangle|0\rangle+|0\rangle|1\rangle+|1\rangle|0\rangle+|1\rangle|1\rangle) = \tfrac{1}{2}(|0\rangle_2+|1\rangle_2+|2\rangle_2+|3\rangle_2). \qquad (2.4)$$

This *generalizes* to the n-fold tensor product of n Hadamard transforms (written $\mathbf{H}^{\otimes n}$) applied to the n-Qbit state $|0\rangle_n$:

$$\mathbf{H}^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}}\sum_{0\le x<2^n}|x\rangle_n. \qquad (2.5)$$

So if the initial state of the input register is $|0\rangle_n$ and we apply an n-fold Hadamard transformation to that register, its state becomes an equally weighted superposition of *all* possible n-Qbit inputs.

If we then apply $\mathbf{U}_f$ to that superposition, with 0 initially in the output register, then by linearity we get from (2.5) and (2.2)

$$\mathbf{U}_f(\mathbf{H}^{\otimes n}\otimes\mathbf{1}_m)(|0\rangle_n|0\rangle_m) = \frac{1}{2^{n/2}}\sum_{0\le x<2^n}\mathbf{U}_f(|x\rangle_n|0\rangle_m) = \frac{1}{2^{n/2}}\sum_{0\le x<2^n}|x\rangle_n|f(x)\rangle_m. \qquad (2.6)$$

This result contains an important part of the magic that underlies quantum computation.

If before letting $\mathbf{U}_f$ act, we merely apply a Hadamard transformation to every Qbit of the input register, initially in the standard state $|0\rangle_n$, the result of the computation is described by a state whose structure **cannot** be explicitly specified without knowing the result of all 2^n evaluations of the function f.

So if we have a mere hundred Qbits in the input register, initially all in the state $|0\rangle_{100}$ (and m more in the output register), if a hundred Hadamard gates act on the input register before the application of $\boldsymbol{U}_f$, then the form of the final state contains the results of $2^{100} \approx 10^{30}$ evaluations of the function f.

A billion billion trillion evaluations!

This *apparent* miracle is called ***quantum parallelism***.

But a major part of the miracle is *only apparent*.

I did not say that the result of the calculation is 2^n evaluations of f, though many practitioners of quantum computation are rather careless about making such a claim.

All I said (and all one can say) is that those explicit evaluations *occur in the form* of the state that describes the output of the computation.

Before drawing extravagant practical or only, as many practitioners of quantum computation are wont to do, metaphysical conclusions from quantum parallelism, it is important to remember that *when you have a collection of Qbits in a definite but unknown state, there is in general no way to find out what that state is*.

If there were a way to learn the state of such a set of Qbits, then I myself would join in the rhapsodic chorus.

Typical verses:

> "Where were all those calculations done? In parallel universes!"
>
> "The possibility of quantum computation has established the existence of the multiverse."
>
> "Quantum computation achieves its power by dividing the computational task among huge numbers of parallel worlds."

But there is no way to learn the state. (and, I add, there are NO parallel worlds anyway!!)

The *only* way to extract information from Qbits is to subject them to a *measurement*.

When we send all n+m Qbits through measurement gates, the Born rule tells us that if the state of the registers is (2.6), then with equal probability the result of measuring the Qbits in the input register will be any one of the values of x less than 2^n, while the result of measuring the Qbits in the output register will be the value of f for that *particular* value of x.

So by measuring the Qbits we can learn a single value of f as well as learning a single (random) x_0 at which f has that value.

After the measurement the state of the registers reduces to $|x_0\rangle|f(x_0)\rangle$ and we are *no longer* able to learn anything about the values of f for any other values of x.

So although we can learn something from the output of the "parallel computation", it is nothing more than what we would have learned had we simply run the computation starting with a classical state $|x\rangle$ in the input register, with the value of x chosen randomly.

To be sure, a hint of a miracle remains — hardly more than the smile of the Cheshire cat — in the fact that in the quantum case the random selection of the x, for which $f(x)$ can be learned, is only made *after* the computation has been carried out.

To assert that the selection was made before the computation was done, is to make the **same error** as asserting that a Qbit in a superposition of the states $|0\rangle$ and $|1\rangle$ is actually in one or the other of them, as discussed earlier.

This is a characteristic instance of what journalists like to call “quantum weirdness”, in that (a) it is indeed vexing to contemplate the fact that the choice of the value of x for which f can be learned is made only after — quite possibly long after — the computation has been finished, but (b) since that choice is inherently random — beyond anyone’s power to control in any way whatever — it does not matter for any practical purpose whether the selection was made miraculously after or boringly before the calculation was executed.

If, of course, there were an easy way to to make copies of the output state prior to making the measurement, without running the whole computation over again, then one could, with high probability, learn the values of f for several different (random) values of x.

But such copying is **prohibited** by an elementary result called the “no-cloning theorem”, which states that there is no such duplication procedure: there is no unitary transformation that can take a state $|\psi\rangle_n|0\rangle_n$ into the state $|\psi\rangle_n|\psi\rangle_n$ for arbitrary $|\psi\rangle_n$.

The no-cloning theorem is an immediate consequence of *linearity*.

If

$$\mathbf{U}\big(|\psi\rangle|0\rangle\big) = |\psi\rangle|\psi\rangle \quad \text{and} \quad \mathbf{U}\big(|\phi\rangle|0\rangle\big) = |\phi\rangle|\phi\rangle, \tag{2.7}$$

then it *follows* from linearity that

$$\mathbf{U}\big(a|\psi\rangle + b|\phi\rangle\big)|0\rangle = a\mathbf{U}|\psi\rangle|0\rangle + b\mathbf{U}|\phi\rangle|0\rangle = a|\psi\rangle|\psi\rangle + b|\phi\rangle|\phi\rangle. \tag{2.8}$$

But if **U** cloned arbitrary inputs, we would have

$$\begin{aligned}\mathbf{U}\big(a|\psi\rangle + b|\phi\rangle\big)|0\rangle &= \big(a|\psi\rangle + b|\phi\rangle\big)\big(a|\psi\rangle + b|\phi\rangle\big) \\ &= a^2|\psi\rangle|\psi\rangle + b^2|\phi\rangle|\phi\rangle + ab|\psi\rangle|\phi\rangle + ab|\phi\rangle|\psi\rangle,\end{aligned} \tag{2.9}$$

which differs from (2.8) unless either a or b is zero.

Surprisingly, this theorem was not proved until half a century after the discovery of quantum mechanics, presumably because it took that long for it to occur to somebody that it was an interesting proposition to formulate.

Of course the ability to clone to a reasonable degree of approximation would also be useful.

But even this is impossible.

Suppose **U** approximately cloned both $|\phi\rangle$ and $|\psi\rangle$:

$$\mathbf{U}\big(|\psi\rangle|0\rangle\big) \approx |\psi\rangle|\psi\rangle \quad \text{and} \quad \mathbf{U}\big(|\phi\rangle|0\rangle\big) \approx |\phi\rangle|\phi\rangle. \tag{2.10}$$

Then since unitary transformations preserve inner products since the inner product of a tensor product of states is the (ordinary) product of their inner products, and since $\langle 0|0\rangle = 1$, it follows from (2.10) that

$$\langle\phi|\psi\rangle \approx \langle\phi|\psi\rangle^2. \tag{2.11}$$

But this requires that $\langle\phi|\psi\rangle$ is either close to 1 or close to 0.

Hence a unitary transformation can come close to cloning both of two states $|\psi\rangle$ and $|\phi\rangle$ only if the states are very nearly the same, or very close to being orthogonal.

In all other cases at *least* one of the two states will be badly copied.

If this were the full story, nobody would be interested in quantum computation except philosophers.

The Department of Defense and the National Security Agency are interested because there are more *clever* things one can do.

Typically these involve applying *additional* unitary transformations to one or both of the input and output registers before measuring the Qbits, sometimes applying such unitaries both before and after applying $\mathbf{U}_f$.

These transformations are cleverly chosen so that when one finally does measure the Qbits, one extracts useful information about relations between the values of f for several different values of x, which a classical computer could only get by making several *independent* evaluations.

The price one pays for this relational information is to *lose* the possibility of learning the actual value f(x) for any individual x whatever.

This tradeoff of one kind of information for another is typical of quantum computation, and typical of quantum physics in general, where it is called the *uncertainty principle*.

The principle was first enunciated by Werner Heisenberg in the context of mechanical information — the position of a particle vs. its momentum.

So it is wrong and deeply misleading to say that in the process that assigns the state (2.6) to the Qbits, the quantum computer has evaluated the function f(x) for all x in the range $0 \le x < 2^n$.

$$\mathbf{U}_f\left(\mathbf{H}^{\otimes n} \otimes \mathbf{1}_m\right)\left(|0\rangle_n|0\rangle_m\right) = \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} \mathbf{U}_f\left(|x\rangle_n|0\rangle_m\right) = \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} |x\rangle_n|f(x)\rangle_m. \tag{2.6}$$

Such assertions are based on the *mistaken* view that the quantum state encodes a property inherent in the Qbits.

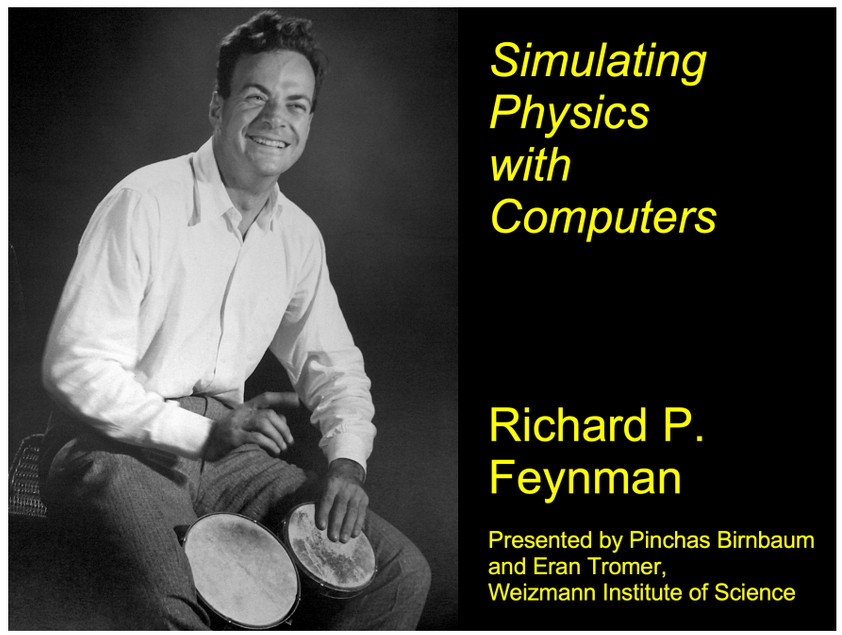
All the state encodes are the *possibilities* available for the extraction of information from those Qbits.

You might keep this in mind as we now examine some of the specific ways in which this nevertheless permits a quantum computer to perform tricks that no classical computer can accomplish.

B. Deutsch's Problem

Deutsch's problem is the simplest example of a quantum tradeoff that sacrifices particular information to acquire relational information.

A crude version of it appeared in a 1985 paper by David Deutsch that, together with a 1982 paper by Richard Feynman(on website), launched the whole field.



In that early version the trick could only be successfully executed half the time.

It took a while for people to realize that the trick could be accomplished every single time.

Here is how it works:

Let both input and output registers contain only a single Qbit, so we are exploring functions f that take a single bit into a single bit.

There are two rather different ways to think about such functions:

(1) The first way is to note that there are just four such functions, as shown in Table 2.1.

	$x = 0$	$x = 1$
f_0	0	0
f_1	0	1
f_2	1	0
f_3	1	1

TABLE 2.1: The 4 distinct functions $f_j(x)$ that take one bit into one bit.

Suppose we are given a black box that calculates one of these four functions in the usual quantum computational format, by performing the unitary transformation

$$\mathbf{U}_f\big(|x\rangle|y\rangle\big) = |x\rangle|y \oplus f(x)\rangle, \tag{2.12}$$

where the state on the left is that of the 1-Qbit input register (i), and the state on the right is that of the 1-Qbit output register (o).

	$x=0$	$x=1$
f_0	0	0
f_1	0	1
f_2	1	0
f_3	1	1

$$\mathbf{U}_f\big(\,|x\rangle|y\rangle\big) = |x\rangle|y \oplus f(x)\rangle,$$

Using the forms in Table 2.1 and the explicit structure (2.12) of $\mathbf{U}_f$ you can easily confirm that

$$\mathbf{U}_{f_0} = \mathbf{1}, \quad \mathbf{U}_{f_1} = \mathbf{C}_{io}, \quad \mathbf{U}_{f_2} = \mathbf{C}_{io}\mathbf{X}_o, \quad \mathbf{U}_{f_3} = \mathbf{X}_o, \tag{2.13}$$

where $\mathbf{1}$ is the (2-Qbit) unit operator, $\mathbf{C}_{io}$ is the controlled-NOT with the input Qbit as control and the output as target, $\mathbf{X}_i$ flips only the input register, and $\mathbf{X}_o$ flips only the output register.

These possibilities are illustrated in the circuit diagrams of Figure 2.1.

Figure 2.1

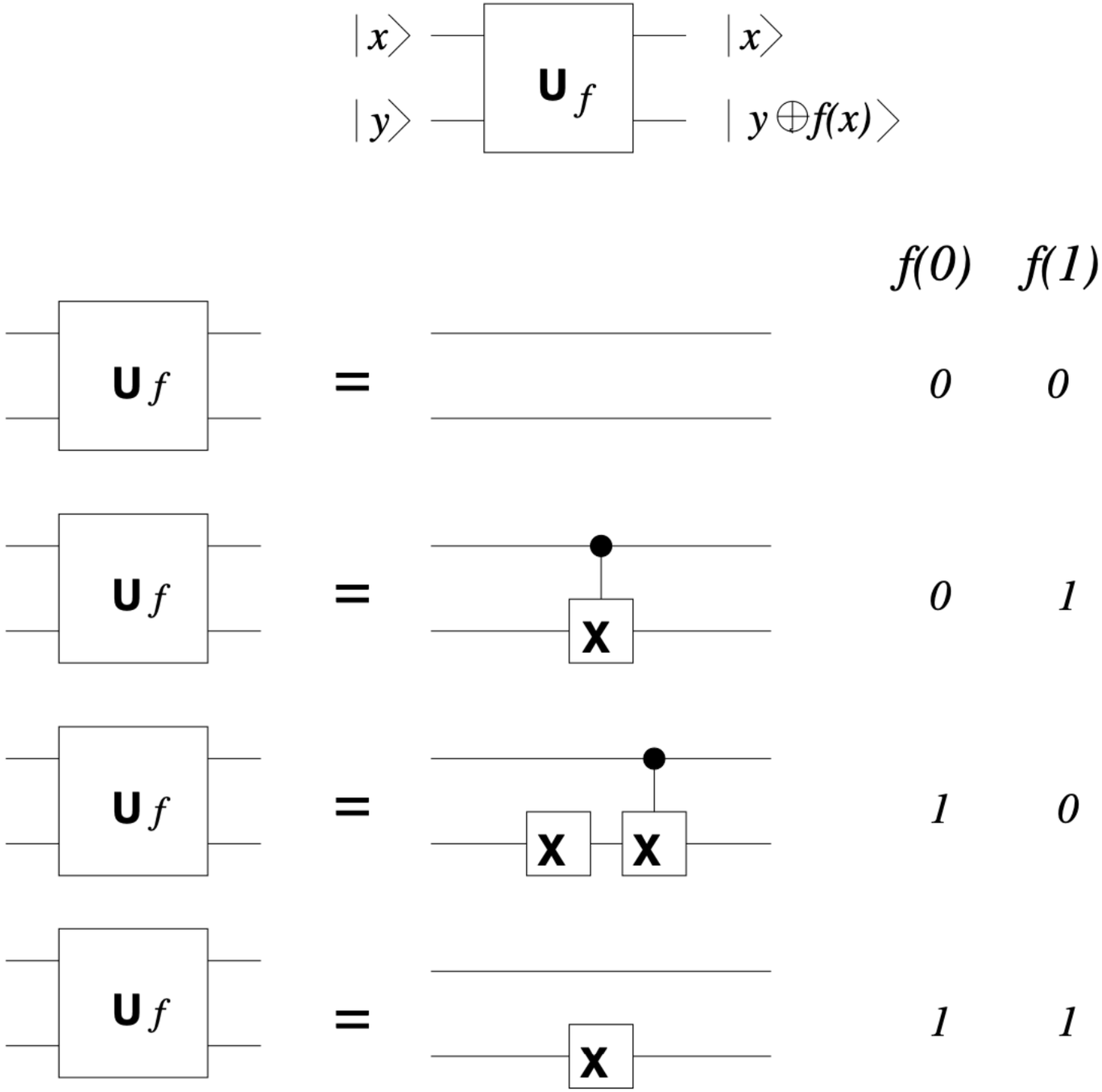


Figure 2.1. A way to construct, with elementary gates, each of the black boxes $\mathbf{U}_f$ that realize the four possible functions f that appear in Deutsch's problem. In case 00 f is identically 0 and it is evident from the general form at the top of the figure that $\mathbf{U}_f$ acts as the identity. In case 01 f acts as the identity so $\mathbf{U}_f$ acts as cNOT with the input register as the control bit. In case 10 f interchanges 0 and 1, so $\mathbf{U}_f$ flips the target bit if and only if the control bit is 0. This is equivalent to combining a cNOT with an unconditional flip of the target bit. And in case 11 f is identically 1, and the effect of $\mathbf{U}_f$ is just to flip the output register, independent of the input register. Note the diagrammatic convention for controlled operations: the control Qbit is represented by the wire with the black dot on it; the target Qbit is connected to the control by a vertical line ending in a box containing the controlled operation. An alternative representation for cNOT appears in Figure 2.7.

Suppose we are given a black box that executes $\mathbf{U}_f$ for one of the four functions, but are not told which of the four operations (2.13) the box carries out.

We can, of course, find out by letting the black box act twice — first on $|0\rangle|0\rangle$ and then on $|1\rangle|0\rangle$.

But suppose we are only allowed to let the box act once.

What can we learn about f ?

In a classical computer, where we are effectively restricted to letting the black box act on Qbits in one of the four computational basis states, we can learn either the value of $f(0)$ (if we let $\mathbf{U}_f$ act on either $|0\rangle|0\rangle$ or $|0\rangle|1\rangle$) or the value of $f(1)$ (if we let $\mathbf{U}_f$ act on either $|1\rangle|0\rangle$ or $|1\rangle|1\rangle$).

If we chose to learn the value of $f(0)$. then we can restrict f to being either f_0 or f_1 (if $f(0) = 0$) and we can restrict f to being either f_2 or f_3 (if $f(0) = 1$).

If we chose to learn the value of $f(1)$, then we can restrict f to being either f_0 or f_2 (if $f(1) = 0$) or to being either f_1 or f_3 (if $f(1) = 1$).

Suppose, however, we want to learn whether f is constant ($f(0) = f(1)$, satisfied by f_0 and f_3) or not constant ($f\,0) \neq f(1)$, satisfied by f_1 and f_2).

We then have no choice with a classical computer but to evaluate both $f(0)$ and $f(1)$ and compare them.

In this way we determine whether or not f is constant, but we have to extract complete information about f to do so.

We have to run $\mathbf{U}_f$ twice.

Remarkably, it turns out that with a quantum computer we do not have to run $\mathbf{U}_f$ twice to determine whether or not f is constant.

We can do this in a single run.

Interestingly, when we do this we learn nothing whatever about the individual values of $f(0)$ and $f(1)$, but we are nevertheless able to answer the question about their relative values: whether or not they are the same.

Thus we get less information than we get in answering the question with a classical computer, but by renouncing the possibility of acquiring that part of the information which is irrelevant to the question we wish to answer, we can get the answer with only a single application of the black box.

(2) There is a second way to look at Deutsch's problem, which gives it some nontrivial mathematical content.

One can think of x as specifying a choice of two different inputs to an elaborate subroutine that requires many additional Qbits, and one can think of $f(x)$ as characterizing a two-valued property of the output of that subroutine.

For example $f(x)$ might be the value of the millionth bit in the binary expansion of $\sqrt{2+x}$ so that $f(0)$ is the millionth bit in the expansion of $\sqrt{2}$ while $f(1)$ is the millionth bit of $\sqrt{3}$.

In this case the input register feeds data into the subroutine and the subroutine reports back to the output register.

In the course of the calculation the input and output registers will in general become entangled with the additional Qbits used by the subroutine.

If the entanglement persists to the end of the calculation, the input and output registers will have no final states of their own, and it will be impossible to describe the computational process as the simple unitary transformation (2.1).

We shall see later, however, that it is possible to set things up so that at the end of the computation the additional Qbits required for the subroutine are *no longer entangled* with the input and output registers, and that the additional Qbits can indeed be *ignored*, with the simple linear transformation (2.1) correctly characterizing the net effect of the computation on those two registers.

Under interpretation (1) of Deutsch's problem, the question of whether f is or is not constant is the somewhat artificial question of learning something about the nature of the black-box that executes $\mathbf{U}_f$ without actually opening it up and looking inside.

Under interpretation (2) it becomes the *nontrivial* question of whether the millionth bits of $\sqrt{2}$ and $\sqrt{3}$ agree or disagree.

Under either interpretation, to answer the question with a classical computer we can do no better than to run the black box twice, with both 0 and 1 as inputs, and compare the two outputs.

In the quantum case we could try the *standard trick*, preparing the input register in the superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

The final state of the 1-Qbit input and output registers would then be

$$\mathbf{U}_f\left(\mathbf{H}\otimes\mathbf{1}\right)\left(|0\rangle|0\rangle\right) = \tfrac{1}{\sqrt{2}}|0\rangle|f(0)\rangle + \tfrac{1}{\sqrt{2}}|1\rangle|f(1)\rangle, \qquad (2.14)$$

as described in (2.6).

If we measured the input and output registers we could learn the millionth bit of either $\sqrt{2}$ or $\sqrt{3}$ (and learn which it is we have learned).

The choice of which we did learn would be completely random, so this procedure offers no improvement on the classical situation.

It was first noticed, however, that there are unitary transformations one can apply to the state (2.14) before carrying out the measurement, that, depending on the outcome, enable you half the time to state with assurance whether or not $f(0) = f(1)$.

Some time after that, it was realized that you can always answer the question, provided you apply appropriate unitary transformations before as well as after running the computation.

Here is how the trick is done:

To get the output (2.14) we took the input to $\mathbf{U}_f$ to be the state

$$\left(\mathbf{H}\otimes\mathbf{1}\right)\left(|0\rangle|0\rangle\right). \qquad (2.15)$$

Instead of doing this, we again start with both input and output registers in the state $|0\rangle$, but then we apply the NOT operation $\mathbf{X}$ to both registers, followed by an application of the Hadamard transformation to both.

Since $\mathbf{X}|0\rangle = |1\rangle$ and $\mathbf{H}|1\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$, the input to $\mathbf{U}_f$ is now described by the state

$$(\mathbf{H}\otimes\mathbf{H})(\mathbf{X}\otimes\mathbf{X})(|0\rangle|0\rangle) = (\mathbf{H}\otimes\mathbf{H})(|1\rangle|1\rangle) = \left(\tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle\right)\left(\tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle\right)$$

$$= \tfrac{1}{2}\Big(|0\rangle|0\rangle - |1\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|1\rangle\Big). \qquad (2.16)$$

If we take the state (2.16) as input to $\mathbf{U}_f$, then by linearity the resulting state is

$$= \tfrac{1}{2}\Big(\mathbf{U}_f(|0\rangle|0\rangle) - \mathbf{U}_f(|1\rangle|0\rangle) - \mathbf{U}_f(|0\rangle|1\rangle) + \mathbf{U}_f(|1\rangle|1\rangle)\Big). \qquad (2.17)$$

It follows from the explicit form (2.12) of the action of $\mathbf{U}_f$ on the computational basis states that this is simply

$$\tfrac{1}{2}\Big(|0\rangle|f(0)\rangle - |1\rangle|f(1)\rangle - |0\rangle|\overline{f}(0)\rangle + |1\rangle|\overline{f}(1)\rangle\Big), \qquad (2.18)$$

where, as earlier, $\overline{x} = 1 \oplus x$ so that $\overline{1} = 0$ and $\overline{0} = 1$, and $\overline{f}(x) = 1 \oplus f(x)$.

So if $f(0) = f(1)$ the output state (2.18) is

$$\tfrac{1}{2}\Big(|0\rangle - |1\rangle\Big)\Big(|f(0)\rangle - |\overline{f}(0)\rangle\Big), \quad f(0) = f(1), \tag{2.19}$$

but if $f(0) \neq f(1)$ then $f(1) = \overline{f}(0)$, $\overline{f}(1) = f(0)$, and the output state (2.18) becomes

$$\tfrac{1}{2}\Big(|0\rangle + |1\rangle\Big)\Big(|f(0)\rangle - |\overline{f}(0)\rangle\Big), \quad f(0) \neq f(1). \tag{2.20}$$

If, finally, we apply a Hadmard transformation to the input register these become

$$|1\rangle \tfrac{1}{\sqrt{2}}\Big(|f(0)\rangle - |\overline{f}(0)\rangle\Big), \quad f(0) = f(1), \tag{2.21}$$

$$|0\rangle \tfrac{1}{\sqrt{2}}\Big(|f(0)\rangle - |\overline{f}(0)\rangle\Big), \quad f(0) \neq f(1). \tag{2.22}$$

Putting together all the operations in a form we can compare with the more straightforward computation (2.14), we have

$$(\mathbf{H}\otimes\mathbf{1})\mathbf{U}_f(\mathbf{H}\otimes\mathbf{H})(\mathbf{X}\otimes\mathbf{X})(|0\rangle|0\rangle) = \begin{cases} |1\rangle\frac{1}{\sqrt{2}}\Big(|f(0)\rangle - |\overline{f}(0)\rangle\Big), & f(0)=f(1) \\ |0\rangle\frac{1}{\sqrt{2}}\Big(|f(0)\rangle - |\overline{f}(0)\rangle\Big), & f(0)\neq f(1). \end{cases} \tag{2.23}$$

Thus the state of the input register has ended up as $|1\rangle$ or $|0\rangle$ depending on whether or not $f(0) = f(1)$, so by measuring the input register we can indeed answer the question of whether $f(0)$ and $f(1)$ are or are not the same!

Notice also that in either case the output register is left in the state $\frac{1}{\sqrt{2}}|f(0)\rangle - \frac{1}{\sqrt{2}}|\overline{f}(0)\rangle$.

Because both terms in the superposition have amplitudes with exactly the same magnitude, if one measures the output register the result is equally likely to be $f(0)$ or $\overline{f}(0)$, so one learns absolutely *nothing* about the actual value of $f(0)$.

The output register contains *no useful information* at all.

Another way to put it is that the final state of the output register is $\pm\frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)$ depending on whether $f(0) = 0$ or $f(0) = 1$.

Since a change in the overall sign of a state (or, more generally, the presence of an overall complex factor of modulus 1) has no effect on the statistical distribution of measurement outcomes, there is no way to distinguish between these two cases.

Thus the price one has paid to learn whether $f(0)$ and $f(1)$ are or are not the same, is the loss of any information whatever about the actual value of either one of them.

One has still eliminated only two of the four possible forms for the function f.

What the quantum computer has given us is the ability to make this particular discrimination with just a single invocation of the black box, which a classical computer cannot do.

There is a rather neat circuit-theoretic way of seeing why this trick enables one to learn whether or not $f(0) = f(1)$ in just one application of $\mathbf{U}_f$, without going through any of the above algebraic manipulations.

This quite different way of looking at Deutsch's problem is illustrated in Figures 2.1-2.3.

Figure 2.1

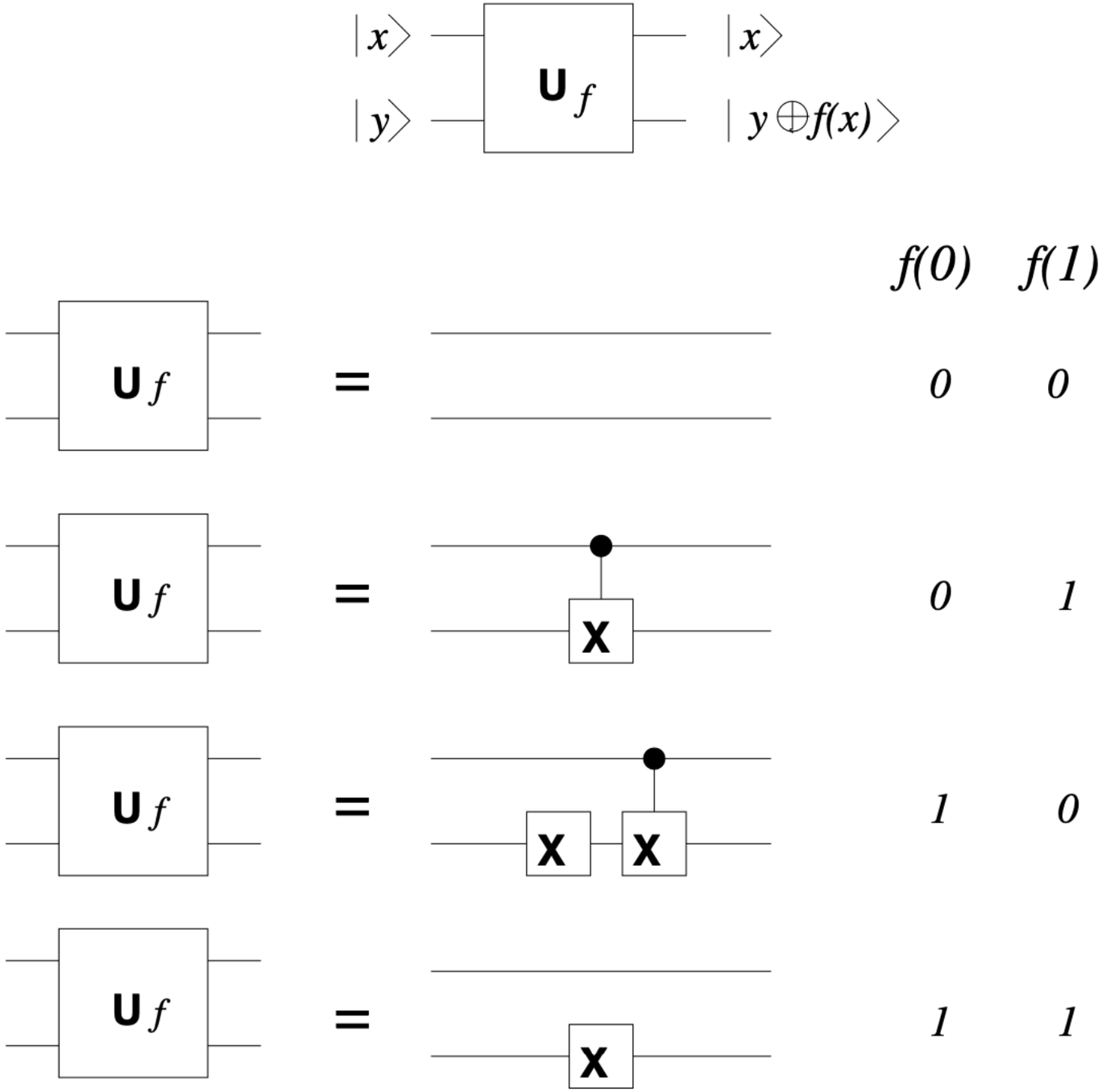


Figure 2.1. A way to construct, with elementary gates, each of the black boxes $\mathbf{U}_f$ that realize the four possible functions f that appear in Deutsch's problem. In case 00 f is identically 0 and it is evident from the general form at the top of the figure that $\mathbf{U}_f$ acts as the identity. In case 01 f acts as the identity so $\mathbf{U}_f$ acts as cNOT with the input register as the control bit. In case 10 f interchanges 0 and 1, so $\mathbf{U}_f$ flips the target bit if and only if the control bit is 0. This is equivalent to combining a cNOT with an unconditional flip of the target bit. And in case 11 f is identically 1, and the effect of $\mathbf{U}_f$ is just to flip the output register, independent of the input register. Note the diagrammatic convention for controlled operations: the control Qbit is represented by the wire with the black dot on it; the target Qbit is connected to the control by a vertical line ending in a box containing the controlled operation. An alternative representation for cNOT appears in Figure 2.7.

Figure 2.2

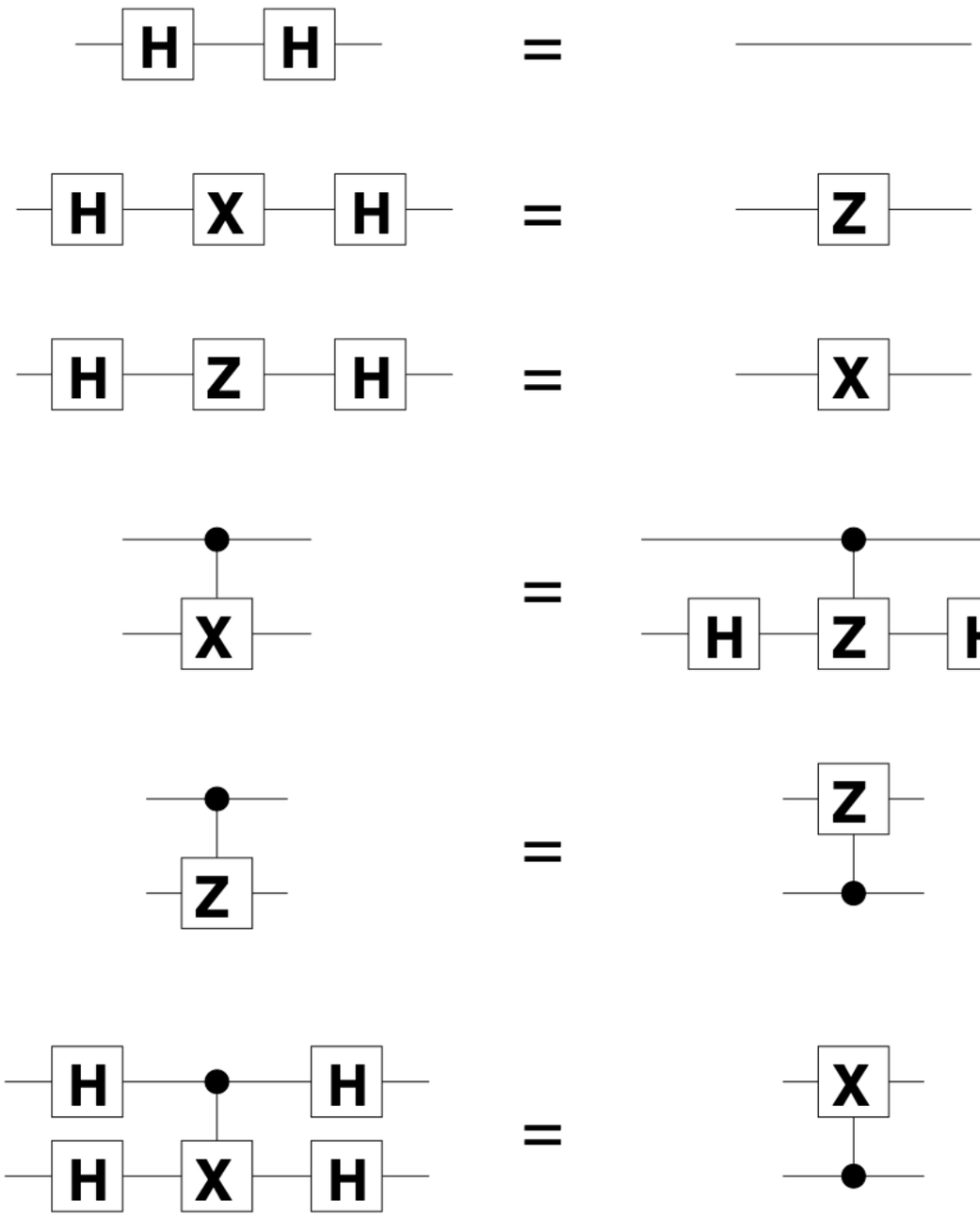


Figure 2.2. Some elementary circuit identities that are useful in seeing why it is possible to determine whether or not $f(0) = f(1)$ with a single invocation of $\mathbf{U}_f$. (a) $\mathbf{H}^2 = \mathbf{1}$. (b) $\mathbf{HXH} = \mathbf{Z}$. (c) A consequence of (a) and (b). (d) A consequence of (a) and (c). (e) The action of the controlled-Z gate does not depend on which Qbit is control and which is target, since it acts as the identity on each of the state $|00\rangle, |01\rangle, |10\rangle$ and multiplies the state $|11\rangle$ by -1. (f) This follows from (d), (a), and (e).

Figure 2.3

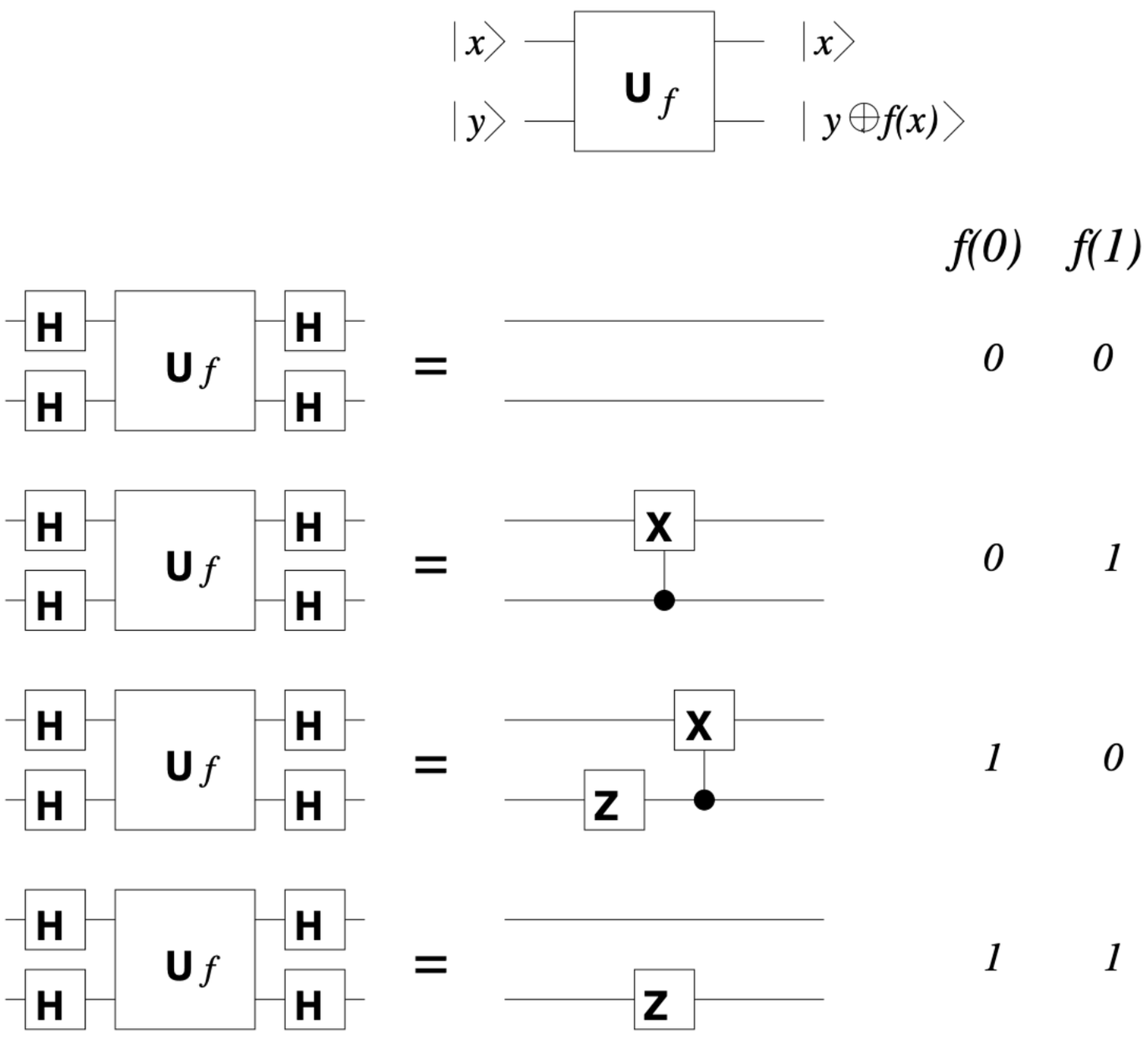


Figure 2.3. We can get the action of $\mathbf{U}_f$, when it is preceded and followed by Hadamards on both Qbits, by applying the appropriate identities of Figure 2.2 to the diagrams of Figure 2.1. Case 00 is unchanged because of Figure 2(a). In case 01 the target and control Qbits of the cNOT are interchanged because of Figure 2.2(f). The form in case 10 follows from the corresponding form in Figure 1 because of Figure 2.2(f) and 2.2(b). And the form in case 11 follows from Figure 2.2(a) and 2.2(b). If the initial state of the output register (lower wire) is $|1\rangle$ and the initial state of the input register is either of the two computational-basis states, then the initial state of the input register will be unchanged in cases 00 and 11, and flipped in cases 01 and 10, so by measuring the input register after the action of $\mathbf{H} \otimes \mathbf{H}\, \mathbf{U}_f\, \mathbf{H} \otimes \mathbf{H}$ one can determine whether or not f(0) = f(1).

When one thinks of applying this to learn whether the millionth bits of $\sqrt{2}$ and $\sqrt{3}$ are the same or different, as in the second interpretation of Deutsch's problem, it is really quite startling that one can do this with no more effort (except for a simple modification of the initial and final states) than one uses to calculate the millionth bit of either $\sqrt{2}$ or $\sqrt{3}$.

In this case, however, there is an *irritating catch*, which we return to at the end of the next section.

C. Why additional subroutine Qbits needn't mess things up.

Now that we have a specific example of a quantum computation to keep in mind, we can address an important issue mentioned earlier.

The computational process generally requires the use of many Qbits besides the n+m in the input and output registers.

In the second way of thinking about Deutsch's problem, it may need to use a great many more.

The action of the computer is then described by a unitary transformation $\mathbf{W}_f$ that acts on the space associated with ***all*** the Qbits: those in the input and output registers, together with the r additional Qbits used in evaluating the function f.

Only under very special circumstances will this global unitary transformation $\mathbf{W}_f$ on all n + m + r Qbits induce a transformation on the input and output registers that can be be described by a unitary transformation $\mathbf{U}_f$ that acts only on those two registers, as in (2.1).

In general the input and output registers will become entangled with the states of the additional r Qbits, and will not even have states of their own.

But if the action of the computer on all n + m + r Qbits has a very special form, then the input and output registers can indeed end up in states of their own, which are related to their initial states through the desired unitary transformation $\mathbf{U}_f$.

Let the additional r Qbits start off in some standard initial state $|\psi\rangle_r$, so that the initial state of input register, output register, and additional Qbits is

$$|\Psi\rangle_{n+m+r} = |x\rangle_n |y\rangle_m |\psi\rangle_r . \qquad (2.24)$$

Although the r additional Qbits may well become entangled with those in the input and output registers in the course of the calculation — indeed they will have to if they are to serve any useful purpose — we require that when the calculation is finished the final state of the computer must be of the form

$$\mathbf{W}|\Psi\rangle_{n+m+r} = |x\rangle_n |y \oplus f(x)\rangle_m |\phi\rangle_r, \tag{2.25}$$

where the additional r Qbits have a state $|\phi\rangle_r$ of their own, which is independent of the initial state of the input and output registers (and therefore offers no hint of the value of x for which *f*(x) was actually calculated).

Because the final state (2.25) is an unentangled product state between the additional Qbits and the input and output registers, the two registers do indeed have a state of their own at the end of the computation.

And because the initial and final states $|\psi\rangle_r$ and $|\phi\rangle_r$ of the additional Qbits are independent of the initial state of the input and output registers, one easily verifies that the final states of the input and output registers will indeed be related to their initial states by a transformation $\mathbf{U}_f$ that is linear on their (n + m)-Qbit subspace, as a consequence of the linearity of $\mathbf{W}_f$ on the whole (n+m+r)-Qbit subspace.

That the linear transformation $\mathbf{U}_f$ is also unitary then follows from the fact that it permutes the states of the computational basis.

Therefore we can indeed ignore the additional r Qbits needed to compute the function *f* provided both the initial and the final states of the additional Qbits are entirely independent of the initial state of the input and output registers.

This independence can be achieved by taking advantage of the fact that unitary transformations are *reversible*.

It is this need to disentangle additional registers from the input and output registers that is the fundamental reason why quantum computation must be reversible.

To do the trick, begin the computation by applying a unitary transformation **V** that acts only on the input register and the r additional Qbits, doing nothing to the output register.

This unitary transformation is designed to construct $f(x)$, given x.

Next change the y initially in the output register to $y \oplus f(x)$ (as (2.1) requires) without altering any Qbits outside the output register, by using m cNOT gates that combine to make up a unitary transformation $\mathbf{C}_m$.

The m control bits are those among the n + r that represent the result of the computation $f(x)$; the m target bits are the ones in the corresponding positions of the output register.

Since the state of the n + r Qbits is not altered by the application of $\mathbf{C}_m$, we can then apply the inverse transformation $\mathbf{V}^\dagger$ to restore them to their original state.

In this way we have produced the required unitary transformation **W**, with the final states of the r additional Qbits being the same as their initial state.

This whole construction is illustrated in the circuit diagrams of Figures 2.4-2.7.

Figure 2.4

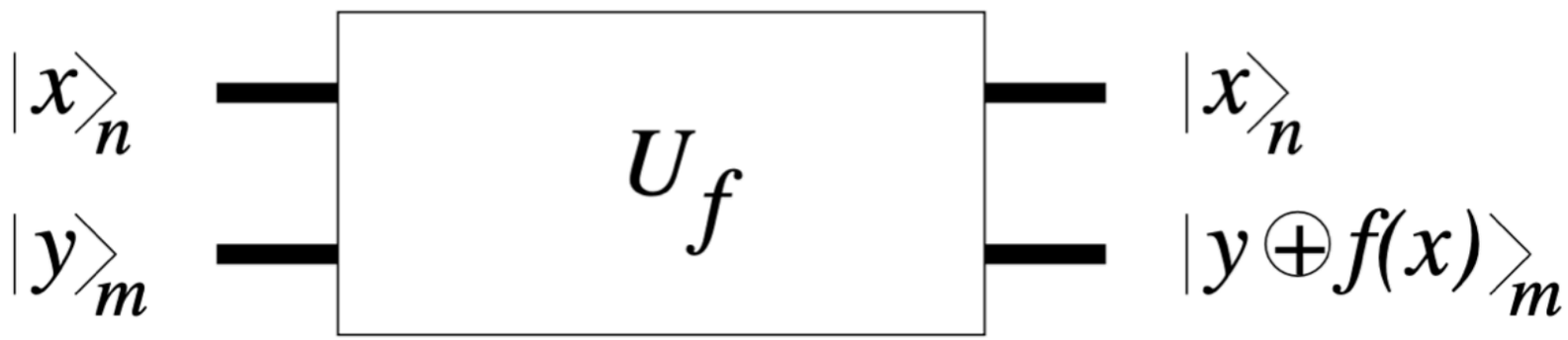


Figure 2.4. Schematic representation of the standard unitary transformation $\mathbf{U}_f$ for evaluating a function f taking a number $0 \leq x < 2n$ into a number $0 \leq f(x) < 2^m$. The heavy horizontal lines represent multiple Qbit inputs. In order for the computation to be reversible even when f is not one-to-one, two multi-Qbit registers are used. After the computation is done the state of the input register again represents x and the state of the output register represents $y \oplus f(x)$, where $\oplus$ is the operation of bitwise modulo 2 addition (without carry). If the output register initially represents 0, after the computation it represents $f(x)$.

Figure 2.5

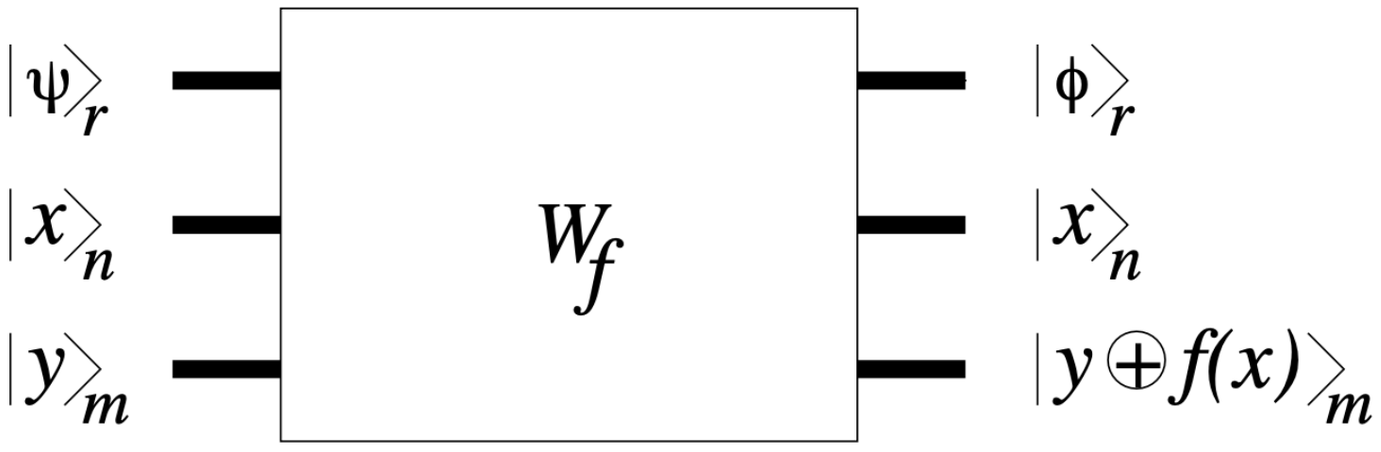


Figure 2.5. A more realistic picture of the computation represented in Figure 2.4. Many additional Qbits may be needed to carry out the calculation. These are represented by an r-Qbit bar in addition to the n- and m-Qbit bars representing the input and output registers. The computation is actually executed by a unitary transformation $\mathbf{W}_f$ that acts on the larger space of all n+ m+ r Qbits. The representation of Figure 2.4 is correct only if the action of this larger unitary transformation W_f on the input and output registers alone can be represented by a unitary transformation $\mathbf{U}_f$. As discussed in the notes, this will be true, for arbitrary initial superpositions of computational-basis states in the input and output registers, provided the action of $\mathbf{W}_f$ on the residual r Qbits is to take them from an initial pure state $|\psi\rangle_r$ to a final pure state $|\phi\rangle_r$ that is independent of the initial contents of the input and output registers, as depicted in the figure.

Figure 2.6

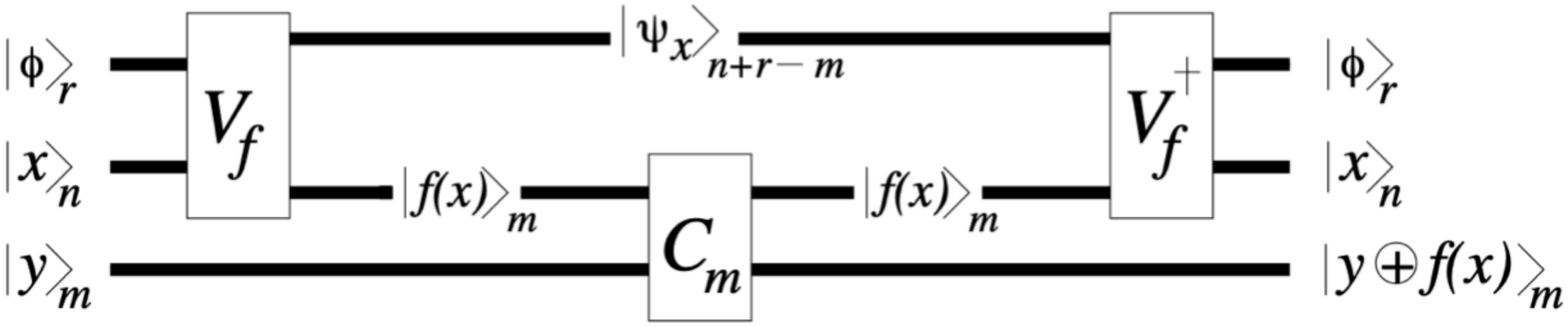
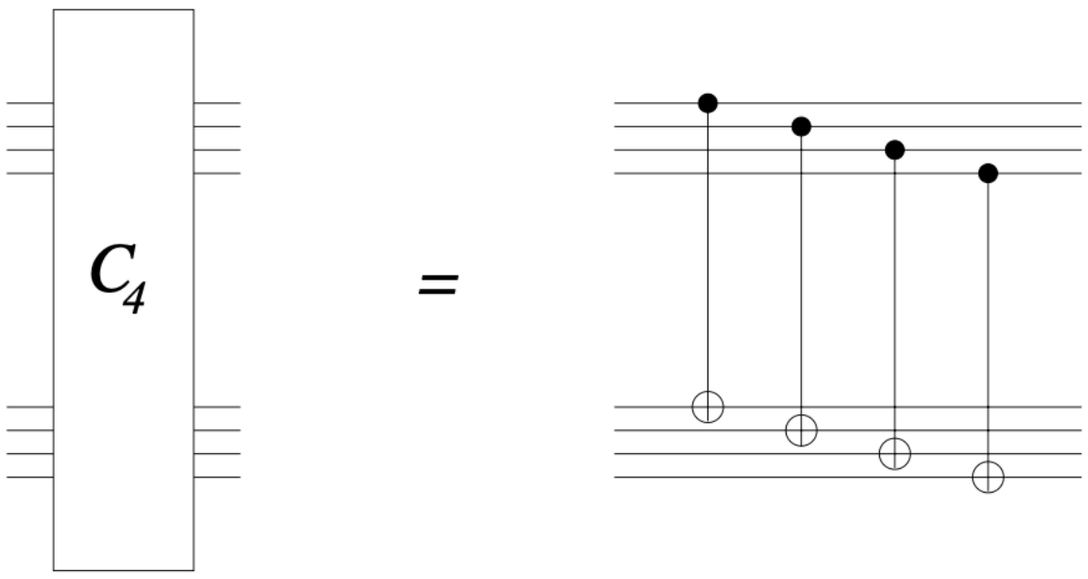


Figure 2.6. A more detailed view of the structure of the unitary transformation $\mathbf{W}_f$ of Figure 2.5. Algebraically, $\mathbf{W}_f = \mathbf{V}^{\dagger}_f \mathbf{C}_{\mathrm{m}} \mathbf{V}_f$. First a unitary transformation $\mathbf{V}_f$ acts only on the input register and residual Qbits (i.e. it acts as the identity on the output register), transforming them into a state that contains among its n+r Qbits an m-Qbit subset that represents the result of the calculation, f(x). At this stage the remaining n+ r− mQbits are in an entangled state. Next an m controlled-NOT transformations (described in more detail in Figure 2.7) act only on the m Qbits representing f(x) and the m Qbits of the output register, leaving the former m unchanged but changing the number represented by the latter m from y to y⊕ f(x). Finally, the inverse $\mathbf{V}^{\dagger}_f$ of $\mathbf{V}_f$ is applied to the residual Qbits and the input register to restore them to their (unentangled) initial states.

Figure 2.7



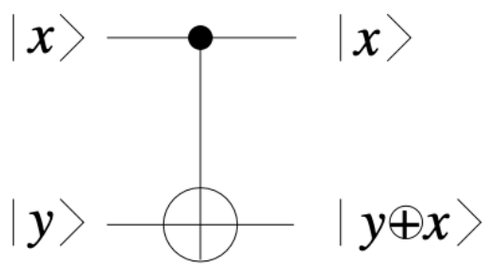


Figure 2.7. Not a simple Gregorian chant, but a more detailed picture of the $\mathbf{C}_m$ unitary transformation in Figure 2.6, for the case m = 4. Each of the input and output bars contains 4 Qbits, represented by sets of 4 thin lines (wires). Four different 2-Qbit controlled-not gates link the 4 upper wires representing $f(x)$ to the four lower wires representing the corresponding positions in the output register. The action of a single such cNOT gate is shown in the lower part of the figure. Note the alternative convention for a cNOT gate: the black dot on the wire representing the control Qbit is connected by a vertical line to an open circle on the wire representing the target Qbit. The alternative convention (used above in Figures 2.1-2.3) replaces the open circle by a square box containing the NOT operator **X** that may act on the target Qbit. The advantage of the circle representation is that it suggests the symbol ⊕ that represents the XOR operation, and that it is easier to draw quickly on a blackboard. The advantage of using **X** is that it makes the algebraic relations more evident when NOT operations **X**, **Z** operations, or controlled-Z operations also appear, and that it follows the same form as is used for all other controlled unitaries.

The need for a procedure like this negates some of the hype one encounters in discussions of Deutsch's problem.

It is often said that by using a quantum computer one can learn whether or not $f(x) = f(y)$ in no more time than it takes to perform a single evaluation of f.

This is true only under the first, arithmetically uninteresting, view of Deutsch's problem.

If, however, one is thinking of f as a function of mathematical interest evaluated by an elaborate subroutine, then to evaluate f for a single value of x there is no need to undo the effect of the unitary transform $\mathbf{V}$ on the additional registers.

But for the trick that determines whether or not $f(x) = f(y)$ it is absolutely essential to apply $\mathbf{V}^\dagger$ to undo the effect of $\mathbf{V}$, which doubles the time of the computation.

This misrepresentation of the situation is not entirely dishonorable, since in almost all other examples the speed-up is considerably more than a factor of two, and the necessary doubling of computational time is an insignificant price to pay.

Illustration of steps in Deutsch Procedure:

Deutsch's Algorithm

SETUP

$$|\psi_0\rangle_{1,1} \leftarrow |0\rangle \otimes |1\rangle$$

STEP 1

$$|\psi_1\rangle_{1,1} \leftarrow \boldsymbol{H}^{\otimes 2}\left(|\psi_0\rangle_{1,1}\right)$$

$$|\psi_1\rangle_{1,1} = \boldsymbol{H}^{\otimes 2}(|0\rangle \otimes |1\rangle) = (\boldsymbol{H}|0\rangle \otimes \boldsymbol{H}|1\rangle) = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = |+\rangle \otimes |-\rangle$$

STEP 2

$$|\psi_2\rangle_{1,1} \leftarrow \boldsymbol{O}_f\left(|\psi_1\rangle_{1,1}\right)$$

$$|\psi_2\rangle_{1,1} = \boldsymbol{O}_f\left(|\psi_1\rangle_{1,1}\right) = \boldsymbol{O}_f\left(|+\rangle \otimes |-\rangle\right) = \boldsymbol{O}_f\left[\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes |-\rangle\right]$$

$$= \frac{\boldsymbol{O}_f\left(|0\rangle \otimes |-\rangle\right) + \boldsymbol{O}_f\left(|1\rangle \otimes |-\rangle\right)}{\sqrt{2}} = \frac{(-1)^{f(0)}|0\rangle \otimes |-\rangle + (-1)^{f(1)}|1\rangle \otimes |-\rangle}{\sqrt{2}}$$

$$= \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right) \otimes |-\rangle$$

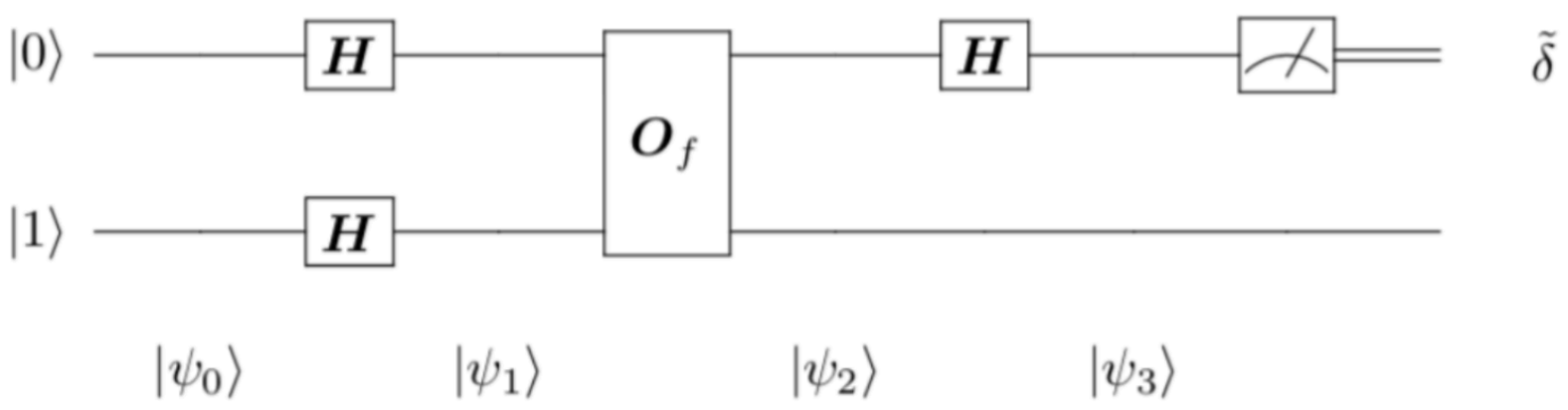
STEP 3

$$|\psi_3\rangle_{1,1} \leftarrow (\boldsymbol{H} \otimes \boldsymbol{I})\left(|\psi_2\rangle_{1,1}\right)$$

$$
\begin{aligned}
|\psi_3\rangle_{1,1} &= (\boldsymbol{H}\otimes\boldsymbol{I})\left(|\psi_2\rangle_{1,1}\right) = (\boldsymbol{H}\otimes\boldsymbol{I})\left[\left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right)\otimes|-\rangle\right] \\
&= \left(\frac{(-1)^{f(0)}\boldsymbol{H}|0\rangle + (-1)^{f(1)}\boldsymbol{H}|1\rangle}{\sqrt{2}}\right)\otimes|-\rangle \\
&= \left(\frac{(-1)^{f(0)}|+\rangle + (-1)^{f(1)}|-\rangle}{\sqrt{2}}\right)\otimes|-\rangle \\
&= \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(0)}|1\rangle + (-1)^{f(1)}|0\rangle + (-1)^{f(1)}|1\rangle}{2}\right)\otimes|-\rangle \\
&= \left(\frac{[(-1)^{f(0)} + (-1)^{f(1)}]|0\rangle + [(-1)^{f(0)} - (-1)^{f(1)}]|1\rangle}{2}\right)\otimes|-\rangle \\
&= (-1)^{f(0)}|f(0)\oplus f(1)\rangle\otimes|-\rangle
\end{aligned}
$$

STEP 4

$\tilde{\delta} \leftarrow$ measure the first register of $|\psi_3\rangle_{1,1}$



D. Some more substantial speed-ups with a quantum computer

1. Bernstein-Vazirani problem.

Like many of the examples discovered before Shor's factoring algorithm, this has a somewhat artificial character.

Its significance lies not in the intrinsic arithmetical interest of the problem, but in the fact that the problem can be solved dramatically and unambiguously faster on a quantum computer.

Let a be an unknown non-negative integer less than 2^n.

Let $f(x)$ take any other such integer x into the modulo 2 sum of the products of corresponding bits of a and x, which we denote by $a \cdot x$ (in recognition of the fact that it is a kind of bitwise modulo-2 inner product):

$$a \cdot x = a_0 x_0 \oplus a_1 x_1 \oplus a_2 x_2 \cdots . \tag{2.26}$$

Suppose we have a subroutine that evaluates $f(x) = a \cdot x$.

If we want to determine the value of the integer a, how many times do we have to call that subroutine?

Here and in all subsequent examples, we shall assume that any Qbits acted on by such subroutines, except for the Qbits of the input and output registers, are returned to their initial state at the end of the computation, as discussed earlier.

The m-th bit of a is $a \cdot 2^m$, since the binary expansion of 2^m has 1 in position m and 0 in all the other positions.

So with a classical computer we can learn the n bits of a by applying f to the n values $x = 2^m$, $0 \leq m < n$.

This, or any other classical method one can think of, requires n different invocations of the subroutine.

But with a quantum computer a *single* invocation is enough to determine a completely, regardless of how big n is!

I first describe the conventional way of seeing how this can be done, and then describe a much simpler way to understand the process.

The conventional way exploits a trick (implicitly exploited in our solution to Deutsch's problem) that is useful in dealing with functions like f that act on n-Qbits with output to a single Qbit.

If the 1-Qbit output register is initially prepared in the state $\mathbf{HX}|0\rangle = \mathbf{H}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ then, since $\mathbf{U}_f$ applied to the computational basis state $|x\rangle_n|y\rangle_1$ flips the value y of the output register if and only if $f(x) = 1$, we have

$$\mathbf{U}_f|x\rangle_n \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) = (-1)^{f(x)}|x\rangle_n \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big). \tag{2.27}$$

So by taking the state of the 1-Qbit output register to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ we convert a bit flip to an overall change of sign.

This becomes useful because of a second trick, which exploits a generalization of the action (2.5) of $\mathbf{H}^{\otimes n}$ on $|0\rangle_n$.

The action of $\mathbf{H}$ on a single Qbit can be compactly summarized as

$$\mathbf{H}|x\rangle_1 = \frac{1}{\sqrt{2}}\big(|0\rangle + (-1)^x|1\rangle\big) = \frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{xy}|y\rangle. \tag{2.28}$$

If we apply $\mathbf{H}^{\otimes n}$ to an n-Qbit computational basis state state $|x\rangle_n$ we can therefore express the result as

$$\mathbf{H}^{\otimes n}|x\rangle_n = \frac{1}{2^{n/2}}\sum_{y_{n-1}=0}^{1}\cdots\sum_{y_0=0}^{1}(-1)^{\sum_{j=0}^{n-1}x_j y_j}|y_{n-1}\rangle\cdots|y_0\rangle = \frac{1}{2^{n/2}}\sum_{y=0}^{2^n-1}(-1)^{x\cdot y}|y\rangle_n, \tag{2.29}$$

where the product $x \cdot y$ is the one defined in (2.26).

Because -1 is raised to the power $\Sigma x_j y_j$, all that matters about that sum is its value modulo 2.

So if we start with the n-Qbit input register in the standard initial state $\mathbf{H}^{\otimes n}|0\rangle$, put the 1-Qbit output register into the state $\mathbf{H}|1\rangle$, apply $\mathbf{U}_f$, and then again apply $\mathbf{H}^{\otimes n}$ to the input register, we get

$$(\mathbf{H}^{\otimes n} \otimes \mathbf{1})\mathbf{U}_f(\mathbf{H}^{\otimes n} \otimes \mathbf{H})|0\rangle_n|1\rangle_1 =$$

$$(\mathbf{H}^{\otimes n} \otimes \mathbf{1})\mathbf{U}_f\Big(\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle\Big)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =$$

$$\frac{1}{2^{n/2}}\Big(\mathbf{H}^{\otimes n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle\Big)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =$$

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1}(-1)^{f(x)+x\cdot y}|y\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \qquad (2.30)$$

We can do the sum over x first.

If the function $f(x)$ is $a \cdot x$ then this sum produces the factor

$$\sum_{x=0}^{2^n-1} (-1)^{(a\cdot x)}(-1)^{(y\cdot x)} = \prod_{j=1}^{n}\sum_{x_j=0}^{1}(-1)^{(a_j+y_j)x_j}. \qquad (2.31)$$

At least one term in the product vanishes unless every bit y_j of y is equal to the corresponding bit a_j of a — i.e. unless y = a.

Therefore the entire computational process (2.30) reduces to

$$\mathbf{H}^{\otimes(n+1)}\mathbf{U}_f\mathbf{H}^{\otimes(n+1)}|0\rangle_n|1\rangle_1 = |a\rangle_n|1\rangle_1, \qquad (2.32)$$

where I have applied a final **H** to the 1-Qbit output register to make the final expression look a little neater and more symmetric.

I've also restored subscripts to the state symbols for the n-Qbit input register and 1-Qbit output register.

So by putting the input and output registers into the appropriate initial states, after a single invocation of the subroutine followed by an application of $\mathbf{H}^{\otimes n}$ to the input register, the state of the input register becomes $|a\rangle$.

All n bits of the number a can now be determined by measuring the input register, even though we have called the subroutine only once!

end 11

Useful results:

$$\mathbf{U}_f\big(\,|x\rangle|y\rangle\big) = |x\rangle|y \oplus f(x)\rangle,$$

$$\mathbf{U}_f\big(\,|x\rangle_n|y\rangle_m\big) = |x\rangle_n|y \oplus f(x)\rangle_m,$$

$$\mathbf{U}_f\big(\,|x\rangle_n|0\rangle_m\big) = |x\rangle_n|f(x)\rangle_m$$

$$\mathbf{U}_f\big(\mathbf{H}^{\otimes n} \otimes \mathbf{1}_m\big)\big(\,|0\rangle_n|0\rangle_m\big) = \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} \mathbf{U}_f\big(\,|x\rangle_n|0\rangle_m\big)$$

$$= \frac{1}{2^{n/2}} \sum_{0 \le x < 2^n} |x\rangle_n|f(x)\rangle_m.$$

Now the derivation we were discussing…….. just do the algebra……..

A 1-Qbit output register is initially prepared in the state

$$\mathbf{HX}|0\rangle = \mathbf{H}|1\rangle = \tfrac{1}{\sqrt{2}}\,(|0\rangle - |1\rangle)$$

then, since $\mathbf{U}_f$ applied to the computational basis state $|x\rangle_n|y\rangle_1$ flips the value y of the output register if and only if $f(x) = 1$, we have

$$\mathbf{U}_f|x\rangle_n \tfrac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) = (-1)^{f(x)}|x\rangle_n \tfrac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big).$$

So by taking the state of the 1-Qbit output register to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ we convert a bit flip to an overall change of sign.

This becomes useful because of a second trick, which exploits a generalization of the action (2.5) of $\mathbf{H}^{\otimes n}$ on $|0\rangle_n$.

The action of **H** on a single Qbit can be compactly summarized as

$$\mathbf{H}|x\rangle_1 = \tfrac{1}{\sqrt{2}}\big(|0\rangle + (-1)^x|1\rangle\big) = \frac{1}{\sqrt{2}}\sum_{y=0}(-1)^{xy}|y\rangle.$$

If we apply $\mathbf{H}^{\otimes n}$ to an n-Qbit computational basis state state $|x\rangle_n$ we can therefore express the result as

$$\mathbf{H}^{\otimes n}|x\rangle_n = \frac{1}{2^{n/2}}\sum_{y_{n-1}=0}^{1}\cdots\sum_{y_0=0}^{1}(-1)^{\sum_{j=0}^{n-1}x_j y_j}|y_{n-1}\rangle\cdots|y_0\rangle$$

$$= \frac{1}{2^{n/2}}\sum_{y=0}^{2^n-1}(-1)^{x\cdot y}|y\rangle_n,$$

So if we start with the n-Qbit input register in the standard initial state $\mathbf{H}^{\otimes n}|0\rangle$,

put the 1-Qbit output register into the state $\mathbf{H}|1\rangle$,

apply $\mathbf{U}_f$,

and then again apply $\mathbf{H}^{\otimes n}$ to the input register, we get

$$(\mathbf{H}^{\otimes n} \otimes \mathbf{1})\mathbf{U}_f(\mathbf{H}^{\otimes n} \otimes \mathbf{H})|0\rangle_n|1\rangle_1 =$$

$$(\mathbf{H}^{\otimes n} \otimes \mathbf{1})\mathbf{U}_f\left(\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =$$

$$\frac{1}{2^{n/2}}\left(\mathbf{H}^{\otimes n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) =$$

***f*(x) not specified yet**

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1}(-1)^{f(x)+x\cdot y}|y\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Just algebra until now. Derivation until now independent of choice of *f*(x)

We can do the sum over x first.

If the function $f(x)$ is $a \cdot x$ then this sum produces the factor

make choice of $f(x) = a \cdot x$

$$\sum_{x=0}^{2^n-1} (-1)^{(a\cdot x)} (-1)^{(y\cdot x)} = \prod_{j=1}^{n} \sum_{x_j=0}^{1} (-1)^{(a_j+y_j)x_j}.$$

The amplitude for $|y\rangle$ is $1/2^n \, \Sigma_x (-1)^{a\cdot x + y\cdot x} = 1/2n \, \Sigma_x (-1)^{(a+y)\cdot x} = 1$ if y= a!

It's easy to see why it is zero for all other values of y.

Key point is ……. At least one term in the product vanishes unless every bit y_j of y is equal to the corresponding bit a_j of a — i.e. unless y = a.

Therefore the entire computational process (2.30) reduces(**just algebra**) to

$$\mathbf{H}^{\otimes(n+1)} \mathbf{U}_f \mathbf{H}^{\otimes(n+1)} |0\rangle_n |1\rangle_1 = |a\rangle_n |1\rangle_1,$$

So by putting the input and output registers into the appropriate initial states, after a single invocation of the subroutine followed by an application of $\mathbf{H}^{\otimes n}$ to the input register, the state of the input register becomes $|a\rangle$. **Crucial result.**

All n bits of the number a can now be determined by measuring the input register, even though we have called the subroutine only once! **must also make n measurements!**

I have probably not been clear enough so far on the endgame!

So, here is what I think is the best way to summarize the final arguments

So, classical computer has to run subroutine n times to get all bit of the n-bit a and then read a register after each run) - basically like reading an n-bit register.

The quantum computer has to run subroutine once and then read the n-bit input register.

If subroutine take t1 seconds to run and reading a bit takes t2 seconds where t1 >> t2, then

Quantum: t1+ n t2

Classical: n t1 + n t2

Classical time = **n** x Quantum time

Speedups are all determined by number of subroutine runs:

Deutsch 2 to 1

Bernstein-Vazirani n to 1

Continuing our lectures ……

There is a second, complementary way to look at the Bernstein-Vazirani problem that bypasses all of the preceding analysis, making it evident why (2.32) holds, by examining a few circuit diagrams.

The idea is to note, just as we did for the black box of Deutsch's problem in (2.13), that the actions of the black boxes that implement $\mathbf{U}_f$ for the different available choices of f are identical to the actions of some simple circuits.

When $f(x) = a \cdot x$, $\mathbf{U}_f$ flips the one-Qbit output register once, whenever a bit of x and the corresponding bit of a are both 1.

When the state of the input register is $|x\rangle_n$ this action can be performed by a collection of cNOT gates all targeted on the output register.

There is one cNOT for each non-zero bit of a, controlled by the Qbit representing the corresponding bit of x.

The combined effect of these cNOT gates on every computational basis state is precisely that of $\mathbf{U}_f$.

Therefore the effect of any other transformations preceding and/or following $\mathbf{U}_f$ can be understood by examining their effect on this equivalent collection of cNOT gates, *even though* $\mathbf{U}_f$ may actually be implemented in a completely different way.

The encoding of the sought-for value of a in the disposition of the equivalent cNOT gates, is illustrated in Figure 2.8.

The application (2.32) of **H** to every Qbit in the input and output registers both before and after the application of $\mathbf{U}_f$, pictured in Figure 2.9, converts every cNOT gate in the equivalent representation of $\mathbf{U}_f$ from **C**ij to $\mathbf{H}_i\mathbf{H}_j\,\mathbf{C}_{ij}\,\mathbf{H}_i\mathbf{H}j = \mathbf{C}ji$, as shown in Part 1 and pictured in Fig. 2.10.

After this reversal of target and control Qbits, the output register controls every one of the cNOT's, and since it contains a 1, every one of the NOT operators acts.

That action is to flip just those Qbits of the input register for which the corresponding bit of a is 1.

Since the input register starts in the state $|0\rangle_n$, this changes the state of each Qbit of the input register to to $|1\rangle$, if and only if it corresponds to a non-zero bit of a.

As a result, the state of the input register is converted from $|0\rangle_n$ to $|a\rangle_n$, just as (2.32) asserts.

Figure 2.8

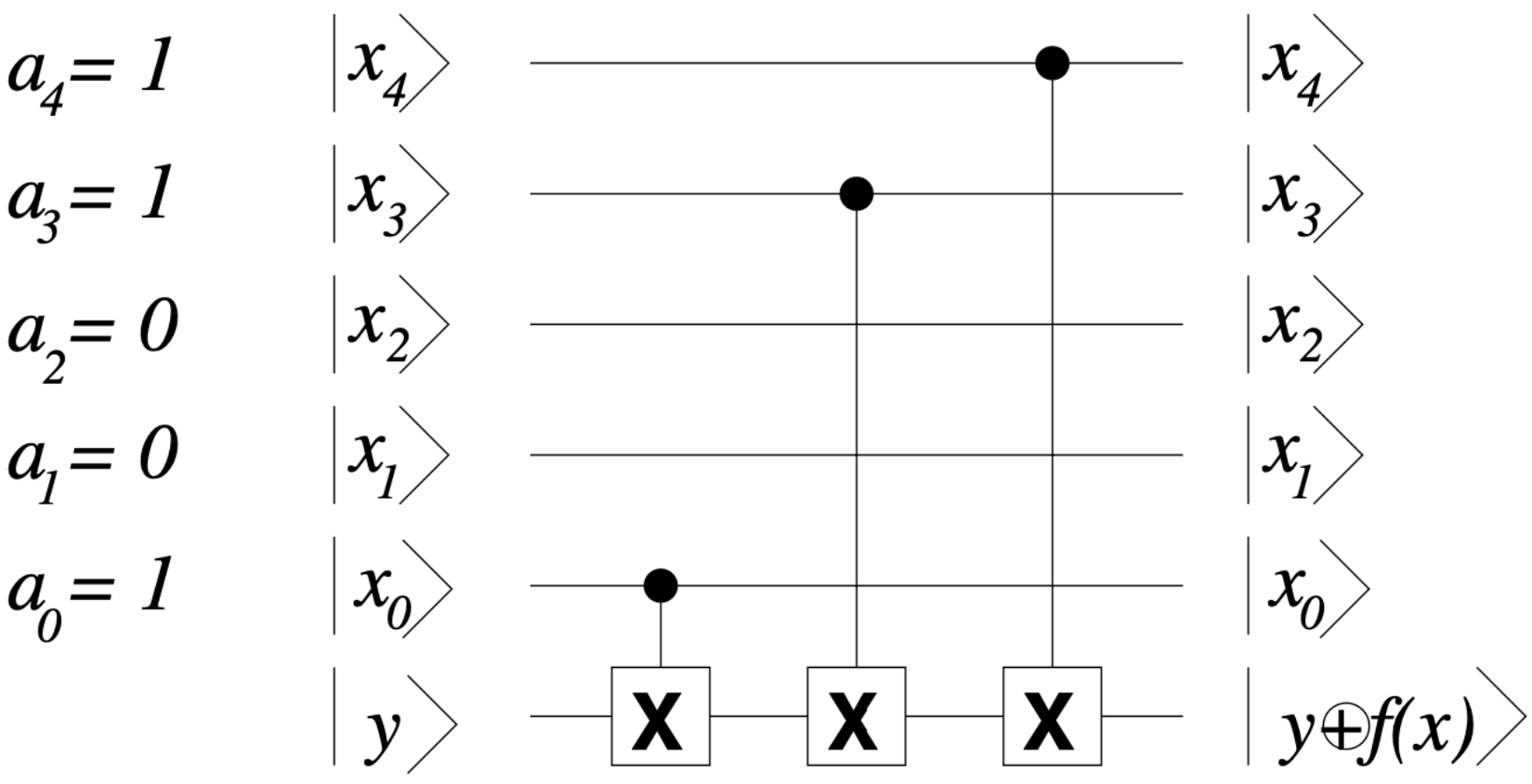


Figure 2.8. An illustration of a circuit that implements the unitary subroutine $\mathbf{U}_f$ taking n-Qbit input and 1-Qbit output registers, initially in the state $|x\rangle_n|y\rangle_1$, into $|x\rangle_n|y \oplus f(x)\rangle_1$, where $f(x) = a \cdot x = \sum_{j=0}^{n-1} a_j x_j \pmod 2$. The Bernstein-Vazarini problem is to determine all the bits of a with a single invocation of the subroutine. In the illustration n = 5 and a = 25 = 11001. For j = 0… n −1 each of the cNOT gates adds 1(mod 2) to the output register if and only if $a_j x_j = 1$. In addition to their normal labeling with the one-Qbit states they represent, the wires of the input register are labeled with the bits of a, to make it clear which (those associated with $a_j = 1$) act as control bits for a cNOT targeted on the output register.

Figure 2.9

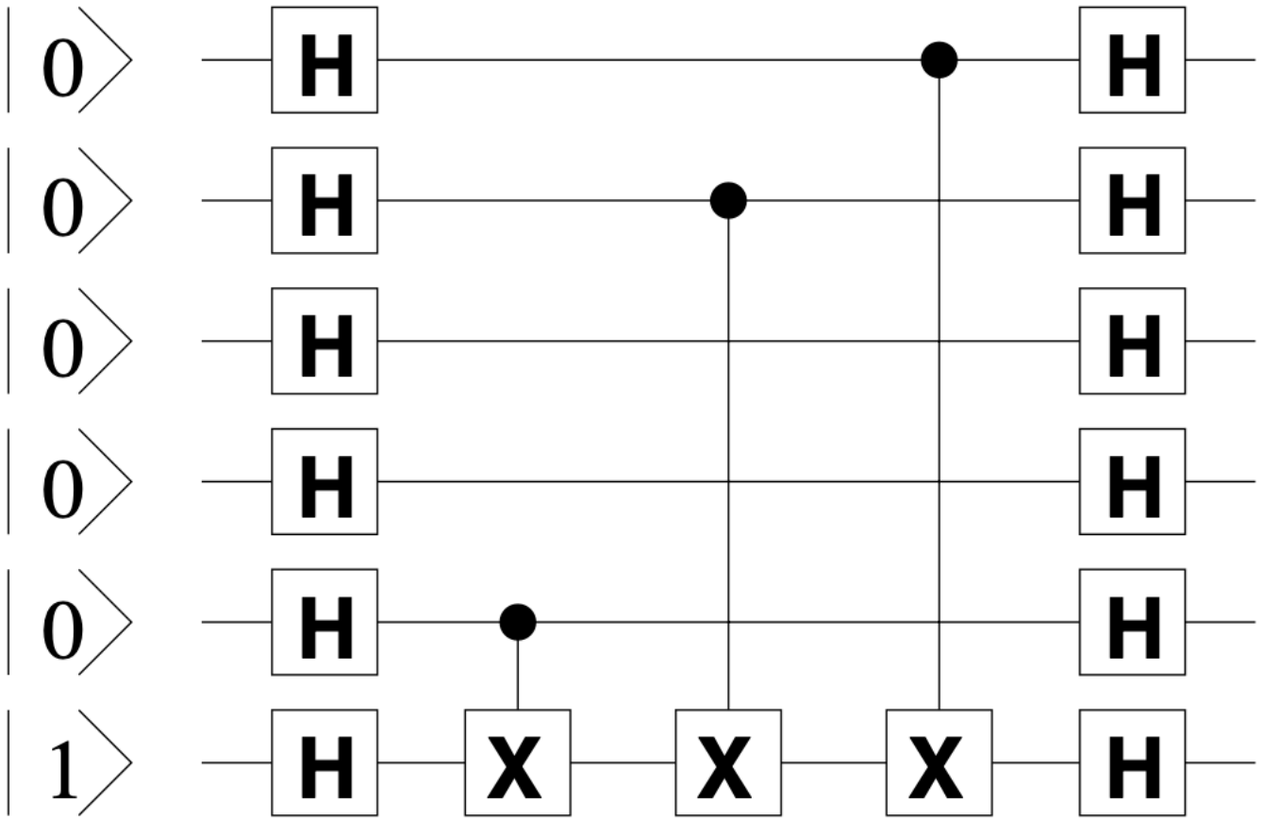


Figure 2.9. The solution to the Berstein-Vazirani problem is to start with the input register in the state $|0\rangle_n$ and the output register in the state $|1\rangle_1$ and apply Hadamard transforms to all n+1 registers before applying $\mathbf{U}_f$. Another n+1 Hadamards are applied after $\mathbf{U}_f$ acts. The cNOT gates reproduce the action of $\mathbf{U}_f$, as shown in Figure 2.10. The conventional analysis (as described above) deduces the final state by exploring the effect of the Hadamards on the initial state of the Qbits and on the state subsequently produced by the action of $\mathbf{U}_f$. A much easier way to understand what is going on is to examine the effect of the Hadamards on the collection of cNOT gates equivalent to $\mathbf{U}_f$. This is shown in Figure 2.10.

Figure 2.10

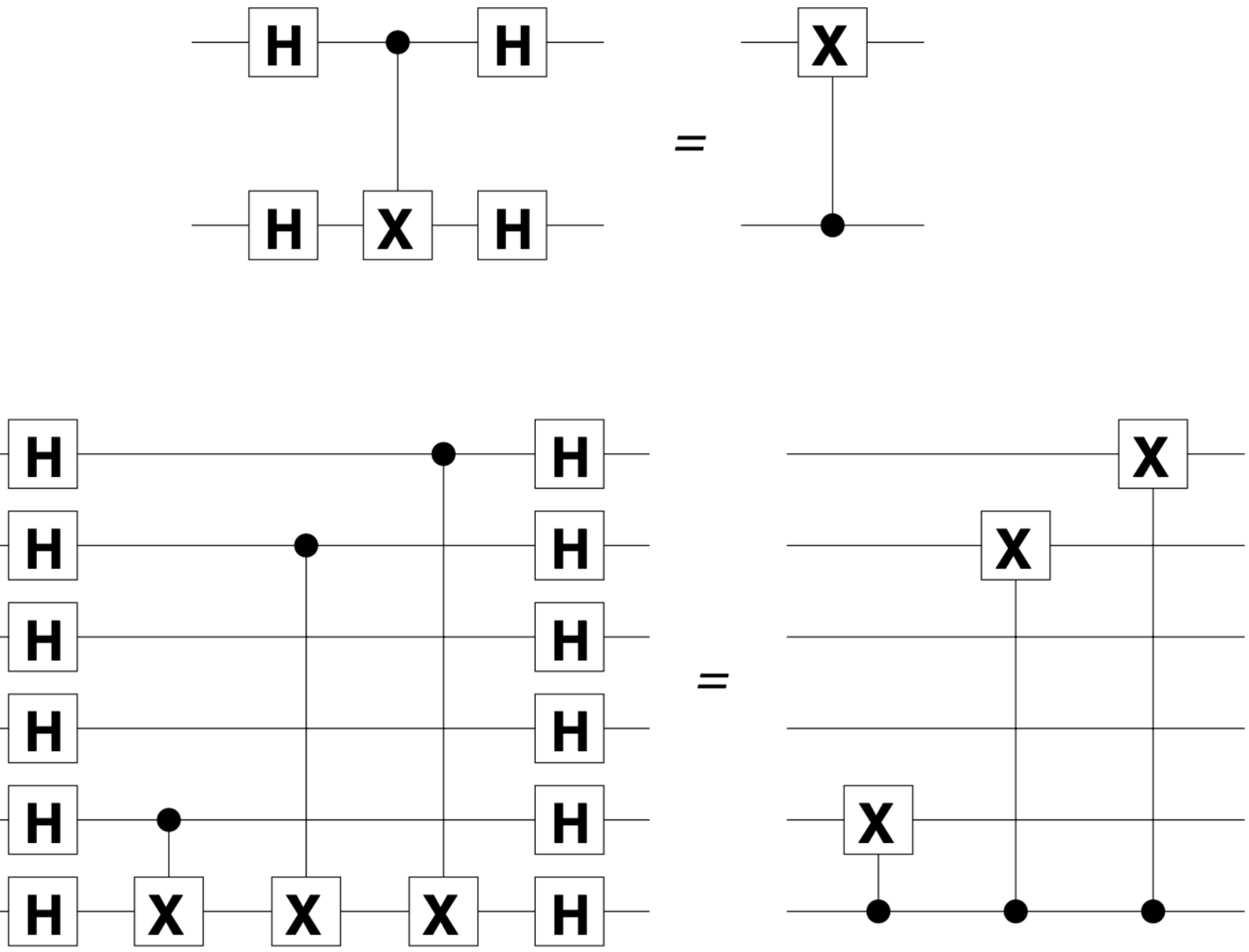


Figure 2.10. Sandwiching a cNOT gate between Hadamards that act on both control and target Qbits, has the effect of interchanging control and target, as shown at the top of the figure. (See also Part 1, Eq. (1.40) or Figure 2.2.) Consequently the action of all the Hadamards in Figure 2.9 on the cNOT gates between them is simply to interchange the control and target Qbits, as shown in the lower part of the figure. (One uses the fact that $\mathbf{H}^2 = \mathbf{1}$, so that (a) the **H** gates on wires that are not control bits have no action at all and (b) pairs of Hadamards can be introduced between every **X** on the lowest wire, converting **HXXXH** into (**HXH**) (**HXH**) (**HXH**).) Because the cNOT gates are now controlled by the output register, if the output register is in the state $|\,1\rangle$, as it is in Figure 2.9, then all the **X** act on their input-register targets. If the initial state of the input register is $|\,0\rangle_n$, as it is in Figure 2.11, the effect of each **X** is to convert to $|\,1\rangle$ the state of each Qbit associated with a bit of a that is 1. This converts the state of the input register to $|\,a\rangle_n$.

Figure 2.11

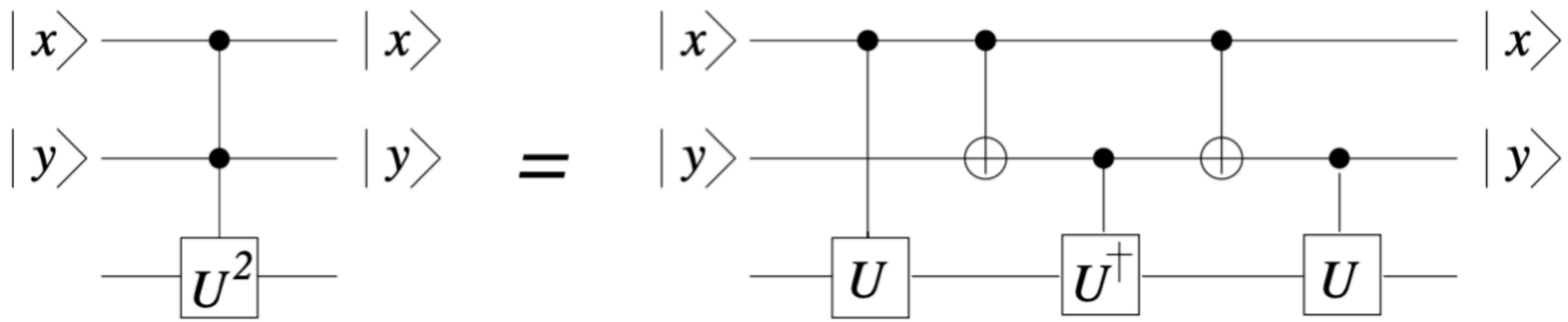


Figure 2.11. How to construct a three-Qbit controlled-controlled-U^2 gate from two-Qbit controlled-NOT, controlled-U, and controlled-$U^\dagger$ gates. The values x and y of the two control bits are indicated to the right and left of the wires that carry them. Note that both are unchanged by this sequence of operations, since the one labeled y is either flipped twice or not at all. To see that the circuit on the right acts as claimed, note first that its three central gates have the effect of applying $\mathbf{U}^\dagger$ to the bottom wire if and only if x and y are different. If x and y are different then just one of the two outermost controlled-U gates acts on the bottom wire, which combines with the $\mathbf{U}^\dagger$ to give 1. If x and y are both 0, neither of the controlled-U gates on the bottom wire, so since $\mathbf{U}^\dagger$ also does not act, the total action on the bottom wire is 1. But if both x and y are 1, then both controlled-U gates act on the bottom wire, and since $\mathbf{U}^\dagger$ does not act, the total action is $\mathbf{U}^2$.

NOTE: In keeping with the informal practice of talking about values of Qbits (or even bits) rather than the computational-basis state $|x\rangle$ (with x = 0 or 1) of the Qbit, one sometimes omits the state symbols $|\ \rangle$ from the input and output of a circuit diagram, specifying only the values of x, y,....

Since the input register starts in the state $|0\rangle_n$, this changes the state of each Qbit of the input register to $|1\rangle$, if and only if it corresponds to a non-zero bit of a.

As a result, the state of the input register is converted from $|0\rangle_n$ to $|a\rangle_n$, just as (2.32) asserts.

Note (as part of your education in quantum mechanics) how different these two explanations are.

The first applies $\mathbf{U}_f$ to the quantum superposition of all possible inputs, and then applies operations which lead to perfect destructive interference of all states in the superposition except for the one in which the input register is in the state $|a\rangle$.

The second suggests a specific mechanism for representing the subroutine that executes $\mathbf{U}_f$ and then shows that sandwiching such a mechanism between Hadamards automatically imprints a on the input register.

Quantum mechanics appears in the second method only through the possibility it allows for reversing the control and target Qbits of a cNOT operation solely through the use of one-Qbit (Hadamard) gates.

One also can reverse control and target bits of a cNOT classically, but this requires the use of 2-Qbit SWAP gates, rather than 1-Qbit Hadamards.

You should convince yourself that this solution to the Bernstein-Vazirani problem using Hadamards no longer works if you try to replace all the Hadamard gates by any arrangement of SWAP gates.

2. Simon's problem.

Simon's problem, like the Bernstein-Vazirani problem, has an n-bit non-zero number a built into the action of a subroutine $\mathbf{U}_f$, and the aim is to learn the value of a with as few invocations of the subroutine as possible.

In the Bernstein-Vazirani problem a classical computer must call the subroutine n times to determine the value of a, while a quantum computer need call the subroutine only once.

The number of calls grows linearly with n in the classical case, while being independent of n in the quantum case.

In Simon's problem the speed-up with a quantum computer is substantially more dramatic.

With a classical computer the number of times one must call the subroutine to solve Simon's problem grows exponentially with increasing n, but with a quantum computer it only grows linearly.

This spectacular speed-up involves a probabilistic element characteristic of many quantum computations.

The characterization of how the number of calls of the subroutine scales with the number of bits in a applies not to calculating it directly, but to learning a with high probability.

The subroutine $\mathbf{U}_f$ in Simon's problem evaluates a function f on n bits that is two to one — i.e. it is a function from n to n – 1 bits.

It is constructed so that $f(x) = f(y)$ if and only if the n-bit integers x and y are related by $x = y \oplus a$ or, equivalently and more symmetrically, $x \oplus y = a$, where $\oplus$ again denotes bitwise modulo-2 addition.

One can think of this as a period-finding problem.

One is told that f is periodic under bitwise modulo-2 addition,

$$f(x \oplus a) = f(x) \tag{2.33}$$

for all x, and the problem is to find the period a.

Simon's problem is thus a precursor of Shor's much more subtle and spectacularly more useful period-finding algorithm — the heart of his factoring procedure — where one finds the unknown period a of a function that is periodic under ordinary addition: $f(x+ a) = f(x)$.

To find the value of a in (2.33) with a classical computer all you can do is feed the subroutine different $x_1, x_2, x_3,...$, listing the resulting values of f until you stumble on an x_j that yields one of the previously computed values $f(x_i)$.

You then know that $a = x_j \oplus x_i$.

At any stage of the process prior to success, if you have picked m different values of x, then all you know is that $a \neq x_i \oplus x_j$ for all pairs of previously selected values of x.

You have therefore eliminated at most $\frac{1}{2}m(m-1)$ values of a.

You would have eliminated fewer values of a if you were careless enough to pick an x equal to $x_i \oplus x_j \oplus x_k$ for three values of x already selected.

Since there are $2^n - 1$ possibilities for a, your chances of success will not be appreciable while $\frac{1}{2}m(m-1)$ remains small compared with 2^n.

You are unlikely to succeed until m becomes of the order of $2^n/2$, so the number of times the subroutine has to be run to give an appreciable probability of determining a grows with the number of bits n as $2^n/2$ — i.e. exponentially.

If a has 100 bits a classical computer would have to run the subroutine about $2^{50} \approx 10^{15}$ times to have a significant chance of determining a.

At ten million calls per second it would take about **three years**.

In contrast, a quantum computer can determine a with high probability (say less than one chance in a million of failing) by running the subroutine not very much more than n times — e.g. with about 120 invocations of the subroutine if a has 100 bits.

This remarkable feat can be accomplished with the following strategy:

As usual, we apply the unitary transformation $\mathbf{U}_f$ only after the state of the input register has been transformed into the uniformly weighted superposition (2.5) of all possible inputs by the application of $\mathbf{H}^{\otimes n}$, so that the effect of $\mathbf{U}_f$ is to take the input and output registers into the entangled state

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle. \qquad (2.34)$$

If we now subject only the output register to a measurement, then the measurement gate is equally likely to indicate each of the 2^{n-1} different values of f.

Since each value is associated in (2.34) with precisely two values of x, the generalized Born rule tells us that the input register will be left in the state

$$\frac{1}{\sqrt{2}}\left(|x_0\rangle + |x_0 \oplus a\rangle\right) \qquad (2.35)$$

for the (random) value of x_0 for which $f(x_0)$ agrees with the (random) value of f given by the measurement.

At first glance this looks like great progress.

We have produced a superposition of just two computational basis states, associated with two n-bit integers that differ (in the sense of $\oplus$) by a.

If we knew those two integers their bitwise modulo 2 sum would be a.

But unfortunately, as we have noted earlier, when a register is in a given quantum state there is in general no way to learn what that state is.

To be sure if we could clone the state, then by measuring a mere 10 copies of it in the computational basis we could with a probability of about 0.998 learn both x_0 and $x_0 \oplus a$ and therefore a itself.

But unfortunately, as we have also noted earlier, one cannot clone an unknown quantum state.

Nor does it help to run the algorithm many times, since we are likely to get states of the form (2.35) for different (random) values of x_0.

By subjecting (2.35) to a direct measurement all we can learn is either x_0 — a random number — or $x_0 \oplus a$ — another random number.

The number a that we would like to know appears only in the relation between these two random numbers, only one of which we can find out.

Nevertheless, as in Deutsch's problem, if we renounce the possibility of learning either individual number (which alone is of no interest), we can get some useful partial information about their relationship (which is potentially of great interest).

The trick now is this:

$$\tfrac{1}{\sqrt{2}}\big(|x_0\rangle + |x_0 \oplus a\rangle\big)$$

With the input register in the state (2.35), before subjecting it to a measurement we apply the n-fold Hadmard transformation $\mathbf{H}^{\otimes n}$.

Eq. (2.29) then gives:

$$\mathbf{H}^{\otimes n}\tfrac{1}{\sqrt{2}}\big(|x_0\rangle + |x_0 \oplus a\rangle\big) = \frac{1}{2^{(n+1)/2}}\sum_{y=0}^{2^n-1}\big((-1)^{x_0\cdot y} + (-1)^{(x_0\oplus a)\cdot y}\big)|y\rangle. \qquad (2.36)$$

Since $(-1)^{(x_0\oplus a)\cdot y} = (-1)^{x_0\cdot y}(-1)^{a\cdot y}$, the coefficient of |y⟩ in (2.36) is 0 if a · y= 1 and $2(-1)^{x_0\cdot y}$ if a · y= 0.

Therefore (2.36) becomes

$$\frac{1}{2^{(n-1)/2}} \sum_{a \cdot y = 0} (-1)^{x_0 \cdot y} |y\rangle, \tag{2.37}$$

where the sum is now restricted to those y for which the modulo-2 bitwise inner product a·y is 0 rather than 1.

So measuring the input register now gives us (with equal probability) any of the values of y for which $a \cdot y = 0$ — i.e. for which

$$\sum_{i=0}^{n-1} y_i a_i = 0 \pmod 2, \tag{2.38}$$

where a_i and y_i are corresponding bits in the binary expansions of a and y.

Note the spectacular improvement on the classical case, where a single invocation of $\mathbf{U}_f$ tells us nothing whatever about a.

In the quantum case, we learn nothing only if we are unlucky enough to get y = 0 (which happens with the very small probability $1/2^{n-1}$).

Otherwise we learn a non-zero value of y, and therefore a non-trivial subset of the n bits of a whose modulo-2 sum vanishes.

One of those bits is thus entirely determined by the others in the subset, so we have cut the number of possible choices for a in half, from 2^n-1 (the -1 reflecting the fact that we are told that $a \neq 0$) to $2^{(n-1)/2} - 1$.

In one invocation of the subroutine we can, with very high probability, eliminate half of the candidates for a!

Furthermore, if we repeat the whole procedure once more, then with very high probability the new value of y that we learn will be neither 0 nor the same as the value we learned the first time.

We will therefore learn a new non-trivial relation among the bits of a which enables us to reduce the number of candidates by another factor of 2, removing 3/4 of the possibilities available for a with our two invocations of the subroutine (compared to the classical situation in which only a single value of a is removed).

If every time we repeat the procedure we have a good chance of reducing the number of choices for a by another factor of 2, then perhaps with n invocations of the subroutine we might have a significant chance of learning a precisely.

And indeed, one can show that if we repeat the procedure a fixed number of times (for example 20 times) more than n, then the odds are high (more than a million to one if we do n+20 runs) that we will learn a, no matter how large n may be.

Our description of the quantum computation and what we learn from it is now finished: each time we run the computation we learn a random y satisfying $a \cdot y = 0$.

To get a quantitative measure of how many runs we need to have a high probability of learning a, there remains some slightly subtle but purely mathematical analysis.

If we invoke $\mathbf{U}_f$ m times, we learn m independently selected random numbers y whose bits y_i satisfy (2.38).

It is then just a matter of thinking about familiar properties of the (somewhat unfamiliar) space of n-dimensional vectors with components restricted to the modulo 2 integers 0 and 1, to establish that a number of such vectors not very much larger than n suffices with high probability to determine a completely.

The point is that if we have $n-1$ relations (2.38) for $n-1$ linearly independent sets of y_i, then this gives us enough equations to determine a unique non-zero a.

"Linearly independent" in this context means linear independence over the integers modulo 2; i.e. no subsets of the y's should satisfy $y \oplus y' \oplus y'' \oplus \cdots = 0 \pmod 2$.

We have to invoke the subroutine enough times to give us a high probability of coming up with n– 1 linearly independent values of y.

But regardless of the size of n, for not terribly large x the probability becomes extremely close to 1 that a set of n+x random vectors from such an n–1 dimensional subspace contains a linearly independent subset.

This is obvious for ordinary vectors with continuous components, since the probability of a randomly selected vector in an n– 1 dimensional space lying in a subspace of lower dimensionality is zero — it is certain to have a non-zero component outside of the lower dimensional subspace.

The argument is trickier here because components are restricted to only two values: 1 or 0.

Introduce a basis in the full n– 1 dimensional subspace of all vectors y with $a \cdot y = 0$, so that a random vector in the subspace can be expressed as a linear combination of the basis vectors with coefficients that are randomly and independently 1 or 0.

Arrange the resulting (n + x) random vectors of 1's and 0's into a matrix of n + x rows and n – 1 columns.

Since the row rank (the number of linearly independent rows) of a matrix is the same as the column rank, even when arithmetic is confined to the integers modulo 2, the probability that some subset of n − 1 of the n + x (n− 1)-dimensional row vectors is linearly independent is the same as the probability that all n − 1 of the (n + x) dimensional column vectors are linearly independent.

But it is easy to find a lower bound for this last probability:

Pick a column vector at random.

The probability that it is non-zero is $1 - \frac{1}{2^{n+x}}$.

Take it as the first member of a basis in which we expand the remaining column vectors.

The probability that a second, randomly selected column vector is independent of the first is $1- \frac{1}{2^{n+x-1}}$, since it will be independent unless every one of its (random) components along the remaining n + x− 1 vectors is zero.

Continuing in this way, we conclude that the probability q of all n − 1 column vectors being linearly independent is

$$q = \left(1 - \frac{1}{2^{n+x}}\right)\left(1 - \frac{1}{2^{n+x-1}}\right)\cdots\left(1 - \frac{1}{2^{x+2}}\right). \tag{2.39}$$

If you're suspicious of this argument, reassure yourself by checking that it gives the right q when n = 3, a = 111, and x = 0, by explicitly enumerating which of the 64 different sets of three y's, all satisfying $a \cdot y = 0$, contain two linearly independent vectors.

Finally, to get a convenient lower bound on the size of q, note that if we have a set of non-negative numbers a, b, c,… whose sum is less than 1, the product (1−a)(1−b)(1−c)· · · exceeds 1− (a+b+c+· · ·).

Easily proved by induction on the number of numbers in the set.

The probability q is therefore greater than

$$1 - \frac{1}{2^{x+2}} - \frac{1}{2^{x+3}} - \cdots - \frac{1}{2^{x+n}}, \tag{2.40}$$

and this, in turn, is greater than

$$1 - \frac{1}{2^{x+1}}. \tag{2.41}$$

So if we want to be able to determine a with less than one chance in a million of failure, it is enough to run the subroutine n + 20 times.

This intrusion of some mildly arcane arithmetic arguments, to confirm that the output of the quantum computer provides the needed information in the advertised number of runs, is characteristic of many quantum-computational algorithms.

The action of the quantum computer itself is rather straightforward, but we must engage in more strenuous mathematical exertions to show that the outcome of the quantum computation does indeed enable us to accomplish the required task.

The remainder of this class now illustrates just how complicated things are in real work in the field. Just follow along as I repeat the details - Know that I could not do it on my own also except for the quantum mechanics part at the end!

E. The importance of cNOT gates or Constructing Toffoli Gates

As noted in Part I, constraints on what is physically feasible have pretty much limited people to considering unitary transformations that can be built up entirely out of 1-Qbit and 2-Qbit gates — unitary transformations that act only on a single Qbit at a time or on a single pair of Qbits.

It is generally assumed that arbitrary 1-Qbit unitary gates are relatively straightforward to construct, though even this can be difficult for many of the systems that have been proposed for Qbits.

Two-Qbit gates present an even tougher challenge to the quantum engineer, since they require one to manipulate the dynamical interaction between two physical Qbits.

The controlled-NOT (cNOT) gate is the great workhorse of 2-Qbit gates because any arithmetical operation on bits can be carried out on computational-basis states with operations built up out of 1-Qbit gates acting in suitable combination with 2-Qbit cNOT gates.

No other 2-Qbit gates are required.

This happy state of affairs follows from three facts.

(a) Any arithmetical operation can be built up on a reversible classical computer out of *three*-Cbit controlled-controlled-not gates (ccNOT gates, called *Toffoli gates*).

(b) The quantum ccNOT gate — the linear extension of ccNOT from computational basis states to arbitrary three-Qbit states — can be built up out of cNOT gates $\mathbf{C}$ and appropriate controlled-U gates $\mathbf{cU}$, defined in analogy to the cNOT gate: acting on computational basis states $\mathbf{cU}_{ij}$ does nothing to the target Qbit j if the state of the control Qbit i is $|0\rangle$, but it applies the unitary transformation $\mathbf{U}$ to the target Qbit if the state of the control Qbit is $|1\rangle$.

(c) Controlled-U gates can be built out of controlled-NOT gates and appropriate 1-Qbit unitary gates.

We take up these three points in turn:

(a) The 3-bit Toffoli gate **T** is the great workhorse of reversible classical computation.

It flips the third (target) bit if and only if both of the first two (control bits) are 1:

$$x, y, z \rightarrow x, y, z \oplus xy. \tag{2.42}$$

Its extension to a unitary transformation on Qbits is correspondingly defined in the computational basis by

$$\mathbf{T}|x\rangle|y\rangle|z\rangle = |x\rangle|y\rangle|z \oplus xy\rangle. \tag{2.43}$$

Since **T** is its own inverse, it is clearly reversible, so its action on the computational basis is simply to permute those states, and therefore its linear extension to arbitrary Qbit states is unitary (by the argument given in Part 1.D).

The Toffoli gate enables one to calculate the logical AND of two bits (i.e., their product) since **T**|x⟩|y⟩|0⟩= |x⟩|y⟩|xy⟩.

Since all Boolean operations can be built up out of AND and NOT, and since all arithmetic can be built up out of Boolean operations, using Toffoli gates one can build up *all* of classical computation through reversible operations.

One can even produce NOT with a Toffoli gate: $\mathbf{T}\,|1\rangle\,|1\rangle = |1\rangle\,|1\rangle\,|\bar{x}\rangle$, but this would be a ridiculous way to implement NOT on a quantum computer, since 3-Qbit gates are much harder to implement than 1-Qbit gates.)

(b) Although there is no way to construct the Toffoli gate out of 1-Cbit and 2-Cbit classical gates, the trick can be done with the corresponding quantum gates.

The first thing to note is that there is a quantum gate whose square is the NOT operator $\mathbf{X}$.

It is called "the square root of NOT" and we shall write it as $\sqrt{\mathbf{X}}$.

Its matrix is just

$$\sqrt{\mathbf{X}} = \frac{1}{1+i}\begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}. \tag{2.44}$$

One easily verifies that

$$(\sqrt{\mathbf{X}})^2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \mathbf{X}, \tag{2.45}$$

and that it is indeed unitary,

$$\sqrt{\mathbf{X}}^{\dagger}\sqrt{\mathbf{X}} = 1. \tag{2.46}$$

To build the action on computational basis states of the 3-Qbit Toffoli gate out of 1-Qbit and 2-Qbit gates we require a *controlled* square root of NOT gate.

More generally, I describe in (c) below how to construct a controlled-U gate for any 1-Qbit unitary U.

Given such a controlled-U gate $\mathbf{cU}_{ij}$ with Qbit i the control and j the target, a doubly controlled $\mathbf{U}^2$ gate, controlled by Qbits 2 and 1 and targeting Qbit 0, can be constructed out of three controlled-U gates and two cNOT gates:

$$\mathbf{cU}_{10}\mathbf{C}_{21}\mathbf{cU}_{10}^{\dagger}\mathbf{C}_{21}\mathbf{cU}_{20}. \tag{2.47}$$

The corresponding circuit diagram is shown in Figure 2.11.

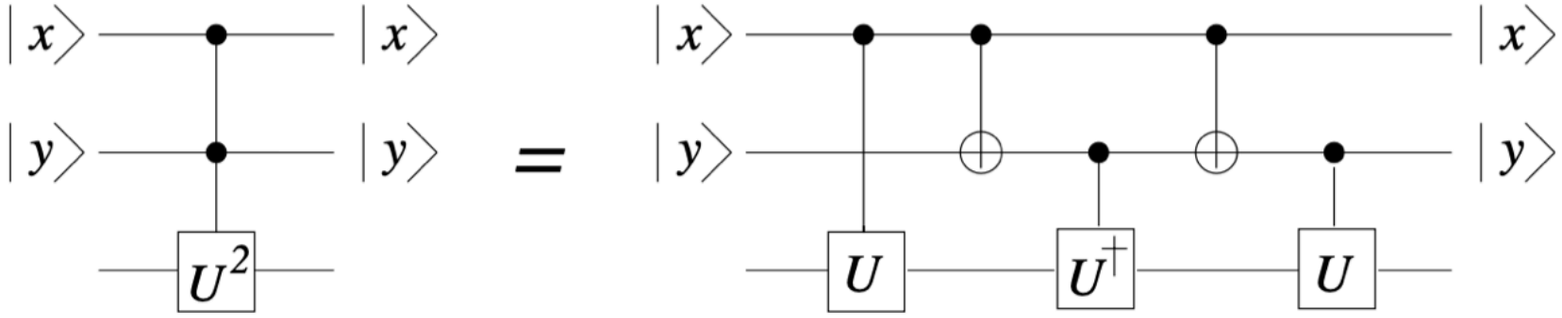


Figure 2.11

Figure 2.11. How to construct a three-Qbit controlled-controlled-U^2 gate from two-Qbit controlled-NOT, controlled-U, and controlled-$U^\dagger$ gates. The values x and y of the two control bits are indicated to the right and left of the wires that carry them. Note that both are unchanged by this sequence of operations, since the one labeled y is either flipped twice or not at all. To see that the circuit on the right acts as claimed, note first that its three central gates have the effect of applying $\mathbf{U}^\dagger$ to the bottom wire if and only if x and y are different. If x and y are different then just one of the two outermost controlled-U gates acts on the bottom wire, which combines with the $\mathbf{U}^\dagger$ to give 1. If x and y are both 0, neither of the controlled-U gates on the bottom wire, so since $\mathbf{U}^\dagger$ also does not act, the total action on the bottom wire is 1. But if both x and y are 1, then both controlled-U gates act on the bottom wire, and since $\mathbf{U}^\dagger$ does not act, the total action is $\mathbf{U}^2$.

One can verify that, acting on the 3-Qbit computational basis states, (2.47) acts as **1** unless Qbits 2 and 1 are both in the state $|1\rangle$, in which case it acts on Qbit 0 as $\mathbf{U}^2$.

An efficient way to see this is first to note that the part in the middle,

$$\mathbf{C}_{21}\mathbf{c}\mathbf{U}_{10}^{\dagger}\mathbf{C}_{21} \tag{2.48}$$

leaves the computational-basis states of Qbits 2 and 1 unchanged, while acting on Qbit 0 as **1** if those states are both $|0\rangle$ or both $|1\rangle$, and acting on Qbit 0 as $\mathbf{U}^\dagger$ if one of the states is $|0\rangle$ and the other, $|1\rangle$.

But the $\mathbf{cU}_{10}$ and $\mathbf{cU}_{20}$, between which the gates in (2.48) are sandwiched in (2.47), both act on the the target Qbit 0 as 1 when the states of both control Qbits are $|0\rangle$; when the states of the control Qbits are $|0\rangle$ and $|1\rangle$, they provide a single **U** on the target Qbit, which combines with the $\mathbf{U}^\dagger$ to give **1**: and they both act on the target as **U** when the states of both control Qbits are $|1\rangle$.

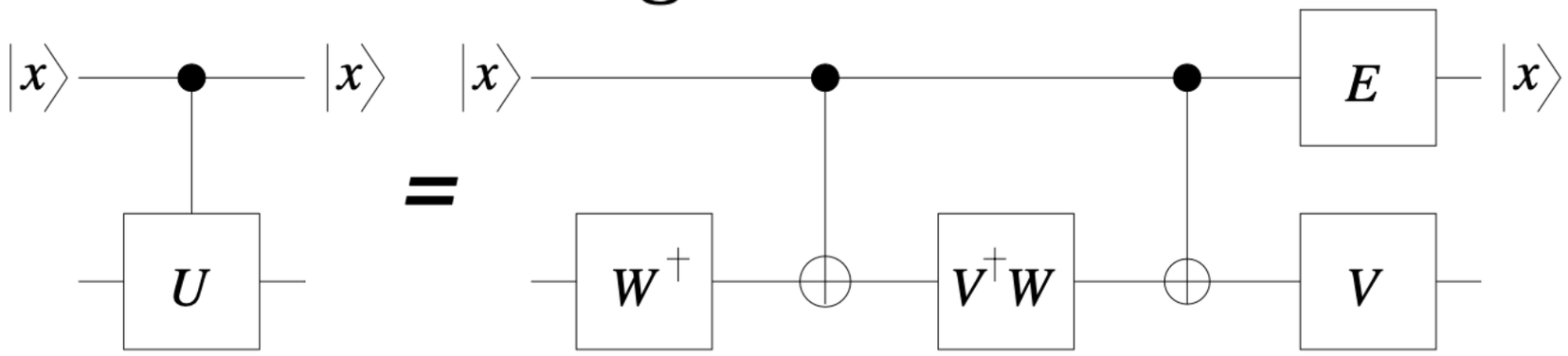
So the circuit does indeed give a doubly controlled $\mathbf{U}^2$.

(c) Finally, we must show that a controlled-U gate can be built out of controlled-NOT gates and 1-Qbit unitary gates.

Let **V** and **W** be arbitrary 1-Qbit unitary transformations and consider the product (see also Figure 2.12, noting there some of the irritating consequences of the convention that operators on the right act first in equations, but operators on the left act first in circuit diagrams)

$$\mathbf{V}_0\mathbf{C}_{10}\mathbf{V}_0^\dagger\mathbf{W}_0\mathbf{C}_{10}\mathbf{W}_0^\dagger. \tag{2.49}$$

Figure 2.12



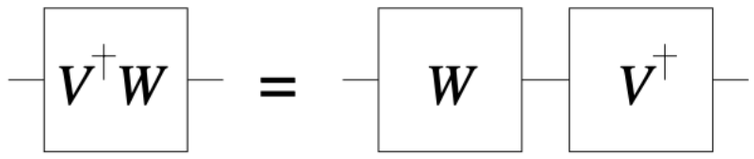
note: 

Figure 2.12. How to construct a controlled-U gate $\mathbf{C}_U$ from the unitary one-Qbit gates and two controlled-NOT gates. If the control-Qbit is in the state $|0\rangle$, the operations on the target wire combine to give the identity **1**. But if the control Qbit is in the state $|1\rangle$ then the operations combine to give $\mathbf{U} = (\mathbf{V}\sigma_x\mathbf{V}^\dagger)(\mathbf{W}\sigma_x\mathbf{W}^\dagger)$, where $\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \mathbf{X}$ is the flip operator. As noted in Part 2.D above, a general two dimensional unitary transformation can always be put in this form for appropriate **V** and **W**, to within an overall constant phase factor. The **E** on the control wire is the unitary transformation $\mathbf{E} = e^{in\alpha} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$, which supplies such a phase factor when the control bit is 1. Note the irritating left-right interchange between the order of symbols in the formula for **U** and the ordering of gates in the diagram. Note also the example, in the lower half of the figure, of the even more irritating general feature of circuit diagrams that follows from this reversal, illustrated in Figure 1.3 of Part 1.

If the state of the control Qbit 1 is $|0\rangle$ then $\mathbf{C}_{10}$ acts as $\mathbf{1}$ on the target Qbit 0 and the action of the unitary transformations on the target reduces to $(\mathbf{V}\mathbf{V}^\dagger)(\mathbf{W}\mathbf{W}^\dagger) = \mathbf{1}$.

But if the state of the control Qbit is $|1\rangle$, then $\mathbf{C}_{10}$ acts as $\mathbf{X}$ on the target Qbit and the action of the unitary transformations on the target becomes $(\mathbf{V}\mathbf{X}\mathbf{V}^\dagger)(\mathbf{W}\mathbf{X}\mathbf{W}^\dagger)$.

To see that an arbitrary 1-Qbit unitary transformation can be produced by appropriate choices for $\mathbf{V}$ and $\mathbf{W}$ it is best to think of the operator $\mathbf{X}$ in its equivalent form as the Pauli operator $\mathbf{X} = \sigma_x = \mathbf{x} \cdot \sigma$,
so that the action on the target Qbit when the state of the control Qbit is $|1\rangle$ becomes

$$\left(\mathbf{V}(\mathbf{x}\cdot\sigma)\mathbf{V}^\dagger\right)\left(\mathbf{W}(\mathbf{x}\cdot\sigma)\mathbf{W}^\dagger\right). \qquad (2.50)$$

As explained earlier in the extra mathematics at the end of Part I, it is possible to pick unitary $\mathbf{U}$ so that $\mathbf{U}(\mathbf{x}\cdot\sigma)\mathbf{U}^\dagger = \mathbf{c}\cdot\sigma$ for any unit vector $\mathbf{c}$ by taking $\mathbf{U}$ to be the unitary transformation associated with any rotation that takes $\mathbf{x}$ into $\mathbf{c}$.

It is therefore possible to choose $\mathbf{V}$ and $\mathbf{W}$ so that

$$\left(\mathbf{V}(\mathbf{x}\cdot\sigma)\mathbf{V}^\dagger\right)\left(\mathbf{W}(\mathbf{x}\cdot\sigma)\mathbf{W}^\dagger\right) = (\mathbf{a}\cdot\sigma)(\mathbf{b}\cdot\sigma) = \mathbf{a}\cdot\mathbf{b} + i\mathbf{a}\otimes\mathbf{b}\cdot\sigma, \qquad (2.51)$$

for any desired unit vectors $\mathbf{a}$ and $\mathbf{b}$.

But an arbitrary 1-Qbit unitary transformation has, to within a multiplicative numerical phase factor, the form

$$\mathbf{u}(\mathbf{n},\theta) = e^{i\frac{1}{2}\theta(\mathbf{n}\cdot\sigma)} = \cos\tfrac{1}{2}\theta + i(\mathbf{n}\cdot\sigma)\sin\tfrac{1}{2}\theta. \qquad (2.52)$$

Hence by taking **a** and **b** to be in the plane perpendicular to **n** and taking the angle between them to be $\frac{1}{2}\theta$, we can reproduce any unitary transformation of the form (2.52).

If the unitary transformation has an additional phase factor $e^{i\alpha}$, then we need to apply an additional controlled-change-of-phase operation, that acts in the computational basis to take $|x\rangle|y\rangle$ into $|x\rangle(e^{i\alpha x}|y\rangle)$. But this state is identical to $e^{i\alpha x}|x\rangle|y\rangle$, which can be produced by applying the 1-Qbit unitary transformation $e^{i\alpha\mathbf{n}}$ directly to the control Qbit.

Eight cNOT gates are required to construct a Toffoli gate in this way: the construction of the doubly controlled $\mathbf{U}^2$ gate in (2.47) (illustrated in Figure 2.11) requires two cNOT gates and three controlled-unitary gates, and each controlled-unitary gate, as described in point (c) above (illustrated in Figure 2.12) requires two more cNOT gates.

There is a more direct construction that uses only six cNOT gates, illustrated in Figure 2.13.

Figure 2.13

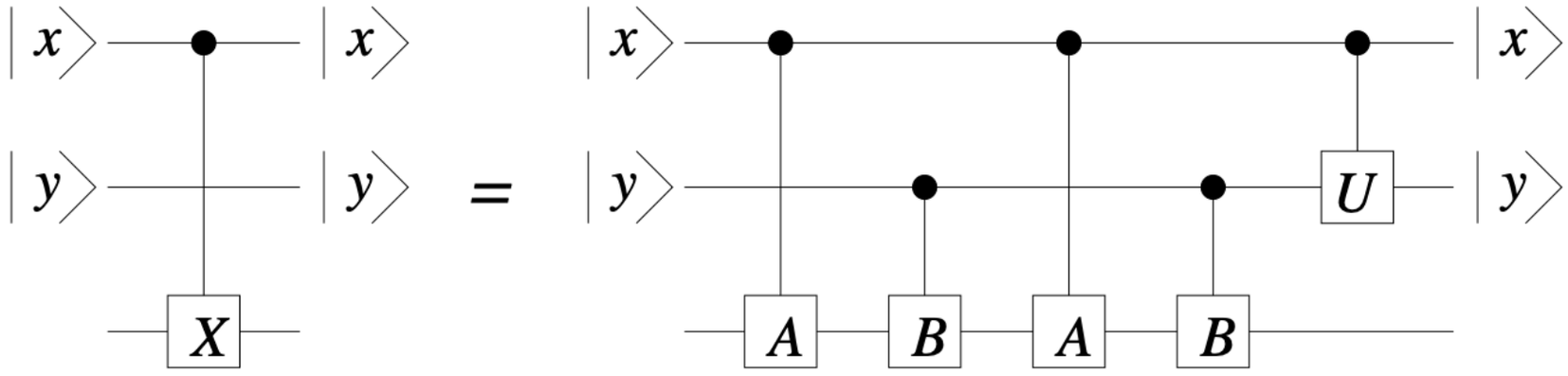


Figure 2.13. How to make a doubly-controlled NOT (Toffoli) gate using only 6 cNOT gates. The unitary operators **A** and **B** are given by $\mathbf{A} = \mathbf{a}\cdot\sigma$ and $\mathbf{B} = \mathbf{b}\cdot\sigma$ for appropriately chosen real unit vectors **a** and **b**. Since $\mathbf{n}\cdot\sigma$ can be expressed as $\mathbf{V}^{\dagger}(\mathbf{x}\cdot\sigma)\mathbf{V}$ for appropriate unitary **V**, and $\mathbf{X} = \mathbf{x}\cdot\sigma$, each controlled-A and controlled-B gate can be constructed with a single controlled-NOT gate and 1-Qbit unitaries. Since $\mathbf{A}^2 = \mathbf{B}^2 = \mathbf{1}$, it follows that the target Qbit (bottom wire) is not acted upon unless $x = y = 1$, in which case the target Qbit is acted on by $(\mathbf{BA})^2$. The product **BA** is the unitary $(\mathbf{b}\cdot\sigma)(\mathbf{a}\cdot\sigma) = \mathbf{b}\cdot\mathbf{a} + i(\mathbf{b}\times\mathbf{a})\cdot\sigma$. Pick the unit vectors **b** and **a** so the angle between them is $\pi/4$ and they lie in the plane perpendicular to **x** with their vector product directed along **x**, so that $(\mathbf{b}\cdot\sigma)(\mathbf{a}\cdot\sigma) = \cos(\pi/4) + i\sin(\pi/4)\mathbf{x}\cdot\sigma$. Then $(\mathbf{BA})^2 = \cos(\pi/2) + i\sin(\pi/2)\mathbf{x}\cdot\sigma = i\mathbf{x}\cdot\sigma = i\mathbf{X}$. So the alternating controlled-A and -B gates produce a doubly-controlled-NOT gate except for an extra factor of i accompanying the NOT. The controlled-U gate on the right corrects for this. Here **U** is the 1-Qbit unitary $e^{-i(\pi/2)\mathbf{n}}$, so controlled-U acts as the identity unless $x = y = 1$ in which case it multiplies the state by $e^{-i\pi/2} = -i$. This cancels the unwanted factor of i produced by the controlled-A and -B gates when $x = y = 1$. Since any controlled-U gate can be constructed with two cNOT gates and 1-Qbit unitaries, using the construction in Figure 2.12, this adds two more cNOT gates to the construction for a total of six. Alternatively, one can view this as constructing a Toffoli gate from four cNOT gates and a single controlled-phase gate of precisely the kind that plays a central role in the quantum Fourier transform described in Part 3.

As the figure makes clear, it is quite straightforward to construct a doubly-controlled i**X** gate, using only four cNOT gates.

To get rid of the extra factor of i, however, requires an additional controlled-phase gate, which can be constructed using two more cNOT gates.

If quantum computation ever becomes a working technology, it might well be easier to construct controlled-phase gates as fundamental gates in their own right — as basic a piece of 2-Qbit hardware as a cNOT gate.

Two other comments about cNOT gates:

(1) As demonstrated in Figure 2.7, n cNOT gates can act jointly to take the 2n Qbit state $|x\rangle_n|0\rangle_n$ into $|x\rangle_n|x\rangle_n$.

But although this clones computational basis states, it does not violate the no-cloning theorem, since it fails to clone nontrivial superpositions of computational basis states.

(2) There is a way in which cNOT gates resemble measurements: the transformation of $|x\rangle_n|0\rangle_n$ into $|x\rangle_n|x\rangle_n$ is reminiscent of the way in which applying a measurement gate to n Qbits in the state $|x\rangle_n$ results in the number x_n appearing on the display of the measurement gate, while the original n Qbits remain in the state $|x\rangle_n$.

But the analogy between the target Qbits and the display of the measurement gate breaks down when we consider superpositions.

If we want to measure n Qbits in the state $a|x\rangle_n + b|y\rangle_n$ a measurement gate will display x with probability $|a|^2$ and y with probability $|b|^2$, leaving the n Qbits in the state $|x\rangle_n$ in the first case and $|y\rangle_n$ in the second.

In contrast, the n controlled-NOT gates take $(a|x\rangle_n + b|y\rangle_n)|0\rangle_n$ into

$$a|x\rangle_n|x\rangle_n + b|y\rangle_n|y\rangle_n. \tag{2.53}$$

The n control Qbits and the n target Qbits are now in an entangled state.

Neither set of Qbits has any state of its own.

In particular the control Qbits are neither in the state $|x\rangle_n$ nor $|y\rangle_n$ as they would be after a measurement.

Nor are the target Qbits associated with either of these two states.

Indeed, in contrast to a measurement gate, the action of the n cNOT gates can be reversed by simply applying them a second time to the 2n Qbits.

If, however, one subjects the target Qbits to a measurement, then, according to the generalized Born rule, the joint state (2.53) of control and target Qbits, reduces to $|x\rangle_n|x\rangle_n$ with probability $|a|^2$ or to $|y\rangle_n|y\rangle_n$ with probability $|b|^2$, and we are in a situation in which the control and target Qbits behaving *exactly* like measured system and measuring device.

What the cNOT gates can do is to shift the measurement process from applying the measurement gate to the control Qbits, to applying it to the target Qbits.

Technical Extras to Part 2

This more technical part is addressed to physicists, curious about how one might, at least in principle, construct a cNOT gate, exploiting physically plausible interactions between two Qbits.

Those with no background in quantum physics are invited to inspect it, but will find many parts quite obscure.

It is of relevance only to those interested in quantum computational hardware, and plays no role in subsequent developments.

As noted in Part 1 (and as easily confirmed directly) the controlled-NOT gate $\mathbf{cX}_{12}$ with control Qbit 1 and target Qbit 2 can be written as

$$\mathbf{cX}_{12} = \tfrac{1}{2}\left(\mathbf{1} + \mathbf{Z}_1 + \mathbf{X}_2 - \mathbf{Z}_1\mathbf{X}_2\right). \tag{2.54}$$

Since $\mathbf{Z} = \mathbf{HXH}$ and $\mathbf{1} = \mathbf{H}^2$, this can also be written as

$$\mathbf{cX}_{12} = \mathbf{H}_2\mathbf{cZ}_{12}\mathbf{H}_2, \tag{2.55}$$

where the more symmetric controlled-Z operation is given by

$$\mathbf{cZ} = \tfrac{1}{2}\left(\mathbf{1} + \mathbf{Z}_1 + \mathbf{Z}_2 - \mathbf{Z}_1\mathbf{Z}_2\right). \tag{2.56}$$

Because of its symmetry we may write $\mathbf{cZ}$ without the subscripts distinguishing control and target Qbits.

So to within one-Qbit Hadamard transformations, the problem of constructing a $\mathbf{cX}$ gate is the same as that of constructing a $\mathbf{cZ}$ gate.

Since $(\mathbf{cZ})^2 = \mathbf{1}$, $\mathbf{cZ}$ satisfies the identity (both sides have identical expansions as power series in θ)

$$\exp(i\mathbf{cZ}\theta) = \cos\theta + i\mathbf{cZ}\sin\theta. \tag{2.57}$$

We can therefore rewrite (2.56) as the less straightforward operator identity:

$$\mathbf{cZ} = -i\exp(i\tfrac{\pi}{2}\mathbf{cZ}) = -i\exp\big(i(\tfrac{\pi}{4})(\mathbf{1}+\mathbf{Z}_1+\mathbf{Z}_2-\mathbf{Z}_1\mathbf{Z}_2)\big)$$

$$= e^{-i(\pi/4)}\exp\big(i(\tfrac{\pi}{4})(\mathbf{Z}_1+\mathbf{Z}_2-\mathbf{Z}_1\mathbf{Z}_2)\big). \qquad (2.58)$$

The point of writing **cZ** in this clumsy way is that the unitary transformations one can construct physically are those of the form

$$\mathbf{U} = \exp(i\mathcal{H}t), \qquad (2.59)$$

where $\hbar\mathcal{H}$ is the Hamiltonian that describes the external fields acting on the Qbits, and the interactions between Qbits.

So to within an overall constant (and therefore irrelevant) phase factor we can realize a **cZ** gate by letting the two Qbits interact through a Hamiltonian proportional to $\mathbf{Z}_1 + \mathbf{Z}_2 - \mathbf{Z}_1\mathbf{Z}_2$ for a precisely specified interval of time.

If each Qbit is a spin-1/2, then (since $\mathbf{Z} = \sigma_z$) this Hamiltonian describes two such spins with a highly anisotropic interaction that couples only their z-components (*Ising interaction*) subject to a uniform magnetic field with a magnitude appropriately proportional to the strength of their coupling.

This is perhaps the simplest example of how to make a cNOT gate.

Ising interactions, however, are rather hard to arrange.

A much more natural interaction between two spins is the *exchange interaction*

$$\sigma^{(1)} \cdot \sigma^{(2)} = \sigma_x^{(1)}\sigma_x^{(2)} + \sigma_y^{(1)}\sigma_y^{(2)} + \sigma_z^{(1)}\sigma_z^{(2)} = \mathbf{X}_1\mathbf{X}_2 + \mathbf{X}_1\mathbf{Z}_1\mathbf{Z}_2\mathbf{X}_2 + \mathbf{Z}_1\mathbf{Z}_2, \qquad (2.60)$$

which is invariant under spatial rotations. (See mathematics extras for Part 1.)

One can also build a **cZ** gate out of two spins interacting through (2.60), provided one applies magnetic fields to each spin that are along the same direction but have different magnitudes and signs.

What we must show is that to within an overall constant phase factor it is possible to express **cZ** in the form

$$\mathbf{cZ} = \exp(i\mathcal{H}t), \qquad (2.61)$$

with a Hamiltonian $\mathcal{H}$ of the form

$$\mathcal{H} = J\,\sigma^{(1)} \cdot \sigma^{(2)} + H_1\,\sigma_z^{(1)} + H_2\,\sigma_z^{(2)}, \qquad (2.62)$$

for appropriate choices of J (known as the exchange coupling), of H_1 and H_2 (proportional to the magnetic fields acting on the two spins — hereafter we ignore the proportionality constant and refer to them simply as the "magnetic fields"), and of the time t during which the spins interact with each other and with the magnetic fields.

To see that the parameters in (2.62) can indeed be chosen so that $\mathcal{H}$ gives rise to **cZ** through (2.61), recall first that the operator $\frac{1}{2}(1 + \sigma^{(1)} \cdot \sigma^{(2)})$ acts as the swap operator on any two-Qbit computational-basis state:

$$\tfrac{1}{2}(1 + \sigma^{(1)} \cdot \sigma^{(2)})|xy\rangle = |yx\rangle. \tag{2.63}$$

It follows from (2.63) that the three states (called *triplet states*)

$$|11\rangle, \quad |00\rangle, \quad \tfrac{1}{\sqrt{2}}\big(|01\rangle + |10\rangle\big) \tag{2.64}$$

are eigenstates of $\sigma^{(1)} \cdot \sigma^{(2)}$ with eigenvalue 1, while the state

$$\tfrac{1}{\sqrt{2}}\big(|01\rangle - |10\rangle\big) \tag{2.65}$$

(called the the *singlet state*) is an eigenstate of $\sigma^{(1)} \cdot \sigma^{(2)}$ with eigenvalue -3.

NOTE: If $|0\rangle$ is the state $|\uparrow\rangle$ of spin-up along **z**, and $|1\rangle$ is $|\downarrow\rangle$, then the singlet state is the state of zero total angular momentum and the three triplet states are the states of angular momentum 1 with z-components $-\hbar$, 0, and $\hbar$.

The four states (2.64) and (2.65) are also eigenstates of $\frac{1}{2}(\sigma_z^{(1)}+\sigma_z^{(2)})$, the three triplet states (in the order in which they appear in (2.64)) having eigenvalues -1, 1, and 0, and the singlet state having eigenvalue 0.

Note also that the first two triplet states in (2.64) are eigenstates of $\frac{1}{2}(\sigma_z^{(1)}-\sigma_z^{(2)})$ with eigenvalue 0, while $\frac{1}{2}(\sigma_z^{(1)}-\sigma_z^{(2)})$ takes the third of the triplet states into the singlet state, and vice-versa.

So the eigenstates of the Hamiltonian

$$\begin{aligned}\mathcal{H} &= J\,\sigma^{(1)}\cdot\sigma^{(2)} + H_1\sigma_z^{(1)} + H_2\sigma_z^{(2)} \\ &= J\,\sigma^{(1)}\cdot\sigma^{(2)} + H_+\tfrac{1}{2}(\sigma_z^{(1)}+\sigma_z^{(2)}) + H_-\tfrac{1}{2}(\sigma_z^{(1)}-\sigma_z^{(2)})\,,\end{aligned} \tag{2.66}$$

where

$$H_\pm = H_1 \pm H_2, \tag{2.67}$$

can be taken to be the first of the two triplet states (2.64) and two appropriately chosen orthogonal linear combinations of the third triplet state and the singlet state (2.65).

The eigenvalues of $\mathcal{H}$ associated with the first and second triplet states are J– H_+ and J+H_+ ; those associated with the last two states are the eigenvalues of the matrix

of $\mathcal{H}$ in the space spanned by the last two; i.e. $-J \pm \sqrt{4J^2 + H_-^2}$

Now the four states (2.64) and (2.65) are also eigenstates of **cZ**, the first of the three triplet states having eigenvalue –1 and the other three having eigenvalue 1.

Consequently these eigenstates of $\mathcal{H}$ are also eigenstates of **cZ** with respective eigenvalues –1, 1, 1, and 1.

We will therefore produce **cZ** (to within a constant phase factor) if we can chose the exchange coupling J, the magnetic fields H_1 and H_2, and the time t during which $\mathcal{H}$ acts to satisfy:

$$-e^{it(J-H_+)} = e^{it(J+H_+)} = e^{it(-J+\sqrt{4J^2+H_-^2})} = e^{it(-J-\sqrt{4J^2+H_-^2})}. \qquad (2.68)$$

The last equality is equivalent to

$$e^{2it\sqrt{4J^2+H_-^2}} = 1, \quad \text{or} \quad e^{it\sqrt{4J^2+H_-^2}} = \pm 1; \tag{2.69}$$

the first is equivalent to

$$e^{2itH_+} = -1, \quad \text{or} \quad e^{itH_+} = \pm i; \tag{2.70}$$

and the second is equivalent to

$$e^{-2itJ} = e^{itH_+} e^{-it\sqrt{4J^2+H_-^2}}. \tag{2.71}$$

The identities (2.69) and (2.70) require the right side of (2.71) to be ±i.

For the (positive) time t for which the gate acts to be as small as possible we should choose –i, which gives

$$Jt = \pi/4. \tag{2.72}$$

With this value of t we can satisfy (2.69) (with the minus sign) and (2.70) (with the plus sign) by taking $\sqrt{4J^2 + H_-^2} = 4J$ and $H_+ = 2J$.

So we can produce the gate **cZ** (to within an overall constant phase factor) by taking the magnetic fields in the Hamiltonian (2.66) and the time for which it acts to be related to the exchange coupling by

$$H_+ = 2J, \quad H_- = 2\sqrt{3}J, \quad t = \tfrac{1}{4}\pi/J, \tag{2.73}$$

or, in terms of the fields on each spin,

$$H_1 = (1+\sqrt{3})J, \quad H_2 = (1-\sqrt{3})J, \quad t = \tfrac{1}{4}\pi/J. \tag{2.74}$$

Note the curious fact that although, as (2.56) makes explicit, the gate **cZ** acts symmetrically on the two spins, the realization of **cZ** by the unitary transformation $e^{i\mathcal{H}t}$ requires the fields acting on the spins to break that symmetry.

Of course the symmetry survives in the fact that the alternative choice of fields $H_1 = (1-\sqrt{3})$ J, $H_2 = (1+\sqrt{3})$. J works just as well.

Part 3. Breaking RSA Encryption with a Quantum Computer: Class 13

Shor's Factoring Algorithm

In Simon's problem we are presented with a subroutine which calculates a function $f(x)$.

We are told that f satisfies $f(x) = f(y)$ for distinct x and y if and only if $y = x \oplus a$, where $\oplus$ denotes the bitwise modulo-2 sum of the n-bit integers a and x.

The number of times a classical computer must invoke the subroutine to determine a grows exponentially with n, but with a quantum computer it grows only linearly.

This is a rather artificial problem, of interest primarily because it gives a simple demonstration of the remarkable computational power a quantum computer can possess.

Simon's problem amounts to finding the unknown period of a function on n-bit integers that is "periodic" under bitwise modulo-2 addition.

A much more natural problem is to find the period r of a function on the integers that is periodic under ordinary addition.

Such a function f satisfies $f(x) = f(y)$ for distinct x and y if and only if x and y differ by an integral multiple of r.

Finding the period of such a periodic function turns out to be the key to factoring products of large prime numbers, a mathematically natural problem with quite practical applications.

One might think that finding the period of a periodic function ought to be easy, but that is only because when one thinks of periodic functions one tends to think of slowly varying continuous functions (like the sine function) whose structure at a small subset of points within a period can give powerful clues about what that period might be.

A better kind of periodic function to keep in mind from the beginning is a function on the integers whose values within a period r are completely random, and therefore give no hint whatever of the value of r.

The best known classical algorithms for finding the period r of such a periodic function take a time that grows faster than any power of the number n of bits of r (exponentially with $n^{1/3}$).

But in 1994 Peter Shor discovered that one can exploit the power of a quantum computer to learn the period r, with probability arbitrarily close to one, in a time that scales only a little faster than n^3.

Shor's discovery is of considerable practical interest because the ability to find periods efficiently, combined with some number-theoretic tricks, enables one to factor efficiently the product of two large prime numbers.

The very great computational effort required by all known classical factorization techniques underlies the security of the widely used RSA method of encryption.

Any computer that can efficiently find periods would be an enormous threat to the security of both military and commercial communications.

This is why research into the feasibility of quantum computers is a matter of considerable interest in the worlds of war and business.

A. Number theoretic preliminaries

Although the elementary number-theoretic tricks that underly the R(ivest)S(hamir)A((Adleman) method of encryption have nothing directly to do with how a quantum computer finds periods, they motivate the problem that Shor's quantum-computational algorithm so effectively solves.

Furthermore, examining the number-theoretic basis of RSA encryption reveals that Shor's period-finding algorithm can be used to decode it directly, without having to take a detour into factorization.

Now we must do some mathematics needed to proceed.

———————————-

Math.1 - Some elementary group theory.

A set of positive integers less than N constitutes a group under multiplication modulo N if the set

(a) contains 1,

(b) contains the modulo-N inverse of any of its members, and

(c) contains the the modulo-N products of all pairs of its members.

A subset of a group meeting conditions (a)-(c) is called a subgroup.

The number of members of a group is called the order of the group.

An important result of the elementary theory of finite groups (Lagrange's theorem) is that the order of any of its subgroups is a divisor of the order of the group itself.

The paragraphs that follow establish these statements (will not cover these in class unless requested).

If S is any subset of a group G (not necessarily a subgroup) and a is any member of G (which might or might not be in S), define aS (called a *coset* of S) to be the set of all members of G of the form $g = as$ where s is any member of S. (Throughout this appendix equality will be taken to mean equality modulo N.) Distinct members of S give rise to distinct members of aS, for if s and s' are in S and $as = as'$, then multiplying both sides by the inverse of a gives $s = s'$. So any coset aS has the same number of members as S itself.

If the subset S is a *subgroup* of G and a is a member s of S, then every member of the coset sS must be in S. Since sS has as many distinct members as S has, $sS = S$. If two cosets aS and bS of a subgroup S have a common member then there are members s and s' of S that satisfy $as = bs'$, so $(as)S = (bs')S$. But $(as)S = a(sS) = aS$, and similarly $(bs')S = bS$. Therefore $aS = bS$: two cosets of a *subgroup* are either identical or have no members in common.

If S is a subgroup then since 1 is in S, any member a of G is in the coset aS. Since every member of G is thus in some coset, and since the cosets of a subgroup are either identical or disjoint, it follows that the distinct cosets of a subgroup S partition the whole group G into disjoint subsets, each of which has the same number of members as S. Consequently the total number of members of G must be an integral multiple of the number of members of any of its subgroups S: *The order of any subgroup S is a divisor of the order of the whole group G.*

Of particular interest is the subgroup given by all the distinct powers of any particular member a of G.

Since G is a finite set, the set of distinct powers of a is also finite, and therefore for some n and m with $n > m$ we must have $a^n = a^m$, or $a^{(n-m)} = 1$.

The *order* of a is defined to be the smallest non-zero k with $a^k = 1$.

The subset $a, a^2, \ldots, a^k$ of G is a subgroup of G, since it contains $1 = a^k$, and the inverses and products of all its members.

It is called the subgroup generated by a, and its order is the order k of a.

Since the order of any subgroup of G divides the order of G, we conclude that the order of any member of G divides the order of G.

This is all the group theory one needs to know to understand RSA encryption.

Class 13 Now believing the mathematics and using it!

Now continuing our discussion, the basic algebraic entities that underly RSA encryption are finite groups, where the group operation is multiplication modulo some fixed integer N.

Those of you not confident with mathematics, just have to believe the conclusions drawn in the discussion.

In modulo-N arithmetic all integers that differ by multiples of N are identified, so there are only N distinct quantities which can be represented by $0, 1, \ldots N-1$.

For example $5 \times 6 \equiv 2 \pmod 7$ since $5 \times 6 = 30 = 4 \times 7 + 2$.

One writes $\equiv \pmod N$ to emphasize that the equality is only up to a multiple of N, reserving $=$ for strict equality.

One can develop the results that follow using ordinary arithmetic rather than group theory, but the group theoretic approach is simpler and uses properties of groups so elementary that they can all be derived from the basic definitions in a single page.

Let G_N be the set of all positive integers less than N (including 1) that have no factors in common with N.

Since factoring into primes is unique, the product of two numbers in G_N (either the ordinary or the modulo-N product) also has no factors in common with N, so G_N is *closed* (all product are in the group)under multiplication modulo N.

If a, b, and c are in G_N with $ab \equiv ac \pmod N$, then $a(b-c)$ is a multiple of N, and since a has no factors in common with N, it must be that $b-c$ is a multiple of N, so $b \equiv c \pmod N$.

It follows that multiplication modulo N by a fixed member a of G_N simply *permutes* all the members of the finite set G_N.

Since 1 is a member of G_N, there must be some d in G_N satisfying $ad = 1$, i.e. a *must* have a multiplicative inverse in G_N.

Thus G_N satisfies the conditions for it to be a group under modulo-N multiplication.

Every member a of a group G is characterized by its *order* k, the smallest integer for which (in the case of G_N)

$$a^k \equiv 1 \pmod{N}. \tag{3.1}$$

As shown later, the order of every member of G is a divisor of the number of members of G, which is called the *order* of G.

If p is a prime number, then the group G_p contains $p - 1$ numbers, since no positive integer less than p has factors in common with p. Since $p - 1$ is then a multiple of the order k of any a in G_p, it *follows* from (3.1) that any integer a less than p satisfies

$$a^{p-1} \equiv 1 \pmod{p}. \tag{3.2}$$

This relation, known as *Fermat's little theorem*, extends to arbitrary integers a not divisible by p , since any such a is of the form a = mp + a′ with m an integer and a′ less than p.

RSA encryption exploits an extension of Fermat's little theorem to a case characterized by two distinct primes, p and q.

If an integer a is divisible by neither p nor q, then no power of a is divisible by either p or q.

Since, in particular, aq −1 is not divisible by p, we conclude from (3.2) that

$$a^{p-1} \equiv 1 \pmod{p}.$$

$$[a^{q-1}]^{p-1} \equiv 1 \pmod{p}. \tag{3.3}$$

For the same reason

$$[a^{p-1}]^{q-1} \equiv 1 \pmod{q}. \tag{3.4}$$

The relations (3.3) and (3.4) *state* that there are integers m and n for which

$$a^{(q-1)(p-1)} = 1 + mp \tag{3.5}$$

and

$$a^{(q-1)(p-1)} = 1 + nq. \tag{3.6}$$

Taken together (3.5) and (3.6) require that mp = nq, which is possible with p and q distinct primes only if m is a multiple of q, and n is the *same* multiple of p.

So

$$a^{(q-1)(p-1)} = 1 + kpq, \tag{3.7}$$

and therefore

$$a^{(q-1)(p-1)} \equiv 1 \pmod{pq}, \tag{3.8}$$

for any integer a divisible by neither of the distinct primes p and q.

As an alternative derivation of (3.8), note that since a is divisible by neither p nor q, it has no factors in common with pq and is therefore in G_{pq}.

The number of elements of G_{pq} is pq − 1− (p− 1) − (q− 1) = (p− 1)(q− 1), since there are pq− 1 integers less than pq among which are p − 1 multiples of q and another q − 1 multiples of p.

Eq. (3.8) follows because the order (p − 1)(q − 1) of G_{pq} must be a multiple of the order of a.

We get the version of (3.8) that is the *basis* for RSA encryption by taking any integral power s of (3.8) and multiplying both sides by a:

$$a^{1+s(q-1)(p-1)} \equiv a \pmod{pq}. \tag{3.9}$$

An unimportant remark: the relation (3.9), unlike (3.8), holds even for integers a that are divisible by p or q.

It holds trivially when a is a multiple of pq.

And if a is divisible by just one of p or q , let $a = kq$.

Since a is not divisible by p neither is any power of a , and therefore Fermat's little theorem tells us that $[a^{s(q-1)}]^{p-1} = 1 + np$ for some integer n.

Multiplying both sides by a we have $a^{1+s(q-1)(p-1)} \equiv a + nap \equiv a + nkqp$, so (3.9) continues to hold.

Note finally that if c is an integer having no factor in common with $(p-1)(q-1)$ then c is in $G_{(p-1)(q-1)}$ and therefore has an inverse in $G_{(p-1)(q-1)}$; i.e. there is a d in $G_{(p-1)(q-1)}$ satisfying

$$cd \equiv 1 \ (\mathrm{mod}\ (p-1)(q-1)). \tag{3.10}$$

So for some integer s,

$$cd = 1 + s(p-1)(q-1). \tag{3.11}$$

In view of (3.11) and (3.9), any integer a must satisfy

$$a^{cd} \equiv a \pmod{pq}. \tag{3.12}$$

So if

$$b \equiv a^{c} \pmod{pq}, \tag{3.13}$$

then

$$b^{d} \equiv a \pmod{pq}. \tag{3.14}$$

The elementary numerical facts summarized here constitute the entire arithmetic basis for RSA encryption.

So, relying on the mathematicians having done their work correctly, we will just use all the results.

B. RSA encryption

Bob wants to receive a message from Alice encoded so that he alone can read it.

To do this he picks two large (say 200 digit) prime numbers p and q.

He gives Alice, through a public channel, their product, N = pq and a large *encoding* number c that he has picked to have *no factors* in common with (p− 1)(q− 1).

He does not, however, reveal the separate values of p and q and, given the practical impossibility of factoring a 400 digit number with currently available computers, he is quite confident that neither Alice nor any eavesdropper Eve will ever be able to calculate p and q knowing *only* their product N.

Bob, however, because he does know p and q , and therefore (p – 1)(q – 1) , can find the multiplicative inverse d of c mod (p – 1)(q – 1), satisfying (3.10).

$$cd \equiv 1 \pmod{(p-1)(q-1)}.$$

He keeps d strictly to himself for use in decoding.

If you are having difficulty following the arguments, then just try to understand the results.

Alice encodes a message by representing it as a string of fewer than 400 decimal digits using, for example, some version of ASCII coding.

If her message requires more than 400 digits she chops it up into smaller pieces.

She interprets each such string as a decimal number a less than N.

Using the coding number c and the value of N = pq she received from Bob, she then calculates b ≡ ac (mod pq), and sends it on to Bob through a public channel.

With c typically a 200 digit number, you might think that this would itself be a huge computational task, but it is not as we will see.

When he receives b , Bob exploits his private knowledge of d to calculate bd (mod pq), which (3.14) assures him is Alice's original message a.

$$b^d \equiv a \pmod{pq}.$$

Were the eavesdropper Eve able to find the factors p and q of N, she could calculate $(p-1)(q-1)$ and find the decoding integer d from the publicly available coding integer c, the same way Bob did.

But factoring a number as large as N is far beyond her classical computational powers.

Period finding is of interest in this cryptographic setting not only because it can form the basis for efficient factoring (more on factoring in a math extension at the end), but also because it can lead Eve directly to an alternative way to decode Alice's message b without knowing or having to compute the factors p and q of N.

Here is how it works:

Eve uses her efficient period finding machine to calculate the order r of Alice's publicly available encoded message $b = a^c$ in G_{pq}.

I assume that Alice's unencoded message a, and hence her coded message b, is in G_{pq} — i.e. that a is *not* a multiple of p or q.

Since p and q are huge numbers the odds against a being such a multiple are astronomical.

But if Eve wants to be insanely careful she can find the greatest common factor of b and N, using the *Euclidean algorithm(see below)*.

In the grossly improbable case that it turns out not to be 1, Eve will have factored N and can decode Alice's message the same way Bob does.

Math.2 - The Euclidean algorithm.

We wish to find the greatest common divisor of two numbers f and c, with $f > c$.

The Euclidean algorithm is the *iterative* procedure that replaces f and c by $f' = c$ and $c' = f - \text{Int}[f/c]c$, where Int[x] is the largest integer less than or equal to x.

Evidently any factors common to f and c are also common to f' and c'.

Furthermore f' and c' decrease with each iteration and each iteration keeps $f' > c'$, *until* the procedure reaches $c' = 0$.

Let f_0 and c_0 be the values of f and c at the *last* stage before $c = 0$.

They are *still* divisible by all common factors of the original f and c, but in addition f_0 is divisible by c_0 (or the next stage would not be $c'_0 = 0$.)

Therefore c_0 is the greatest common divisor of the original f and c.

If f and c (less than f) have no common factors, then iterating the Euclidean algorithm eventually leads to $c_0 = 1$.

This stage must have been immediately preceded by a pair f_1 and c_1 satisfying $f_1 - mc_1 = 1$ for some integer m.

But f_1 and c_1 are given by explicit integral linear combinations of the pair at the preceding stage, f_2 and c_2.

That pair in turn are explicit integral linear combinations of f_3 and c_3, etc.

So one can work backwards through the iterations to construct integers j and k with $1 = jf + kc$.

We can express k as $lf + d$ with $1 < d < f$ and with l a (possibly negative) integer; d is then the inverse of c modulo f.

———————————-

Continuing our discussion using the mathematics ……

Since the order of any integer in G_{pq} divides the order $(p - 1)(q - 1)$ of G_{pq}, this gives her a divisor of $(p - 1)(q - 1)$, and it turns out that knowing such a divisor is just as useful for decoding Alice's message as is knowing $(p - 1)(q - 1)$ itself.

To see this note first that the order r of Alice's encoded message b = ac in G_{pq} is the same as the order of a .

This is because the subgroup of G_{pq} generated by a contains ac = b, and hence it contains the subgroup generated by b; but the subgroup generated by b contains bd = a, and hence the subgroup generated by a . Since each subgroup contains the other they must be identical.

Since the order of a or b is the number of elements in the subgroup it generates, their orders are the same.

So if Eve can find the order r of Alice's code message b, then she has also learned the order of Alice's original text a.

Since Bob has picked c to have no factors in common with (p – 1)(q – 1), and since r divides (p – 1)(q – 1), the coding integer c can have no factors in common with r.

So c is congruent modulo r to a member c′ of G_r, which has an inverse d′ in G_r, and d′ is also a modulo-r inverse of c:

$$cd' \equiv 1 \pmod{r}. \qquad (3.15)$$

Therefore given c (which Bob has publicly announced) and r (which Eve can get with her period-finding program from Alice's encoded message b and the publicly announced value of N = pq), it is easy for Eve to calculate d′ with a classical computer, using, modulo r , the same extension of the Euclidean algorithm that Bob used to find d , modulo (p− 1)(q− 1).

It then follows that for some integer m

$$b^{d'} \equiv a^{cd'} = a^{1+mr} = a\left(a^{r}\right)^{m} \equiv a \pmod{pq}. \tag{3.16}$$

Eve has thus used her ability to find periods efficiently to decode Alice's encoded message b = ac to reveal Alice's original message a.

This use of period finding to defeat RSA encryption is now summarized in Table 1.

BOB KNOWS	**ALICE KNOWS**	**PUBLIC KNOWS**
p and q (primes) c and d, $cd \equiv 1 \pmod{(p-1)(q-1)}$	a (her message) only c (not d) and only $N = pq$ $b \equiv a^c \pmod{N}$ (encoded message)	b (encoded message) only c (not d), and only $N = pq$
Decoding: $a \equiv b^d \pmod{N}$		**Quantum decoding:** Quantum computer finds r: $b^r \equiv 1 \pmod{N}$; Classical computer finds d': $cd' \equiv 1 \pmod{r}$; $a \equiv b^{d'} \pmod{N}$

Table 1. Summary of RSA encryption and how to break it with a fast period-finding routine on a quantum computer. Bob has chosen the encoding number c to have an inverse d modulo (p− 1)(q− 1) so c can have no factors in common with (p− 1)(q− 1). Since Alice's encoded message b is in G_{pq}, its order r is a factor of the order (p− 1)(q− 1) of G_{pq}. So c can have no factors in common with r, and therefore has an inverse d′ modulo r. Because b is a power of a and vice-versa, each has the same order r in G_{pq}. Therefore $b^{d'} \equiv a^{cd'} \equiv a^{1+mr} \equiv a$ modulo N.

C. Quantum period-finding: setting things up

So we can crack the RSA code if we have a fast way to find the period r of the known periodic function

$$f(x) = b^x \pmod{N}. \tag{3.17}$$

As remarked earlier, this might appear to be a simple task, especially since periodic functions of the special form (3.17) have the simplifying feature that $f(x+s) = f(x)$ only if s is a multiple of the period r.

If we can find two different values of x at which f has the same value, then we have found a multiple of the period r, and given a few random multiples of r we can, with high probability find r itself, i.e., given two random multiples kr and $k'r$, if k and k' have no common factors then r is their greatest common divisor, and is easily extracted using the Euclidean algorithm.

The probability of two random numbers having no common factors is greater than $1/2$, for the probability that they are both divisible by any prime p is $1/p^2$, and therefore the probability that they share no prime factors at all is primes $(1 - 1/p^2) = .6079\ldots (= 6/\pi^2)$.

So with a fairly small number of random multiples of r one can, with high probability, extract r.

But unlike the smooth, uncomplicated periodic functions one is used to dealing with, the function $b^x \pmod N$ looks like random noise within a period.

Its structure over any stretch of integers less than the period offers *no clue* about the value of that period.

In trying to determine the period with a classical computer by this direct approach one can do no better than to calculate f repeatedly for a random collection of integers until one finally gets a value that agrees with one of the values already calculated.

The scale of the problem is best measured by the number of bits n_0 in $N = pq$; 2^{n_0} is the smallest power of 2 that exceeds N.

If N is a 500 digit number — a typical size for cryptographic applications — n_0 will be around 1700.

This sets the scale for the number of bits in N and in the other relevant numbers a, b, and their modulo N period r.

To have an appreciable probability of finding r by random searching requires a number of evaluations of f that is exponential in n_0 (just like the classical situation in Simon's problem, described earlier)

Although there are ways to improve on random searching, using, for example, Fourier analysis, no classical approach is known that does not require a time that grows faster than any power of n_0.

With a quantum computer, however, quantum parallelism gets us tantalizingly close (but, as in Simon's problem, *not* close enough) to solving the problem with a single application of $\mathbf{U}_f$, and enables us to solve it completely with probability arbitrarily close to 1 in a time that grows only as a polynomial in n_0.

To deal with values of x and $f(x)$ between 0 and N, both the input and output registers must contain at least n_0 Qbits.

For reasons that will emerge later, however, to find the period r efficiently the input register must actually have $n = 2^{n_0}$ Qbits.

Doubling the number of Qbits in the input register ensures that the range of values of x for which $f(x)$ is calculated contains at least N full periods of f.

This redundancy turns out to be absolutely essential for a successful determination of the period by Shor's method. (We shall see later that if p and q both happen to be primes of the form $2^j + 1$ then — and only then — the method works without doubling the size of the input register.

Thus N = 15 can be a highly atypical test case for laboratory attempts to demonstrate Shor's algorithm for small p and q with real Qbits.)

We begin by using our quantum computer in the usual way to construct the state

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n |f(x)\rangle_{n_0} \tag{3.18}$$

with a single application of $\mathbf{U}_f$.

Later, we take a closer look at how this might efficiently be done in the case of interest $f(x) = b^x \pmod N$.

Once the state of the registers has become (3.18), we can measure the n-Qbit output register.

It is not, in fact, necessary to measure the output register.

One can continue to work with the full state (3.18) in which one breaks down the sum on x into a sum over all the different values of f and a sum over all the values of x associated with each value of f.

The only function of the measurement is to eliminate a lot of uninteresting additional structure, coming from the sum on the values of f, that plays no role beyond making many of the subsequent expressions somewhat lengthier.

Thus, if the measurement yields the value f_0, then the generalized Born rule tells us that the state of the n-Qbit input register can be taken to be

$$|\Psi\rangle_n = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle_n. \qquad (3.19)$$

Here x_0 is the smallest value of x $(0 \le x_0 < r)$ at which $f(x_0) = f_0$, and m is the smallest integer for which $mr + x_0 \ge 2^n$, so

$$m = \left[\frac{2^n}{r}\right] \quad \text{or} \quad m = \left[\frac{2^n}{r}\right] + 1, \qquad (3.20)$$

depending on the value of x_0 (where [x] is the integral part of x).

As in the examples seen earlier, if we could produce a small number of identical copies of the state (3.19) we would be done, for a measurement in the computational basis would yield a random one of the values $x_0 + kr$, and the difference between the results of pairs of measurements on such identical copies would give us a collection of random multiples of r from which r itself could straightforwardly be extracted, as noted above.

But we are again done in by the *no-cloning* theorem.

All we can extract is a *single* value of $x_0 + kr$ for unknown random x_0, which is completely useless for determining r.

And, of course, if we ran the whole algorithm again, we would end up with a state of the form (3.19) for another random value of x_0, which would permit *no* useful comparison with what we had learned from the first run.

But, as with Simon's problem, we can do something more *clever* to the state (3.19) *before* making our final measurement.

The problem is the displacement by the unknown random x_0, which prevents any information about r from being extracted in a single measurement.

We need a unitary transformation that transforms the x_0 dependence into a harmless overall phase factor.

This is accomplished with the *Quantum Fourier Transform*.

D. The Quantum Fourier Transform

The heart of Shor's algorithm is a superfast quantum Fourier transform procedure, carried out by an efficient quantum circuit built out of one- and 2-Qbit gates.

The n-Qbit quantum Fourier transform is *defined* to be that unitary transformation U_{FT} whose action on the computational basis is given by

$$\mathbf{U}_{FT}|x\rangle_n = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i x y/2^n} |y\rangle_n. \qquad (3.21)$$

The product xy is here ordinary multiplication.

Continuing where we left off,

$$\mathbf{U}_{FT}|x\rangle_n = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i x y/2^n} |y\rangle_n. \tag{3.21}$$

A warning to physicists (which others can ignore): This looks deceptively like the (discretized) transformation from a position to a momentum representation, and one's first reaction might be that it is (perhaps disappointingly) familiar.

But it has, in fact, an entirely different character.

The number x is the integer represented by the state $|x\rangle$, and *not* the position of anything.

Changing x to $x + 1$ induces an arithmetically natural but physically quite unnatural transformation on the computational basis states, determined by the laws of binary addition, including carrying.

It bears *no resemblance* to anything that could be associated with a spatial translation in the physical space of Qbits.

So your eyes should not glaze over, and you should *regard* $\mathbf{U}_{FT}$ as a new and unfamiliar physical transformation of Qbits.

One easily verifies that $\mathbf{U}_{FT}\,|x\rangle$ is normalized to unity and that $\mathbf{U}_{FT}\,|x\rangle$ is orthogonal to $\mathbf{U}_{FT}\,|x'\rangle$ unless $x = x'$, so $\mathbf{U}_{FT}$ is unitary.

Unitarity *also* emerges directly from the analysis that follows, which explicitly constructs $\mathbf{U}_{FT}$ out of one- and 2-Qbit unitary gates.

The unitary $\mathbf{U}_{FT}$ is useful because, as one also easily verifies, applied to a superposition of states $|x\rangle$ with complex amplitudes $\gamma(x)$, it produces another superposition with amplitudes that are related to $\gamma(x)$ by the appropriate discrete Fourier transform:

$$\mathbf{U}_{FT}\Big(\sum_{x=0}^{2^n-1}\gamma(x)|x\rangle\Big)=\sum_{x=0}^{2^n-1}\tilde{\gamma}(x)|x\rangle, \tag{3.22}$$

where

$$\tilde{\gamma}(x)=\frac{1}{2^{n/2}}\sum_{y=0}^{2^n-1}e^{2\pi ixy/2^n}\gamma(y). \tag{3.23}$$

The celebrated classical fast Fourier transform is an algorithm requiring a time that grows with the number of bits as $n2^n$ (rather than $(2^n)^2$ as the obvious direct approach would require) to evaluate $\tilde{\gamma}$.

But there is a quantum algorithm for executing the unitary transformation $\mathbf{U}_{FT}$ exponentially faster than fast, in a time that grows only as n^2.

The *catch* as usual, is that one does not end up knowing the complete set of Fourier coefficients, as one does after applying the classical fast Fourier transform.

One only has n Qbits described by the state given by the right side of (3.22), and as we have repeatedly noted, having a collection of Qbits in a given state does not enable one to learn what that state actually is.

$$\mathbf{U}_{FT}\Big(\sum_{x=0}^{2^n-1}\gamma(x)|x\rangle\Big)=\sum_{x=0}^{2^n-1}\tilde{\gamma}(x)|x\rangle,$$

There is no way to extract all the Fourier coefficients $\tilde{\gamma}$, given an n-Qbit register in the state (3.22).

Nevertheless, we shall see later that if γ is a periodic function with a period that is no bigger than $2^{n/2}$, then a register in the state (3.22) can give *powerful clues* about the precise value of the period r, even though r can be hundreds of digits long,

To construct such a circuit to execute the quantum Fourier transform (QFT) $\mathbf{U}_{FT}$, it is convenient to introduce an n-Qbit unitary operator $\mathcal{Z}$, diagonal(eigenvectors) in the computational basis:

$$\mathcal{Z}|y\rangle_n = e^{2\pi i y/2^n}|y\rangle_n. \qquad (3.25)$$

This can be viewed as a generalization to n Qbits of the 1-Qbit operator **Z**, to which it reduces to when n = 1.

Using the familiar relation

$$\mathbf{H}^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} |y\rangle_n, \tag{3.26}$$

we can reexpress the definition (3.21) as

$$\mathbf{U}_{FT}|x\rangle_n = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i x y/2^n} |y\rangle_n.$$

$$\mathbf{U}_{FT}|x\rangle_n = \mathcal{Z}^x \mathbf{H}^{\otimes n}|0\rangle_n. \tag{3.27}$$

This gives $\mathbf{U}_{FT}\,|x\rangle_n$ as an x-dependent operator acting on the state $|0\rangle$.

We next reexpress the right side of (3.27) as an x-independent operator acting on the state $|x\rangle_n$.

Since the computational basis states $|x\rangle_n$ are a basis, we will then have found an alternative expression for $\mathbf{U}_{FT}$ itself.

The detailed construction of this alternative form for (3.27) is made much more transparent by specializing to the case of 4 Qbits.

We will find that the structure that emerges in the case $n = 4$ has an entirely obvious extension to general n ; dealing with the case of general n from the start would only *obscure* this simple structure.

When $n = 4$ we want to find an appropriate form for

$$\mathbf{U}_{FT}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle = \mathcal{Z}^x\mathbf{H}_3\mathbf{H}_2\mathbf{H}_1\mathbf{H}_0|0\rangle|0\rangle|0\rangle|0\rangle. \qquad (3.28)$$

As usual, we number the Qbits by the power of 2 with which they are associated, with the least significant on the right, so that, reading from right to left, the Qbits are labeled 0, 1, 2, and 3; $\mathbf{H}_i$ acts on the Qbit labeled i (and as the *identity* on all other Qbits).

$$\mathcal{Z}|y\rangle_n = e^{2\pi iy/2^n}|y\rangle_n.$$

If $|y\rangle_4 = |y_3\rangle|y_2\rangle|y_1\rangle|y_0\rangle$ in (3.25) so that $y = 8y_3 + 4y_2 + 2y_1 + y_0$, then the operator $\mathcal{Z}$ can be constructed out of single-Qbit number operators:

$$\mathcal{Z} = \exp\left[\tfrac{i\pi}{8}\left(8\mathbf{n}_3 + 4\mathbf{n}_2 + 2\mathbf{n}_1 + \mathbf{n}_0\right)\right]. \tag{3.29}$$

The operator $\mathcal{Z}^x$ appearing in (3.28) then becomes

$$\mathcal{Z}^x = \exp\left[\tfrac{i\pi}{8}\left(8x_3 + 4x_2 + 2x_1 + x_0\right)\left(8\mathbf{n}_3 + 4\mathbf{n}_2 + 2\mathbf{n}_1 + \mathbf{n}_0\right)\right]. \tag{3.30}$$

Because the 1-Qbit operator $\exp(2\pi i\mathbf{n})$ acts as the identity on either of the 1-Qbit states $|0\rangle$ or $|1\rangle$, and because any 1-Qbit state is a superposition of these two, $\mathbf{n}$ obeys the operator identity

$$\exp(2\pi i\mathbf{n}) = \mathbf{1}. \tag{3.31}$$

Therefore in multiplying out the two terms

$$\left(8x_3 + 4x_2 + 2x_1 + x_0\right)\left(8\mathbf{n}_3 + 4\mathbf{n}_2 + 2\mathbf{n}_1 + \mathbf{n}_0\right) \tag{3.32}$$

appearing in the exponential (3.30), we can drop those whose coefficient is a power of 2 greater than 8, getting

$$\mathcal{Z}^x = \exp\left[i\pi\left(x_0\mathbf{n}_3+(x_1+\tfrac{1}{2}x_0)\mathbf{n}_2+(x_2+\tfrac{1}{2}x_1+\tfrac{1}{4}x_0)\mathbf{n}_1+(x_3+\tfrac{1}{2}x_2+\tfrac{1}{4}x_1+\tfrac{1}{8}x_0)\mathbf{n}_0\right)\right]. \quad (3.33)$$

Note next that the number and Hadamard operators for any single Qbit obey the relation

$$\exp(i\pi x\mathbf{n})\mathbf{H}|0\rangle = \mathbf{H}|x\rangle. \quad (3.34)$$

This is trivial when $x = 0$, and when $x = 1$ it reduces to the correct statement

$$(-1)^{\mathbf{n}}\tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \quad (3.35)$$

Alternatively, note that $\exp(i\pi\mathbf{n}) = \mathbf{Z}$ and $\mathbf{ZH} = \mathbf{HX}$.

The effect on (3.28) of the four terms in (3.33) that do not contain factors of $\frac{1}{2}$, $\frac{1}{4}$, or $\frac{1}{8}$ is to produce the generalization of (3.34) to several Qbits:

(after a lot of algebra)

$$\exp\left[i\pi\left(x_0\mathbf{n}_3 + x_1\mathbf{n}_2 + x_2\mathbf{n}_1 + x_3\mathbf{n}_0\right)\right]\mathbf{H}_3\mathbf{H}_2\mathbf{H}_1\mathbf{H}_0|0\rangle|0\rangle|0\rangle|0\rangle$$

$$= \left[\exp\left(i\pi x_0\mathbf{n}_3\right)\mathbf{H}_3\right]\left[\exp\left(i\pi x_1\mathbf{n}_2\right)\mathbf{H}_2\right]\left[\exp\left(i\pi x_2\mathbf{n}_1\right)\mathbf{H}_1\right]\left[\exp\left(i\pi x_3\mathbf{n}_0\right)\mathbf{H}_0\right]|0\rangle|0\rangle|0\rangle|0\rangle$$

$$= \mathbf{H}_3\mathbf{H}_2\mathbf{H}_1\mathbf{H}_0|x_0\rangle|x_1\rangle|x_2\rangle|x_3\rangle. \qquad (3.36)$$

We have used the fact that number operators associated with different Qbits commute with one another.

Since $\mathbf{n}_i$ has the coefficient x_{3-i}, the Qbit labeled i in the final form is in the state $|x_{3-i}\rangle$.

Also note the inversion: because the number operator $\mathbf{n}_i$ is multiplied by x_{3-i} on the left side of (3.36), the state of the Qbit labeled i on the right is, as we said, $|x_{3-i}\rangle$.

The remaining six terms in (3.33) (containing fractional coefficients) further convert (3.28) to the form

$$\mathbf{U}_{FT}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle =$$

$$\exp\left[i\pi\left(\tfrac{1}{2}x_0\mathbf{n}_2+(\tfrac{1}{2}x_1+\tfrac{1}{4}x_0)\mathbf{n}_1+(\tfrac{1}{2}x_2+\tfrac{1}{4}x_1+\tfrac{1}{8}x_0)\mathbf{n}_0\right)\right]\mathbf{H}_3\mathbf{H}_2\mathbf{H}_1\mathbf{H}_0|x_0\rangle|x_1\rangle|x_2\rangle|x_3\rangle. \quad (3.37)$$

Since the Hadamard transformation $\mathbf{H}_i$ commutes with the number operator $\mathbf{n}_j$ when $i \neq j$, we can regroup the terms in (3.37) so that each number operator $\mathbf{n}_i$ appears immediately to the left of its corresponding Hadamard operator $\mathbf{H}_i$:

$$\mathbf{U}_{FT}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle = \mathbf{H}_3\exp\left[i\pi\mathbf{n}_2\tfrac{1}{2}x_0\right]\mathbf{H}_2\exp\left[i\pi\mathbf{n}_1(\tfrac{1}{2}x_1+\tfrac{1}{4}x_0)\right]\mathbf{H}_1\times$$

$$\times\exp\left[i\pi\mathbf{n}_0(\tfrac{1}{2}x_2+\tfrac{1}{4}x_1+\tfrac{1}{8}x_0)\right]\mathbf{H}_0|x_0\rangle|x_1\rangle|x_2\rangle|x_3\rangle. \quad (3.38)$$

The state $|x_0\rangle|x_1\rangle|x_2\rangle|x_3\rangle$ is an eigenstate of the number operators $\mathbf{n}_3, \mathbf{n}_2, \mathbf{n}_1, \mathbf{n}_0$ with respective eigenvalues x_0, x_1, x_2, x_3.

If we did not have to worry about Hadamard operators interposing themselves between number operators and their eigenstates, we could replace each x_i in (3.38) by the number operator $\mathbf{n}_{3-i}$ of which it is the eigenvalue to get:

$$\mathbf{U}_{FT}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle = \mathbf{H}_3 \exp\left[i\pi \tfrac{1}{2}\mathbf{n}_2\mathbf{n}_3\right]\mathbf{H}_2 \exp\left[i\pi\mathbf{n}_1(\tfrac{1}{2}\mathbf{n}_2 + \tfrac{1}{4}\mathbf{n}_3)\right]\times$$
$$\times\mathbf{H}_1 \exp\left[i\pi\mathbf{n}_0(\tfrac{1}{2}\mathbf{n}_1 + \tfrac{1}{4}\mathbf{n}_2 + \tfrac{1}{8}\mathbf{n}_3)\right]\mathbf{H}_0|x_0\rangle|x_1\rangle|x_2\rangle|x_3\rangle. \qquad (3.39)$$

But as (3.39) makes clear, we do indeed not have to *worry*, because every $\mathbf{H}_i$ appears safely to the left of every $\mathbf{n}_i$ that has replaced an x_{3-i} .

If we define 2-Qbit unitary operators by

$$\mathbf{V}_{ij} = \exp\left(i\pi\mathbf{n}_i\mathbf{n}_j/2^{|i-j|}\right), \qquad (3.40)$$

then (3.39) assumes the more readable form

$$\mathbf{U}_{FT}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle = \mathbf{H}_3\left(\mathbf{V}_{32}\mathbf{H}_2\right)\left(\mathbf{V}_{31}\mathbf{V}_{21}\mathbf{H}_1\right)\left(\mathbf{V}_{30}\mathbf{V}_{20}\mathbf{V}_{10}\mathbf{H}_0\right)|x_0\rangle|x_1\rangle|x_2\rangle|x_3\rangle. \qquad (3.41)$$

I have put in unnecessary parenthesis to guide the eye to the simple structure, whose generalization to more than four Qbits is, as promised, obvious - I hope you can see it!

If we define the unitary operator $\mathbf{P}$ to bring about the permutation of computational basis states

$$\mathbf{P}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle = |x_0\rangle|x_1\rangle|x_2\rangle|x_3\rangle \tag{3.42}$$

then (3.41) becomes

$$\mathbf{U}_{FT}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle = \mathbf{H}_3\big(\mathbf{V}_{32}\mathbf{H}_2\big)\big(\mathbf{V}_{31}\mathbf{V}_{21}\mathbf{H}_1\big)\big(\mathbf{V}_{30}\mathbf{V}_{20}\mathbf{V}_{10}\mathbf{H}_0\big)\mathbf{P}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle. \tag{3.43}$$

Since (3.43) holds for *all* computational basis states it holds for *arbitrary* states and is therefore *equivalent* to the operator identity

$$\mathbf{U}_{FT} = \mathbf{H}_3(\mathbf{V}_{32}\mathbf{H}_2)(\mathbf{V}_{31}\mathbf{V}_{21}\mathbf{H}_1)(\mathbf{V}_{30}\mathbf{V}_{20}\mathbf{V}_{10}\mathbf{H}_0)\mathbf{P}. \tag{3.44}$$

The form (3.44) expresses $\mathbf{U}_{\mathrm{FT}}$ as a product of unitary operators, thereby independently establishing what we already noted directly from its definition, that $\mathbf{U}_{\mathrm{FT}}$ is unitary.

More importantly it gives an *explicit* construction of $\mathbf{U}_{\mathrm{FT}}$ entirely out of one- and two-Qbit unitary gates, whose number *grows only* quadratically with the number n of Qbits.

The permutation **P** can be constructed out of cNOT gates and *one* additional Qbit, initially in the state $|0\rangle$ — an instructive exercise to think about — but in the application that follows it is much easier to build directly into the circuitry the rearranging of Qbits accomplished by **P**.

$$\mathbf{U}_{FT}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle = \mathbf{H}_3\left(\mathbf{V}_{32}\mathbf{H}_2\right)\left(\mathbf{V}_{31}\mathbf{V}_{21}\mathbf{H}_1\right)\left(\mathbf{V}_{30}\mathbf{V}_{20}\mathbf{V}_{10}\mathbf{H}_0\right)\mathbf{P}|x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle \tag{3.43}$$

A circuit diagram that compactly expresses the content of (3.43) is shown in Figure 3.1.

Figure 3.1

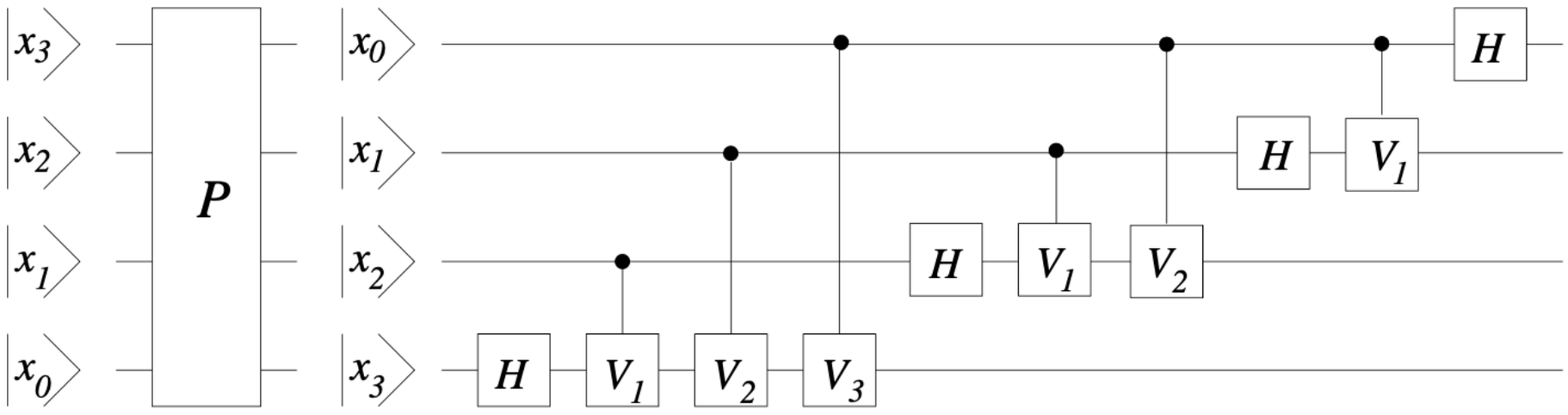
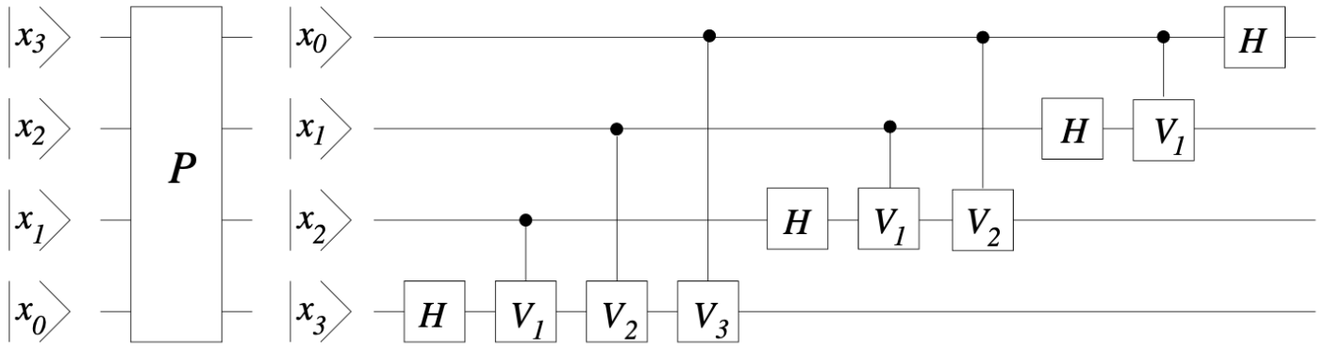


Figure 3.1. Diagram of a circuit that illustrates, for four Qbits, the construction of the quantum Fourier transform $\mathbf{U}_{\mathrm{FT}}$ defined in (3.21), as the product of one- and two-Qbit gates given in (3.43). (Note that in the figure the order in which the gates act is from left to right, while in (3.43) the order in which they act is from right to left.)



The convention, as usual, is that the Qbits representing more significant bits are represented by higher lines in the figure.

Note, however, that acting on the computational basis, the first gate **P** permutes the states of the Qbits, exchanging the states of the most and least significant Qbits, and the states of the next most significant and next least significant Qbits.

Rather than going to the trouble of constructing such a gate, it obviously makes more sense simply to reverse the convention for the input state, representing Qbits that represent more significant bits by lower lines in the figure.

This is what one usually encounters in diagrams of the quantum Fourier transform: the gate **P** is omitted and the conventional ordering of significant bits is reversed for the input.

The complete figure then reduces to the portion to the right of the permutation gate **P**. For the output, of course, the conventional ordering remains in effect.

If the input on the left is the computational basis state $|x\rangle_4 = |x_3\rangle|x_2\rangle|x_1\rangle|x_0\rangle$ the output will be $\mathbf{U}_{FT}\,|x\rangle_4$, the superposition (3.21) of computational basis states $|y\rangle_4 = |y_3\rangle|y_2\rangle|y_1\rangle|y_0\rangle$, defined in (3.21).

There is no need for the figure to have subscripts on the Hadamard gates appearing in (3.44), since each is explicitly attached to the line associated with the Qbit on which it acts.

For the same reason each two-Qbit controlled-V gate requires only a single subscript, which specifies the unitary operator $\mathbf{V}_k$ that acts on the target Qbit; the subscript is the number of "wires" the target Qbit is away from the control Qbit.

The explicit form of $\mathbf{V}_k$ is $e^{i\pi \mathbf{n}/2^k}$ where $\mathbf{n}$ is the number operator for the target Qbit.

In the computational basis the controlled-V gate acts as the identity unless control and target Qbits are both in the state $|1\rangle$, in which case it multiplies the state of the target Qbit by $e^{i\pi/2^k}$.

Note that the diagram introduces an artificial asymmetry into the 2-Qbit unitary gate $\mathbf{V}_{ij}$, by treating one Qbit as a control bit, which determines whether or not the unitary operator $e^{i\pi n/2|i-j|}$ acts on the target Qbit.

Although this is the conventional way of representing the circuit for the quantum Fourier transform, the figure could equally well have been drawn as it is in Figure 3.2.

This second, less conventional form, reveals a further simplification of great practical interest, if all the Qbits are measured as soon as the action of the quantum Fourier transformation is completed.

The simplification allows the 2-Qbit controlled-V gates to be replaced by single-Qbit gates that act or not, depending on the outcome of a prior measurement of the control Qbit.

This is described in the caption of Figure 3.2.

Figure 3.2

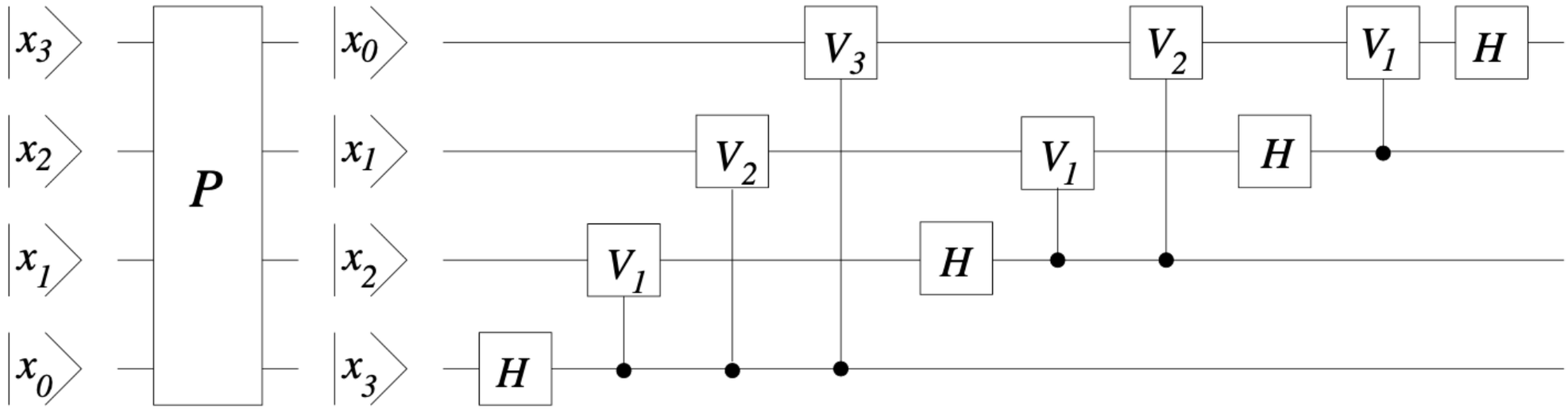
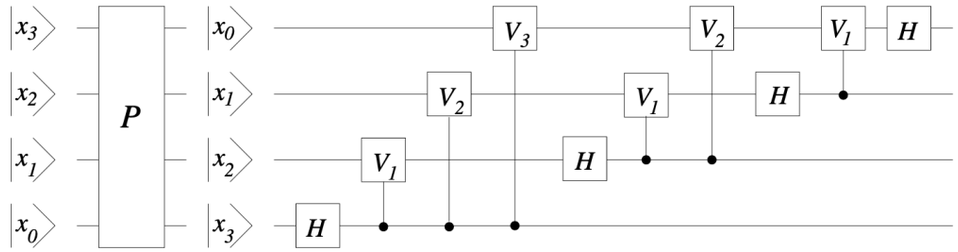


Figure 3.2. Since the action (3.40) of the controlled-V gates is symmetric in i and j, Figure 3.1 can be redrawn with control and target Qbits interchanged. If the quantum Fourier transformation is immediately followed by a measurement of all the Qbits, as it is when applied to period finding, then this equivalent representation of the quantum Fourier transformation reveals that the two-Qbit controlled-V gates can be replaced by one-Qbit unitary gates, which act or not depending on the outcome of earlier measurements.

To how this comes about consider first the bottom wire.



Once **H** and the three controlled-V gates have acted on it, nothing further happens to it until the final measurement of the least significant output Qbit.

If the result of that measurement is 1, the state of the four Qbits reduces to that component of the full superposition in which $\mathbf{V}_1$, $\mathbf{V}_2$, and $\mathbf{V}_3$ have acted on the three wires above the bottom wire; if the result of the measurement is 0, the four-Qbit state reduces to the component in which they have not acted.

We can produce exactly the same effect if we measure the least significant output Qbit immediately after **H** has acted on the bottom wire, before any of the other gates have acted, and then do or do not apply the three unitary transformations to the other three Qbits, depending on whether the outcome of the measurement is 1 or 0.

Next, we apply the Hadamard transformation to the second wire from the bottom.

We then immediately measure the next least significant output Qbit and, depending on the outcome, apply or not apply the appropriate one-Qbit unitary transformations to each of the remaining two Qbits.

Continuing in this way, we end up producing exactly the same statistical distribution of measurement results as we would have produced had we used the two-Qbit controlled-V gates, measuring none of the Qbits until the full unitary transformation $\mathbf{U}_{\mathrm{FT}}$ was produced.

This replacement of controlled operators (2-Qbit operators) by measurements followed by 1-Qbit operators is shown explicitly in Figure 3.3.

Figure 3.3

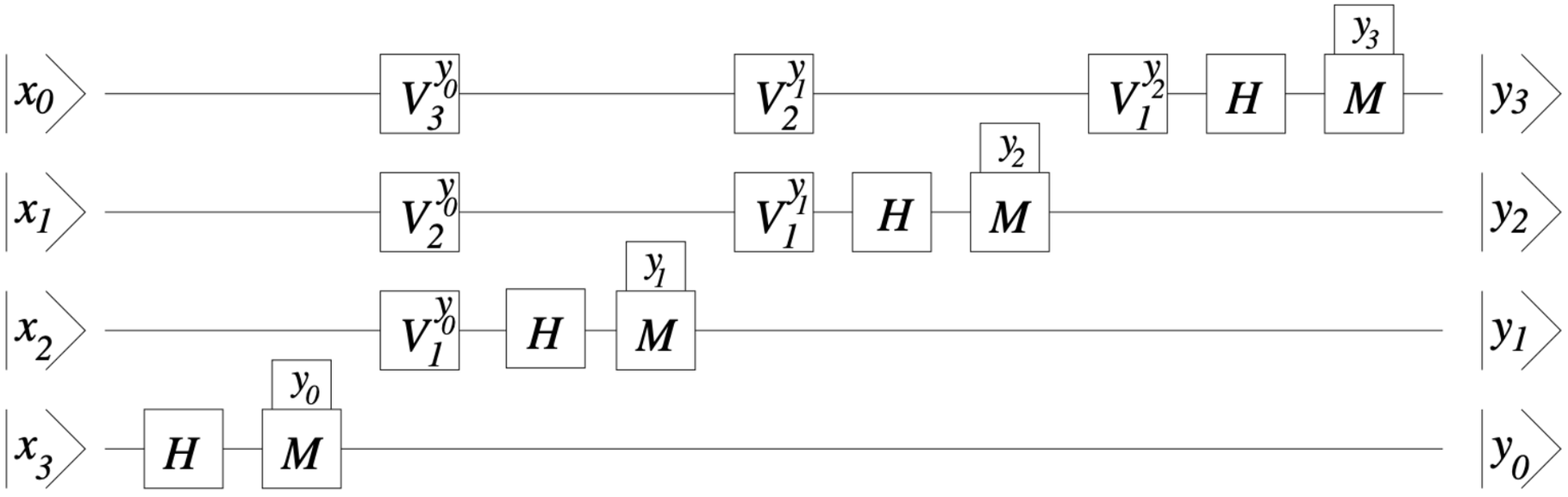


Figure 3.3. If the Qbits are all measured immediately after all the gates of the quantum Fourier transform have all acted, then the one-Qbit measurement gates can be applied to each Qbit immediately after the action of the Hadamard gate on that Qbit, and the controlled-V gates that follow the action of the Hadamard (see Figure 3.2) can be replaced by one-Qbit gates that act or not depending on whether the outcome y of the one-Qbit measurement is 1 or 0. Note that the final state on the right can now be indicated, since it is just the computational basis state associated with the outcomes of the measurements of the individual Qbits. If the input state is a computational basis state, the output state is entirely random, all states $|y\rangle_4$ having equal probability. However the distribution of output states can have a highly informative structure when the input is an appropriate superposition of computational basis states.

It is made possible by the (easily verified) fact that if a controlled operation, or a series of consecutive controlled operations all with the same control Qbit, is immediately followed by a measurement of the control Qbit, then the possible final states of all the Qbits and the probabilities of those states are exactly the same as they would be if the measurement of the control Qbit took place before the application of the controlled operation, and the target Qbit(s) were then acted upon or not, depending on whether the result of the prior measurement was 0 or 1.

We shall see that if one's aim is to find the period of the function f, one can indeed measure each Qbit immediately after applying the quantum Fourier transform.

So this replacement of controlled unitary gates by 1-Qbit unitary gates, which act or not depending on the outcome of the measurement, is a major simplification from the technological point of view, 1-Qbit unitaries being far easier to implement than 2-Qbit controlled gates.

The most attractive (but least common) way of representing the quantum Fourier transform with a circuit diagram is shown in Figure 3.4.

Figure 3.4

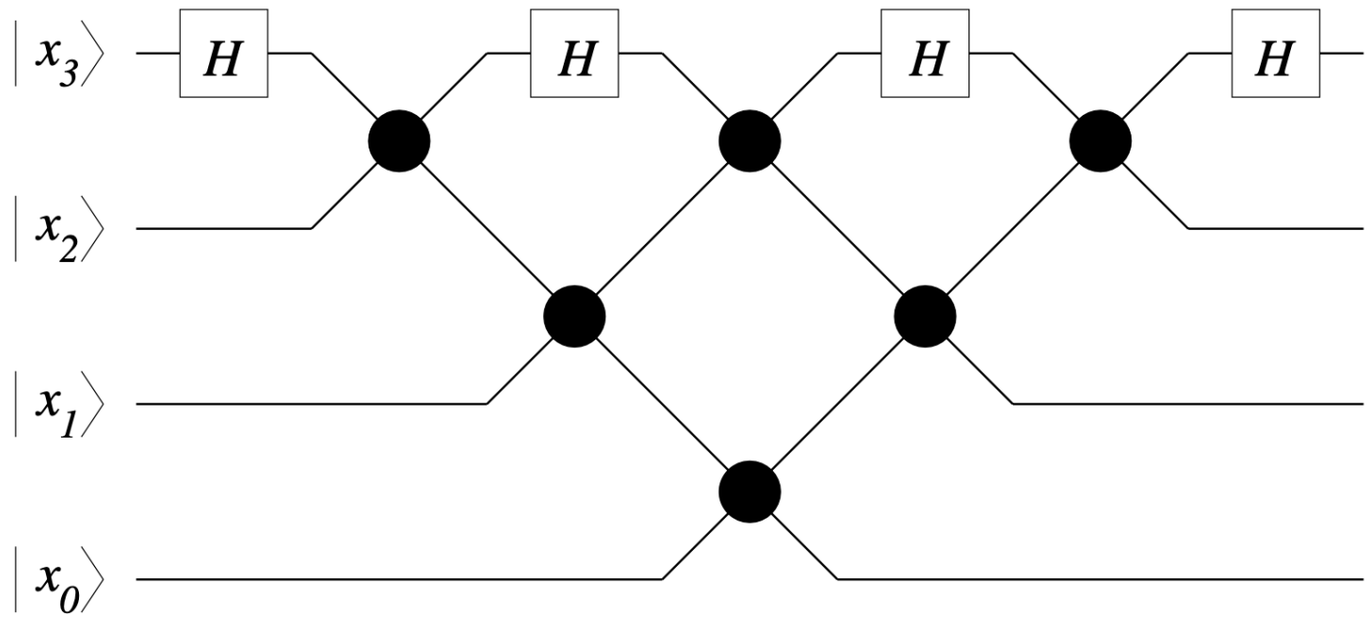
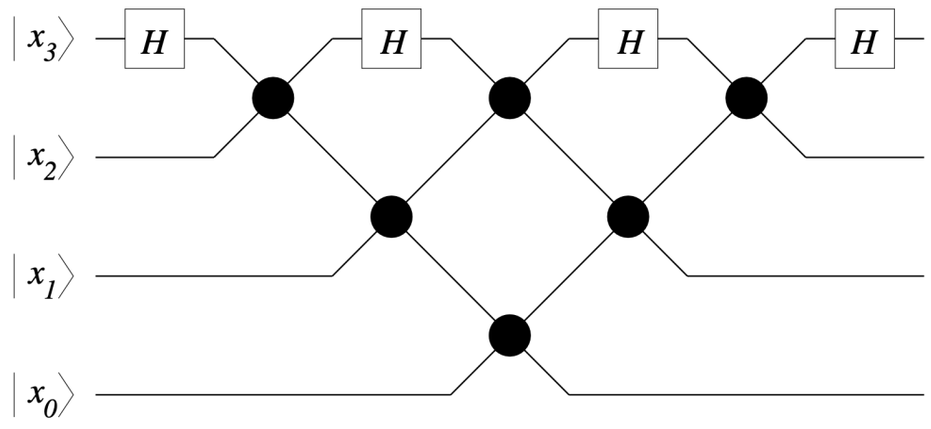


Figure 3.4. A much more symmetric way of drawing Figs. 1 or 2. (Although it is superior to the conventional diagram, it does not seem to have caught on.) The permutation **P** that in effect permutes the Qbits in the input register is now built into the diagram by using lines that no longer connect input register Qbits to output register Qbits at the same horizontal level. Because the lines now cross one another, the unitary operators **V** can be represented by the black circles at the intersection of the lines associated with the Qbits that they couple. This eliminates the artificial distinction between control and target Qbits used in Figs. 1 and 2. The form of each such operator is $\mathbf{V} = \exp(i\pi \mathbf{n}\mathbf{n}'/2^k)$, where **n** and **n**′ are the Qbit number operators associated with the two lines that cross at the dot, and k = 1, 2, or 3 depending on whether the dot lies in the first, second, or third horizontal row below the top row of Hadamard transformations.

In this form the inversion in order from most to least significant Qbits between the input and the output is shown by bending the Qbit lines, rather than by inverting the order in the state symbols.



The 2-Qbit gates **V** are also displayed in a way that does not suggest a nonexistent asymmetry between control and target Qbits.

The permutation operator **P** plays a crucial role in establishing the operator form of the inverse Fourier transform operator $\mathbf{U}^{\dagger}_{\mathrm{FT}}$.

Since the adjoint of a product is the product of the adjoints in the opposite order, and since Hadamards and **P** are self-adjoint, we have from (3.44)

$$\mathbf{U}_{FT} = \mathbf{H}_3(\mathbf{V}_{32}\mathbf{H}_2)(\mathbf{V}_{31}\mathbf{V}_{21}\mathbf{H}_1)(\mathbf{V}_{30}\mathbf{V}_{20}\mathbf{V}_{10}\mathbf{H}_0)\mathbf{P}.$$

$$\mathbf{U}^{\dagger}_{FT} = \mathbf{P}(\mathbf{H}_0\mathbf{V}^{\dagger}_{10}\mathbf{V}^{\dagger}_{20}\mathbf{V}^{\dagger}_{30})(\mathbf{H}_1\mathbf{V}^{\dagger}_{21}\mathbf{V}^{\dagger}_{31})(\mathbf{H}_2\mathbf{V}^{\dagger}_{32})\mathbf{H}_3. \qquad (3.45)$$

$$\mathbf{U}_{FT}^{\dagger} = \mathbf{P}(\mathbf{H}_0\mathbf{V}_{10}^{\dagger}\mathbf{V}_{20}^{\dagger}\mathbf{V}_{30}^{\dagger})(\mathbf{H}_1\mathbf{V}_{21}^{\dagger}\mathbf{V}_{31}^{\dagger})(\mathbf{H}_2\mathbf{V}_{32}^{\dagger})\mathbf{H}_3. \tag{3.45}$$

One can insert $\mathbf{1} = \mathbf{PP}$ on the extreme right of (3.45) and then note that the effect of sandwiching all the Hadamard's and 1-Qbit unitaries between two **P**'s is simply to alter all their indices by the permutation taking $0123 \rightarrow 3210$.

Therefore

$$\mathbf{U}_{FT}^{\dagger} = (\mathbf{H}_3\mathbf{V}_{23}^{\dagger}\mathbf{V}_{13}^{\dagger}\mathbf{V}_{03}^{\dagger})(\mathbf{H}_2\mathbf{V}_{12}^{\dagger}\mathbf{V}_{02}^{\dagger})(\mathbf{H}_1\mathbf{V}_{01}^{\dagger})\mathbf{H}_0\mathbf{P}. \tag{3.46}$$

If we now move every $\mathbf{V}^{\dagger}$ to the right past as many Hadamard's as we can, keeping in mind that each **V** commutes with all Hadamards except those sharing either of its indices, then we have

$$\mathbf{U}_{FT}^{\dagger} = (\mathbf{H}_3\mathbf{V}_{23}^{\dagger})(\mathbf{H}_2\mathbf{V}_{13}^{\dagger}\mathbf{V}_{12}^{\dagger})(\mathbf{H}_1\mathbf{V}_{03}^{\dagger}\mathbf{V}_{02}^{\dagger}\mathbf{V}_{01}^{\dagger})\mathbf{H}_0\mathbf{P}. \tag{3.47}$$

$$\mathbf{V}_{ij} = \exp\left(i\pi \mathbf{n}_i \mathbf{n}_j / 2^{|i-j|}\right),$$

Finally, if we note from (3.40) that each **V** is symmetric in its indices, and rearrange the parentheses in (3.47) to make the comparison with (3.44) easier, we have

$$\mathbf{U}_{FT}^{\dagger} = \mathbf{H}_3(\mathbf{V}_{32}^{\dagger}\mathbf{H}_2)(\mathbf{V}_{31}^{\dagger}\mathbf{V}_{21}^{\dagger}\mathbf{H}_1)(\mathbf{V}_{30}^{\dagger}\mathbf{V}_{20}^{\dagger}\mathbf{V}_{10}^{\dagger}\mathbf{H}_0)\mathbf{P}. \qquad (3.48)$$

This is precisely the form (3.44) of $\mathbf{U}_{FT}$ itself, except that each **V** is replaced by its adjoint, which (3.40) shows amounts to replacing each i by –i in the arguments of all the phase factors.

This is exactly what one does to invert the ordinary functional Fourier transform.

3.E. Getting the period from the quantum Fourier transform

As noted above, the spectacular increase in computational speed of the quantum Fourier transform over the classical fast Fourier transform is undermined by the usual proviso: having a specimen of a system described by the state (3.22) does not enable one to learn what the state is, and therefore there is no way to learn what **all** the Fourier-transformed amplitudes $\tilde{\gamma}$ are.

But if $\gamma(x)$ has a period r which is smaller than $2^n/2$, then if we have a small number of specimens of states of the form (3.22), even though we cannot learn the values of the $\tilde{\gamma}$, we can, with high probability, learn the value of the period r.

$$\mathbf{U}_{FT}\Big(\sum_{x=0}^{2^n-1}\gamma(x)|x\rangle\Big)=\sum_{x=0}^{2^n-1}\tilde{\gamma}(x)|x\rangle,$$

The period r of f appears in the state (3.19) of the input-register Qbits produced from a single application of $\mathbf{U}_f$.

$$|\Psi\rangle_n=\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}|x_0+kr\rangle_n. \qquad 9)$$

To get valuable information about r we apply the quantum Fourier transformation (3.21) to the input register:

$$\mathbf{U}_{FT}|x\rangle_n=\frac{1}{2^{n/2}}\sum_{y=0}^{2^n-1}e^{2\pi ixy/2^n}|y\rangle_n.$$

$$\mathbf{U}_{FT}\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}|x_0+kr\rangle=\frac{1}{2^{n/2}}\sum_{y=0}^{2^n-1}\frac{1}{\sqrt{m}}\sum_{k=0}^{m-1}e^{2\pi i(x_0+kr)y/2^n}|y\rangle=$$

$$\sum_{y=0}^{2^n-1}e^{2\pi ix_0y/2^n}\frac{1}{\sqrt{2^nm}}\Big(\sum_{k=0}^{m-1}e^{2\pi ikry/2^n}\Big)|y\rangle. \qquad (3.49)$$

If we now make a measurement, the probability p(y) of getting the result y is just the squared magnitude of the coefficient of $|y\rangle$ in (3.49).

$$\sum_{y=0}^{2^n-1} e^{2\pi i x_0 y/2^n} \frac{1}{\sqrt{2^n m}} \Big(\sum_{k=0}^{m-1} e^{2\pi i k r y/2^n} \Big) |y\rangle .$$

The factor $e^{2\pi i x_0 y/2^n}$, in which the formerly troublesome x_0 explicitly occurs, drops out of this probability and we are left with

$$p(y) = \frac{1}{2^n m} \Big| \sum_{k=0}^{m-1} e^{2\pi i k r y/2^n} \Big|^2 . \tag{3.50}$$

This completes the quantum mechanical part of the process, except that, as noted below, we may have to repeat the procedure a small number of times (of order 10 or so) to achieve a high probability of learning the period r .

To see why this is so, we require some more purely mathematical analysis, that includes the use of another branch of elementary number theory.

The probability (3.50) is a simple explicit function of y , whose magnitude has maxima when y is close to integral multiples of $2^n/r$.

$$p(y) = \frac{1}{2^n m}\left|\sum_{k=0}^{m-1} e^{2\pi i k r y/2^n}\right|^2.$$

Indeed we now show that the probability is at least 40% that the measured value of y will be as close as possible to — i.e. within 1/2 of — an integral multiple of $2^n/r$.

To see this we calculate a lower bound for p(y) when

$$y = y_j = j\,2^n/r + \delta_j, \tag{3.51}$$

with $|\delta_j| \le 1/2$.

Only the term in δ_j contributes to the exponentials in (3.50).

The summation is a *geometric series* which can be explicitly summed to give

$$p(y_j) = \frac{1}{2^n m} \frac{\sin^2(\pi \delta_j m r/2^n)}{\sin^2(\pi \delta_j r/2^n)}. \qquad (3.52)$$

Since (3.20) tells us that m is within an integer of $2^n/r$, and since $2^n/r \geq N^2/r > N$, we can with negligible error replace $mr/2^n$ by 1 in the numerator of (3.52), and replace the sine in the denominator by its (extremely small) argument.

This gives

$$p(y_j) = \frac{1}{2^n m} \left(\frac{\sin(\pi \delta_j)}{\pi \delta_j r/2^n} \right)^2 = \frac{1}{r} \left(\frac{\sin(\pi \delta_j)}{\pi \delta_j} \right)^2. \qquad (3.53)$$

When x is between 0 and $\pi/2$, the graph of sin x lies above the straight line connecting the origin to the maximum at $x = \pi/2$:

$$x/\tfrac{1}{2}\pi \leq \sin x, x \quad 0 \leq x \leq \pi/2. \qquad (3.54)$$

Since $\delta_j \leq 1/2$ we can therefore bound the probability (3.53) below by

$$p(y_j) \geq (4/\pi^2)/r. \qquad (3.55)$$

Since there are at least $r - 1$ different values of j, and since r is a large number one has at least a 40% chance ($4/\pi^2 = .4053$) of getting one of the special values (3.51) for y — a value that is within 1/2 of an integral multiple of $2^n/r$.

Note, in passing, that as $\delta_j \to 0$ in (3.53) the probability $p(y_j)$ becomes $1/r$, so that if all the δ_j are 0 — i.e. if the period r is exactly a power of 2 — then the probability of measuring an integral multiple of $2^n/r$ is essentially 1.

Indeed, you can easily check that in this (highly unlikely) case the probability remains 1 even if we do not double the number of Qbits in the input register and take $n = n_0$.

Thus the case $r = 2^j$ avoids many of the mathematical and practical (i.e. having to double the size of the input register) difficulties of quantum period finding.

Since r divides $(p-1)(q-1)$, all periods modulo pq will necessarily be powers of 2 if p and q are both primes of the form $2^n + 1$.

The smallest such primes are 3, 5, 17, and 257.

Hence claims to have realized the Shor algorithm for factoring 15 are to be taken with a grain of salt, as should possible future claims to have factored 51, 85, and 771.

$$p(y_j) \geq (4/\pi^2)/r.$$

Note also that the derivation of (3.55) only requires the argument of the sine in the denominator of (3.52) to be small.

This will be the case provided 2^n is any large multiple of N — i.e. provided the input register is large enough to contain many periods of $b^x (\mathrm{mod}\ N)$.

The stronger requirement that 2^n should be as large as N^2 — that the input register should actually be able to accommodate at least N full periods — emerges when we examine whether it is possible to learn r itself, given an integral multiple of $2^n/r$.

Suppose, then, that we have found a y that is within 1/2 of $j2^n/r$ for some integer j.

It follows that

$$\left| \frac{y}{2^n} - \frac{j}{r} \right| \leq \frac{1}{2^{n+1}}. \qquad (3.56)$$

Since y is the result of our measurement and we know n , the number of input-register Qbits, we have an estimate for the fraction j/r .

It is here that our use of an n-Qbit input register with $2^n > N^2$ is crucial.

By using twice as many Qbits as needed to represent all the integers up to N, we have ensured that our estimate (3.56) of j/r is off by no more than $1/2N^2$.

But since $r < N$, and since any two distinct fractions with denominators less than N must differ by at least $1/N^2$, the measured value of y and the fact that r is less than N is enough to determine a unique value of the rational number j/r.

That value of j/r can be efficiently extracted from the known value of $y/2^n$ by a simple application of the theory of continued fractions.

This gives us the fraction j/r reduced to lowest terms — i.e. it gives us integers j_0 and r_0 which have no common factors satisfying $j_0/r_0 = j/r$.

Since r_0 is r divided by the factors it has in common with j, if we were lucky enough to get a value of j that is coprime to r, then we have r.

Since, as noted above, two random numbers j and r have a better than even chance of having no common factors, we do not have to be terribly lucky.

We can easily check to see if r_0 is the sought for period by computing (with a classical computer) b^{r_0} modulo (N) and seeing whether or not it is b.

If it is not, we can try several low multiples $2r_0, 3r_0, 4r_0, \ldots$ since the probability of j sharing a large common factor with r is not large.

If this fails, we can repeat the whole procedure.

We now get j′/r where j′ is another (random) integer, yielding an r'_0 which is r divided by the factors it has in common with j′.

So r is the product of r_0 with the factors r has in common with j, and r is also the product of r'_0 with the factors r has in common with j'.

If j and j' have no factors in common — which again has a better than even chance of happening — then r will be the least common multiple of r_0 and r'_0.

We can again test to see if we have the right r by evaluating $b^r \pmod N$ to see if it is indeed equal to b.

If it is not, we can again try some of the lower multiples of our candidate for r and, if necessary, go through the whole business one more time to get yet another random multiple of $1/r$.

Because we are only 40% sure that our measurement gives us one of the y_j, we may have to repeat the whole procedure several (but not a great many) times before succeeding.

Some not terribly taxing mathematical detective work, carried out with the aid of a classical computer, will quickly give us the period r.

Class 14 - NOTE: I have completely modified the endgame of this class. We continue believing the mathematics and using it and then get help at the end from Peter Young notes!

3.F. Calculating the periodic function

We have assumed the existence of an efficient subroutine that calculates the quantity $b^x \pmod N$.

You might think that calculating $f(x) = b^x \pmod N$ for arbitrary values of x less than, say, $2^n = 10^{800}$ would require astronomical numbers of multiplications, but it does not.

We simply square $b \pmod N$, square the result (mod N), square that, etc., calculating the comparatively small number of powers $b^{2^j} \pmod N$ with $j < n$. .

The binary expansion of $x = x_{n-1}x_{n-2}\ldots x_1x_0$ tells us which of these must be multiplied together to get $b^x = \prod_j (b^{2^j})^{x_j}$.

So if we start with x in the input register, 1 (i.e. 000. . .001) in the output register, and b in an additional work register, then we can proceed as follows:

(a) multiply the ouput register by the work register if and only if $x_0 = 1$; (b) replace the contents of the work register by its modulo-N square; (a′) repeat (a) with the multiplication now conditional on $x_1 = 1$; (b′) repeat (b); (a″) repeat (a) with the multiplication now conditional on $x_2 = 1$; etc.

At the end of this process we will still have x in the input register (which serves only as a set of control bits for the n controlled multiplications), and we will have b^x mod (N) in the output register.

The work register will contain b^{2^n} independent of the value of x in the input register, and it will therefore be unentangled with the input and output registers and can be ignored when we take our starting point to be a superposition of classical inputs.

Note the striking difference between classical and quantum programming styles.

One's classical computational instincts would direct one to make a look-up table of all n modulo-N multiple squares of b , since (a) Cbits are cheap and stable and (b) otherwise to get b^x modulo (N) for all the needed values of x one would have to recalculate the successive squares so many times that this would become ridiculously inefficient.

$$\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n |f(x)\rangle_{n_0}$$

But the situation is quite the opposite with a quantum computer, since (a) Qbits are expensive and fragile and (b) "quantum parallelism" makes it possible to produce the state (3.18) with only a single execution of the procedure that does the successive squarings, thereby relieving us of any need to store all the modulo-N squares, at a substantial saving in Qbits.

As usual with quantum parallelism, there is the major catch that a measurement can reveal the value of only a single one of the modulo-N powers of b.

But, as we have now seen, by applying $\mathbf{U}_{FT}$ to the input register of the state (3.18) and only then making a measurement, one can get important collective information about the modulo-N values of b^x — in this case clues about the crucial period r — at the price of *losing* all information about the individual values.

3.G. The unimportance of unavoidable small phase errors(a slight digression)

To execute the quantum Fourier transform one needs 2-Qbit gates $\mathbf{V}_{ij} = e^{i\pi \mathbf{n}_i \mathbf{n}_j / 2^{|i-j|}}$ or, if one exploits the trick mentioned earlier, 1-Qbit gates $\mathbf{V}_j = e^{i\pi \mathbf{n}_j / 2^j}$.

Since we need to deal with numbers of many hundreds of digits, the 2^j appearing in these phase gates can be larger than 10^{100}.

Producing such tiny phase shifts requires a degree of control over the gates that is *impossible* to achieve.

Typically such phase-shift gates would allow two Qbits to interact in a carefully controlled way for an interval of time that was very precisely specified, but obviously not to hundreds of significant figures.

It is therefore crucial that the effectiveness of the period finding algorithm not be greatly affected by small (but not absurdly small) errors in the phase shifts.

On the face of it this seems worrisome.

Since we need to know the period r to a hundred or more digits of precision, don't we have to get the phase shifts right to a comparable accuracy?

Here the fundamentally *digital* character of the actual output of a quantum computation *saves* the day.

To learn r we require the outcomes of several hundreds of 1-Qbit measurements, *each* of which has just two outcomes (0 or 1).

While the action of the unitary gates that precede the measurements is like that of an analog computer, involving continuously variable phase shifts that cannot be controlled with perfect precision, this analog evolution *only* affects the probabilities of the sharply defined digital outputs.

Small alterations in the phases will produce small alterations in the probabilities of getting that extremely precise digital information, but *not* the precision of the information itself, once it is acquired.

Suppose that the phase of each term in the quantum Fourier transform (3.21) is incorrect by an amount $\varphi(x, y)$, and that each of these altered phases is bounded in magnitude by $\varphi \ll 1$.

The probability p(y) in (3.50) will be *changed* to

$$p_\varphi(y) = \frac{1}{2^n m} \left| \sum_{k=0}^{m-1} e^{2\pi i k r y / 2^n} e^{i\varphi_k(y)} \right|^2, \qquad (3.57)$$

where $\varphi_k(y) = \varphi(x_0 + kr, y)$.

Since all the phases $\varphi_k(y)$ are small compared with unity,

$$e^{i\varphi_k(y)} \approx 1 + i\varphi_k(y), \tag{3.58}$$

and therefore

$$p_\varphi(y) \approx \frac{1}{2^n m}\left|\sum_{k=0}^{m-1} e^{2\pi i k r y/2^n}\left(1 + i\varphi_k(y)\right)\right|^2. \tag{3.59}$$

What affect does this have on the probability of learning from the measurement one of the special values y_j give in (3.51)?

$$y = y_j = j\,2^n/r + \delta_j,$$

We have

$$p_\varphi(y_j) \approx \frac{1}{2^n m}\left|\sum_{k=0}^{m-1} e^{2\pi i k r \delta_j/2^n}\left(1 + i\varphi_{jk}\right)\right|^2, \tag{3.60}$$

where $\varphi_{jk} = \varphi_k(y_j)$.

If we expand to linear order in the small quantities φ_{jk} , we get

$$p_\varphi(y_j) \approx p(y_j) + \frac{2}{2^n m}\mathrm{Im}\Big[\Big(\sum_{k=0}^{m-1} e^{-2\pi i k r \delta_j/2^n}\varphi_{jk}\Big)\Big(\sum_{k'=0}^{m-1} e^{2\pi i k' r \delta_j/2^n}\Big)\Big]. \tag{3.61}$$

We can get an upper bound on the magnitude of the difference between the exact and approximate probabilities by replacing the imaginary part of the product of the two sums by the product of the absolute values of the sums, and then replacing each term in each sum by its absolute value.

Since the absolute value of each φ_{jk} is bounded by φ , we can conclude that

$$|p(y_j) - p_\varphi(y_j)| \le \frac{2m}{2^n}\varphi = \frac{2}{r}\varphi. \tag{3.62}$$

Since there are r different y_j , the probability of getting one of the special values y_j is altered by less than 2φ.

So if one is willing to settle for a probability of getting a special value that is at worst 1% less than the ideal value of about 0.4, one can tolerate phase errors up to $\varphi = 0.4/200 = 1/500$.

If one leaves out of the quantum Fourier transform circuit all controlled phase gates $e^{\pi i \mathbf{n}_i \mathbf{n}_j / 2^{|i-j|}}$ with $|i-j| > \ell$, the maximum phase error φ this can produce in any term is $\varphi = n\pi/2^{\ell}$, and therefore the probability will be within 1% of its ideal value if $1/2^{\ell} < 1/500n\pi$.

The number of n of Qbits in the input register might be as large as 3000 for problems of interest (factoring a 500 digit N).

Consequently for all practical purposes one can omit from the quantum Fourier transform all controlled-phase gates connecting Qbits that are more than about $\ell = 22$ wires apart in the circuit diagram.

This has two major advantages.

Of crucial importance, quantum engineers will not have to produce impossibly precise phase changes.

Furthermore the size of the circuit executing the quantum Fourier transform only has to grow linearly with large n rather than quadratically.

Since n is likely to be of order 10^3 for practical code-breaking, this too is a significant improvement

3.H. Period finding and factoring

Since Shor's period-finding quantum algorithm is always described as a factoring algorithm, I conclude this part by noting how period finding leads to factoring.

I consider only the case relevant to RSA encryption, where one wants to factor the product of two large primes, $N = pq$, although the connection between period finding and factoring is more general.

If we have a way to determine periods (such as Shor's algorithm) and want to find the large prime factors of $N = pq$, we pick a random number a coprime to N.

The odds that a random a happens to be a multiple of p or of q are minuscule when p and q are enormous, but if you are the worrying kind you can check that it isn't using the Euclidean algorithm.

In the overwhelmingly unlikely event that it is, then the Euclidean algorithm will give you p or q directly, and you will have factored N. Lucky you!!

Using our period finding routine, we find the order of a in G_{pq}: i..e., the smallest r for which

$$a^r \equiv 1 \pmod{pq}. \tag{3.63}$$

We can use this information to factor N provided our choice of a was lucky in two ways:

Suppose first that we are fortunate enough to get an r that is even.

We can then calculate

$$x = a^{r/2} \pmod{pq} \tag{3.64}$$

and note that

$$0 \equiv x^2 - 1 \equiv (x-1)(x+1) \pmod{pq}. \tag{3.65}$$

Now $x - 1 = a^{r/2} - 1$ is not congruent to 0 mod pq, since r is the *smallest* power of a congruent to 1.

Congruent: Of or relating to two numbers that have the same remainder when divided by a third number.
For example, 11 and 26 are congruent when the modulus is 5.

Suppose in addition — our second piece of good fortune — that

$$x + 1 = a^{r/2} + 1 \not\equiv 0 \pmod{pq}. \tag{3.66}$$

In that case neither $x - 1$ nor $x + 1$ is divisible by $N = pq$, but (3.65) tells us that their product is.

Since p and q are prime this is only possible if one of them, say p, divides $x - 1$ and the other, q, divides $x+1$.

Because the only divisors of N are p and q, it follows that p is the greatest common divisor of N and $x- 1$, while q is the greatest common divisor of N and $x + 1$.

We can therefore find p or q by a straightforward application of the Euclidean algorithm.

So it all comes down to the likelihood of our being lucky.

Mermin shows that the probability is at least 50% that a random number a in G_{pq} has an order r that is even with $a^{r/2} \not\equiv -1 \pmod{pq}$ so we do not have to repeat the procedure an enormous number of times to achieve a very high probability of success.

If you're willing to accept the fact that you don't have to try out very many random a's to succeed, then this elementary argument is all you need to know about why period-finding enables you to factor N= pq

But if you're curious about why the probability of good fortune is so high, then you must contend with reading the derivations, where Mermin has constructed an elementary but rather elaborate argument, by condensing a fairly large body of number-theoretic lore into the comparatively simple form it assumes when applied to the special case in which the number N is the product of two primes.

Math-3. More on Factoring - here we go……

We establish here that the probability is at least 1/2 that if a is a random member of G_{pq}, then the order r of a satisfies both

$$r \text{ even} \tag{3.67}$$

and

$$a^{r/2} \not\equiv -1 \pmod{pq}. \tag{3.68}$$

It has been shown that given such an a and its order r, the problem of factoring N = pq is trivially solved.

Note first that the order r of a in G_{pq} is the least common multiple of the orders r_p and r_q of a in G_p and in G_q.

That r must be some multiple of both r_p and r_q is immediate, since $a^r \equiv 1 \pmod{pq}$ implies that $a^r \equiv 1 \pmod{p}$ and $a^r \equiv 1 \pmod{q}$.

Furthermore any common multiple r′ of r_p and r_q satisfies $a^{r'} \equiv 1 \pmod{pq}$, because if $a^{r'} = 1 + mp$ and $a^{r'} = 1 + nq$, then mp = nq.

But since the primes p and q have no common factors this requires $m = kq$ and $n = kp$, and hence $a^r = 1 + kpq \equiv 1 \pmod{pq}$.

Since r is the least integer with $a^r \equiv 1 \pmod{pq}$, since r must be a common multiple of r_p and r_q, and since any such multiple works, r must be the least common multiple of r_p and r_q.

Consequently condition (3.67) can fail only if r_p and r_q are both odd.

Condition (3.68) can fail only if r_p and r_q are both odd multiples of the same power of 2; for if r_p contains a higher power of 2 than r_q, then since r is a common multiple of r_p and r_q, it will remain a multiple of r_q if a single factor of 2 is removed from it, and therefore that $a^{r/2} \equiv 1 \pmod{q}$.

But this is inconsistent with a failure of condition (3.68), which would imply that $a^{r/2} \equiv -1 \pmod{q}$.

So a necessary condition for failure to factor $N = pq$ is that either r_p and r_q are both odd, or they are both odd multiples of the same power of 2.

These neatly combine into a single necessary condition: Our effort to factor N can fail only if we have picked a random a for which r_p and r_q are both odd multiples of 2^j for any $j \geq 0$.

We next show that the probability p_f of this happening is less than 1/2.

To calculate a upper bound for the probability of failure p_f, note first that the mod-p and mod-q orders, r_p and r_q, of a are the same as the mod-p and mod-q orders of the numbers a_p and a_q in G_p and G_q, where

$$a \equiv a_p \pmod{p}, \qquad a \equiv a_q \pmod{q}. \tag{3.69}$$

Furthermore, every number a in G_{pq} is associated through (3.69) with a distinct pair from G_p and G_q.

For if $a_p = b_p$ and $a_q = b_q$ then $a - b$ is a multiple of both p and q, and therefore, since p and q are distinct primes, $a - b$ is a multiple of pq itself, so $a \equiv b \pmod{pq}$.

Since the $(p-1)(q-1)$ different members of G_{pq} are thus in one-to-one correspondence with the number of distinct pairs, one from the $p - 1$ members of G_p and one from the $q - 1$ members of G_q, the modulo-p and modulo-q orders r_p and r_q of a random integer a in G_{pq} will have exactly the same statistical distribution as the orders r_p and r_q of randomly and independently selected integers in G_p and G_q.

So to show that the probability of failure is at most 1/2, we must show that the probability is at most 1/2 that the orders r_p and r_q of such a randomly and independently selected pair are both odd multiples of 2^j for some $j \geq 0$.

We do this by showing that for any prime p, no more than half of the numbers in G_p can have orders r_p that are odd multiples of any given power of 2.

Given this, if $P_p(j)$ and $P_q(j)$ are the probabilities that random elements of G_p and G_q have orders that are odd multiples of 2^j, then the probability of failure p_f is less than

$$\sum_{j\geq 0} P_p(j)P_q(j) \leq \frac{1}{2}\sum_{j\geq 0} P_q(j) \leq \frac{1}{2}.]$$

This in turn follows from the following fact: Let the order $p-1$ of G_p be an odd multiple of 2^k for some $k \geq 0$; then exactly half the elements of G_p have orders that are odd multiples of 2^k.

This last fact follows from use the important theorem (proved in shortly) that if p is a prime, then G_p has at least one *primitive* element b of order $p-1$, whose successive powers therefore generate the entire group.

Given this theorem, we complete the argument by showing that the the orders of the odd powers of any such primitive b are odd multiples 2^k, but the orders of the even powers are not.

If r_0 is the order of b^j with j odd, then

$$1 \equiv (b^j)^{r_0} \equiv b^{jr_0} \pmod{p}, \qquad (3.70)$$

so jr_0 must be a multiple of $p-1$, the order of b.

Since j is odd r_0 must contain at least as many powers of 2 as does $p - 1$.

But since the order r_0 of any element must divide the order $p - 1$ of the group, r_0 cannot contain more powers of 2 than $p - 1$.

So r_0 is an odd multiple of 2^k.

On the other hand if j is even, then b^j satisfies

$$(b^j)^{(p-1)/2} = \left(b^{p-1}\right)^{j/2} \equiv 1 \pmod{p}, \tag{3.71}$$

so the order r_0 of b^j divides $(p - 1)/2$.

Therefore r_0 must contain at least one less power of 2 than does $p - 1$.

This completes the proof that the probability is at least 1/2 that a random choice of a in G_{pq} will satisfy both the conditions (3.67) and (3.68) that lead, with the aid of an efficient period-finding routine, to an easy factorization of $N = pq$.

The mathematical arguments certainly are beautiful!!

Illustration of Shor's Factoring Algorithm - all the steps……I need numbers!! skip to example

Shor's Factoring Algorithm

$$N = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k} = \prod_{i=1}^{k} p_i^{\alpha_i}$$

The classical part

for sufficiently large x

STEP 1

$x \leftarrow$ random integer such that $1 < x < N$

$d \leftarrow \gcd(x, N)$

STEP 2

$r \leftarrow O_N(x)$

The quantum part

SETUP

$|\psi_0\rangle_{t,n} \leftarrow |0\rangle_t \otimes |0\rangle_n$

STEP 2.1

$|\psi_1\rangle_{t,n} \leftarrow (\boldsymbol{H}^{\otimes t} \otimes \boldsymbol{I}^{\otimes n}) \left(|\psi_0\rangle_{t,n}\right)$

$$|\psi_1\rangle_{t,n} = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle_t \otimes |0\rangle_n$$

STEP 2.2

$$|\psi_2\rangle_{t,n} \leftarrow \boldsymbol{M}_{x,N}\left(|\psi_1\rangle_{t,n}\right)$$

$$\boldsymbol{M}_{x,N}\,|j\rangle_t \otimes |k\rangle_n \rightarrow |j\rangle \otimes \left|k + x^j \bmod N\right\rangle$$

$$\begin{aligned}
|\psi_2\rangle_{t,n} &= \boldsymbol{M}_{x,N}\left(|\psi_1\rangle_{t,n}\right) \\
&= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \boldsymbol{M}_{x,N}(|j\rangle_t \otimes |0\rangle_n) \\
&= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle_t \otimes \left|x^j \bmod N\right\rangle_n
\end{aligned}$$

$$|\psi_2\rangle_{t,n} = \frac{1}{\sqrt{2^t}} \sum_{b=0}^{r-1} \left[\left(\sum_{a=0}^{\frac{2^t}{r}-1} |ar+b\rangle_t \right) \otimes \left|x^b \bmod N\right\rangle_n \right]$$

STEP 2.3

$\tilde{\delta} \leftarrow$ measure the second register

$|\psi_3\rangle_t \leftarrow |\psi_2\rangle_{t,n}$ after measuring the second register

$$|\psi_3\rangle_t = \sqrt{\frac{r}{2^t}} \sum_{a=0}^{\frac{2^t}{r}-1} |ar + b_0\rangle_t$$

STEP 2.4

$$|\psi_4\rangle_t \leftarrow \boldsymbol{F}_n\left(|\Psi_3\rangle_t\right)$$

$$\boldsymbol{F}_n \left|j\right\rangle_n \rightarrow \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{-2\pi i j k/2^n} \left|k\right\rangle_n$$

$$\begin{aligned}
\left|\psi_4\right\rangle_t &= \sqrt{\frac{r}{2^t}} \sum_{a=0}^{\frac{2^t}{r}-1} \boldsymbol{F}_t \left|ar + b_0\right\rangle_t \\
&= \sqrt{\frac{r}{2^t}} \sum_{a=0}^{\frac{2^t}{r}-1} \left(\frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{-2\pi i j (ar+b_0)/2^t} \left|j\right\rangle_t \right) \\
&= \frac{1}{\sqrt{r}} \left(\sum_{j=0}^{2^t-1} \left[\frac{r}{2^t} \sum_{a=0}^{\frac{2^t}{r}-1} e^{-\frac{2\pi i j a}{2^t/r}} \right] e^{-2\pi i j b_0/2^t} \left|j\right\rangle_t \right) \\
&= \frac{1}{\sqrt{r}} \left(\sum_{k=0}^{r-1} e^{-2\pi i \frac{k}{r} b_0} \left| \frac{k 2^t}{r} \right\rangle_t \right)
\end{aligned}$$

$$\frac{1}{N} \sum_{j=0}^{N-1} e^{2\pi i j k/N} = \begin{cases} 1 & \text{if } N \mid K \\ 0 & \text{otherwise} \end{cases}$$

$$\left|\psi_4\right\rangle = \frac{1}{\sqrt{r}} \left(\sum_{k=0}^{r-1} e^{-2\pi i \frac{k}{r} b_0} \left| \frac{k 2^t}{r} \right\rangle \right) \left|x^{b_0}\right\rangle_n$$

STEP 2.5

$\tilde{\omega} \leftarrow$ measure $\left|\psi_4\right\rangle_t$

$$\tilde{\omega} = \frac{k_0 2^t}{r}$$

I thought nice job, but totally non-transparent!!!!!

Example: Factoring 217 ——————— use numbers but Mermin still leaves much to be desired!!

Find prime factors of 217 using Shor's Factoring Algorithm.

First, algorithm chooses random integer x such that $1 < x < 217$.

Suppose $x = 5$, which fulfills first condition: $\gcd(5, 217) = 1$.

Now is the time of finding multiplicative order of 5 modulo 217, which is achieved with help of a quantum computer.

We calculate t and n and initialize quantum system.

In this case, $n = \lceil \log_2 217 \rceil = 8$ and consequently $t = 2n = 16$.

Thus, our quantum registers have the following initial states:

$$|\psi_0\rangle_{16,8} \leftarrow |0\rangle_{16} \otimes |0\rangle_8$$

Next, apply Hadamard transformation, hence obtaining superposition of all basis states in first registe all with identical amplitudes.

This way, all integers between 0 and $2^{16} - 1$ are now **somewhere** in first register.

$$|\psi_1\rangle_{16,8} \leftarrow (\boldsymbol{H}^{\otimes 16} \otimes \boldsymbol{I}^{\otimes 8}) \left(|\psi_0\rangle_{16,8}\right)$$

$$|\psi_1\rangle_{16,8} = \frac{1}{\sqrt{2^{16}}} \sum_{j=0}^{2^{16}-1} |j\rangle_{16} \otimes |0\rangle_8$$

Apply quantum gate that calculates powers of 5 modulo 217, which has following effect

$$|\psi_2\rangle_{16,8} \leftarrow \boldsymbol{M}_{5,217}\left(|\psi_1\rangle_{16,8}\right)$$

$$|\psi_2\rangle_{16,8} = \frac{1}{\sqrt{2^{16}}} \sum_{j=0}^{2^{16}-1} \boldsymbol{M}_{5,217}\,|j\rangle_{16} \otimes |0\rangle_8 = \frac{1}{\sqrt{2^{16}}} \sum_{j=0}^{2^{16}-1} |j\rangle_{16} \otimes \left|5^j \bmod 217\right\rangle_8$$

Expand sum, can alternatively express result as follows (for simplicity, omitted subindices and tensor product operators):

$$\begin{aligned}|\psi_2\rangle = \frac{1}{\sqrt{2^{16}}} \big(&|0\rangle|1\rangle + |1\rangle|5\rangle + |2\rangle|25\rangle + |3\rangle|125\rangle + |4\rangle|191\rangle + |5\rangle|87\rangle + \\ &|6\rangle|1\rangle + |7\rangle|5\rangle + |8\rangle|25\rangle + |9\rangle|125\rangle + |10\rangle|191\rangle + |11\rangle|87\rangle + \\ &|12\rangle|1\rangle + |13\rangle|5\rangle + |14\rangle|25\rangle + |15\rangle|125\rangle + |16\rangle|191\rangle + |17\rangle|87\rangle + \\ &|18\rangle|1\rangle + |19\rangle|5\rangle + |20\rangle|25\rangle + |21\rangle|125\rangle + |22\rangle|191\rangle + |23\rangle|87\rangle + \\ &|24\rangle|1\rangle + |25\rangle|5\rangle + |26\rangle|25\rangle + |27\rangle|125\rangle + |28\rangle|191\rangle + |29\rangle|87\rangle + \\ &|30\rangle|1\rangle + \ldots.\big)\end{aligned}$$

After close inspection, can observe that values on second register are periodic. If make common factor, end up with state:

$$
\begin{aligned}
|\psi_2\rangle = \frac{1}{\sqrt{2^{16}}} \Big(&(|0\rangle + |6\rangle + |12\rangle + |18\rangle + \ldots + |65526\rangle + |65532\rangle)\,|1\rangle + \\
&(|1\rangle + |7\rangle + |13\rangle + |19\rangle + \ldots + |65527\rangle + |65533\rangle)\,|5\rangle + \\
&(|2\rangle + |8\rangle + |14\rangle + |20\rangle + \ldots + |65528\rangle + |65534\rangle)\,|25\rangle + \\
&(|3\rangle + |9\rangle + |15\rangle + |21\rangle + \ldots + |65529\rangle + |65535\rangle)\,|125\rangle + \\
&(|4\rangle + |10\rangle + |16\rangle + |22\rangle + \ldots + |65530\rangle)\,|191\rangle + \\
&(|5\rangle + |11\rangle + |17\rangle + |23\rangle + \ldots + |65531\rangle)\,|87\rangle \Big)
\end{aligned}
$$

Thanks to this representation, easier to understand what will happen if measure second register.

$\tilde{\delta} \leftarrow$ measure the second register

$|\psi_3\rangle_{16} \leftarrow |\psi_2\rangle_{16,8}$ after measuring the second register

Clear that obtain value $\tilde{\delta}$ such that $\tilde{\delta} = 5^{\tilde{j}} \bmod 217$ for a certain $\tilde{j} \in \{0, \ldots, 2^{16} - 1\}$.
Thus, $\tilde{\delta} \in \{1, 5, 25, 125, 191, 87\}$ (which are powers of 5 modulo 217). Suppose get $\tilde{\delta} = 25$
Register will collapse into $|25\rangle$ and all other possible values will be destroyed and gone forever, information about rest of possible powers of 5 modulo 217 lost. However, first register will also collapse into values that were tensored with $|25\rangle$, thus discarding remaining ones. What interests us is that all basis states tensored with $|25\rangle$ correspond with exponents $\tilde{j}$ such that $\tilde{\delta} = 5^{\tilde{j}} \bmod 217$
More specifically:

$$|\psi_3\rangle_{16} = \frac{1}{\sqrt{10923}} (|2\rangle_{16} + |8\rangle_{16} + |14\rangle_{16} + ... + |65528\rangle_{16} + |65534\rangle_{16})$$

$$= \frac{1}{\sqrt{10923}} \left(\sum_{a=0}^{10922} |6a + 2\rangle_{16} \right)$$

Pattern has arisen, as basis states on first register display some periodic behavior. Period naturally corresponds with multiplicative order of 5 modulo 217, which happens to be equal to 6.

Of course, this information is yet hidden to us, we are just using some knowledge of the problem for providing a mathematical explanation of the performance of the algorithm in this particular case.

On the other hand, constant 10923 is just a normalization of amplitudes after collapse of register, corresponding to total number of exponents that return 25 modulo 217 between 0 and $2^{16} - 1$.

As explained previously, if want to obtain period from within bowels of our quantum system, should use quantum Fourier transform:

$$|\psi_4\rangle_{16} \leftarrow \boldsymbol{F}_{16} (|\psi_3\rangle_{16})$$

$$|\psi_4\rangle_{16} = \boldsymbol{F}_{16} \left(\frac{1}{\sqrt{10923}} \sum_{a=0}^{10922} |6a + 2\rangle_{16} \right) = \frac{1}{\sqrt{10923}} \sum_{a=0}^{10922} \boldsymbol{F}_{16} (|6a + 2\rangle_{16})$$

$$= \frac{1}{\sqrt{10923}} \sum_{a=0}^{10922} \left(\frac{1}{\sqrt{2^{16}}} \sum_{k=0}^{2^{16}-1} e^{-2\pi i \frac{(6a+2)k}{2^{16}}} |k\rangle_{16} \right)$$

$$= \frac{1}{\sqrt{2^{16}}} \sum_{k=0}^{2^{16}-1} \left(\left[\frac{1}{\sqrt{10923}} \sum_{a=0}^{10922} e^{-2\pi i \frac{6ak}{2^{16}}} \right] e^{-2\pi i \frac{6ak}{2^{16}}} |k\rangle_{16} \right)$$

Thus, end up again with distribution of amplitudes defined by

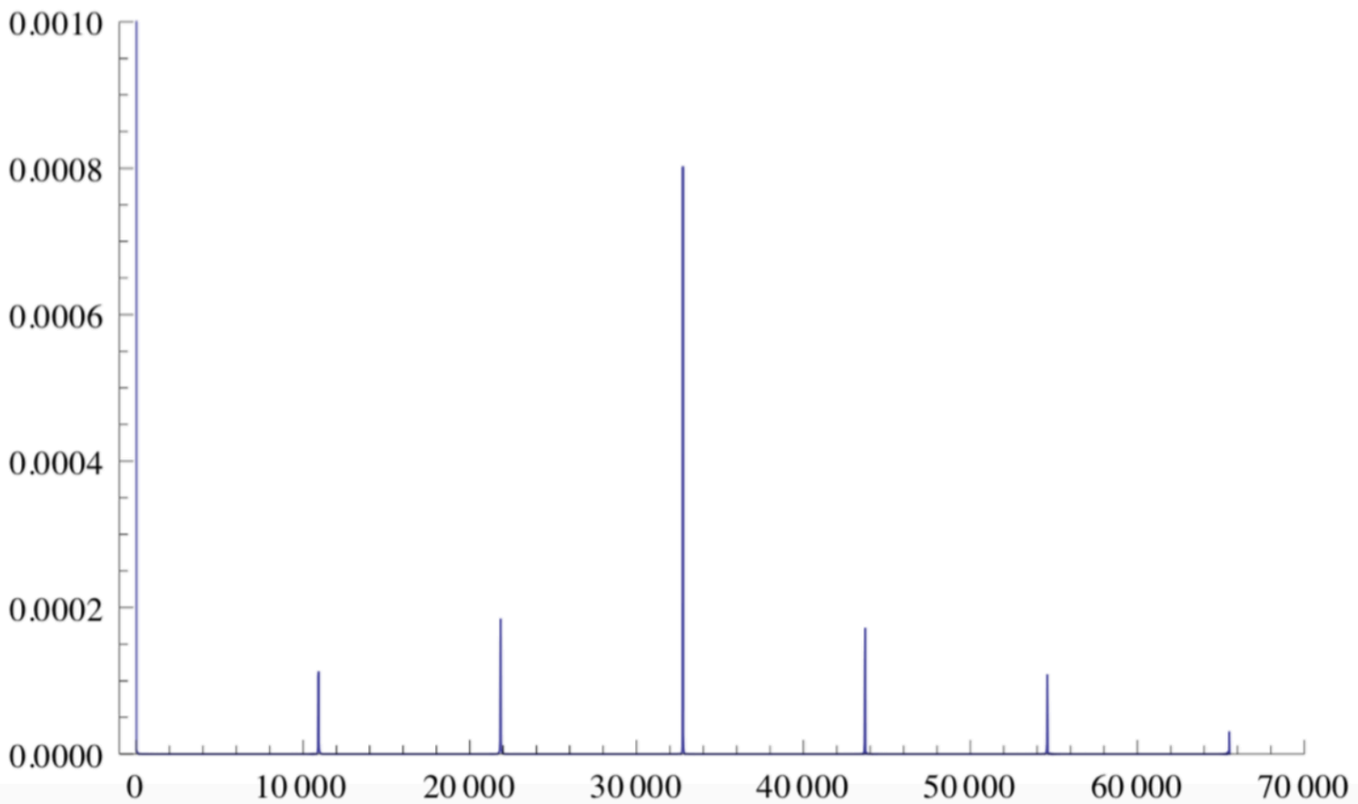
$$\alpha_k = \frac{1}{\sqrt{2^{16} \cdot 10923}} \left(\sum_{a=0}^{10922} e^{-2\pi i \frac{6ja}{2^{16}}} \right) e^{-2\pi i \frac{6jk}{2^{16}}}$$

which gives following probability distribution:

$$|\alpha_k|^2 = \frac{1}{2^{16} \cdot 10923} \left| \sum_{a=0}^{10922} e^{-2\pi i \frac{6ja}{2^{16}}} \right|^2$$

with k = 0, ..., $2^{16} - 1$.

If represent distribution in graph, easier to understand final steps of algorithm (each of peaks is composed of many numbers with probability greater than 0, not just of a single integer).



$\tilde{\omega} \leftarrow$ measure $|\psi_4\rangle_{16}$

Then measure first register, and obtain non-deterministically a value from one of seven peaks. First peak yields a 0, thus forcing us to start again the quantum part of algorithm. If, otherwise, obtain value from other six peaks, can retrieve (using continued fractions) the order r from it.

Explain last step more formally. Recall that a continued fraction is represented as

$$[a_0; a_1, \ldots., a_K] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\cdots + \cfrac{1}{a_K}}}}}$$

At this point we need help so I bring back Peter Young.

Continued Fractions - reminder from earlier(Peter Young)

Continued fractions are a convenient way of finding a simple rational approximation to a number.

The continued fraction representation of a number x is obtained as follows.

If there is an integer part of x call this c_0.

Subtract c_0 from x and call the inverse of the remainder x_1, so

$$x = c_0 + \frac{1}{x_1}. \tag{17.39}$$

Let the integer part of x_1 be c_1.

Subtract c_1 from x_1 and call the inverse of the remainder x_2 so $x_1 = c_1 + 1/x_2$.

Continuing in the same way for c_2 and x_3 etc. we get

$$x = c_0 + \cfrac{1}{c_1 + \cfrac{1}{x_2}} = c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{x_3}}} \cdots = c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2 + \cfrac{1}{c_3 + \cdots}}}. \tag{17.40}$$

To evaluate continued fractions we start at the bottom.

For example if we wish to evaluate

$$x = \cfrac{1}{2 + \cfrac{1}{5 + \cfrac{1}{4}}} \tag{17.41}$$

we determine first that

$$5 + \frac{1}{4} = \frac{21}{4} \tag{17.42}$$

and then that

$$2 + \frac{4}{21} = \frac{46}{21} \tag{17.43}$$

so

$$x = \frac{21}{46}. \tag{17.44}$$

If we stop after a certain number of iterations and ignore the remainder we have a “partial sum”, which is an approximation for x.

After each iteration the approximation improves.

If x is a rational number (ratio of two integers) the continued fraction will eventually terminate.

If x is irrational (like π) the continued fraction will go on for ever.

The first few continued fraction coefficients $c_i (i = 0,1,2\cdots)$ for $\pi = 3.141592654...$ are

$$3, 7, 15, 1, 292, 1, \cdots . \tag{17.45}$$

It is a property of continued fractions, which you can verify, that if a relatively large coefficient appears at some point, stopping the continued fraction at the previous coefficient gives an accurate approximation to the number.

For, example, omitting 15 and subsequent coefficients in Eq. (17.45) gives the well known approximation

$$\mathbf{3} + \frac{1}{\mathbf{7}} = \frac{22}{7} = 3.14286\ldots , \tag{17.46}$$

has an error of about 10^{-3} (the continued fraction coefficients are in bold).

A much more accurate result is obtained by omitting 292 and subsequent terms, which gives a value 355/113 = 3.141592920..., which has an error of a bit less than 3×10^{-7}.

This rational approximation to π was apparently first obtained by a Chinese mathematician Zu Chouygzhi about **1500** years ago.

In the present case we are interested in the continued fraction representation of $y/2^n$, which is a rational fraction so the continued fraction will eventually terminate.

As discussed in the earlier, the value of $y/2^n$ is close to m/r where r is no bigger than N (N can be represented by n_0 qubits with $n_0 = n/2$).

So we are interested in a *continued fraction approximation* to $y/2^n$ with a denominator no bigger than N.

Recall that $2^n = (2^{n_0})^2$ which is greater than N^2.)

Consider the example described earlier which has N = 91, a = 4 and n = 14 so $2^n = 16384$.

The most probable results for y are those in the column labeled “nearest integer” in Table 17.2.

Suppose the measurement of y gives the nearest integer for m = 5, i.e. 13653.

The continued fraction representation of 13653/16384 is obtained as follows:

$$
\begin{aligned}
x &= \frac{13653}{16384}, \\
c_0 &= [x] = 0, \qquad & x_1 &= (x - c_0)^{-1} &&= \frac{16384}{13653} \\
c_1 &= [x_1] = 1, \qquad & x_2 &= (x_1 - c_1)^{-1} &&= \frac{13653}{2731} \\
c_2 &= [x_2] = 4, \qquad & x_3 &= (x_2 - c_2)^{-1} &&= \frac{2731}{2729} \\
c_3 &= [x_3] = 1, \qquad & x_4 &= (x_3 - c_3)^{-1} &&= \frac{2729}{2} \\
c_4 &= [x_4] = 1364, & x_5 &= (x_4 - c_4)^{-1} &&= 2 \\
c_5 &= [x_5] = 2,
\end{aligned}
\tag{17.47}
$$

and the series terminates since x_5 is an integer.

Hence the exact continued fraction coefficients of 13653/16384 are

$$
0, 1, 4, 1, 1364, 2\,. \tag{17.48}
$$

Peter Young give a better example so let us relook at it …….

An example

The last part was probably dense, so we will try to clarify things by discussing a simple example.

Consider the following, which was also discussed ealrlier, N = 91, a = 4.

As shown there, the period is r = 6.

Since the period is not a power of 2 this is a general example.

One needs $n_0 = 7$ bits to represent N so we take $n = 2n_0 = 14$. Hence

$$\frac{2^n}{r} = 2730.67 \tag{17.29}$$

so

$$Q = 2730\,. \tag{17.30}$$

Hence there are 2730 (and two thirds) periods in our data.

As seen in the Mermin approach we need at least N(= 91) periods so 2730 is something of an overkill.

The peaks in the Fourier transform, which are at integers next to multiples of $2^n/r$ as discussed above, are shown in Table.

order (m)	peak position $(y_m = m\,2^n/r)$	nearest integer	P(nearest int.)
0	0	0	0.167
1	2730.67	2731	0.114
2	5461.33	5461	0.114
3	8192	8192	0.167
4	10922.67	10923	0.114
5	13653.33	13653	0.114

Table: The peak positions in the Fourier transform for the example discussed in this chapter. The output is at integer values of y and the nearest integers to the peaks are shown along with the probability at those nearest integer values, computed numerically. Neglecting the zeroth order peak at y= 0, which doesn't give useful information, the sum of the other probabilities at the nearest integers is 0.623, so we have a greater than 60% probability of obtaining the nearest integer to a non-zero multiple of 2n/r, from which one can deduce r using continued fractions, as discussed in the earlier and will discuss again later..

P(y) has been evaluated numerically and the results are shown in Figure.

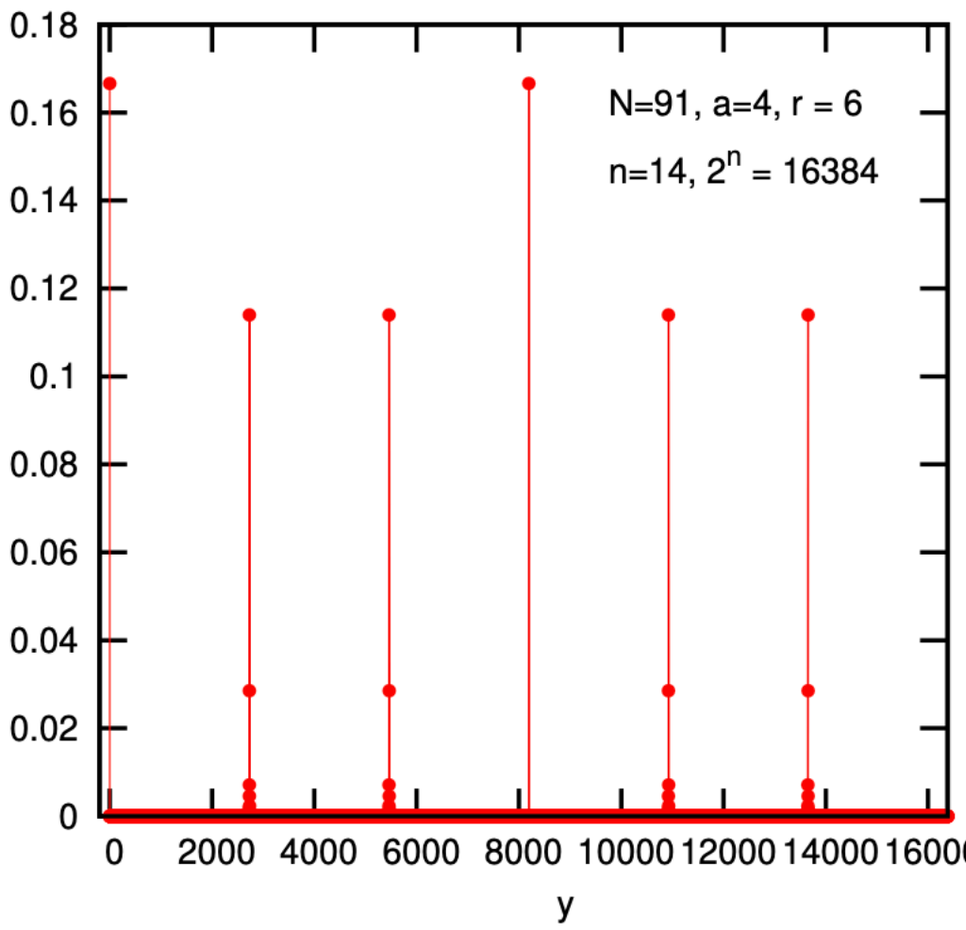


Figure: Probabilities for the different components of the Fourier transformed state for the example studied with N = 91, a = 4 for which the period is r = 6. These are computed numerically. There are six sharp peaks near $y_m = m2^n/r$, for m = 0, 1,⋯ , 5. The one at y = 0 (m = 0) doesn't give useful information. However, the probability of hitting the highest point of one of the other five peaks, i.e. the nearest integer to a non-zero multiple of $2^n/r$, is greater than 60%, see Table. If, as is likely, the measurement gives one of these results, it can then be used to determine the period r, as discussed earlier and will again be discussed shortly. A blowup of the m= 2 peak is shown in next Figure.

There are r = 6 peaks at values close to $y_m = m2^2/r$.

There is a trivial peak at exactly y = 0 (m= 0) but this can not give any useful information about the period r.

For the other 5 peaks, the peaks are not, in general, centered at exactly integer values, so the possible observed (integer) values of y are a set of discrete values around each peak, as shown in the histogram in next Figure which blows up the region around the m= 2 peak.

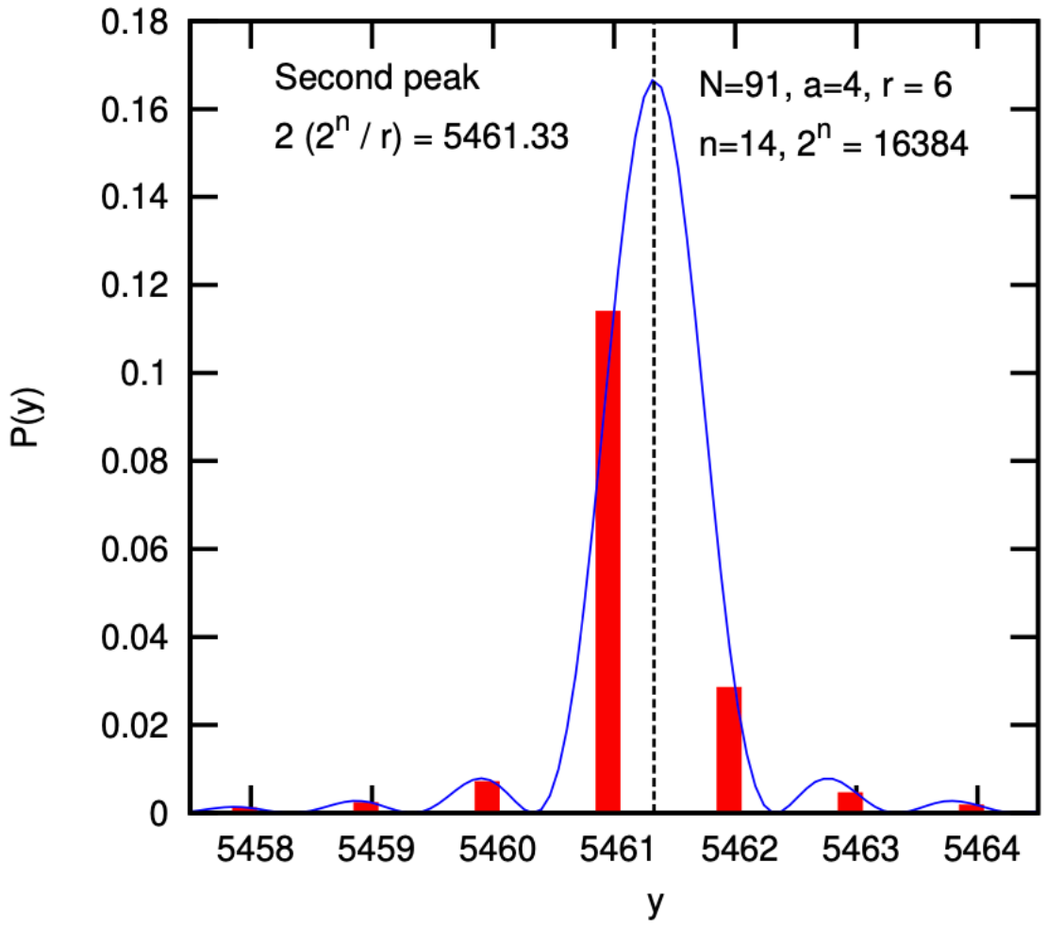


Figure: A blowup of the region around the m = 2 peak in last figure. The histogram is obtained by numerical evaluation. The probability is dominated by the biggest bar, which is at y = 5461 the nearest integer to $y_2 = 2 \times (2^n/r) = 5461.33$ (indicated by the vertical dashed line). The sum of the weights in the histogram is 1/r (= 1/6 here). The solid curve is the expression with y (= $y_m + \delta_m$) considered to be a continuous variable.

As discussed, the sum can be evaluated, and is given, to a good approximation in the region of the m-th peak, where one can determine y and y_m, which indicates the peak position.

Recall that y itself is an integer.

The function is plotted for continuous y as the solid curve.

When evaluated at integer y, it agrees very well with the values numerically computed which are shown as the histogram in the figure.

Note that δ_m

$$\delta_m = \epsilon + \ell \tag{17.31}$$

where ℓ is an integer and $|\epsilon| < 0.5$.

Note too that

$$\sum_{\ell=-\infty}^{\infty} \left(\frac{\sin(\pi(\epsilon+\ell))}{\pi(\epsilon+\ell)} \right)^2 = 1, \tag{17.32}$$

for arbitrary ϵ.

Hence, the weight around each of the peaks is equal to 1/r (= 1/6 here).

There are r peaks so the total probability is $r \times (1/r) = 1$ as required.

Referring to last figure, the weight in the largest bar is 0.114 which is 68% of 1/6, the total weight in all the bars for this (m = 2) peak.

One sees that if we calculate the probability of getting the nearest integer to an integral multiple of $2^n/r$, it is greater than 60%.

Let's suppose we get one of these.

In fact, lets suppose we get the large bar at y = 5461 in figure.

Recall that figure is a blowup of the m = 2 peak in previous figure.

Given the measured value, y = 5461, we will now see how to determine the period r using continued fractions.

We define $x = y/2^n$.

This is close to m/r, where r, the period, is what we want to determine.

Since r is no greater than N, the best guess for x is the partial sum having the largest denominator less than N.

As stated above we assume in this example that the measurement gave the value y = 5461, the highest histogram for the peak in the last figure..

We therefore determine the continued fraction representation for x = 5461/16384 (since n = 14 we have $2^n = 16384$).

Since this is a rational fraction the continued fraction terminates.

We use the methods I will derive shortly to determine coefficients as follows.

We have $c_0 = [x] = 0$ (note: $[\cdots]$ means the integer part of what is in the brackets).

We subtract c_0 from x and call the inverse of the remainder x_1, so $x_1 = 16384/5461$.

c_1 is the integer part of x_1 so $c_1 = 3$.

Subtract c_1 from x_1 and call the inverse of the remainder x_2.

Since $x_1 - c_1 = 1/5416$, we have $x_2 = 5461$.

Since this is an integer, the continued fraction terminates at this point.

Hence the coefficients are

$$c_0 = 0,\ c_1 = 3,\ c_2 = 5461, \tag{17.33}$$

and the corresponding partial sums are

$$\begin{aligned} c_0 &= 0, \\ c_0 + \frac{1}{c_1} &= \frac{1}{3}, \\ c_0 + \cfrac{1}{c_1 + \cfrac{1}{c_2}} &= \frac{5416}{16384}. \end{aligned} \tag{17.34}$$

The last result has a denominator bigger than N (= 91) so we neglect it and conclude that

$$\frac{m}{r} = \frac{1}{3}. \tag{17.35}$$

It is possible that m and r have a common factor, i.e. m = c, r = 3c for some integer c.

We try some small values for c.

Starting with c = 1, so r = 3, we compute a^3 (mod 91) and find that it is not 1.

However, we find that c = 2 does work, since $a^6 \equiv 1$ (mod 91).

Hence the period r is equal to 6, the desired result.

Summary

What is the operation count for Shor's period finding algorithm?

To factor an integer with n bits, the QFT requires, in principle, $O(n^2)$ operations.

Note, however, as discussed later, and also is shown in the Mermin approach, in practice one only needs of order $n\log_2 n$ gates to perform the QFT to within the necessary precision.

The computation of the function values using modular exponentiation takes $O(n^3)$ operations, but details of the calculation which state that the operation count is $O(n^2 \log n \log\log n)$, not much more than $O(n^2)$, if one uses a sophisticated method for multiplying two large numbers.

What about the continued fraction part, which is, of course, done on a classical computer?

Each division of an n-bit number takes of order n^2 operations if the division is done in a simple way.

In fact, division can be rewritten as several multiplications, so the operation count can be reduced to that for multiplication, i.e. O(n log n log log n).

The depth of the continued fraction where the denominator is O(N) is O(log N), since the coefficients in the continued fraction multiply to get the numerator and denominator.

This is O(n) since N contains no more than n/2 bits.

Hence the operation count for the continued fraction post-processing is $O(n^3)$, but recall that this is done on a classical computer.

Again the count is not much more than $O(n^2)$ if one uses a sophisticated method for dividing two large numbers.

Hence, the overall operation count of Shor's algorithm is $O(n^3)$.

Shor's algorithm for factoring integers therefore runs in polynomial time as a function of n, the number of bits in N.

For comparison, no polynomial time classical algorithm for factoring integers is known.

The fastest classical algorithm at present, the general number field sieve (GNFS), takes a time $\exp(\text{const.}\ n^{1/3} \log^{2/3} n)$.

It is currently not known whether there exists a yet to be determined polynomial time classical algorithm for factorization.

Even though the power of n in the exponent of the GNFS algorithm is less than one, it still much slower for large n than Shor's polynomial-time algorithm.

Hence, if the considerable technical difficulties could be overcome, and a quantum computer with a sufficiently large number of qubits built with the error rate made sufficiently low, then such a device could decode encrypted messages currently being sent down the internet which are currently impossible to decode on a classical computer.

Now to fill in some of the items I kept promising for later.

Math-4. G_p has an element of order p – 1 when p is prime.

It remains to prove the theorem that when p is prime, G_p contains at least one number of order p – 1.

The relevant property of the multiplicative group of integers {1, 2, 3, . . ., p – 1} modulo a prime is that together with 0 these integers also constitute a group under addition modulo p, which has all the structure necessary to ensure that a polynomial equation of degree d has at most d roots(proved by induction).

We can exploit this fact as follows:

Write the order s = p – 1 of G_p in terms of its prime factors q_i :

$$s = p - 1 = q_1^{n_1} \cdots q_m^{n_m}. \tag{3.72}$$

For each q_i, the polynomial $x^{s/q_i} - 1$ has at most s/q_i roots, and since $s/q_i < s$, the number of elements in G_p, there must be elements a_i in G_p satisfying

$$a_i^{s/q_i} \not\equiv 1 \pmod{p}. \tag{3.73}$$

Given such an a_i, define

$$b_i = a_i^{s/(q_i^{n_i})}. \tag{3.74}$$

We next show that the order of b_i is $q_i^{n_i}$. This is because

$$b_i^{(q_i^{n_i})} \equiv a_i^s \equiv 1 \pmod{p}, \tag{3.75}$$

so the order of order of b_i must divide $q_i^{n_i}$ and therefore be a power of q_i, since q_i is prime.

But if that order were any power of q_i less than n_i, then we would have $a_i^{s/q_i^k} \equiv 1 \pmod{p}$ with $k \geq 1$, which contradicts (3.73).

Because each b_i has order $q_i^{n_i}$, the product $b_1 b_2 \cdots b_m$ has order $q_1^{n_1} q_2^{n_2} \cdots q_m^{n_m} = p - 1$.

This follows from the fact that if two numbers in G_p have orders that are coprime, then the order of their product is the product of their orders.

Therefore since $q_1^{n_1}$ and $q_2^{n_2}$ are coprime, b_1b_2 has order $q_1^{n_1}q_2^{n_2}$.

But since $q_1^{n_1}q_2^{n_2}$ and $q_3^{n_3}$ are coprime, it follows that $b_1b_2b_3$ has order $q_1^{n_1}q_2^{n_2}q_3^{n_3}$.

Continuing in this way, we conclude that $b_1b_2 \cdots b_m$ has order $q_1^{n_1}q_2^{n_2} \cdots q_m^{n_m} = s = p - 1$.

End Class 14

Class 15

Part 4 - Searching with a Quantum Computer

Suppose you know that exactly one n-bit integer satisfies a certain condition, and suppose you have a black-boxed subroutine that acts on the $N = 2n$ different n-bit integers, outputting 1 ("true") if the integer is the special one and 0 ("false") otherwise.

In the absence of any other information, to find the special integer you can do no better with a classical computer than to apply repeatedly the subroutine to different random numbers until you hit on the special one.

If you apply it to M different integers the probability of your finding the special number is M/N.

You must test N/2 different integers to have a 50% chance of success.

If, however, you have a quantum computer with a subroutine that performs such a test, then for large (but not astronomically large) N, you can find the special integer with a probability that is very close to 1, by a method that requires you to call the subroutine only $(\pi/4)\sqrt{N}$ times.

This very general capability of quantum computers was discovered by Lov Grover, and goes under the name of Grover's search algorithm.

Shor's period-finding algorithm, Grover's search algorithm, along with their various modifications and extensions, constitute the two masterpieces of quantum computational software.

One can think of the black-boxed subroutine in various ways.

It could perform a mathematical calculation to determine whether the input integer is the special one.

Here is an example.

If an odd number p can be expressed as the sum of two squares, $m^2 + n^2$, then since one of m or n must be even and the other odd, p must be of the form 4k+ 1.

It is a fairly elementary theorem of number theory that if p is a prime number of the form 4k+1 then it can always be expressed as the sum of two squares in exactly one way.

Thus 5 = 4+1, 13 = 9+4, 17 = 16+1, 29 = 25 + 4, 37 = 36+1, 41 = 25+16, etc.

Given any such prime p, the simple-minded way to find the two squares is to take randomly selected integers x with $1 \leq x \leq N$, with N the largest integer less than $\sqrt{p/2}$, until you find the one for which $\sqrt{p - x^2}$ is an integer a.

If p is of the order of a trillion, then following the simple-minded procedure you would have to calculate $\sqrt{p - x^2}$ for nearly a million x to have a better than even chance of succeeding.

But using Grover's procedure with an appropriately programmed quantum computer you could succeed with a probability of success extremely close to 1 by calling the quantum subroutine that evaluated $\sqrt{p - x^2}$ fewer than a thousand times.

Mathematically well-informed friends tell me that this particular example admits of much more efficient ways to proceed than random testing, but the quantum algorithm to be described below enables even mathematical ignoramuses like me, equipped with a quantum computer, to do better than random testing by a factor of $1/\sqrt{N}$.

And Grover's algorithm will provide this speed-up even on problems that might stump the mathematically well-informed.

Alternatively, the black box could contain Qbits that have been loaded with a body of data — for example alphabetically ordered names and phone numbers — and one might be looking for the name that went with a particular phone number.

It is with this kind of application in mind that the Grover's neat trick has been called searching a database.

Using as precious a resource as Qbits, however, merely to store classical information would be insanely extravagant, given our current or currently forseeable ability to manufacture Qbits.

Finding a unique solution — or one of a small number of solutions (see Part 4.C below) — to a tough mathematical puzzle seems a more promising application.

A. The Grover iteration.

Grover's algorithm requires a quantum subroutine that indicates, when presented with any n-bit integer, whether or not that integer is the special one a being sought, returning this information as the value of a function *f*(x) that satisfies

$$f(x) = 0,\ x \neq a; \qquad f(x) = 1,\ x = a. \tag{4.1}$$

Grover discovered a completely general way to do significantly better than merely letting the subroutine operate on different numbers from a list of 2^n candidates, until it produced the output 1 — the best classical method.

The quantum computational speed-up relies on the usual implementation of the subroutine that calculates f, in the form of a unitary transformation $\mathbf{U}_f$ that acts on an n-Qbit input register that contains x and a 1-Qbit output register that is or is not flipped from 0 to 1, depending on whether x is or is not the number a having the special property:

$$\mathbf{U}_f\left(|x\rangle_n|y\rangle_1\right) = |x\rangle_n|y \oplus f(x)\rangle_1. \tag{4.2}$$

A simple circuit that has precisely this action is shown in Figure 4.1.

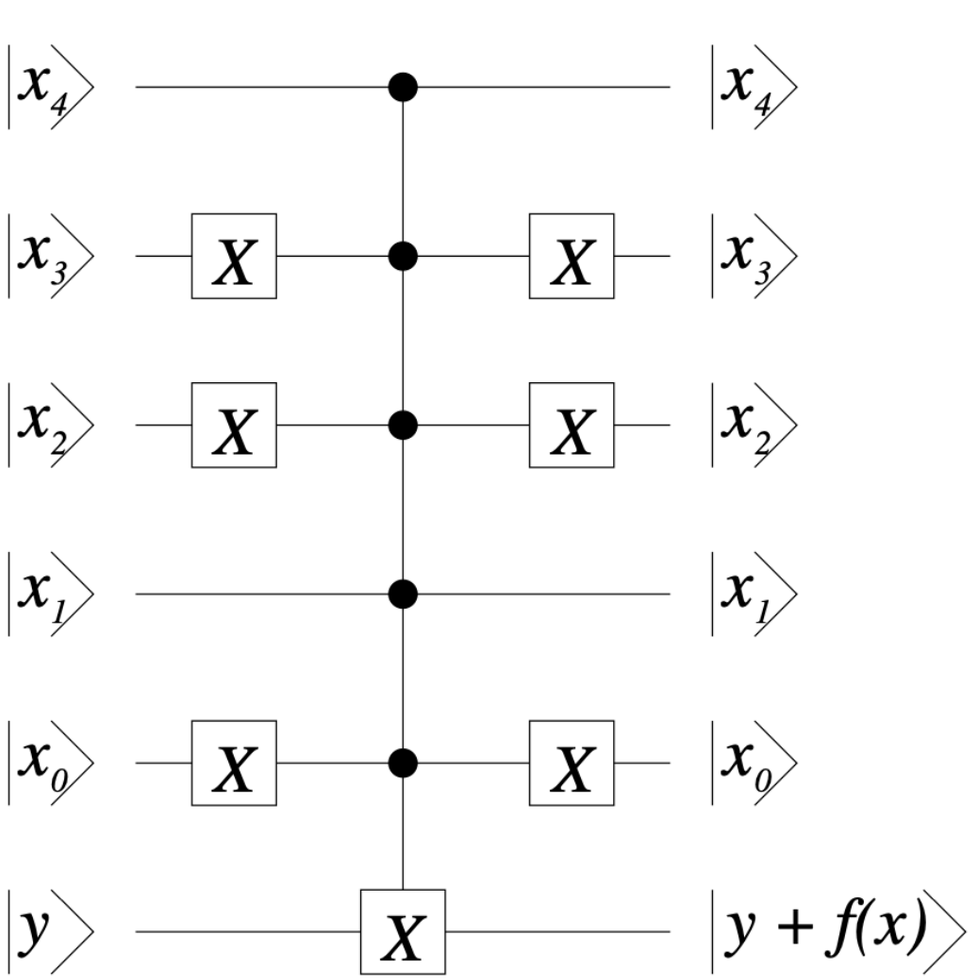


Figure 4.1. A possible realization of a black box that executes the unitary transformation $\mathbf{U}_f\,|x\rangle_n|y\rangle_1 = |x\rangle_n|y \oplus f(x)\rangle_1$, where $f(x) = 0$, $x \neq a$; $f(x) = 1$, $x = a$. The input register has n= 5 Qbits and the special number a is 10010. The 6-Qbit gate in the center of the figure is a five-fold controlled-NOT, which acts on the computational basis to flip the target bit if and only if every one of the five control bits is in the state $|1\rangle$. Ways to construct such a gate out of more elementary gates are shown in Figures 4.4-4.7(see later).

Figure 4.1

The figure can be viewed as providing a minimalist version of Grover's algorithm, reminiscent of the Bernstein-Vazirani problem (Part 2), though not susceptible to the special trick that worked in that simpler case.

We are given a black box containing a circuit of the type shown in Figure 4.1, but we are not told which Qbits are acted on by the **X** gates and which are not — information specified by the unknown n-bit integer a.

If there were n Qbits in the input register and the computer were classical, we could do no better than to try each of the 2^n possible inputs until we found the one for which the output register was flipped.

But using Grover's algorithm we can determine this information with probability quite close to 1, by invoking the subroutine fewer than $2^{n/2}$ times — more precisely $(\pi/4)2^{n/2}$) times — when N is large.

As in the Bernstein-Vazarani problem, it is useful to change the flip of the state of the output register into an overall sign change, by transforming the 1-Qbit output register into the state

$$\mathbf{H}|1\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) \tag{4.3}$$

prior to the application of $\mathbf{U}_f$.

The action of $\mathbf{U}_f$ is then to multiply the (n + 1)-Qbit state by −1 if and only if x = a:

$$\mathbf{U}_f\Big(|x\rangle \otimes \mathbf{H}|1\rangle\Big) = (-1)^{f(x)}|x\rangle \otimes \mathbf{H}|1\rangle. \tag{4.4}$$

In this form, the effect of $\mathbf{U}_f$ on the states $|x\rangle \otimes \mathbf{H}|1\rangle$, is exactly the same as doing nothing at all to the 1-Qbit output register, while acting on the n-Qbit input register with an n-Qbit unitary transformation **V** that acts on the computational basis as follows:

$$\mathbf{V}|x\rangle = (-1)^{f(x)}|x\rangle = \begin{cases} |x\rangle, & x \neq a, \\ -|a\rangle, & x = a. \end{cases} \tag{4.5}$$

We can write **V** as

$$\mathbf{V} = \mathbf{1} - 2|a\rangle\langle a|, \tag{4.6}$$

where |a⟩⟨a| is the projection operator(see Part 1.F) on the state |a⟩.

Since $\mathbf{U}_f$ is linear, so is **V**.

Acting on a general superposition |Ψ⟩ of computational basis states, **V** changes the sign of the component of the state along |a⟩, while leaving unchanged the component orthogonal to |a⟩:

$$\mathbf{V}|\Psi\rangle = |\Psi\rangle - 2|a\rangle\langle a|\Psi\rangle. \tag{4.7}$$

As we shall see, $\mathbf{U}_f$ is the only unitary transformation appearing in Grover's algorithm (where it appears repeatedly) that acts as anything other than the identity on the output register.

Because the output register starts in the state $\mathbf{H}|1\rangle$, unentangled with the input register, and because $\mathbf{U}_f$ maintains the output register in this state, the output register remains unentangled with the input register and in the state $\mathbf{H}|1\rangle$ throughout Grover's algorithm.

We could continue to describe things in terms of $\mathbf{U}_f$ and retain the 1-Qbit output register, expanding (4.7), for example, to the form

$$\mathbf{U}_f\Big(|\Psi\rangle \otimes \mathbf{H}|1\rangle\Big) = \big[|\Psi\rangle - 2|a\rangle\langle a|\Psi\rangle\big] \otimes \mathbf{H}|1\rangle. \qquad (4.8)$$

But it is simpler to suppress all explicit reference to the unaltered output register —always unentangled with the input register and always in the state $\mathbf{H}|1\rangle$— by replacing the (n+ 1)-Qbit unitary $\mathbf{U}_f$ by the n-Qbit unitary $\mathbf{V}$ that acts just on the n-Qbit input register, and defining all other operators that appear in the algorithm only by their action on the input register, with the implicit understanding that they act as the identity on the output register.

To execute Grover's algorithm, we once again initially transform the n-Qbit input register into the uniform superposition of all possible inputs:

$$|\phi\rangle = \mathbf{H}^{\otimes n}|0\rangle_n = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_n. \tag{4.9}$$

It is now useful to give this state a name of its own: $|\phi\rangle$.

In addition to **V**, Grover's algorithm requires a second n-Qbit unitary **W** that acts on the input register in a manner similar to **V**, but with a fixed form that does not depend on a.

The unitary transformation **W** preserves the component of any state along the standard state $|\phi\rangle$, while changing the sign of its component orthogonal to $|\phi\rangle$:

$$\mathbf{W} = 2|\phi\rangle\langle\phi| - \mathbf{1}, \tag{4.10}$$

where $|\phi\rangle\langle\phi|$ is the projection operator on the state $|\phi\rangle$.

We defer to section Part 4.B the not entirely obvious matter of how to build **W** out of 1-Qbit and 2-Qbit unitary gates.

Given implementations of **V** and **W**, Grover's algorithm is quite straightforward.

It consists of simply applying many times the product **WV** to the input register, taken initially to be in the state $|\phi\rangle$.

If you prefer you can think of applying the (n + 1)-Qbit transformations $\mathbf{U}_f = \mathbf{V} \otimes \mathbf{1}$ and $\mathbf{W} \otimes \mathbf{1}$ to the state $|\phi\rangle \otimes \mathbf{H}|1\rangle = \mathbf{H}^{\otimes(n+1)} (|0\rangle_n \otimes |1\rangle)$ of the combined input and output registers.

But since both transformations leave unchanged the 1-Qbit output register in the state $|1\rangle$, disentangled from the input register, it is simpler to restrict attention to the action of **V** and **W** on the input register alone.

To see what is accomplished by repeatedly applying **WV** to the initial state $|\phi\rangle$, note that both **V** and **W** acting on either $|\phi\rangle$ or $|a\rangle$ give linear combinations of these two states.

Since $\langle a|\phi\rangle = \langle\phi|a\rangle = 1/2^{n/2}$, independent of a, the linear combinations have real coefficients and are given by:

$$\mathbf{V}|a\rangle = -|a\rangle, \quad \mathbf{V}|\phi\rangle = |\phi\rangle - \tfrac{2}{2^{n/2}}|a\rangle;$$

$$\mathbf{W}|\phi\rangle = |\phi\rangle, \quad \mathbf{W}|a\rangle = \tfrac{2}{2^{n/2}}|\phi\rangle - |a\rangle. \tag{4.11}$$

So if we start with the state |ϕ⟩ and let any sequence of these two operators act successively, we will never be taken out of the two-dimensional plane spanned by real linear combinations of |ϕ⟩ and |a⟩.

Finding the result of repeated applications of **WV** to the initial state |ϕ⟩ reduces to an exercise in plane geometry.

It follows from the form (4.9) of |ϕ⟩, that |ϕ⟩ and |a⟩, considered as vectors in the plane of their real linear combinations, are very nearly perpendicular, since the cosine of the angle γ between them is given by

$$\cos\gamma = \langle a|\phi\rangle = 2^{-n/2} \leq 1/\sqrt{N}, \tag{4.12}$$

which is small when N is large.

It is convenient to define $|a_\perp\rangle$ to be the normalized real linear combination of $|\phi\rangle$ and $|a\rangle$ that is strictly orthogonal to $|a\rangle$ and makes the small angle $\theta = \pi/2 - \gamma$ with $|\phi\rangle$.

See Figures 4.2 and 4.3.

Figure 4.2

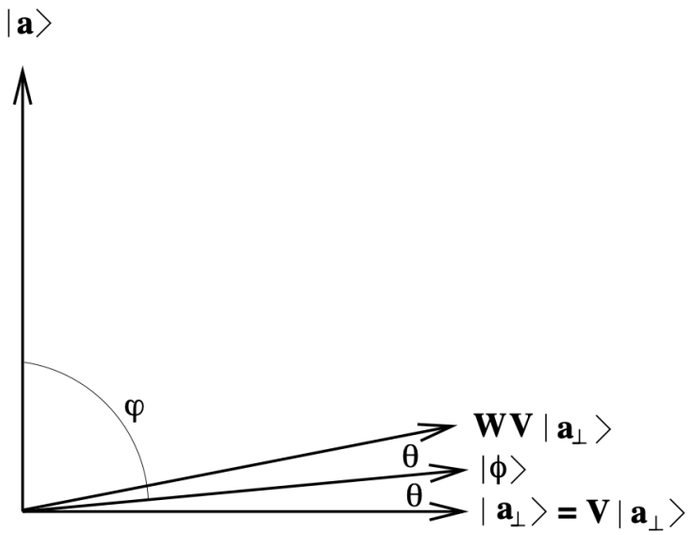


Figure 4.2. Real linear combinations of the special state $|a\rangle$, and the uniform superposition $|\phi\rangle = 2^{-n/2} \Sigma |x\rangle$, define a plane in which these two states are very nearly orthogonal. The state $|a_\perp\rangle$ in that plane, making a small angle θ with $|\phi\rangle$, is exactly orthogonal to $|a\rangle$. The unitary transformation **V** takes any vector in the plane into its reflection in the line through the origin along $|a_\perp\rangle$, so it leaves $|a_\perp\rangle$ invariant. The unitary transformation **W** takes any vector in the plane into its reflection in the line through the origin along $|\phi\rangle$, so it rotates $|a_\perp\rangle$ counterclockwise through the angle 2θ. Therefore the combined rotation **WV** rotates $|a_\perp\rangle$ counterclockwise through 2θ, and since **WV** is indeed a rotation it does the same to any vector in the plane.

Figure 4.3

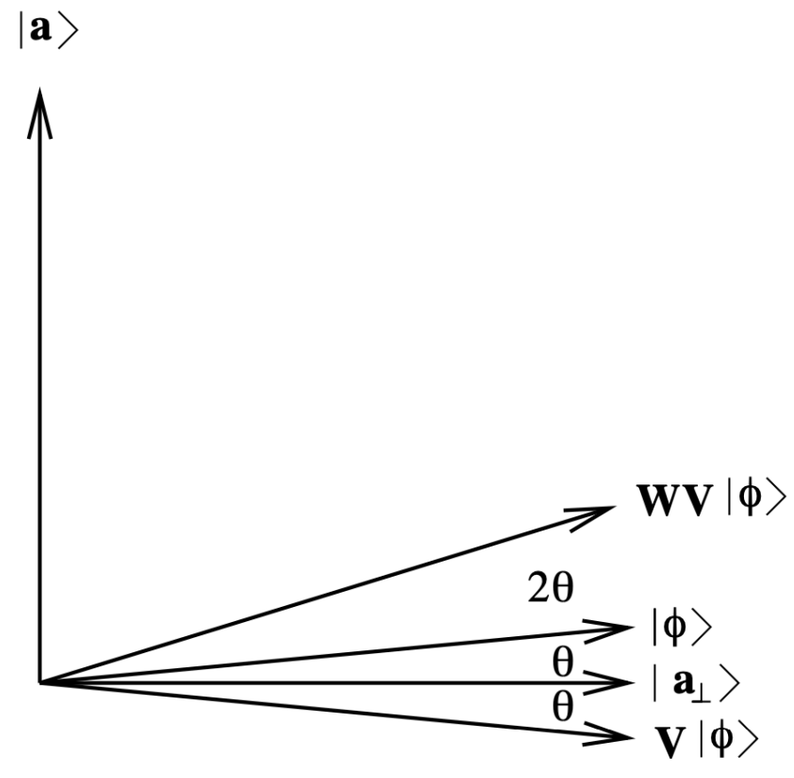


Figure 4.3. Since the rotation **WV** rotates any vector in the plane of real linear combinations of $|a\rangle$ and $|\phi\rangle$ clockwise through an angle 2θ, it takes $|\phi\rangle$ into a vector $\mathbf{WV}|\phi\rangle$ that makes an angle 3θ with $|a_\perp\rangle$. This can also be seen directly from the separate behaviors of **V** and **W**: **V** takes $|\phi\rangle$ into its mirror image in $|a_\perp\rangle$, and **W** then takes $\mathbf{V}|\phi\rangle$ into its mirror image in $|\phi\rangle$.

Since

$$\sin\theta = \cos\gamma = 2^{-n/2} \leq 1/\sqrt{N}, \tag{4.13}$$

θ is very accurately given by

$$\theta \approx 2^{-n/2} \tag{4.14}$$

when √N is large.

Since **W** leaves $|\phi\rangle$ invariant and reverses the direction of any vector orthogonal to $|\phi\rangle$, its geometrical action on any vector in the two-dimensional plane containing $|\phi\rangle$, $|a\rangle$, and $|a_\perp\rangle$ is simply to replace the vector by its reflection in the mirror line through the origin along $|\phi\rangle$.

On the other hand **V** reverses the direction of $|a\rangle$ while leaving any vector orthogonal to $|a\rangle$ invariant, so it acts on a general vector in the two-dimensional plane by replacing it with its reflection in the mirror line through the origin along $|a_\perp\rangle$.

The product **WV**, being a product of two two-dimensional reflections, is a two-dimensional rotation.

The angle of that rotation is most easily seen by considering the effect of **WV** on $|a_\perp\rangle$ (see Figure 4.2).

The application of **V** leaves $|a_\perp\rangle$ invariant, and the subsequent action of **W** on $|a_\perp\rangle$ reflects it in the line through the origin along the direction of $|\phi\rangle$.

So the net effect of the rotation **WV** on $|a_\perp\rangle$ is to rotate $|a_\perp\rangle$ past $|\phi\rangle$ through a total angle that is twice the angle θ between $|a_\perp\rangle$ and $|\phi\rangle$.

Because **WV** is a rotation, the result of applying it to any other vector in the plane is also to rotate that vector through the angle 2θ in the direction from $|a_\perp\rangle$ to $|\phi\rangle$.

So applying **WV** to the initial state $|\phi\rangle$ gives a vector rotated away from $|a\perp\rangle$ by 3θ, since $|\phi\rangle$ is already rotated away from $|a_\perp\rangle$ by θ (Figure 4.3).

Applying **WV** a second time results in a vector rotated away from $|a_\perp\rangle$ by 5θ, and each subsequent application of **WV** increases the angle between the final state and $|a_\perp\rangle$ by another 2θ.

Since θ is very close to $2^{-n/2}$, after an integral number of applications as close as possible to

$$(\pi/4)2^{n/2}, \tag{4.15}$$

the resulting state will be very nearly orthogonal to $|a_\perp\rangle$ in the plane spanned by $|\phi\rangle$ and $|a\rangle$— i.e. it will be very nearly equal to $|a\rangle$ itself.

Consequently a measurement of the input register in the computational basis will yield a with a probability very close to 1.

We can check to see whether we have been successful by "querying the oracle".

If $f(a)$ is 1, as it will be with very high probability, this confirms that we have found the desired a.

If we are very unlucky we might have to repeat the procedure a few more times before achieving success.

4.B. How to construct W

It remains to specify how to construct **W** out of 1-Qbit and 2-Qbit unitary gates.

Now **−W** works just as well as **W** for purposes of the search algorithm, since it leads to a final state that differs, if at all, only by a harmless overall minus sign.

It follows from (4.9) and (4.10) and the fact that $\mathbf{H}^{\otimes n}$ is its own inverse, that

$$-\mathbf{W} = \mathbf{1} - 2|\phi\rangle\langle\phi| = \mathbf{H}^{\otimes n}\big(\mathbf{1} - 2|00\ldots00\rangle\langle 00\ldots00|\big)\mathbf{H}^{\otimes n}, \qquad (4.16)$$

so we need a gate that acts as the identity on every computational basis state except $|00...00\rangle$, which it multiplies by -1.

This is just the action of an $(n-1)$-fold controlled-$\mathbf{Z}$ gate, with the roles of the 1-Qbit states $|0\rangle$ and $|1\rangle$ interchanged.

The interchange is accomplished by sandwiching the $(n-1)$-fold controlled-$\mathbf{Z}$ between $\mathbf{X}^{\otimes n}$ gates, and we therefore have

$$-\mathbf{W} = \mathbf{H}^{\otimes n}\mathbf{X}^{\otimes n}\left(\mathbf{c}^{n-1}\mathbf{Z}\right)\mathbf{X}^{\otimes n}\mathbf{H}^{\otimes n}. \tag{4.17}$$

We can construct W by constructing $\mathbf{c}^{n-1}\mathbf{Z}$, the $(n-1)$-fold controlled-$\mathbf{Z}$.

Figure 4.4 shows a straightforward but not terribly efficient way to make a $\mathbf{c}^{n-1}\mathbf{Z}$ gate for the case $n = 6$.

Figure 4.4

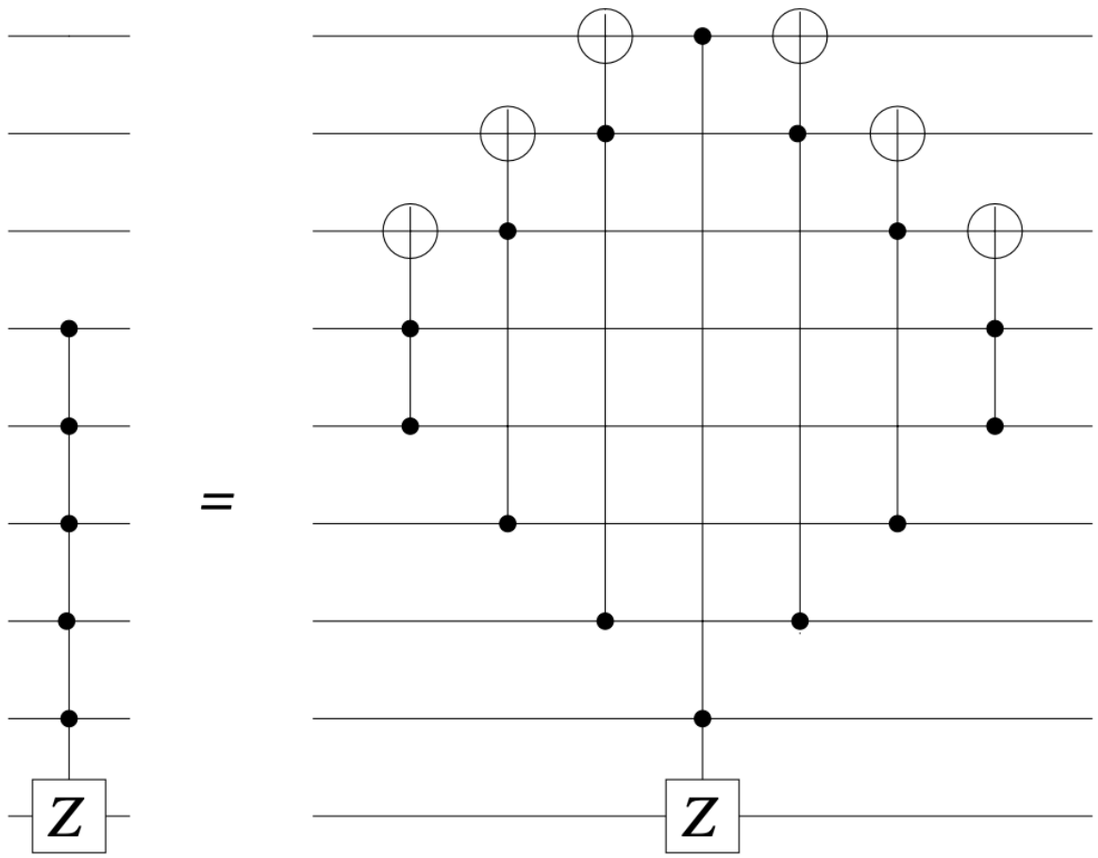


Figure 4.4. The n-fold controlled-**Z** transformation, $\mathbf{c}^n\mathbf{Z}$, acts as the identity on states of the computational basis unless all n control Qbits are in the state $|1\rangle$, when it acts on the target Qbit as **Z**. It can be constructed if one has an additional n− 2 ancilliary Qbits, all initially in the state $|0\rangle$. One uses 2(n− 2) $\mathbf{c}^2\mathbf{X}$ (Toffoli) gates and a $\mathbf{c}^2\mathbf{Z}$ gate. The construction is illustrated for the case n= 5. The top 3 wires are the 3 ancilliary Qbits. The next 5 wires from the top are the 5 control Qbits, and the bottom wire is the target Qbit. One easily verifies (applying the circuit to computational basis states, with each of the ancilliary Qbits in the state $|0\rangle$) that **Z** acts on the target Qbit if and only if every one of 5 control Qbits is in the state $|1\rangle$. The Toffoli gates are symmetrically disposed on both sides of the diagram to ensure that at the end of the process each of the 3 ancilliary Qbits is set back to its initial state $|0\rangle$. This is essential if the ancilliary Qbits are not to become entangled with the Qbits on which the Grover iteration acts, represented by the bottom 6 wires.

We use n− 3 ancillary Qbits, all initially in the state $|0\rangle$, 2(n− 3) $\mathbf{c^2X}$ (Toffoli) gates, and one $\mathbf{c^2Z}$ gate.

As noted in Partr 2 (See Figures 2.11 and 2.12), these can all be built out of 1-Qbit and 2-Qbit gates.

It is essential for the success of the algorithm that each ancillary Qbit be restored to its initial state $|0\rangle$, since our analysis of the Grover algorithm assumes that the input and output registers have states of their own, unentangled with any other Qbits after each application of **W** and **V**.

The construction of Figure 4.4 is rather expensive in Qbits, requiring n− 3 ancillas to apply the algorithm to an n-bit set of possibilities for the special number a.

At a cost of four times as many Toffoli gates, one can reduce the number of ancillas to a single one, regardless of the size of n.

The way to do this is developed in Figures 4.5-4.7. Figures 4.5 and 4.6 show how nearly doubling the number of gates makes it possible for the construction of Figure 4.4 to work for arbitrary initial states of the ancillas.

Figure 4.7 then shows how by an additional doubling one can, with the aid of a single ancilla, divide up an n-fold controlled-**Z** into two multiply controlled NOT gates and two multiply controlled-Z gates, each acting on about n/2 Qbits.

Since **X= HZH**, one can convert a multiply controlled-**Z** gate to a multiply controlled-NOT gate by applying Hadamard gates to the target Qbit at the beginning and end of the circuit.

The controlled-**Z** gates are able nondisruptively to use the control Qbits of the controlled-NOT gates as their ancillary Qbits in the construction of Figure 4.5, and the controlled-NOT gates can make similar use of the control Qbits of the controlled-**Z** gates.

Figure 4.5

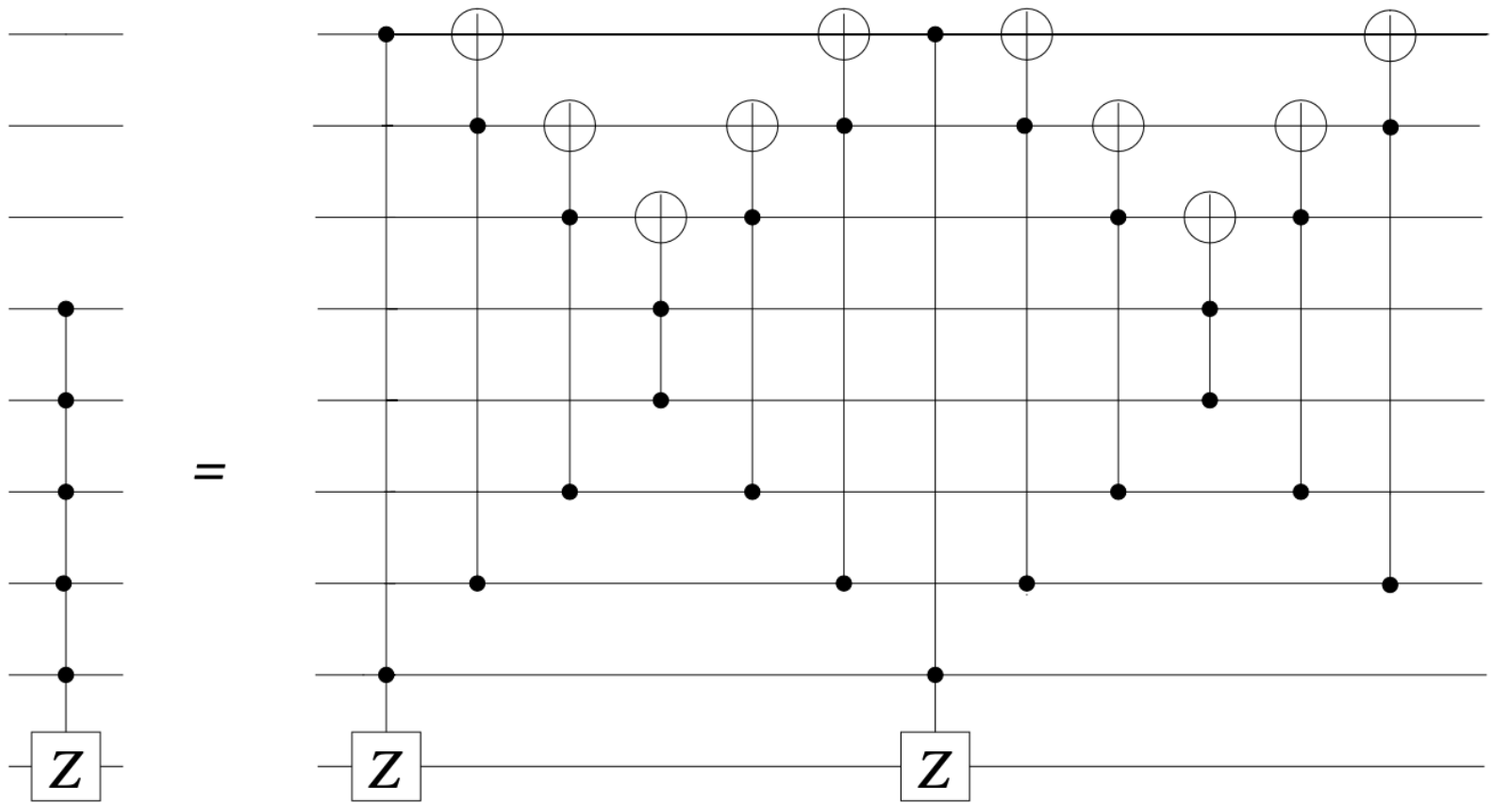
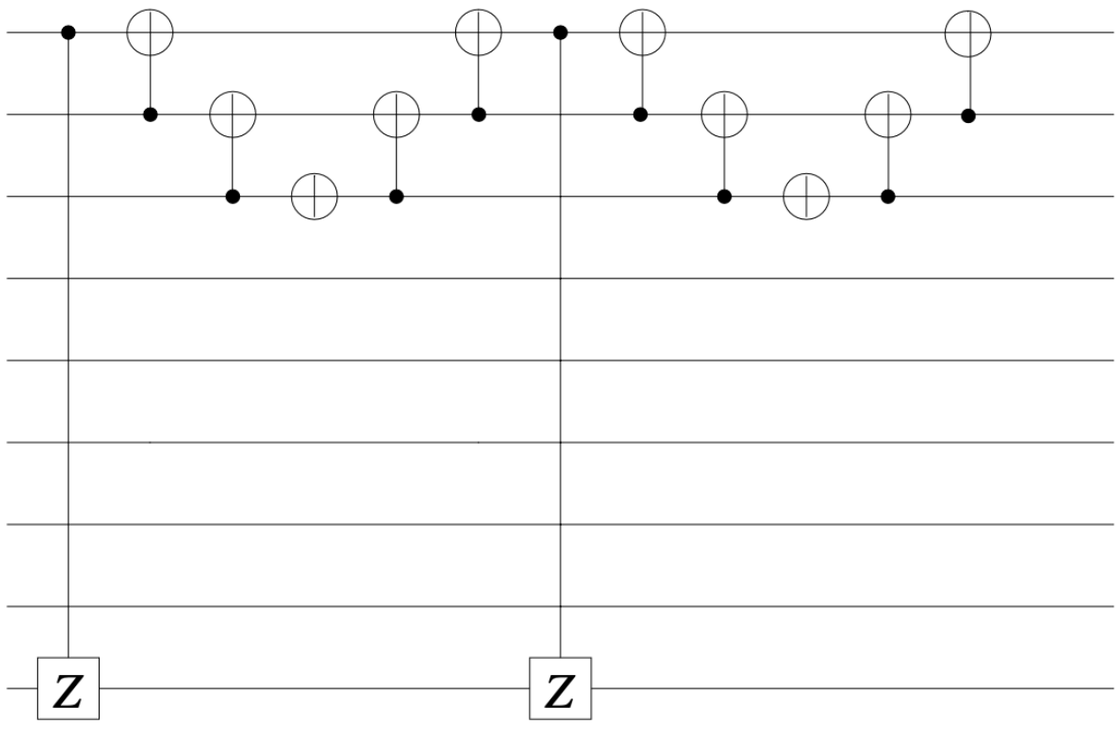
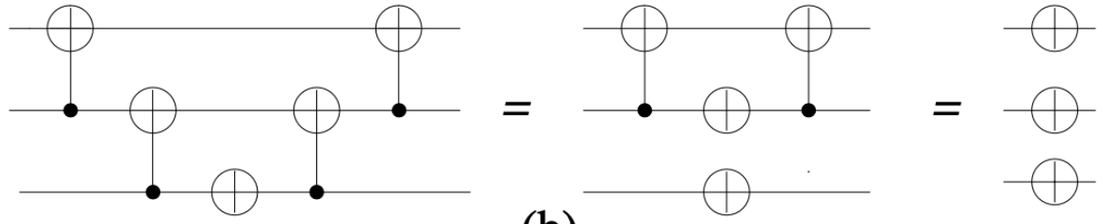


Figure 4.5. An improved version of Figure 4.4, with twice as many gates. Gates have been added on the left and right to ensure that the circuit works for *arbitrary* initial computational basis states of the 3 ancillary Qbits at the top, restoring them to their initial states at the end of the computation. To see this note, first that because Toffoli gates or $\mathbf{c^2Z}$ gates are their own inverses, the circuit acts as the identity on those computational basis states of all 9 Qbits in which any one of the 5 control Qbits (2nd through 6th wires from the bottom) is in the state $|0\rangle$, regardless of the computational-basis states of the other Qbits. This is because, as an examination of the figure reveals, replacing the gate governed by any one of the 5 control Qbits with the identity always results in a pairwise cancellation of all the remaining gates. It remains only to confirm that when all 5 control Qbits are in the state $|1\rangle$, the circuit acts as **Z** on the target Qbit at the bottom, and the state of the three ancillary Qbits at the top is unchanged. This is established in Figure 4.6, which shows the operation of the gates in Figure 4.5 when the 5 control Qbits are all in the state $|1\rangle$. Note that because **X= HZH** one can use this circuit to produce a multiply controlled-NOT gate by applying Hadamard gates to the bottom wire on the far right and left

Figure 4.6



(a)



(b)

Figure 4.6. Part (a) reproduces what remains of Figure 4.5 when all five control Qbits are in the state $|1\rangle$. One easily verifies that two identical cNOT gates, separated by a NOT acting on their control Qbit, have exactly the same action on the computational basis as a pair of NOT gates that act on both the control and target Qbit. As a result each of the two identical sets of 5 adjacent gates acting on the three ancillary Qbits at the top of part (a) (Since **X= HZH** Figure 4.5 works for either type.) reduces simply to three NOT gates, as shown in part (b). Making this further simplification in part (a), note that because each of the three ancillary Qbits is acted on by two NOT gates, its state is unaltered. The two NOT gates acting on the upper wire also ensure that precisely one of the two **cZ** gates applies **Z** to the bottom Qbit, independent of the state of the upper wire.

Figure 4.7

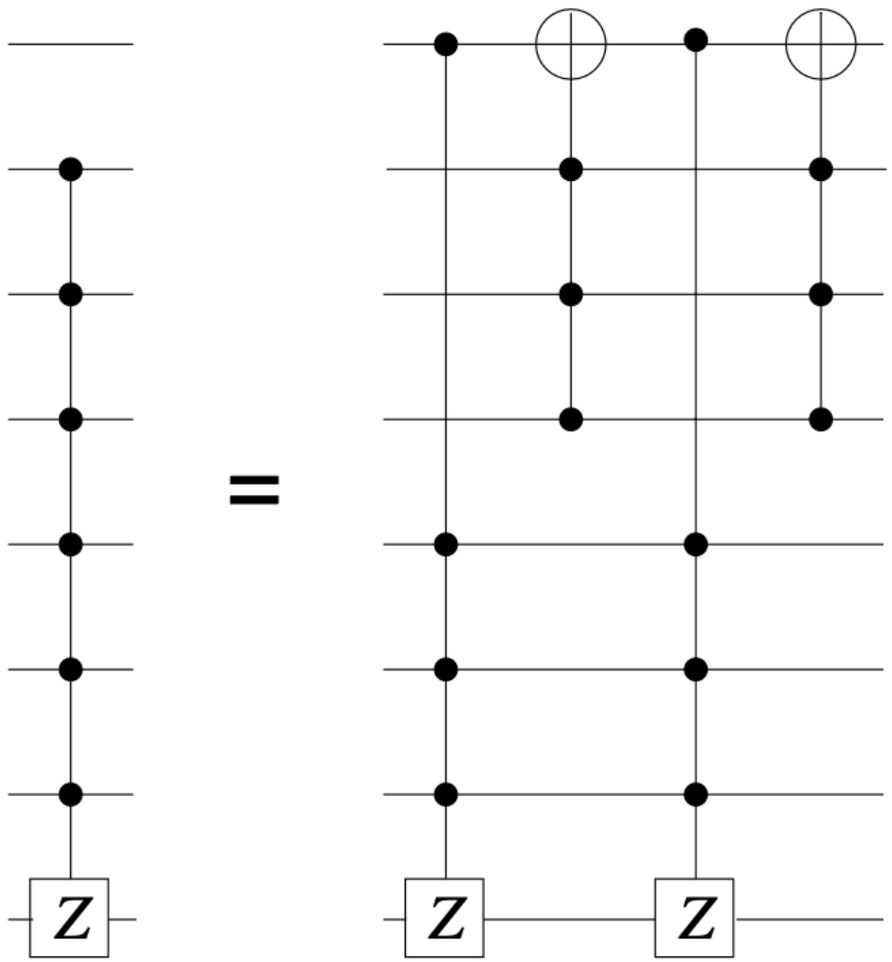


Figure 4.7. The identity illustrated by the circuit is easily confirmed. There is only one ancilla, whose state is left unchanged. By introducing circuits of the form in Figure 4.5 into this circuit one can produce $\mathbf{c^nZ}$ or $\mathbf{c^nX}$ gates with the aid of just a single ancilla. The point is that in constructing each of the multiply controlled-NOT gates in Figure 4.7 out of Toffoli gates, as prescribed in Figure 4.5, one can borrow the control Qbits of the multiply controlled-**Z** gates to use as ancillary Qbits in the expansions of Figure 4.5, since those expansions work regardless the state of their ancillary Qbits, and restore it to its original form. For the same reasons one can also borrow the control Qbits of the multiply controlled-NOT gates to construct the multiply controlled-**Z** gates.

4.C. Generalization to several special numbers.

If there are several special numbers, essentially the same algorithm can be used to find one of them, provided we know how many there are.

The function f in (4.1) now becomes

$$f(x) = 0,\ x \neq a_1, \ldots a_m; \qquad f(x) = 1,\ x = a_1, \ldots a_m. \tag{4.18}$$

The n-Qbit unitary transformation **V** extracted from (4.4) becomes one whose action on computational basis states in the input register is given by

$$\mathbf{V}|x\rangle = |x\rangle,\ x \neq a_1, \ldots a_m; \qquad \mathbf{V}|x\rangle = -|x\rangle,\ x = a_1, \ldots a_m. \tag{4.19}$$

If we replace the state |a⟩ by

$$|\psi\rangle = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} |a_i\rangle,$$

then starting with |ϕ⟩, which continues to have the form (4.9), the transformations **V** and **W** now keep the state of the input register in the two-dimensional plane spanned by the real linear combinations of |ψ⟩ and |ϕ⟩.

The unitary transformation **V** changes the sign of |ψ⟩ but preserves the linear combination of |ϕ⟩ and |ψ⟩ orthogonal to |ψ⟩, so **V** is now a reflection in the line through the origin along the vector $|\psi_\perp\rangle$ perpendicular to |ψ⟩ in the plane.

Everything else is just as in the case of a single special number except that now the angle Θ between $|\psi_\perp\rangle$ and |ϕ⟩ satisfies

$$\sin\Theta = \cos(\pi/2 - \Theta) = \langle\psi|\phi\rangle = \sqrt{m/2^n}. \qquad (4.20)$$

When $m/2^n << 1$, we can arrive at a state very close to $|\psi\rangle$ with

$$(\pi/4)2^{n/2}/\sqrt{m} \tag{4.21}$$

applications of **WV**.

A measurement then gives us, with a probability very close to 1, a random one of the special values a_i.

Note that the mean number of invocations of the subroutine only decreases as $1/\sqrt{m}$ with the number m of marked items, in contrast to a classical search, where doubling the number of acceptable solutions would halve the time of the search.

When $m/2^n$ is not small we have to reexamine the expression (4.21) for the optimal number of iterations, but at that point the quantum search offers little significant advantage over a classical one.

We must know how many special numbers there are for the procedure to work, since we have to know how many times to do the Grover iteration before making our measurement.

By exploiting the fact that the Grover iteration is periodic, restoring the initial state after about $\pi 2^n/\sqrt{m}$ iterations, it is possible to combine Grover iterations with a clever application of the quantum Fourier transform to learn the value of m with enough accuracy to enable one then to apply the Grover iteration the right number of times to ensure a high probability of success.

Part 5 - Quantum Error Correction

Correcting errors might sound like a dreary practical problem, of little aesthetic or conceptual interest.

But aside from being of crucial importance for the feasibility of quantum computation, it is also one of the most beautiful and surprising parts of the subject.

The surprise is that error correction is possible at all, since the only way to detect errors is to make measurements.

But measurement gates disruptively alter the states of the measured Qbits, apparently making things even worse.

“Quantum error correction” would seem to be an oxymoron.

The beauty, which I hope will emerge below, lies in the elegant and ingenious ways that people have found to get around this apparently insuperable obstacle.

The discovery in 1995 of quantum error correction by Peter Shor and, independently, Andrew Steane, had an enormous impact on the prospects for actual quantum computation.

It changed the dream of building a quantum computer, capable of useful computation, from a clearly unattainable vision, to a program that poses an enormous but not necessarily insuperable technological challenge.

Error correction is simply not an issue in classical computation.

In a classical computer the physical systems that embody individual bits — the Cbits — are immense on the atomic scale.

The two states of a Cbit representing 0 and 1 are so grossly different that the probability is infinitesimal for flipping from one to the other as a result of thermal fluctuations, mechanical vibrations, or other irrelevant extraneous interactions.

Error correction does become an issue, even classically, in the transmission of information over large distances, because the farther the signal travels, the more it attenuates.

One can deal with this in a variety of straightforward or ingenious ways.

One of the crudest is to encode each logical bit in three actual bits, replacing $|0\rangle$ and $|1\rangle$ with the codewords

$$|\overline{0}\rangle = |0\rangle|0\rangle|0\rangle = |000\rangle, \quad |\overline{1}\rangle = |1\rangle|1\rangle|1\rangle = |111\rangle. \tag{5.1}$$

One can then monitor each codeword, checking for flips in any of the individual Cbits and restoring them by the principle of majority rule, whenever a flip is detected.

Monitoring has to take place often enough to make negligible the probability that more than a single bit flips in a single codeword between inspections.

Quantum error correction also uses multi-Qbit codewords and it requires monitoring at a rate that renders certain kinds of errors highly improbable.

But there are several ways in which error correction in a quantum computer is quite different:

(a) In a quantum computer, unlike a classical computer, error correction is essential. The physical Qbits are individual atomic-scale physical systems such as atoms, photons, trapped ions, nuclear magnetic moments, etc. Any coupling whatever to any other degrees of freedom, not under the explicit control of the computer and its program, can substantially disrupt the state associated with such Qbits, entangling them with computationally irrelevant degrees of freedom, thereby destroying the computation. For a quantum computer to work without error correction each Qbit would have to be impossibly well isolated both from computationally irrelevant interactions with other parts of the computer and from interactions with anything else in its environment.

(b) In contrast to classical error correction, checking for errors in a quantum computer is problematic. The obvious way to monitor a Qbit is to measure it. But the result of measuring a Qbit is to alter its state, if it has one of its own, and, more generally, to destroy its quantum correlations with other Qbits with which it might be entangled. Such disruptions are stochastic — i.e. unpredictable — and introduce major errors of their own. One must turn to less obvious forms of monitoring.

(c) Bit flips are not the only errors. There are entirely nonclassical sources of trouble. For example phase errors, such as the alteration of $|0\rangle + |1\rangle$ to $|0\rangle - |1\rangle$, can be just as damaging.

(d) Unlike the discrete all-or-nothing bit-flip errors suffered by classical bits, errors in the quantum state grow continuously out of the uncorrupted state.

We begin our discussion of error correction by examining in Part 5.A a simple model of quantum error correction that works when the possible errors are artificially limited to a few specific kinds of disruption.

While this is clearly unrealistic, the error-correction procedure is easy to follow.

It also introduces in a simple setting most of the tricks that continue to work in the more realistic case.

5.A. A simplified example of quantum error correction.

Much of the flavor of quantum error correction is conveyed by an artificially simple model in which the only errors a collection of Qbits are allowed to experience are the classically meaningful errors: random flips of individual Qbits.

We shall examine the more general possibilities for quantum errors in Part 5.B below.

Bit-flip errors in a computation can be modeled by a circuit that differs from the ideal error-free circuit only in the occasional presence of extraneous 1-Qbit NOT gates.

If such randomly occurring error-producing NOT gates are suffciently rare, then since the only allowed errors are bit-flip errors, one might hope to be able to correct the corruptions they introduce by tripling the number of Qbits and using precisely the 3-Qbit code (5.1) that corrects for bit-flip errors in the classical case.

Because of the disruptive effect of measurement gates in diagnosing errors, it is not at all clear that such a 3-Qbit code can be effective for bit-flip errors in the quantum case.

Nevertheless it can be made to work, though the way in which one does the encoding and performs the error-correction is much more subtle for Qbits than it is for Cbits.

To begin with, there is the question of encoding.

Classically one merely replaces each of the two computational basis states $|x\rangle$ with the codeword states $|x\rangle = |x\rangle|x\rangle|x\rangle$, for x = 0 or 1.

Quantum bits, however, can also be in superpositions $\alpha|0\rangle + \beta|1\rangle$, and one requires a circuit that automatically encodes this into $\alpha|0\rangle + \beta|1\rangle = \alpha|0\rangle|0\rangle|0\rangle + \beta|1\rangle|1\rangle|1\rangle$ for arbitrary $\alpha$and β, in the absence of any knowledge of what the values of $\alpha$and β might be.

This can be with two cNOT gates that target two additional Qbits initially both in the state $|0\rangle$, as illustrated in Fig. 5.1:

$$\alpha|\overline{0}\rangle + \beta|\overline{1}\rangle = \alpha|0\rangle|0\rangle|0\rangle + \beta|1\rangle|1\rangle|1\rangle = \mathbf{C}_{21}\mathbf{C}_{20}\big(\alpha|0\rangle + \beta|1\rangle\big)|0\rangle|0\rangle. \qquad (5.2)$$

Figure 5.1

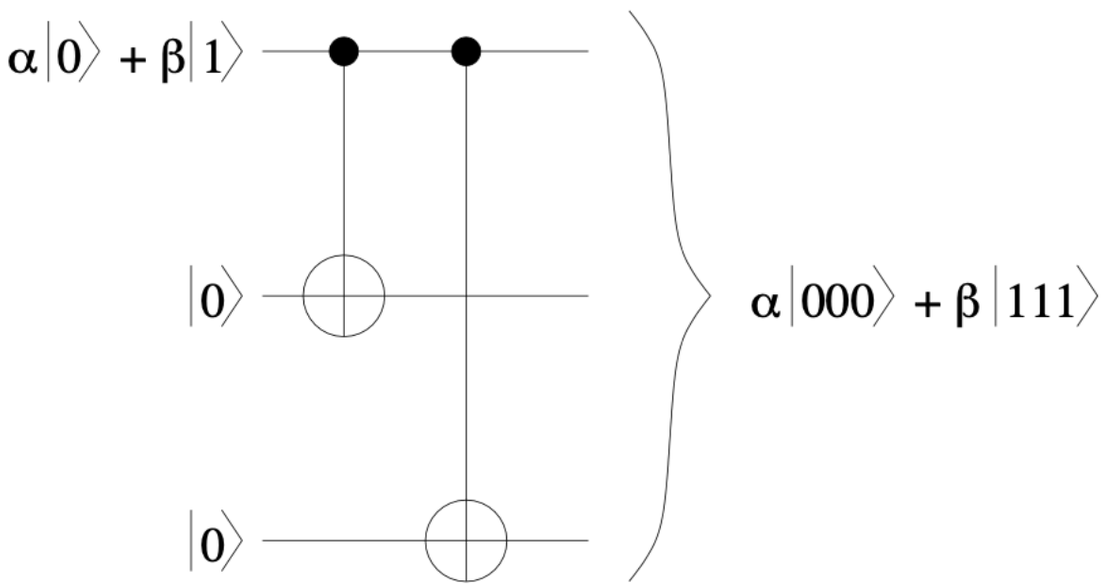


Figure 5.1. A unitary circuit that encodes the 1-Qbit state $\alpha|0\rangle + \beta|1\rangle$ into the 3-Qbit code state $\alpha|0\rangle|0\rangle|0\rangle + \beta|1\rangle|1\rangle|1\rangle$,using two cNOT gates and two other Qbits each initially in the state $|0\rangle$. The circuit clearly works for the computational basis states $|0\rangle$ and $|1\rangle$, and therefore, by linearity, it works for arbitrary superpositions.

Having produced such a 3-Qbit codeword state, we must then guard against its corruption by the possible action of an extraneous NOT gate that acts on at most one of the three Qbits, as illustrated in Figure 5.2.

Figure 5.2

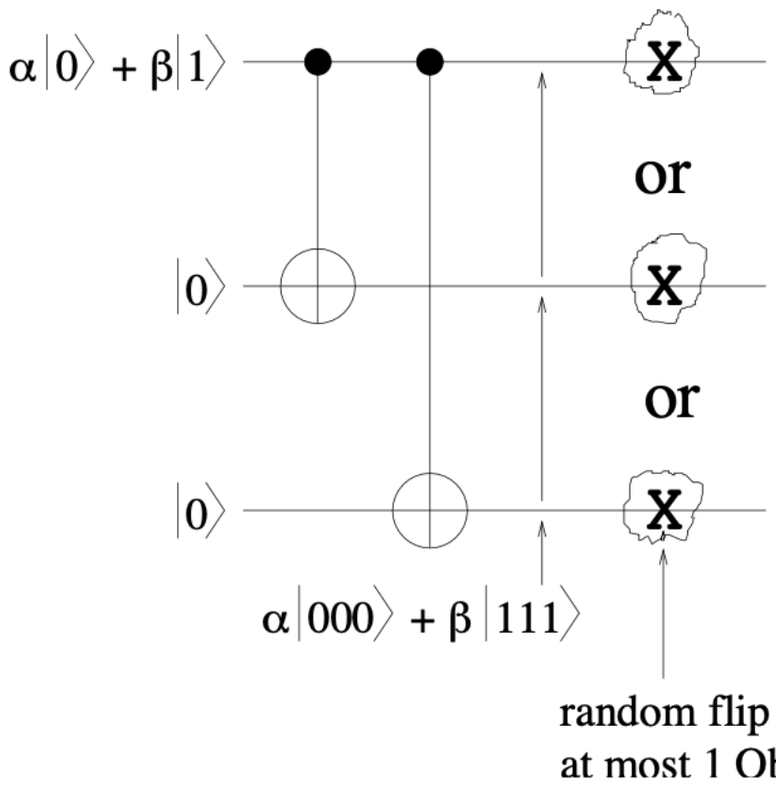


Figure 5.2. The encoded state of Figure 5.1 may or may not be corrupted by the action of a single extraneous NOT gate. The error inducing gates are depicted in a lighter font — **X** instead of **X** — and inside a noisy looking corrupted box.

This is easily done for Cbits, where there are only two possible uncorrupted initial states: $|000\rangle$ and $|111\rangle$, and examining them is unproblematic.

To see if either initial state has been corrupted by the action of a single NOT gate, one nondisruptively reads the three Cbits.

If this reveals all three Cbits to be in the same state, there is no corruption to correct.

If one of them is found to be in a different state from the other two, that particular Cbit is the one that was acted upon by the extraneous NOT gate, and applying a second NOT gate to that same Cbit restores the initial state.

In the case of Qbits, however, one cannot nondisruptively "read" their state.

The only way to extract information from a set of Qbits is by the action of measurement gates, but measuring one or more of the three Qbits immediately destroys the uncorrupted superposition

$$|\Psi\rangle = \alpha|000\rangle + \beta|111\rangle, \tag{5.3}$$

converting it either to $|000\rangle$ (with probability $|\alpha|^2$) or to $|111\rangle$ (with probability $|\beta|2$).

There is a similar coherence-destroying effect on each of the three possible corrupted states,

$$|\Psi_0\rangle = \mathbf{X}_0|\Psi\rangle = \alpha|001\rangle + \beta|110\rangle,$$

$$|\Psi_1\rangle = \mathbf{X}_1|\Psi\rangle = \alpha|010\rangle + \beta|101\rangle,$$

$$|\Psi_2\rangle = \mathbf{X}_2|\Psi\rangle = \alpha|100\rangle + \beta|011\rangle, \tag{5.4}$$

obliterating any dependence of the post-measurement state on the complex amplitudes α and β.

This might appear (and for some time was thought) to be the end of the story: quantum error correction is impossible because of the disruptive effect of the measurement needed to diagnose the error.

But there are more subtle ways to extract the information needed to diagnose and correct possible errors.

Although there continues to be a disruption in these refined procedures, the damaging effects are entirely shifted from the codeword Qbits to certain ancillary Qbits.

By coupling the codeword Qbits to these ancillary Qbits with appropriate 2-Qbit unitary gates, and then applying measurement gates only to the ancillas, one can extract information about certain relations prevailing among the codeword Qbits.

This more limited information turns out to be enough to diagnose and correct certain errors in a coherence-preserving manner, without revealing anything about the original uncorrupted codeword state.

The production of no information about the uncorrupted state is a necessary restriction on any error-correction procedure capable of perfectly restoring the uncorrupted state.

If one could get even partial information about the structure of a state without disrupting it, one could continue collecting additional information nondisruptively until one was well on they way to violating the no-cloning theorem.

An elementary but important point to notice is that all possible forms for the uncorrupted 3-Qbit codeword (5.3) — given by assigning all possible values to the amplitudes α and β— lie in a 2-dimensional subspace of the full 8-dimensional space containing all possible 3-Qbit states.

Furthermore each of the three allowed corruptions (5.4) also lies in a 2-dimensional subspace of the full 3-Qbit space.

The three subspaces containing the three allowed corruptions are each orthogonal to the subspace space containing the uncorrupted codeword, and each of the three is also orthogonal to the other two.

This turns out to be crucial to the success of the enterprise.

More generally, if we wanted to use an n-Qbit codeword in a model in which the only allowed errors were flips of a single Qbit, then we would require 2(1 + n) dimensions to accommodate the n+ 1 mutually orthogonal 2-dimensional subspaces associated with a general uncorrupted state and its n different 1-Qbit corruptions.

Since all possible states of nQbits span a 2n dimensional space, a necessary condition for an n-Qbit error correcting code to be possible is

$$2^{n-1} \geq 1 + n. \tag{5.5}$$

The smallest n satisfying (5.5) is n = 3, for which it holds as an equality.

This shows that the 3-Qbit code is, in this sense, perfect for the purpose of correcting errors limited to flips of a single Qbit.

Figure 5.3 shows that 3-Qbit codewords, as well as meeting this necessary condition for the correction of quantum bit-flip errors, actually do permit it to be carried out.

This shows that the 3-Qbit code is, in this sense, perfect for the purpose of correcting errors limited to flips of a single Qbit.

Figure 5.3 shows that 3-Qbit codewords, as well as meeting this necessary condition for the correction of quantum bit-flip errors, actually do permit it to be carried out.

The error detection and correction requires two additional ancillary Qbits (upper two wires), initially both in the state $|0\rangle$.

Both ancillas are targeted by pairs of cNOT gates controlled by subsets of the 3 codeword Qbits.

Figure 5.3

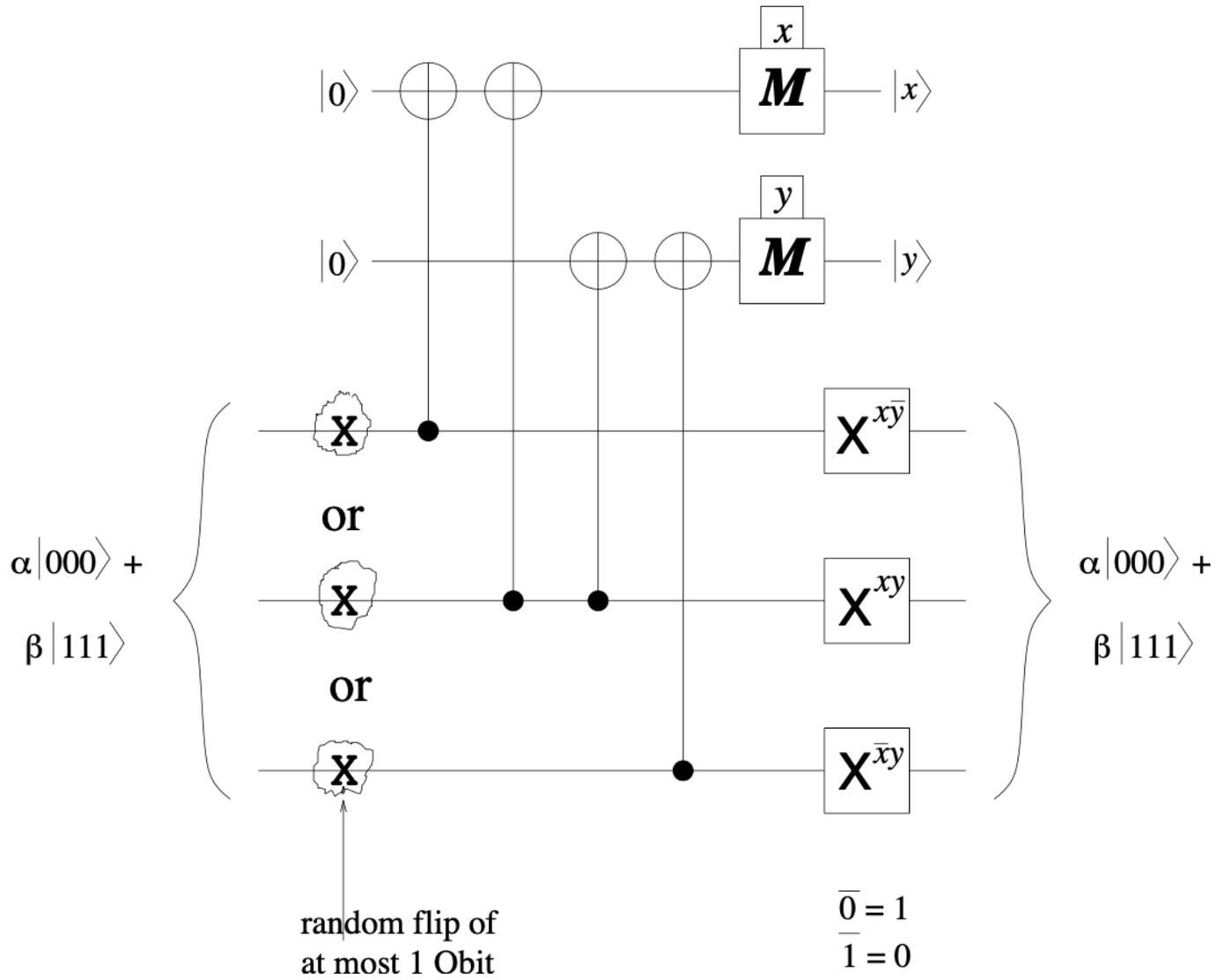


Figure 5.3. How to detect and correct the possible errors shown in Figure 5.2. One requires two ancillary Qbits (upper two wires), each initially in the state $|0\rangle$, coupled to the codeword Qbits by cNOT gates. After the cNOT gates have acted each ancilla is measured. If both measurements give 0, then none of the erroneous NOT gates on the left have acted and none of the error-correcting NOT gates on the right need to be applied (a = b= c = 0). If the upper measurement gate shows x = 1 and the lower one shows y= 0, then the uppermost of the three erroneous NOT gates has acted on the left. Its action is undone by applying the uppermost of the three NOT gates on the right. The other two possible 1-Qbit errors are similarly corrected.

Note first that if the 3-Qbit codeword has not been corrupted, so its state remains (5.3), then both the ancillary Qbits remain in the state $|0\rangle$ after the action of the cNOT gates, since the term $|000\rangle$ in the codeword results in none of the target Qbits being flipped, while the term $|111\rangle$ results in both of the target Qbits being flipped twice, which is equivalent to no flip.

In a similar way each of the three corruptions (5.4) results in a different unique final state for the ancillary Qbits.

The first of those corruptions results in $|0\rangle$ for the upper ancilla and $|1\rangle$ for the lower, since either term in the superposition $\alpha|001\rangle + \beta|110\rangle$ results in zero or two flips for the upper ancilla, and a single flip for the lower ancilla.

The next form in (5.4) produces a single flip for both ancillas, resulting in $|1\rangle$ for both.

The third results in $|1\rangle$ for the upper and $|0\rangle$ for the lower ancilla.

So if the two ancillary Qbits are measured after the cNOT gates have acted, the four possible readings, 00, 01, 10, and 11, of the two measurement gates reveal whether or not a random one of the codeword Qbits has been flipped and, in the event of a flip, which of the three has suffered it.

On the basis of this information one can either accept the codeword as uncorrupted or apply a NOT gate to the Qbit that has been identified as the flipped one, thereby restoring the initial uncorrupted state.

One easily confirms that this is precisely what is accomplished by the NOT gates on the extreme right of Figure 5.3.

This accomplishes what any valid quantum error-correction procedure must do: it restores the original uncorrupted state without revealing any information whatever about what the form of that state — the particular values of the amplitudes α and β— might actually be.

The procedure succeeds in preserving the superposition by extracting information only about correlations among the Qbits making up the codeword, without ever extracting information about individual Qbits.

Working only with correlations makes it possible to apply a single linear test that works equally well for diagnosing errors in either $|000\rangle$ or $|111\rangle$, and therefore also works for any superposition of them.

This simple example of quantum error correction requires the use of measurement gates to diagnose the error.

The outputs of the measurement gates are noted, and then used to determine which, if any, of a collection of error correcting NOT gates should be applied.

The procedure can be automated into a bigger quantum circuit, that eliminates (or almost eliminates) the need to use measurement gates combined with unitary gates which are or are not applied depending on the readings of the measurement gates.

This can be achieved by a combination of cNOT and Toffoli gates, controlled by the ancillary Qbits, as shown in Figure 5.4.

Figure 5.4

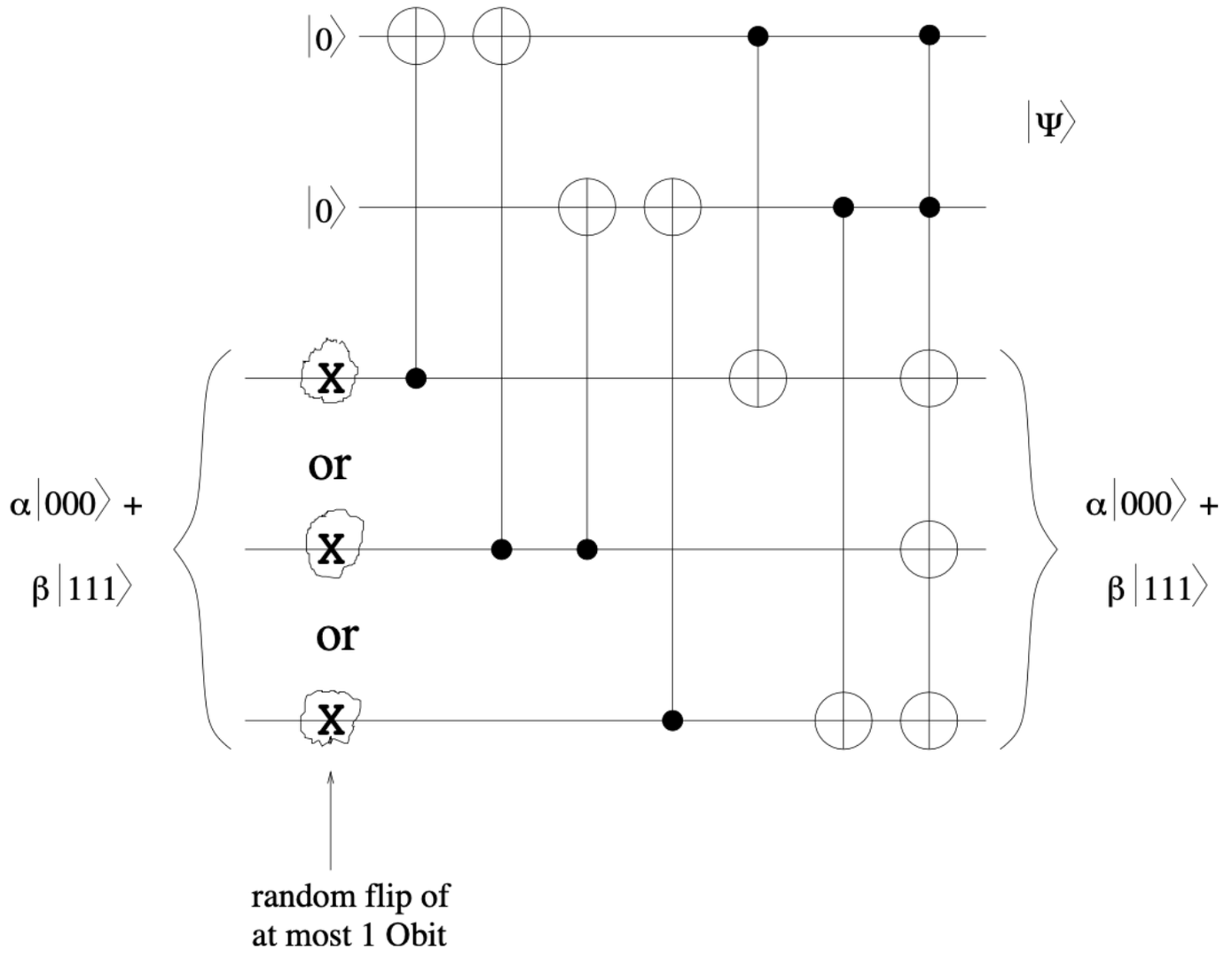


Figure 5.4. Automation of the error correction process of Figure 5.3. The three controlled gates on the right — one of them a doubly controlled Toffoli gate, and each of them with multiple targets — have precisely the same error-correcting effect on the three codeword Qbits as does the application of NOT gates contingent on measurement outcomes in Figure 5.3. The final state $|\Psi\rangle$ of the ancillas (which is also the state that determines the action of the three controlled gates on the right) is $|00\rangle$ if none of the erroneous NOT gates on the left has acted. It is $|10\rangle$ if only the upper erroneous NOT gate has acted, $|11\rangle$ if only the middle one has acted, and $|01\rangle$ if only the lower one has acted.

I said that this replacement of measurement gates by additional cNOT gates almost eliminates the need for measurement, because at the end of the process the state of the ancillary Qbits will depend on the character of the error and will in general no longer be the state $|0\rangle|0\rangle$ with which the error correction procedure starts.

If one wishes to reuse these ancillary Qbits for further error correction, it is necessary to reset them to their initial state $|0\rangle|0\rangle$.

This can efficiently be done by *measuring* them and applying the appropriate NOT gates if either is found to be in the state $|1\rangle$.

Thus measurement followed by NOT gates depending on the measurement outcome is still needed to prepare the circuit for a possible future error correction.

This procedure (automated or not) will continue to work even when the codeword Qbits are entangled with many other codeword Qbits, as they will be in the course of a nontrivial computation.

In such a case the codeword Qbits have no state of their own, the state of all the many codeword Qbits being of the form

$$\alpha|000\rangle|\Psi\rangle + \beta|111\rangle|\Phi\rangle, \qquad (5.6)$$

with the error correction applied to the 3 Qbits on the left.

One easily confirms that the added complication of entanglement with other Qbits has no affect on the validity of the error correction procedure.

There is an alternative way of representing the use of cNOT gates in Figure 5.3 to diagnose the error, which is useful in correcting quantum errors in more realistic cases.

The alternative point of view is based on the easily confirmed fact that the uncorrupted state (5.3) is left unaltered by either of the operators $\mathbf{Z}_2\mathbf{Z}_1$ and $\mathbf{Z}_1\mathbf{Z}_0$, while the three corruptions (5.4) are each eigenstates of $\mathbf{Z}_2\mathbf{Z}_1$ and $\mathbf{Z}_1\mathbf{Z}_0$ with distinct sets of eigenvalues: 1 and -1 in the case of $|\Psi_0\rangle$, -1 and -1 in the case of $|\Psi_1\rangle$, and -1 and 1 in the case of $|\Psi_2\rangle$.

While these last three facts can be confirmed directly from the explicit forms of $|\Psi_0\rangle$, $|\Psi 1\rangle$, and $|\Psi 2\rangle$ on the left of (5.4), it is worth noting, for purposes of comparison with some of the more complex cases that follow, that they also follow from the fact that $\mathbf{Z}_2\mathbf{Z}_1$ and $\mathbf{Z}_1\mathbf{Z}_0$ act as the identity on the uncorrupted state $|\Psi\rangle$, that the corrupted states are of the form $|\Psi_j\rangle = X_j\,|\Psi\rangle$, and that X_j commutes with $\mathbf{Z}_i$ when $i \neq j$, while $\mathbf{X}_j$ anticommutes with $\mathbf{Z}_j$: $\mathbf{Z}_j\mathbf{X}_j = -\mathbf{X}_j\mathbf{Z}_j$.

The resulting pattern of commutations (+) or anticommutations (–) is summarized in the following table:

	$\mathbf{X}_2$	$\mathbf{X}_1$	$\mathbf{X}_0$	$\mathbf{1}$
$\mathbf{Z}_2\mathbf{Z}_1$	$-$	$-$	$+$	$+$
$\mathbf{Z}_1\mathbf{Z}_0$	$+$	$-$	$-$	$+$

The joint eigenvalues of the commuting operators $\mathbf{Z}_2\mathbf{Z}_1$ and $\mathbf{Z}_1\mathbf{Z}_0$ thus nicely distinguish among the uncorrupted state and each of the three possible corruptions.

A procedure that takes advantage of this by sandwiching controlled $\mathbf{Z}_2\mathbf{Z}_1$ and controlled $\mathbf{Z}_1\mathbf{Z}_0$ gates between Hadamards acting on the control Qbits, is shown in Figure 5.5.

Figure 5.5

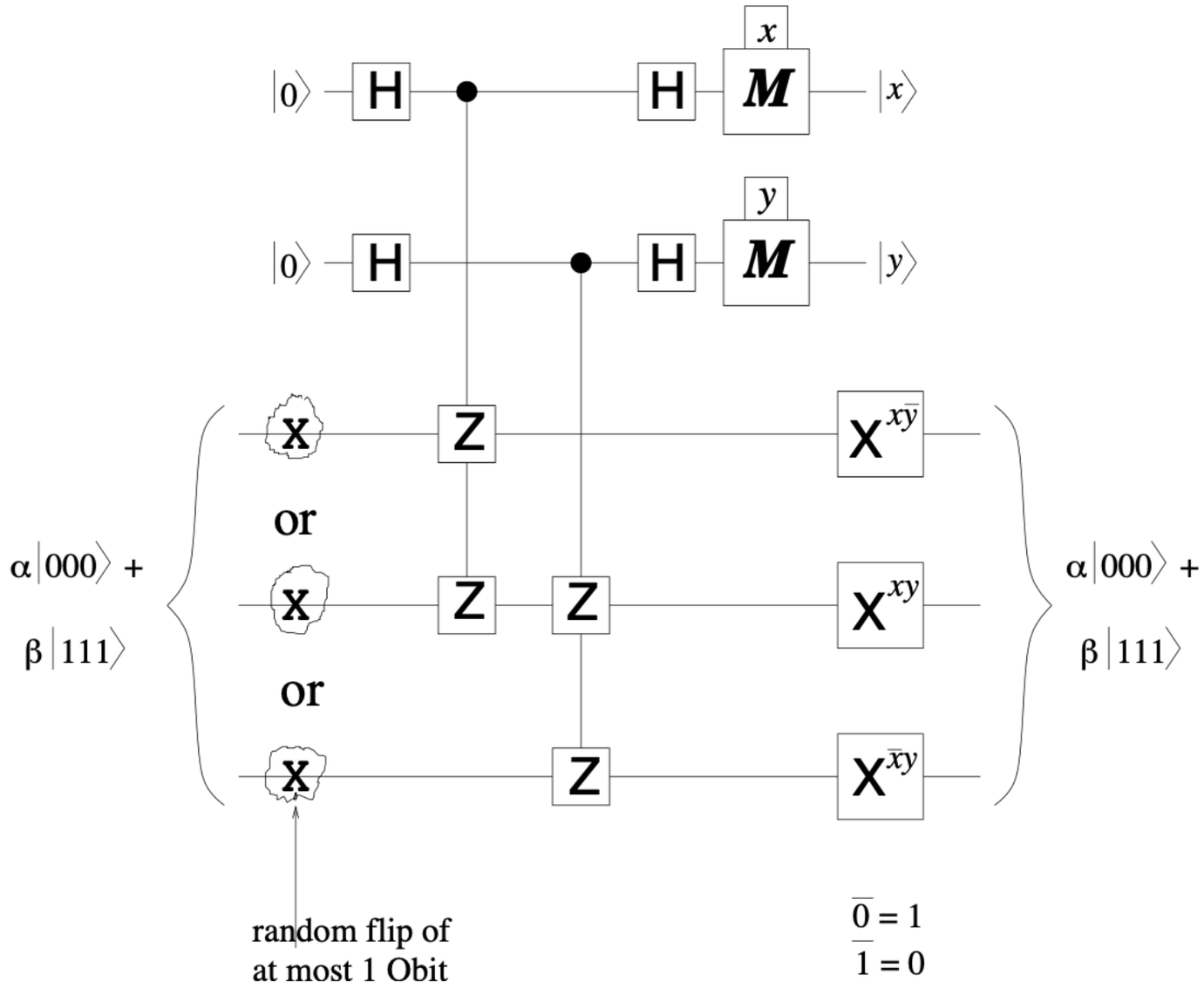


Figure 5.5. An apparently unnecessary complication of the error-correcting circuit in Figure 5.3, which transforms it into the more general form described in Sections C and D. The circuit is equivalent to that in Figure 5.3: (1) The cNOT gates in 5.3 can be replaced by controlled-Z gates provided Hadamard gates act on each ancilla before and after the controlled gates act; (2) Each of the four controlled-Z gates acts in the same way if its control and target Qbits are interchanged; and (3) Pairs of controlled gates with the same control Qbit and two different targets, can be combined into a single controlled gate with that control Qbit and a 2-Qbit target operation that is just the product of the two 1-Qbit target operations. The part of the circuit between and including the pairs of Hadamards on the right and left is a simple example of the more complex error-diagnosing circuits appearing in Figures 5.8, 5.9, and 5.25.

While it takes a little thought to confirm directly from the figure that Figure 5.5 does indeed accomplish error correction — we shall work this out in Part 5.C as a special case of a much more general procedure — one can confirm that it does by simply noting that Figure 5.5 is mathematically equivalent to Figure 5.3.

This equivalence follows from the facts that $\mathbf{X} = \mathbf{HZH}$, that $\mathbf{H}^2 = \mathbf{1}$, and that the action of controlled-**Z** is unaltered by exchanging the target and control Qbits.

This oversimplified example, in which only bit-flip errors are allowed, illustrates most of the features of quantum error correction one encounters in more realistic cases.

The more general procedure is complicated by the fact that, as noted above and made precise in Part 5.B shortly, the general error a Qbit can experience is more complicated than just a single bit flip.

As a result, one needs codewords containing more than 3 Qbits to correct general single-Qbit errors, and one requires more complicated diagnostic and corrective procedures than those of Figures 5.3-5.5, involving more than just a pair of ancillary Qbits.

But although the code words and error-correcting circuits are more elaborate, once we have identified the more general form of quantum errors, there are no further conceptual complications in understanding the kinds of procedures that can correct them.

The more general form Qbit errors can assume is discussed in Part 5.B.

Somewhat surprisingly, it turns out that the general 1-Qbit error can be viewed as a simple extension of what we have just described: in addition to the possibility of an extraneous **X** gate acting on the Qbit, there might also be an extraneous **Z** gate or an extraneous **Y= ZX** gate.

If we can diagnose and correct for each of these three possible corruptions, then we can correct for arbitrary 1-Qbit errors.

Section 5.C describes a generalization of the diagnostic scheme we have just exploited for extracting relational information about the Qbits that make up a codeword, by coupling groups of them to ancillary Qbits which are then measured.

It turns out that the operators needed to diagnose the error — generalizations of the operators $\mathbf{Z}_2\mathbf{Z}_1$ and $\mathbf{Z}_1\mathbf{Z}_0$ for the 3-Qbit code — are also extremely convenient for defining the more general code words.

In Part 5.D we examine two of the most important n-Qbit codes with $n > 3$ that are able to correct general single-Qbit errors: the 5-Qbit and 7-Qbit codes.

The 5-Qbit code is the ideal code for general 1-Qbit errors in the same sense that the 3-Qbit code is ideal for bit-flip errors.

The 7-Qbit code is more likely to be of practical interest, for reasons we shall describe.

The original 9-Qbit code discovered by Shor is now of only historical interest, and is relegated to later.

End Class 15 Class 16

5.B. The physics of error generation

Errors are not, of course, produced by extraneous gates, accidentally appearing in a circuit, as in the oversimplified example of Part 5.A.

They are produced by extraneous interactions with the world external to the computer, or with computationally irrelevant degrees of freedom of the computer itself.

Although one would like the state of the Qbits to evolve only under the action of the unitary transformations imposed by the gates of the computer, inevitably Qbits will interact, even if only weakly, with other physical systems or degrees of freedom, extraneous to the computation in which the Qbits are participating.

In a well designed computer such spurious interactions will be kept to a minimum, but their disruptive effects on the quantum state of the Qbits can grow continuously from zero, in contrast to disruptive effects on Cbits, which have to exceed a large threshold before a Cbit can change its state.

In a quantum computer such spurious changes of state will eventually accumulate to the point where the calculation falls apart, unless ongoing efforts are made to eliminate them.

To characterize the most general way in which a Qbit can be deflected from its computational task, we must finally acknowledge that Qbits are not the only things in the world that are described by quantum states.

The quantum theory provides the most fundamental description we have of everything in the world, and it describes everything in the world — not just Qbits — using quantum states.

This spectacular expansion of the scope of quantum states might not come as a complete surprise to the nonphysicist reader.

I have stressed all along that the quantum state of a Qbit or a collection of Qbits is not a property carried by those Qbits, but a way of concisely summarizing everything that we know has happened to them, to enable us to make statistical predictions about the information we might then be able to extract from them.

If quantum states are not properties inherent in the system they describe, but states of the knowledge we have managed to acquire about the prior history of the system — if they somehow incorporate fundamental aspects of how we exchange information with the world outside of us — then they might indeed have an applicability going beyond the particular kinds of systems we have applied them to up until now.

Indeed, nowhere in this exposition of quantum computation has it been necessary to refer to the individual character of the Qbits.

Whether they are spinning electrons, polarized photons, atoms in cavities, or any number of other things, the core of the quantum mechanical description of their computational behavior has been exactly the same.

So insofar as the assignment of quantum states to physical systems is a general feature of how we come to grips with the external world, it might not be unreasonable to assign a quantum state $|e\rangle$ to whatever part of the world comes into interactive contact with the Qbit or Qbits — their environment.

We will not make any specific assumptions about the character of that environment or of the quantum state $|e\rangle$ associated with it, beyond noting that, unlike the state of a single Qbit, the state of the environment is likely to be a state in a space of enormously many dimensions if there is any complexity to the environment that couples, however weakly, to the Qbit

If, in spite of this point of view, you nevertheless feel uncomfortable applying quantum states to non-computational degrees of freedom, then I invite you to regard $|e\rangle$ as the state of some enormous collection of irrelevant Qbits, from which one would like the computation to be completely decoupled, but which, for reasons beyond our control, somehow manage to interact weakly with the Qbits we are actually interested in.

I offer this invitation as a conceptual aid to computer scientists, uncomfortable with my claim that quantum states apply to the description of arbitrary physical systems, and not just Qbits.

But I also note that in recent years some physicists have suggested that the entire world should indeed be viewed as an enormous collection of Qbits.

This position has not attracted many adherents to date.

Returning from grand world views to the practical reality of errors in a quantum computation, we shall regard a single Qbit, initially in the state $|x\rangle$ ($x = 0$ or 1), as being, in the presence of an environment, part of a larger system consisting of the environment plus the Qbit, initially in the state $|e\rangle|x\rangle$.

In the ideal case, as the Qbit evolves under 1-Qbit unitary gates or interacts with other Qbits under 2-Qbit unitary gates, it stays uncorrelated with the environment.

The environmental component of the state is then irrelevant to the computational process and can be ignored, as we have been doing up to now.

Unfortunately, however, interactions with the environment will in general transform and entangle the states the Qbit and its environment.

The most general way in which this can come about can be expressed in the form

$$|e\rangle|0\rangle \rightarrow |e_0\rangle|0\rangle + |e_1\rangle|1\rangle,$$

$$|e\rangle|1\rangle \rightarrow |e_2\rangle|0\rangle + |e_3\rangle|1\rangle, \qquad (5.7)$$

where $|e\rangle$ is the initially uncorrelated state of the environment and $|e_0\rangle ... |e_3\rangle$ are possible final environmental states.

The environmental final states are not necessarily orthogonal or normalized, and are constrained only by the requirement that the two states on the right side of (5.7) should be orthogonal, since the Qbit-environment interaction is required, like any other physical interaction, to lead to a unitary development in time.

This corruption of a computation by the entanglement of the state of Qbits with the state of their environment is called *decoherence*.

It is the number-one enemy of quantum computation.

Included in (5.7) are cases like the oversimplified one we examined in Part 5.A, in which the Qbit remains isolated from the environment ($|e_i\rangle = a_i|e\rangle$, i = 0...3) but still suffers in that isolation an unintended unitary evolution.

But (5.7) also includes the case of major practical interest, in which the interaction with the environment has a small but otherwise quite general entangling effect on the Qbit:

$$|e_0\rangle \approx |e_3\rangle \approx |e\rangle; \qquad \langle e_1|e_1\rangle, \quad \langle e_2|e_2\rangle \ll 1. \tag{5.8}$$

By using the projection operators

$$\mathbf{P}_x = \frac{1 + (-1)^x \mathbf{Z}}{2}, \tag{5.9}$$

which project onto the 1-Qbit states |x⟩, x = 0, 1, we can combine the two time evolutions in (5.7) into a single form:

$$|e\rangle|x\rangle \rightarrow \Big(\Big[|e_0\rangle\mathbf{1} + |e_1\rangle\mathbf{X}\Big]\mathbf{P}_0\Big)|x\rangle + \Big(\Big[|e_2\rangle\mathbf{X} + |e_3\rangle\mathbf{1}\Big]\mathbf{P}_1\Big)|x\rangle. \tag{5.10}$$

In (5.10) I have introduced the convenient notation |e⟩**U** to describe the linear operator from a 1-Qbit to a many-Qbit space that takes the 1-Qbit state |ψ⟩ into the many-Qbit state |e⟩ ⊗ **U**|ψ⟩; like most embellishments of Dirac notation it is defined so that the appropriate form of the associative law holds:

$$\big(|e\rangle\mathbf{U}\big)|\psi\rangle = |e\rangle \otimes \mathbf{U}|\psi\rangle. \tag{5.11}$$

Using the explicit form (5.9) of the two projection operators, defining

$$\mathbf{Y} = \mathbf{ZX}, \tag{5.12}$$

and continuing to use the notational convention of (5.11), we can rewrite (5.10) as

$$|e\rangle|x\rangle \rightarrow \Big(\frac{|e_0\rangle + |e_3\rangle}{2}\mathbf{1} + \frac{|e_0\rangle - |e_3\rangle}{2}\mathbf{Z} + \frac{|e_2\rangle + |e_1\rangle}{2}\mathbf{X} + \frac{|e_2\rangle - |e_1\rangle}{2}\mathbf{Y}\Big)|x\rangle, \tag{5.13}.$$

There is nothing special about the particular environmental states appearing in (5.13), so we can rewrite it more compactly in terms of four other (in general unnormalized) states |a⟩, |b⟩, |c⟩, and |d⟩ of the environment as

$$|e\rangle|x\rangle \rightarrow \Big(|d\rangle\mathbf{1} + |a\rangle\mathbf{X} + |b\rangle\mathbf{Y} + |c\rangle\mathbf{Z}\Big)|x\rangle. \tag{5.14}$$

The time development represented by the arrow in (5.14) is unitary and therefore linear, and the combination of environmental states and unitary operators on the right side of (5.14) also acts linearly on $|x\rangle$.

Therefore (5.14) holds not only for $|e\rangle|0\rangle$ and $|e\rangle|1\rangle$ but for any superposition $\alpha|e\rangle|0\rangle + \beta|e\rangle|1\rangle = |e\rangle(\alpha|0\rangle + \beta|1\rangle) = |e\rangle|\psi\rangle$, in the form

$$|e\rangle|\psi\rangle \rightarrow \Big(|d\rangle\mathbf{1} + |a\rangle\mathbf{X} + |b\rangle\mathbf{Y} + |c\rangle\mathbf{Z}\Big)|\psi\rangle. \qquad (5.15)$$

The actions of **X**, **Z**, and **Y** on the uncorrupted state $|\psi\rangle$ are often described as subjecting the Qbit to a bit-flip error, a phase error, or a combined bit-flip and phase error.

With this terminology, a general environmental degradation of the state of a Qbit, which can always be put in the form (5.15), can be viewed as a superposition of no error (1), a bit-flip error (**X**), a combined bit-flip and phase error (**Y**), and a phase error (**Z**).

The oversimplified example of Part 5.A ignored the possibility of phase errors (**Z**) and combined errors (**Y**).

If we were to apply this analysis to the corruption of an n-Qbit code word $|\Psi\rangle_n$, we would end up with a combined state of the code word and the environment of the form

$$|e\rangle|\Psi\rangle \rightarrow \sum_{\mu_1=0}^{3} \cdots \sum_{\mu_n=0}^{3} |e_{\mu_1 \cdots \mu_n}\rangle \mathbf{X}^{(\mu_1)} \otimes \cdots \otimes \mathbf{X}^{(\mu_n)} |\Psi\rangle_n, \qquad (5.16)$$

where

$$\mathbf{X}^{(0)} = \mathbf{1},\ \mathbf{X}^{(1)} = \mathbf{X},\ \mathbf{X}^{(2)} = \mathbf{Y},\ \mathbf{X}^{(3)} = \mathbf{Z}. \qquad (5.17)$$

Crucial to the possibility of error correction by the use of codewords is a physical assumption, analogous to our assumption in Part 5.A that only a single Qbit suffers a flip:

If $|\Psi\rangle$ is a state of a small number n of Qbits making up such an encoded n-Qbit word, then the probability of corruption of the code word is so small that the terms in (5.16) differing from the term $\mathbf{1} \otimes \cdots \otimes \mathbf{1}$ that reproduces the uncorrupted state, are dominated by those in which only a single one of the $\mathbf{X}^{(\mu_i)}$ differs from $\mathbf{1}$.

If this condition is met, then the general form of a corrupted n-Qbit code word is a superposition of terms in which each individual Qbit making up the word has suffered a degradation of the form (5.15):

$$|e\rangle|\Psi\rangle \rightarrow \left(|d\rangle\mathbf{1} + \sum_{i=0}^{n-1} |a_i\rangle\mathbf{X}_i + |b_i\rangle\mathbf{Y}_i + |c_i\rangle\mathbf{Z}_i\right)|\Psi\rangle. \qquad (5.18)$$

One can allow for the more general possibility of two or more Qbits being corrupted together if one is willing to use longer codewords to correct such errors.

The examples of error correction given below are all at the level of single-Qbit errors of the form (5.18) in the code word.

The extent to which the dominant sources of error will actually be of this form may well depend on the kind of physical system used to realize the Qbits.

Eventually the theory of quantum error correction will have to face this issue.

Meanwhile this possible future source of difficulty should not distract you from appreciating how remarkable it is that an error-correction procedure exists at all, even in the restricted setting of single-Qbit errors.

To correct 1-Qbit errors we require a procedure that restores a corrupted state of the form

$$|d\rangle|\Psi\rangle + \sum_{i=0}^{n-1}\Big(|a_i\rangle\mathbf{X}_i|\Psi\rangle + |b_i\rangle\mathbf{Y}_i|\Psi\rangle + |c_i\rangle\mathbf{Z}_i|\Psi\rangle\Big) \tag{5.19}$$

to the uncorrupted form

$$|e\rangle|\Psi\rangle, \tag{5.20}$$

where $|e\rangle$ is the environmental state accompanying whichever of the 3n+1 terms in (5.19) our error correction procedure has projected the corrupted state onto.

If the term in $\mathbf{X}_i$ were the only one present in (5.19), we could use a 3-Qbit codeword (n = 2) and achieve this projection by applying precisely the error correction technique described in Part 5.A.

But to deal with the additional possibilities associated with the terms in $\mathbf{Y}_i$ and $\mathbf{Z}_i$ we require longer codewords, and more elaborate diagnostic methods.

5.C. Diagnosing error syndromes.

Before turning to specific quantum error-correcting codes, it is useful to anticipate the general structure of the gates we will be using to identify and project onto a particular term in the general 1-Qbit corruption (5.19) of a codeword.

As noted, these will be generalizations of the controlled $\mathbf{Z}_2\mathbf{Z}_1$ and $\mathbf{Z}_1\mathbf{Z}_0$ gates used to diagnose errors in the artificial case where only bit-flip errors are allowed.

Let **A** be *any* n-Qbit hermitian operator whose square is the unit operator:

$$\mathbf{A}^2 = \mathbf{1}. \tag{5.21}$$

It follows from (5.21) that **A** is unitary, since $\mathbf{A}^\dagger = \mathbf{A}$. The eigenvalues of **A** can only be 1 or −1, since **A** acting twice on an eigenstate leaves it unchanged, and therefore if $\mathbf{A}|\psi\rangle = a\psi$, then
$|\psi\rangle = \mathbf{A}^2|\psi\rangle = \mathbf{A}a|\psi\rangle = a\mathbf{A}|\psi\rangle = a^2|\psi\rangle$, which requires $a^2 = 1$, so $a = \pm 1$.

The projection operators onto the subspaces of states with eigenvalue +1 and −1 are, respectively,

$$\mathbf{P}_0^A = \frac{\mathbf{1} + \mathbf{A}}{2} \text{ and } \mathbf{P}_1^A = \frac{\mathbf{1} - \mathbf{A}}{2}. \tag{5.22}$$

Since $\mathbf{P}_0 + \mathbf{P}_1 = 1$, any state $|\psi\rangle$ can be expressed as a superposition of its projections into these two subspaces: $|\psi\rangle = \mathbf{P}_0|\psi\rangle + \mathbf{P}_1|\psi\rangle$.

Examples of such **A** are, of course, the operators $\mathbf{Z}_2\mathbf{Z}_1$ and $\mathbf{Z}_1\mathbf{Z}_0$ encountered in the 3-Qbit code for correcting bit-flip errors.

In the more general cases we shall be examining the operators **A** will be more general products of both **Z** and **X** operators associated with different Qbits in the code word; for example $\mathbf{A} = \mathbf{Z}_4\mathbf{X}_3\mathbf{Z}_2\mathbf{X}_1\mathbf{X}_0$.

In addition to the n-Qbits on which **A** acts, we introduce an ancillary Qbit and consider the controlled operator **cA** which acts as **A** on the n-Qbits when the state of the ancilla is $|1\rangle$ and as the identity when the state of the ancilla is $|0\rangle$.

If the state of the ancilla is a superposition of $|0\rangle$ and $|1\rangle$, the action of **cA** is defined by linearity.

When **A** is a product of 1-Qbit operators, the operator **cA** can be taken to be a product of ordinary 2-Qbit controlled operators.

In the last example of the preceding paragraph **cA** would be $\mathbf{cZ}_4\mathbf{cX}_3\mathbf{cZ}_2\mathbf{cX}_1\mathbf{cX}_0$ where each of the five terms has a different target Qbit, but all are controlled by one and the same ancilla.

If the ancilla is initially in the state $|0\rangle$ and one applies a Hadamard transform **H** to the ancilla both before and after applying **cA** to the n + 1 Qbits, then if the initial state of the n Qbits is $|\Psi\rangle$, then the n Qbits will end up entangled with the ancilla in the state

$$(\mathbf{H}\otimes\mathbf{1})\mathbf{cA}(\mathbf{H}\otimes\mathbf{1})|0\rangle|\Psi\rangle = (\mathbf{H}\otimes\mathbf{1})\mathbf{cA}\tfrac{1}{\sqrt{2}}\big(|0\rangle+|1\rangle\big)|\Psi\rangle =$$

$$(\mathbf{H}\otimes\mathbf{1})\tfrac{1}{\sqrt{2}}\big(|0\rangle|\Psi\rangle+|1\rangle\mathbf{A}|\Psi\rangle\big) = \tfrac{1}{2}\big(|0\rangle+|1\rangle\big)|\Psi\rangle+\tfrac{1}{2}\big(|0\rangle-|1\rangle\big)\mathbf{A}|\Psi\rangle\big) =$$

$$|0\rangle\tfrac{1}{2}\big(\mathbf{1}+\mathbf{A}\big)|\Psi\rangle+|1\rangle\tfrac{1}{2}\big(\mathbf{1}-\mathbf{A}\big)|\Psi\rangle = |0\rangle\mathbf{P}_0^A|\Psi\rangle+|1\rangle\mathbf{P}_1^A|\Psi\rangle. \qquad (5.23)$$

If we now measure the ancilla, then according to the generalized Born rule, if the measurement gate indicates 0 or 1, then the state of the n Qbits becomes the (renormalized) projection of $|\Psi\rangle$ into the subspace of positive (eigenvalue +1) or negative (eigenvalue −1) eigenstates of **A**.

This is illustrated for the case $\mathbf{A} = \mathbf{Z}_4\mathbf{X}_3\mathbf{Z}_2\mathbf{X}_1\mathbf{X}_0$ in Figure 5.6.

Figure 5.6

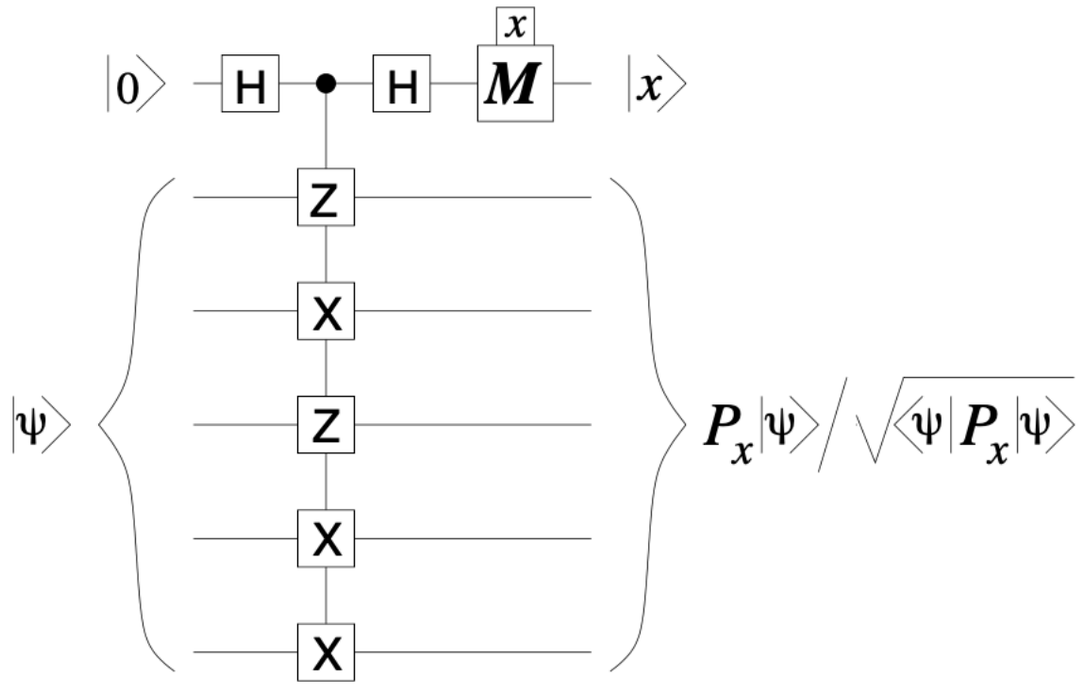


Figure 5.6. The way in which measurement gates are employed in quantum error correction. The ancilla (upper wire) is initially in the state zero. The remaining 5 Qbits are initially in the state $|\Psi\rangle$. If the measurement gate acting on the ancilla gives the result x(0 or 1) then the final state of the five Qbits will be the (renormalized) projection $\mathbf{P}_x|\Psi\rangle$ of the initial state into the subspace spanned by the eigenstates of $\mathbf{Z}_4\mathbf{X}_3\mathbf{Z}_2\mathbf{X}_1\mathbf{X}_0$ with eigenvalue $(-1)^x$.

This procedure is called *measuring* **A** or a *measurement* of **A**.

The terminology reflects the fact that it is a generalization of the ordinary process of measuring a single Qbit, to which it reduces when n = 1 and **A** = **Z**.

In that case the subspaces spanned by the positive and negative eigenstates of **Z** are just the one-dimensional subspaces spanned by $|0\rangle$ and $|1\rangle$, and the probabilities of the two outcomes, as one can easily check, are indeed given by the Born rule.

In error correction one needs several such hermitian operators, each squaring to unity, all acting on the same nQbits.

For concreteness consider the case of three such operators, **A**, **B**, and **C**.

Introduce an ancillary Qbit for each operator, labeling the ancillas 2, 1, and 0, and introduce controlled operators **cA**, **cB**, and **cC**, each controlled by the corresponding ancilla.

Now apply Hadamards to each of the ancillas (each initially taken to be in the state $|0\rangle$), both before and after the product of all the controlled operators acts.

The result (see Figure 5.7) is the obvious generalization of (5.23), taking $|0\rangle|0\rangle|0\rangle|\Psi\rangle$ into

$$\big(\mathbf{H}_2\mathbf{H}_1\mathbf{H}_0\big)\big(\mathbf{cC}\,\mathbf{cB}\,\mathbf{cA}\big)\big(\mathbf{H}_2\mathbf{H}_1\mathbf{H}_0\big)|0\rangle|0\rangle|0\rangle|\Psi\rangle =$$

$$\sum_{x_2=0}^{1}\sum_{x_1=0}^{1}\sum_{x_0=0}^{1}|x_2\rangle|x_1\rangle|x_0\rangle\Big(\frac{1+(-1)^{x_2}\mathbf{C}}{2}\Big)\Big(\frac{1+(-1)^{x_1}\mathbf{B}}{2}\Big)\Big(\frac{1+(-1)^{x_0}\mathbf{A}}{2}\Big)|\Psi\rangle =$$

$$\sum_{x_2=0}^{1}\sum_{x_1=0}^{1}\sum_{x_0=0}^{1}|x_2\rangle|x_1\rangle|x_0\rangle\mathbf{P}^C_{x_2}\mathbf{P}^B_{x_1}\mathbf{P}^A_{x_0}|\Psi\rangle. \tag{5.24}$$

Figure 5.7

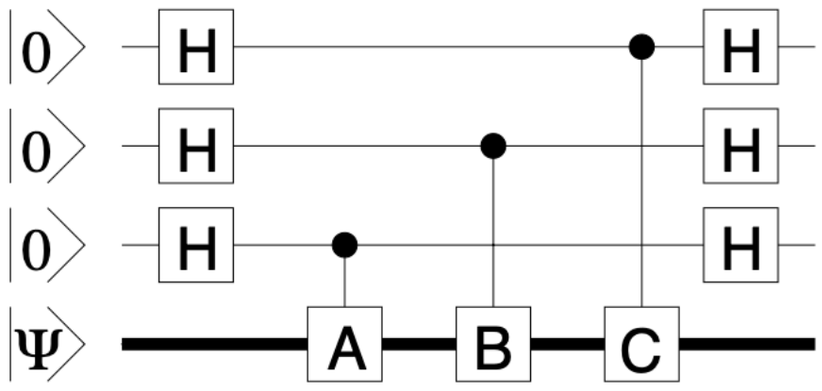


Figure 5.7. A, B, and C are commuting operators satisfying $\mathbf{A}^2 = \mathbf{B}^2 = \mathbf{C}^2 = \mathbf{1}$. They act on the n-Qbit state $|\Psi\rangle$, associated with the thick lower wire. The effect of measuring the three ancillas (top three wires) is to project the state of the n Qbits associated with the lower wire into its component in one of the eight eigenspaces of **A**, **B**, and **C**. If the results of measuring the control bits associated with **A**, **B**, and **C** are x_0, x_1, and x_2 then the projection is into the eigenspace with eigenvalues $(-1)^{x_0}$, $(-1)^{x_1}$, and $(-1)^{x_2}$. Such a process is called "measuring **A**, **B**, and **C**." When n = 3 and **A**, **B**, and **C** are three different one-Qbit **Z** operators, the process is equivalent to an ordinary measurement of the three Qbits on which **A**, **B**, and **C** act.

If **A**,**B**, and **C** all *commute*— which is always the case in the examples relevant to error correction — then the state

$$\mathbf{P}^{C}_{x_2}\mathbf{P}^{B}_{x_1}\mathbf{P}^{A}_{x_0}|\Psi\rangle = \left(\frac{1+(-1)^{x_2}\mathbf{C}}{2}\right)\left(\frac{1+(-1)^{x_1}\mathbf{B}}{2}\right)\left(\frac{1+(-1)^{x_0}\mathbf{A}}{2}\right) \tag{5.25}$$

is an eigenstate of all the operators **C**, **B**, and **A**, with respective eigenvalues

$$(-1)^{x_2}, (-1)^{x_1}, \text{ and } (-1)^{x_0}. \tag{5.26}$$

This follows directly from the fact that if $\mathbf{V}^2 = 1$ then

$$\mathbf{V}\left(\frac{1+(-1)^{x}\mathbf{V}}{2}\right) = (-1)^{x}\left(\frac{1+(-1)^{x}\mathbf{V}}{2}\right). \tag{5.27}$$

So measurement of the three ancillas projects the nQbits into one of the eight simultaneous eigenspaces of the three commuting operators **C**, **B**, and **A**, and the outcome $x_2x_1x_0$ of the measurement determines which eigenspace it is.

This process is described as a joint measurement of **C**, **B**, and **A**.

Note that if **A**, **B**, and **C** are 1-Qbit operators $\mathbf{Z}_i$, $\mathbf{Z}_j$, and $\mathbf{Z}_k$ that act on the i-th, j-th, and k-th of the nQbits, then this process reduces to the the ordinary measurement of those three Qbits, since $\frac{1}{2}(1+(-1)^{x}\mathbf{Z})$ projects on the 1-Qbit state $|x\rangle$.

The two equivalent error-correction circuits in Figures 5.3 and 5.5 are measurements, in this generalized sense, of the two commuting operators $\mathbf{A}= \mathbf{Z}_2\mathbf{Z}_1$ and $\mathbf{B}= \mathbf{Z}_1\mathbf{Z}_0$.

5.D. Error correcting codes

The form (5.18) of a general 1-Qbit error on an n-Qbit codeword reveals that to correct errors it is necessary to make a measurement, in the sense of Part 5.C, that projects a possibly corrupted codeword into an identifiable one of 1+3n orthogonal 2-dimensional spaces: one 2-dimensional subspace for the uncorrupted codeword $|\Psi\rangle$, and 3n additional 2-dimensional subspaces for each of the 1-Qbit error terms $X_i|\Psi\rangle$, $Y_i|\Psi\rangle$, and $Z_i|\Psi\rangle$, i= 0...n– 1, in (5.18).

Thus the 2n-dimensional space spanned by all the states of the n Qbits must be large enough to contain 1 + 3n orthogonal 2-dimensional subspaces, giving us the condition

$$2^{n-1} \geq (3n+1) \qquad (5.28)$$

on an n-Qbit code capable of correcting a general 1-Qbit error.

The lowest n satisfying this condition is n = 5, where it holds as an equality.

Remarkably, there is indeed a 5 Qbit code for which this can be done.

This is reminiscent of the situation in Part A, where it was necessary only to discriminate between the uncorrupted codeword $|\Psi\rangle$ and the n NOT-corruptions $X_i|\Psi\rangle$.

Here the number of Qbits had to satisfy (5.5), which is first satisfied (again as an equality) when n= 3.

The 5-Qbit code is the most compact and elegant of the quantum error correcting codes, but it suffers from the fact that it is difficult to construct the appropriate generalizations of one- and 2-Qbit gates between codewords.

I therefore go on to describe a second 7-Qbit code which overcomes this problem.

The first quantum error correcting code, discovered by Peter Shor, which uses a 9-Qbit generalization of the 3-Qbit code of Part A, and is now only of historical interest is described later.

The 5-Qbit code

The two 5-Qbit code words $|\overline{0}\rangle$ and $|\overline{1}\rangle$ are most conveniently defined in terms of the very operators, described in general terms in Part C, that will be used to diagnose the error syndrome.

So we begin by specifying those operators.

To distinguish 1+3× 5 = 16 mutually orthogonal 2-dimensional subspaces we require four such mutually commuting hermitian operators that square to unity, since each can independently have two eigenvalues (±) and 24 = 16.

These operators are defined as follows:

$$\begin{aligned}
\mathbf{M}_0 &= \mathbf{Z}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Z}_4, \\
\mathbf{M}_1 &= \mathbf{Z}_2\mathbf{X}_3\mathbf{X}_4\mathbf{Z}_0, \\
\mathbf{M}_2 &= \mathbf{Z}_3\mathbf{X}_4\mathbf{X}_0\mathbf{Z}_1, \\
\mathbf{M}_3 &= \mathbf{Z}_4\mathbf{X}_0\mathbf{X}_1\mathbf{Z}_2. \qquad (5.29)
\end{aligned}$$

Each of the $\mathbf{M}_i$ squares to unity because each is a product of commuting operators that square to unity.

To check that the $\mathbf{M}_i$ are mutually commuting note that all the individual $\mathbf{M}_i$ and $\mathbf{Z}j$ operators commute with one another except for an $\mathbf{X}_i$ and $\mathbf{Z}_i$ with the same index, which anti-commute: $\mathbf{X}_i\mathbf{Z}_i = -\mathbf{Z}_i\mathbf{X}_i$.

But in converting the product of any two different $\mathbf{M}_i$ to the product in the reverse order by reversing the orders of the individual $\mathbf{X}_i$ and $\mathbf{Z}_i$ operators that make them up, one always encounters exactly two interchanges that result in a minus sign.

One might be tempted to break the irritating asymmetry of (5.29) by adding to the list

$$\mathbf{M}_4 = \mathbf{Z}_0\mathbf{X}_1\mathbf{X}_2\mathbf{Z}_3, \tag{5.30}$$

but it is not independent of the other four.

Every $\mathbf{X}_i$ and $\mathbf{Z}_i$ appears exactly twice in the product of all five $\mathbf{M}_i$, and since $\mathbf{X}_i$ and $\mathbf{Z}_j$ anticommute if $i = j$ and commute if $i \neq j$, the product of all 5 $\mathbf{M}_i$ will be either 1 or -1.

One easily checks that

$$\mathbf{M}_0\mathbf{M}_1\mathbf{M}_2\mathbf{M}_3\mathbf{M}_4 = \mathbf{1}, \tag{5.31}$$

and therefore

$$\mathbf{M}_4 = \mathbf{M}_0\mathbf{M}_1\mathbf{M}_2\mathbf{M}_3. \tag{5.32}$$

The 5-Qbit code words are most clearly and usefully defined in terms of the $\mathbf{M}_i$ (rather than giving their explicit expansion in computational-basis states):

$$|\overline{0}\rangle = \tfrac{1}{4}(\mathbf{1}+\mathbf{M}_0)(\mathbf{1}+\mathbf{M}_1)(\mathbf{1}+\mathbf{M}_2)(\mathbf{1}+\mathbf{M}_3)|00000\rangle,$$

$$|\overline{1}\rangle = \tfrac{1}{4}(\mathbf{1}+\mathbf{M}_0)(\mathbf{1}+\mathbf{M}_1)(\mathbf{1}+\mathbf{M}_2)(\mathbf{1}+\mathbf{M}_3)|11111\rangle. \qquad (5.33)$$

We defer a discussion of how to produce five Qbits in either of these states until after discussing how the states work to correct 1-Qbit errors.

Since each **M** flips two Qbits, $|\overline{0}\rangle$ is a superposition of computational basis states with an odd number of 0's (and even number of 1's), while $|\overline{1}\rangle$ is a superposition of states with an odd number of 1's (and even number of 0's).

Consequently the two codeword states are orthogonal.

They are also normalized to unity. Since $\mathbf{M}^2_i = \mathbf{1}$, $\langle\overline{1}|\overline{1}\rangle$

$$(\mathbf{1}+\mathbf{M}_i)^2 = 2(\mathbf{1}+\mathbf{M}_i). \qquad (5.34)$$

So we have

$$\langle\overline{0}|\overline{0}\rangle = \langle 00000|(\mathbf{1}+\mathbf{M}_0)(\mathbf{1}+\mathbf{M}_1)(\mathbf{1}+\mathbf{M}_2)(\mathbf{1}+\mathbf{M}_3)|00000\rangle,$$

$$\langle\overline{1}|\overline{1}\rangle = \langle 11111|(\mathbf{1}+\mathbf{M}_0)(\mathbf{1}+\mathbf{M}_1)(\mathbf{1}+\mathbf{M}_2)(\mathbf{1}+\mathbf{M}_3)|11111\rangle. \qquad (5.35)$$

If we expand the products of $1 + \mathbf{M}_i$ into sixteen terms, the term 1 contributes 1 to $\langle\overline{0}|\overline{0}\rangle$ and to $\langle\overline{1}|\overline{1}\rangle$.

Each of the remaining 15 terms can be reduced, using (5.31) (and the fact that each $\mathbf{M}^2_i = \mathbf{1}$) to either a single $\mathbf{M}_i$ or a product of two (i= 0. . . 4).

So each of the 15 terms flips either two or four Qbits and contributes 0 to the inner products.

Because the $\mathbf{M}_i$ all commute and because

$$\mathbf{M}_i(\mathbf{1} + \mathbf{M}_i) = (\mathbf{1} + \mathbf{M}_i), \tag{5.36}$$

the states $|0\rangle, |1\rangle$, and their superpositions

$$|\Psi\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle \tag{5.37}$$

are all eigenstates of each of the $\mathbf{M}_i$ with eigenvalue 1.

The fifteen possible corruptions of (5.37) appearing in the corrupted state (5.18) are also eigenstates, distinguished by the $15 = 2^4 - 1$ other possible sets of eigenvalues ± 1 that the four $\mathbf{M}_i$ can have.

To see this note first that each $\mathbf{X}_i$, $\mathbf{Y}_i$, and $\mathbf{Z}_i$ commutes or anticommutes with all four $\mathbf{M}_i$.

Therefore each of the terms $\mathbf{X}_i|\Psi\rangle$, $\mathbf{Y}_i|\Psi\rangle$, or $\mathbf{Z}_i|\Psi\rangle$ appearing in (5.18) is indeed an eigenstate of each $\mathbf{M}_i$ with eigenvalue 1 or -1.

The following table indicates whether each $\mathbf{M}_i$ commutes (+) or anticommutes (−) with each of the $\mathbf{X}_i$, $\mathbf{Y}_i$, $\mathbf{Z}_i$, and (trivially) the unit operator $\mathbf{1}$.

Inspection of the table reveals that each of the 16 possible binary columns of 4 symbols (+ or−) appears in exactly one column.

Therefore when the four $\mathbf{M}_i$ are measured, the corrupted state (5.18) is projected back to its original form if all four eigenvalues are +1, or projected onto one of the fifteen corrupted states $\mathbf{X}_1|\Psi\rangle, \ldots, \mathbf{Z}_5|\Psi\rangle$ depending on which column in the table describes the eigenvalues.

In each corrupted case the original state can be restored by application of the corresponding unitary transformation $\mathbf{X}_i$, $-\mathbf{Y}_i = \mathbf{X}_i\mathbf{Z}_i$, or $\mathbf{Z}_i$ to the appropriate Qbit.

	$\mathbf{X}_0\mathbf{Y}_0\mathbf{Z}_0$	$\mathbf{X}_1\mathbf{Y}_1\mathbf{Z}_1$	$\mathbf{X}_2\mathbf{Y}_2\mathbf{Z}_2$	$\mathbf{X}_3\mathbf{Y}_3\mathbf{Z}_3$	$\mathbf{X}_4\mathbf{Y}_4\mathbf{Z}_4$	$\mathbf{1}$
$\mathbf{M}_0 = \mathbf{Z}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Z}_4$	$+\,+\,+$	$-\,-\,+$	$+\,-\,-$	$+\,-\,-$	$-\,-\,+$	$+$
$\mathbf{M}_1 = \mathbf{Z}_2\mathbf{X}_3\mathbf{X}_4\mathbf{Z}_0$	$-\,-\,+$	$+\,+\,+$	$-\,-\,+$	$+\,-\,-$	$+\,-\,-$	$+$
$\mathbf{M}_2 = \mathbf{Z}_3\mathbf{X}_4\mathbf{X}_0\mathbf{Z}_1$	$+\,-\,-$	$-\,-\,+$	$+\,+\,+$	$-\,-\,+$	$+\,-\,-$	$+$
$\mathbf{M}_3 = \mathbf{Z}_4\mathbf{X}_0\mathbf{X}_1\mathbf{Z}_2$	$+\,-\,-$	$+\,-\,-$	$-\,-\,+$	$+\,+\,+$	$-\,-\,+$	$+$

A circuit that measures the four operators (5.29) is shown in Figure 5.8

Figure 5.8

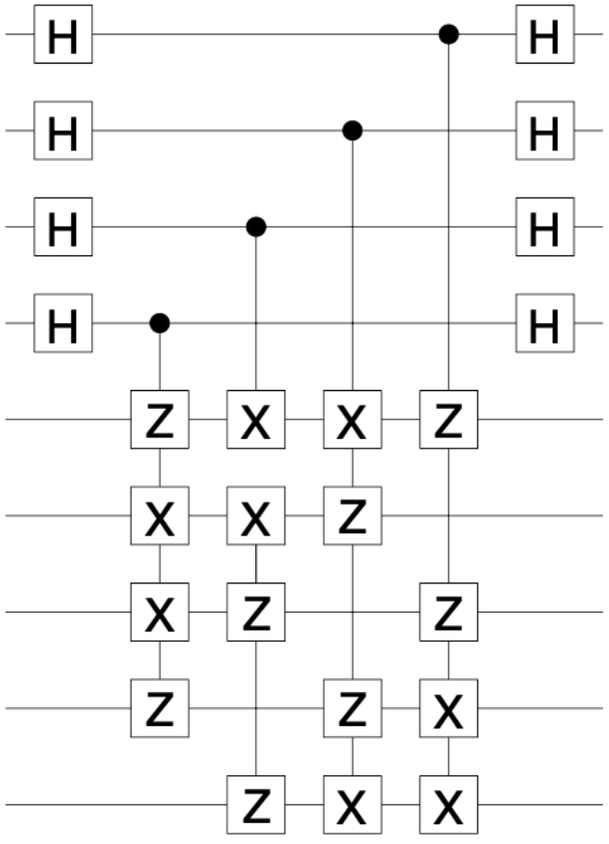


Figure 5.8. A circuit to measure the error syndrome for the 5-Qbit code. The five Qbits are the five lower wires. The four upper wires are the ancillas to be measured in the manner of Figure 5.1, associated with measuring the four commuting operators $\mathbf{Z}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Z}_4$, $\mathbf{Z}_2\mathbf{X}_3\mathbf{X}_4\mathbf{Z}_0$, $\mathbf{Z}_3\mathbf{X}_4\mathbf{X}_0\mathbf{Z}_1$, and $\mathbf{Z}_4\mathbf{X}_0\mathbf{X}_1\mathbf{Z}_2$ of (5.29). (When controlled Z gates are present together with controlled NOT gates, a figure can be made more attractive by representing the cNOT gates as controlled X gates.)

The perfect efficiency of the 5-Qbit code leads to a straightforward way to manufacture the two 5-Qbit codeword states (5.33).

As noted above, the 16 distinct sets of eigenvalues for the four mutually commuting operators $\mathbf{M}_i$ decompose the 32-dimensional space of 5-Qbits into 16 mutually orthogonal 2-dimensional subspaces, spanned by $|0\rangle$ and $|1\rangle$ and by each of their 15 pairs of 1-Qbit corruptions.

The 2-fold degeneracy of the four $\mathbf{M}_i$ within each of these 16 subspaces is lifted by the operator

$$\mathbf{N} = \mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{Z}_3\mathbf{Z}_4, \tag{5.38}$$

which commutes with all the $\mathbf{M}_i$.

Since $|00000\rangle$ and $|11111\rangle$ are eigenstates of $\mathbf{N}$ with eigenvalues 1 and -1, and since $\mathbf{N}$ commutes with $\mathbf{Z}_i$, while anticommuting with $\mathbf{X}_i$ and $\mathbf{Y}_i$, it follows that

$$\mathbf{N}|\overline{0}\rangle = |\overline{0}\rangle,\ \mathbf{N}|\overline{1}\rangle = -|\overline{1}\rangle, \qquad \mathbf{N}\mathbf{Z}_i|\overline{0}\rangle = \mathbf{Z}_i|\overline{0}\rangle,\ \mathbf{N}\mathbf{Z}_i|\overline{1}\rangle = -\mathbf{Z}_i|\overline{1}\rangle,$$

$$\mathbf{N}\mathbf{X}_i|\overline{0}\rangle = -\mathbf{X}_i|\overline{0}\rangle,\ \mathbf{N}\mathbf{X}_i|\overline{1}\rangle = \mathbf{X}_i|\overline{1}\rangle, \qquad \mathbf{N}\mathbf{Y}_i|\overline{0}\rangle = -\mathbf{Y}_i|\overline{0}\rangle,\ \mathbf{N}\mathbf{Y}_i|\overline{1}\rangle = \mathbf{Y}_i|\overline{1}\rangle. \tag{5.39}$$

Consequently if one takes five Qbits in any state you like (perhaps most conveniently $|00000\rangle$) and measures the four $\mathbf{M}_i$ together with $\mathbf{N}$, one projects the Qbits into one of the 32 states

$$\begin{aligned} &|\overline{0}\rangle,\ \ \mathbf{X}_i|\overline{0}\rangle,\ \ \mathbf{Y}_i|\overline{0}\rangle,\ \ \mathbf{Z}_i|\overline{0}\rangle, \\ &|\overline{1}\rangle,\ \ \mathbf{X}_i|\overline{1}\rangle,\ \ \mathbf{Y}_i|\overline{1}\rangle,\ \ \mathbf{Z}_i|\overline{1}\rangle, \end{aligned} \tag{5.40}$$

and learns from the results of the measurement which it is.

Just as in the error correction procedure, if the state is not $|\overline{0}\rangle$ or $|\overline{1}\rangle$ we can restore it to either of these forms by applying the appropriate $\mathbf{X}_i$, $\mathbf{Y}_i$, or $\mathbf{Z}_i$.

If we wish to initialize the 5-Qbits to $|\overline{0}\rangle$ we can apply $\mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4$ should the measurement indicate that the error-corrected state is $|\overline{1}\rangle$.

This process of using a generalized measurement to produce 5-Qbits in the state $|\overline{0}\rangle$, is analogous to the procedure of using an ordinary measurement to produce a single Qbit in the state $|0\rangle$ described at the end of Part 1.E.

An alternative procedure for constructing 5-Qbit codewords by applying unitary gates is described there.

The 7-Qbit code

The 5-Qbit code is theoretically ideal but suffers from the problem that circuits performing the basic logical operations on the 5-Qbit code words are cumbersome.

The popular favorite seems to be a 7-Qbit code, devised by Andrew Steane, which permits extremely simple implementations of operations such as NOT or controlled NOT on the codewords.

The Steane code uses six mutually commuting operators to diagnose the error syndrome:

$$\begin{aligned}
\mathbf{M}_0 &= \mathbf{X}_0\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6, \quad & \mathbf{N}_0 &= \mathbf{Z}_0\mathbf{Z}_4\mathbf{Z}_5\mathbf{Z}_6, \\
\mathbf{M}_1 &= \mathbf{X}_1\mathbf{X}_3\mathbf{X}_5\mathbf{X}_6, \quad & \mathbf{N}_1 &= \mathbf{Z}_1\mathbf{Z}_3\mathbf{Z}_5\mathbf{Z}_6, \\
\mathbf{M}_2 &= \mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_6, \quad & \mathbf{N}_2 &= \mathbf{Z}_2\mathbf{Z}_3\mathbf{Z}_4\mathbf{Z}_6. \qquad (5.41)
\end{aligned}$$

The six operators in (5.41) clearly square to give the unit operator.

The $\mathbf{M}_i$ trivially commute among themselves as do the $\mathbf{N}_i$, and each $\mathbf{M}_i$ commutes with each $\mathbf{N}_j$, in spite of the anti-commutation of each $\mathbf{X}_k$ with the corresponding $\mathbf{Z}_k$, because in every case they share an even number of such pairs.

A circuits that measures the six operators (5.41) is shown in Figures 5.9.

Figure 5.9

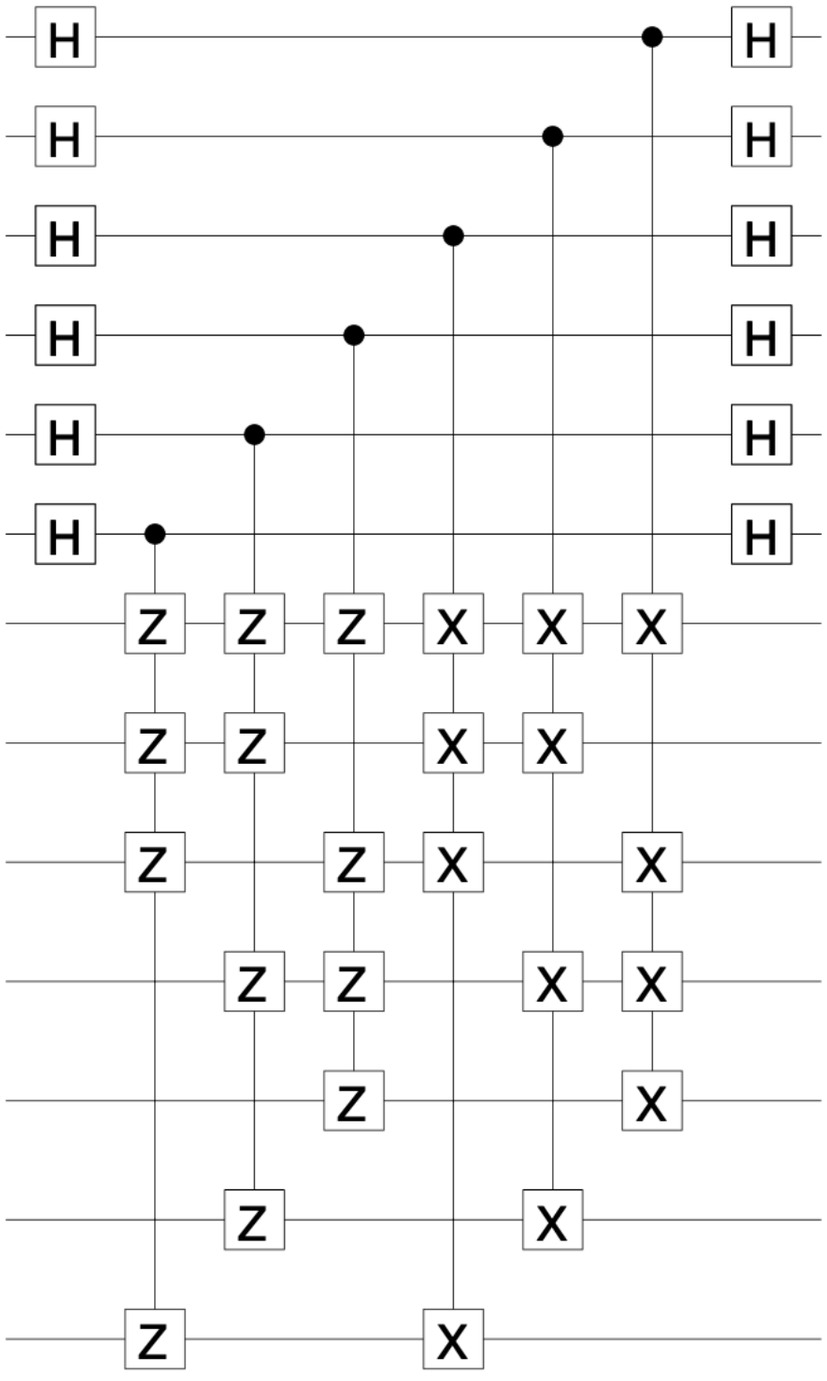


Figure 5.9. A circuit to measure the error syndrome for the 7-Qbit code. The seven Qbits are the seven lower wires. The six upper wires are the ancillas to be measured, resulting in a measurement of the six commuting operators $\mathbf{Z}_0\mathbf{Z}_4\mathbf{Z}_5\mathbf{Z}_6$, $\mathbf{Z}1\mathbf{Z}_3\mathbf{Z}_5\mathbf{Z}_6$, $\mathbf{Z}_2\mathbf{Z}_3\mathbf{Z}_4\mathbf{Z}_6$, $\mathbf{X}_0\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6$, $\mathbf{X}_1\mathbf{X}_3\mathbf{X}_5\mathbf{X}_6$, and $\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_6$ of (5.41).

The 7-Qbit code words are defined by

$$|\overline{0}\rangle = 2^{-3/2}(\mathbf{1}+\mathbf{M}_0)(\mathbf{1}+\mathbf{M}_1)(\mathbf{1}+\mathbf{M}_2)|0\rangle_7,$$

$$|\overline{1}\rangle = 2^{-3/2}(\mathbf{1}+\mathbf{M}_0)(\mathbf{1}+\mathbf{M}_1)(\mathbf{1}+\mathbf{M}_2)\overline{\mathbf{X}}|0\rangle_7, \tag{5.42}$$

where

$$\overline{\mathbf{X}} = \mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6, \tag{5.43}$$

so that

$$|1111111\rangle = \overline{\mathbf{X}}|0000000\rangle. \tag{5.44}$$

We again defer our discussion of how to produce these states until after our discussion of how they are used in error correction.

The two states in (5.42) are orthogonal, since each $\mathbf{M}$ flips four Qbits while $\overline{\mathbf{X}}$ flips all seven of them, so the first state is a superposition of 7-Qbit states with an odd number of 0's while the second is a superposition with an even number of 0's.

They are normalized to unity, for the same reason as given in the case of 5-Qbit code.

Since $\overline{\mathbf{X}}$ commutes with all the $\mathbf{M}_i$, a general superposition of the two code words can be written as

$$|\Psi\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle = \left(\alpha\mathbf{1} + \beta\overline{\mathbf{X}}\right)|\overline{0}\rangle, \tag{5.45}$$

and its corruption (5.18) assumes the form

$$|e\rangle|\Psi\rangle \rightarrow \left(|d\rangle\mathbf{1} + \sum_{i=1}^{7}\left[\,|a_i\rangle\mathbf{X}_i + |b_i\rangle\mathbf{Y}_i + |c_i\rangle\mathbf{Z}_i\right]\right)|\Psi\rangle. \qquad (5.46)$$

Because the $\mathbf{M}_i$ all commute and $\mathbf{M}_i(1 + \mathbf{M}_i) = (1 + \mathbf{M}_i)$, and because the $\mathbf{N}_j$ commute with the the $\mathbf{M}_i$ and with X and have $|0000000\rangle$ as an eigenstate with eigenvalue 1, it follows that $|\overline{0}\rangle, |\overline{1}\rangle$, and the general superposition (5.45) are eigenstate of each of the $\mathbf{M}_i$ and $\mathbf{N}_i$ with eigenvalue 1.

The 21 possible corruptions of (5.45) appearing in (5.46) are also eigenstates, distinguished by the possible sets of eigenvalues ±1 that the three $\mathbf{M}_i$ and three $\mathbf{N}_i$ can have.

As in the 5-Qbit case, this is because each $\mathbf{X}_i$, $\mathbf{Y}_i$, and $\mathbf{Z}_i$ commutes or anticommutes with each of the $\mathbf{M}_i$ and $\mathbf{N}_i$, so each state appearing in (5.46) is indeed an eigenstate of each $\mathbf{M}_i$ and $\mathbf{N}_i$ with eigenvalue 1 or −1.

To see why the results of the six measurements of the $\mathbf{M}_i$ and $\mathbf{N}_i$ determine a unique one of the twenty-two terms in (5.46), look at the following tables, which indicate by a bullet (•) whether an $\mathbf{X}_i$ appears in each of the $\mathbf{M}_i$ and whether a $\mathbf{Z}_i$ appears in each of the $\mathbf{N}_i$:

	$\mathbf{X}_0$	$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{X}_3$	$\mathbf{X}_4$	$\mathbf{X}_5$	$\mathbf{X}_6$
$\mathbf{M}_0$	•				•	•	•
$\mathbf{M}_1$		•		•		•	•
$\mathbf{M}_2$			•	•	•		•

(5.47)

	$\mathbf{Z}_0$	$\mathbf{Z}_1$	$\mathbf{Z}_2$	$\mathbf{Z}_3$	$\mathbf{Z}_4$	$\mathbf{Z}_5$	$\mathbf{Z}_6$
$\mathbf{N}_0$	•				•	•	•
$\mathbf{N}_1$		•		•		•	•
$\mathbf{N}_2$			•	•	•		•

(5.48)

Each $\mathbf{M}_i$ commutes with every $\mathbf{X}_j$; it anticommutes with $\mathbf{Y}_j$ and $\mathbf{Z}_j$ if a bullet appears in the column associated with $\mathbf{X}_j$ and commutes if there is no bullet; each $\mathbf{N}_i$ commutes with every $\mathbf{Z}_j$; it anticommutes with $\mathbf{X}_j$ and $\mathbf{Y}_j$ if a bullet appears in the column associated with $\mathbf{Z}_j$ and commutes if there is no bullet.

The signature of an $\mathbf{X}_i$ error (or no error) is that all three $\mathbf{M}_i$ measurements give +1.

The pattern of −1's in the $\mathbf{N}_i$ measurements then determines which of the 7 possible $\mathbf{X}_i$ characterize the error. (If all three $\mathbf{N}_i$ measurements also give +1 there is no error.)

In the same way, the signature of an $\mathbf{Z}_i$ error (or no error) is that all three $\mathbf{N}_i$ measurements give +1 and then the pattern of −1's in the Ni measurements determine which of the 7 possible $\mathbf{Z}_i$ characterize the error.

Finally, the signature of a $\mathbf{Y}_i$ error is that at least some of both the $\mathbf{M}_i$ and $\mathbf{N}_i$ measurements give−1.

The resulting pattern of−1's (which will be the same for both the $\mathbf{M}_i$ and $\mathbf{N}_i$ measurements) then determines which of the 7 possible $\mathbf{Y}_i$ characterize the error.

So the six measurements project the corrupted state into a unique one of the twenty-two terms in (5.46) and establish which term it is.

One can then undo the corruption by applying the appropriate one of the twenty-two operators $\mathbf{1}$, $\mathbf{X}_0$, ... $\mathbf{Z}_6$.

To produce the 7-Qbit code states one cannot immediately extend the method we used above to produce the 5-Qbit code states, because the two 7-Qbit code words and their 21 1-Qbit corruptions constitute only 44 mutually orthogonal states, while the space of seven Qbits has dimension 27 = 128.

One can, however, provide the missing 84 dimensions by noting the following:

The $2 \times 7 \times 6 = 84$ states given by

$$\mathbf{X}_i \mathbf{Z}_j |\overline{0}\rangle \quad \text{and} \quad \mathbf{X}_i \mathbf{Z}_j |\overline{1}\rangle, \quad i \neq j, \tag{5.49}$$

are also easily verified to be eigenstates of all the $\mathbf{M}_i$ and $\mathbf{N}_i$.

These states can be associated with a subset of all possible 2-Qbit corruptions, but this is not pertinent to the use to which we put them here.

As with the 1-Qbit $\mathbf{Y}_i$ corruptions, with these states at least some of both the $\mathbf{M}_i$ and $\mathbf{N}_i$ measurements give -1, but unlike the $\mathbf{Y}_i$ corruptions, the resulting pattern of -1's will not be the same for both the $\mathbf{M}_i$ and $\mathbf{N}_i$ measurements, since $i \neq j$.

Each of the $7 \times 6 = 42$ possibilities for $\mathbf{X}_i \mathbf{Z}_j$ clearly leads to its own characteristic pattern of $+1$ and -1 eigenvalues.

This gets us back to the situation we encountered in the 5-Qbit case.

By measuring the seven mutually commuting operators $\mathbf{M}_i$, $\mathbf{N}_i$, and

$$\overline{\mathbf{Z}} = \mathbf{Z}_0 \mathbf{Z}_1 \mathbf{Z}_2 \mathbf{Z}_3 \mathbf{Z}_4 \mathbf{Z}_5 \mathbf{Z}_6, \tag{5.50}$$

we can produce from seven Qbits in an arbitrarily chosen state a unique one of the 128 mutually orthogonal states given by $|\overline{0}\rangle$, $|\overline{1}\rangle$, their 42 different 1-Qbit corruptions, and their 84 different special kinds of 2-Qbit corruptions.

The results of the measurement tell us the character (if any) of the corruption, from which we know what operators ($\mathbf{X}_i$, $\mathbf{Y}_i$, $\mathbf{Z}_i$, or $\mathbf{X}_i\mathbf{Z}_j$, possibly combined with $\mathbf{X}$) we must apply to the post-measurement state to convert it into $|\overline{0}\rangle$.

This method can be simplified by a judicious choice of the initial state.

Suppose we start with seven Qbits in the standard initial state $|0\rangle_7$.

If one then measures $\mathbf{M}_0$, $\mathbf{M}_1$ and $\mathbf{M}_2$, the resulting state of the Qbits will be one of the 8 states

$$2^{-3/2}(\mathbf{1} \pm \mathbf{M}_0)(\mathbf{1} \pm \mathbf{M}_1)(\mathbf{1} \pm \mathbf{M}_2)|0\rangle_7, \tag{5.51}$$

with the specific pattern of + and − signs being revealed by the measurement.

The table in (5.47) permits one to choose a unique $\mathbf{Z}_i$ that commutes or anticommutes with each $\mathbf{M}_i$ depending on whether it appears in (5.51) with a + or a − sign.

Since $\mathbf{Z}_i|0\rangle_7 = |0\rangle_7$, acting on the seven Qbits with that particular $\mathbf{Z}_i$ does indeed convert their state to

$$2^{-3/2}(\mathbf{1} + \mathbf{M}_0)(\mathbf{1} + \mathbf{M}_1)(\mathbf{1} + \mathbf{M}_2)|0\rangle_7 = |\overline{0}\rangle. \tag{5.52}$$

In Part 5.E we shall examine a surprisingly simple circuit that encodes a general 1-Qbit state into a 7-Qbit codeword state in the manner of Figure 5.1, without using any measurement gates.

The virtue of the 7-Qbit code, that makes it preferable to the 5-Qbit code in spite of its greater expenditure of Qbits, is that many of the fundamental one- and 2-Qbit gates are trivially extended to 7- and 14-Qbit gates acting on the code words.

Because, for example, $\overline{\mathbf{X}}$ commutes with the $\mathbf{M}_i$ and flips all seven Qbits, it implements the logical NOT on the codewords (5.42):

$$\overline{\mathbf{X}}|\overline{0}\rangle = |\overline{1}\rangle, \quad \overline{\mathbf{X}}|\overline{1}\rangle = |\overline{0}\rangle. \tag{5.53}$$

Similarly, $\overline{\mathbf{Z}}$ commutes with the **Mi**, anticommutes with $\overline{\mathbf{X}}$, and leaves $|0\rangle 7$ invariant, so it implements the logical **Z** on the codewords:

$$\overline{\mathbf{Z}}|\overline{0}\rangle = |\overline{0}\rangle, \quad \overline{\mathbf{Z}}|\overline{1}\rangle = -|\overline{1}\rangle. \tag{5.54}$$

This much works equally well for the 5-Qbit code.

More remarkably, for the 7-Qbit code the bitwise Hadamard transformation,

$$\overline{\mathbf{H}} = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{H}_3\mathbf{H}_4\mathbf{H}_5\mathbf{H}_6, \tag{5.55}$$

also implements the logical Hadamard transformation on the codewords:

$$\overline{\mathbf{H}}|\overline{0}\rangle = \tfrac{1}{\sqrt{2}}\big(|\overline{0}\rangle + |\overline{1}\rangle\big), \quad \overline{\mathbf{H}}|\overline{1}\rangle = \tfrac{1}{\sqrt{2}}\big(|\overline{0}\rangle - |\overline{1}\rangle\big). \tag{5.56}$$

The relation analogous to (5.56) is not valid for the 5-Qbit code.

To see this note that two normalized states $|\phi\rangle$ and $|\psi\rangle$ are identical if and only if their inner product is 1.

For one can always express $|\psi\rangle$ in the form $|\psi\rangle = \alpha|\phi\rangle + \beta|\chi\rangle$ where $|\chi\rangle$ is orthogonal to $|\phi\rangle$ and $|\alpha|^2 + |\beta|^2 = 1$.

We then have $\langle\phi|\psi\rangle = \alpha$, so if $\langle\phi|\psi\rangle = 1$, then $\alpha = 1$, $\beta = 0$ and indeed $|\phi\rangle = |\psi\rangle$.

Since $|\bar{0}\rangle$ and $|\bar{1}\rangle$ are normalized and orthogonal and since $\overline{\mathbf{H}}$ is unitary and therefore preserves the normalization of $|\bar{0}\rangle$ and $|\bar{1}\rangle$, the four states appearing in the two equalities in (5.56) are all normalized.

Therefore to establish those equalities it suffices to show that

$$1 = \tfrac{1}{\sqrt{2}}\big(\langle\bar{0}|\mathbf{H}|\bar{0}\rangle + \langle\bar{0}|\mathbf{H}|\bar{1}\rangle\big), \quad 1 = \tfrac{1}{\sqrt{2}}\big(\langle\bar{1}|\mathbf{H}|\bar{0}\rangle - \langle\bar{1}|\mathbf{H}|\bar{1}\rangle\big). \tag{5.57}$$

This in turn would follow if we could show that the matrix of the encoded Hadamard in the encoded states is the same as the matrix of the 1-Qbit Hadamard in the 1-Qbit states:

$$\langle\bar{0}|\overline{\mathbf{H}}|\bar{0}\rangle = \langle\bar{0}|\overline{\mathbf{H}}|\bar{1}\rangle = \langle\bar{1}|\overline{\mathbf{H}}|\bar{0}\rangle = \tfrac{1}{\sqrt{2}}, \quad \langle\bar{1}|\overline{\mathbf{H}}|\bar{1}\rangle = -\tfrac{1}{\sqrt{2}}. \tag{5.58}$$

To establish (5.58) note that it follows from the definition (5.42) of the codewords $|\bar{0}\rangle$ and $|\bar{1}\rangle$ that the four matrix elements appearing in (5.58) are

$$\langle\bar{x}|\overline{\mathbf{H}}|\bar{y}\rangle = 2^{-3}\,{}_7\langle 0|\overline{\mathbf{X}}^x(\mathbf{1}+\mathbf{M}_0)(\mathbf{1}+\mathbf{M}_1)(\mathbf{1}+\mathbf{M}_2)\overline{\mathbf{H}}(\mathbf{1}+\mathbf{M}_0)(\mathbf{1}+\mathbf{M}_1)(\mathbf{1}+\mathbf{M}_2)\overline{\mathbf{X}}^y|0\rangle_7 \quad (5.59).$$

Since $\mathbf{HX} = \mathbf{ZH}$ and $\mathbf{XH} = \mathbf{HZ}$, and since each $\mathbf{N}_i$ differs from $\mathbf{M}_i$ only by the replacement of each $\mathbf{X}$ by the corresponding $\mathbf{Z}$, it follows that

$$\overline{\mathbf{H}}\,\mathbf{M}_i = \mathbf{N}_i\,\overline{\mathbf{H}}, \quad \mathbf{M}_i\,\overline{\mathbf{H}} = \overline{\mathbf{H}}\,\mathbf{N}_i. \tag{5.60}$$

So we can bring all three remaining terms 1 + **M**i in (5.59) on the right of H over to the left if we replace each by 1 + **N**i.

But since the M's and N's all commute we can then bring all three terms 1 + **M**i on the left of $\overline{\mathbf{H}}$ over to the right if we again replace each by
1 + **N**i.

The effect of these interchanges is simply to change all the M' s in (5.59) to N's:

$$\langle\overline{x}|\overline{\mathbf{H}}|\overline{y}\rangle = 2^{-3}\,{}_7\langle 0|\overline{\mathbf{X}}^x(\mathbf{1}+\mathbf{N}_0)(\mathbf{1}+\mathbf{N}_1)(\mathbf{1}+\mathbf{N}_2)\overline{\mathbf{H}}(\mathbf{1}+\mathbf{N}_0)(\mathbf{1}+\mathbf{N}_1)(\mathbf{1}+\mathbf{N}_2)\overline{\mathbf{X}}^y|0\rangle_7 \quad (5.61).$$

Since each **N**i commutes with $\overline{\mathbf{X}}$ (there are four anticommutations) we have

$$\langle\overline{x}|\overline{\mathbf{H}}|\overline{y}\rangle = 2^{-3}\,{}_7\langle 0|(\mathbf{1}+\mathbf{N}_0)(\mathbf{1}+\mathbf{N}_1)(\mathbf{1}+\mathbf{N}_2)\overline{\mathbf{X}}^x\,\overline{\mathbf{H}}\,\overline{\mathbf{X}}^y(\mathbf{1}+\mathbf{N}_0)(\mathbf{1}+\mathbf{N}_1)(\mathbf{1}+\mathbf{N}_2)|0\rangle_7 \quad (5.62),$$

but since each **N**i acts as the identity on $|0\rangle_7$ each of the six **1** + **N**i can be replaced by a factor of 2, reducing (5.62) simply to

$$\langle\overline{x}|\overline{\mathbf{H}}|\overline{y}\rangle = 2^3\,{}_7\langle 0|\overline{\mathbf{X}}^x\,\overline{\mathbf{H}}\,\overline{\mathbf{X}}^y|0\rangle_7 \tag{5.63}.$$

Since $\overline{\mathbf{X}}$, $\overline{\mathbf{H}}$, and $|0\rangle_7$ are just tensor products of the seven 1-Qbit quantities **X**, **H**, and $|0\rangle$, (5.63) is just

$$\langle\overline{x}|\overline{\mathbf{H}}|\overline{y}\rangle = 2^3\,\langle x|\mathbf{H}|y\rangle^7. \tag{5.64}$$

But since

$$\langle 0|\mathbf{H}|0\rangle = \langle 0|\mathbf{H}|1\rangle = \langle 1|\mathbf{H}|0\rangle = \tfrac{1}{\sqrt{2}}, \quad \langle 1|\mathbf{H}|1\rangle = -\tfrac{1}{\sqrt{2}}, \tag{5.65}$$

(5.64) does indeed reduce to (5.59), establishing that $\overline{\mathbf{H}} = \mathbf{H}^{\otimes 7}$ does indeed act as a logical Hadmard gate on the codewords.

An alternative circuit-theoretic demonstration that $\mathbf{H}^{\otimes 7}$ acts as the logical Hadamard on the codewords is given in Part 5.E.

Nor is it difficult to make a 14-Qbit logical cNOT gate, that takes the pair of codewords $|\overline{x}\rangle|\overline{y}\rangle$ into $|\overline{x}\rangle|\overline{x \oplus y}\rangle$. One simply applies ordinary cNOT gates to each of the 7 pairs of corresponding Qbits in the two code words. This works because each of the codewords in (5.42) is left invariant by each of the $\mathbf{M}_i$. If the *control* codeword is in the state $|\overline{0}\rangle$ then the pattern of flips applied to the *target* codeword for each of the eight terms in the expansion of the control codeword

$$|\overline{0}\rangle = 2^{-3/2}\Big(\mathbf{1} + \mathbf{M}_0 + \mathbf{M}_1 + \mathbf{M}_2 + \mathbf{M}_1\mathbf{M}_2 + \mathbf{M}_2\mathbf{M}_0 + \mathbf{M}_0\mathbf{M}_1 + \mathbf{M}_0\mathbf{M}_1\mathbf{M}_2\Big)|0\rangle_7, \tag{5.66}$$

is simply given by the corresponding product of $\mathbf{M}_i$. Since each $\mathbf{M}_i$ acts as the identity on both $|\overline{0}\rangle$ and $|\overline{1}\rangle$, the target codeword is unchanged. On the other hand if the control codeword is in the state $|\overline{1}\rangle$ then the pattern of flips applied to the target codeword differs from this by an additional application of $\overline{\mathbf{X}}$ which has precisely the effect of interchanging $|\overline{0}\rangle$ and $|\overline{1}\rangle$.

Because of the simplicity of all these encoded gates, one can use error-correction to eliminate malfunctions of the elementary gates themselves, provided the rate of malfunctioning is so low that only a single one of the seven elementary gates is likely to malfunction. In the case of the 1-Qbit encoded gates, their elementary components act only on single Qbits in the codeword, so if only a single one of them malfunctions then only a single Qbit in the codeword will be corrupted and the error-correction procedure described above will restore the correct output. But this also works for the encoded cNOT gate, since if only a single one of the elementary 2-Qbit cNOT gates malfunctions, this will affect only single Qbits in each of the two encoded 7-Qbit words, and the correct output will again be restored by applying error correction to both of the codewords.

Another virtue of codeword gates that can be constructed a simple tensor products of the uncoded gates is that they cannot (when functioning correctly) convert single-Qbit errors to multiple-Qbit errors, as more more complex constructions of codeword gates might do. This highly desirable property is called *fault tolerance*. The great virtue of the 7-Qbit code is that many of the most important logical gates can be implemented in a fault tolerant way.

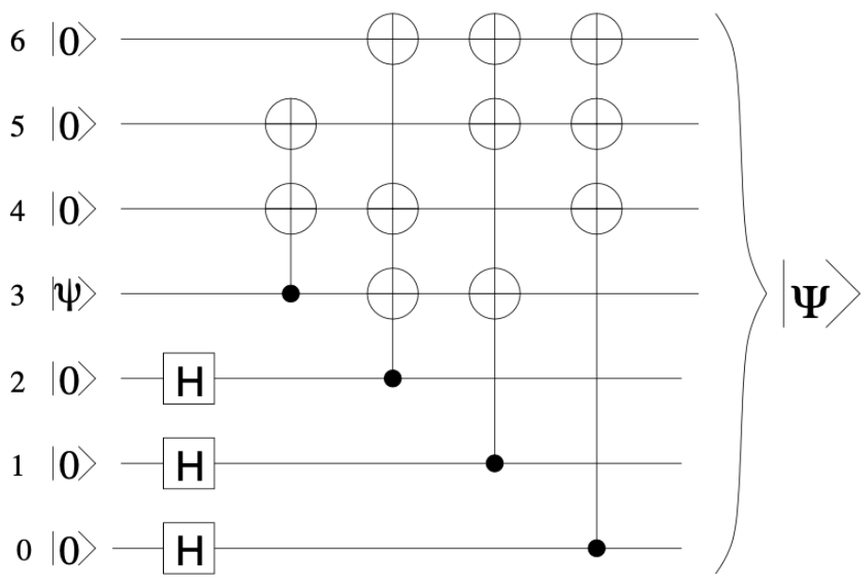
5.E. Circuits that make the 7- and 5-Qbit codewords.

The circuit in Figure 5.10 encodes a general 1-Qbit state into a 7-Qbit codeword without using any measurement gates, in a manner analogous to the way Figure 5.1 produces 3-Qbit codewords. A full explanation of how the circuit works is given in the figure caption, but the basic idea is quite simple: each of the three controlled triple-NOT gates acts in combination with the Hadamard on its control Qbit to produce one of the operators $\mathbf{1} + \mathbf{M}_i$ in (5.52). And the controlled double-NOT gate, which acts only if the state of its control-Qbit is $|1\rangle$, has precisely the effect of converting the 7-Qbit output from $|\overline{0}\rangle$ to $|\overline{1}\rangle$, when the 1-Qbit input $|\psi\rangle$ changes from $|0\rangle$ to $|1\rangle$.

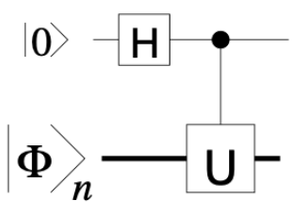
There is also a less direct method to confirm that Figure 5.10 produces the 7-Qbit encoding, which can be applied equally well to establishing the validity of the 5-Qbit encoding circuit we examine next, for which I know no version of the simple direct argument. The indirect argument starts with the observation (already noted when we discussed how to construct 7-Qbit codewords by extending the error-syndrome measurement to include $\overline{\mathbf{Z}}$) that the seven mutually commuting operators $\mathbf{M}_i$, $\mathbf{N}_i$, $(i = 0, 1, 2)$ and $\overline{\mathbf{Z}}$, each with eigenvalues ± 1, have a set of 2^7 non-degenerate eigenvectors that form an orthonormal basis for the entire 7-dimensional codeword space. In particular the two codeword states $|\overline{0}\rangle$ and $|\overline{1}\rangle$ are the unique eigenstates of all the $\mathbf{M}_i$ and $\mathbf{N}_i$ with eigenvalues 1, and of $\overline{\mathbf{Z}}$ with eigenvalues 1 and -1, respectively.

It follows from this that if a circuit produces a state $|\Psi\rangle$ that is invariant under all the $\mathbf{M}_i$ and $\mathbf{N}_i$ then $|\Psi\rangle$ must be a superposition of the codeword states $|\overline{0}\rangle$ and $|\overline{1}\rangle$, and if $|\Psi\rangle$ is additionally an eigenstate of $\overline{\mathbf{Z}}$ then to within factors $e^{i\varphi}$ of modulus 1, $|\Psi\rangle$ must be $|\overline{0}\rangle$ or $|\overline{1}\rangle$ depending on whether the eigenvalue is 1 or -1.

Figure 5.10



(a)



(b)

Figure 5.10. A 7-Qbit encoding circuit (a) that takes $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ into the corresponding superposition of the two 7-Qbit code words given in (5.42), $|\Psi\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$. Since the circuit is unitary and therefore linear, it is enough to show that it works when $|\psi\rangle = |0\rangle$ and when $|\psi\rangle = |1\rangle$. The numbering of the Qbits from 6 to 0 is made explicit to facilitate comparison with the form (5.42) of the code words. When $|\psi\rangle = |0\rangle$ the controlled double-NOT on the left acts as the identity. Note that a circuit of the form (b), acting on a state of the form $|0\rangle \otimes |\Phi\rangle_n$ produces the state $\frac{1}{\sqrt{2}}(\mathbf{1} + \mathbf{X} \otimes \mathbf{U})|0\rangle \otimes |\Phi\rangle_n$. If this is applied to the three controlled triple-NOT gates in (a) then, reading from left to right, the resulting operations are $\frac{1}{\sqrt{2}}(\mathbf{1} + \mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_6) = \frac{1}{\sqrt{2}}(\mathbf{1} + \mathbf{M}_2)$, $\frac{1}{\sqrt{2}}(\mathbf{1} + \mathbf{X}_1\mathbf{X}_3\mathbf{X}_5\mathbf{X}_6) = \frac{1}{\sqrt{2}}(\mathbf{1} + \mathbf{M}_1)$, and $\frac{1}{\sqrt{2}}(\mathbf{1} + \mathbf{X}_0\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6) = \frac{1}{\sqrt{2}}(\mathbf{1} + \mathbf{M}_0)$. Thus the circuit does indeed produced the codeword $|\overline{0}\rangle$ in (5.42). When $|\psi\rangle = |1\rangle$, the controlled double-NOT on the left acts as $\mathbf{X}_4\mathbf{X}_5$. The circuit after that action is exactly the same as when $|\psi\rangle = |0\rangle$, except that the initial state on the left of Qbits #3, #4, and #5 is $|1\rangle$ rather than $|0\rangle$. Since all $\mathbf{X}_i$ commute, the state that results is not $|\overline{0}\rangle$ but $\mathbf{X}_3\mathbf{X}_4\mathbf{X}_5|\overline{0}\rangle$. But

$$\mathbf{X}_3\mathbf{X}_4\mathbf{X}_5 = \mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6\mathbf{M}_0\mathbf{M}_1\mathbf{M}_2 = \overline{\mathbf{X}}\mathbf{M}_0\mathbf{M}_1\mathbf{M}_2.$$

Since $\mathbf{M}_0\mathbf{M}_1\mathbf{M}_2$ acts as the identity on $|\overline{0}\rangle$, the resulting state is indeed

$$|\overline{1}\rangle = \overline{\mathbf{X}}|\overline{0}\rangle.$$

Figure 5.11 shows that the state $|\Psi\rangle$ produced by the circuit in Figure 5.10 is indeed invariant under $\mathbf{M}_0 = \mathbf{X}_6\mathbf{X}_5\mathbf{X}_4\mathbf{X}_0$. The figure demonstrates that when $\mathbf{M}_0$ is brought to the left through all the gates in the circuit it acts directly as $\mathbf{Z}_0$ on the input state on the left, which is invariant under $\mathbf{Z}_0$. The caption explains why essentially the same argument applies to the other $\mathbf{M}_i$: when brought all the way to the left, $\mathbf{M}_1$ reduces to $\mathbf{Z}_1$ acting on the input state, and $\mathbf{M}_2$, to $\mathbf{Z}_2$. Figure 5.12 similarly establishes the invariance of $|\Psi\rangle$ under the three $\mathbf{N}_i$.

Figure 5.13 establishes that the effect of $\overline{\mathbf{Z}} = \mathbf{Z}_6\mathbf{Z}_5\mathbf{Z}_4\mathbf{Z}_3\mathbf{Z}_2\mathbf{Z}_1\mathbf{Z}_0$ acting on the right is the same as $\mathbf{Z}_6\mathbf{Z}_5\mathbf{Z}_4\mathbf{Z}_3$ acting on the left. But since $\mathbf{Z}_6$, $\mathbf{Z}_5$, and $\mathbf{Z}_4$ all act on the 1-Qbit states $|0\rangle$ this leaves only $\mathbf{Z}_3$ which converts $|\psi\rangle$ to $\mathbf{Z}|\psi\rangle$, which multiplies by $(-1)^x$ when $|\psi\rangle = |x\rangle$. This shows that, as required, $\overline{\mathbf{Z}}|\Psi\rangle = (-1)^x|\Psi\rangle$ when $|\psi\rangle = |x\rangle$. Similarly, Figure 5.14 establishes that the effect of $\overline{\mathbf{X}} = \mathbf{X}_6\mathbf{X}_5\mathbf{X}_4\mathbf{X}_3\mathbf{X}_2\mathbf{X}_1\mathbf{X}_0$ acting on the right is the same as $\mathbf{X}_3\mathbf{Z}_2\mathbf{Z}_1\mathbf{Z}_0$ acting on the left. But since $\mathbf{Z}_2$, $\mathbf{Z}_1$, and $\mathbf{Z}_0$ all act on the 1-Qbit states $|0\rangle$ this leaves only $\mathbf{X}_3$ which interchanges $|1\rangle$ and $|0\rangle$ when $|\psi\rangle = |x\rangle$. This shows that $\overline{\mathbf{X}}$ interchanges the corresponding states produced by the circuit. It also establishes that if $|\Psi\rangle$ differs by a phase factor $e^{i\varphi}$ from $|\overline{0}\rangle$ when $|\psi\rangle = |0\rangle$, then it will differ by the *same* phase factor from $|\overline{1}\rangle$ when $|\psi\rangle = |1\rangle$.

Figure 5.11

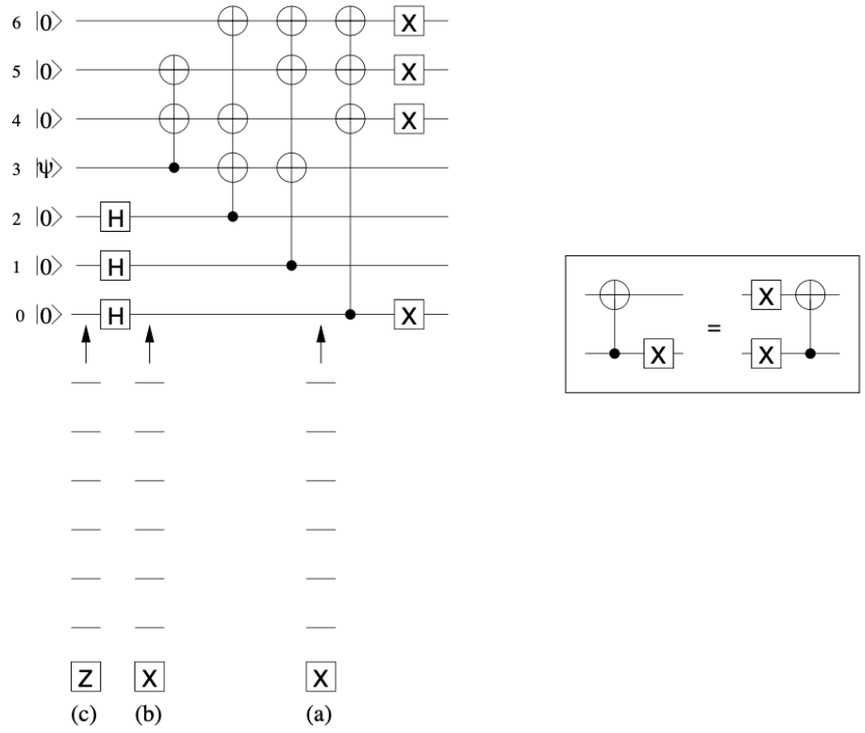


Figure 5.11. Seven-Qbit code. Demonstration that the state $|\Psi\rangle$ constructed by the circuit in Figure 5.10 is invariant under $\mathbf{M}_0 = \mathbf{X}_0\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6$. The circuit in the upper right is the same as in Figure 5.10 except that after the state $|\Psi\rangle$ has been produced **X** operations are applied to Qbits #0, #4, #5, and #6. We then exploit the easily verified fact, illustrated in the box on the right, that bringing an **X**, acting on the control Qbit of a cNOT, from one side of the cNOT to the other introduces an additional **X** acting on the target Qbit. (We also use the fact that an **X** acting on the target Qbit commutes with the cNOT.) In particular, bringing the **X** acting on Qbit #0 to the left of all three cNOT gates represented by the controlled triple-NOT on the right introduces **X** operators on all three target Qbits which combine with the three **X** already acting on those Qbits to produce unit operators. So all four **X** gates on the right reduce to $\mathbf{X}_0$ — a single **X** acting on Qbit #0 — as indicated in inset (a). The $\mathbf{X}_0$ can be moved further to the left until it encounters $\mathbf{H}_0$ as shown in inset (b), and it can then be moved to the left through $\mathbf{H}_0$ provided it is changed into $\mathbf{Z}_0$, as shown in inset (c). So the figure demonstrates that $\mathbf{M}_0$ acting to the right of all the gates of the circuit is equivalent to $\mathbf{Z}_0$ acting to the left of all the gates. But $\mathbf{Z}_0$ acts on the 1-Qbit state $|0\rangle$ which it leaves invariant. Consequently $|\Psi\rangle$ is invariant under $\mathbf{M}_0$. Essentially the same argument applies to $\mathbf{M}_1 = \mathbf{X}_1\mathbf{X}_3\mathbf{X}_5\mathbf{X}_6$, since the $\mathbf{X}_i$ all commute with the first controlled triple-NOT on the right, and then produce a single **X** on Qbit #1 when moved through the middle controlled triple-NOT, resulting in $\mathbf{Z}_1$ when moved the rest of the way to the left. And in the same way $\mathbf{M}_2 = \mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_6$ produces $\mathbf{Z}_2$ when moved all the way to the left.

Figure 5.12

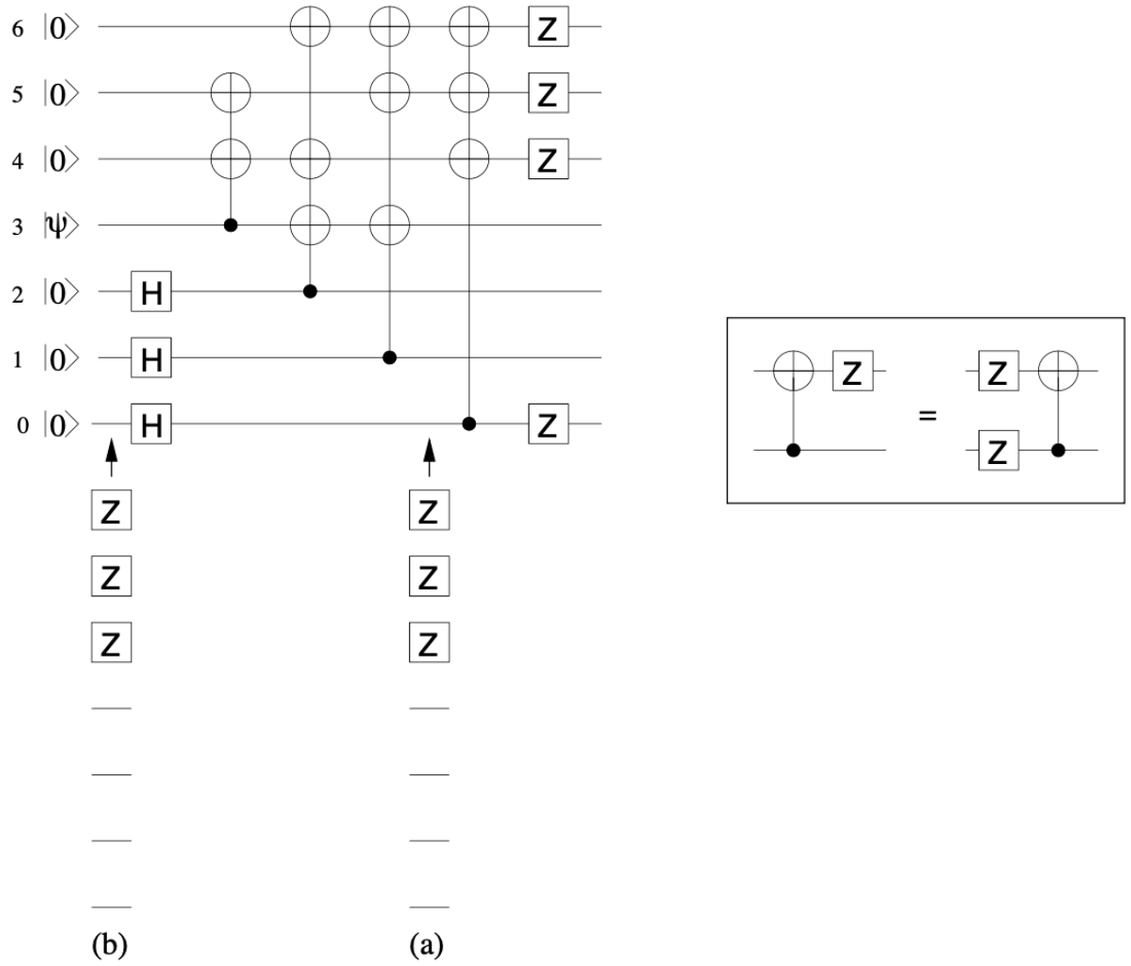

6 |0⟩
5 |0⟩
4 |0⟩
3 |ψ⟩
2 |0⟩
1 |0⟩
0 |0⟩
H
H
H
Z
Z
Z

Figure 5.13

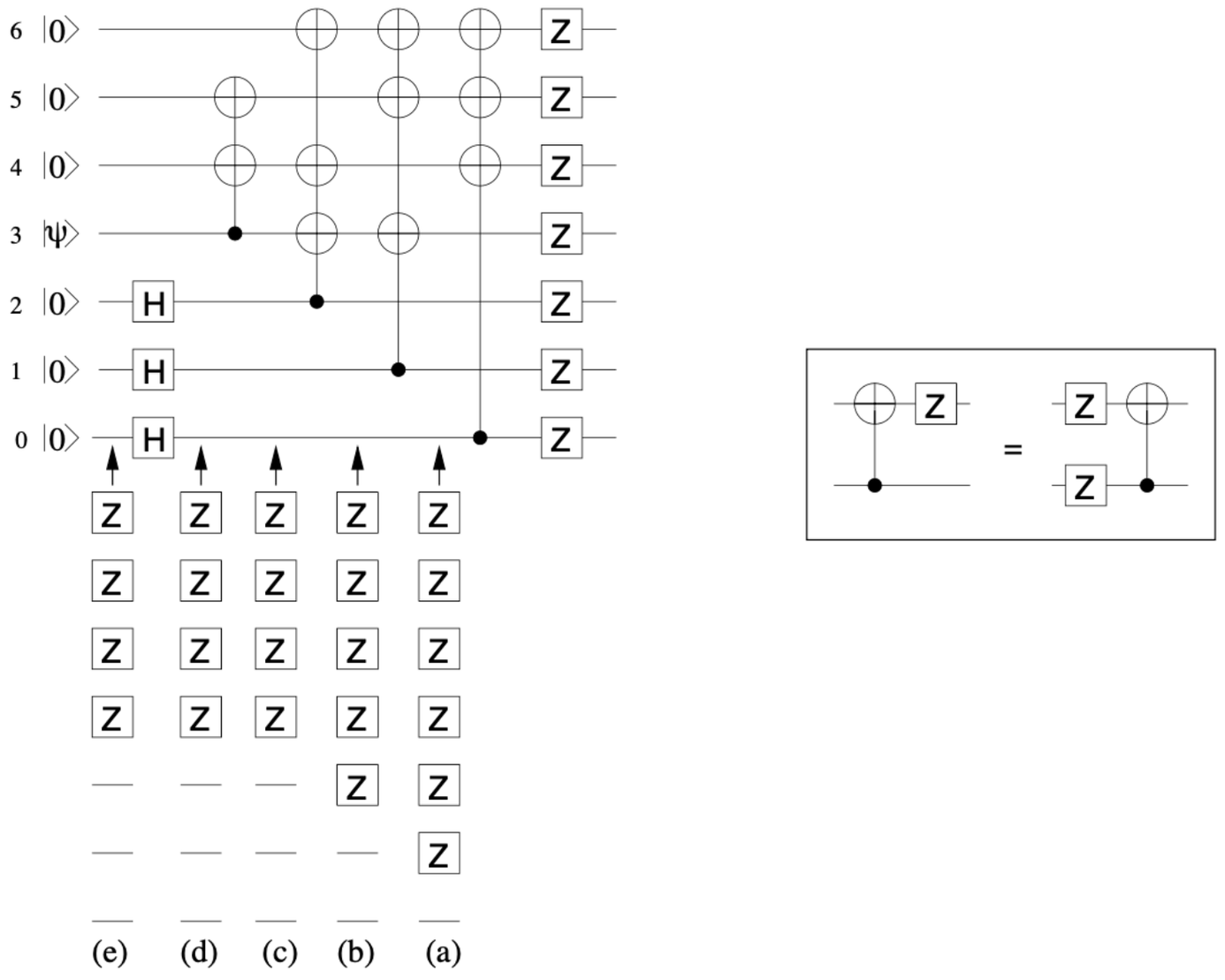


Figure 5.13. Seven-Qbit code. Demonstration that $\overline{\mathbf{Z}} = \mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{Z}_3\mathbf{Z}_4\mathbf{Z}_5\mathbf{Z}_6$ acting on the right of the circuit is the same as $\mathbf{Z}_3\mathbf{Z}_4\mathbf{Z}_5\mathbf{Z}_6$ acting on the left. Since $\mathbf{Z}_4$, $\mathbf{Z}_5$, and $\mathbf{Z}_6$ all act as the identity on the 1-Qbit states $|0\rangle$ this leaves only $\mathbf{Z}_3$ which converts $|\psi\rangle$ to $\mathbf{Z}|\psi\rangle$. This results in a factor of $(-1)^x$ when $|\psi\rangle = |x\rangle$, showing that $\overline{\mathbf{Z}}|\Psi\rangle = (-1)^x|\Psi\rangle$ when $|\psi\rangle = |\overline{x}\rangle$.

It remains to show that when $|\psi\rangle = |0\rangle$ in Figure 5.14, the resulting state is given by $|\overline{0}\rangle$ in (5.42) without any non-trivial phase factor $e^{i\varphi}$. Since $|0\rangle_7$ appears in the expansion of $|\overline{0}\rangle$ in (5.42) with the amplitude $1/2^{3/2}$, we must show that when the input to the circuit in 5.14 is $|0\rangle_7$ the inner product of the output with $|0\rangle_7$ is $1/2^{3/2}$, without any accompanying nontrivial $e^{i\varphi}$. This is established in a simple circuit theoretic manner in Figure 5.15, as explained in the caption.

Figure 5.16 gives an alternative circuit-theoretic proof that the logical Hadamard operation on codewords is indeed produced by the simple 7-fold tensor product $\mathbf{H}^{\otimes 7}$ of 1-Qbit hadamards.

The circuit in Figure 5.17 encodes a general 1-Qbit state into a 5-Qbit codeword without using any measurement gates. Unlike the superficially similar circuit for the 7-Qbit code in Figure 5.10, there does not seem to be a transparently simple argument for why the circuit produces the codewords.[1] But the analog of the alternative argument in Figures 5.10-5.15 does work, though in a somewhat more complicated way.

Figure 5.14

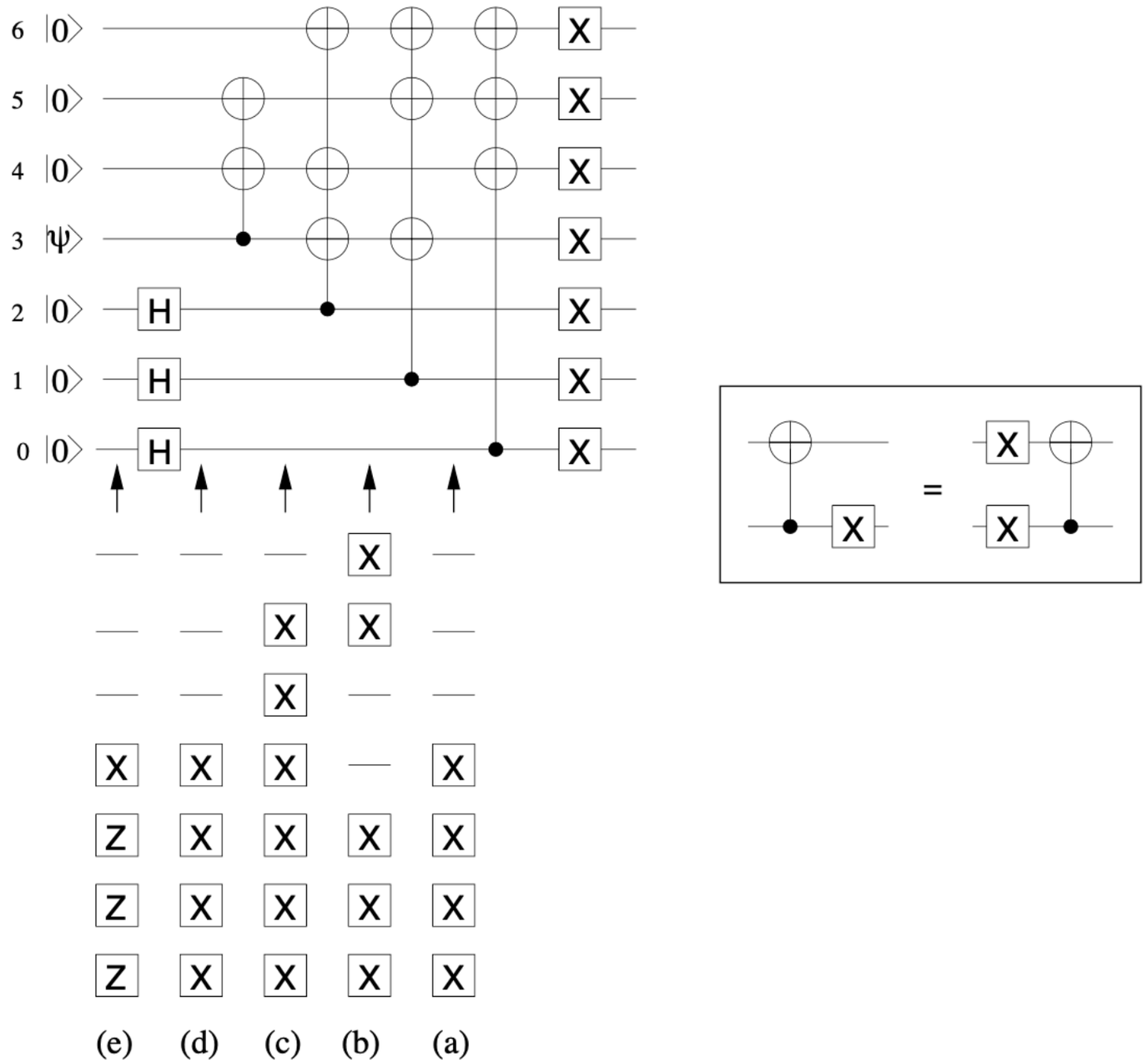


Figure 5.14. Seven-Qbit code. Demonstration that $\overline{\mathbf{X}} = \mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6$ acting on the right of the circuit is the same as $\mathbf{X}_3\mathbf{Z}_2\mathbf{Z}_1\mathbf{Z}_0$ acting on the left. Since $\mathbf{Z}_2$, $\mathbf{Z}_1$, and $\mathbf{Z}_0$ all act as the identity on the 1-Qbit states $|0\rangle$ this leaves only $\mathbf{X}_3$ which converts $|\psi\rangle$ to $\mathbf{X}|\psi\rangle$. When $|\psi\rangle = |x\rangle$ this interchanges $|0\rangle$ and $|1\rangle$, and therefore $\overline{\mathbf{X}}$ interchanges the corresponding states produced by the circuit.

Figure 5.15

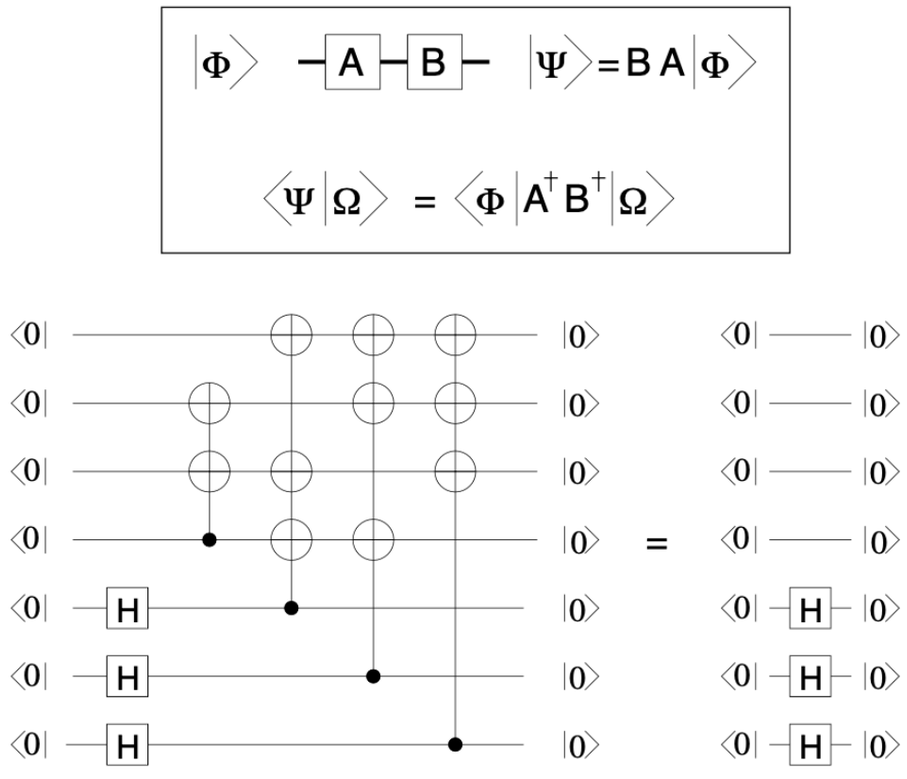


Figure 5.15. Seven-Qbit code. Demonstration that the state produced by the circuit in Figure 5.10 when $|\psi\rangle = |0\rangle$ has an inner product with the state $|0\rangle_7$ that is $1/2^{3/2}$, thereby establishing that the phase factor $e^{i\varphi} = 1$ — i.e. that the state is precisely $|\bar{1}\rangle$ without any additional phase factor. The box at the top establishes a simple circuit-theoretic way to evaluate inner products. The circuit diagram in the box shows an n-Qbit gate **A** acting on an n-Qbit input state $|\Phi\rangle$, followed by a second gate **B**. The output state is $|\Psi\rangle = \mathbf{BA}|\Phi\rangle$. (Remember that operators in a circuit diagram have the reverse order from the corresponding equation.) The inner product $\langle\Psi|\Omega\rangle$ of the output state with some other state $|\Omega\rangle$ is given by $\langle\Psi|\Omega\rangle = \langle\Phi|\mathbf{A}^\dagger\mathbf{B}^\dagger|\Omega\rangle$. This establishes the rule: to evaluate the inner product of the output state of a circuit diagram with another state $|\Omega\rangle$, replace all the gates in the circuit by their adjoints (i.e. by their inverses, since all gates are unitary) without changing their order in the diagram, let them all act to the *right* on $|\Omega\rangle$, and take the inner product of what results with the input state. In the case we are interested in the input and output states are the same, $|0\rangle_7$, and all the gates are identical to their inverses. So we simply sandwich the circuit of Figure 5.10 between $|0\rangle_7$ and ${}_7\langle 0|$, as shown on the lower left. But now all the cNOT gates have $|0\rangle$ for their control bits, and therefore act as the identity. The diagram simplifies to the form on the right, consisting of four inner products $\langle 0|0\rangle = 1$, and three matrix elements $\langle 0|\mathbf{H}|0\rangle = \frac{1}{\sqrt{2}}$. So the inner product is indeed $1/2^{3/2}$.

Figure 5.16

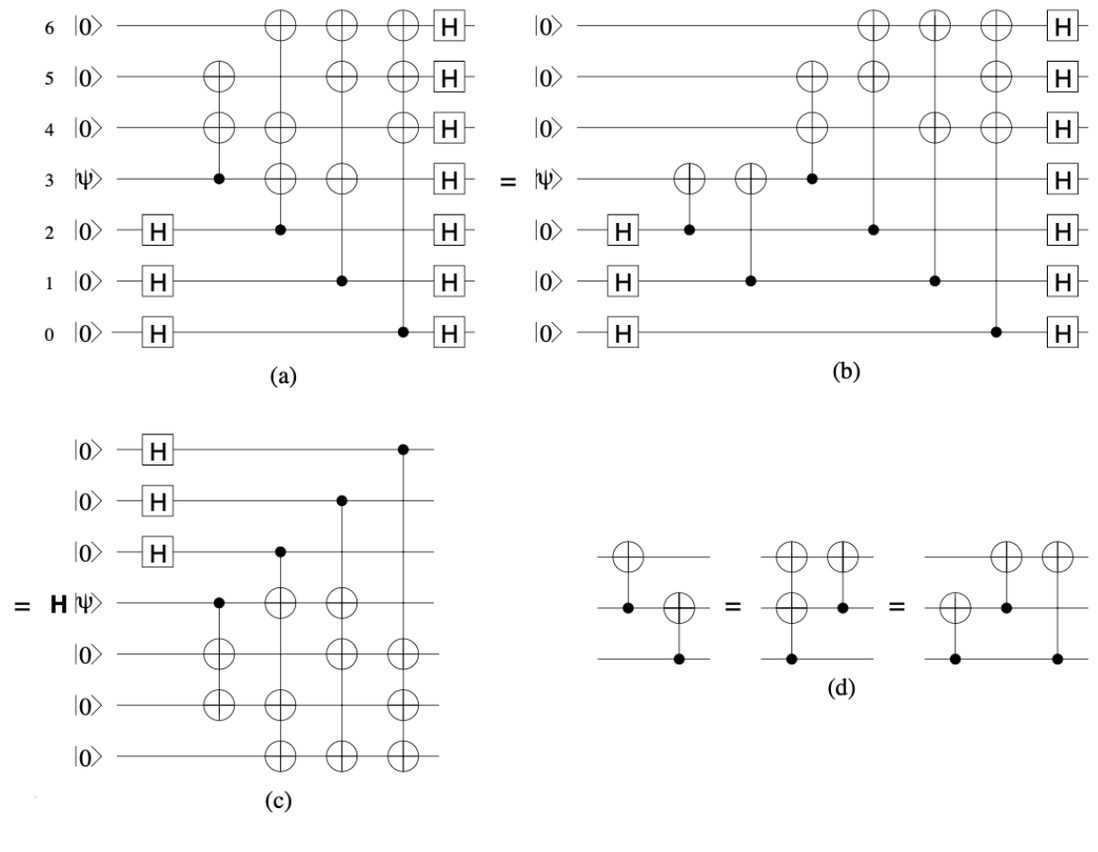


Figure 5.16. Seven-Qbit code. A circuit-theoretic demonstration that $\overline{\mathbf{H}} = \mathbf{H}_6 \otimes \cdots \otimes \mathbf{H}_0$ acts as a logical Hadamard on the 7-Qbit codewords. Part (a) shows $\overline{\mathbf{H}}$ acting on the 7-Qbit encoding of the 1-Qbit state $|\psi\rangle$ shown in Figure 5.10. In part (b) the two cNOT gates that target Qbit #3 have been moved to the left through the two cNOT gates that are controlled by Qbit #3, using four applications of the easily verified rule shown in part (d). In part (c) the seven 1-Qbit Hadamards in part (b) have been moved all the way to the left, applying to each of the eleven cNOT gates in (b) the rule that sandwiching a cNOT between Hadamards on both its control and target Qbits converts it to a cNOT controlled by the former target, and targetted on the former control. But the circuit in part (c) works just as well as the circuit in Figure 5.10 for producing the 7-Qbit encoding. Following the argument in the caption of Figure 5.10, one sees that the three controlled triple-NOT gates in (c) produce the operator $2^{-3/2}(\mathbf{1} + \mathbf{X}_4\mathbf{X}_3\mathbf{X}_1\mathbf{X}_0)(\mathbf{1} + \mathbf{X}_5\mathbf{X}_3\mathbf{X}_2\mathbf{X}_0)(\mathbf{1} + \mathbf{X}_6\mathbf{X}_2\mathbf{X}_1\mathbf{X}_0)$, but using the definitions (5.41) of the $\mathbf{M}_i$ one easily confirms by multiplying out the products that this operator is identical to $2^{-3/2}(\mathbf{1} + \mathbf{M}_0)(\mathbf{1} + \mathbf{M}_1)(\mathbf{1} + \mathbf{M}_2)$, so when Qbit #3 is in the state $|0\rangle$ one produces the correct codeword. When Qbit #3 is in the state $|1\rangle$ the effect of the controlled double-NOT is to produce the additional operator $\mathbf{X}_3\mathbf{X}_2\mathbf{X}_1$ acting on $|\overline{0}\rangle$. But $\mathbf{X}_3\mathbf{X}_2\mathbf{X}_1 = \overline{\mathbf{X}}\mathbf{M}_0$, and $\mathbf{M}_0$ leaves $|\overline{0}\rangle$ invariant. So the result is the state $|\overline{1}\rangle$. Therefore the state produced in (c) is indeed the 7-Qbit encoding of the 1-Qbit state $\mathbf{H}|\psi\rangle$.

Figure 5.17

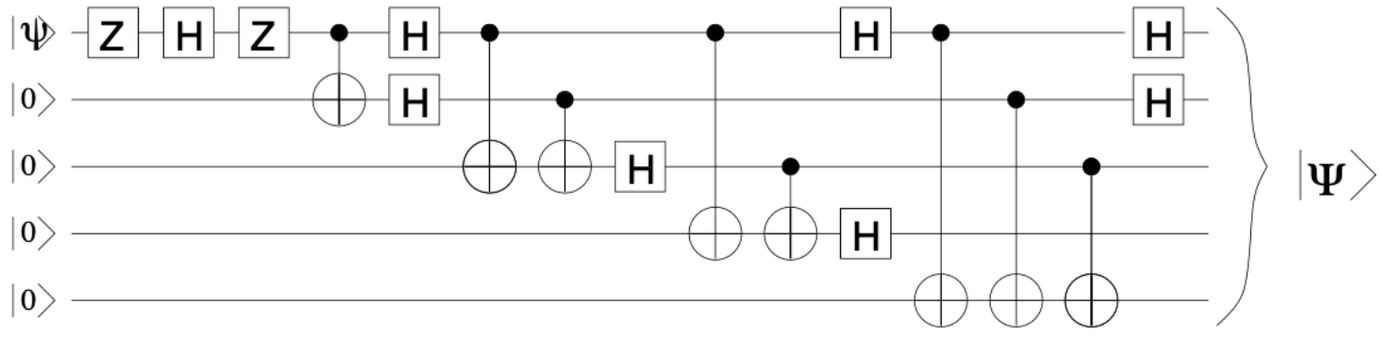


Figure 5.17. A 5-Qbit encoding encoding circuit that takes $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ into the corresponding superposition of the two 5-Qbit code words given in (5.33), $|\Psi\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$. I know no simple direct way to see from the diagram that the circuit works as claimed, like the argument for the 7-Qbit encoding circuit in the caption of Figure 5.10. But an indirect argument like the one given for the 7-Qbit code in Figures 5.11-5.15, is given in Figures 5.19-5.23.

Figure 5.18 shows that $\mathbf{M}_0 = \mathbf{Z}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Z}_4$ leaves both codewords invariant, by demonstrating that it can be brought to the left through all the gates in the circuit to act on the input state $|x0000\rangle$ as $\mathbf{Z}_2$. Figure 5.19 shows the same for $\mathbf{M}_1 = \mathbf{Z}_2\mathbf{X}_3\mathbf{X}_4\mathbf{Z}_0$, $\mathbf{M}_2 = \mathbf{Z}_3\mathbf{X}_4\mathbf{X}_0\mathbf{Z}_1$, and $\mathbf{M}_3 = \mathbf{Z}_4\mathbf{X}_0\mathbf{X}_1\mathbf{Z}_2$, which can be brought to the left through all the gates to act on the input state as $\mathbf{Z}_0$, $\mathbf{Z}_3$, and $\mathbf{Z}_1$. The successive transformations of the $\mathbf{M}_i$ as they are brought to the left exploit the boxed identities in Figures 5.11 and 5.12, that bringing an $\mathbf{X}$ through a cNOT along the control Qbit produces an additional $\mathbf{X}$ on the target Qbit, while bringing a $\mathbf{Z}$ through a cNOT along the target Qbit produces an additional $\mathbf{Z}$ on the control. (The transformations also exploit the fact that an $\mathbf{X}$ on the target Qbit commutes with cNOT as does a $\mathbf{Z}$ on the control Qbit, and that bringing an $\mathbf{X}$ through a $\mathbf{H}$ changes it to a $\mathbf{Z}$ and vice versa.)

All All All All All

Figure 5.18

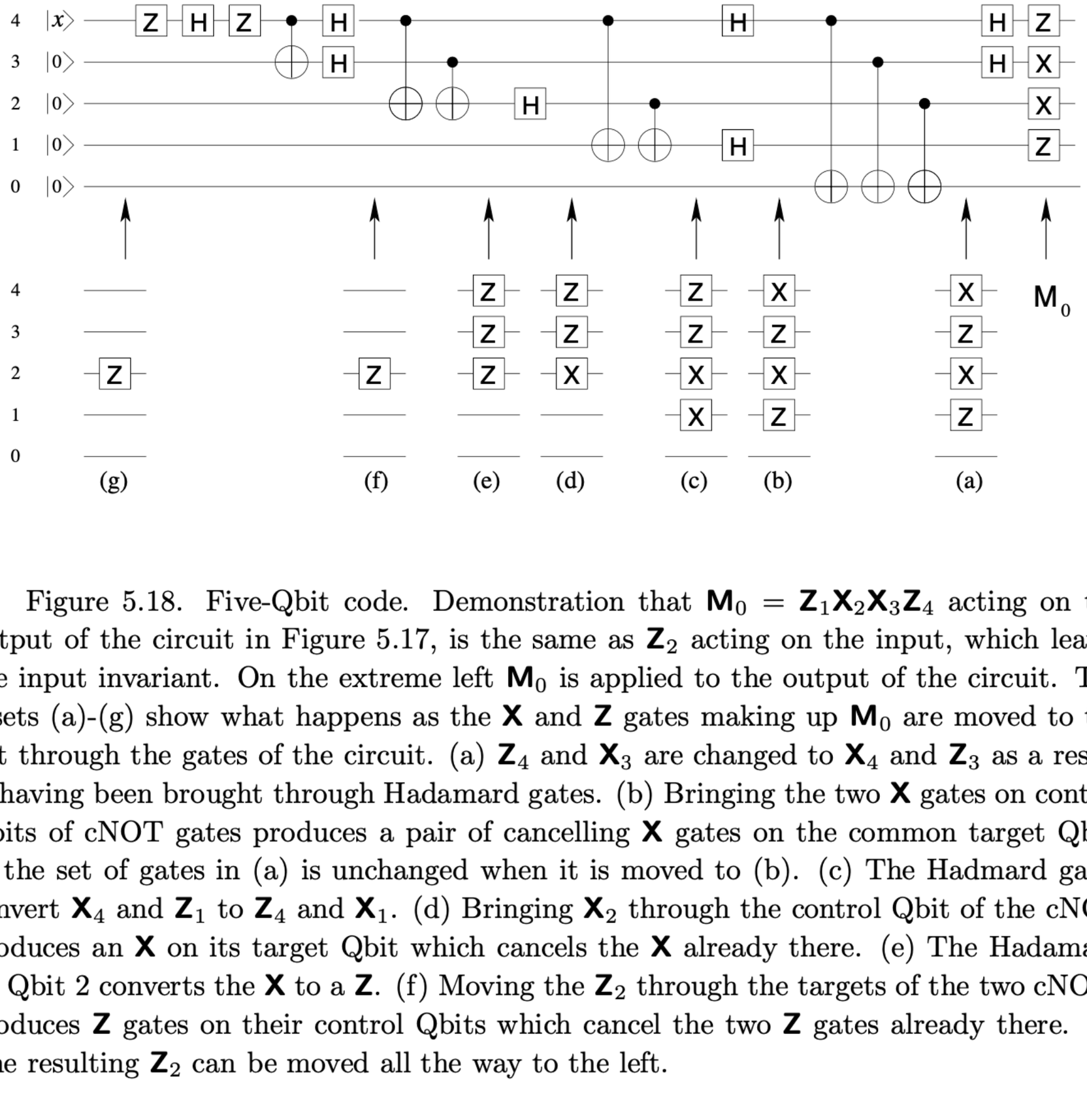


Figure 5.18. Five-Qbit code. Demonstration that $\mathbf{M}_0 = \mathbf{Z}_1\mathbf{X}_2\mathbf{X}_3\mathbf{Z}_4$ acting on the output of the circuit in Figure 5.17, is the same as $\mathbf{Z}_2$ acting on the input, which leaves the input invariant. On the extreme left $\mathbf{M}_0$ is applied to the output of the circuit. The insets (a)-(g) show what happens as the $\mathbf{X}$ and $\mathbf{Z}$ gates making up $\mathbf{M}_0$ are moved to the left through the gates of the circuit. (a) $\mathbf{Z}_4$ and $\mathbf{X}_3$ are changed to $\mathbf{X}_4$ and $\mathbf{Z}_3$ as a result of having been brought through Hadamard gates. (b) Bringing the two $\mathbf{X}$ gates on control Qbits of cNOT gates produces a pair of cancelling $\mathbf{X}$ gates on the common target Qbit, so the set of gates in (a) is unchanged when it is moved to (b). (c) The Hadmard gates convert $\mathbf{X}_4$ and $\mathbf{Z}_1$ to $\mathbf{Z}_4$ and $\mathbf{X}_1$. (d) Bringing $\mathbf{X}_2$ through the control Qbit of the cNOT produces an $\mathbf{X}$ on its target Qbit which cancels the $\mathbf{X}$ already there. (e) The Hadamard on Qbit 2 converts the $\mathbf{X}$ to a $\mathbf{Z}$. (f) Moving the $\mathbf{Z}_2$ through the targets of the two cNOTs produces $\mathbf{Z}$ gates on their control Qbits which cancel the two $\mathbf{Z}$ gates already there. (g) The resulting $\mathbf{Z}_2$ can be moved all the way to the left.

Figure 5.19

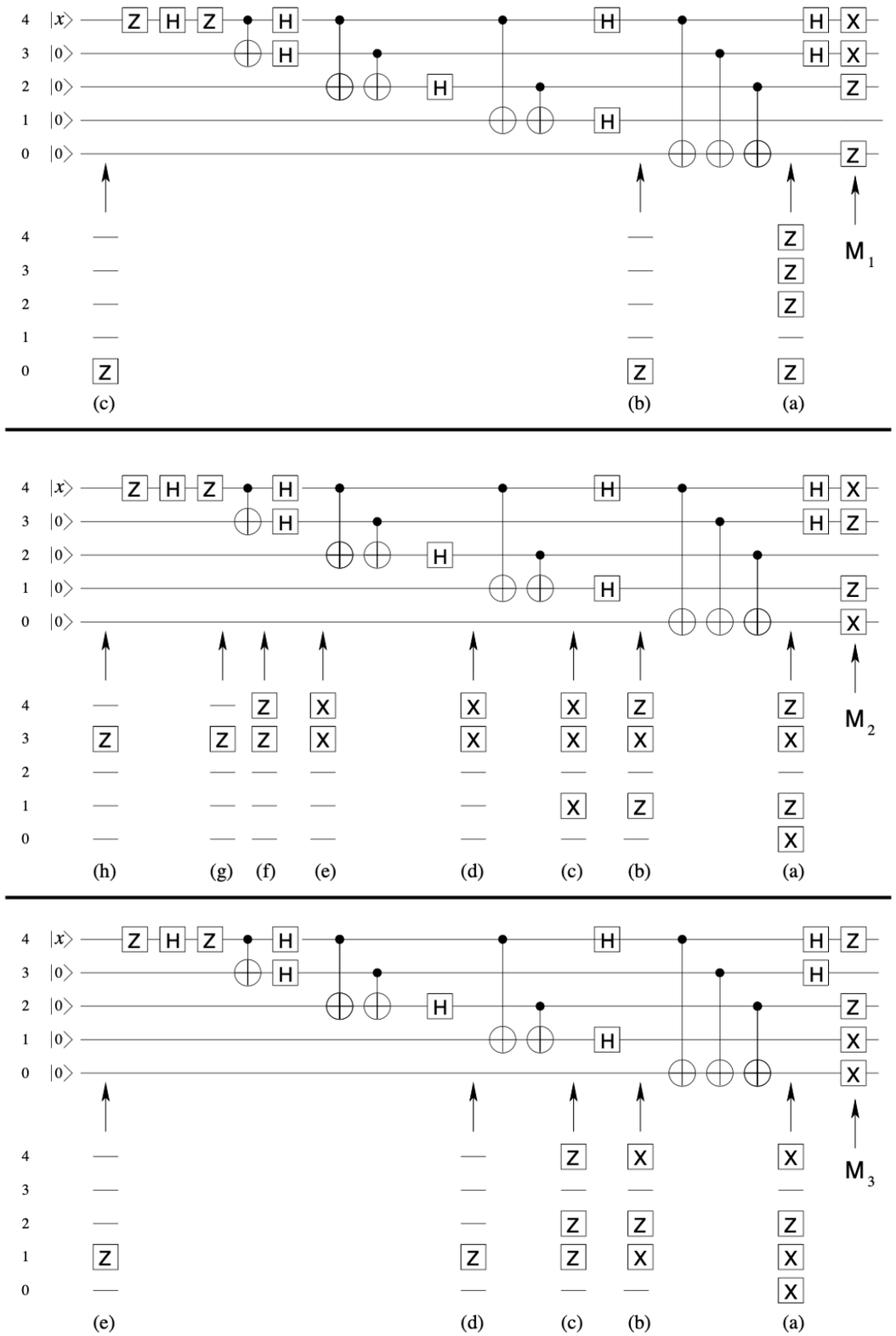


Figure 5.19. Five-Qbit code. Constructions similar to that of Figure 5.18 show that $\mathbf{M}_1$ can be brought to the left through the gates of the encoding circuit to act directly on $|x0000\rangle$ as $\mathbf{Z}_0$, $\mathbf{M}_2$ acts directly on $|x0000\rangle$ as $\mathbf{Z}_3$, and $\mathbf{M}_3$ as $\mathbf{Z}_1$.

Figure 5.20 shows that $\overline{\mathbf{X}} = \mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4$ can be brought to the left through all the gates of the circuit to act on the input state $|x0000\rangle$ as $\mathbf{X}_4\mathbf{Z}_2\mathbf{Z}_1$, which simply interchanges $x = 0$ and $x = 1$, thereby demonstrating that $\overline{\mathbf{X}}$ acts as logical X on the codewords. Figure 5.21 shows the analogous property for $\overline{\mathbf{Z}} = \mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{Z}_3\mathbf{Z}_4$, which can be brought to the left through all the gates of the circuit to act on the input state $|x0000\rangle$ as $\mathbf{Z}_4\mathbf{Z}_3\mathbf{Z}_0$, which multiplies it by $(-1)^x$, thereby demonstrating that $\overline{\mathbf{Z}}$ acts as logical Z on the codewords.

Finally Figure 5.22 shows, using the general rule illustrated in the box in Figure 5.15, that the inner product of the circuit acting on $|00000\rangle$ with the computational basis state $|00000\rangle$ is equal to $1/4$, thereby demonstrating that the circuit produces the codewords (5.33) with the right phase.

Figure 5.20

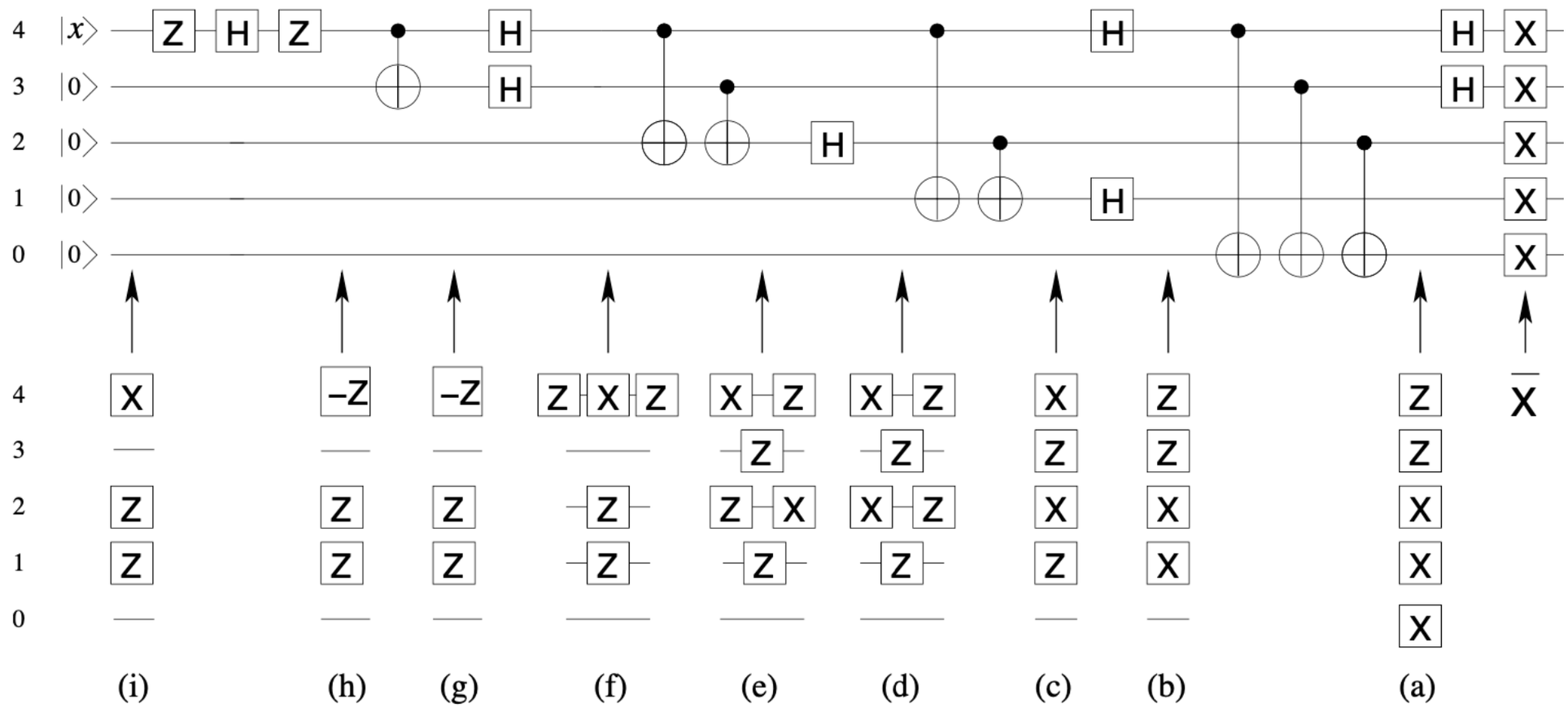


Figure 5.20. Five-Qbit code. Demonstration that $\overline{\mathbf{X}} = \mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4$ acting on the output of the circuit in Figure 5.17, is the same as $\mathbf{X}_4\mathbf{Z}_2\mathbf{Z}_1$ acting on the input, which interchanges $|00000\rangle$ and $|10000\rangle$. (a) Bringing $\mathbf{X}_4$ and $\mathbf{X}_3$ through the Hadamards converts them to $\mathbf{Z}_4$ and $\mathbf{Z}_3$. (b) Bringing $\mathbf{X}_2$ through the cNOT controlled by Qbit #2 produces an $\mathbf{X}$ on the target Qbit #0, which cancels the $\mathbf{X}$ already there. (c) The Hadamards convert $\mathbf{Z}_4$ and $\mathbf{X}_1$ to $\mathbf{X}_4$ and $\mathbf{Z}_1$. (d) Bringing $\mathbf{X}_4$ and $\mathbf{X}_2$ to the left produces two $\mathbf{X}_1$ gates which cancel. Bringing $\mathbf{Z}_1$ to the left then produces additional $\mathbf{Z}_4$ and $\mathbf{Z}_2$ gates. (e) The Hadamard $\mathbf{H}_2$ interchanges the $\mathbf{X}_2$ and $\mathbf{Z}_2$ gates. (f) First bring to the left the $\mathbf{Z}_2$ gate, then the $\mathbf{X}_4$ gate. (g) The $\mathbf{H}_4$ converts $\mathbf{ZXZ}$ to $\mathbf{XZX} = -\mathbf{Z}$. (h) No further changes. (g) $\mathbf{Z}$ commutes with itself, is changed to $\mathbf{X}$ on passing through $\mathbf{H}$, and acquires another minus sign on passage through $\mathbf{Z}$.

Figure 5.21

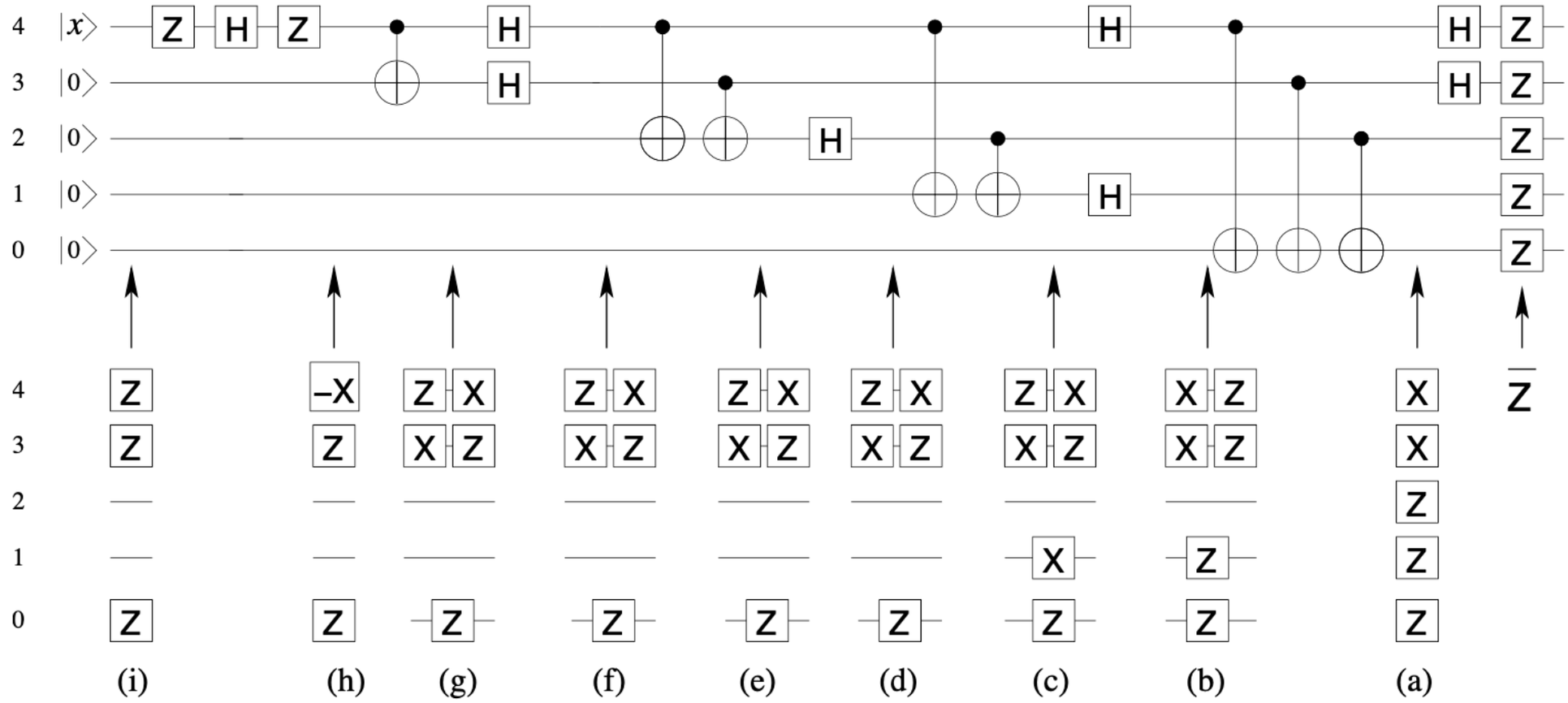


Figure 5.21. Five-Qbit code. Demonstration that $\overline{\mathbf{Z}} = \mathbf{Z}_0\mathbf{Z}_1\mathbf{Z}_2\mathbf{Z}_3\mathbf{Z}_4$ acting on the output of the circuit in Figure 5.17, is the same as $\mathbf{Z}_4\mathbf{Z}_3\mathbf{Z}_0$ acting on the input, which takes $|x0000\rangle$ into $(-1)^x|x0000\rangle$.

Figure 5.22

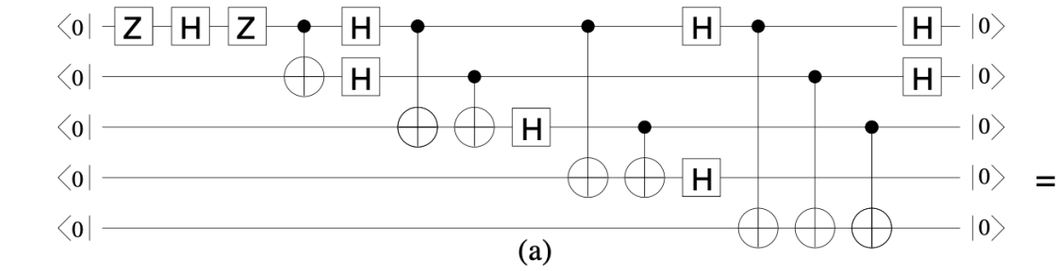


Figure 5.22. Five-Qbit code. Demonstration, using the method of Figure 16, that the state produced by the circuit in Figure 5.17 when $|\psi\rangle = |0\rangle$ has an inner product with the state $|0\rangle_5$ that is 1/4, thereby establishing that the phase factor $e^{i\varphi} = 1$ — i.e. that the state is precisely $|\overline{1}\rangle$ without any additional phase factor. (a) Circuit-theoretic representation of the inner product. (b) Elimination of operations in (a) that act as the identity: The cNOT on the extreme right of (a) can be dropped since its control Qbit is in the state $|0\rangle$. Since $\mathbf{H}|0\rangle$ is invariant under $\mathbf{X}$, the pair of cNOT gates targetting Qbit #1 can be dropped, as can the pair targetting Qbit #2. (c) A pair of Hadamards on Qbit #4 in (b) cancel. A Hadamard on Qbit #3 in (b) is moved to the left converting a cNOT to a controlled-Z. Qbits #2 and #1 in (b) simply give the matrix element $\langle 0|\mathbf{H}|0\rangle = \frac{1}{\sqrt{2}}$, resulting in an overall factor of $\frac{1}{2}$.

Extension to Part 5: The 9-Qbit code

Shor demonstrated that quantum error correction was possible using the two orthogonal 9-Qbit codeword states)

$$|\overline{0}\rangle = 2^{-3/2}\big(|000\rangle + |111\rangle\big)\big(|000\rangle + |111\rangle\big)\big(|000\rangle + |111\rangle\big),$$

$$|\overline{1}\rangle = 2^{-3/2}\big(|000\rangle - |111\rangle\big)\big(|000\rangle - |111\rangle\big)\big(|000\rangle - |111\rangle\big). \qquad (5.67)$$

These can be viewed as an extension of the simple 3-Qbit code words we examined in Section A, making it possible to deal with 1-Qbit phase errors, as well as bit-flip errors. An encoding circuit for the 9-Qbit code — with an obvious resemblance to Figure 5.1 for the 3-Qbit code — is shown in Figure 5.23.

The form (5.18) of a general 1-Qbit corruption simplifies slightly when the state $|\Psi\rangle$ is a superposition of the codeword states (5.67), for it follows from (5.67) that

$$\mathbf{Z}_0|\Psi\rangle = \mathbf{Z}_1|\Psi\rangle = \mathbf{Z}_2|\Psi\rangle, \quad \mathbf{Z}_3|\Psi\rangle = \mathbf{Z}_4|\Psi\rangle = \mathbf{Z}_5|\Psi\rangle, \quad \mathbf{Z}_6|\Psi\rangle = \mathbf{Z}_7|\Psi\rangle = \mathbf{Z}_8|\Psi\rangle. \quad (5.68)$$

As a result, the general form of a 1-Qbit corruption of $|\Psi\rangle$ contains only 22 independent terms (rather than $28 = 3 \times 9 + 1$):

$$|e\rangle|\Psi\rangle \to \Big(|d\rangle + |c\rangle\mathbf{Z}_0 + |c'\rangle\mathbf{Z}_3 + |c''\rangle\mathbf{Z}_6 + \sum_{i=1}^{9}\big(|a_i\rangle\mathbf{X}_i + |b_i\rangle\mathbf{Y}_i\big)\Big)|\Psi\rangle. \qquad (5.69)$$

Shor's diagnoses the error syndrome with eight commuting hermitian operators that square to unity:

$$\mathbf{Z}_0\mathbf{Z}_1,\ \ \mathbf{Z}_1\mathbf{Z}_2,\ \ \mathbf{Z}_3\mathbf{Z}_4,\ \ \mathbf{Z}_4\mathbf{Z}_5,\ \ \mathbf{Z}_6\mathbf{Z}_7,\ \ \mathbf{Z}_7\mathbf{Z}_8$$

$$\mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_5,\ \ \mathbf{X}_3\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6\mathbf{X}_7\mathbf{X}_8. \tag{5.70}$$

All six Z-operators trivially commute with each other as do the two X-operators, and any of the six Z-operator commutes with any of the two X-operators because in every case the number of anti-commutations is either zero or two.

One easily confirms from (5.67) that $|\overline{0}\rangle$, $|\overline{1}\rangle$, and hence any superposition $|\Psi\rangle$ of the two, is invariant under all 8 operators (5.70). Each one of the 21 corrupted terms in (5.69) is also an eigenstate of the eight operators (5.70) with eigenvalues 1 or -1, because each of the 8 operators either commutes (resulting in the eigenvalue 1) or anti-commutes (resulting in the eigenvalue -1) with each of the $\mathbf{X}_i$, $\mathbf{Y}_i$, and $\mathbf{Z}_i$. And each of the 21 terms in (5.69) gives rise to a distinct pattern of negative eigenvalues for the eight operators:

(a) The three errors $\mathbf{Z}_0$, $\mathbf{Z}_3$, and $\mathbf{Z}_6$ are distinguished from the $\mathbf{X}_i$ and $\mathbf{Y}_i$ by the fact that they commute with every one of the six Z operators in (5.70). These three $\mathbf{Z}_i$ can be distinguished from each other because $\mathbf{Z}_0$ anticommutes with one of the two X operators, $\mathbf{Z}_6$ anticommutes with the other, and $\mathbf{Z}_3$ anticommutes with both.

(b) All nine errors $\mathbf{X}_i$ are distinguished from both the $\mathbf{Z}_i$ and the $\mathbf{Y}_i$ by the fact that they commute with both X operators. They can be distinguished from each other because $\mathbf{X}_0$, $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_5$, $\mathbf{X}_6$, and $\mathbf{X}_8$ each anticommutes with a single one of the six Z operators in (5.70) (respectively $\mathbf{Z}_0\mathbf{Z}_1$, $\mathbf{Z}_1\mathbf{Z}_2$, $\mathbf{Z}_3\mathbf{Z}_4$, $\mathbf{Z}_4\mathbf{Z}_5$, $\mathbf{Z}_6\mathbf{Z}_7$, $\mathbf{Z}_7\mathbf{Z}_8$) while $\mathbf{X}_1$, $\mathbf{X}_4$, and $\mathbf{X}_7$ each anticommutes with two distinct Z operators (respectively $\mathbf{Z}_0\mathbf{Z}_1$ and $\mathbf{Z}_1\mathbf{Z}_2$, $\mathbf{Z}_3\mathbf{Z}_4$ and $\mathbf{Z}_4\mathbf{Z}_5$, and $\mathbf{Z}_6\mathbf{Z}_7$ and $\mathbf{Z}_7\mathbf{Z}_8$.

(c) Finally, the nine errors $\mathbf{Y}_i$ have the same pattern of commutations with the Z operators in (5.70) as the corresponding $\mathbf{X}_i$ operators, permitting them to be distinguished from each other in the same way. They can be distinguished from the $\mathbf{X}_i$ operators by their failure to commute with at least one of the two X operators in (5.70).

So, as with the other codes we have examined, the simultaneous measurement of the eight commuting operators (5.70) projects the corrupted state onto a single one of the terms in (5.69), and the set of eigenvalues reveals which term it is. One then applies the appropriate inverse unitary transformation to restore the uncorrupted state. A circuit that diagnoses the 9-Qbit error syndrome is shown in Figure 5.24.

Figure 5.23

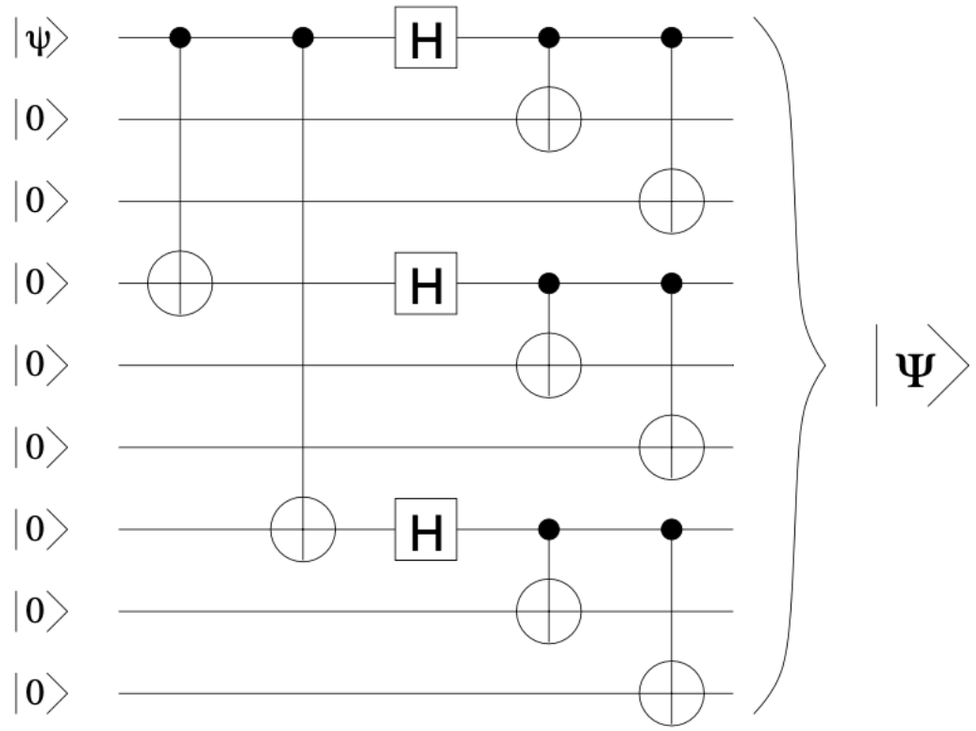


Figure 5.23. A circuit that transforms the 1-Qbit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ into its 9-Qbit encoded form $|\Psi\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle$, where $|\overline{0}\rangle$ and $|\overline{1}\rangle$ are given in (5.67). Note the relation to the simpler 3-Qbit encoding circuit in Figure 5.1.y

Figure 5.24

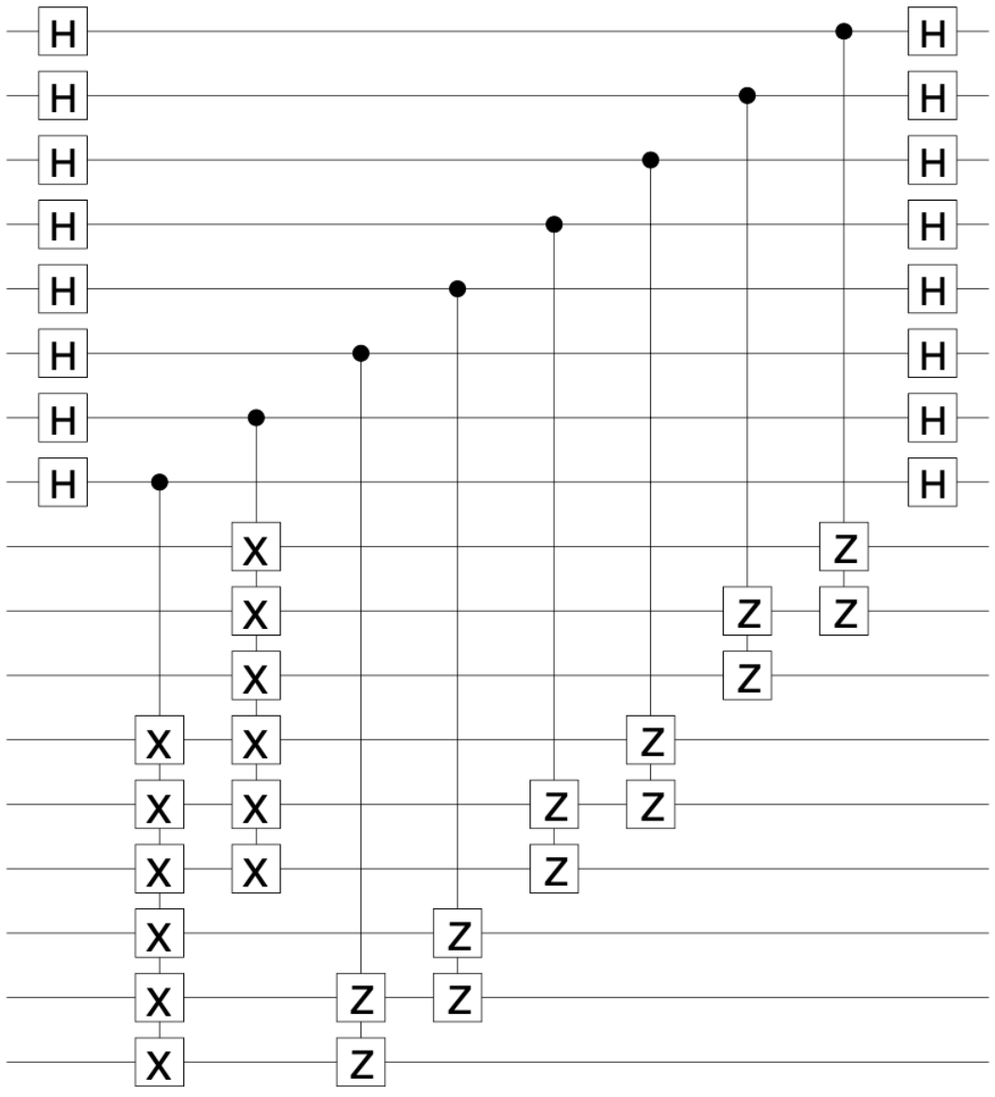


Figure 5.24. A circuit to measure the "error syndrome" for Shor's 9-Qbit code. The nine Qbits are the nine lower wires. The circuit is of the type illustrated in Figure 5.1, but with eight ancillary Qbits (the eight upper wires) associated with the measurement of the 8 commuting operators in (5.70), $\mathbf{Z}_0\mathbf{Z}_1$, $\mathbf{Z}_1\mathbf{Z}_2$, $\mathbf{Z}_3\mathbf{Z}_4$, $\mathbf{Z}_4\mathbf{Z}_5$, $\mathbf{Z}_6\mathbf{Z}_7$, $\mathbf{Z}_7\mathbf{Z}_8$, $\mathbf{X}_0\mathbf{X}_1\mathbf{X}_2\mathbf{X}_3\mathbf{X}_4\mathbf{X}_5$, and $\mathbf{X}_3\mathbf{X}_4\mathbf{X}_5\mathbf{X}_6\mathbf{X}_7\mathbf{X}_8$. Measurement of the eight ancillas projects the state of the nine lower Qbits into the appropriate simultaneous eigenstate of those eight operators.

Part 6 - Quantum cryptography and some simple uses of entanglement

Most of the examples that follow make use of a very simple entangled state, the 2-Qbit state

$$|\psi_{00}\rangle = \tfrac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big). \tag{6.1}$$

It can be produced from two Qbits each in the state $|0\rangle$ by applying a Hadamard to one of them, and then using it as the control Qbit for a cNOT that targets the other (Figure 6.1(a)):

$$|\psi_{00}\rangle = \mathbf{C}_{10}\mathbf{H}_1|00\rangle. \tag{6.2}$$

We can generalize (6.2) by letting the original pair of unentangled Qbits be in any of the four 2-Qbit computational basis states $|00\rangle, |01\rangle, |10\rangle$, or $|11\rangle$ (Figure 6.1(b)):

$$|\psi_{xy}\rangle = \mathbf{C}_{10}\mathbf{H}_1|xy\rangle. \tag{6.3}$$

Since the four states $|xy\rangle$ are an orthonormal set and the Hadamard and cNOT gates are unitary, the four entangled states $|\psi_{xy}\rangle$ are also an orthonormal set, called the *Bell basis* to honor the memory of the physicist John S. Bell, who discovered, all the way back in 1964 ("Bell's Theorem"), one of the most extraordinary facts about 2-Qbit entangled states.

Figure 6.1

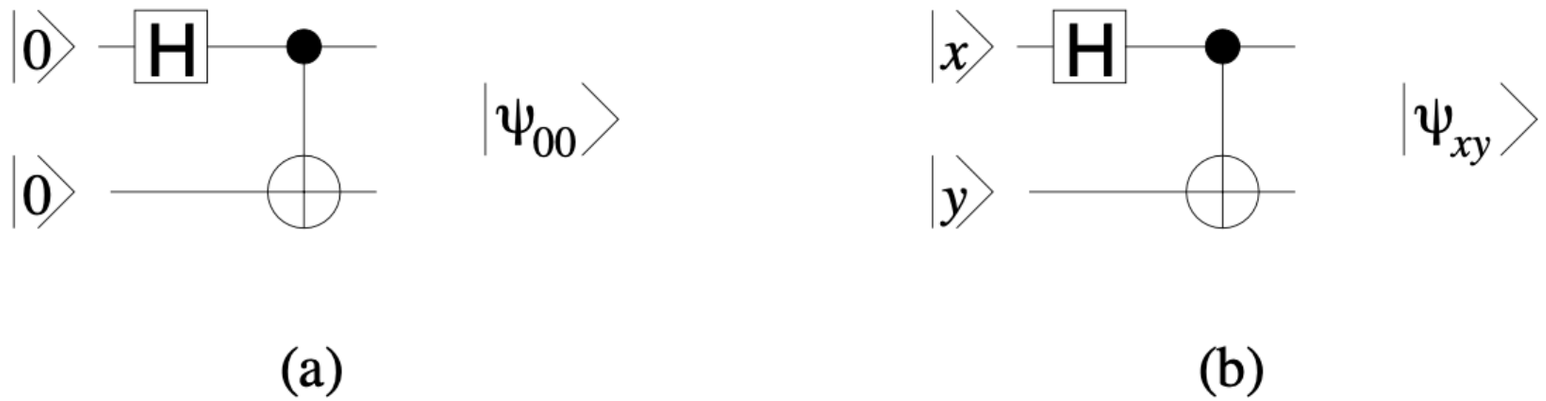


Figure 6.1. (a) Circuit that creates the entangled state $|\psi_{00}\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big)$ from the unentangled computational basis state $|00\rangle$. (b) Circuit that creates the four orthonormal entangled Bell states $|\psi_{xy}\rangle$ from the unentangled computational basis state $|xy\rangle$.

Rewrite (6.3) as

$$|\psi_{xy}\rangle = \mathbf{C}_{10}\mathbf{H}_1\mathbf{X}_1^x\mathbf{X}_0^y|00\rangle, \tag{6.4}$$

and recall that $\mathbf{HX} = \mathbf{ZH}$ and that either a $\mathbf{Z}$ on the control Qbit or an $\mathbf{X}$ on the target Qbit commute with a cNOT. We then have (see also Figure 6.2)

$$|\psi_{xy}\rangle = \mathbf{Z}_1^x\mathbf{X}_0^y\mathbf{C}_{10}\mathbf{H}_1|00\rangle = \mathbf{Z}_1^x\mathbf{X}_0^y\tfrac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big). \tag{6.5}$$

Figure 6.2

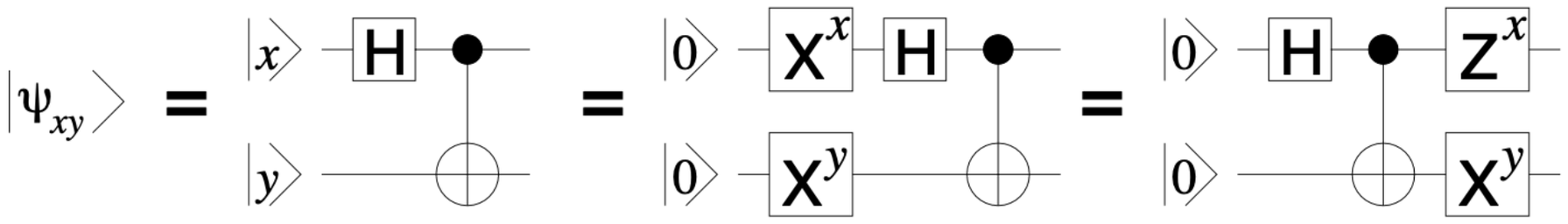


Figure 6.2. Demonstration that the Bell states $|\psi_{xy}\rangle$ can be constructed from $|\psi_{00}\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big)$ by either flipping a single Qbit, changing the sign from $+$ to $-$, or doing both of these.

So the other Bell states are obtained from $\frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big)$ by either flipping one of the Qbits, changing the $+$ to a $-$. or both. This, of course, can also be derived directly from evaluating (6.3) for the 4 choices of x amd y, , but it is more simply understood from (6.5).

We now examine a few simple protocols in which some or all of the Bell states play an important role.

A. Quantum cryptography

A decade before Shor's discovery that quantum computation posed a threat to the security of RSA encryption, it was pointed out that Qbits (though the term did not exist at that time) offered a quite different and demonstrably secure basis for the exchange of secret messages.

Of all the various applications and gedanken applications of quantum mechanics to information processing, quantum cryptography arguably holds the most promise for someday becoming a practical technology. There are several reasons for this. First of all, it works Qbit by Qbit. The only relevant gates are a small number of simple 1-Qbit gates. Interactions between pairs of Qbits like those mediated by cNOT gates play no role, at least in the most straightforward versions of the protocol.

Furthermore, in actual physical realizations of quantum cryptography the physical Qbits are extremely simple. Each Qbit is a single photon of light. The state of the Qbit is the linear polarization state of the photon. If the states $|0\rangle$ and $|1\rangle$ describe photons with vertical or horizontal polarization, then the states $\mathbf{H}|0\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)$ and $\mathbf{H}|1\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)$ describe photons diagonally polarized, either at $45°$ or $-45°$ to the vertical. Photons in any of these four polarization states can be prepared in any number of ways, most simply (if not most efficiently) by sending a weak beam of light through an appropriately oriented polaroid filter. Once a photon has been prepared in such a polarization state it does not have to be manipulated any further beyond eventually measuring either its horizontal-vertical or its diagonal polarization by, for example, sending it through an appropriately oriented birefringent crystal and seeing which beam it emerges in, or seeing whether it does or does not get through an appropriately oriented polaroid filter. Photons can effectively be shielded from extraneous interactions by sending them through optical fibers, where they can travel in a polarization-preserving manner at the speed of light.

The great utility of easily transportable single Qbits for secret communication comes from one important cryptographic fact: Alice and Bob can have an unbreakable code if they share newly created identical strings of random bits, called *one-time codepads*. Suppose they both have such identical random strings. Alice can then take her message, in the form of a long string of 0's and 1's (obtained by translating her original text using some transparent, generally known binary code such as ASCII coding) and transform it into its bitwise modulo-2 sum (also called the *exclusive or* or XOR) with a random string of 0's and 1's of the same length taken from her one-time codepad. Flipping or not flipping each bit of a coherent message according to whether the corresponding bit of a random string is 0 or 1, converts the message into another random string. If this is not immediately evident, think of the process as flipping or not flipping each bit of the random string, according to whether the corresponding bit of the coherent message is 0 or 1.

There is no way anyone can reconstruct the original string without knowing the random string used to encode it. So only Bob, who does have a copy of that random string, can decode the message. He can do this quite simply by taking the XOR of the now meaningless string of 0's and 1's he received from Alice with his copy of the same random string she used to do the encoding. The string he gets in this way is $M \oplus S \oplus S$, where M is the message, S is the random string, and $M \oplus S$ is the encoded message from alice. Since $S \oplus S = 0$, Bob has recovered the original message.

The problem with one-time codepads is that they can be used only once. If an eavesdropper (Eve) picks up two messages encoded with the same pad, she can take the XOR of the two encoded messages. The random string used to encode the two messages drops out of this process, leaving the XOR of the two unencoded messages. But the XOR of two meaningful messages, combined with the usual code-breaking tricks based on letter frequencies, can be used (with more sophistication than would be required for a single message) to separate and decode both texts. So to be perfectly secure Alice and Bob must continually refresh their one-time codepad with identical new random strings of bits.

The problem of exchanging such random strings in a secure way would appear to be identical to the original problem of exchanging meaningful messages in a secure way. But at this point quantum mechanics comes to the rescue, providing an entirely secure means for exchanging identical sequences of random bits. Pause to savour the strange character of this situation. Nobody has figured out how to exploit quantum mechanics to provide a secure means for directly exchanging meaningful messages. The secure exchange is possible only because the bit sequences are random. On the face of it one would think nothing could be more useless than such a transmission of noise. What is bizarre is that human ingenuity combined with human perversity has succeeded in inventing a context in which the need to hide information from a third party actually provides a purpose for such an otherwise useless exchange of random strings of bits.

The scheme for doing this is known as BB84 after its inventors, Charles Bennett and Gilles Brassard, who published the idea in 1984. Alice sends Bob a long sequence of Qbits in the form of photons, randomly chosen to be in one of four states: polarized horizontally (i.e. in the state $|0\rangle$), polarized vertically (in the state $|1\rangle$), polarized diagonally along 45° (in the state $\mathbf{H}|0\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)$), or polarized diagonally along −45° (in the state $\mathbf{H}|1\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)$). So for each photon Alice randomly chooses a polarization type (horizontal-vertical or diagonal) and within each type randomly choses a polarization value — one of the two orthogonal states associated with that type of polarization.

For the rest of this discussion we revert from photon polarization language to our more familiar quantum computational language. We divide the four equally likely types of Qbits that Alice sends to Bob into two categories: those with states $|0\rangle$ or $|1\rangle$, which we call type-1 Qbits, and those with state $\mathbf{H}|0\rangle$ or $\mathbf{H}|1\rangle$, which we call type-H Qbits. As each Qbit arrives Bob randomly decides whether to send it directly through a measurement gate, or to apply a Hadamard and only then send it through a measurement gate. We call these two options type-1 and type-H measurements. The Qbits must be individually identifiable in a that enables Bob and Alice to tell which is which way — for example by the sequence in which they arrive — so that they can compare certain information about what each of them did to and knows about each individual Qbit.

When Bob has measured all the Qbits in this way, Alice tells him over an insecure channel which Qbits were type-1 and which were type-H. But she does not tell him which of the two states she prepared within each type. For those Qbits (about half of them) for which Bob's random choice of measurement type agreed with Alice's random choice of which type to send, Bob learns from the result of his measurement the actual randomly selected value — 0 or 1 — that Alice sent. For those Qbits (the other half) where Bob's choice of which type to measure disagreed with Alice's choice of which type to send, the result of his measurement reveals nothing about Alice's randomly selected value. This is illustrated in Figure 6.3.

Finally, Bob tells Alice, over an insecure channel, which of the Qbits he subject to a type of measurement that agreed with her choice of which type to prepare — i.e. which Qbits were of the kind that can provide them with identical random bits. They discard the half of their data associated with the useless Qbits for which Bob's type of measurement differed from Alice's type of preparation. They are then able to construct their one-time codepads from the identical strings of random bits they have acquired.

You might wonder why Bob doesn't wait to decide what type of measurement to make on each Qbit until he learns Alice's choice of type for that photon, thereby doubling the number of shared random bits. This would indeed be a sensible strategy if Bob could store the Qbits he received from Alice. However storing individual photons in a polarization-preserving manner is technologically difficult. For currently feasible quantum cryptography Bob must make his decision and measure the polarization of each photon as it arrives.

Figure 6.3

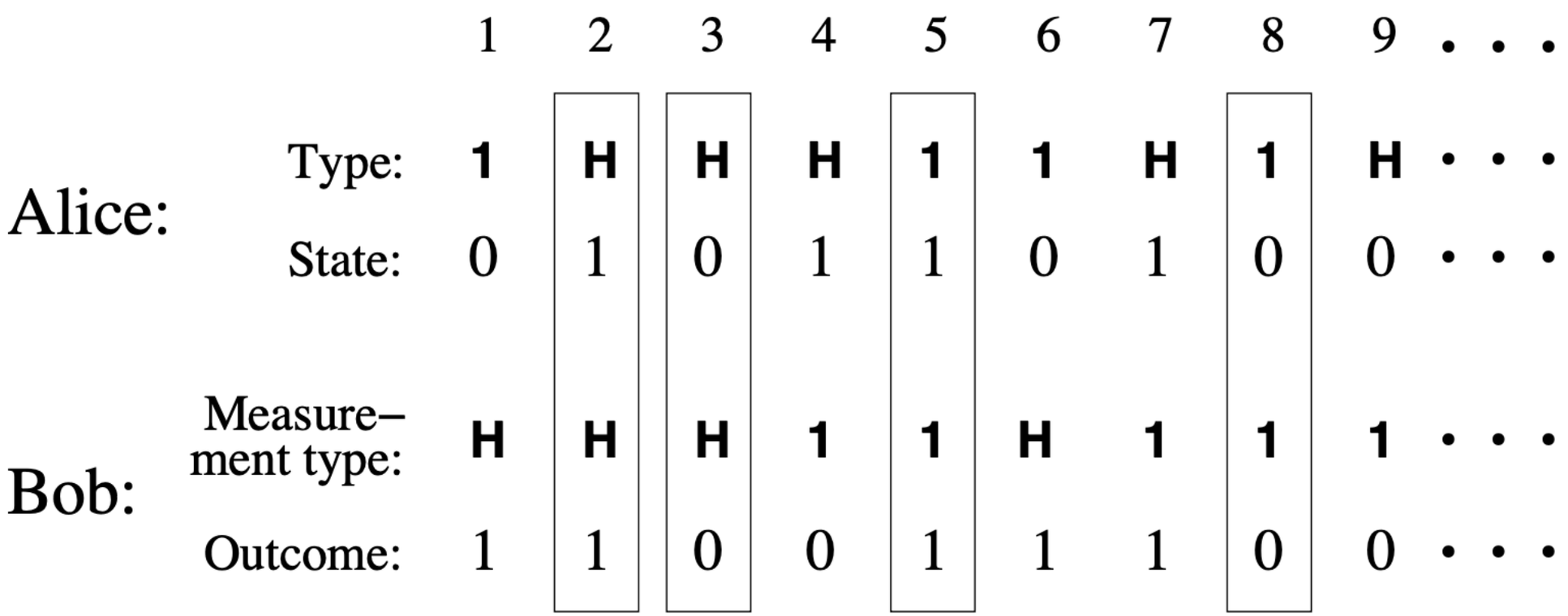


Figure 6.3. Quantum cryptography. For each Qbit she sends to Bob, Alice randomly decides which type of state to prepare it in (type **1** means $|x\rangle$ and type **H** means $\mathbf{H}|x\rangle$) and which state of that type ($x = 0$ or 1) to prepare. For each Qbit he receives from Alice, Bob randomly decides whether (**H**) or not (**1**) to apply a Hadamard gate before measuring it. In those cases (about half) where Bob's choice of measurement type is the same as Alice's choice of state, they acquire identical random bits. When their choices differ they acquire no useful information.

The reason they have to waste half of their Qbits is security against eavesdroppers. If Alice sent all Qbits of the same type and Bob always made that type of measurement so he always got a bit that agreed with Alice's (or if Alice sent with each Qbit classical information about its type), then an eavesdropper, Eve, could acquire the same information as Bob without being detected.

If, for example, Alice and Bob had agreed that all the Qbits would be type-1 and Eve learned of this, then she could intercept each Qbit before it reached Bob and send it directly through a measurement gate without altering its state, then sending it (or another Qbit she prepared in the state she just learned) on to Bob. In this way she could acquire exactly the same random bit that Alice sent out and that Bob subsequently acquires when he makes his own type-1 measurement. There is nothing to give Bob any hint that Eve was listening in.

By making each Qbit secretly and randomly of type-1 or type-H Alice deprives Eve of this strategy. The best Eve can do, like Bob, is to randomly make type-1 or type-H measurements. But in doing so she necessarily reveals her presence. Bob and Alice can determine that Eve has compromised the security of their bits by using the same protocol as before, but sacrificing some of the identical random bits they extracted from the Qbits that they both ended up treating in the same way.

They take a sample of these bits and check (over an insecure channel) to see whether they actually do agree, as they will in the absence of eavesdropping. If Eve was intercepting the Qbits, randomly making type-1 or type-H measurements of her own before sending them on to Bob, then for about half of the useful Qbits her choice will differ from the common choice of Alice and Bob. In about half of *those* cases, Eve's intervention will result in the outcome of Bob's measurement disagreeing with what Alice sent him. If, for example, Eve makes a type-1 measurement of a Qbit that Alice has prepared in the state $\mathbf{H}|0\rangle$, then she will necessarily change its state to one or the other of the two states $|0\rangle$ or $|1\rangle$. In either case if Bob then applies a Hadamard before measuring he will get the result 0 only half the time.

So if Eve is systematically intercepting Qbits, Bob's result will fail to agree with Alice's preparation for about a quarter of their sample data. This warns them that the transmission was insecure. If all the sample data agrees except for some tiny fraction, then they can set an upper limit to the fraction that Eve might have picked up, enabling them to make an informed judgment of the security with which they can use the remaining ones.

One might wonder whether Eve could do better by a more sophisticated attack, that involved capturing each of Alice's Qbits and processing it in a quantum computer that restored it to its initial state, before sending it on to Bob. This would eliminate the possibility of her eavesdropping being revealed to Bob. But the requirement that Alice's Qbit be returned to its initial state also eliminates the possibility of Eve learning anything useful.

To see this, let $|\phi_\mu\rangle$, $\mu = 0 \ldots 3$, be the four possible states of Alice's Qbit: $|0\rangle$, $|1\rangle$, $\mathbf{H}|0\rangle$, and $\mathbf{H}|1\rangle$. Let $|\Phi\rangle$ be the initial state of the n Qbits in Eve's computer, and let $\mathbf{U}$ be the $(n+1)$-Qbit unitary transformation the computer executes on its own Qbits and Alice's. Since Alice's Qbit must emerge in its original state, we have

$$\mathbf{U}\Big(|\phi_\mu\rangle) \otimes |\Phi\rangle\Big) = |\phi_\mu\rangle \otimes |\Psi_\mu\rangle. \tag{6.6}$$

Eve's hope is to devise a $\mathbf{U}$ that yields four $|\Psi_\mu\rangle$ whose differences enable her by subsequent processing and measurement to extract useful information about which of the four possible states $|\phi_\mu\rangle$ was. This, however, is impossible. Because neither $|\phi_0\rangle$ nor $|\phi_1\rangle$ are orthogonal to $|\phi_2\rangle$ or $|\phi_3\rangle$ one can easily show that all four $|\Psi_\mu\rangle$ must be identical.

This follows from the unitarity of $\mathbf{U}$. Unitary transformations preserve inner products, and therefore the inner product of any two input states to Eve's computation must be the same as the inner product of the two corresponding output states. We therefore have

$$\langle\phi_\nu|\phi_\mu\rangle\langle\Phi|\Phi\rangle = \langle\phi_\nu|\phi_\mu\rangle\langle\Psi_\nu|\Psi_\mu\rangle. \tag{6.7}$$

Because $\langle\Phi|\Phi\rangle = 1$ and because $\langle\phi_\nu|\phi_\mu\rangle \neq 0$ for $\mu\nu = 02, 03, 12, 13$, it follows that

$$\langle\Psi_\nu|\Psi_\mu\rangle = 1, \quad \mu\nu = 02, 03, 12, 13. \tag{6.8}$$

But the inner product of two normalized states can be 1 only if they are identical, so it follows from (6.8) that

$$|\Psi_0\rangle = |\Psi_1\rangle = |\Psi_2\rangle = |\Psi_3\rangle. \tag{6.9}$$

The price Eve pays for eliminating all traces of her eavesdropping is that she can learn, from the resulting state of her quantum computer, nothing whatever about the four possible states of Alice's Qbit.

There is a less practical (but more elegant) version of this cryptographic protocol that appears, at first sight, to be different, but turns out to be exactly the same. Suppose there were some central source that produced pairs of Qbits in the entangled state

$$|\Psi\rangle = \tfrac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big), \tag{6.10}$$

and then sent one member of each pair to Alice and the other to Bob. One easily verifies that

$$\big(\mathbf{H}\otimes\mathbf{H}\big)\tfrac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big) = \tfrac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big). \tag{6.11}$$

Therefore if Alice and Bob make measurements of the same type, they will get the same results.

This might appear to be even more secure than the first protocol, since the Qbits are in an entangled state until Alice or Bob actually makes a measurement. The correlated bits — the outcomes of the measurement — do not even *exist* until such a moment, and that moment does not happen until both Qbits are safely in their separate possession. But this is only the case if Eve does not intercept a Qbit. If she does measure one before it gets to Bob or Alice, then the correlated bits do come into existence at the moment of her measurement. This is later than in the first protocol (when each bit exists the moment Alice performs her own measurement) but early enough to help Eve in the same way as before.

If Alice and Bob decided to produce their perfectly correlated random bits by always making type-1 measurements then if Eve finds this out she can intercept one member of the pair with type-1 measurements of her own, disentangling the state prematurely, but in a way that enables her to learn what each random bit is, while not altering the the perfect correlations between the values Alice and Bob will subsequently measure. Alice and Bob can guard against this possibility by each randomly (and, necessarily, independently) alternating between type-1 and type-H measurements, and then following a procedure identical to the one they used when Alice sent Bob Qbits in definite states.

This gets us back to the original protocol that made no use of entangled pairs. Indeed, if Alice measures her member of the entangled pair (making either a type-1 or type-H measurement) before Bob measures his, this is completely equivalent to her sending Bob a Qbit with a randomly selected state that she knows. The only difference is that now the random choice of which of the two states to send within each type is not made by her tossing a coin, but by the basic laws of quantum mechanics that guarantee that the outcome of her own measurement is random.

B. Bit commitment

One can try to formulate a similar protocol for a procedure called bit commitment. Suppose Alice wishes to assure Bob that she has made a binary decision by a certain date, but does not wish to reveal that decision until some future time. She can do this by writing YES or NO on a card, putting the card in a box, locking the box, and sending the box, but not the key to Bob. Once the box is in Bob's possession he can be sure that Alice has not altered her decision, but while the key is in Alice's possession she can be sure that Bob has not learned what that decision was. When it is time for her to reveal the decision she sends the key to Bob who opens the box and learns what it was.

Of course Alice might worry about Bob breaking into the box by other means. Quantum mechanics offers a more secure procedure (but with a loophole, which we return to momentarily.) Alice prepares a large number n of labeled Qbits. If her answer is YES, she takes each Qbit to be randomly in the state $|0\rangle$ or the state $|1\rangle$. If her answer is NO she prepares each Qbit randomly in the state $\mathbf{H}|0\rangle$ or $\mathbf{H}|1\rangle$. In either case she notes which Qbits are in which state, and then sends them all off to Bob, who stores them in a way that preserves both their state and their labels.

Now if Bob has a collection of N Qbits, each of which has been chosen with equal probability to be in one of two orthogonal states $|\phi\rangle$ or $|\psi\rangle$, then there is no way for Bob to get any hint of what $|\phi\rangle$ and $|\psi\rangle$ are, for if he measures every pair then the probabilities of getting 0 or 1 are given by

$$p(0) = 1 - p(1) = \tfrac{1}{2}|\langle 0|\phi\rangle|^2 + \tfrac{1}{2}|\langle 0|\psi\rangle|^2. \tag{6.12}$$

But

$$|\langle 0|\phi\rangle|^2 + |\langle 0|\psi\rangle|^2 = 1, \tag{6.13}$$

since this is the sum of the squared moduli of the amplitudes of the expansion of $|0\rangle$ in the basis given by $|\phi\rangle$ and $|\psi\rangle$:

$$|0\rangle = |\phi\rangle\langle\phi|0\rangle + |\psi\rangle\langle\psi|0\rangle. \tag{6.14}$$

Therefore $p(0) = p(1) = \frac{1}{2}$, and Bob's measurement outcomes are completely random, independent of what the orthogonal pair of states actually is. The same conclusion holds if he applies a unitary transformation $\mathbf{U}$ to each Qbit before his measurement — i.e. if he measures in a basis other than the computational basis — since $\mathbf{U}|\phi\rangle$ and $\mathbf{U}|\psi\rangle$ are also orthogonal. So the statistics of any measurements Bob can perform on a collection of Qbits, each of which has a 50-50 chance of being in the state $|0\rangle$ or $|1\rangle$, are indistinguishable from the statistics he would get if each had a 50-50 chance of being in the state $\mathbf{H}|0\rangle$ or $\mathbf{H}|1\rangle$. There is no way Bob can learn Alice's choice by measuring the Qbits Alice has sent him.

It is important for this indistinguishability that Alice send Bob a collection of Qbits each of whose two possible states was selected randomly. If, on the contrary, she sent him *exactly* $\frac{1}{2}N$ Qbits in the state $|0\rangle$ and $\frac{1}{2}N$ in the state $|1\rangle$, in some random order, then with probability 1 Bob would get an equal number of 0's and 1's if he measured in the computational basis. But if he applied $\mathbf{H}$ before measuring, the outcome of each measurement would be random, and the probability of getting equal numbers of 0's and 1's for his measurements would be quite small (asymptotically $\sqrt{2/\pi N}$ for large N). Therefore if he found equal numbers he could be rather sure that Alice had sent him photons in the states $|0\rangle$ and $|1\rangle$ rather than in the states $\mathbf{H}|0\rangle$ and $\mathbf{H}|1\rangle$.

When the time comes for Alice to reveal her choice for the pair of orthogonal states, she says to Bob something like this: My answer was YES, so each of the Qbits I sent you was either in the state $|0\rangle$ or $|1\rangle$. To prove this I will now tell you that I put Qbits 1, 2, 4, 6, 7, 11,... into the state $|0\rangle$ and I put Qbits 3, 5, 8, 9, 10, 12... into the state $|1\rangle$. You can confirm that I'm telling the truth by measuring **Z** on each of them. Here Alice uses the language in which a measurement in the computational basis is referred to as "measuring **Z**" while a measurement in the computational basis preceded by an application of **H** is referred to as "measuring **X**.

Bob makes the measurements and gets every one of Alice's predicted outcomes. If Alice had instead sent him Qbits whose states were randomly $\mathbf{H}|0\rangle$ or $\mathbf{H}|1\rangle$ she could do the same trick for **X** measurements. But there is no way she could do the trick for **X** measurements in the first case or for **Z** measurements in the second. The best she could do if she wanted to deceive Bob would be to make random guesses for each outcome, and with n Qbits she would succeed only with probability $1/2^n$. So this works perfectly well, and without the worry of Bob possessing unexpected safe-cracking skills.

But as I warned above, there is a fatal problem. The technological skills required to take advantage of this loophole are spectacularly greater than those required for the naive protocol, so one could imagine a stretch of years or decades when the naive protocol might actually be useful, but ultimately it will be insecure. Suppose Alice cheats and, unknown to Bob, prepares n labeled entangled pairs in the state (6.11), sending one member to Bob while retaining the other for herself. Then in actual fact the Qbits Bob received would have no states of their own, being entangled with the Qbits Alice kept for herself. Nevertheless, if Bob chose to test some of them with measurements, the statistics he got would be indistinguishable from those he would have got if Alice had been playing the game honestly. No hint of her deception will be revealed by any test Bob can perform.

Now when the time comes for Alice to reveal her choice, if she wants to prove to Bob that it was YES, she measures **Z** on each of the Qbits she has kept and correctly informs Bob what he will get if he measures **Z** on each of the paired Qbits. But if she wants to prove that it was NO, she instead measures **X** on each of her Qbits. In this way she can use entangled pairs of Qbits to cheat at what would otherwise be a perfectly secure bit-commitment protocol.

She can cheat in the same way even if Bob measures **Z** or **X** randomly on his Qbits before she "reveals" her commitment. If she wants to "prove" to Bob she had sent him YES she measures **Z** on each of her Qbits and tells Bob all her results. He notes that they do indeed agree with all the results he found for his **Z** measurements, and is persuaded that she had indeed sent him YES.

Of course the success of Alice's cheating depends crucially on Bob's knowing all about one-Qbit states, but never having had the kind of course in quantum mechanics that taught him anything about entangled two-Qbit states. If Bob is as sophisticated a student of the quantum theory as Alice, they will both realize that the protocol is fatally flawed, since it can be defeated by entanglement.

It is in this context that Einstein's famous complaint about spooky actions at a distance ("*spukhafte Fernwirkungen*") seems particularly compelling. By finally measuring her members of the entangled pairs, Alice seems to convert the distant Qbits in Bob's possession into the kind she deceptively said she had sent him long ago, while retaining until the last minute the option of which of the two kinds to pick. But of course Alice's action is not so much on the Qbits in Bob's possession as it is on what it is possible for her to *tell him* about those Qbits. It is this peculiar tension between the objective (ontology) and what is known (epistemology) that makes quantum mechanics such a source of delight (or anguish) to the philosophically inclined.

Something like Alice's discovery of the value of entanglement for cheating actually happened in the historical development of these ideas about quantum information processing. When the bit-commitment protocol described above was first put forth it was realized that entangled pairs could be used to thwart it, but more sophisticated versions were proposed that were believed to be immune to cheating with entanglement. There developed a controversy over whether some form of bit commitment could or could not be devised that would be secure even if entanglement were fully exploitable. The current consensus is that there is no way to use Qbits in a bit-commitment protocol that cannot be defeated by using entangled states. Indeed, it has even been suggested that some clever person might try to prove that the structure of quantum mechanics was uniquely specified by requiring it to lead to the kind of physics that enables the secure exchange of random strings of bits, as in quantum cryptography, but does not enable bit commitment. As far as I know nobody has taken up this challenge. It does seem implausible that God would have taken as a fundamental principle of design that certain kinds of covert activity should be possible while others should be forbidden.

C. Quantum dense coding

Although an infinite amountof information is required to specify the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ of a single Qbit, there is no way for somebody who has acquired possession of the Qbit to learn what that state is, as we have often noted. If Alice prepares a Qbit in the state $|\psi\rangle$ and sends it to Bob, all he can do is apply a unitary transformation of his choice and then measure the Qbit, getting the value 0 or 1 — just one bit of information. After that the state of the Qbit is either $|0\rangle$ or $|1\rangle$ and no further measurement can teach him anything about its original state $|\psi\rangle$. The most Alice can communicate to Bob in this way is a single bit of information.

If, however, Alice has one member of an entangled pair of Qbits in the state

$$|\Psi\rangle = \tfrac{1}{\sqrt{2}}\big(|0\rangle|0\rangle + |1\rangle|1\rangle\big) \tag{6.15}$$

and Bob has the other, then by suitably preparing her member of the pair and then sending it to Bob, she can convey to him *two* bits of information. She does this by first applying the transformation $\mathbf{1}, \mathbf{X}, \mathbf{Z}$, or $\mathbf{ZX}$ to her Qbit, depending on whether she wants to send Bob the message 00, 01, 10, or 11. If hers is the Qbit on the left in (6.15) these transform the state of the pair into one of the four mutually orthogonal states, called the Bell basis (after John S. Bell):

$$\begin{aligned}
\mathbf{1}_a|\Psi\rangle &= \tfrac{1}{\sqrt{2}}\big(|0\rangle|0\rangle + |1\rangle|1\rangle\big),\\
\mathbf{X}_a|\Psi\rangle &= \tfrac{1}{\sqrt{2}}\big(|1\rangle|0\rangle + |0\rangle|1\rangle\big),\\
\mathbf{Z}_a|\Psi\rangle &= \tfrac{1}{\sqrt{2}}\big(|0\rangle|0\rangle - |1\rangle|1\rangle\big),\\
\mathbf{Z}_a\mathbf{X}_a|\Psi\rangle &= q\tfrac{1}{\sqrt{2}}\big(|0\rangle|1\rangle - |1\rangle|0\rangle\big).
\end{aligned} \tag{6.16}$$

She then sends her Qbit over to Bob. He applies to the pair the controlled-NOT transform $\mathbf{cX}_{ab}$ using the Qbit he received from Alice as control, to get

$$\begin{aligned}
\mathbf{cX}_{ab}\mathbf{1}_a|\Psi\rangle &= \tfrac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)|0\rangle,\\
\mathbf{cX}_{ab}\mathbf{X}_a|\Psi\rangle &= \tfrac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)|1\rangle,\\
\mathbf{cX}_{ab}\mathbf{Z}_a|\Psi\rangle &= \tfrac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)|0\rangle,\\
\mathbf{cX}_{ab}\mathbf{Z}_a\mathbf{X}_a|\Psi\rangle &= \tfrac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big)|1\rangle,
\end{aligned} \tag{6.17}$$

and then he applies a Hadamard transform to get

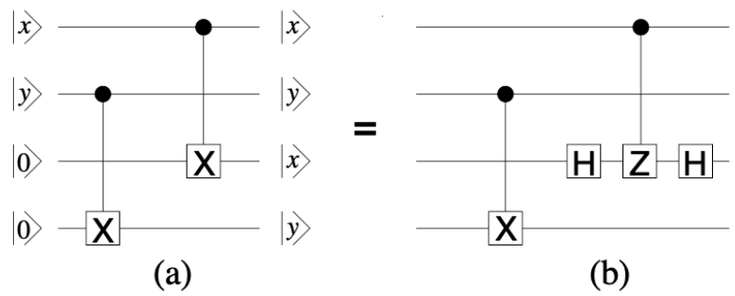
$$\begin{aligned}
\mathbf{H}_a\mathbf{cX}_{ab}\mathbf{1}_a|\Psi\rangle &= |0\rangle|0\rangle, \\
\mathbf{H}_a\mathbf{cX}_{ab}\mathbf{X}_a|\Psi\rangle &= |0\rangle|1\rangle, \\
\mathbf{H}_a\mathbf{cX}_{ab}\mathbf{Z}_a|\Psi\rangle &= |1\rangle|0\rangle, \\
\mathbf{H}_a\mathbf{cX}_{ab}\mathbf{Z}_a\mathbf{X}_a|\Psi\rangle &= |1\rangle|1\rangle.
\end{aligned} \tag{6.18}$$

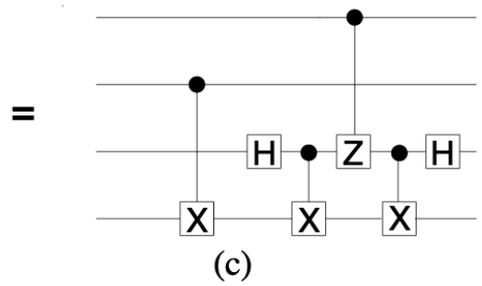
Measuring the two Qbits then gives him 00, 01, 10, or 11 — precisely the two-bit message Alice wished to send. This process of transforming the Bell basis into the computational basis and then measuring, is called "measuring in the Bell basis."

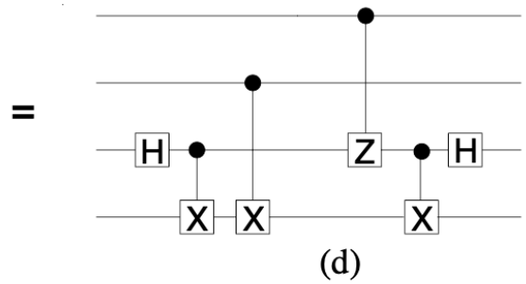
One can directly demonstrate that this works with circuit diagrams, without going through any of the analysis in (6.15)-(6.18). Suppose Alice represents the two bits x and y she wishes to transmit to Bob as the computational basis state $|x\rangle|y\rangle$ of two Qbits (top two wires, Fig. 6.4(a)). If Bob has two Qbits initially in the state $|0\rangle|0\rangle$ (bottom two wires in Fig. 6.4(a)), then the circuit in Fig. 6.4(a) gets the two bits to Bob in a straightforward classical way, transforming the state $|x\rangle|y\rangle|0\rangle|0\rangle$ on the right to $|x\rangle|y\rangle|x\rangle|y\rangle$ on the left by means of direct Qbit-to-Qbit coupling via two cNOT gates. The procedure involves only classical operations on classically meaningful states.

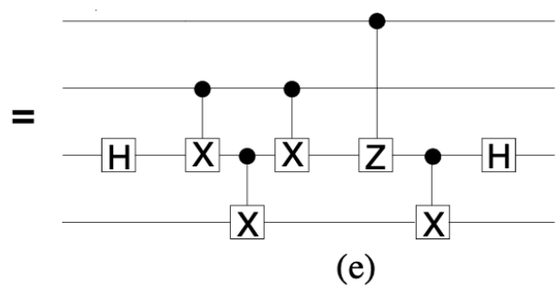
But now one can introduce quantum gates, beginning with the expansion of one of the **cX** into **H cZ H** in Fig. 6.4(b). Note next that because the computational basis states are eigenstates of **Z**, when **Z** acts on the control Qbit of a **cX**, the two operators commute. Since **cX** is its own inverse, we can further expand Fig. 6.4(b) to Fig. 6.4(c). We can then bring the **H** and **cX** gates on either side of the **cZ** to the extreme left and right to get Fig. 6.4(d).

Figure 6.4









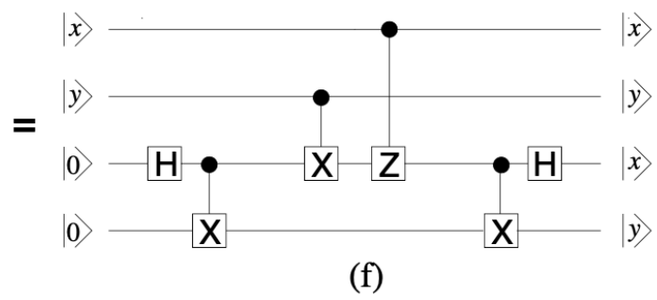


Figure 6.4. A circuit-theoretic derivation of the quantum dense coding protocol.

We can also expand the two **cX** gates on the right of Fig. 6.4(d) into the three **cX** gates on the right of Fig. 6.4(e), since the action of either set is to flip the target Qbit if and only if the computational basis states of the two control Qbits are different, while leaving the states of the control Qbits unaltered. Because the state $\mathbf{H}|0\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big)$ is invariant under the action of **X**, the **cX** on the extreme right acts as the identity, and Fig. 6.4(e) simplifies to Fig. 6.4(f). The fact that Fig. 6.4(f) has the same action as Fig. 6.4(a) contains all the content of the dense-coding protocol:

The two gates $\mathbf{cX}_{23}\mathbf{H}_2$ on the right of Fig. 6.4(f) act on the state $|0\rangle|0\rangle$ to produce the entangled state (6.15). The bottom Qbit of the pair, #3, is given to Bob and the upper one, #2, is given to Alice, who also possesses the upper two, #0 and #1. The two gates $\mathbf{cZ}_{02}\,\mathbf{cX}_{12}$ act as $\mathbf{1}, \mathbf{X}, \mathbf{Z}$, or $\mathbf{ZX}$ on Qbit #2 depending on whether the states of Qbits #0 and #1 are $|0\rangle|0\rangle, |0\rangle|1\rangle, |1\rangle|0\rangle$, or $|1\rangle|1\rangle$. This reproduces the transformation Alice applies to the member of the entangled pair in her possession, depending on the values of the two bits she wishes to transmit to Bob. Alice then sends Qbit #2 to Bob. The final $\mathbf{H}_2\mathbf{cX}_{23}$ on the left is precisely the transformation (6.18) that Bob performs on the reunited entangled pair before making his measurement, which yields the values x, y that Alice wished to transmit.

Like dense coding, many tricks of quantum information theory, including the one we examine next, teleportation, rely on two or more people sharing entangled Qbits, prepared some time ago, carefully stored in their remote locations awaiting an occasion for their use. While the preparation of entangled Qbits (in the form of photons) and their transmission to distant places has been achieved, as far as I know there has been no progress in putting them into entanglement-preserving, local, long-term storage.

D. Teleportation

Suppose Alice has a Qbit in a general state

$$|\psi\rangle = a_0|0\rangle + a_1|1\rangle. \tag{6.19}$$

She does not, in general, know the amplitudes a_0 and a_1. Carol, for example, may have prepared the Qbit for Alice by first applying a unitary transformation to a Qbit initially in the standard state $|0\rangle$ and then sending the output Qbit to Alice without telling her what she did to prepare it.

Alice would like to reassign that precise state to another Qbit possessed by Bob. Neither Alice nor Bob (who could be far away from Alice) has any access to the other's Qbit. Alice is, however, allowed to send "classical information" to Bob — e.g. she can talk to him over the telephone. And, most importantly, Alice and Bob again share a pair of Qbits in the entangled state

$$|\Phi\rangle = \tfrac{1}{\sqrt{2}}\big(|0\rangle|0\rangle + |1\rangle|1\rangle\big). \tag{6.20}$$

The no-cloning theorem prohibits the duplication of the unknown state of Alice's Qbit, either far away from her or nearby. But it turns out to be possible, and does not violate that theorem, for Alice and Bob to cooperate over the telephone to transform the state of Bob's Qbit from $|0\rangle$ to $|\psi\rangle$ provided Alice is willing to obliterate all traces of its former state from her own Qbit. The term "teleportation" is apt, because the state that is newly associated with Bob's far away Qbit must necessarily cease to be associated with Alice's.

The initial 3-Qbit state of the Alice's special Qbit and the entangled pair she shares with Bob is

$$\big(a|0\rangle_a + b|1\rangle_a\big)\tfrac{1}{\sqrt{2}}\big(|0\rangle_a|0\rangle_b + |1\rangle_a|1\rangle_b\big), \tag{6.21}$$

where I have given the state symbols for the Qbits in Alice's possession a subscript a, and the state symbols for the Qbit in Bob's possession the subscript b. To teleport the unknown state of her special Qbit to Bob's member of the entangled pair Alice first applies a **cX** gate using her special Qbit as the control and her member of the entangled pair as the target. This produces the state

$$a|0\rangle_a\tfrac{1}{\sqrt{2}}\big(|0\rangle_a|0\rangle_b + |1\rangle_a|1\rangle_b\big) + b|1\rangle_a\tfrac{1}{\sqrt{2}}\big(|1\rangle_a|0\rangle_b + |0\rangle_a|1\rangle_b\big). \tag{6.22}$$

Next she applies a Hadmard transformation **H** to her special Qbit (on the left in (6.22)), which puts all three Qbits into the state

$$a\tfrac{1}{\sqrt{2}}\big(|0\rangle_a + |1\rangle_a\big)\tfrac{1}{\sqrt{2}}\big(|0\rangle_a|0\rangle_b + |1\rangle_a|1\rangle_b\big) + b\tfrac{1}{\sqrt{2}}\big(|0\rangle_a - |1\rangle_a\big)\tfrac{1}{\sqrt{2}}\big(|1\rangle_a|0\rangle_b + |0\rangle_a|1\rangle_b\big) =$$

$$\begin{aligned}&\tfrac{1}{2}|0\rangle_a|0\rangle_a\big(a|0\rangle_b + b|1\rangle_b\big) + \tfrac{1}{2}|1\rangle_a|0\rangle_a\big(a|0\rangle_b - b|1\rangle_b\big) + \\ &\tfrac{1}{2}|0\rangle_a|1\rangle_a\big(a|1\rangle_b + b|0\rangle_b\big) + \tfrac{1}{2}|1\rangle_a|1\rangle_a\big(a|1\rangle_b - b|0\rangle_b\big).\end{aligned} \tag{6.23}$$

Alice now measures the two Qbits in her possession. (As noted earlier, such application of **cX** and **H** gates immediately followed by measurement gates is called "measuring in the Bell basis.") If the result is 00, Bob's Qbit (on the right) will indeed be projected into the state (6.19) originally possessed by Alice's Qbit (whose state would then be reduced to $|0\rangle$). But if the result of Alice's measurement is 10, 01, or 11 then the state of Bob's Qbit becomes

$$a|0\rangle_b - b|1\rangle_b, \quad a|1\rangle_b + b|0\rangle_b, \quad \text{or} \quad a|1\rangle_b - b|0\rangle_b. \tag{6.24}$$

Note that in each of these three cases there is a unitary transformation that restores the state of Bob's Qbit to Alice's original (unknown) state (6.19). In the first case we can apply **Z**, (which leaves $|0\rangle$ alone but changes the sign of $|1\rangle$), in the second case, **X** (which interchanges $|0\rangle$ and $|1\rangle$), and in the third case, **ZX**.

So all Alice need do to transfer the state of her Qbit to Bob's member of their entangled pair is to telephone Bob and report to him the results of her two measurements. He then knows whether the state has already been transferred (if Alice's result is 00) or what unitary transformation he must apply to his member of the entangled pair in order to complete the transfer (if Alice's result is one of the other three.) This is very similar to the error correction procedures described in Chapter 5. By making a measurement Alice acquires the information necessary to reconstruct a particular quantum state without acquiring any information about what that state actually is.

Note what has been achieved. An unknown state of a Qbit is described by two complex numbers a and b taking on a continuum of values, constrained only by the requirement that $|a|^2+|b|^2 = 1$. Yet, with the aid of an entangled pair, Alice is able to provide Bob with a Qbit described by this state, at the price of only two bits of classical information (giving the results of her two measurements) and the loss of the entanglement of their carefully preserved entangled pair.

Of course the teleportation process does not communicate to Bob anything close to all the information that can be encoded in a and b. Bob is no more able to learn the values of a and b from manipulating his Qbit, now guaranteed to be in the the state $|\psi\rangle$, than Alice was able to do when it was her Qbit that was in the same state $|\psi\rangle$. On the other hand Alice's state could be produced at a crucial stage of an elaborate quantum computation, and its transference to Bob could enable him to continue with the computation on his own faraway quantum computer, so one can achieve a nontrivial objective by such teleportations.

Like dense coding, teleportation can also be constructed by manipulating an elementary classical circuit diagram, without going through any of the analysis in (6.21)-(6.24). Fig. 6.5(a) shows an elementary classical circuit that obviously works whether $|\psi\rangle = |0\rangle$ or $|\psi\rangle = |1\rangle$. As a quantum circuit it therefore works for arbitrary superpositions, $|\psi\rangle = a|0\rangle + b|1\rangle$. Such a circuit can clearly be used to transfer an unknown state $|\psi\rangle$ from Alice's Qbit (upper wire) to Bob's (lower wire). The entire teleportation protocol can be constructed by appropriately expanding the two gates in Fig. 6.5(a), with the aid of an ancillary Qbit.

In Fig. 6.5(b) the ancillary Qbit Qbit, unacted upon throughout the process, is introduced in the state

$$|\phi\rangle = \mathbf{H}|0\rangle = \tfrac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big). \tag{6.25}$$

In Fig. 6.5(c) the identities $\mathbf{X} = \mathbf{HZH}$ and $\mathbf{1} = \mathbf{HH}$ have been used to rewrite the **cX** gate on the left of Fig. 6.5(b), and an additional **cX** gate has been added on the right, which acts as the identity, since **X** acts as the identity on the state (6.25).

Fig. 6.5(d) follows from Fig. 6.5(c) because the action of **cZ** is independent of which Qbit is the control and which, the target, and because the two **cX** gates on the right of Fig. 6.5(c) have exactly the same action as the three **cX** gates on the right of Fig. 6.5(d): acting on the computational basis both sets of gates apply **X** on both of the bottom two wires if the state of the top wire is $|1\rangle$ and act as the identity if the state of the top wire is $|0\rangle$.

Fig. 6.5(e) follows from Fig. 6.5(d) by explicitly writing the $|\phi\rangle$ on the right of Fig. 6.5(d) as $\mathbf{H}|0\rangle$ and explicitly writing the $|0\rangle$ on the left of Fig. 6.5(d) as $\mathbf{H}|\phi\rangle$. But Fig. 6.5(e) is an automated version of teleportation. To see this introduce measurements (represented by the squares with M inside them) of the upper two Qbits after the circuit of Fig. 6.5(e) has acted, as in Fig. 6.5(f). Their effect is to collapse the states of each of the two upper wires randomly and independently to $|0\rangle$ or $|1\rangle$. But as noted in Chapter 4, measurement of a control Qbit commutes with any operation controlled by that Qbit, so the measurement gates can be moved to the positions they occupy in Fig. 6.5(g).

Fig. 6.5(g) is precisely the teleportation protocol. The two gates on the right transform the two lower Qbits into the entangled state (6.20). The subsequent applications to the top two Qbits of **cX** followed by **H** are exactly as described above in (6.21)-(6.23). These are followed by measurements of the top two Qbits. Since Alice knows the outcomes of the measurements, she knows whether the subsequent **cX** and **cZ** gates will or will not act, and can replace these physical couplings by a phone call to Bob telling him whether or not directly to apply **X** and/or **Z** to his own Qbit.

Figure 6.5

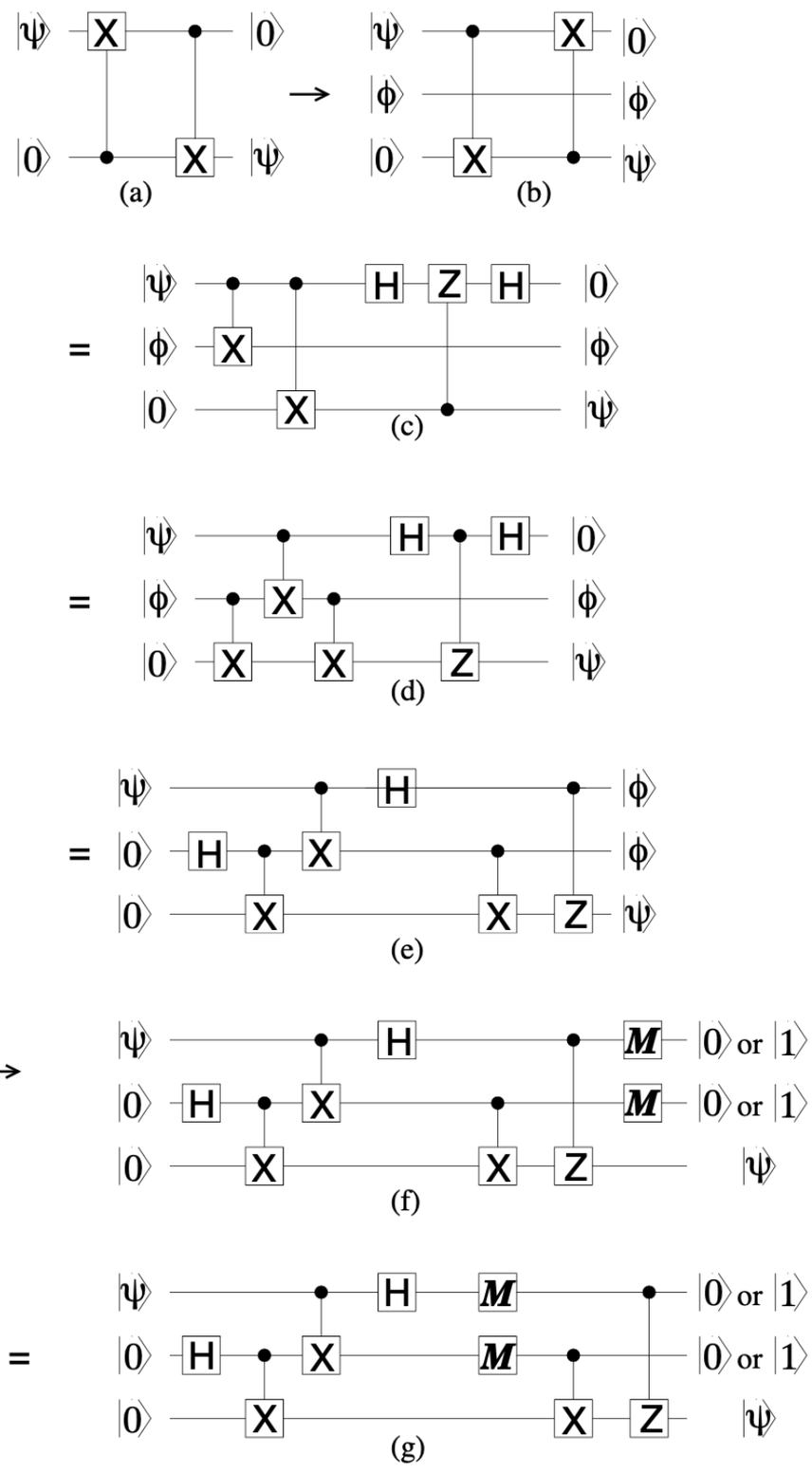


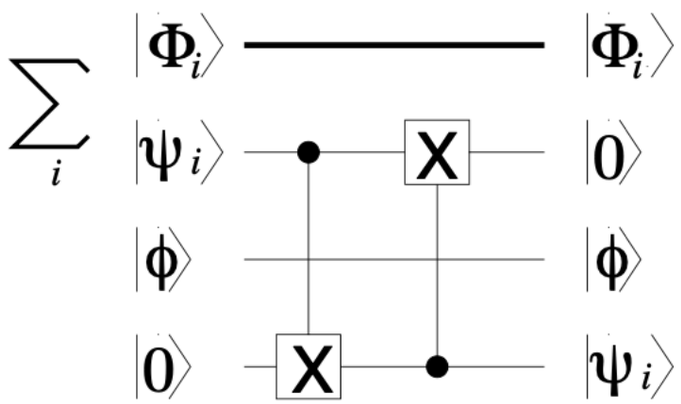
Figure 6.5. A circuit-theoretic derivation of the quantum teleportation protocol.

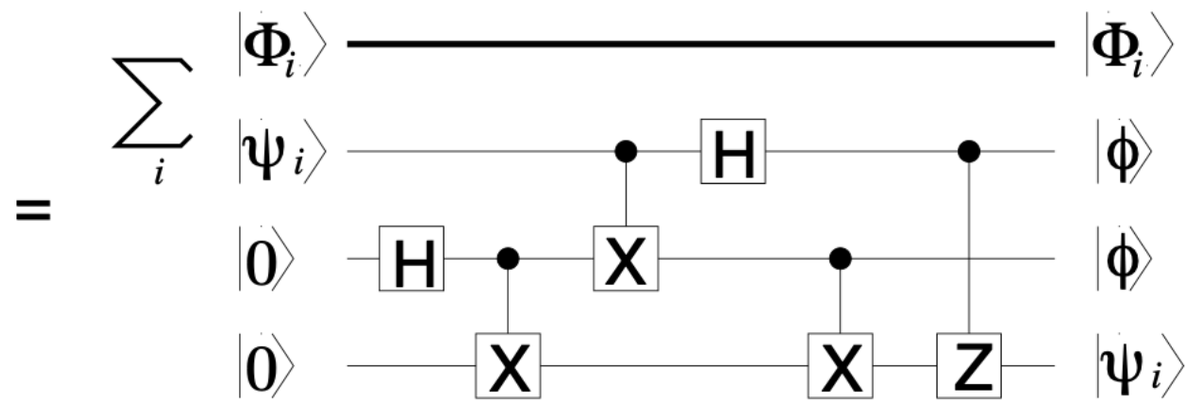
Figure 6.6 demonstrates that entanglement can also be teleported. The figure reproduces parts (b), (e), and (g) of Figure 6.5 with two changes. (1) A fourth bundle of n wires in the n-Qbit state $|\Phi\rangle_i$ has been added above each part of the figure. No operations act on these wires. (2) The state to be teleported has been given a subscript i so it is now $|\psi_i\rangle$. The steps from (b) to (e) to (g) in Figure 6.5 clearly are unaffected by these changes.

Because of the linearity of the unitary gates the identity between Fig. 6.5(a) and Fig. 6.5(b) continues to hold if one sums over many values of the index i, demonstrating that the effect of the circuits in (a) or (b) is to transfer participation in the entangled state $\sum_i |\Phi_i\rangle|\psi_i\rangle$ from the third wire from the bottom to the bottom wire. Because the final states of the second and third wires from the bottom in (b) are completely disentangled from the state of the other Qbits, the replacement of the two controlled gates on the left by measurement followed by classical communication in Fig. 6.6(c) also continues be valid.

So even if Alice's Qbit has no state of its own but is entangled with other Qbits, Alice can teleport its role in the entangled state over to Bob's Qbit resulting in Bob's Qbit being entangled in exactly the same way Alice's was, and Alice's Qbit becoming entirely unentangled. If, for example, Alice shares pairs of Qbits in the entangled state (6.20) with both Carol and Bob, then by local operations on the two Qbits in her possession and the communication of two bits of classical information, she can create the entangled state (6.20) between Carol's Qbit and Bob's. After this is done her own Qbit is completely unentangled with either of the other two.

Figure 6.6





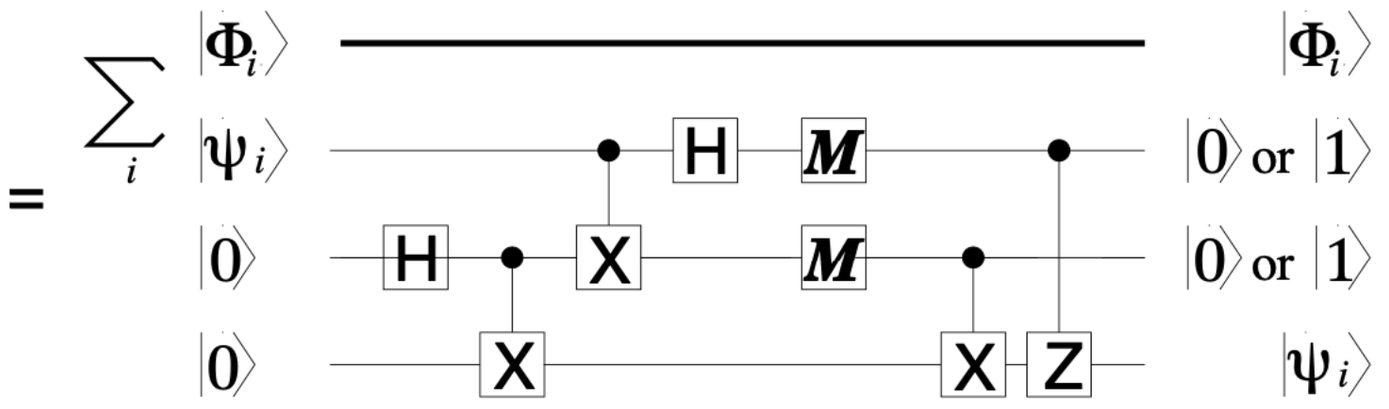


Figure 6.6. A demonstration that entanglement can be teleported.

E. The GHZ state

I conclude with an elementary but striking reminder of just how strange the behavior of Qbits can be. The possibility of such behavior was first pointed out in a rather different context by Daniel Greenberger, Michael Horen, and Anton Zeilinger ("GHZ") in the late 1980's. Here is a Qbit version:

Consider the 3-Qbit state

$$|\Psi\rangle = \tfrac{1}{2}\big((|000\rangle - |110\rangle - |101\rangle - |011\rangle\big). \tag{6.26}$$

Note that the form of $|\Psi\rangle$ is explicitly invariant under any permutation of the three Qbits. Numbering the Qbits from left to right 2, 1, and 0, one easily confirms that $|\Psi\rangle$ can be written in either of the alternative forms

$$|\Psi\rangle = \mathbf{Z}_1\mathbf{C}_{21}\mathbf{H}_2\tfrac{1}{\sqrt{2}}\big(|000\rangle + |111\rangle\big) = \mathbf{Z}_2\mathbf{C}_{12}\mathbf{H}_1\tfrac{1}{\sqrt{2}}\big(|000\rangle + |111\rangle\big) \tag{6.27}$$

(or any of the four other forms associated with permutations of the subscripts 0,1, and 2.) Since

$$\tfrac{1}{\sqrt{2}}\big(|000\rangle + |111\rangle\big) = \mathbf{C}_{21}\mathbf{C}_{20}\mathbf{H}_2|000\rangle \tag{6.28},$$

(6.27) and (6.28) provide an explicit way to construct $|\Psi\rangle$ by operating on $|0\rangle_3$ with elementary gates.

Note next that

$$\mathbf{H}_2\mathbf{H}_1|\Psi\rangle = \mathbf{H}_2\mathbf{H}_1\Big(\mathbf{Z}_1\mathbf{C}_{21}\mathbf{H}_2\tfrac{1}{\sqrt{2}}\big(|000\rangle+|111\rangle\big)\Big) = \mathbf{X}_1\big[\mathbf{H}_2\mathbf{H}_1\mathbf{C}_{21}\mathbf{H}_2\mathbf{H}_1\big]\mathbf{H}_1\tfrac{1}{\sqrt{2}}\big(|000\rangle+|111\rangle\big). \tag{6.29}$$

Since the quantity in brackets is just $\mathbf{C}_{12}$, a comparison of the forms on the right in (6.29) and (6.27) reveals that

$$\mathbf{H}_2\mathbf{H}_1|\Psi\rangle = \mathbf{Z}_2\mathbf{X}_1|\Psi\rangle. \tag{6.30}$$

Because of the invaraince of $|\Psi\rangle$ under permutation of the three Qbits we also have

$$\mathbf{H}_2\mathbf{H}_0|\Psi\rangle = \mathbf{Z}_2\mathbf{X}_0|\Psi\rangle, \tag{6.31}$$

$$\mathbf{H}_1\mathbf{H}_0|\Psi\rangle = \mathbf{Z}_1\mathbf{X}_0|\Psi\rangle. \tag{6.32}$$

Suppose we prepare three Qbits in the state $|\Psi\rangle$ using the sequence of gates specified in (6.28) and (6.27) and then allow no further interactions among them. If we now measure each Qbit, it follows from the form (6.26) that because $|\Psi\rangle$ is a superposition of computational basis states having either 0 or 2 of the Qbits in the state $|1\rangle$, the three outcomes are constrained to satisfy

$$x_2 \oplus x_1 \oplus x_0 = 0 \tag{6.33}$$

(where $\oplus$, as usual, denotes addition modulo 2.)

Suppose, on the other hand, we apply Hadamards to Qbits #2 and #1 before measuring all three. According to (6.30) this has the effect of flipping the state of Qbit #1 in the superposition (6.26) (and changing the sign of some of the terms). As a result the 3-Qbit state (6.26) is changed into a superposition of computational basis states having either 1 or 3 of the Qbits in the state $|1\rangle$. So if the outcomes are x_2^H, x_1^H and x_0, we must have

$$x_2^H \oplus x_1^H \oplus x_0 = 1. \tag{6.34}$$

Similarly if we apply Hadamards to Qbits #2 and #0 before measuring all three, the outcomes must obey

$$x_2^H \oplus x_1 \oplus x_0^H = 1, \tag{6.35}$$

and if Hadamards are applied to Qbits #1 and #0 and then all three are measured we will have

$$x_2 \oplus x_1^H \oplus x_0^H = 1. \tag{6.36}$$

Consider now the following question: If we are talking about a single trio of Qbits in the state $|\Psi\rangle$, must the x_0 appearing in (6.33) be the same as the x_0 appearing in (6.34)? A little reflection reveals that this question makes no sense. After all, (6.33) describes the outcomes of immediately measuring the three Qbits, while (6.34) describes the outcomes of measuring them after Hadamards have been applied to Qbits #2 and #1. Since only one of these two possibilities can actually be carried out, there is no way to compare the result of measuring Qbit #0 in both cases. You can't compare the x_0 you found in the case you actually carried out with the x_0 you might have found in the case you didn't carry out. It's just a stupid question.

Or is it? Suppose Qbits #2 and #1 are measured before Qbit #0. If no Hadamards were applied before those measurements then (6.33) assures us that when #0 is finally measured the result will be

$$x_0 = x_1 \oplus x_2. \tag{6.37}$$

So the outcome of measuring #0 is predetermined by the outcomes of the earlier measurements of #1 and #2. Since all interactions among the Qbits ceased after the state $|\Psi\rangle$ was prepared, subjecting #1 and #2 to measurement gates can have no effect on #0. If the outcomes of the measurements of #1 and #2 determine (as they do) the outcome of the subsequent measurement of #0, this can only be because they reveal that #0 was already predisposed to give the result (6.37) upon being itself measured. And because the Qbits do not interact, #0 must have had that predisposition even before #1 and #2 were actually measured to reveal what the result of measuring #0 would have to be.

This is a bit disconcerting, since prior to any measurements the Qbits were in the state $|\Psi\rangle$ in which none of them are individually predisposed to reveal any particular value. Indeed, we seem forced to conclude that the 3-Qbit state $|\Psi\rangle$ gives an *incomplete* description of the three Qbits. The omitted predisposition of #0 is an additional *element of reality* that a more complete description than afforded by the quantum theory would take into account.

Suppose all this is true, so that Qbit #0 has a predetermined value even before Qbits #1 and #2 are measured to reveal what that value actually is. Then that predetermined the value cannot be altered if Hadamards are applied to Qbits #1 and #2 before they are measured. For the Qbits have ceased to interact, and the predetermined value of #0 is already present before the decision whether or not to apply Hadamards has been made.

So if all this highly plausible reasoning is correct, then the value x_0 appearing in (6.33) must indeed be identical to the value of x_0 appearing in (6.34). The question is not meaningless. The answer is Yes!

Such an argument for elements of reality — predetermined values — was put forth in 1935 (in a different context) by Albert Einstein, Boris Podolsky, and Nathan Rosen (EPR). The controversy and discussion it has given rise to has steadily increased over the past 7 decades. The terms "incomplete" and "element of reality" originated with EPR. Today it is Einstein's most cited paper.

The wonderful thing about three Qbits in the GHZ state is that they not only provide a beautiful illustration of the EPR argument, but also, when examined further, their behavior reveals that that the argument for predetermined values cannot be correct. To see this note that exactly the same reasoning establishes that the values of x_1 appearing in (6.33) and (6.35) must be the same, as well as the values of x_2 appearing in (6.33) and (6.36). And the same line of thought establishes that the values of x_0^H in (6.36) and (6.35) must be the same, as well as the values of x_1^H in (6.36) and (6.34) and the values of x_2^H in (6.35) and (6.34).

If all this is true, then adding together the left sides of (6.33)-(6.36) we must get 0 modulo 2, since each of x_2, x_1, x_0, x_2^H, x_1^H, and x_0^H appears in exactly two of the equations. But the modulo 2 sum of the right sides is $0 \oplus 1 \oplus 1 \oplus 1 = 1$.

So the EPR argument doesn't work. There are no elements of reality — no predetermined values that a more complete theory would take into account. The answer to what is wrong with the simple and persuasive reasoning that led EPR to the existence of elements of reality is still a matter of heated debate after nearly 70 years. How, after all, can Qbit #0 and its measurement gate "know" that if they interact only after Qbits #1 and #2 have gone through their own measurement gates (and no Hadamards were applied) that the result of the measurement of #0 *must* be given by (6.37)? The best explanation anybody has come up with to this day is to insist that no explanation is needed beyond what one can infer from the laws of quantum mechanics.

55 pages per class roughly

Fourier Series Introduction

Many complicated functions can be represented by power series.

Another powerful way of representing such functions is using a sum of sine and cosine terms, which is called a Fourier series.

Unlike Taylor series, a Fourier series can describe functions that are not everywhere continuous and/or differentiable.

Derivation

The world is full of vibrations. The sound we hear is an acoustic wave, the things we see are electromagnetic waves and the surfers in Santa Cruz, CA ride on gravity waves of the ocean.

The simplest wave motion in 1-dimension is described by the 1-dimensional wave equation

$$\left(\frac{\partial^2}{\partial x^2} - \frac{1}{v^2}\frac{\partial^2}{\partial t^2}\right) u(x,t) = 0$$

Let us assume a solution (we will see why this is a solution later when we study partial differential equations))

$$u(x,t) = \chi(x)e^{i\omega t}$$

Substitution into the full wave equation gives an equation for the wave amplitude $\chi(x)$.

$$\left(\frac{d^2\chi(x)}{dx^2}+\frac{\omega^2}{v^2}\chi(x)\right)e^{i\omega t}=0\rightarrow\left(\frac{d^2}{dx^2}+k^2\right)\chi(x)=0\ \ ,\ \ \frac{\omega^2}{v^2}=k^2$$

This new equation(an ordinary differential equation) is easily solved(we will see why later) by the function

$$\chi(x)=a\cos kx+b\sin kx$$

which gives the shape (at fixed time t) of the vibrating system (say a string).

This is like taking a photograph of the vibrating string at one instant of time.

Let us choose a string of length π fixed at both ends.

We then have the **boundary conditions**:

$$\chi(0) = 0 = \chi(\pi)$$

which implies that

$$\chi(0) = a = 0 \quad \text{and} \quad \chi(\pi) = 0 = b \sin k\pi \rightarrow k = n = \text{ integer}$$

so that

$$\chi(x) = b \sin nx \quad , \quad n = 1, 2, 3, ...$$

which is called the nth **normal mode** of the string (given by its shape).

Each such mode has a different frequency and shape.

Historically, the wave equation was first studied in the 1700s.

In 1742, Bernoulli showed that

vibrations of different modes (frequency) could coexist in the string

In 1753, D'Alembert, Euler and Bernoulli showed that

all possible shapes of a vibrating string, even when the ends were not fixed, were representable by the series

$$f(x) = \sum_{n=1}^{\infty} b_n \sin nx$$

There was great dispute about this result.... what about the cosine series, i.e., how does one represent even functions of x?

Even though $f(x)$ solves the wave equation, others disputed the claim that this was the most general solution.

In 1807, Fourier (in a paper on heat conduction) showed that

every function in the closed interval

$$[-\pi, \pi] \quad \text{or} \quad -\pi \leq x \leq \pi$$

could be represented in the form

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx)$$

He re-derived integral formulas for the coefficients a_n, b_n had already been obtained by Euler in 1777.

Fourier, however, broke new ground by pointing out that these integral formulas were well-defined even for arbitrary functions and that the resulting coefficients were identical for different functions that agreed within the interval, but not outside it.

The paper by Fourier **was rejected** by Lagrange, Laplace and Legendre on behalf of the Academy of Sciences on the grounds that it lacked mathematical rigor.

A second version of the paper won the Academy's Grand prize in 1812.

This work has had a great impact on the development of mathematical physics in the 1800s and it is still influencing things now.

The Sine-Cosine Series

The general Fourier series expansion is sum of sine and cosine terms of the form

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos \omega_n t + \sum_{n=1}^{\infty} b_n \sin \omega_n t$$

where the frequencies

$$\omega_n = \frac{2\pi n}{T_0}$$

are integer multiples of a fundamental frequency

$$\omega_0 = \frac{2\pi}{T_0}$$

and T_0 is determined either by the natural periodicity of $f(t)$ or possibly by an enforced periodicity of some sort.

Different treatments of this subject can have definitions of T_0 which differ by various factors.

The results of all calculations are the same.

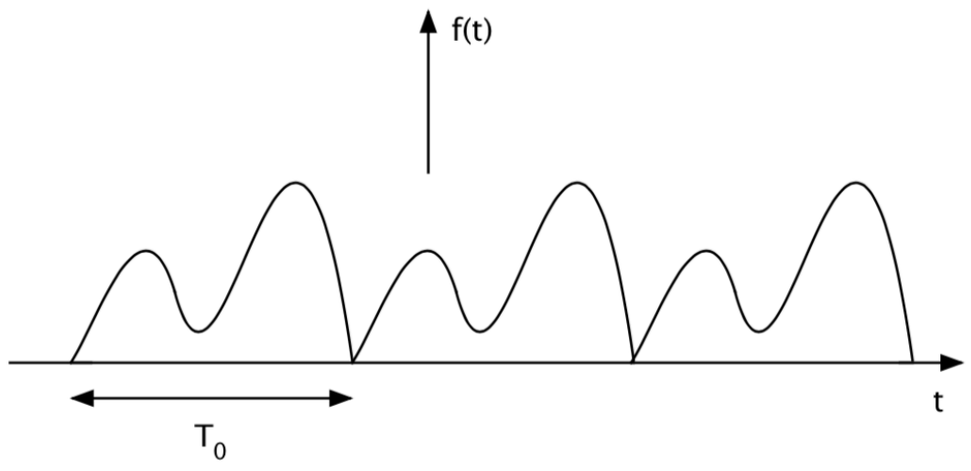
The assumed form guarantees that $f(t)$ has periodicity T_0, i.e.,

$$f(t) = f(t + T_0) \quad \text{for all} \quad t$$

In terms of the independent variable t, $f(t)$ has this periodicity for the entire interval

$$-\infty < t < \infty$$

as shown below



The typical Fourier series problem is such that we are given a function $g(t)$ and we then determine both T_0 and the a_n and b_n coefficients so that the series expansion is equal to $g(t)$ for all t.

This requires that

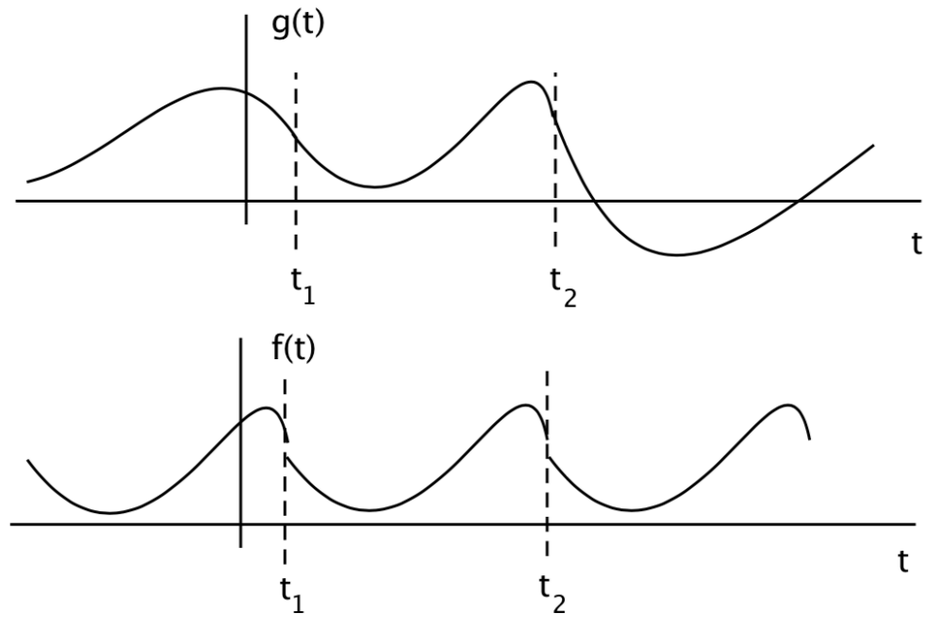
$$g(t) = g(t + T_0) \quad \text{for all} \quad t$$

The **smallest value** of T_0 that satisfies this equation is the **period** of $g(t)$.

What if $g(t)$ is not periodic?

In this case we cannot use a general Fourier series to represent $g(t)$ for **all** t.

On the other hand, we can make the series equal to $g(t)$ for some finite interval as shown below



Clearly, $g(t)$ above is not periodic.

Suppose however we define the basic period for the Fourier series to be $T_0 = t_2 - t_1$ where the interval $t_1 < t < t_2$ is as shown.

The Fourier series $f(t)$ can then be made identical to $g(t)$ in that interval.

Outside the interval, the Fourier series $f(t)$ is periodic and will not match $g(t)$ as shown.

Digression: The formal mathematical requirements that a function $f(x)$ must satisfy in order that it may be expanded in a Fourier series are known as the **Dirichlet conditions**, which are summarized as follows:

- the function must be periodic
- it must be single-valued and continuous, except possibly at a finite number of finite discontinuities
- it must have only a finite number of maxima and minima within one period
- the integral over one period of $|f(x)|$ must converge

The Orthogonality Conditions

We can determine the coefficients a_n and b_n using so-called **orthogonality conditions** (proved using calculus)

$$\int_{t_0}^{t_0+T_0} dt\, \sin\left(\frac{2\pi n}{T_0}t\right) \sin\left(\frac{2\pi m}{T_0}t\right) = \delta_{nm}\frac{T_0}{2}$$

$$\int_{t_0}^{t_0+T_0} dt\, \cos\left(\frac{2\pi n}{T_0}t\right) \cos\left(\frac{2\pi m}{T_0}t\right) = \delta_{nm}\frac{T_0}{2}$$

$$\int_{t_0}^{t_0+T_0} dt\, \sin\left(\frac{2\pi n}{T_0}t\right) \cos\left(\frac{2\pi m}{T_0}t\right) = 0$$

for **integer** values of m and n.

In addition we have

$$\int_{t_0}^{t_0+T_0} dt\, \sin\left(\frac{2\pi n}{T_0}t\right) = 0 \quad , \quad \int_{t_0}^{t_0+T_0} dt\, \cos\left(\frac{2\pi n}{T_0}t\right) = \delta_{n0}\, T_0$$

We can now evaluate the coefficients as follows.

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos \omega_n t + \sum_{n=1}^{\infty} b_n \sin \omega_n t$$

The integral operation

$$\int_{t_0}^{t_0+T_0} dt\, f(t) = \int_{t_0}^{t_0+T_0} dt\, \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \int_{t_0}^{t_0+T_0} dt\ \cos \omega_n t$$

$$+ \sum_{n=1}^{\infty} b_n \int_{t_0}^{t_0+T_0} dt\ \sin \omega_n t$$

$$= \int_{t_0}^{t_0+T_0} dt\, \frac{a_0}{2} = \frac{a_0 T_0}{2}$$

determines a_0, i.e.,

$$a_0 = \frac{2}{T_0} \int_{t_0}^{t_0+T_0} dt\, f(t)$$

The integral operation

$$\int_{t_0}^{t_0+T_0} dt\ \cos\omega_m t f(t) = \int_{t_0}^{t_0+T_0} dt\ \cos\omega_m t \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \int_{t_0}^{t_0+T_0} dt\ \cos\omega_m t \cos\omega_n t + \sum_{n=1}^{\infty} b_n \int_{t_0}^{t_0+T_0} dt\ \cos\omega_m t \sin\omega_n t$$

$$= \sum_{n=1}^{\infty} a_n \int_{t_0}^{t_0+T_0} dt\ \cos\omega_m t \cos\omega_n t$$

$$= \sum_{n=1}^{\infty} a_n \delta_{nm} \frac{T_0}{2} = a_m \frac{T_0}{2}$$

determines a_m, $m > 0$, i.e.,

$$a_m = \frac{2}{T_0} \int_{t_0}^{t_0+T_0} dt\ \cos\omega_m t f(t)$$

The integral operation

$$\int_{t_0}^{t_0+T_0} dt\ \sin\omega_m t f(t) = \int_{t_0}^{t_0+T_0} dt\ \sin\omega_m t \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \int_{t_0}^{t_0+T_0} dt\ \sin\omega_m t \cos\omega_n t + \sum_{n=1}^{\infty} b_n \int_{t_0}^{t_0+T_0} dt\ \sin\omega_m t \sin\omega_n t$$

$$= \sum_{n=1}^{\infty} b_n \int_{t_0}^{t_0+T_0} dt\ \sin\omega_m t \sin\omega_n t$$

$$= \sum_{n=1}^{\infty} b_n \delta_{\boldsymbol{nm}} \frac{T_0}{2} = b_m \frac{T_0}{2}$$

determines b_m, $m > 0$, i.e.,

$$b_m = \frac{2}{T_0} \int_{t_0}^{t_0+T_0} dt\ \sin\omega_m t f(t)$$

Several questions arise:

(1) Do these Fourier coefficients exist?

(2) Is the Fourier series convergent?

(3) Does it converge to the original function?

The answer is **YES for all physically realizable systems!!!!**

Examples:

(1) The function $f(t)$ is

$$f(t) = \sin(t)$$

This function is periodic with period $T_0 = 2\pi \rightarrow \omega_n = n$.

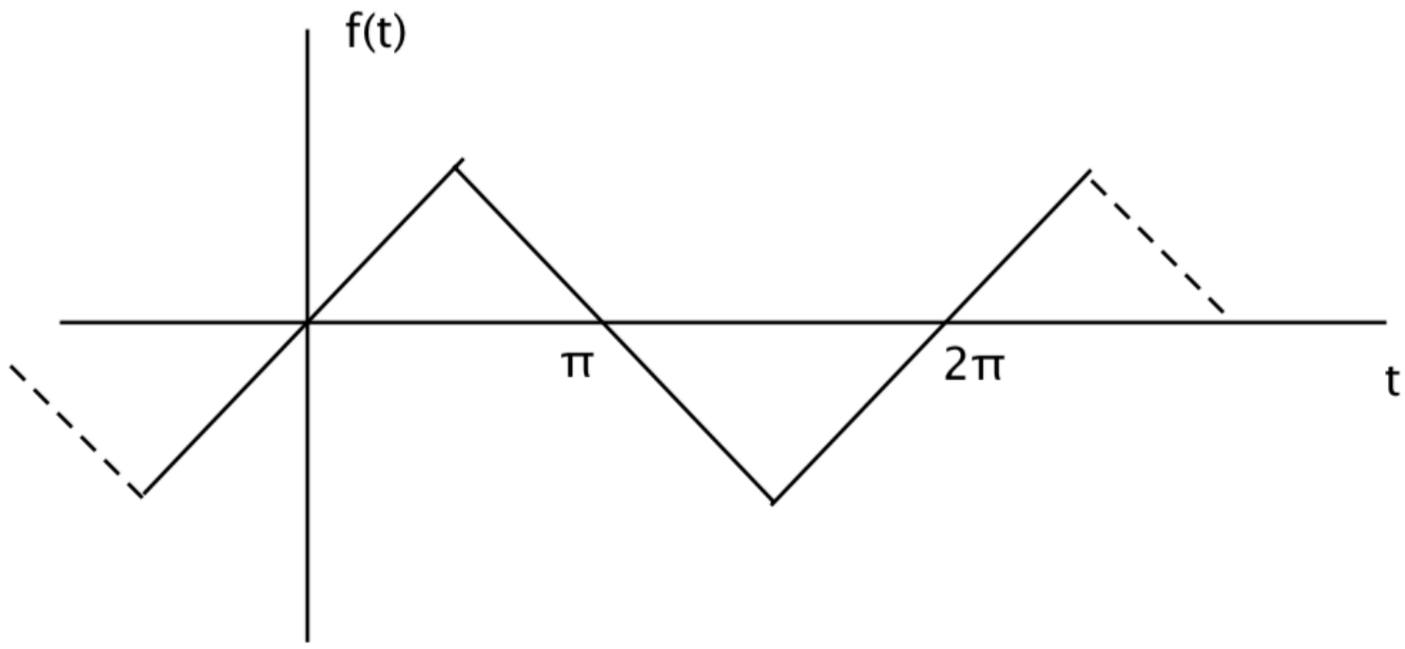
Therefore the Fourier series looks like

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \sin nt + \sum_{n=1}^{\infty} b_n \cos nt$$

This example requires no calculations. It is clear that

$$a_m = 0 \quad \text{for all} \quad m \quad , \quad b_m = 0 \quad \text{for all} \quad m \neq 1 \quad , \quad b_1 = 1$$

(2) The function $f(t)$ equals the periodic triangle wave shown above.



In the interval $-\pi/2 < t < 3\pi/2$ this has the functional form

$$f(t) = \begin{cases} \frac{2t}{\pi} & -\frac{\pi}{2} \leq t \leq \frac{\pi}{2} \\ 2 - \frac{2t}{\pi} & \frac{\pi}{2} \leq t \leq \frac{3\pi}{2} \end{cases}$$

Once again the basic period is $T_0 = 2\pi$ which again gives $\omega_n = n$.

We have

$$a_m = \frac{2}{T_0} \int_{-\pi/2}^{3\pi/2} dt \, \cos \omega_n t f(t) = 0 \quad \text{for all} \quad m$$

since the integrand is the **product of an odd and an even function**.

We also have

$$b_m = \frac{2}{T_0} \int_{-\pi/2}^{3\pi/2} dt \, \sin \omega_n t f(t) = \frac{8}{\pi^2 n^2} \sin \frac{n\pi}{2}$$

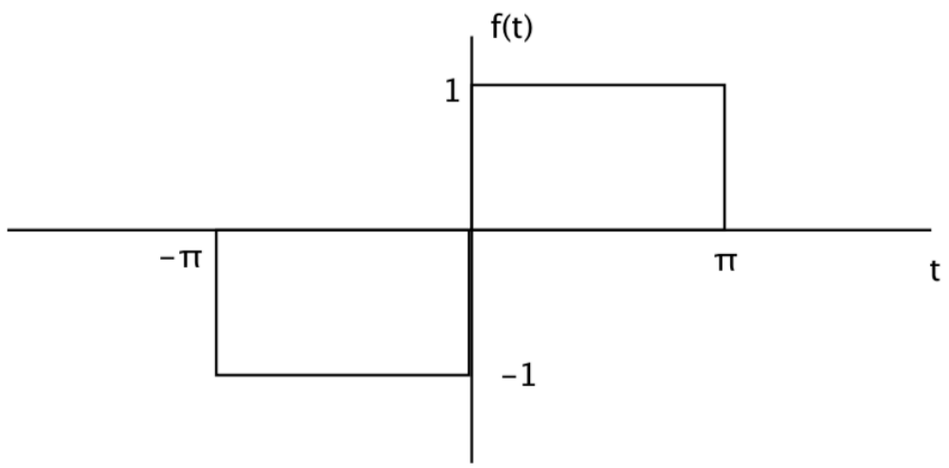
Therefore, we get

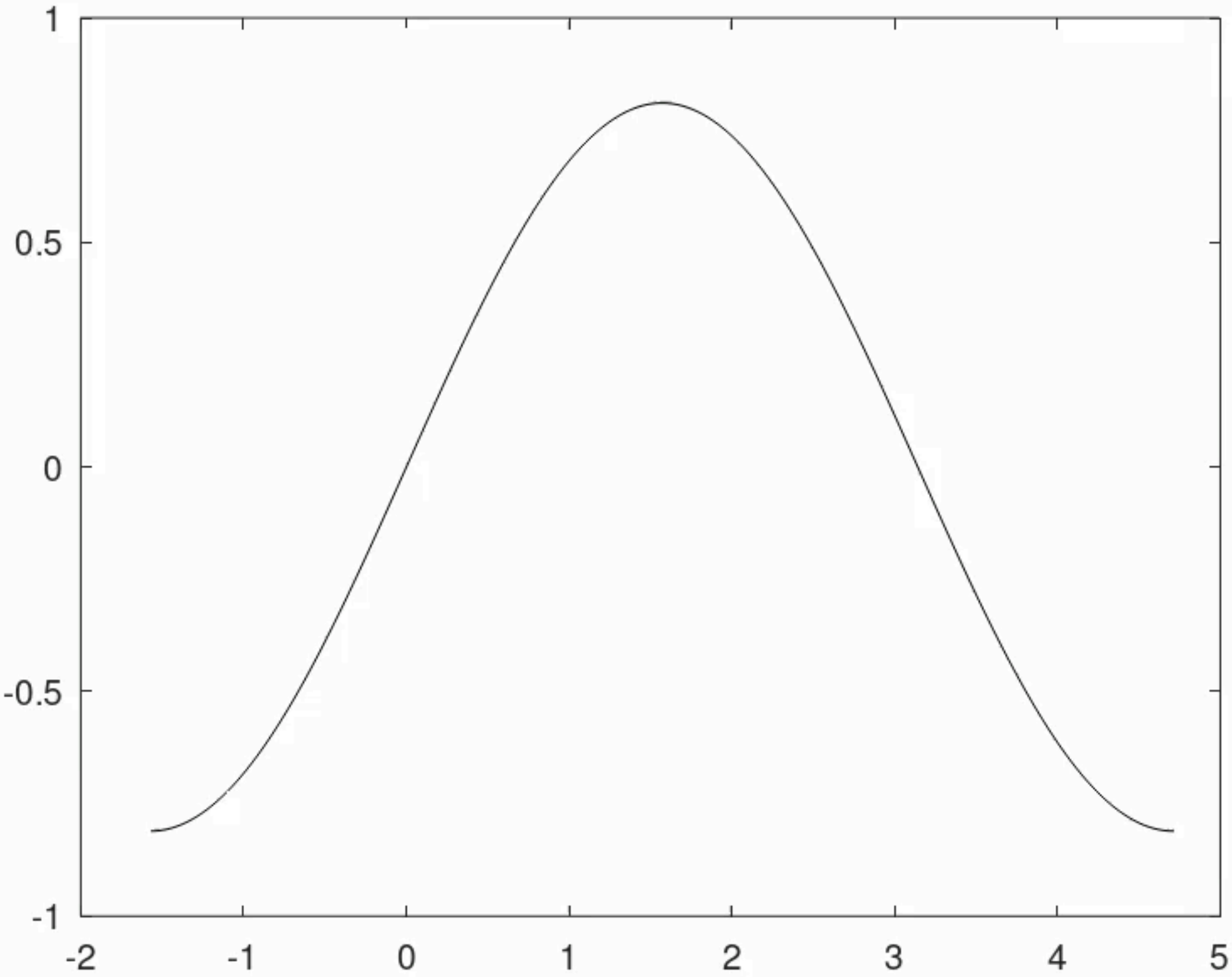
$$f(t) = \sum_{n=1,odd} \frac{8}{\pi^2 n^2} \sin \frac{n\pi}{2} \sin nx$$

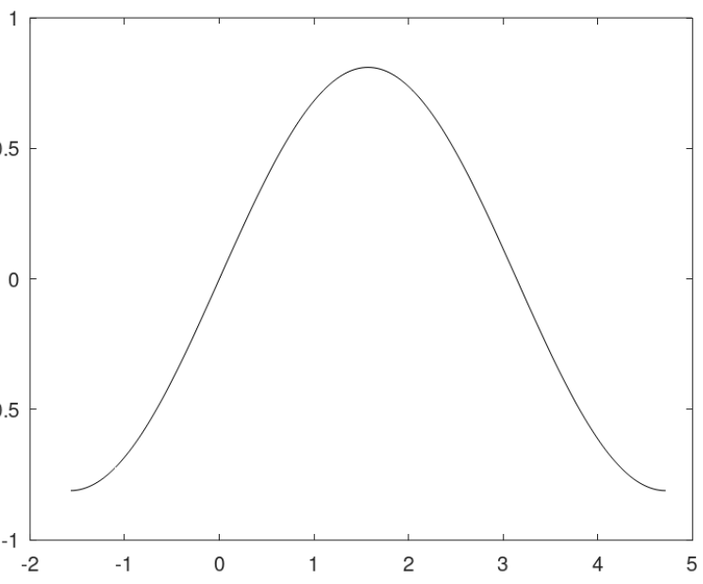
Illustrate in class with Octave program:

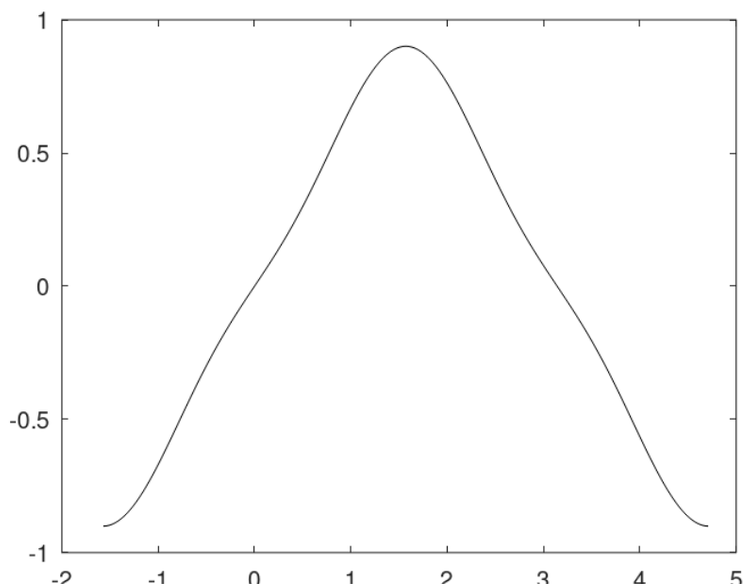
```
% illustrate Fourier Series
% triangle wave
% gives sequence of images building
% up to final curve
x=-pi/2:pi/100:3*pi/2;
val = zeros(1,length(x));
figure(1);
for m=1:2:100
 f=8/(m*pi)^2*sin(m*pi/2)*sin(m*x);
 val=val+f; m
 clf;
 plot(x,val, '-k');
  axis([-pi/2 3*pi/2 -1.5 1.5]);
 hold on;
 pause(1);
end
```

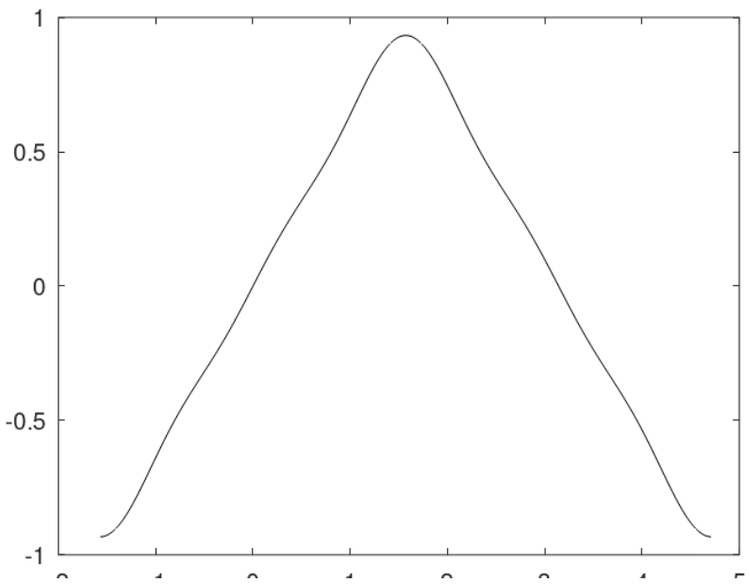
(3) $f(t)$ is the square wave function

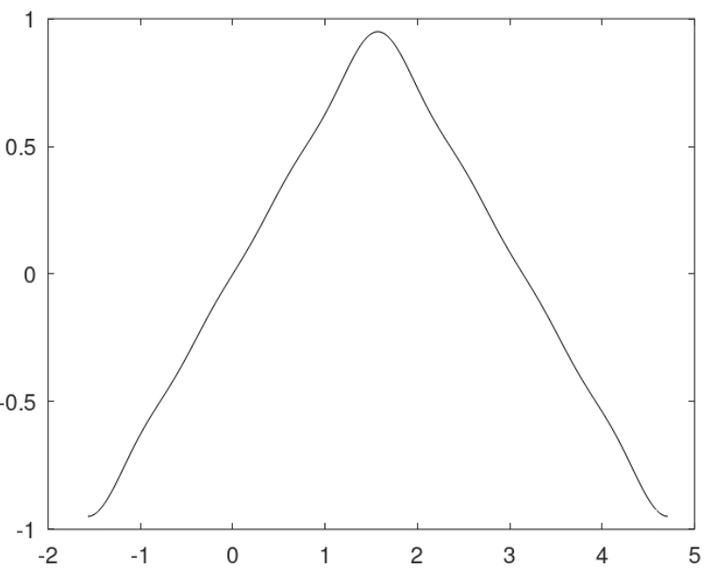


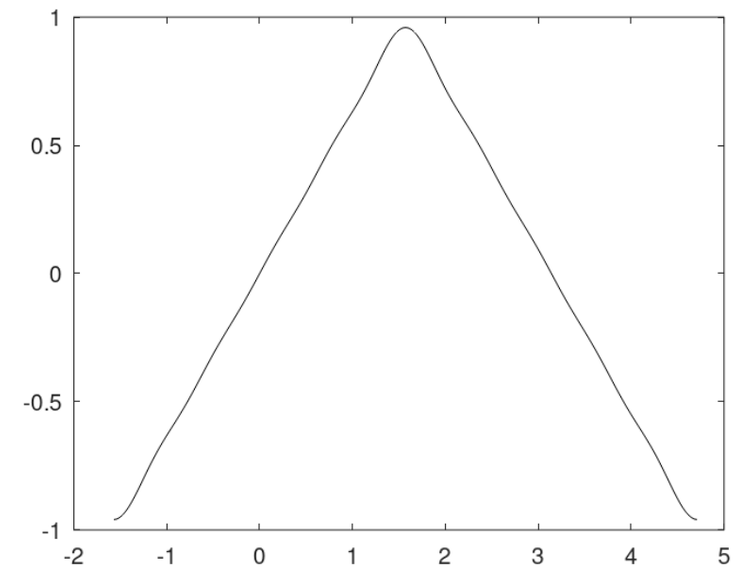

1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5

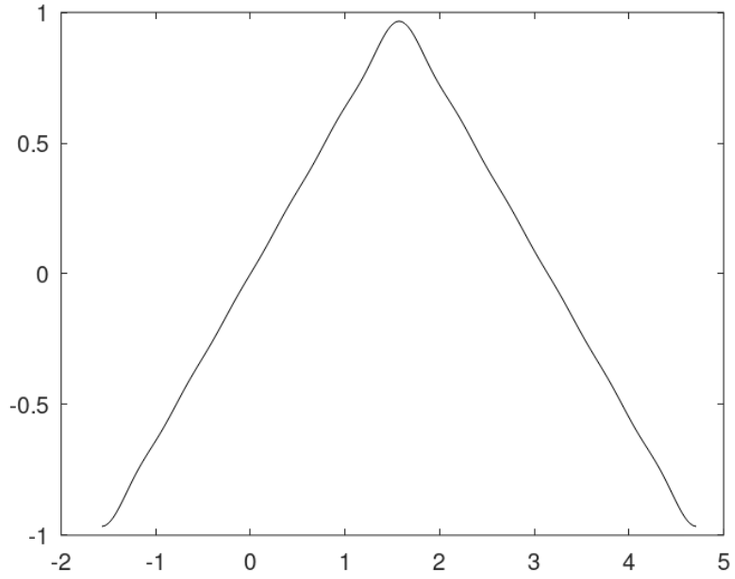

1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5

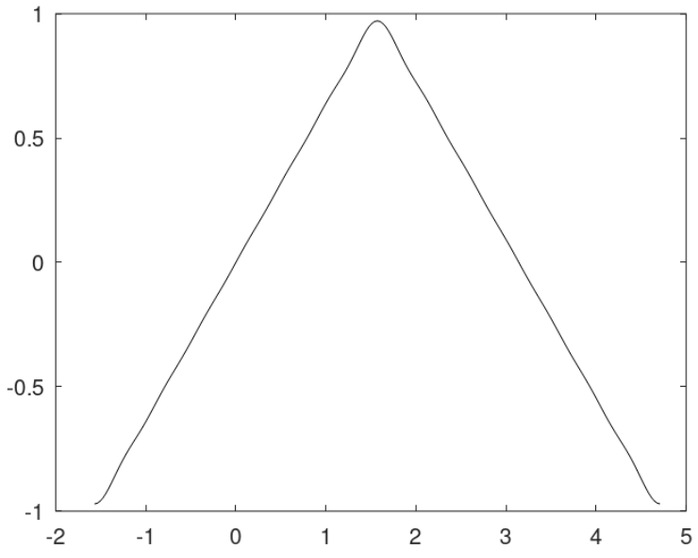

1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5

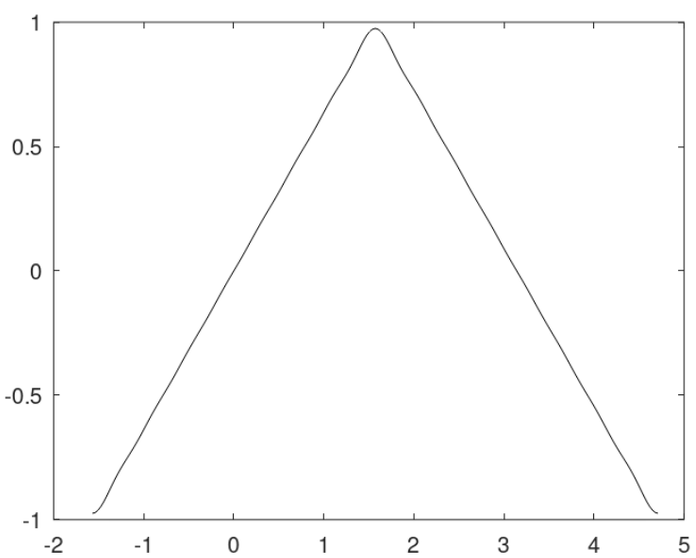

1
0.5
0
-0.5
-1

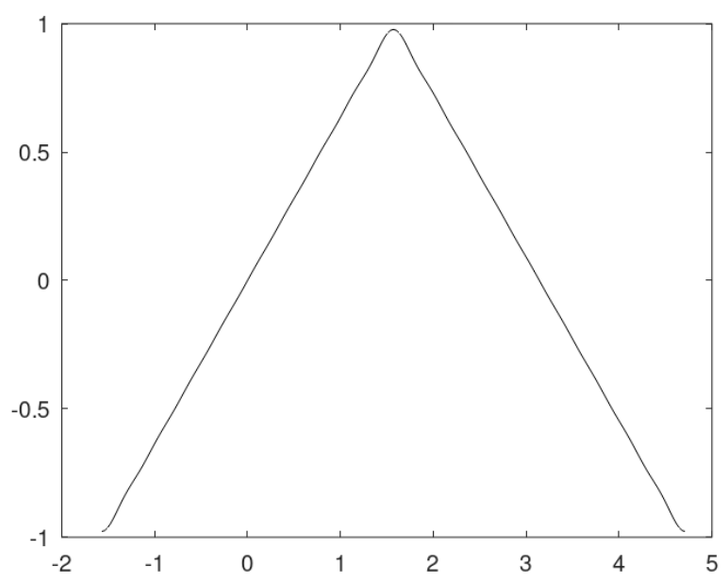

1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5


1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5


1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5


1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5


1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5


1
0.5
0
-0.5
-1
-2
-1
0
1
2
3
4
5

In the interval $[-\pi, \pi]$ we have

$$f(t) = \begin{cases} +1 & 0 \leq t \leq \pi \\ -1 & -\pi \leq t \leq 0 \end{cases}$$

Once again the basic period is $T_0 = 2\pi$ which again gives $\omega_n = n$.

Again this is an odd function so all the $a_n = 0$ and we find

$$b_n = \begin{cases} \frac{4}{n\pi} & n = 1, 3, 5, 6, ... \\ 0 & n = 2, 4, 6, 8, ... \end{cases}$$

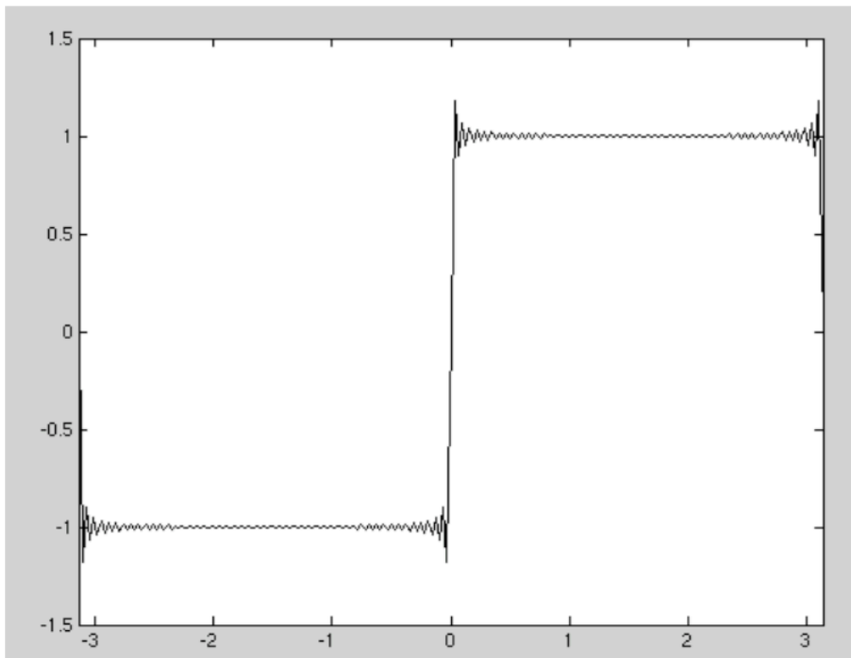
which gives

$$f(t) = \frac{4}{\pi} \sum_{n=odd} \frac{sinnt}{n}$$

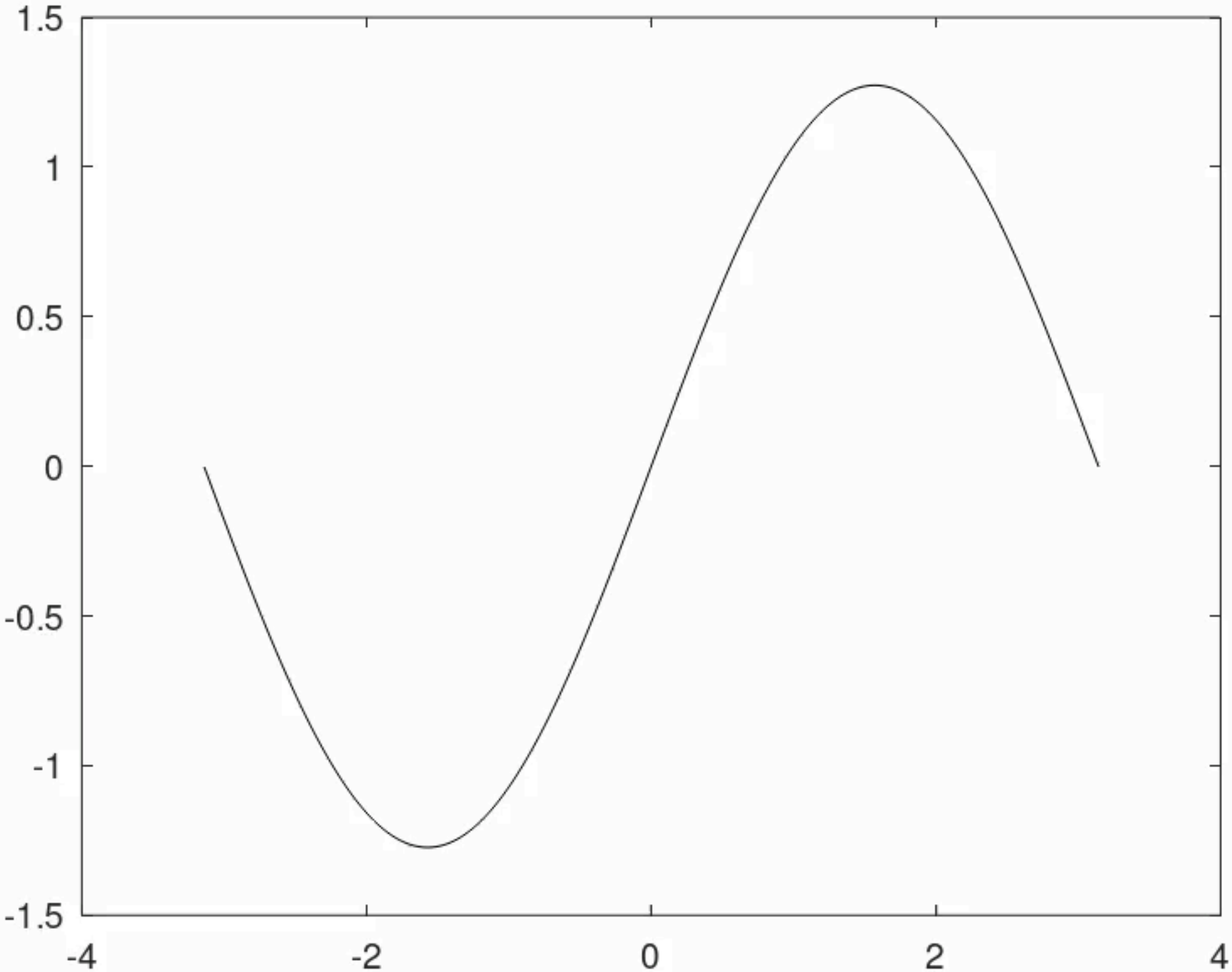
Illustrate in class with Octave program:

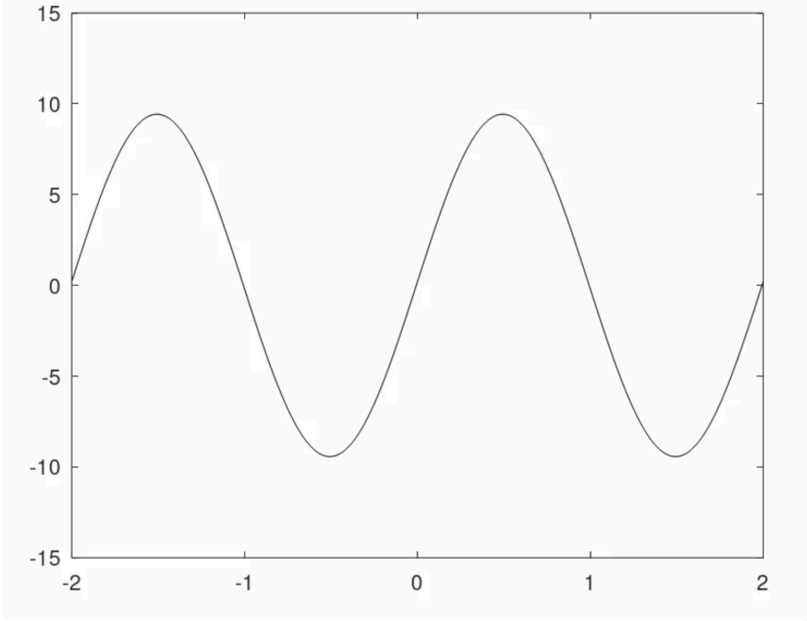
```
% illustrate Fourier Series for square wave pulse
% gives sequence of images building up to final curve x=-pi:pi/100:pi;
val = zeros(1,length(x));
figure(1);
for m=1:2:500
 f=4/(m*pi)*sin(m*x);
 val=val+f;
 m
 clf;
 plot(x,val,'-k');
 axis([-pi pi -1.5 1.5]);
 hold on;
 pause(1);
end
```

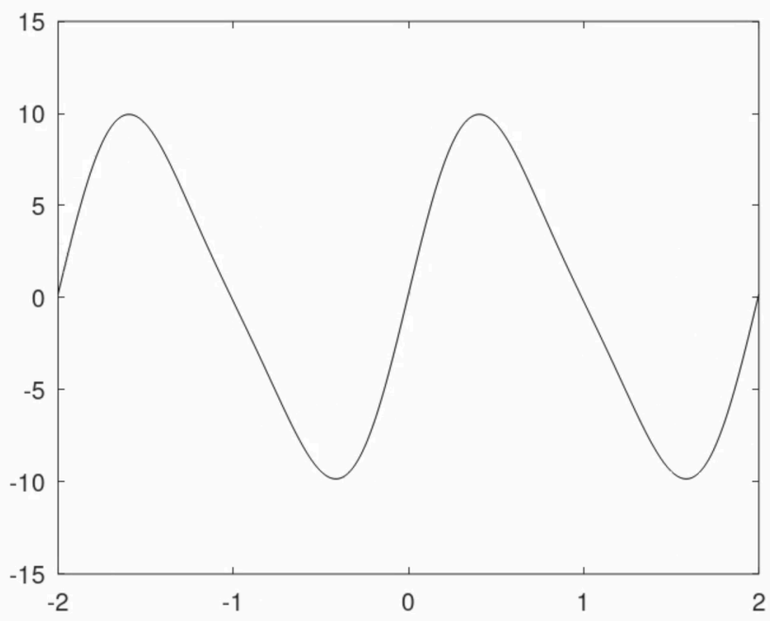
It looks like the figure below after including a 100 terms in the sum....

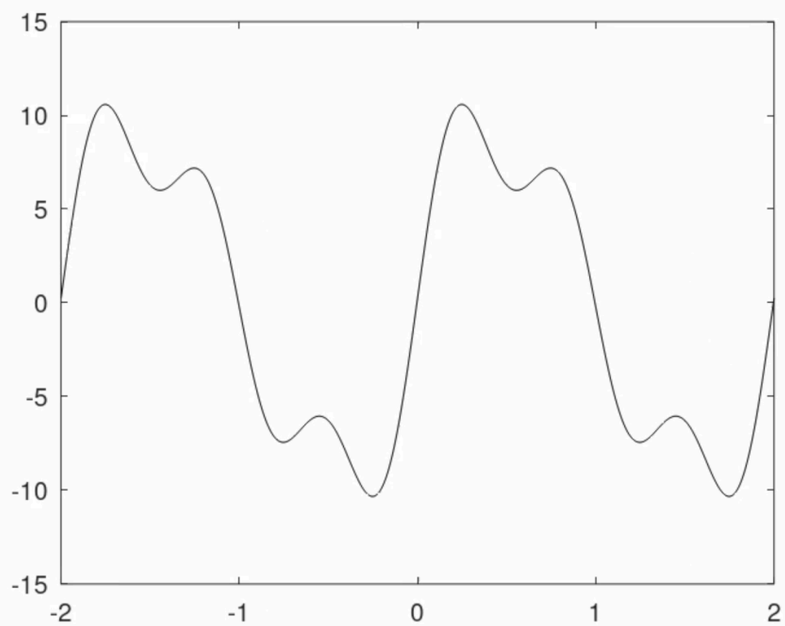


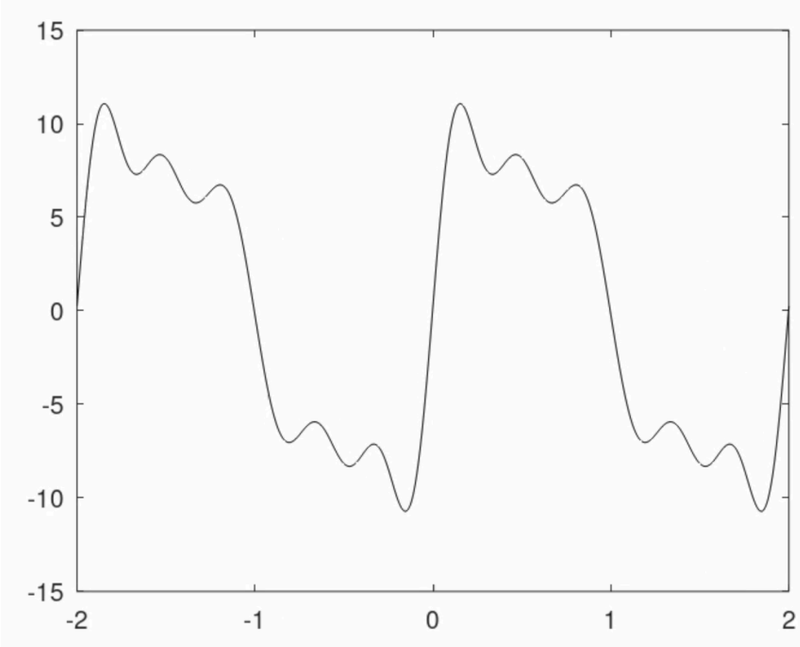
We will do better in class.

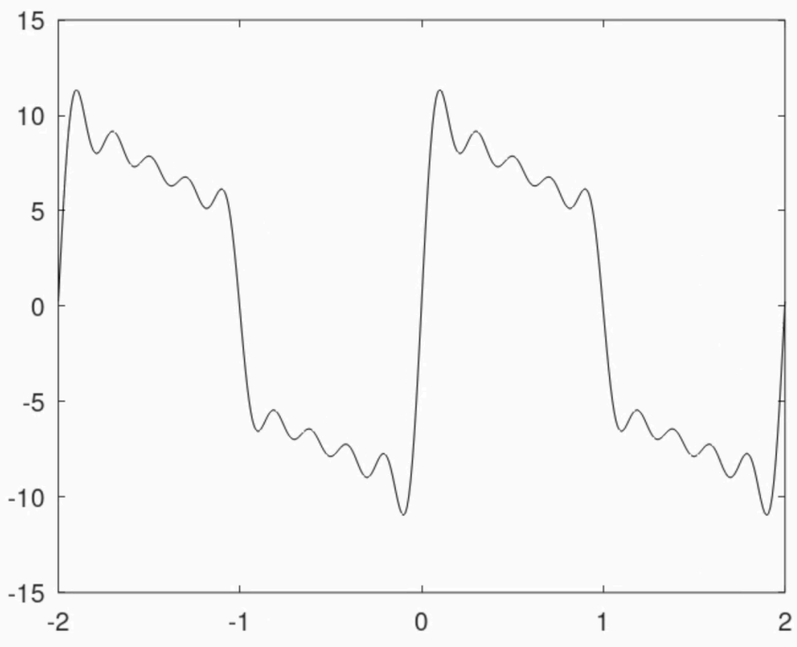

1.5
1
0.5
0
-0.5
-1
-1.5
-4
-2
0
2
4

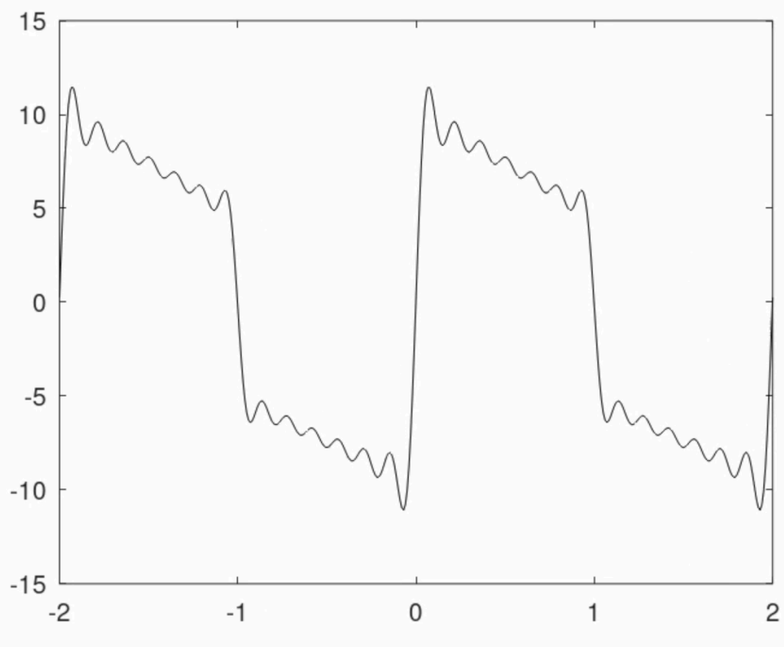




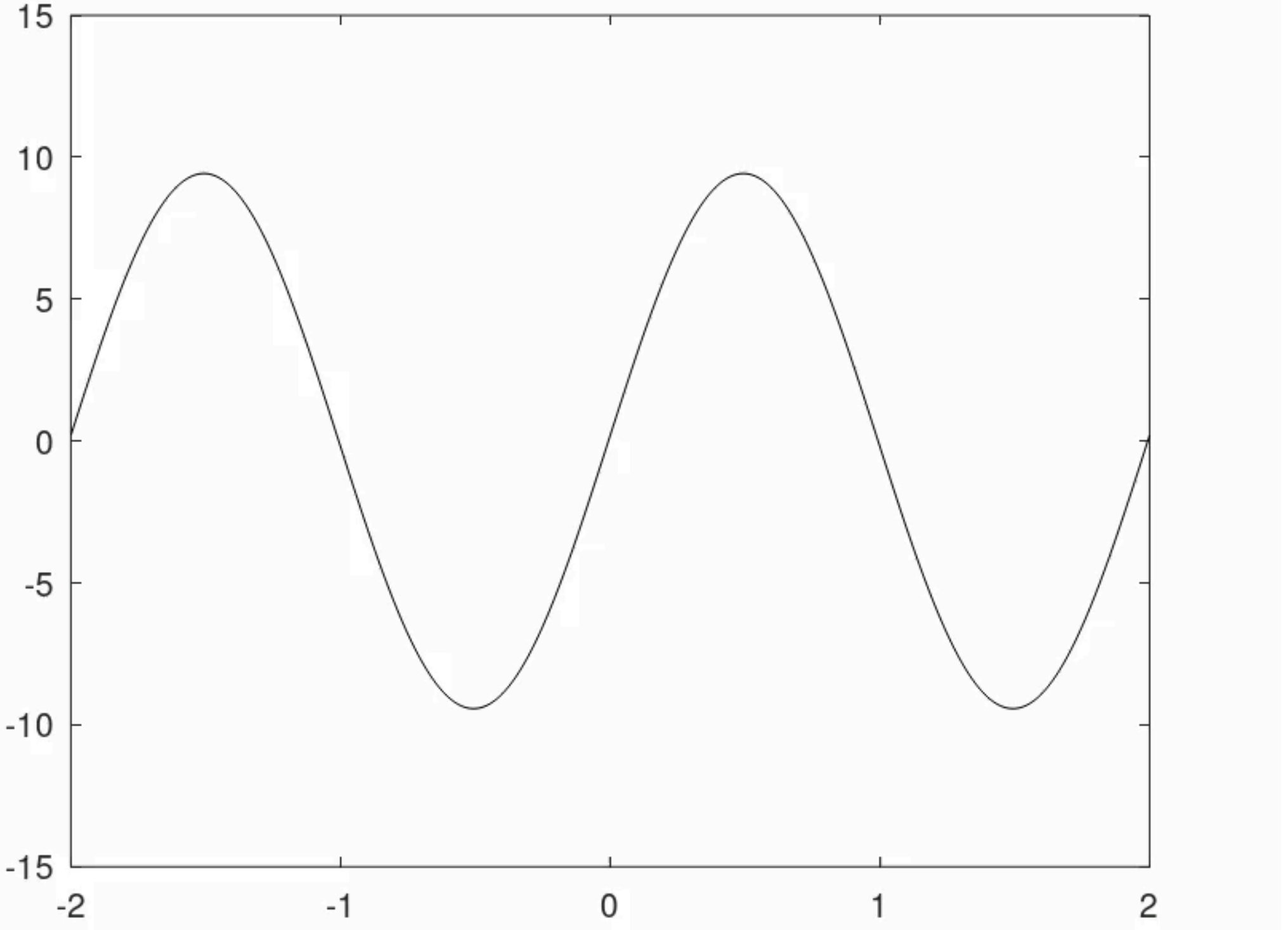








another example - first 6 slide because it happens fast


15
10
5
0
-5
-10
-15
-2
-1
0
1
2

Even and Odd Stuff

We can always rewrite any function as:

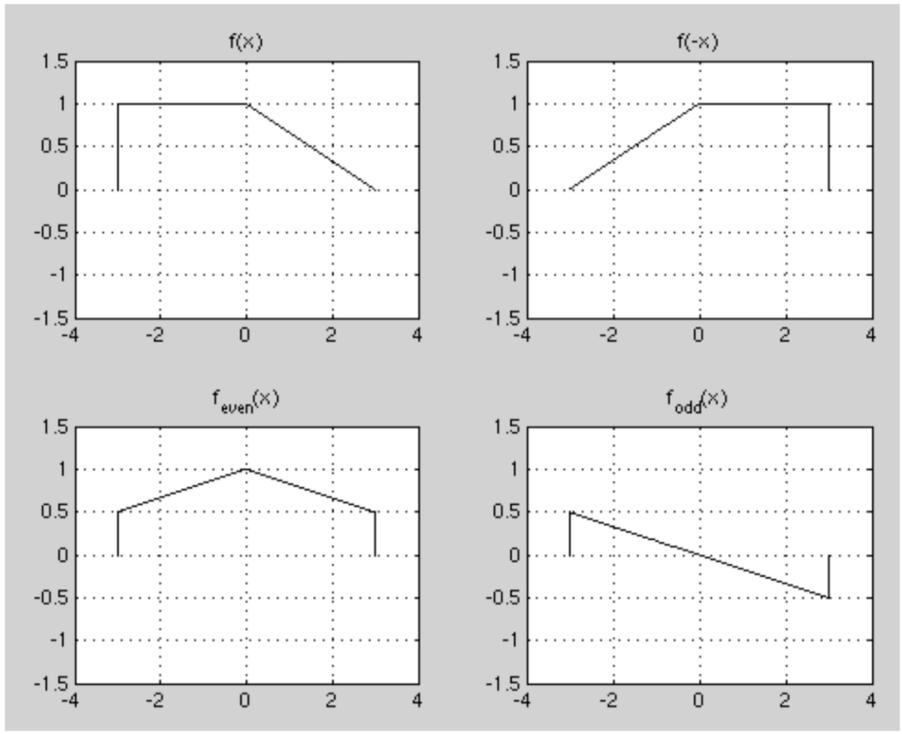
$$f(x) = \frac{1}{2}[f(x) + f(-x)] + \frac{1}{2}[f(x) - f(-x)] = f_{even}\ (x) + f_{odd}(x)$$

We can then use Fourier series to write:

$$f_{even}\ (x) = \frac{a_0}{2} \sum_n a_n \cos nx \quad , \quad f_{odd}(x) = \sum_n b_n \sin nx$$

where we have assumed that the function has a period $T_0 = 2\pi$.

Example:



Now we allow for an **Arbitrary Interval**.

If $f(x)$ is defined for an interval $[-L, L]$ of length(period) $2L$ instead of the standard interval of length (period) 2π, then a simple change of variable and integration range deals with the problem.

We have

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos \frac{2\pi n}{T_0} t + \sum_{n=1}^{\infty} b_n \sin \frac{2\pi n}{T_0} t$$

with

$$a_0 = \frac{2}{T_0} \int_{-T_0/2}^{T_0/2} dt\, f(t)\ ,\ a_n = \frac{2}{T_0} \int_{-T_0/2}^{T_0/2} dt\ \cos \frac{2\pi nt}{T_0} f(t)$$

$$b_n = \frac{2}{T_0} \int_{-T_0/2}^{T_0/2} dt\ \sin \frac{2\pi nt}{T_0} f(t)$$

We let $T_0 = 2L$ to get

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos \frac{\pi n}{L} x + \sum_{n=1}^{\infty} b_n \sin \frac{\pi n}{L} x$$

with

$$a_0 = \frac{1}{L}\int_{-L}^{L} dx\, f(x)\ ,\ a_n = \frac{1}{L}\int_{-L}^{L} dx\, \cos\frac{\pi n x}{L} f(x)$$

$$b_n = \frac{1}{L}\int_{-L}^{L} dx\, \sin\frac{\pi n x}{L} f(x)$$

Another Example:

Suppose

$$f(x) = \begin{cases} 0 & -\pi < x < 0 \\ h & 0 < x < \pi \end{cases}$$

The is a square pulse(wave).

We might imagine this is a signal being sent into some electronic apparatus.

We can calculate its Fourier coefficients

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} dx\ f(x)\frac{1}{\pi}\int_0^{\pi} h dx = h$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} dx\ \cos nx f(x) = 0 \quad \text{for all} \quad n$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} dx\ \sin nx f(x) = \frac{h}{n\pi}(1-\cos n\pi) = \begin{cases} 2h/n\pi & n \ \text{odd} \\ 0 & n \ \text{even} \end{cases}$$

which gives

$$f(x) = \frac{h}{2} = \frac{2h}{\pi}\left[\frac{\sin x}{1} + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \cdots\right]$$

We note that the terms fall off only as $1/n$, which implies physically that a square wave contains lots of **high frequency components**.

This implies that if an electronic apparatus does not pass high-frequency components well, the square wave input will emerge with corners rounded off in the best case and, possibly, an amorphous blob in the worst case.

and the corresponding Fourier series is

$$f(x) = \frac{4}{\pi}\sin(\pi x) + \frac{4}{3\pi}\sin(3\pi x) + \frac{4}{5\pi}\sin(5\pi x) + ...$$

Such expansion can also be made in terms of other special functions such as Bessel functions, etc and we will use this fact to great advantage when solving partial differential equations. **F**

Fourier Transforms

Transforms The complex Fourier series has an important limiting form when the period approaches infinity, i.e., $T_0 \to \infty$ or $L \to \infty$.

Suppose that in this limit

(1) $k = \frac{n\pi}{L}$ remains large (ranging from $-\infty$ to ∞) and

(2) $c_n \to 0$ since it is proportional to L, but

$$g(k) = \lim_{\substack{L\to\infty \\ c_n\to 0}} \frac{L}{\pi} c_n = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx = \quad \text{finite}$$

then we have

$$f(x) = \sum_{-\infty}^{\infty} c_n e^{ikx} = \lim_{\substack{L\to\infty \\ c_n\to 0}} \frac{1}{\sqrt{2\pi}} \sum_{-\infty}^{\infty} \frac{\pi}{L} g(k) e^{ikx}$$

where $k - \frac{n\pi}{L}$.

The sum over n is in steps of $\Delta n = 1$.

Thus, we can write using

$$\Delta k = \frac{\pi}{L} \Delta n$$

which becomes infinitesimally small when L becomes large, as a sum over k, which becomes an integral in the limit

$$\begin{aligned} f(x) &= \lim_{\substack{L\to\infty \\ c_n\to 0}} \frac{1}{\sqrt{2\pi}} \sum_{-\infty}^{\infty} \frac{\pi \Delta n}{L} g(k) e^{ikx} = \lim_{\Delta k\to 0} \frac{1}{\sqrt{2\pi}} \sum_{-\infty}^{\infty} \Delta k g(k) e^{ikx} \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(k) e^{ikx} dk \end{aligned}$$

We call $g(k)$ the Fourier Transform of $f(x)$

$$g(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx = F(f)$$

and the last equation is the so-called Fourier inversion formula.

We can now obtain an integral representation of the delta-function.

This corresponds to the orthogonality condition for the complex exponential Fourier series.

We substitute the definition of $g(k)$ into the inversion formula to get

$$
\begin{aligned}
f(x) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(k) e^{ikx} dk \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dk e^{ikx} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x') e^{-ikx'} dx' \\
&= \int_{-\infty}^{\infty} dx' f(x') \left[\frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ik(x-x')} dk \right] \\
&= \int_{-\infty}^{\infty} dx' f(x') \delta(x - x')
\end{aligned}
$$

where

$$
\delta(x - x') = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ik(x-x')} dk
$$

Properties

The evaluation of the integrals involved in many Fourier transforms involves complex integration, which we shall learn later.

We will just state some properties

Examples:

(1) The Fourier transform of the box function

$$f(x) = \begin{cases} 1 & |x| \leq a \\ 0 & |x| \geq a \end{cases}$$

is

$$F(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx = \frac{1}{\sqrt{2\pi}} \int_{-a}^{a} e^{-ikx} dx$$
$$= \frac{1}{\sqrt{2\pi}} \left. \frac{e^{-kx}}{ik} \right|_{-a}^{a} = \frac{1}{\sqrt{2\pi}} \frac{2 \sin ka}{k}$$

(2) The Fourier transform of the derivative of a function is

$$\begin{aligned} F\left(\frac{df}{dx}\right) &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\frac{df(x)}{dx}e^{-ikx}dx \\ &= \frac{1}{\sqrt{2\pi}}\left[f(x)e^{-ikx}\Big|_{-\infty}^{\infty} - (-ik)\int_{-\infty}^{\infty}f(x)e^{-ikx}dx\right] \\ &= \frac{ik}{\sqrt{2\pi}}\int_{-\infty}^{\infty}f(x)e^{-ikx}dx = ikF(f) \end{aligned}$$

where we have assumed that $f(x) \to 0$ as $|x| \to \infty$.

This generalizes to

$$F\left(\frac{d^n f}{dx^n}\right) = (ik)^n F(f)$$

(3) Other useful properties of the Fourier transform are:

$$F(f(x)) = g(k) \quad , \quad F(f(x-a)) = e^{-ika} g(k) \quad , \quad F(f(x)e^{ax}) = g(k+ia)$$

A short table of Fourier Transforms is shown below:

$f(x)$	$g(k)$
$\delta(x)$	$\frac{1}{\sqrt{2\pi}}$
$\begin{cases} 0 & x>0 \\ e^{-\lambda x} & x<0 \end{cases}$	$\frac{1}{\sqrt{2\pi}} \frac{1}{a+ik}$
$e^{-\frac{cx^2}{2}}$	$\frac{1}{\sqrt{c}} e^{-\frac{k^2}{2c}}$
$\frac{1}{1+x^2}$	$\sqrt{\frac{\pi}{2}} e^{-\lvert k \rvert}$

Fourier Transform Examples:

(1) **The Square Pulse** - Consider the function

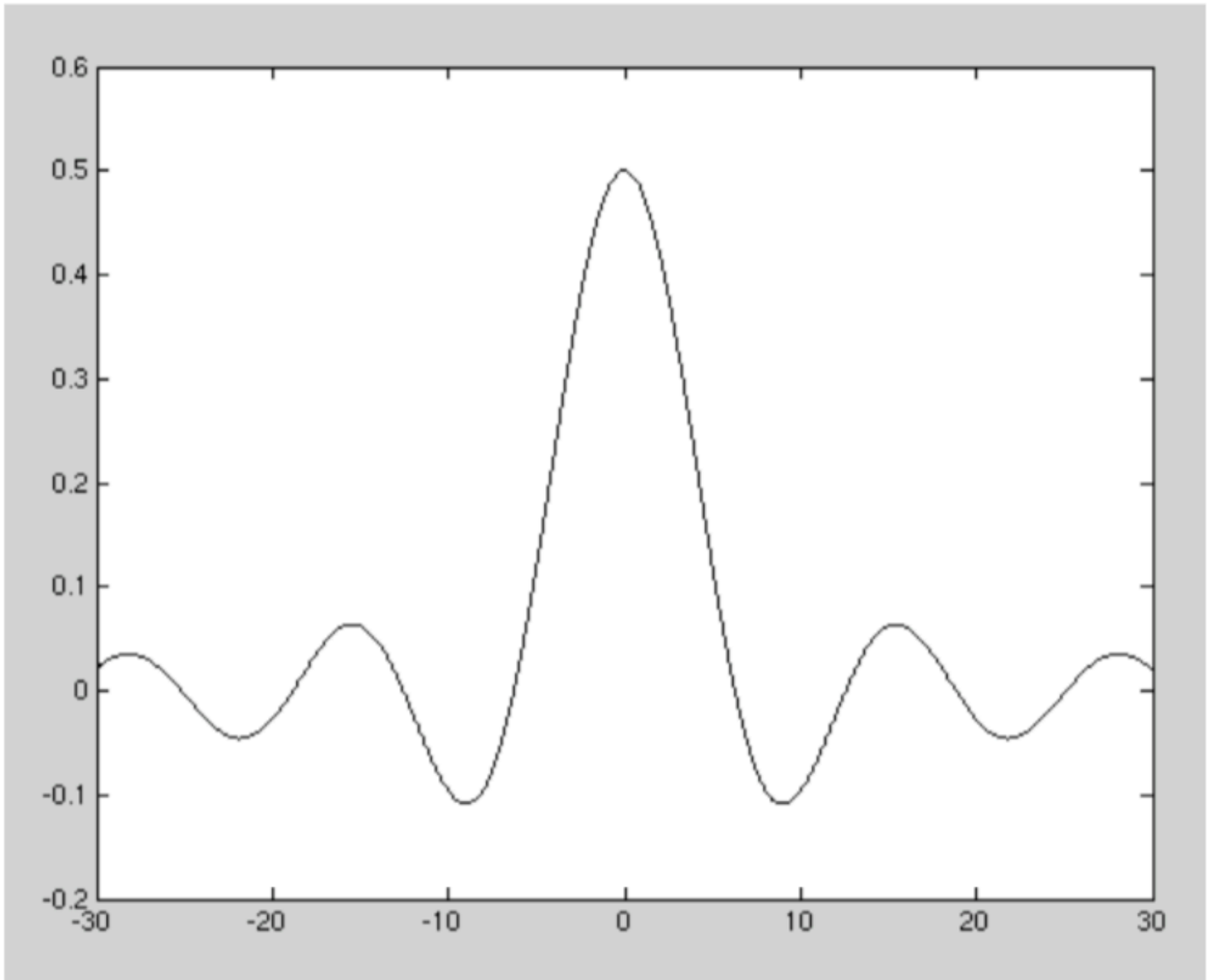
$$f(x) = \begin{cases} 1 & -T/2 < t < T/2 \\ 0 & \text{otherwise} \end{cases}$$

f(t) is absolutely integrable so it has a valid Fourier transform.

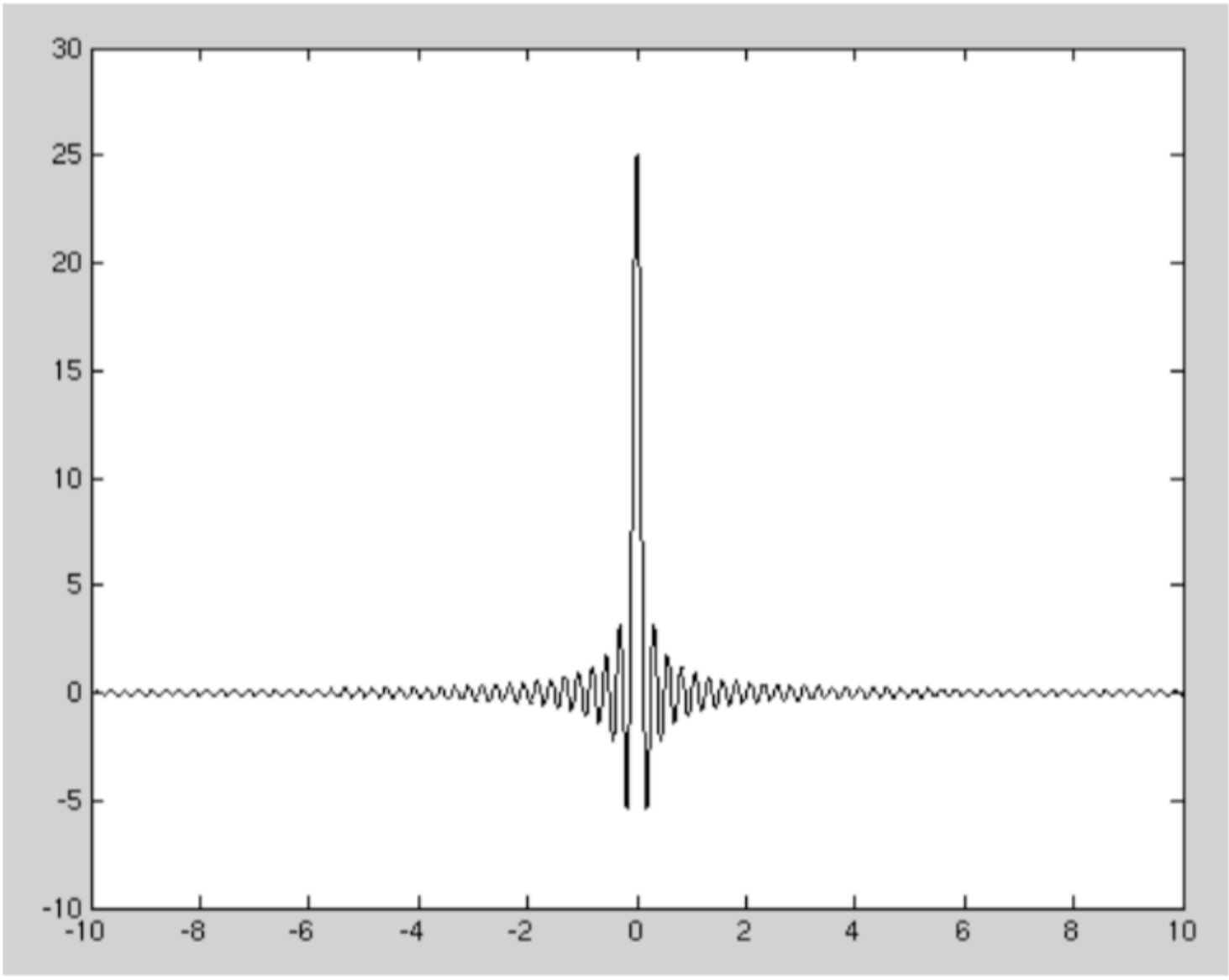
It is given by

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-T/2}^{T/2} dt\, e^{-i\omega t} = \sqrt{\frac{2}{\pi}} \left[\frac{\sin \frac{\omega T}{2}}{\omega} \right]$$

which looks like (for $T = 1$)



In the limit $T \to \infty$ we have ($T = 50$, in fact, here)



We get a sharp spike, but the area remains constant.

This implies that as $T \to \infty$

$$F(\omega) \to \delta(\omega)$$

Formally, we have

$$F(\omega) = \lim_{T\to\infty} \frac{1}{\sqrt{2\pi}} \int_{-T/2}^{T/2} dt\, e^{-i\omega t} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dt\, e^{-i\omega t} = \sqrt{2\pi}\delta(\omega)$$

(2) **Transform of a Delta-Function** - Consider the function

$$f(t) = \delta t$$

The transform is

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dt\, e^{-i\omega t}\delta(t) = \frac{1}{\sqrt{2\pi}}$$

The inverse transform is

$$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} dt\, e^{-i\omega t}\frac{1}{\sqrt{2\pi}} = \delta(t)$$

Now

$$F\left(\frac{df}{dt}\right) = i\omega F(f) = i\omega F(\omega)$$

Therefore for the square pulse we have

$$\frac{df}{dt} = \delta(t + T/2) - \delta(t - T/2)$$

$$\mathcal{I}\left(\frac{df}{dt}\right) = \mathcal{I}(\delta(t + T/2) - \delta(t - T/2))$$

But

$$F(f(t - t_0)) = e^{-i\omega t_0}F(f(t))$$

Thus,

$$F\left(\frac{df}{dt}\right) = (e^{-i\omega(-T/2)} - e^{-i\omega(T/2)} F(\delta(t)) = i\sqrt{\frac{2}{\pi}} \sin\frac{\omega T}{2} = i\omega F(\omega)$$

$$\rightarrow F(\omega) = sqrt\frac{2}{\pi}\left[\frac{\sin\frac{\omega T}{2}}{\omega}\right] \quad \text{for the square pulse (as before)}$$

Remember this only makes sense inside an integral.

(3) **Transform of a Gaussian** - Consider the function

$$f(t) = \frac{\alpha}{\sqrt{\pi}} e^{-\alpha^2 t^2} = \quad \text{normalized Gaussian pulse}$$

We choose $\alpha = 1$.

The peak is at α/π .

The 1/2 maximum points are separated by $\Delta t = 1/\alpha$.

The area under the curve is $= 1$.

The Fourier transform is

$$F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dt \, \frac{\alpha}{\sqrt{\pi}} e^{-\alpha^2 t^2} e^{-i\omega t} = \frac{\alpha}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dt \, e^{(\alpha^2 t^2 + i\omega t)}$$

We complete the square to evaluate the integral. We have

$$\alpha^2 t^2 + i\omega t = \alpha^2 t^2 + i\omega t + \gamma - \gamma = (\alpha t + \beta)^2 - \gamma$$

$$\rightarrow 2\alpha\beta = i\omega t \rightarrow \beta = \frac{i\omega}{2\alpha}$$

$$\rightarrow \gamma = \beta^2 = -\frac{\omega^2}{4\alpha^2}$$

We thus have

$$F(\omega) = \frac{\alpha}{\sqrt{2\pi}} e^{-\frac{\omega^2}{4\alpha^2}} \int_{-\infty}^{\infty} dt\, e^{-\left(\alpha t + \frac{i\omega}{2\alpha}\right)^2}$$

Let

$$x = \alpha t + \frac{i\omega}{2\alpha} \rightarrow dx = \alpha dt$$

then

$$F(\omega) = \frac{1}{\pi\sqrt{2}} e^{-\frac{\omega^2}{4\alpha^2}} \int_{-\infty}^{\infty} dx\, e^{-x^2} = \frac{1}{\sqrt{2\pi}} e^{-\frac{\omega^2}{4\alpha^2}}$$

which is a different Gaussian.