1 Introduction

Quantum algorithms are often efficient at solving problems with periodic functions. Two examples that we have seen so far are Simon's alorithm and order finding. As it turns out, these are related a more general problem in group theory known as the hidden subgroup problem.

In these notes, we will discuss Fourier transforms over other abelian groups, which is a crucial ingredient for many quantum algorithms. We will then define the hidden subgroup problem and outline an efficient quantum algorithm for solving it. Lastly, we will discuss classical error correction codes and their quantum counterparts.

2 Fourier transforms over finite abelian groups

Suppose we have finite Abelian group G. The Fourier transform maps G to a dual group, denoted $\hat{G}$. It is defined as:

$$|g\rangle \mapsto \frac{1}{\sqrt{|G|}} \sum_{k \in \hat{G}} \chi_k(g) |k\rangle$$

where $|G|$ is the order of G and $\chi_k(g)$ are given by $\chi_k(1) = \omega^k$ and $\chi_k(g) = \omega^{kg}$.

For example, consider $G = Z_N$, the group of integers under addition modulo N. For this group, $\chi_k(j)$ is defined as ω^{jk}. Thus, the Fourier transform is given by the familiar matrix F, with $F_{jk} = \frac{1}{\sqrt{N}} \omega^{jk}$.

Let us now consider the action of the Fourier transform on a subgroup $H \subseteq G$. Given that $\chi_k(h) = 1$ for all $h \subseteq H$, the Fourier transform would take an even superposition of elements in H and produce an even superposition of elements in a subgroup $H_\perp \subseteq \hat{G}$:

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |h\rangle \overset{FT_G}{\mapsto} \frac{1}{\sqrt{|H||G|}} \sum_{h \in H, k \in \hat{G}} \chi_k(h) |k\rangle \tag{1}$$

$$= \sqrt{\frac{|H|}{|G|}} \sum_{k \in H_\perp} |k\rangle \tag{2}$$

The amplitude of each element $k \in H^\perp$ is $\sqrt{\frac{|H|}{|G|}}$. But since $|H^\perp| = \frac{|G|}{|H|}$, the sum of squares of these amplitudes is 1. We therefore conclude that the amplitudes of elements not in $H^\perp$ is 0.

What if instead we take the Fourier transform over a coset of H? It turns out that this also forms an even superposition over $H_\perp$ but with a complex phase factor in front of each element:

$$\frac{1}{\sqrt{|H|}} \sum_{h \in H} |hg\rangle \overset{FT_G}{\mapsto} \sqrt{\frac{|H|}{|G|}} \sum_{k \in H^\perp} \chi_g(k) |k\rangle$$

This follows from the convolution-multiplication property of Fourier transforms. An equal superposition on the coset Hg can be obtained by convolving the equal superposition over the subgroup H with a delta function at g. So after a Fourier transform, we get the pointwise multiplication of the two Fourier transforms: namely, an equal superposition over $H^\perp$, and χ_g.

Since the phase $\chi_g(k)$ has no effect on the probability of measuring $\left|k\right\rangle$, Fourier sampling on an equal superposition on a coset of H will yield a uniformly random element $k \in H^\perp$. This is a fundamental primitive in the quantum algorithm for the hidden subgroup problem.

3 The hidden subgroup problem

The hidden subgroup problem is defined as follows: We are given a finite Abelian group G and a function $f : G \to S$. The function is constrained such that it has constant value for all elements in a subgroup $H \subseteq G$ and a different value for all cosets of H. Our challenge is to find H.

Here we outline the quantum algorithm to solve the hidden subgroup problem:

Step I Start with two quantum registers, each large enough to store an element of the group G. Initialize both registers to $\left|0\right\rangle$. Then create superposition over G in the first register by computing a Fourier transform.

$$\left|0\right\rangle\left|0\right\rangle \overset{FT_G}{\mapsto} \frac{1}{\sqrt{|G|}} \sum_{g\in G} \left|g\right\rangle\left|0\right\rangle$$

Step II Apply f to the first register, storing the results in the second register.

$$\overset{f}{\mapsto} \frac{1}{\sqrt{|G|}} \sum_{g\in G} \left|g\right\rangle\left|f(g)\right\rangle$$

Step III Measure the second register. This will collapse the first register to an even superposition over a coset of H for some $g \subseteq G$:

$$\overset{\text{measure}}{\mapsto} \frac{1}{\sqrt{|H|}} \sum_{h\in H} \left|hg\right\rangle$$

Step IV Apply another Fourier transform. As shown above, this will create a superposition over $H_\perp$.

$$\overset{FT_G}{\mapsto} \sqrt{\frac{|H|}{|G|}} \sum_{k\in H^\perp} \chi_g(k)\left|k\right\rangle$$

Step IV Measure the first register. Since, $\chi_g(k)$ is a root of 1, this will return a random element of $H^\perp$. Repeating this process over and over, we will get a number of such random constraints on H, which can then be solved to obtain H.

Let's now reconsider Simon's algorithm under this general framework. In his problem, $G = Z_2^N$, a string of N bits under bitwise addition. The function satisfies $f(x) = f(y)$ if and only if $y = x \oplus s$. Our goal is to find s, or equivalently the subgroup $H = \{0, s\}$.

Simon's algorithm follows the general steps outlined above, where the Fourier transform for $G = Z_2^N$ is defined as the Hadamard transform. In the end, by measuring an element k in $H_\perp$, we determine an orthogonality condition on s, i.e. $s \cdot k = 0$. We repeat this many times until we accumulate enough information to determine s.

4 Fourier transform on $\mathbf{Z}_N$

Let f be a complex-valued function on $\mathbf{Z}_N$. Then its Fourier transform is

$$\hat{f}(t) = \frac{1}{\sqrt{N}} \sum_{x \in \mathbf{Z}_N} f(x) w^{xt}$$

where $w = \exp(2\pi i/N)$. Let B_1 be the standard basis for $\mathscr{C}^{Z_N}$ consisting of vectors $f_i(j) = \delta_{i,j}$. In the standard basis the matrix for the Fourier transform is

$$FT_N = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & w^3 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & w^6 & \cdots & w^{2N-2} \\ 1 & w^3 & w^6 & w^9 & \cdots & w^{3N-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{N-1} & w^{2N-2} & w^{3N-3} & \cdots & w^{(N-1)(N-1)} \end{pmatrix}$$

where i, j'th entry of FT_N is w^{ij}.

Classical fast Fourier transform

Straightforward multiplication of the vector f by FT_N would take $\Omega(N^2)$ steps because multiplication of f by each row requires N multiplications. However, there is an algorithm known as fast Fourier transform (FFT) that performs Fourier transform in $O(N \log N)$ operations.

In our presentation of FFT we shall restrict ourselves to the case $N = 2^n$. Let B_2 be a basis for $\mathscr{C}^{\mathbf{Z}_N}$ consisting of vectors

$$f_i(j) = \begin{cases} \delta_{2i,j}, & i \in \{0, 1, \ldots, N/2-1\}, \\ \delta_{2i-N+1,j}, & i \in \{N/2, N/2+1, \ldots, N-1\}, \end{cases}$$

i.e., the vectors of the standard basis sorted by the least-significant bit. Then as a map from B_2 to B_1 the Fourier transform has the matrix representation

$$\begin{array}{c} \text{bit \#} \\ j \\ j+N/2 \end{array} \begin{array}{c} \begin{array}{cc} 2k & 2k+1 \end{array} \\ \left(\begin{array}{c|c} w^{2jk} & w^{2jk} w^j \\ \hline w^{2jk} & w^{2jk} w^j \end{array}\right) \end{array} = \begin{pmatrix} FT_{N/2} & w^j FT_{N/2} \\ FT_{N/2} & -w^j FT_{N/2} \end{pmatrix}.$$

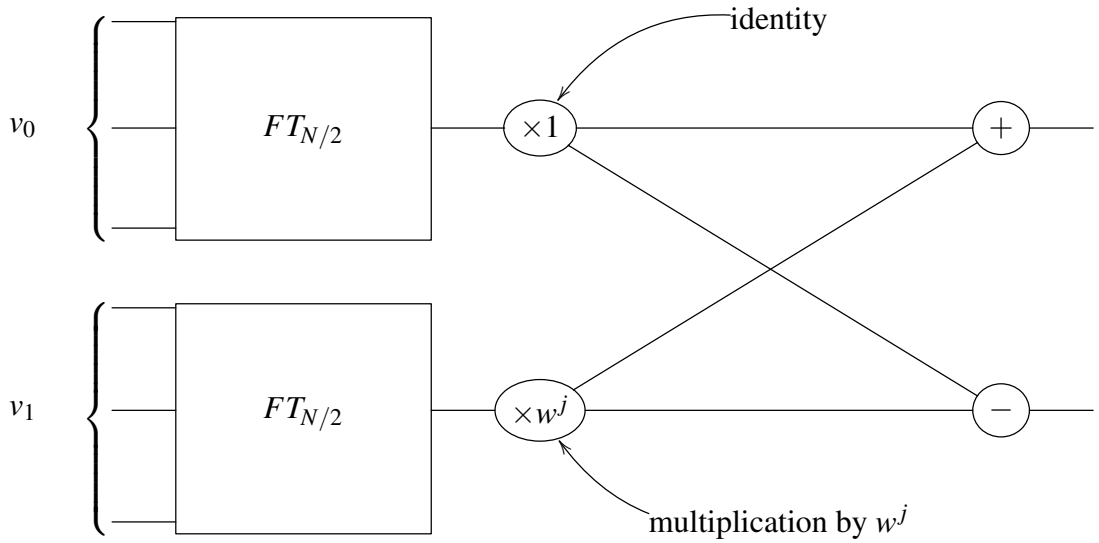

Figure 1: A circuit for classical fast Fourier transform

Hence,

$$\left(\begin{array}{c|c} w^{2jk} & w^{2jk}w^j \\ \hline w^{2jk} & w^{2jk}w^j \end{array}\right)\left(\frac{v_0}{v_1}\right) = \begin{pmatrix} FT_{N/2}v_0 + w^j FT_{N/2}v_1 \\ FT_{N/2}v_0 - w^j FT_{N/2}v_1 \end{pmatrix}.$$

This representation gives a recursive algorithm for computing the Fourier transform in time $T(N) = 2T(N/2) + O(N) = O(N \log N)$. As a circuit the algorithm can be implemented as

Quantum Fourier transform

Let $N = 2^n$. Suppose a quantum state α on n qubits is given as $\sum_{j=0}^{N-1} \alpha_j |j\rangle$. Let the Fourier transform of ϕ be $FT_N|\phi\rangle = \sum_{j=0}^{N-1} \beta_j |j\rangle$ where

$$FT_N \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{pmatrix} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{N-1} \end{pmatrix}.$$

The map $FT_N = |\alpha\rangle \mapsto |\beta\rangle$ is unitary (see the proof below), and is called the quantum Fourier transform (QFT). A natural question arises whether it can be efficiently implemented quantumly. The answer is that it can be implemented by circuit of size $O(\log^2 N)$. However, this does not constitute an exponential speed-up over the classical algorithm because the result of quantum Fourier transform is a superposition of states which can be observed, and any measurement can extract at most $n = \log N$ bits of information.

A quantum circuit for quantum Fourier transform is where R_K is the controlled phase shift by angle

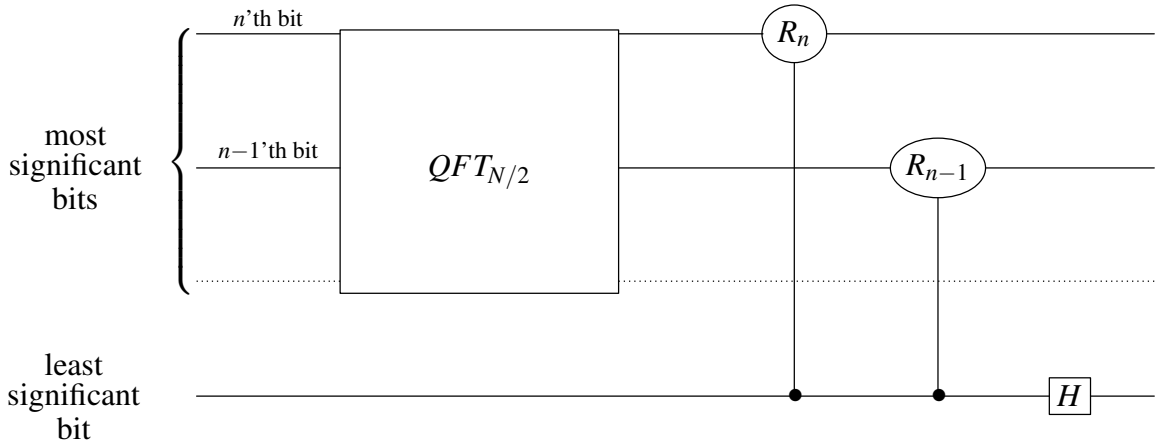

Figure 2: Circuit for quantum Fourier transform

$2\pi/2^K$ whose matrix is

$$R_K = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi/2^K} \end{pmatrix}.$$

In the circuity above the quantum Fourier transform on $n-1$ bits corresponds to two Fourier transforms on $n-1$ bits in the figure **??**. The controlled phase shifts correspond to multiplications by w^j in classical circuit. Finally, the Hadamard gate at the very end corresponds to the summation.

Properties of Fourier transform

- FT_N is unitary. Proof: the inner product of the i'th and j'th column of FT_N where $i \neq j$ is

$$\frac{1}{N}\sum_{k\in\mathbf{Z}_N} w^{ik}\overline{w^{jk}} = \frac{1}{N}\sum_{k\in\mathbf{Z}_N} w^{ik-jk} = \frac{1}{N}\sum_{k\in\mathbf{Z}_N} (w^{i-j})^k = \frac{1}{N}\frac{w^{N(i-j)}-1}{w^{i-j}-1} = \frac{1}{N}\frac{1-1}{w^{i-j}-1}$$

  which is zero because $w^{i-j} \neq 1$ due to $i \neq j$. The norm of i'th column is

$$\sqrt{\frac{1}{N}\sum_{k\in\mathbf{Z}_N} w^{ik}\overline{w^{ik}}} = \sqrt{\frac{1}{N}\sum_{k\in\mathbf{Z}_N} 1} = 1.$$

- FT_N^{-1} is FT_N with w replaced by w^{-1}. Proof: since FT is unitary we have $F_N^{-1} = FT_N^*$. Since FT_N is symmetric and $\bar{w} = w^{-1}$, the result follows.

- Fourier transform sends translation into phase rotation, and vice versa. More precisely, if we let the translation be $T_l\colon \big|x\big\rangle \mapsto \big|x+l \pmod N\big\rangle$ and rotation by $P_k\colon \big|x\big\rangle \mapsto w^{kx}\big|x\big\rangle$, then $FT_N P_l P_k = P_l T_{-k} FT_N$. Proof: by linearity it suffices to prove this for a vector of the form $\big|x\big\rangle$. We have

$$FT_N T_l P_k\big|x\big\rangle = FT_N w^{kx}\big|x+l \pmod N\big\rangle = \frac{1}{\sqrt{N}} w^{kx}\sum_{y\in\mathbf{Z}_N} w^{y(x+l)}\big|y\big\rangle$$

and by making the substitution $y = y' - k$

$$= \frac{1}{\sqrt{N}} w^{y'x} \sum_{y' \in \mathbf{Z}_N} w^{(y'-k)l} \left| y' - k \right\rangle = \frac{1}{\sqrt{N}} P_l T_{-k} \sum_{y' \in \mathbf{Z}_N} w^{xy} \left| y' \right\rangle$$
$$= P_l T_{-k} F T_N \left| x \right\rangle .$$

Corollary: FT_N followed by Fourier sampling is equivalent to $T_l FT_N$ followed by Fourier sampling.

- Suppose $r \mid N$. Let $\left|\phi\right\rangle = \frac{1}{\sqrt{N/r}} \sum_{j=0}^{N/r-1} \left| jr \right\rangle$. Then $FT_N \left|\phi\right\rangle = \frac{1}{\sqrt{r}} \sum_{i=0}^{r-1} \left| i\frac{N}{r} \right\rangle$. Proof: the amplitude of $\left| i\frac{N}{r} \right\rangle$ is

$$\frac{1}{\sqrt{N}} \frac{1}{\sqrt{N/r}} \sum_{j=0}^{N/r-1} w^{(jr)(iN/r)} = \frac{\sqrt{r}}{N} \sum_{j=0}^{N/r-1} 1 = \frac{1}{\sqrt{r}}$$

Since FT_N is unitary, the norm of $FT_N \left|\phi\right\rangle$ has to be equal to the norm of $\left|\phi\right\rangle$ which is 1. However the orthogonal projection of $FT_N \left|\phi\right\rangle$ on the space spanned by vectors of the form $\left| i\frac{N}{r} \right\rangle$ has norm 1. Therefore $FT_N \left|\phi\right\rangle$ lies in that space.

If we apply the corollary above to $\left|\phi\right\rangle$ we conclude that the result of Fourier sampling of $T_l \left|\phi\right\rangle = \frac{\sqrt{r}}{\sqrt{N}} \sum_{j=0}^{N/r-1} \left| jr + l \right\rangle$ is a random multiples of N/r.

5 Classical error correction

Classical error correction codes (ECCs) are outlined here to serve as a comparison and inspiration for the quantum ECCs.

5.1 Polynomial codes

A classical approach to represent a code (e.g. list of numbers) in an error-correctable fashion is to extropolate the code into a polynomial and send additional numbers that lie on the polynomial. As a result, if some of the numbers are altered, the receiver may still be able to determine the correct polynomial and therefore extract the original message.

Consider, for example, a two-integer message $a_1 a_2$ (modulo p). We can use these numbers to define two points with unit separation, e.g. $(x1, y1) = (1, a_1)$ and $(x2, y2) = (2, a_2)$. We can then form a line through the two points and determine other values of $(x_i, y_i) = (i, b_i)$ that lie on the line. We secure the message by sending the original two values plus several extrapolated values, i.e. $a_1 a_2 b_1 b_2 ... b_n$. The receiver will then form a line through the most points and determine the values at $x = 1$ and $x = 2$.

How many errors does this procedure protect against? Suppose the message contained $2k$ extra numbers, so $2k + 2$ total. In the worst case, all the errors would form another line and this line would also intersect with one of the correct points. If there were k such errors, then the receiver would see two lines, the correct one with $k + 2$ points and the incorrect one with $k + 1$ points.

He or she can still resolve which is the correct, but for any additional errors he or she would not. Therefore, we conclude that $2k$ extra points would hold up against up to k errors.

In general, for a message with $d+1$ numbers, we could repeat this procedure with a d-degree polynomial. Following the same logic as before, to protect against k errors, the sender would have to send an additional $2k$ numbers, for a total of $2k+d+1$. Importantly, there exists an efficient algorithm for the receiver to deduce the polynomial that passes through the most points.

5.2 Classical linear error correcting code over $\mathbb{F}_2$

We consider an error correcting code (ECC) $C \subseteq \mathbb{F}_2^n$ with $|C| = 2^k$. Such a code has two important matrices, the $k \times n$ generator matrix G and the $(n-k) \times n$ parity matrix P.

5.2.1 The generator matrix

The rows of the generator matrix G span the codewords of C so that a k-bit message $m \in \mathbb{F}_2^k$ can be encoded to an n-bit codeword $c \in \mathbb{F}_2^n$ via

$$c = mG.$$

In general, such a code is classified as an $[n,k,d]$ code where n and k are as above and d is the minimum Hamming distance between any two codewords in C. We note that since C is a linear code (i.e., $c_1, c_2 \in C \Rightarrow c_1 + c_2 \in C$), d may be equivalently defined as the minimum Hamming weight among non-zero codewords. A general $[n,k,d]$ code allows for the detection of up to $d-1$ bit errors (these errors would necessarily produce an invalid codeword), and allows for the correction of up to $(d-1)/2$ errors (by choosing the codeword of minimum Hamming distance from the received message).

5.2.2 The parity matrix

The parity matrix P has the property that its columns generate $C^\perp$ and thus

$$cP = 0 \iff c \in C.$$

For a general received message of the form $c' \equiv c + e$ where $c \in C$ and $e \in \mathbb{F}_2^n$ represents errors in the received message, the syndrome of c' is defined to be

$$c'P = (c+e)P = eP.$$

For $0 \leq |e| \leq d-1$, the syndrome eP will always be non-zero and hence permit error detection. Moreover, if $|e| \leq (d-1)/2$, the error e can be determined *uniquely* and hence corrected. Finally, we note that there exist classical $[n,k,d]$ ECCs with $n = O(k)$ and $d = \Theta(n)$.

6 Quantum error correction

Throughout the discussion of quantum computation so far, we assumed that the quantum system we were interested in was error-free. In reality, however, this is far from being true. The system

continuously interacts with its environment. This interaction may corrupt the state of the system and the need for some error correcting mechanism becomes inevitable.

We first discuss two kinds of errors the environment may introduce in the system:

- bit flips
- bit measurements

Later we shall introduce a formal model of errors and discuss how to tackle them.

6.1 Bit flips

Suppose for simplicity that our system consists of a single qubit. We start with errors we are familiar with from the classical setting – bit-flips. Such an error converts the original state, say $\alpha|0\rangle + \beta|1\rangle$, into $\alpha|1\rangle + \beta|0\rangle$. We can correct these kind of errors using classical error correcting codes.

For example, we may use a repetition code, i.e., encoding $|0\rangle$ as $|000\rangle$ and $|1\rangle$ as $|111\rangle$. Thus, we encode $\alpha|0\rangle + \beta|1\rangle$ as $\alpha|000\rangle + \beta|111\rangle$. Now suppose the second bit gets flipped. Then the new state becomes $\alpha|010\rangle + \beta|101\rangle$. We assume that when any bit gets flipped, it is flipped in all superpositions. In order to locate the error, we attach an "error locating register" initially set to zero. This register is supposed to store the location of the bit in error. The overall state prior to any computation is thus

$$(\alpha|010\rangle + \beta|101\rangle) \otimes |0\rangle.$$

We then perform the required computations to determine the location of bits in error as we do in the classical setting, resulting in the state

$$\alpha|010\rangle \otimes |2\rangle + \beta|101\rangle \otimes |2\rangle.$$

Notice that the register which stores the error location is in tensor with the data qubits, i.e., the current state is

$$(\alpha|010\rangle + \beta|101\rangle) \otimes |2\rangle.$$

We can now recover the original qubits by flipping the bit at the location given by the error register to get

$$\alpha|000\rangle \otimes |2\rangle + \beta|111\rangle \otimes |2\rangle,$$

which is

$$(\alpha|000\rangle + \beta|111\rangle) \otimes |2\rangle.$$

From this state, the original state $\alpha|0\rangle + \beta|1\rangle$ can be recovered.

This scheme works even when the bit-flip errors occur in superposition. Suppose, for example, the first bit gets flipped with amplitude $1/\sqrt{2}$ and the second bit gets flipped with amplitude $1/\sqrt{2}$. In this case, the state after the error is

$$\frac{\alpha}{\sqrt{2}}|100\rangle + \frac{\beta}{\sqrt{2}}|011\rangle + \frac{\alpha}{\sqrt{2}}|010\rangle + \frac{\beta}{\sqrt{2}}|101\rangle.$$

After adding the error register and locating the bit in error we get

$$\frac{\alpha}{\sqrt{2}}|100\rangle\otimes|1\rangle+\frac{\beta}{\sqrt{2}}|011\rangle\otimes|1\rangle+\frac{\alpha}{\sqrt{2}}|010\rangle\otimes|2\rangle+\frac{\beta}{\sqrt{2}}|101\rangle\otimes|2\rangle$$
$$=\frac{1}{\sqrt{2}}(\alpha|100\rangle+\beta|011\rangle)\otimes|1\rangle+\frac{1}{\sqrt{2}}(\alpha|010\rangle+\beta|101\rangle)\otimes|2\rangle$$

Now we measure the error register so that the system collapses to either $(\alpha|100\rangle+\beta|011\rangle)\otimes|1\rangle$ or $(\alpha|010\rangle+\beta|101\rangle)\otimes|2\rangle$. We then, as before, flip the bit in the location given by the error register to recover the original state.

6.2 Bit measurements

Another kind of error which is unique to the quantum setting is an unwanted measurement by the environment resulting in collapse of the current state. For example, if the current state is $\frac{1}{\sqrt{3}}(|001\rangle+|010\rangle+|101\rangle)$ and the second qubit gets measured, the state collapses to either $\frac{1}{\sqrt{2}}(|001\rangle+|101\rangle)$ or $|010\rangle$. During the measurement, some of the states in superposition get lost. As a consequence of the linearity of quantum mechanics, each basis state being unaware of the others, cannot tell that some of the others have disappeared. Therefore at first look, recovering from such errors seems very hard or even impossible.

One may be tempted to use the repetition code again to tackle these errors. For instance, we may try encoding the state $|\Psi\rangle=\alpha|0\rangle+\beta|1\rangle$ as $|\Psi\rangle\otimes|\Psi\rangle\otimes|\Psi\rangle$. In such an encoding, if one of the three registers gets measured, we may hope to recover $|\Psi\rangle$ from the other two. However, such an encoding needs copying a qubit into three qubits and hence is impossible by No-cloning theorem.

We defer the discussion of how to take care of measurements or any general error.

6.3 Model of quantum errors

Now we introduce the model of general quantum errors as follows. Abstractly, our system is coupled with the environment. An error is modeled as a unitary transformation U that is applied to some qubits in our system and some other in the environment. Finally, some qubits get measured. The resulting state of our system depends on the result of the measurement. This overall transformation is linear. It is enough to consider only bit-flips, phase-flips, and their combinations since the space of linear operators is spanned by the operators representing the different possible bit-flips, phase flips, combinations of bit and phase-flips, and the identity operator. We illustrate the above fact for one bit errors. Suppose that the initial state is $\alpha|0\rangle+\beta|1\rangle$. The space of one bit linear operators is spanned by the following four operators.

$$\text{no errors:} \quad I\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

$$\text{bit-flip:} \quad X\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha\end{pmatrix}$$

$$\text{phase-flip:} \quad Z\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha\\ -\beta\end{pmatrix}$$

$$\text{bit+phase-flip:} \quad XZ\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ -\alpha\end{pmatrix}$$

This suggests that in order to correct general errors it is enough to be able to correct bit-flips and phase-flips. In Section **??**, we saw how to correct bit-flip errors using repetition code. Now we see how to correct phase-flip errors by transforming them to bit-flips.

6.4 Phase flips

Recall that the Hadamard operator

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$$

transforms $|0\rangle$ into $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ into $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. It also transforms X into Z and vice versa, in the sense that $H^\dagger XH = Z$ and $H^\dagger ZH = X$. Thus in the Hadamard basis $\{|+\rangle, |-\rangle\}$, the phase-flip errors get transformed into "bit-flip" errors, i.e., $Z|+\rangle = |-\rangle$ and $Z|-\rangle = |+\rangle$.

With these observations, to correct the phase-flip errors, it is natural to encode $|0\rangle$ as $|+++\rangle$ and $|1\rangle$ as $|---\rangle$. We illustrate how to correct one qubit phase-flip error as follows. We encode the state $\alpha|0\rangle + \beta|1\rangle$ as $\alpha|+++\rangle + \beta|---\rangle$. A phase-flip errors, say in the second qubit, results in the state $\alpha|+-+\rangle + \beta|-+-\rangle$. To correct this, we first transform this into standard basis by applying tensored Hadamard operator $H^{\otimes 3}$:

$$\alpha|+-+\rangle + \beta|-+-\rangle \xmapsto{H^{\otimes 3}} \alpha|010\rangle + \beta|101\rangle.$$

Now using the bit error correction, we restore the state to $\alpha|000\rangle + \beta|111\rangle$. Again applying $H^{\otimes 3}$, we get $\alpha|+++\rangle + \beta|---\rangle$ we desired.