register = [thick, rectangle, draw=black, minimum height=.7cm, text centered
]

1 Fourier Transform over Finite Abelian Groups

Given a finite abelian group G with n elements, we want to study the Fourier transform over it. Usually, we are interested in the following two cases: (1) $G = \mathbf{Z}_n := \{0, 1, \ldots, n-1\}$, the group of integers modulo n under addition; and (2) $G = \mathbf{Z}_2^m$ with $m = \log_2 n$, the group of m-bit strings under bitwise addition modulo two.

To define the Fourier transform, we consider the characters of G. A map $\chi_j \colon G \to \mathscr{C}$ is a character if it is a group homomorphism, i.e. $\chi_j(gg') = \chi_j(g)\chi_j(g')$ for any $g, g' \in G$. There are exactly $|G|$ characters, and they form a group under pointwise multiplication. This group is called the dual group of G, and is denoted as $\hat{G}$. The Fourier transform maps $\left|g\right\rangle$ to $\sum_{j\in\hat{G}} \frac{1}{\sqrt{|G|}}\chi_j(g)\left|j\right\rangle$.

For example, when $G = \mathbf{Z}_n$, and $\chi_j(1) = \omega^j$, then we have $\chi_j(k) = \omega^{jk}$, where $\omega = e^{2\pi i/n}$ is the n-th primitive root of unity. When written as a vector, $\left|\chi_j\right\rangle = (1, \omega^j, \omega^{2j}, \ldots, \omega^{(n-1)j})^\top$, and the matrix of Fourier transform $\mathscr{F}$ is defined as $\mathscr{F}_{ij} = \frac{1}{\sqrt{|G|}}\omega^{ij}$ (where i and j runs from zero to $n-1$).

In general, when $G \cong \mathbf{Z}_{N_1} \times \mathbf{Z}_{N_2} \times \cdots \times \mathbf{Z}_{N_\ell}$, with its elements identified as $(g_1, g_2, \ldots, g_\ell)$ ($g_i \in \mathbf{Z}_{N_i}$), G's characters have the form $\chi_{k_1,k_2,\ldots,k_\ell}(g_1, g_2, \ldots, g_\ell) = \omega_{N_1}^{k_1 g_1}\omega_{N_2}^{k_2 g_2}\cdots\omega_{N_\ell}^{k_\ell g_\ell}$, where $\omega_{N_i} = e^{2\pi i/N_i}$ is the N_i-th primitive root of unity. In this case, the Fourier transform maps $\left|g_1, g_2, \ldots, g_\ell\right\rangle$ to $\frac{1}{\sqrt{|G|}}\sum_{k_1,k_2,\ldots,k_\ell}\chi_{k_1,k_2,\ldots,k_\ell}(g_1, g_2, \ldots, g_\ell)\left|k_1, k_2, \ldots, k_\ell\right\rangle$.

If we identify the n distinguishable states $\left|0\right\rangle, \left|1\right\rangle, \ldots, \left|n-1\right\rangle$ in a quantum system with the group elements, we can express any state $\left|\phi\right\rangle$ as a superposition $\sum_{g\in G}\alpha_g\left|g\right\rangle$.

Fix any $g' \in G$, and consider the action of multiplication by g'. It maps $\sum_{g\in G}\alpha_g\left|g\right\rangle$ to $\sum_{g\in G}\alpha_g\left|g'g\right\rangle$. If we express this linear transformation as a matrix $A_{g'}$, mapping $\left|\phi\right\rangle$ to $A_{g'}\left|\phi\right\rangle$, then $A_{g'}$ is a permutation matrix (a zero-one matrix having exactly one 1 on each row and each column) in the basis of G. However, if we express the multiplication in the Fourier basis, we should get a diagonal matrix.

Claim For any $g' \in G$, the multiplication by g' is diagonal in the Fourier basis.
Proof Let $A_{g'}$ be the multiplication by g', and $\left|\chi_j\right\rangle = (1, \omega^j, \omega^{2j}, \ldots, \omega^{(n-1)j})^\top = \sum_{g\in G}\chi_j(g)\left|g\right\rangle$ be a vector representing a character (up to a normalizing factor of $\frac{1}{\sqrt{|G|}}$).

$$\mathbf{A}_{g'}\left|\chi_j\right\rangle = \mathbf{A}_{g'}\sum_{g\in G}\chi_j(g)\left|g\right\rangle = \sum_{g\in G}\chi_j(g)\left|g'g\right\rangle = \sum_{g\in G}\chi_j(g'^{-1}g)\left|g\right\rangle = \chi_j(g'^{-1})\sum_{g\in G}\chi_j(g)\left|g\right\rangle = \chi_j(g'^{-1})\left|\chi_j\right\rangle,$$

so each $\left|\chi_j\right\rangle$ in $\hat{G}$ is an eigenvector of $A_{g'}$ with an eigenvalue of $\chi_j(g'^{-1})$, i.e. $A_{g'}$ diagonalizes in the basis of $\hat{G}$.

2 Subgroups and Cosets

Every subgroup H of G corresponds to a subgroup $H^\perp$ of $\hat{G}$, given by $H^\perp = \{k \in \hat{G} \mid \chi_k(h) = 1 \quad \forall h \in H\}$. If we define $\left|H\right\rangle := \frac{1}{\sqrt{|H|}}\sum_{h\in H}\left|h\right\rangle$ to be the uniform superposition of states in H, its Fourier transform is $\left|H^\perp\right\rangle = \sqrt{\frac{|H|}{|G|}}\sum_{k\in H^\perp}\left|k\right\rangle$, which is the uniform superposition of states in $H^\perp$ (note that $|H^\perp| = \frac{|G|}{|H|}$).

Moreover, if Hg is a coset of H, the uniform superposition of its states $\left|Hg\right\rangle := \frac{1}{\sqrt{|H|}}\sum_{h\in Hg}\left|h\right\rangle$ is mapped to $\sqrt{\frac{|H|}{|G|}}\sum_{k\in H^\perp}\chi_k(g)\left|k\right\rangle$ under Fourier transform.

Claim Fourier transform takes $\left|H\right\rangle$ to $\left|H^\perp\right\rangle$ and $\left|Hg\right\rangle$ to $\sqrt{\frac{|H|}{|G|}}\sum_{k\in H^\perp}\chi_k(g)\left|k\right\rangle$.

Proof For the first case, consider $k \in H^\perp$, its amplitude $= \sum_{h\in H}\frac{1}{\sqrt{|H|}}\frac{\chi_k(h)}{\sqrt{|G|}} = \sqrt{\frac{|H|}{|G|}}$ (because $\chi_k(h) = 1$ when $k \in H^\perp$ and $h \in H$).

For $k \notin H^\perp$, there is a $h' \in H$ such that $\chi_k(h') \neq 1$. Let $\beta_k :=$ the amplitude of $k = \sum_{h\in H}\frac{1}{\sqrt{|H|}}\frac{\chi_k(h)}{\sqrt{|G|}} = \sum_{h\in H}\frac{1}{\sqrt{|H|}}\frac{\chi_k(h'h)}{\sqrt{|G|}} = \chi_k(h')\sum_{h\in H}\frac{1}{\sqrt{|H|}}\frac{\chi_k(h)}{\sqrt{|G|}} = \chi_k(h')\beta_k$, so we get $(1-\chi_k(h'))\beta_k = 0$ and deduce that $\beta_k = 0$. This completes the first case.

For the second case, we observe that the amplitude of k in $\left|Hg\right\rangle$ is $\chi_k(g)$ multiplied by its amplitude in $\left|H\right\rangle$. Indeed, its amplitude in $\left|Hg\right\rangle = \sum_{h\in Hg}\frac{1}{\sqrt{|H|}}\frac{\chi_k(h)}{\sqrt{|G|}} = \sum_{h\in H}\frac{1}{\sqrt{|H|}}\frac{\chi_k(hg)}{\sqrt{|G|}} = \chi_k(g)\sum_{h\in H}\frac{1}{\sqrt{|H|}}\frac{\chi_k(h)}{\sqrt{|G|}} = \chi_k(g)$ multiplied by its amplitude in $\left|H\right\rangle$. From this, we deduce that, for $k \in H^\perp$, its amplitude is $\chi_k(g)\sqrt{\frac{|H|}{|G|}}$; and for $k \notin H^\perp$, its amplitude is for some random coset Hg', after ignoring (or, equivalently by the safe storage principle, measuring) the second component.

Step 2 **Fourier sampling for a random element in $H^\perp$:**
By using the fact that all cosets of H map to the same vector $H^\perp$ (up to varying phases but same amplitude),

$$\left|H\right\rangle \xrightarrow{\text{Fourier transform}} \left|H^\perp\right\rangle \qquad \left|Hg\right\rangle \xrightarrow{\text{Fourier transform}} \chi(g)\cdot\left|H^\perp\right\rangle^{1},$$

a measurement on the Fourier transform of $\left|Hg\right\rangle$ gives a random element in $H^\perp$. Each element in $H^\perp$ enforces a constraint that (elements of) H has to satisfy, and we can then recover H with high probability given enough random samples of $H^\perp$.

4 Factoring

To factorize an integer N, we find the order of a random element by solving a hidden subgroup problem.

First, we need some definitions. Let $\mathbf{Z}_N^* := \{0 < a < N \mid \gcd(a,N) = 1\}$ denotes the set of integers co-prime to N, which forms a group under multiplication modulo N. Let $\phi(N) := |\mathbf{Z}_N^*|$ be the Euler ϕ function. Recall that if $N = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, then we know $\phi(N) = (p_1-1)p_1^{e_1-1}(p_2-1)p_2^{e_2-1}\cdots(p_k-1)p_k^{e_k-1}$.

We want to pick a random element x mod N and compute its order given by $\mathrm{ord}_N(x) := \min_{r>0} x^r \equiv 1 \bmod N$. In fact, the order of an element x is defined (or finite) iff $\gcd(x,N) = 1$. However, if $\gcd(x,N) \neq 1$, we already get a non-trivial factor of N. Let's assume that $\gcd(x,N) = 1$, and let $r := \mathrm{ord}_N(x)$. Clearly r divides $\phi(N)$ (by the Lagrange theorem).

Let $M = \phi(N)$ (or a multiple of $\phi(N)$). We consider the function $f\colon \mathbf{Z}_M \to \mathbf{Z}_N$ given by $f(a) = x^a \bmod N$. If we denote $\langle r\rangle :=$ the subgroup generated by r in Z_M (using the fact that r divides $\phi(N)$), then f is constant on cosets and is distinct on different cosets of $\langle r\rangle$. Indeed, $f(a) = f(a')$ iff $x^a = x^{a'} \bmod N$ iff $x^{a-a'} = 1 \bmod N$ iff $a = a' \bmod r$.

The quantum algorithm gives us the value of r as a non-trivial factor of $\phi(N)$ (with high probability), which can be used to factorize N. More precisely, the Fourier sampling step gives us a random element of $\langle r\rangle^\perp \cong \langle \frac{M}{r}\rangle$, i.e. we get an element $\frac{sM}{r}$ for a random $s \in \{0,1,\ldots,r-1\}$. With enough samples of the form $\frac{sM}{r}$ (for different s), we know their $\gcd = \frac{M}{r}$.

The above works if M is a multiple of $\phi(N)$. But the value of $\phi(N)$ is not known (or we could have factorized N directly with it). Therefore, we run the algorithm with a large enough $M \gg N$ instead, which works because the error between the Fourier transform over $\mathbf{Z}_M$ and that over $\mathbf{Z}_{\text{a large multiple of } \phi(N)}$ is small.

5 Discrete Logarithm

In this section, we consider another hardness assumption used in cryptography (e.g. Diffie-Helman cryptosystem), namely the problem of discrete logarithm.

Let p be a prime, then $\mathbf{Z}_p$ forms a group under addition modulo p. Its non-zero elements $\mathbf{Z}_p^*$ forms a group under multiplication modulo p, which in fact is cyclic, i.e. there is an element g that generates $\mathbf{Z}_p^* = \{g^0, g^1, \ldots, g^{p-2}\}$. The question of discrete logarithm is, given x, p and g, to compute an exponent r such that $g^r \bmod p \equiv x$.

Let $G := \mathbf{Z}_{p-1} \times \mathbf{Z}_{p-1}$, and we define $f(a,b) := g^a x^{-b} \bmod p$. Note that $f(a,b) = 1$ iff $a - br = 0 \bmod p-1$. Hence, if we let $H :=$ the subgroup generated by $(r,1)$, then f is constant on cosets and distinct on different cosets of H. By the quantum algorithm, we get a random sample $(c,d) \in H^\perp$, satisfying $cr + d = 0 \bmod p-1$. With non-negligible probability, such a random sample satisfies an additional relation that $\gcd(c, p-1) = 1$, which allows us to compute $r = dc^{-1} \bmod p-1$. We can amplify the success probability by repeating the whole algorithm, and checking for those r satisfying $g^r = x \bmod p$.

[1] Here, we use the notation $\chi(g)\cdot\left|H^\perp\right\rangle$ to mean the pointwise multiplication of the two vectors. That is, for $k \in \hat{G}$, its amplitude in $\chi(g)\cdot\left|H^\perp\right\rangle$ is $\chi_k(g)\sqrt{\frac{|H|}{|G|}}$ if $k \in H^\perp$, and is 0 otherwise.

6 Quantum Error Correction

From time to time, the qubits in quantum registers get corrupted with certain probability. Therefore, we need a method to correct errors in quantum registers for fault tolerant quantum computation. In order to study quantum error correction, we first need to understand what quantum errors are possible.

The first kind of errors are bit flips, analogous to the classical bit flips. For example, a state $\alpha|0000\rangle + \beta|1111\rangle$ may be corrupted to $\alpha|0010\rangle + \beta|1101\rangle$ if the third bit is flipped. We model a bit flip as an application of a NOT gate to a certain qubit, represented by the matrix $X := \begin{bmatrix}01\\10\end{bmatrix}$.

[scale=1, auto] (data) [register, text width = 3cm] at (0, 0) Original Data; (codeword) [register, text width = 9cm] at (9, 0) Codeword; (corrupted) [register, text width = 9cm] at (9, -1.4) Corrupted Codeword; (recovered) [register, text width = 3cm] at (0, -1.4) Recovered Data; [-¿, thick] (data.east) to node Encode (codeword.west); [-¿, thick] (codeword.south) to node Bit Flips (corrupted.north); [-¿, thick] (corrupted.west) to node [swap] Reconstruct (recovered.east);
`data.northwest)` +0,.2) coordinate (data north west); `data.northeast)` +0,.2) coordinate (data north east); `codeword.northwest)` +0,.2) coordinate (codeword north west); `codeword.northeast)` +0,.2) coordinate (codeword north east); [¡-¿] (data north west) to node n (data north east); [¡-¿] (codeword north west) to node $3n$ (codeword north east);

Figure 1: Flow Diagram of the Imagined Error Correction for Bit Flips

If we take the quantum analogue of classical error correction, we may want to introduce redundancy into the data by repeating qubits, so that when a small fraction of qubits in the codeword is corrupted, hopefully we can reconstruct the original data. For example, we may naïvely want to stretch a data of n qubits into a codeword of $3n$ qubits by repeating each qubit three times. However, there is no transformation for turning $|\phi\rangle$ into $|\phi\rangle|\phi\rangle|\phi\rangle$ for a general state ϕ in superposition, by the no-cloning theorem. So we cannot directly translate the classical error correction techniques for quantum error correction, even for the analogous error of bit flips.

The second kind of errors are unwanted interaction with the environment, which is modeled as bit measurements by the environment on certain qubits. For example, a state $\frac{1}{\sqrt{2}}|0000\rangle + \frac{1}{\sqrt{2}}|1111\rangle$ may turn into $|0000\rangle$ or $|1111\rangle$ with equal probability, if the environment measures it on any single qubit.

7 Noise Model

Let us be more specific about what quantum error correction means.

Assume that we have some qubits $|\phi\rangle$ in a quantum register. To guard against quantum errors, we append a number of qubits initialized to the all-zero state $|0\rangle$, and transform (encode) the resultant qubits with a unitary operator $\mathscr{U}$ into a state suitable for correction. After the codeword is corrupted, we append a number of fresh all-zero qubits, and apply an error correction circuit to extract and throw away the error, recovering the encoded codeword $\mathscr{U}(|\phi\rangle|0\rangle)$. The original qubits $|\phi\rangle$ can be reconstructed with an application of $\mathscr{U}^{-1}$.

We are going to model a general quantum error (e.g. bit flips and bit measurements) as a unitary operation $\mathscr{U}'$ performed by the environment, with part of the output thrown away.

Under this model, the corrupted state $|\text{Corrupted}\rangle$ is a linear (and in general non-unitary) transformation of the original state $|\text{Original}\rangle$. For a system with a single qubit (modeled by the Hilbert space $\mathscr{C}^2$), all linear transformations are spanned by the four matrices: identity $\mathbf{I}$, bit flip X, phase flip Z, and bit-and-phase flip $Y = XZ$.

$$\text{Identity } \mathbf{I} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \qquad \text{Bit Flip } X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

$$\text{Phase Flip } Z = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix} \qquad \text{Bit-and-Phase Flip } Y = \begin{bmatrix}0 & 1\\-1 & 0\end{bmatrix}$$

[scale=1, auto] (data) [register, text width = 2cm] at (0, 0) $\left|\phi\right\rangle$; (data zero) [register, text width = 2cm] at (2.25, 0) $\left|0\right\rangle$; (codeword) [register, text width = 4cm] at (8, 0) $\mathscr{U}(\left|\phi\right\rangle\left|0\right\rangle)$; (corrupted) [register, text width = 4cm] at (8, -1.5) $\left|\text{Corrutped}\right\rangle$; (corrupted zero) [register, text width = 3cm] at (11.75, -1.5) $\left|0\right\rangle$; (correction) [register, text width = 7cm] at (9.65, -2.5) Error Correction; (recovered) [register, text width = 4cm] at (8, -3.5) $\mathscr{U}(\left|\phi\right\rangle\left|0\right\rangle)$; (recovered error) [register, text width = 3cm] at (11.75, -3.5) $\left|\text{Error}\right\rangle$; (reconstructed) [register, text width = 2cm] at (0, -3.5) $\left|\phi\right\rangle$; (reconstructed zero) [register, text width = 2cm] at (2.25, -3.5) $\left|0\right\rangle$;
`corrupted.south)` +-1.5 , 0) coordinate (corrupted south); `correction.north)` +-3.15, 0) coordinate (correction north); **x**in 0, 10, ..., 180 `corruptedsouth)` +**x**pt, 0 cm) coordinate (corrupted south **x**) ; `correctionnorth)` +**x**pt, 0 cm) coordinate (correction north **x**) ; (corrupted south **x**) – (correction north **x**); `correction.south)` +-3.15, 0) coordinate (correction south); `recovered.north)` +-1.5 , 0) coordinate (recovered north); **x**in 0, 10, ..., 180 `correctionsouth)` +**x**pt, 0 cm) coordinate (correction south **x**); `recoverednorth)` +**x**pt, 0 cm) coordinate (recovered north **x**); (correction south **x**) – (recovered north **x**); [-¿, thick] (data zero.east) to node $\mathscr{U}$ (codeword.west); [-¿, thick] (codeword.south) to node Corruption (corrupted.north); [-¿, thick] (recovered.west) to node [swap] $\mathscr{U}^{-1}$ (reconstructed zero.east);

Figure 2: Quantum Error Correction

[scale=1, auto] (data) [register, text width = 3cm] at (0, 0) $\left|\text{Original}\right\rangle$; (environment) [register, text width = 3cm] at (3.25, 0) $\left|\text{Environment}\right\rangle$; (corruption) [register, text width = 6.25cm] at (1.625, -1) $\mathscr{U}'$; (corrupted) [register, text width = 3cm] at (0, -2) $\left|\text{Corrupted}\right\rangle$;
`data.south)` +-1.25,0) coordinate (data south); `corruption.north)` +-2.875,0) coordinate (corruption north west); **x**in 0, 10, ..., 70 `datasouth)` +**x**pt, 0 pt) coordinate (data south **x**); `corruptionnorthwest)` +**x**pt, 0 pt) coordinate (corruption north west **x**); (data south **x**) – (corruption north west **x**); `corruption.south)` +-2.875,0) coordinate (corruption south); `corrupted.north)` +-1.25,0) coordinate (corrupted north); **x**in 0, 10, ..., 70 `corruptionsouth)` +**x**pt, 0 pt) coordinate (corruption south **x**); `corruptednorth)` +**x**pt, 0 pt) coordinate (corrupted north **x**); (corruption south **x**) – (corrupted north **x**); `environment.south)` +1.25, 0) coordinate (environment south); `corruption.north)` +2.875, 0) coordinate (corruption north east); **x**in 0, 10, ..., 70 `environmentsouth)` +-**x**pt, 0 pt) coordinate (data south **x**); `corruptionnortheast)` +-**x**pt, 0 pt) coordinate (corruption north east **x**); (data south **x**) – (corruption north east **x**);
`corruption.south)` +2.875, 0) coordinate (corruption south east); **x**in 0, 10, ..., 70 (corruption south east) + (-**x**pt, 0 pt) – ++ (-**x**pt, -.8cm);

Figure 3: Noise Model for a General Quantum Error

And for a system with n qubits, all linear transformations are spanned by all n-fold tensor products of the above four matrices. Therefore, to correct a general quantum error, it is enough to be able to correct bit flips and phase flips.

8 Classical Error Correction Codes

We review the elementary theory of (binary) classical error correcting codes. An error correcting code C is a collection of codewords in $\mathbf{F}_2^n$ (i.e. $C \subseteq \mathbf{F}_2^n$), where $\mathbf{F}_2$ is the finite field with two elements. We will only be interested in linear codes, where C is in fact a linear subspace of $\mathbf{F}_2^n$, regarded as an n dimensional vector space over $\mathbf{F}_2$. If we denote $k := \dim C$ to be the dimension of C as a linear space, we think of the encoding as stretching a length k message into a length n codeword (i.e. a linear map from $\mathbf{F}_2^k$ to $\mathbf{F}_2^n$).

The distance of a code $d := \min_{c_1 \neq c_2 \in C} \text{Hamming-distance}(c_1, c_2)$ is the minimum hamming distance (i.e. number of unequal symbols) between two distinct codewords in C. For linear codes, the distance $d = \min_{c \neq 0 \in C} \text{Hamming-weight}(c)$ is given by the minimum hamming weight (i.e. number of ones) of a non-zero codeword. As can be easily seen, an error correcting code can detect up to $d-1$ errors, and can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors.

We can associate with a linear code a k by n matrix G, known as the generator matrix. The encoding of a length k message $m \in \mathbf{F}_2^k$ is given by the length n codeword $c \in \mathbf{F}_2^n$, where $mG = c$. Similarly, we can associate with a linear code an n by k matrix P, known as the parity check matrix. The columns of P generates $C^\perp$, so a word $c \in \mathbf{F}_2^n$ is a valid codeword (i.e. $c \in C$) iff $cP = 0$. Assume that a valid codeword $c \in C$ is corrupted by some error $e \in \mathbf{F}_2^n$, forming a word $c+e$. We can use the parity check matrix to detect errors, simply by computing $(c+e)P$. By properties of P, we know that $(c+e)P = cP + eP = eP$, which is known as the syndrome. We can detect errors by checking if $eP = 0$ (provided that the hamming weight of e is at most $d-1$), and can correct errors by deducing a unique e from eP (provided that the hamming weight of e is at most $\lfloor \frac{d-1}{2} \rfloor$).

9 CSS code

We describe a quantum error correcting code due to Calderbank, Shor and Stein.

Our starting point is two linear error correcting codes C_1 and C_2, where $C_1 \subseteq C_2 \subseteq \mathbf{F}_2^m$ with $|C_1| = 2^{k_1}$ and $|C_2| = 2^{k_2}$. We require that the second code C_2 and the dual code of the first code $C_1^\perp$ be good error correcting codes, each correcting up to t errors.

In our quantum error correcting code, the codewords are cosets of C_1 in C_2, having the form

$$\big|C_1 + x\big\rangle = \frac{1}{\sqrt{|C_1|}} \sum_{y \in C_1} \big|y + x\big\rangle,$$

for some $x \in C_2$. Hence, the quantum error correcting code has $2^{k_2 - k_1}$ codewords. CSS code encodes $k_2 - k_1$ qubits into m qubits and can correct up to t qubit errors. The theory of classical error correcting codes shows that we can find codes C_1 and C_2 with reasonable value of t, given parameters $k_2 - k_1$ and m.

Recall that it is enough to be able to correct all bit flips and all phase flips. Assume that a codeword $\big|C_1 + x\big\rangle$ is first corrupted by a phase flip error e_p and then a bit flip error e_b, giving the corrupted state

$$\frac{1}{\sqrt{|C_1|}} \sum_{y \in C_1} (-1)^{(y+x)\cdot e_p} \big|y + x + e_b\big\rangle = \sum_{y \in C_1} \beta_y \big|y + x + e_b\big\rangle.$$

Our quantum error correcting algorithm has two stages, one for bit flips and one for phase flips.

In the first stage we correct up to t bit flips. Essentially, we apply the parity check matrix P_{C_2} for code C_2 to compute the syndrome, allowing us to undo the bit flips.

$$\sum_{y \in C_1} \beta_y \big|y + x + e_b\big\rangle \big|0\big\rangle \xrightarrow{\text{apply } P_{C_2}} \sum_{y \in C_1} \beta_y \big|y + x + e_b\big\rangle \big|e_b P_{C_2}\big\rangle \xrightarrow{\text{compute and undo } e_b} \sum_{y \in C_1} \beta_y \big|y + x\big\rangle \big|e_b P_{C_2}\big\rangle.$$

We discard (or measure) the second register to obtain the state $\sum_{y \in C_1} \beta_y \big|y + x\big\rangle$, which is only corrupted by phase flips.

In the second stage we correct up to t phase flips. Essentially, we convert phase flips into bit flips and repeat the first stage to correct the errors. Recall that $\big|C_1 + x\big\rangle$ is mapped to $\chi(x) \cdot \big|C_1^\perp\big\rangle$ under Fourier transform, meaning that Fourier transform turns bit flips into phase flips, and vice versa. Indeed,

$$\sum_{y \in C_1} \beta_y \big|y + x\big\rangle = \frac{1}{\sqrt{|C_1|}} \sum_{y \in C_1} (-1)^{(y+x)\cdot e_p} \big|y + x\big\rangle \xrightarrow{\text{Fourier transform}} \frac{1}{\sqrt{|C_1^\perp|}} \sum_{k \in C_1^\perp} (-1)^{x \cdot k} \big|k + e_p\big\rangle = \sum_{k \in C_1^\perp} \alpha_k \big|k + e_p\big\rangle.$$

We repeat the first stage, using the parity check matrix $P_{C_1^\perp}$ of the code $C_1^\perp$ instead.

$$\sum_{k \in C_1^\perp} \alpha_k \big|k + e_p\big\rangle \big|0\big\rangle \xrightarrow{\text{apply } P_{C_1^\perp}} \sum_{k \in C_1^\perp} \alpha_k \big|k + e_p\big\rangle \big|e_p P_{C_1^\perp}\big\rangle \xrightarrow{\text{compute and undo } e_p} \sum_{k \in C_1^\perp} \alpha_k \big|k\big\rangle \big|e_p P_{C_1^\perp}\big\rangle.$$

After discarding (or measuring) the second register to get $\sum_{k \in C_1^\perp} \alpha_k \big|k\big\rangle$, we recover the codeword with a Fourier transform.

$$\sum_{k \in C_1^\perp} \alpha_k \big|k\big\rangle = \frac{1}{\sqrt{|C_1^\perp|}} \sum_{k \in C_1^\perp} (-1)^{x \cdot k} \big|k\big\rangle \xrightarrow{\text{Fourier transform}} \frac{1}{\sqrt{|C_1|}} \sum_{y \in C_1} \big|y + x\big\rangle = \big|C_1 + x\big\rangle.$$