Quantum Error Correction

Dilip Sau

February, 2022

1 Introduction

Quantum error correction (QEC) comes from the marriage of quantum mechanics with the classical theory of error correcting codes. Error correction is a central concept in classical information theory, and quantum error correction is similarly foundations in quantum information theory. Both are concerned with the fundamental problem of communication, and/or information storage, in the presence of noise. The code words which arise in QEC are also interesting objects in their own right, displaying rich forms of entanglement.

2 Basic Idea of classical error correction

Suppose we want send a classical bit from one place to another place through a classical communication channel. Due to the noise of the channel the bit may be flipped. Let say the bit flipping probability of one bit is $p > 0$, therefore with probability $(1-p)$ we can get the correct transmitted bit. Such a channel is called binary symmetric channel. A simple means of protecting the bit against the effects of noise in the binary symmetric channel is to replace the bit we wish to protect with three copies of itself:

$$0 \to 000$$
$$1 \to 111$$

The bit strings 000 and 111 are some time referred to as the logical 0 and logical 1 , since the play the role of 0 and 1 respectively. We now send all three bits trough the channel. At the receiver's end of the channel three bits are output, and the receiver has decide what the value of the original bit was. Suppose 001 were output from the channel. Provided the probability p of a bit flip is not to high , it very likely that the third bit was flipped by the channel, and that 0 was the bit that was send.

This type of decoding is called majority decoding, he the decoded output 0, 1 is depend upon the majority of 0 and 1 in that string, this gives the correct until two or more bit are flipped. So, the probability that two or more bit are flipped is $3p^2(1-p) + p^3$, so the probability of error is $p_e = 3p^2 - 2p^3$. Without encoding, the probability of an error was p, so the code makes the transmission more reliable provided $p_e < p$, which occurs whenever $p < 1/2$.

3 Three bit code

We will begin by analysing in detail the workings of the most simple quantum error correcting code. Exactly what is meant by a quantum error correcting code will become apparent. Suppose a source A

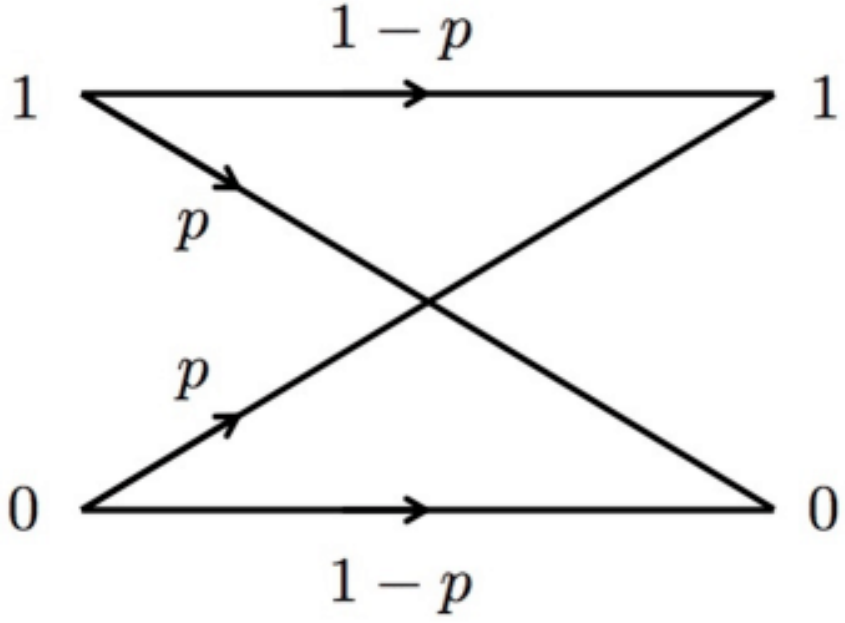

Figure 1: Binary symmetric channel

wishes transmit quantum information via a noisy communication channel to a receiver B. Obviously the channel must be noisy in practice since no channel is perfectly noise-free. However, in order to do better than merely sending quantum bits down the channel, we must know something about the noise. For this introductory section, the following properties will be assumed: the noise acts on each qubit independently, and for a given qubit has an effect chosen at random between leaving the qubit's state unchanged (probability $1-p$) and applying a Pauli σ_x operator (probability $p < 1/2$). This is a very artificial type of noise, but once we can correct it, we will find that our correction can also offer useful results for much more realistic types of noise.

3.1 Examples

The simplest quantum error correction method is summarised. We adopt the convention of calling the source Alice and the receiver Bob. The state of any qubit that Alice wishes to transmit can be written without loss of generality $a|0\rangle + b|1\rangle$ Alice prepares two further qubits in the state $|0\rangle$, so the initial state of all three is $a|000\rangle + b|100\rangle$ Alice now operates a controlled-not gate from the first qubit to the second, producing $a|000\rangle + b|110\rangle$, followed by a controlled-not gate from the first qubit to the third, producing $a|000\rangle + b|111\rangle$. Finally, Alice sends all three qubits down the channel.
Bob receives the three qubits, but they have been acted on by the noise in the channel. Their state is one of the following:

State	Probability
$a\|000\rangle + b\|111\rangle$	$(1-p)^3$
$a\|100\rangle + b\|011\rangle$	$p(1-p)^2$
$a\|010\rangle + b\|101\rangle$	$p(1-p)^2$
$a\|001\rangle + b\|110\rangle$	$p(1-p)^2$
$a\|110\rangle + b\|001\rangle$	$p^2(1-p)$
$a\|101\rangle + b\|010\rangle$	$p^2(1-p)$
$a\|011\rangle + b\|100\rangle$	$p^2(1-p)$
$a\|111\rangle + b\|000\rangle$	p^3

(1)

Bob now introduces two more qubits of his own, prepared in the state $|00\rangle$ This extra pair of qubits, referred to as an ancilla, is not strictly necessary, but makes error correction easier to understand and becomes necessary when fault-tolerant methods are needed. Bob uses the ancilla to gather information about the noise. He first carries out controlled-nots from the first and second received qubits to the first ancilla qubit, then from the first and third received qubits to the second ancilla bit. The total state of all five qubits is now

State	Probability
$(a\|000\rangle + b\|111\rangle)\|00\rangle$	$(1-p)^3$
$(a\|100\rangle + b\|011\rangle)\|11\rangle$	$p(1-p)^2$
$(a\|010\rangle + b\|101\rangle)\|10\rangle$	$p(1-p)^2$
$(a\|001\rangle + b\|110\rangle)\|01\rangle$	$p(1-p)^2$
$(a\|110\rangle + b\|001\rangle)\|01\rangle$	$p^2(1-p)$
$(a\|101\rangle + b\|010\rangle)\|10\rangle$	$p^2(1-p)$
$(a\|011\rangle + b\|100\rangle)\|11\rangle$	$p^2(1-p)$
$(a\|111\rangle + b\|000\rangle)\|00\rangle$	p^3

(2)

Bob measures the two ancilla bits in the basis $\{|0\rangle, |1\rangle\}$. This gives him two classical bits of information. This information is called the error syndrome, since it helps to diagnose the errors in the received qubits. Bob's next action is as follows:

Measure Syndrome	Action
00	do nothing
01	applying σ_x on the third qubit
10	applying σ_x on the second qubit
11	applying σ_x on the first qubit

Suppose for example that Bob's measurements give 10 (i.e. the ancilla state is projected onto $|10\rangle$). We see that the state of the received qubits must be either $a|010\rangle + b|101\rangle$ probability $(p(1-p)^2)$ or $a|101\rangle + b|010\rangle$ (probability $p^2(1-p)$). Since the former is more likely, Bob corrects the state by applying a Pauli σ_x operator to the second qubit. He thus obtains either $a|000\rangle + b|111\rangle$(most likely) or $a|111\rangle + b|000\rangle$. Finally, to extract the qubit which Alice sent, Bob applies controlled-not from the first qubit to the second and third, obtaining either $(a|0\rangle + b|1\rangle)|001\rangle$ or $(a|0\rangle + b|1\rangle)$ $|00\rangle$. Therefore Bob has either the exact qubit sent by Alice, or Alice's qubit operated on by σ_x. Bob does not know which he has, but the important point is that the method has a probability of success greater than $1-p$. The correction is designed to succeed whenever either no or just one qubit is corrupted by the channel, which are the most likely possibilities. The failure probability is the probability that at least two qubits are corrupted by the channel, which is $3p^2(1_p) + p^3 = 3p^2 - 2p^3$, i.e. less than p (as long as $p < 1/2$).

To summarise, Alice communicates a single general qubit by expressing its state as a joint state of three qubits, which are then sent to Bob. Bob first applies error correction, then extracts a single qubit state. The probability that he fails to obtain Alice's original state is $O(p^2)$, whereas it would have been $O(p)$ if no error correction method had been used. We will see later that with more qubits the same basic ideas lead to much more powerful noise suppression, but it is worth noting that we already have quite an impressive result: by using just three times as many qubits, we reduce the error probability by a factor $\sim 1/3p$, i.e. a factor ~ 30 for $p = 0.01$, ~ 300 for $p = 0.001$, and so on.

4 Binary fields and discrete vector spaces

Classical error correction is concerned with classical bits, not quantum states. The mathematical treatment is based on the fact that linear algebraic operations such as addition and multiplication can be consistently defined using finite rather than infinite sets of integers, by using modular arithmetic. The simplest case, that of modulo 2 arithmetic, will cover almost everything in this article. The addition operation is defined by $0+0 = 0$, $0+1 = 1+0 = 1$, $1+1 = 0$. The set $\{0, 1\}$ is a group under this operation, since 0 is the identity element, both elements are their own inverse and the operation is associative. The set $\{0, 1\}$ is also a group under multiplication, with identity element 1. Furthermore, we can also define division (except division by zero) and subtraction, and the commutative and distributive laws hold. These properties together define a finite field, also called a Galois field. Thus the set $\{0, 1\}$ is referred to as the field $GF(2)$, where addition and multiplication are as defined.

A string of n bits is considered to be a vector of n components, for example 011 is the vector $(0, 1, 1)$. Vector addition is carried out by the standard method of adding components, for example $(0, 1, 1) + (1, 0, 1) = (0 + 1, 1 + 0, 1 + 1) = (1, 1, 0)$. It is easy to see that this operation is equivalent to the exclusive-or operation $\oplus$ carried out bit-wise between the binary strings: $011 \oplus 101 = 110$. Note that $u + u \equiv 0$ and $u - v = u + v$ (prove by adding v to both sides).

We can define the inner product (or scalar product) by the standard rule of multiplying corresponding components, and summing the results: $(1, 1, 0, 1)(1, 0, 0, 1) = 1 + 0 + 0 + 1 = 0$. Note that

all the arithmetic is done by the rules of the Galois field, so the final answer is only ever 0 or 1. The inner product is also called a parity check or check sum since it indicates whether the second vector 'satisfies' the parity check specified by the first vector (or equivalently whether the first vector satisfies the parity check specified by the second). To satisfy a parity check u, a vector v must have an even number of $1's$ at the positions (coordinates) specified by the $1's$ in u. If u and v are row vectors then $u\dot{v} = uv^T$ where T is the transpose operation.

The number of non-zero components of a binary vector u is important in what follows, and is called the weight (or Hamming weight), written $wt(u)$. For example, $wt(0001101) = 3$. The number of places (coordinates) where two vectors differ is called the Hamming distance between the vectors; the distance between u and v is equal to $wt(u + v)$. There are 2^n vectors of n components (stated in other language, there are 2^n n-bit words). This set of vectors forms a linear vector space, sometimes called Hamming space. It is a discrete vector space since vector components are are only ever equal to 0 or 1. The vectors point to the vertices of a square lattice in n dimensions. The space is spanned by any set of n linearly independent vectors. The most obvious set which spans the space is $100000, 010000, 001000, ..., 000001$.

There are sub spaces within Hamming space. A linear subspace C is any set of vectors which is closed under addition, i.e. $u + v \in C \forall u, v \in C$. For example the set $0000, 0011, 1100, 1111$ is a 2^2 linear subspace of the $2^4 4$ Hamming space. A linear subspace containing 2^k vectors is spanned by k linearly independent vectors (for example 0011 and 1100 in the case just given). Any linear subspace is thus completely specified by its generator matrix G, which is just the matrix whose k rows are any k vectors which span the space. We can always linearly combine rows to get an equivalent generator matrix, for example

$$G = \begin{pmatrix} 0011 \\ 1100 \end{pmatrix} = \begin{pmatrix} 0011 \\ 1111 \end{pmatrix} \tag{3}$$

The minimum distance d of a subspace is the smallest Hamming distance between any two members of the subspace. If the two closest vectors are u and v, then $d = wt(u + v)$. For the case of a linear space, $w = u + v$ is also a member of the space. From this we deduce that the minimum distance of a linear space is equal to the smallest weight of a non-zero member of the space. This fact is useful in calculating the value of d, since it is much easier to find the minimum weight than to evaluate all the distances in the space.Now, if $u.v = 0$ and $u.w = 0$ then $u.(v + w) = 0$. From this it follows, that if all the rows of a generator satisfy the parity check u, then all so do all the vectors in the subspace. Any given parity check u divides Hamming space exactly in half, into those vectors which satisfy u and those that do not. Therefore, the 2^k vectors of a linear subspace in 2^n Hamming space can satisfy at most $n - k$ linearly independent parity checks. These parity checks together form the parity check matrix H, which is another way to define the linear subspace. H has n columns and $n - k$ rows. For any given subspace, the check and generator matrices are related by

$$HG^T = 0 \tag{4}$$

where G^T is the transpose of G, and 0 is the $(n - k) \times k$ zero matrix.

The simple error correction method described in the previous section is based around the very simple binary vector space $000, 111$. Its generator matrix is $G = (111)$ and the parity check matrix is,

$$H = \begin{pmatrix} 110 \\ 101 \end{pmatrix} \tag{5}$$

A useful relationship enables us to derive each of H and G from the other. It is always possible to convert G to the form $G = (I_k, A)$ where I_k is the $k \times k$ identity matrix, and A is the rest of G (so A is $k \times n - k$). To do the conversion we can linearly combine rows of G, and if necessary swap columns of G. Once we have the right form for G, then H can be written down immediately, it is $H = (A^T, I_{n-k})$. The last concept which we will need in what follows is that of the dual. The dual space $C^\perp$ is the set of all vectors u which have zero inner product with all vectors in C, $u.v = 0 \forall v \in C$. It is simple to deduce that the parity check matrix of C is the generator matrix of $C^\perp$ and vice versa. If $H = G$ then $C = C^\perp$, such spaces are termed self-dual.

The notation (n, m, d) is a shorthand for a set of mn-bit vectors having minimum distance d. For linear vector spaces, the notation $[n, k, d]$ is used, where k is now the dimension of the vector space, so it contains 2^k vectors.Let us conclude this section with another example of a linear binary vector space which will be important in what follows. It is a $[7, 4, 3]$ space discovered by Hamming. The generator matrix is

$$G = \begin{pmatrix} 1010101 \\ 0110011 \\ 0001111 \\ 1110000 \end{pmatrix} \tag{6}$$

so the sixteen number of space are

$$\begin{array}{cccc} 0000000 & 1010101 & 0110011 & 1100110 \\ 0001111 & 1011010 & 0111100 & 1101001 \\ 1110000 & 0100101 & 1000011 & 0010110 \\ 1111111 & 0101010 & 1001100 & 0011001 \end{array} \tag{7}$$

these have been written in the following order: first the zero vector, then the first row of G. Next add the second row of G to the two vectors so far obtained, then add the third row to the four vectors previously obtained, and so on. We can see at a glance that the minimum distance is 3 since the minimum non-zero weight is 3. The parity check matrix is

$$H = \begin{pmatrix} 1010101 \\ 0110011 \\ 0001111 \end{pmatrix} \tag{8}$$

It is simple to confirm that HG^T. Note also that since H is made of rows of G, this code contains its dual: $C^\perp \in C$

5 Classical error correction

Classical communication can be considered without loss of generality to consist in the communication of strings of binary digits, i.e. the binary vectors introduced in the previous section. A given binary vector, also called a binary word, which we wish to send from A to B, is called a message. A noisy communication channel will corrupt the message, but since the message u is a binary vector the only effect the noise can have is to change it to some other binary vector $u^{'}$. The difference $e = u^{'} - u$ is called the error vector. Error correction consists in deducing u from $u^{'}$.

5.1 Error correcting code

A classical error correcting code is a set of words, that is, a binary vector space. It need not necessarily be linear, though we will be concerned almost exclusively with linear codes. Each error correcting code C allows correction of a certain set $\mathcal{S} \equiv \{e_i\}$ of error vectors. The correctable errors are those which satisfy

$$u + e_i \neq v + e_j \forall u, v \in C(u \neq v) \tag{9}$$

The case of no error, $e = 0$, is included in $\mathcal{S}$, so that error-free messages are 'correctable'. To achieve error correction, we use the fact that each message u is corrupted to $u^{'} = u + e$ for some $e \in \mathcal{S}$. However, the receiver can deduce u unambiguously from $u + e$ since by condition (9), no other message v could have been corrupted to $u + e$, as long as the channel only generates correctable error vectors. In practice a noisy channel causes both correctable and uncorrectable errors, and the problem is to match the code to the channel, so that the errors most likely to be generated by the channel are those the code can correct.

Suppose the noise affects each bit independently, with a fixed error probability $p < 1/2$. This noise has less structure than that we just considered, but it still has some predictable features, the most important being that the most likely error vectors are those with the smallest weight. We use the code $C = \{000, 111\}$. The errors which the channel can produce are, in order of decreasing probability, $\{000, 001, 010, 100, 011, 101, 011, 111\}$. With $n = 3$ and $m = 2$, the set of correctable errors can have at most $2^3/2 = 4$ members, and these are $000, 001, 010, 100$. This is the classical equivalent of the quantum code described next.

5.2 Minimum distance coding

The noisy channel just described is called the binary symmetric channel. This is a binary channel (the only kind we are considering) in which the noise affects each bit sent down the channel independently. It is furthermore symmetric, meaning that the channel causes errors $0 \to 1$ and $1 \to 0$ with equal probability.If n bits are sent down a binary symmetric channel, the probability that m of them are flipped ($0 \leftrightarrow 1$) is the probability for m independent events in n opportunities:

$$C(n, m)p^m(1 - p)^{n-m} \tag{10}$$

where the binomial coefficient $C(n, m) = \frac{n!}{m!(n-m)!}$.

The binary symmetric channel is important because other types of noise can be treated as a combination of a structured component and a random component. The structured component can be tackled in more efficient ways, for example burst-error correcting codes, and then the remaining random component forms a problem equivalent to that of the binary symmetric channel.To code for the binary symmetric channel, we clearly need a code in which error vectors of small weight are correctable, since these are the most likely ones. A cod which corrects all error vectors of weight up to and including t is called a t-error correcting code. A simple but important observation is the following:
A code of minimum distance d can correct all error vectors of weight less than or equal to t if and only if $d > 2t$.

5.3 Bounds on the size of codes

In order to communicate k bits of information using an error correcting code, $n > k$ bits must be sent down the channel. The ratio k/n, called the rate of the code, gives a measure of the cost of error correction. Various bounds on k/n can be deduced. In a Hamming space of n dimensions, that is, one consisting P of n-bit vectors, there are $C(n,t)$ error vectors t of weight t. Any member of a t-error correcting code has $\sum_{i=0}^{t} C(n,i)$ erroneous versions (including the error-free version) which must all be distinct from the other members of the code and their erroneous versions if correction is to be possible. They are also distinct from each other because

$$e_1 \neq e_2 \Rightarrow u + e_1 \neq u + e_2 \tag{11}$$

However, there are only 2 possible vectors in the whole Hamming space, therefore the number of vectors m in n-bit t-error correcting codes is limited by the Hamming bound

$$m \sum_{i=0}^{t} C(n,i) \leq 2^n \tag{12}$$

This is also called the sphere packing bound because the problem is equivalent to that of packing as many spheres as possible in the n-dimensional rectangular space, where each sphere has radius t. For linear codes $m = 2^k$ so the Hamming bound becomes

$$k \leq n - \log_2 \left(\sum_{i=0}^{t} C(n,i) \right) \tag{13}$$

From this we can deduce in limit of large n, k, t:

$$\frac{k}{n} \leq n \left(1 - H\left(\frac{t}{n} \right) \right) (1 - \eta) \tag{14}$$

where $\eta \to 0$ as $n \to \infty$ and $H(x)$ is the entropy function

$$H(x) \equiv -x \log_2 x - (1-x) \log_2 (1-x). \tag{15}$$

The Hamming bound makes precise the intuitively obvious fact that error correcting codes can not achieve arbitrarily high rate k/n while still retaining their correction ability. As yet we have no definite guide as to whether good codes exist in general. A very useful result is the Gilbert-Varshamov bound: it can be shown that for given n, d, there exists a linear $[n,k,d]$ code provided

$$2^k \sum_{i=0}^{d-2} C(n-1,i) \leq 2^n \tag{16}$$

In the limit of large n, k, d and putting $t = d/2$, this becomes

$$\frac{k}{n} \geq n \left(1 - H\left(\frac{2t}{n} \right) \right) (1 - \eta) \tag{17}$$

where $\eta \to 0$ as $n \to \infty$. It can be shown that there exists an infinite sequence of $[n,k,d]$ codes satisfying with $d/n \geq \delta$ if $0 \leq \delta \leq 1/2$.

5.4 Linear codes, error syndrome

So far the only error correction method we have mentioned is the simple idea of a look-up table, in which a received vector w is compared with all the code vectors $u \in C$ and their erroneous versions $u + e$, $e \in \mathcal{S}$, until a match $w = u + e$ is found, in which case the vector is corrected to u. This method makes inefficient use of either time or memory resources since there are 2^n vectors $u+e$.

For linear codes, a great improvement is to calculate the error syndrome s given by

$$s = Hw^T \tag{18}$$

where H is the parity check matrix. Since H is a $(n-k) \times n$ matrix, and w is an n bit row vector, the syndrome $s$is an $n-k$ bit column vector. The transpose of w is needed to allow the normal rules of matrix multiplication, though the notation $H.w$ is sometimes also used. Consider

$$s = H(u+e)^T = Hu^T + He^T = He^T \tag{19}$$

where we used eq. (4) for the second step. This shows that the syndrome depends only on the error vector, not on the transmitted word. If we could deduce the error from the syndrome, which will be shown next, then we only have 2^{n-k} syndromes to look up, instead of 2^n erroneous words. Furthermore, many codes can be constructed in such a way that the error vector can be deduced from the syndrome by analytical methods, such as the solution of a set of simultaneous equations.

Proof that the error can be deduced from the syndrome. The parity check matrix consists of $n-k$ linearly independent parity check vectors. Each check vector divides Hamming space in half, into those vectors which satisfy the check, and those which do not. Hence, there are exactly 2^k vectors in Hamming space which have any given syndrome s. Using eq. (19), these vectors must be the vectors $u + e_s$ where $u \in C$ is one of the 2^k members of the code C. Hence the only error vectors which give the syndrome s are the members of the set $e_s + u$, $u \in C$. Such a set is called a co set. The syndrome cannot distinguish among errors in a given co set. In choosing which errors will be correctable by the code, we can choose at most one from each co set. Then, we have proved that each correctable error is uniquely determined by its syndrome.

To conclude, let us consider an example using the $[7, 4, 3]$ Hamming code given at the end of section 3. Since this code has minimum distance 3, it is a single error correcting code. The number of code vectors is limited by the Hamming bound to $2^7/(C(7,0)+C(7,1)) = 2^7/(1+7) = 2^4$. Since there are indeed 2^4 vectors the code saturates the bound; such codes are called perfect. The set of correctable errors is $0000000, 0000001, 0000010, 0000100, 0001000, 0010000, 0100000, 1000000$. Suppose the message is 0110011 and the error is 0100000. The received word is $0110011 + 0100000 = 0010011$. The syndrome, using eq. (8), is

$$H(0010011)^T = (010)^T \tag{20}$$

The only word of weight ≤ 1 which fails the second parity check and passes the others is 0100000, so this is the deduced error. We thus deduce the message to be $0010011 - 0100000 = 0110011$.

6 Quantum error correction

QEC is based on three central ideas: digitization of noise, the manipulation of error operators and syndromes, and quantum error correcting code (QECC) construction. The degree of success of QEC relies on the physics of noise; we will turn to this after discussing the three central ideas.

6.1 Digitization of noise

"Digitization of noise" is based on the observation that any interaction between a set of qubits and another system (such as the environment) can be expressed in the form:

$$|\phi\rangle |\psi_0\rangle_e \rightarrow \sum_i (E_i |\phi\rangle) |\psi_i\rangle_e \tag{21}$$

where each 'error operator' E_i is a tensor product of Pauli operators acting on the qubits,$|\psi\rangle$ is the initial state of the qubits, and $|\psi_i\rangle$ are states of the environment, not necessarily orthogonal or normalised. We thus express general noise and/or decoherence in terms of Pauli operators σ_x,σ_y,σ_z acting on the qubits. These will be written $X \equiv \sigma_x$, $Z \equiv \sigma_z$,$Y \equiv -i\sigma_y = XZ$.

It is worth mulling over equation (21). The statement is true because the Pauli matrices are a complete set: they can describe any transformation of the qubits, and because we allow any sort of entanglement with the environment. You may want to take a moment to write your favourite type of noise process in this way, to familiarise yourself with the approach. Physicists often consider coupling to the environment by starting from a Hamiltonian, or through a density matrix approach; it is important to see that we are being completely general here, so those other methods can be completely encompassed by (21).

To write tensor products of Pauli matrices acting on n qubits, we introduce the notation $X_u Z_v$ where u and v are n-bit binary vectors. The non-zero coordinates of u and v indicate where X and Z operators appear in the product. For example,

$$X \otimes I \otimes Z \otimes Y \otimes X \equiv X_{10011} Z_{00110} \tag{22}$$

Error correction is a process which takes a state such as $E_i |\psi\rangle$ to ψ. Correction of X errors takes $X_u Z_v |\phi\rangle$ to $X_u |\phi\rangle$; correction of Z errors takes $X_u Z_v |\phi\rangle$ to $X_u |\phi\rangle$. Putting all this together, we discover the highly significant fact that to correct the most general possible noise, it is sufficient to correct just X and Z errors. Following for a proof. It may look like we are only correcting unitary 'bit flip' and 'phase flip' rotations, but this is a false impression: actually we will correct everything, including non-unitary relaxation processes!

6.2 Error operators,stabilizer and syndrome extrations

Consider the set I, X, Y, Z consisting of the identity plus the three Pauli operators. The Pauli operators all square to $I : X^2 = Y^2 = Z^2 = I$, and have eigenvalues ± 1. Two members of the set only ever commute ($XI = IX$) or anticommute: $XZ = -ZX$. Tensor products of Pauli operators, i.e. error operators, also square to one and either commute or anticommute. N.B. the term 'error operator' is here just a shorthand for 'product of Pauli operators'; such an operator will sometimes play the role of an error, sometimes of a parity check, c.f. classical coding theory.

If there are n qubits in the quantum system, then error operators will be of length n. The weight of an error operator is the number of terms not equal to I. For example $X_{10011} Z_{00110}$ has length 5, weight 4. Let $\mathcal{H} = \{M\}$ be a set of commuting error operators. Since the operators all commute, they can have simultaneous eigenstates. Let $C = |u\rangle$ be an orthonormal set of simultaneous eigenstates all having eigenvalue +1:

$$M |u\rangle = |u\rangle \, \forall u \in C, \;\; \forall M \in \mathcal{H} \tag{23}$$

The set $\mathcal{C}$ is a quantum error correcting code, and $\mathcal{H}$ is its stabilizer. The orthonormal states $|u\rangle$ are termed code vectors or quantum code words. In what follows, we will restrict attention to the case that $\mathcal{H}$ is a group. Its size is 2^{n-k} , and it is spanned by $n-k$ linearly independent members of $\mathcal{H}$. In this case $\mathcal{C}$ has 2^k members, so it encodes k qubits, since its members span 2^k dimensional subspace of the 2^n dimensional Hilbert space of the whole system. A general state in this subspace, called an encoded state or logical state, can be expressed as a superposition of the code vectors:

$$|\phi\rangle_L = \sum_{u\in\mathcal{C}} a_u |u\rangle \tag{24}$$

Naturally, a given QECC does not allow correction of all possible errors. Each code allows correction of a particular set $\mathcal{S} = \{E\}$ of correctable errors. The task of code construction consists of finding codes whose correctable set includes the errors most likely to occur in a given physical situation. We will turn to this important topic in the next section. First, let us show how the correctable set is related to the stabilizer, and demonstrate how the error correction is actually achieved.

First, error operators in the stabilizer are all correctable, $E \in \mathcal{S} \forall E \in \mathcal{H}$, since these operators actually have no effect on a general logical state (24). If these error operators are themselves the only terms in the noise of the system under consideration, then the QECC is a noise-free subspace, also called decoherence-free subspace of the system.There is a large set of further errors which do change encoded states but are nevertheless correctable by a process of extracting an error syndrome, and then acting on the system depending on what syndrome is obtained. We will show that $\mathcal{S}$ can be any set of errors $\{E_i\}$ such that every product E_1E_2 of two members is either in $\mathcal{H}$, or anti commutes with a member of $\mathcal{H}$. To see this, take the second case first:

$$E_1E_2 = -ME_1E_2 \text{ for some } M \in \mathcal{H}. \tag{25}$$

We say that the combined error operator E_1E_2 is decidable. This can only happen if

$$\text{either } \{ME_1 = -E_1M, ME_2 = E_2M\} \text{or } \{ME_1 = E_1M, ME_2 = -E_2M\} \tag{26}$$

To extract the syndrome we measure all the observable in the stabilizer. To do this, it is sufficient to measure any set of $n-k$ linearly independent $M \in \mathcal{H}$. Note that such a measurement has no effect on a state in the encoded subspace, since such a state is already an eigen state of all these observable. The measurement projects a noisy state onto an eigen state of each M , with eigenvalue ± 1. The string of $n-k$ eigenvalues is the syndrome. Equations (26) guarantee that E_1 and E_2 have different syndromes, and so can be distinguished from each other. For, when the observable M is measured on the corrupted state $E|\phi\rangle_L$, (26) means that a different eigenvalue will be obtained when $E = E_1$ than when $E = E_1$. Therefore, the error can be deduced from the syndrome, and reversed by re-applying the deduced error to the system (taking advantage of the fact that error operators square to 1).

Let us see how this whole process looks when applied to a general noisy encoded state. The noisy state is

$$\sum_i (E_i |\phi\rangle_L) |\psi_i\rangle_e . \tag{27}$$

The syndrome extraction can be done most simply by attaching an $n-k$ qubit ancilla a to the system, and storing in it the eigenvalues by a sequence of CNOT gates and Hadamard rotations. The exact network can be constructed either by thinking in terms of parity check information stored

into the ancilla, or by the following standard eigenvalue measurement method. To extract the $\lambda = \pm 1$ eigenvalue of operator M , prepare an ancilla in $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Operate controlled-$M$ with ancilla as control, system as target, then Hadamard-rotate the ancilla. The final state of the ancilla is $\frac{1}{2}[(1+\lambda)|0\rangle + (1-\lambda)|1\rangle]$. Carrying out this process for the $n-k$ operators M which span $\mathcal{H}$, the effect is to couple system and environment with the ancilla as follows:

$$|0\rangle_a \sum_i (E_i |\phi\rangle_L) |\psi_i\rangle_e \rightarrow \sum_i (s_i |\phi\rangle_L) |\psi_i\rangle_e . \tag{28}$$

The syndromes s_i are $(n-k)$-bit binary strings. So far the treatment is completely general.

Now suppose the E_i all have different syndromes. Then a protective measurement of the ancilla will collapse the sum to a single term taken at random: $|s_i\rangle_a \ (E_i |\phi\rangle_L) |\psi_i\rangle_e$, and will yield s_i as the measurement result. Since there is only one E_i with this syndrome, we can deduce the operator E_i which should now be applied to correct the error. The system is therefore "magically" disentangled from its environment, and perfectly restored to $|\phi\rangle_L$!

6.3 Condition for quantum error correcting Codes

The discussion based on the stabilizer is useful because it focuses attention on operators rather than states. Quantum codewords are nevertheless very interesting states, having a lot of symmetry and interesting forms of entanglement. The codewords in the QECC can readily be shown to allow correction of the set $\mathcal{S}$ if and only if

$$\langle u| E_1 E_2 |v\rangle = 0 \tag{29}$$

$$\langle u| E_1 E_2 |v\rangle = \langle v| E_1 E_2 |v\rangle \tag{30}$$

for all $E_1, E_2 \in \mathcal{S}$ and $|u\rangle, |v\rangle$, $|u\rangle \neq |v\rangle$. These are the requirements for quantum code words which correspond to the requirement (9) for classical codes. In the case that $E_1 E_2$ always anti commutes with a member of the stabilizer, we have $\langle u| E_1 E_2 |u\rangle = \langle u| E_1 E_2 M |u\rangle = -\langle u| M E_1 E_2 |u\rangle = -\langle u| E_1 E_2 |u\rangle$ therefore $\langle u| E_1 E_2 |u\rangle = 0$. This is a non degenerate code; all the code vectors and their erroneous versions are mutually orthogonal, and the quantum Hamming bound must be satisfied.

The first condition says that to be correctable, an error acting on one code word must not produce a state which overlaps with another code word, or with an erroneous version of another code word. This is what we would expect intuitively.
Proof. Let the unitary operator $\mathcal{R}$ describe the whole recovery operation (e.g. syndrome extraction followed by correction), then

$$\begin{aligned} \mathcal{R} |\alpha_j\rangle E_j |v\rangle &= |\alpha_j'\rangle |v\rangle \\ \mathcal{R} |\alpha_i\rangle E_i |u\rangle &= |\alpha_i'\rangle |u\rangle \\ \Rightarrow \langle u| E_i^\dagger \langle \alpha_i| \mathcal{R}^\dagger \mathcal{R} |\alpha_j\rangle E_j |v\rangle &= \langle \alpha_i'|\alpha_j'\rangle \langle u|v\rangle \\ \Rightarrow \langle u| E_i^\dagger E_j |v\rangle &= 0. \end{aligned}$$

where $|\alpha\rangle$ are states of the ancilla, environment and any other systems used during recovery. The last step uses the fact that the recovery must work when $|\alpha_i\rangle = |\alpha_j\rangle$ (as well as when $|\alpha_i\rangle \neq |\alpha_j\rangle$). The second condition, (31) is surprising since it permits one erroneous version of a code word $|u\rangle$ to overlap with another, as long as all the other code words when subject to the same error have

the same overlap with their respective other erroneous version. This is not possible in classical error correction because of eq. (11). Proof of (31):

$$\begin{aligned} \mathcal{R}\left|\alpha\right\rangle E_i\left|u\right\rangle &= \left|\alpha_i^{'}\right\rangle\left|u\right\rangle \\ \Rightarrow \left\langle u\right| E_i^{\dagger}\left\langle\alpha\right|\mathcal{R}^{\dagger}\mathcal{R}\left|\alpha\right\rangle E_j\left|u\right\rangle &= \left\langle\alpha_i^{'}|\alpha_j^{'}\right\rangle \\ \Rightarrow \left\langle u\right| E_i^{\dagger}E_j\left|u\right\rangle &= \left\langle\alpha_i^{'}|\alpha_j^{'}\right\rangle. \end{aligned}$$

The same result is obtained starting from $|v\rangle$, from which (30) is derived.

6.4 Quantum Hamming bound

A t-error correcting quantum code is defined to be a code for which all errors of weight less than or equal to t are correctable. Since there are 3 possible single-qubit errors, the number of error operators of t acting on n qubits is $3^tC(n,t)$. Therefore a t-error correcting code must be able to correct $\sum_{i=0}^{t} 3^iC(n,t)$ error operators. For non degenerate codes,$\langle u| E_1E_2 |v\rangle = 0$ in (31), every code word $|u\rangle$ and all its erroneous versions $M|u\rangle$ must be orthogonal to every other code word and all its erroneous versions. All these orthogonal states can only fit into the 2^n-dimensional Hilbert space of n qubits if

$$\sum_{i=0}^{t} 3^iC(n,t) \leq 2^n. \tag{31}$$

This bound is known as the quantum Hamming bound. For $m = 2^k$ and large n, t it becomes

$$\frac{k}{n} \leq 1 - \frac{t}{n}\log_2 3 - H(\frac{t}{n}). \tag{32}$$

The rate $\frac{k}{n}$ falls to zero at $\frac{t}{n} \simeq 0.18929$. What is the smallest single-error correcting orthogonal quantum code? A code with $m = 2$ code words represents a Hilbert space of one qubit (it 'encodes' a single qubit). Putting $m = 2$ and $t = 1$ in the quantum Hamming bound, we have $1 + 3n \leq 2^{n-1}$ which is saturated by $n = 5$, and indeed a 5-qubit code exists.

7 Conclusion

A very different case in which QEC is also highly successful is when a set of correlated errors, also called burst errors, dominate the system-environment coupling, but we can find a QEC whose stabilizer includes all these correlated errors. This is sometimes called 'error avoiding' rather than 'error correction' since by using such a code, we don't even need to correct the logical state: it is already decoupled from the environment. The general lesson is that the more we know about the environment, and the more structure there exists in the system-environment coupling, the better able we are to find good codes.

[1] Preskill J., Proc. Roy. Soc. Lond. A, 454 (1998) 385, (quant-ph/9705031).
[2] Nielsen M. A. and Chuang I. L., Quantum Computation and Quantum Information (Cambridge University Press, Cambridge) 2000.
[3] Lo H.-K., Popescu S., Spiller T. (Editors), Introduction to quantum computation and information, (World Scientific, Singapore) 1998.