Classical and Quantum Error Correction

Chien Hsing James Wu

David Gottesman

Andrew Landahl

Outline

- Classical and quantum channels
- Overview of error correction
- Classical linear codes
- Quantum codes
- Conclusions

So What's Information Good For?

Transmission through space
(a.k.a. *Communication*)

- Sending Information from here to there
- "Teleporting" (quantum) states

Transmission through time
(a.k.a. *Memory*)

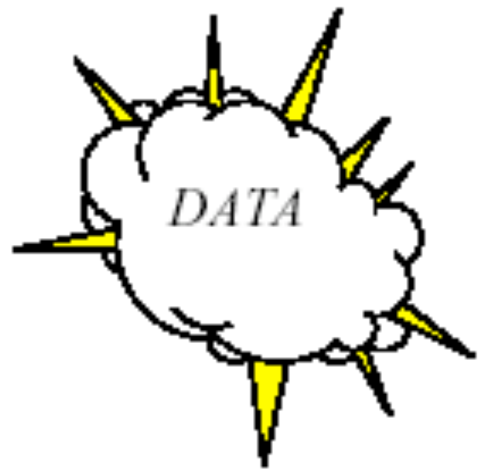
- Preserving the state of the system

Redistribution into convenient forms
(a.k.a. *Computation*)

- Combining remote pieces of information into the answer you seek

But Beware! The Devil can cause ERRORS!

Errors: What does the Devil do?



Classically:

- $Z_2^{\oplus n}$ finite, closed under $\oplus$ implies Devil applies an XOR at most:

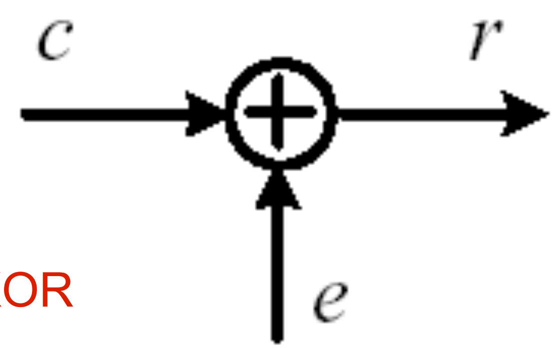
- $Z_2^{\oplus n}$ finite, closed under $\oplus$ implies Devil applies an XOR at most:

$$\begin{array}{rll} & 1101010 & \text{Data} \\ \oplus & \underline{0100001} & \text{Devil} \\ & 1001011 & \text{Output} \end{array}$$

$\therefore$ If we can correct single bit flips, we can correct any error.

Classical Channel Models

Two types of channels are discussed:



XOR

1 Binary channels

$$r = c \oplus e = \begin{cases} c, & e = 0 \quad \text{(no error)} \\ \overline{c}, & e = 1 \quad \text{(error)} \end{cases}$$

2 Gaussian noise channel

$$r = c + e,$$

e : zero-mean white Gaussian noise

Standard addition

Quantum Channel Models

$$|\psi_r\rangle = E|\psi\rangle$$

- Quantum operators are unitary: $EE^H = I$.
- Depolarizing channel:

$$E \in G_n \triangleq \{\pm I, \pm X, \pm Y, \pm Z\}^{\otimes n}$$

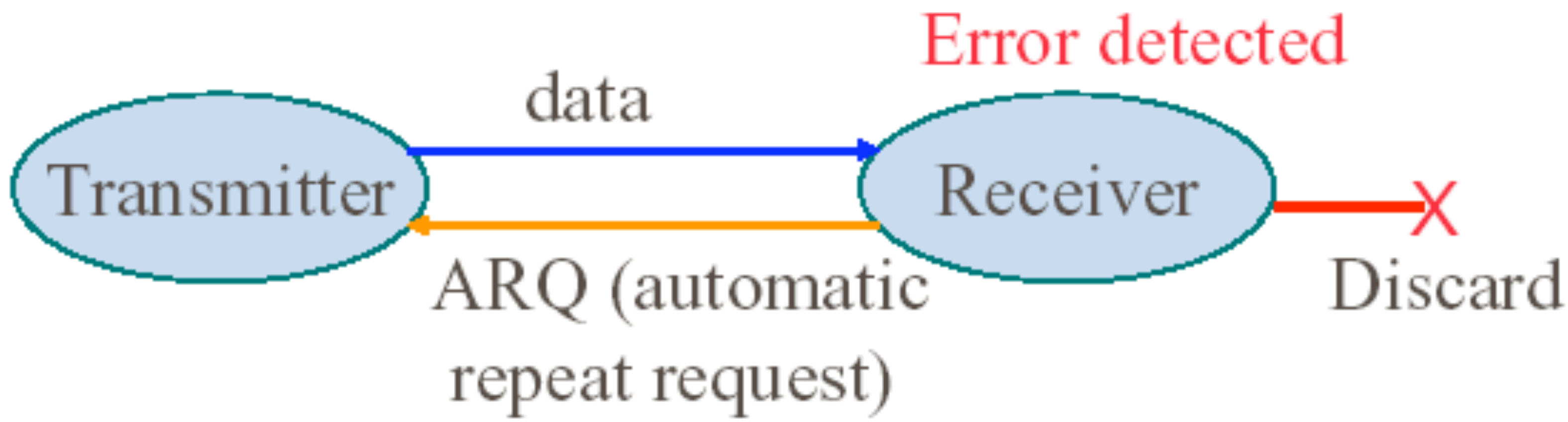
Pauli rotations in each qubit

Computing Power versus Error Control

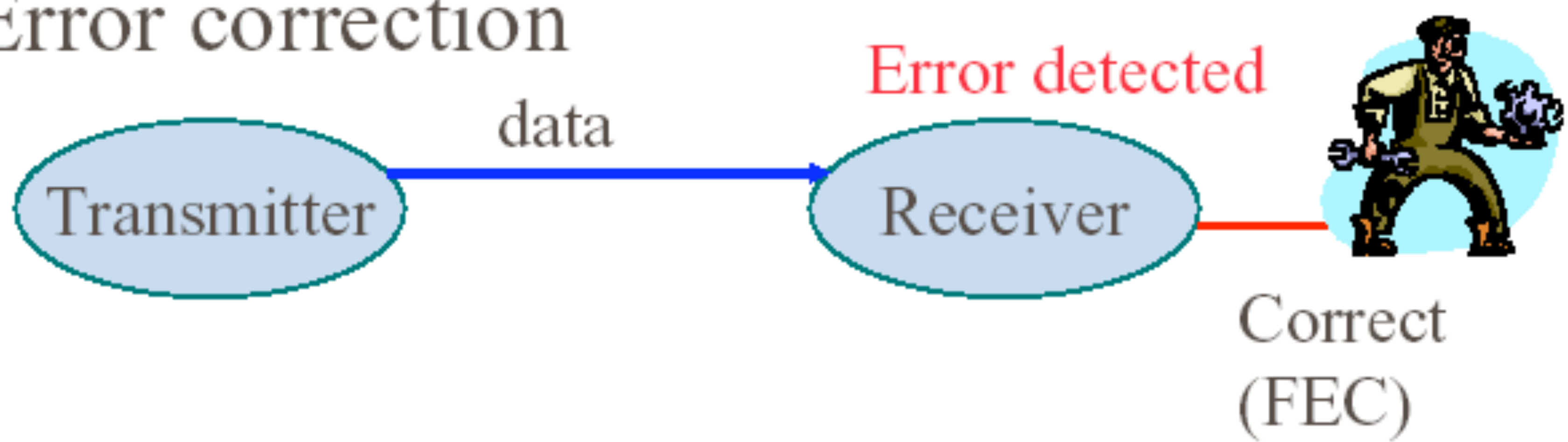
- A quantum state $|\psi\rangle$ with n qubits is a (unit-norm) vector in $\mathcal{H}_{2^n}$.
- The power of quantum computing increases exponentially with the number of qubits.
- The dimension of errors also increases exponentially. Error control is complicated.
- Reliable quantum computers can't be built without error control and fault tolerance.

Basic Concepts in Error Control

- Error detection



- Error correction



Error Control Everywhere

- Parity-check codes for memory chips
- Cyclic redundancy codes (CRC) for packets or cells in networks
- Noise margins in Volts for “0” and “1” in digital circuits
- Reed-Solomon codes for satellites and DVD
- Stabilizer codes for quantum computing

History of Classical Error Correction Codes (ECC)

- Hamming codes (1948) Bell Lab
- Golay codes (1949) Voyager, (Jupiter'79, Saturn'81)
- Reed-Muller ('6x), Reed-Solomon codes
- Convolutional codes
- Turbo codes (1993)
- Low-density parity check codes
- Space-time codes (1998)

Classical Error Correction: Encoding

Idea: Spread the information out

Encoding is a mapping

Please remember our hypercube illustration of codes for interpretation

- $E: Z_2^{\oplus k} \to Z_2^{\oplus n}$ maps k bits to n bits
- $E: v_k \mapsto v_k G$ for *linear* codes. (G is the code *generator*.)
- Columns of G form a basis for the k-dimensional *coding subspace* of $Z_2^{\oplus k}$
- A *codeword* is a vector in this codespace

- The ***weight*** of a codeword v, $wt(v)$, is the number of ones in the codeword

- The ***distance*** between codewords is the weight of their difference

- To correct t errors, the minimum distance between codewords must be $d = 2t + 1$. (d is the distance of the code)

$$v \xrightarrow{t}) \quad (\xleftarrow{t} w$$

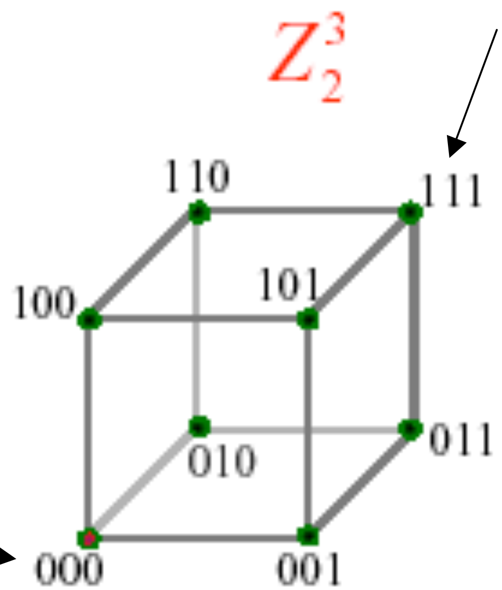
Forms an $[n, k, d]$ code

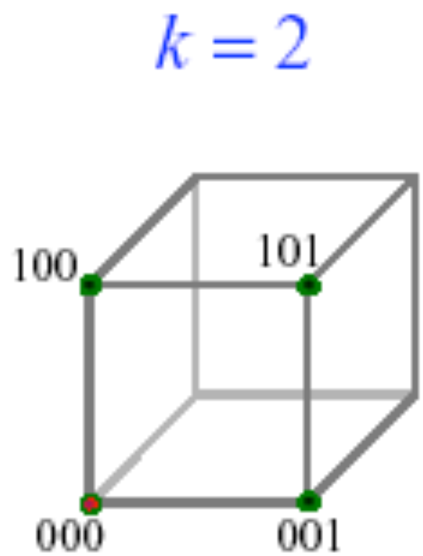
Draw yourself hypercube pictures for these, illustrate our (3,1,1) code from previous lecture

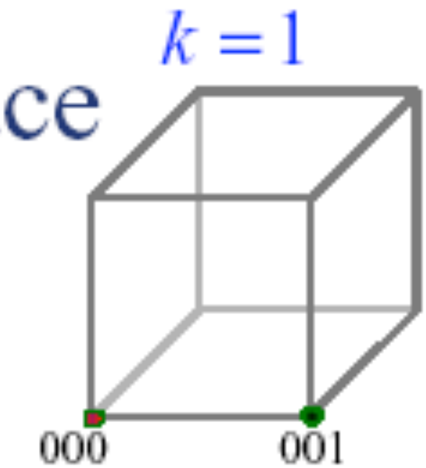
Vector Space (n=3) & Subspace

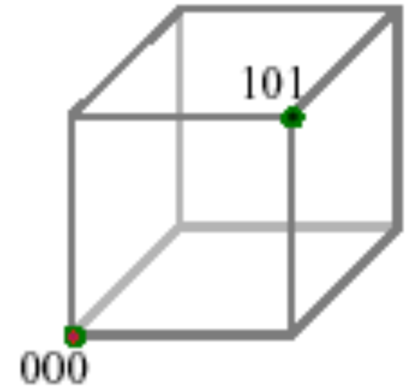
1

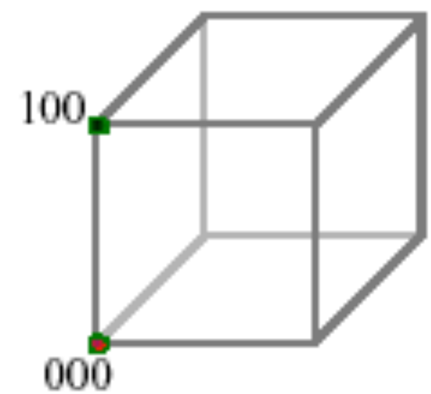
0











t=1, correct one error

(3,1,1)

n k d

w=3

d=2t+1, t=1,
2t+1=3=d

Classical Error Correction: Decoding

Idea: Measure the error and fix it

- Dual matrix P to G is called the *parity check matrix*:

transpose

$$P^T G = PG^T = 0,\ P \text{ has maximal rank } n-k$$

- P and G may be written in *standard form*:

identity

$$G = \left[I_k \mid -A^T \right] \Rightarrow P = \left[A \mid I_{n-k} \right]$$

- P *annihilates codewords* only!

- P returns the *error syndrome* $\Longrightarrow$

Role of Parity Check Matrix P

Explanation that P returns only error syndrome since it anihilates codewords v

- P returns the error syndrome

$$(v \oplus e)P = vP \oplus eP = 0 \oplus eP = eP$$

- A distance d code has $eP \neq 0$ for all errors e having $wt(e) < d$

- P^T is the generator and G^T is the parity check matrix of the dual code

  (The basis vectors of the dual code are those vectors which are ⊥ to each original codeword.)

Classical Linear Error Control Codes

- Linear block codes
  - CRC, Hamming, Reed-Muller codes
  - BCH, Reed-Solomon codes
  - algebraic-geometric code
- Linear convolutional codes
  - recursive/non-recursive
  - parallel/serial turbo codes

Linear Operator $\mathcal{L}$

$$\mathcal{L}\{a\cdot x_1 + b\cdot x_2\} = a\cdot\mathcal{L}\{x_1\} + b\cdot\mathcal{L}\{x_2\}$$

a, b : constants

x_1, x_2: inputs

(Example) linear: $\boldsymbol{y}=A\boldsymbol{x}+\boldsymbol{b}$

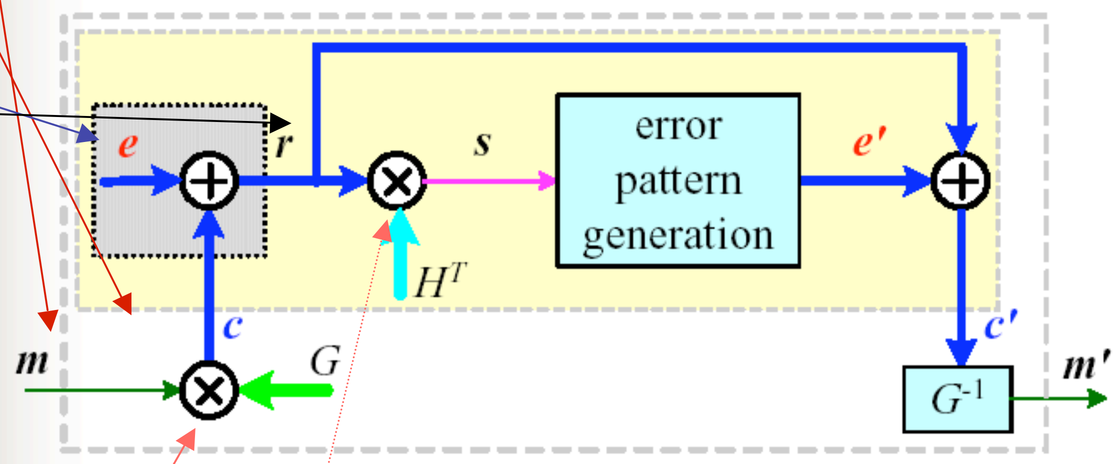
nonlinear: $z=\boldsymbol{x}^T A\boldsymbol{x}+\boldsymbol{b}^T\boldsymbol{x}+c$

General idea of block linear codes

(n,k) linear block codes

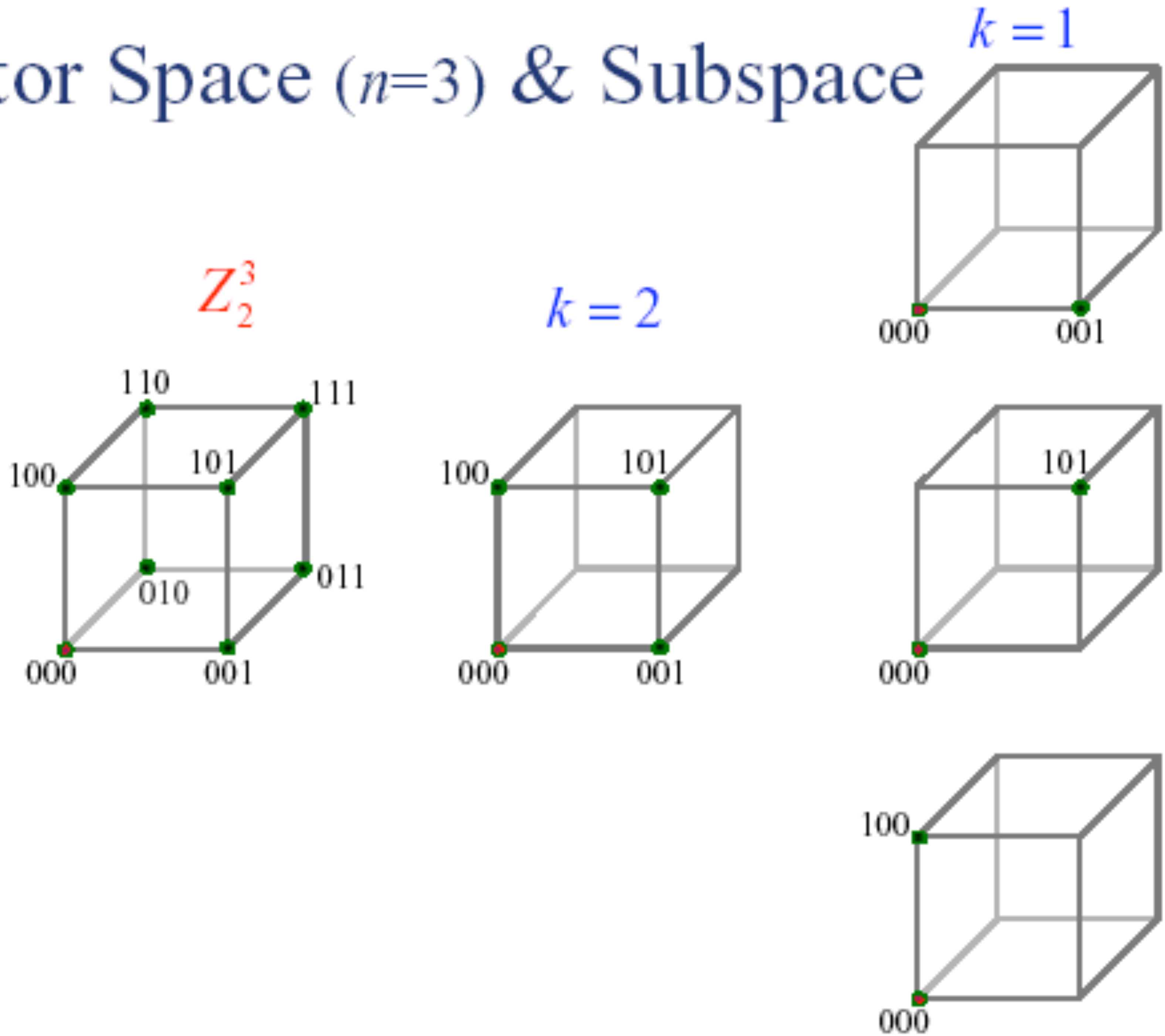
m	message vector	$1 \times k$
c	codeword $c=mG$	$1 \times n$
e	error vector	$1 \times n$
r	received vector $r=c+e$	$1 \times n$
G	generator matrix, $GH^T=0$	$k \times n$
H	parity check matrix	$(n-k) \times n$
s	syndrome $s=rH^T=eH^T$	$1 \times (n-k)$
e'	estimated error vector	$1 \times n$
c'	estimated codeword	$1 \times n$
m'	decoded message vector	$1 \times k$

Binary Linear Block Codes (LBC)



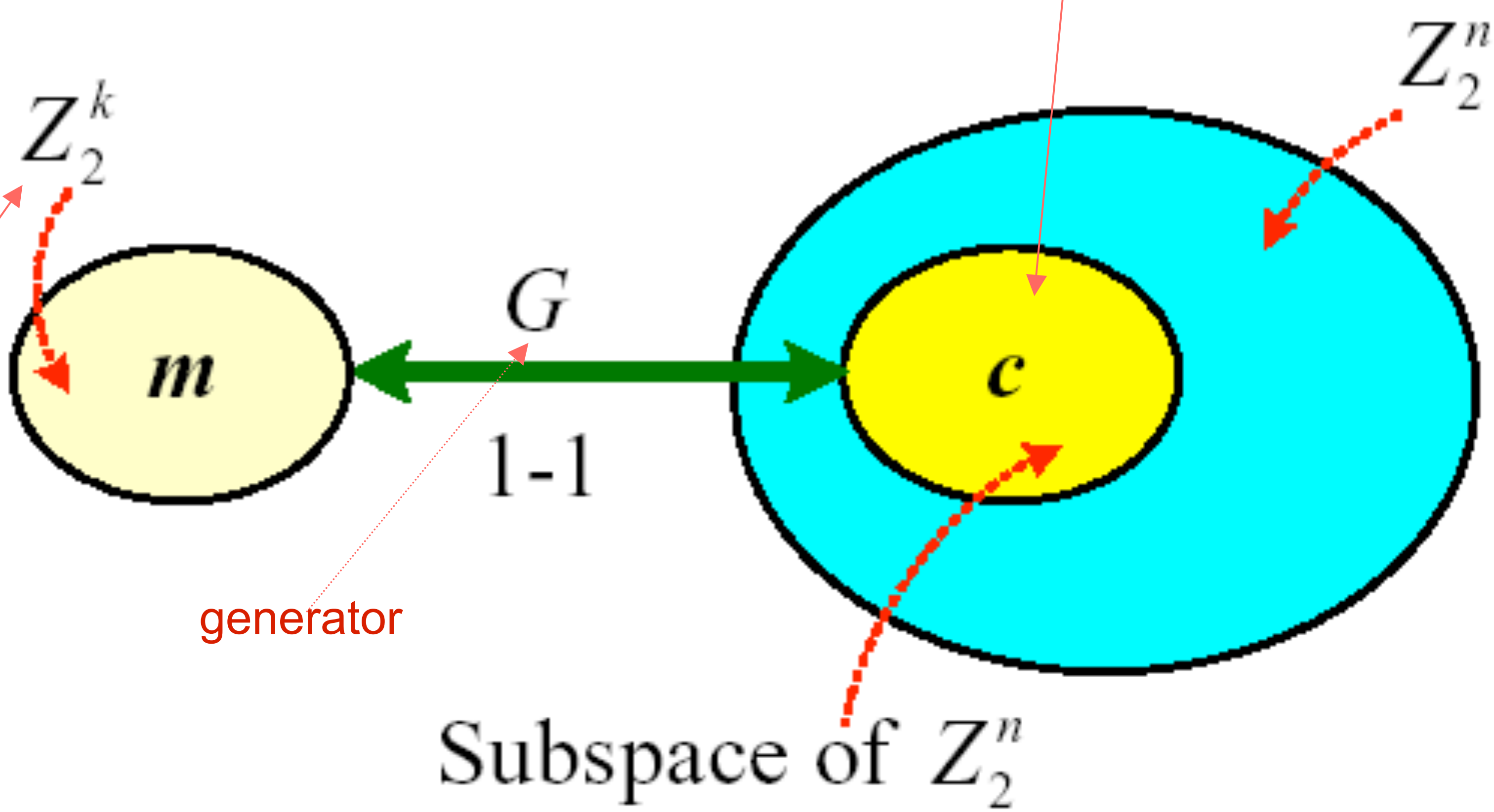
Matrix vector multiplication

Vector Space (n=3) & Subspace



Galois Field hypercube

An (n,k) linear block code is a subspace of Z_2^n.



Smaller space

We denote it by $\mathcal{C}$

n= length of vector

Features of Binary (n,k,d) LBC $\mathcal{C}$

Big space

Smaller space

distance

- $\mathcal{C}$ is a vector subspace of Z_2^n .

1. The addition (subtraction) of any two codewords remains a codeword.
2. The n-bit all-zero vector **0** is always a codeword for any linear block code.

Error Detection and Correction Capability

As in general case

- The weight of a codeword is defined as the number of nonzero elements in a codeword.
- The minimum distance d is equal to the minimum weight of nonzero codewords.

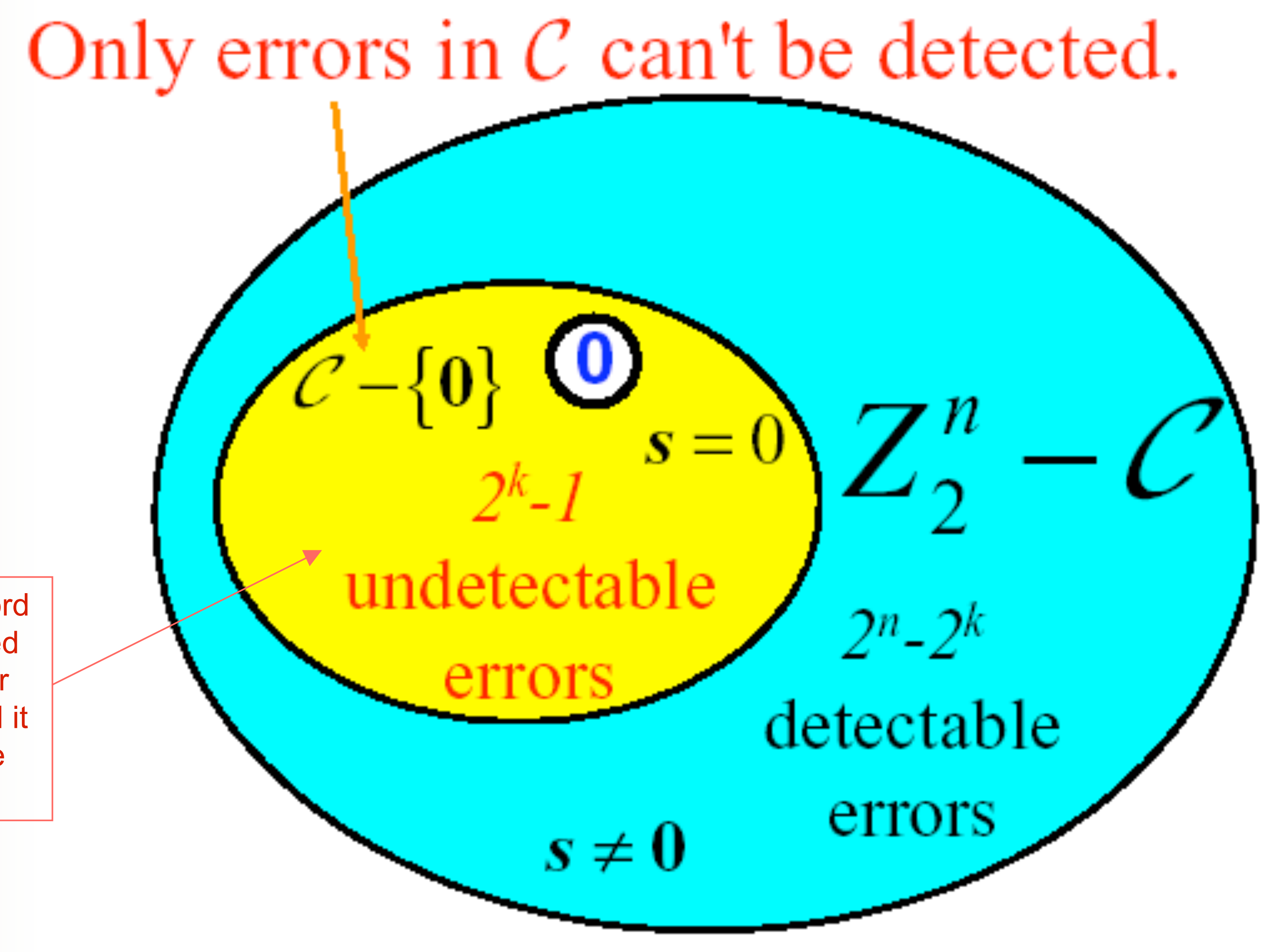
$$d = \min_{\substack{u,v \in \mathcal{C} \\ u \neq v}} dist(u,v) = \min_{\substack{u,v \in \mathcal{C} \\ u \neq v}} w(u \oplus v) = \min_{\substack{c \in \mathcal{C} \\ c \neq \mathbf{0}}} w(c)$$

3 in our case

- An (n,k,d) linear block code can detect any $(d\text{-}1)$-bit errors and correct any $\left\lfloor \frac{d-1}{2} \right\rfloor$-bit errors.

1 in our case

Detection Capability of Linear Block Codes



If codeword is changed to another codeword it cannot be detected

Detection & Correction of (n,k) Linear Block Codes

- Error detection is easier than error correction.
- #detectable errors = $2^n - 2^k$

$2^3-2^1=6$

0	
	1

- #correctable errors = #nonzero syndromes

$$= 2^{n-k}-1$$

$2^{3-1} = 4-1=3$

Linear (*n*,*k*) Cyclic Codes over GF(2)

- A computationally efficient subset of linear block codes. Good linear cyclic codes exist.
- Generator polynomial

$$g(x) = \sum_{i=0}^{n-k} g_i x^i \Leftrightarrow g = (g_{n-k}, \cdots, g_1, g_0)$$

- Parity check polynomial

$$h(x) = \sum_{i=0}^{k} h_i x^i \Leftrightarrow h = (h_k, \cdots, h_1, h_0)$$

- Constraint: $g(x)h(x) = x^n - 1$

Easy hardware to operate on these polynomials

Encoding a Cyclic Code

$$c(x) = m(x)g(x)$$

From slide with general diagram of linear codes

$$\Updownarrow$$

$$c = (c_{n-1}, \cdots, c_1, c_0)$$

$$c_j = m_j * g_j \triangleq \sum_{i=0}^{j} m_{j-i} g_i$$

Polynomial multiplication = Convolution of coefficients

Cyclic Shifts in Cyclic Codes

- Multiplication by x^j modulo $(x^n\text{-}1)$ is equivalent to a cyclic left shift by j positions.

$$c^{(j)}(x) = x^j \cdot c(x) \mod (x^n - 1)$$

$$\Updownarrow$$

$$c^{(j)} = (c_{n-j-1}, \cdots, c_1, c_0, c_{n-1}, \cdots, c_{n-j})$$

Cyclic property

- Any cyclic shifted version of a codeword is also a valid codeword.

$$\forall c = (c_{n-1}, c_{n-2}, c_{n-3}, \cdots, c_1, c_0) \triangleq c^{(0)} = c^{(n)} \in \mathcal{C},$$

$$c^{(1)} \triangleq (c_{n-2}, c_{n-3}, \cdots, c_1, c_0, c_{n-1}) \in \mathcal{C}$$

$$\Rightarrow c^{(2)} \triangleq (c_{n-3}, \cdots, c_1, c_0, c_{n-1}, c_{n-2}) \in \mathcal{C}$$

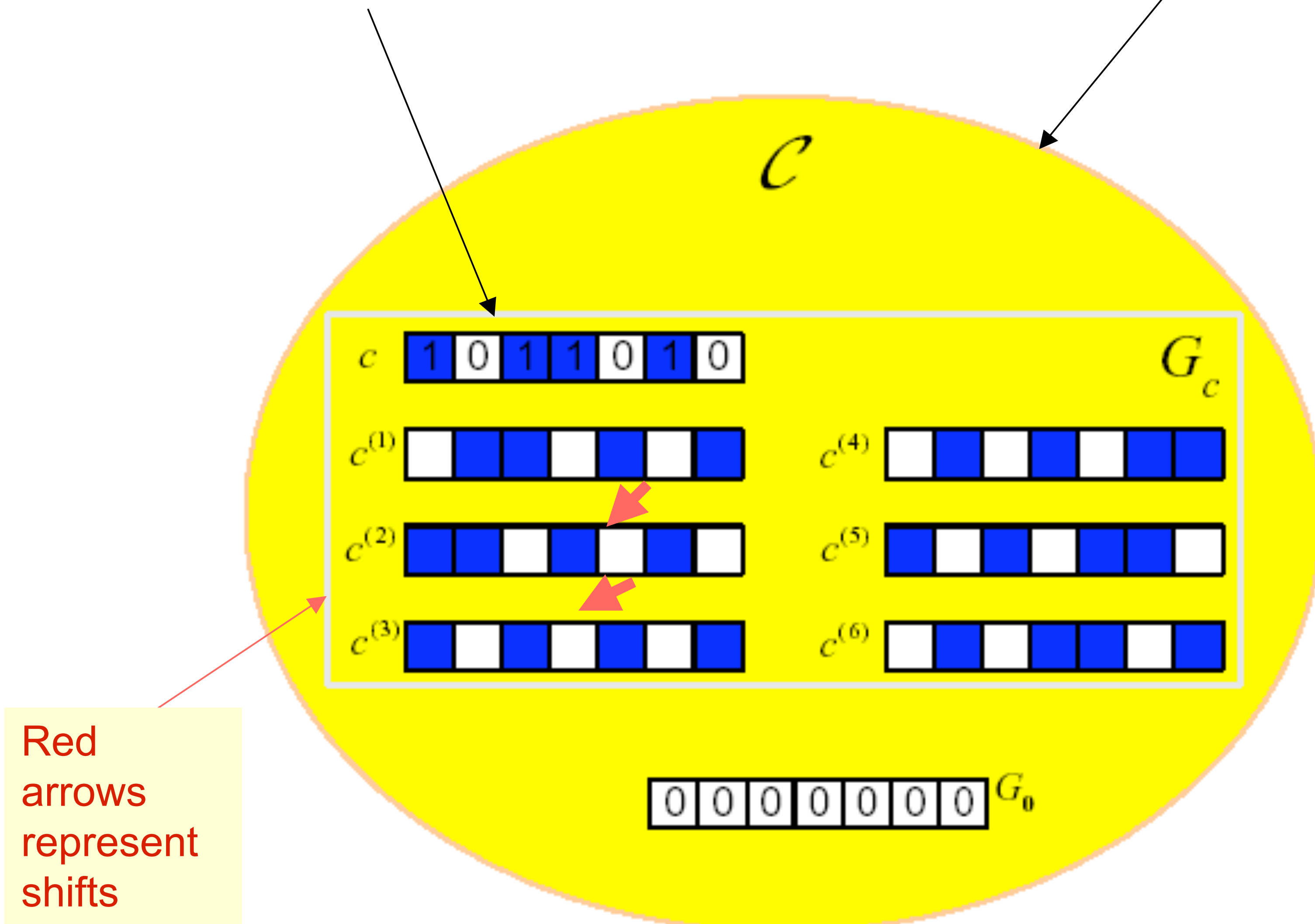
$$\Rightarrow c^{(j)} \in \mathcal{C}, \forall j$$

Thus we can talk about a group

- $\boxed{\text{Cyclic group}}$ of a codeword c

$$G_c = \{c^{(j)}, j = 0, 1, \cdots, n-1\}, \quad |G_c| \le n$$

Cyclic Group G_c in Code Subspace



Quantum Error Correction

Outline

- Sources and types of errors
- Differences between classical and quantum error correction
- Quantum error correcting codes

Introduction: why quantum error correction?

- Quantum states of superposition (which stores quantum information) extremely fragile.
- Quantum error correction more tricky than classical error correction.
- In the field of quantum computation, what is possible in theory is very far off from what can be implemented.
- Complex quantum computation impossible without the ability to recover from errors

What can go wrong?

- Internal:
  - Initial states on input qubits not prepared properly.
  - Quantum gates used may not be accurate
    - Quantum gates may introduce small errors which will accumulate.
- External:
  - Dissipation
    - A qubit loses energy to the environment.
  - Decoherence

Decoherence

- Decoherence is the loss of quantum information of a quantum system due to its interaction with the environment.
- Almost impossible to isolate a quantum system from the environment.
- Over time, our quantum system will be entangled with the environment.

Detrimental role of environment

- Information encoded in our quantum system will be encoded instead in the correlations between the quantum system and the environment.
- The environment can be seen as measuring the quantum system, collapsing its superposition state.
- Hence quantum information (encoded in the superposition) is irreversibly lost from the qubit.

How to Deal With Decoherence?

First method to deal with decoherence

- Design quantum algorithms to finish before decoherence ruins the quantum information.
  - Can be difficult as
    - Decoherence occurs very quickly.
    - Quantum algorithms may be very complex and long.

Dealing With Decoherence

Second method to deal with decoherence

- Try to lower the rate at which decoherence occurs.
  - Accomplished by using a right combination of:
    - Quantum particle type
    - Quantum computer size
    - Environment

Decoherence times in practice

- **Decoherence time** refers to the time available before decoherence ruins quantum information.
- Decoherence time is affected by the size of the system, as well as the environment.

Approximate decoherence time (in seconds) for various system sizes and environment

System size (cm)	Cosmic Radiation	Room Temperature	Sunlight	Vacuum (10^6 particles/cm^3)	Air
10^{-3}	10^{-7}	10^{-14}	10^{-16}	10^{-18}	10^{-35}
10^{-5}	10^{15}	10^{-3}	10^{-8}	10^{-10}	10^{-23}
10^{-6}	10^{25}	10^{5}	10^{-2}	10^{-6}	10^{-19}

  – Decoherence time affected by environmental factors like temperature and amount of surrounding particles in the environment

Gate completion time

- Time needed for a quantum gate operation is as important as decoherence time.

Max no of operations that can be performed before decoherence

$$= \frac{\text{decoherence time}}{\text{time per quantum gate operation}}$$

- Different types of quantum systems have different decoherence time and per gate operation time.

In next time we will compare these coefficients for real technologies

Maximum number of operations before decoherence for various quantum systems

Decoherence time versus time required for a quantum gate operation for various quantum systems

Quantum system	Decoherence time (sec)	Time per Gate Operation (sec)	Max number of operations before decoherence
Electrons from gold atom	10^{-8}	10^{-14}	10^{6}
Trapped indium atoms	10^{-1}	10^{-14}	10^{13}
Optical microcavity	10^{-5}	10^{-14}	10^{9}
Electron spin	10^{-3}	10^{-7}	10^{4}
Electron quantum dot	10^{-3}	10^{-6}	10^{3}
Nuclear spin	10^{4}	10^{-3}	10^{7}

- The better the decoherence time, the slower the quantum gate operations.

Dealing With Decoherence and other sources of errors

Third method to deal with decoherence

- Use Quantum Error correcting codes
  - Encode qubits (together with extra ancillary qubits) in a state where subsequent errors can be corrected.
  - Allows long algorithms requiring many operations to run, as errors can be corrected after they occur.

History of **Quantum Error Correction** Codes (QECC)

- Dark age (before 1995) (No-cloning theorem)
- Calderbank-Shor-Steane (CSS) code
  - Shor (1995) (unitary quantum operators)
  - Steane (1996)
- Stabilizer code
- Quantum block code
  - Reed-Muller, Reed-Solomon, algebraic-geometric, …
- Quantum convolutional code

Quantum Error Correcting Codes

- CSS (Calderbank-Shor-Steane) code
  - special case of stabilizer codes
- Stabilizer codes
  - codewords are eigenvectors with eigenvalue=1
- Quantum linear block codes
- Quantum linear convolutional codes

Quantum Errors

- $E: H_2^{\otimes n} \to H_2^{\otimes n}$ can be *any* (not necessarily injective) map

- Usual model simplification: *Tensor products* of single qubit errors:

Quantum Errors

$$\begin{aligned} E &= A_1 \otimes A_1 \otimes A_1 \otimes A_1 \\ V &= |0\rangle \otimes |1\rangle \otimes |1\rangle \otimes |0\rangle \end{aligned}$$

- Single qubit error operators have a *finite basis*:

$$|0\rangle \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad |1\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\begin{aligned} E &= \tfrac{1}{2}\left[\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \alpha_1 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \alpha_2 \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix} + \alpha_3 \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right] \\ &= \tfrac{1}{2}\left[I + \alpha_1 \sigma_x + \alpha_2 \sigma_y + \alpha_3 \sigma_z\right] \\ &= \tfrac{1}{2}\left[I + \boldsymbol{\alpha} \cdot \boldsymbol{\sigma}\right] \end{aligned}$$

General representation of single qubit

$$E = \tfrac{1}{2}\left[\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} + \alpha_1\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} + \alpha_2\begin{pmatrix}0 & i\\ -i & 0\end{pmatrix} + \alpha_3\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\right]$$

$$= \tfrac{1}{2}\left[I + \alpha_1\sigma_x + \alpha_2\sigma_y + \alpha_3\sigma_z\right]$$

$$= \tfrac{1}{2}\left[I + \boldsymbol{\alpha}\cdot\boldsymbol{\sigma}\right]$$

- Note that if we take $X=\sigma_x$, $Y=-i\sigma_y$, and $Z=\sigma_z$ that the error basis $\{I, X, Y, Z\}$ forms a representation of the *quaternion* group

∴ If we can correct σ_x, σ_y , σ_z we can correct any error! (more about this later)

Quantum Error Correction: Naive Encoding

First approach: Let's try classical codes

$$0 \mapsto 000$$
$$1 \mapsto 111$$

"Repetition" $[3,1,3]$ code

$$|\psi\rangle \mapsto |\psi\rangle \otimes |\psi\rangle \otimes |\psi\rangle \ ?$$

But wait! The quantum world is a

Cloning (copying) operator U does not exist

Why?

Assume that such U exists

$$U\left(|\alpha\rangle|0\rangle\right)=|\alpha\rangle|\alpha\rangle$$

$$\Rightarrow U\left(\left(|\alpha\rangle+|\beta\rangle\right)|0\rangle\right)=|\alpha\rangle|\alpha\rangle+|\beta\rangle|\beta\rangle$$

$$\neq\left(|\alpha\rangle+|\beta\rangle\right)\left(|\alpha\rangle+|\beta\rangle\right)$$

So we apply it to general superposed state

And we obtain this

Which is not what we wanted

But this is still useful. Although not copying , this is a redundancy introducing operator so it may be used for error correcting codes. This was one of main ideas

Commuting and Anti-Commuting Quantum Operators

Definitions : $[A,B] = AB - BA$

Commutator of A and B

$\{A,B\} = AB + BA$

Anti-commutator of A and B

A and B commute : $[A,B]=0$

A and B anti-commute: $\{A,B\}=0$

(1-qubit) Pauli Operators

Bit flip: $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Phase flip: $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

We express Y in terms of X and Z

Bit & phase flip:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = -i\ XZ$$

global phase

Properties of Pauli Operators

$$X^2 = Z^2 = Y^2 = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Adjoint operator

$$X = X^H = X^{-1}, \quad Y = Y^H = Y^{-1}, \quad Z = Z^H = Z^{-1}$$

$$XZ = -ZX = iY, \quad \{X, Z\} = 0$$

anticommutative

$$XY = -YX = iZ, \quad \{X, Y\} = 0$$

$$YZ = -ZY = iX, \quad \{Y, Z\} = 0$$

Pauli operators are self-inverses and anti-commute

1-qubit Pauli Group G_1

A (non-abelian) group of cardinality 8:

$$G_1 = \{\pm I, \pm X, \pm Y, \pm Z\}$$

4 * 2 = 8 elements in this group

Any two operators in G_1 either commute or anti-commute:

$$[A, B] = 0 \quad \text{or} \quad \{A, B\} = 0, \quad \forall A, B \in G_1$$

Please remember, this is important → **Pauli operators are a group**

Now we extend to group G_n

Group G_n: n-qubit Pauli group containing all the n-fold tensor products of one-bit Pauli operators.

$$\boxed{G_n = \pm\{I, X, Y, Z\}^{\otimes n}}$$

We model faults in channels by G_n

n-qubit Depolarizing Channels:
error operators $\in G_n$

Example: error operator in G_5

Tensor product

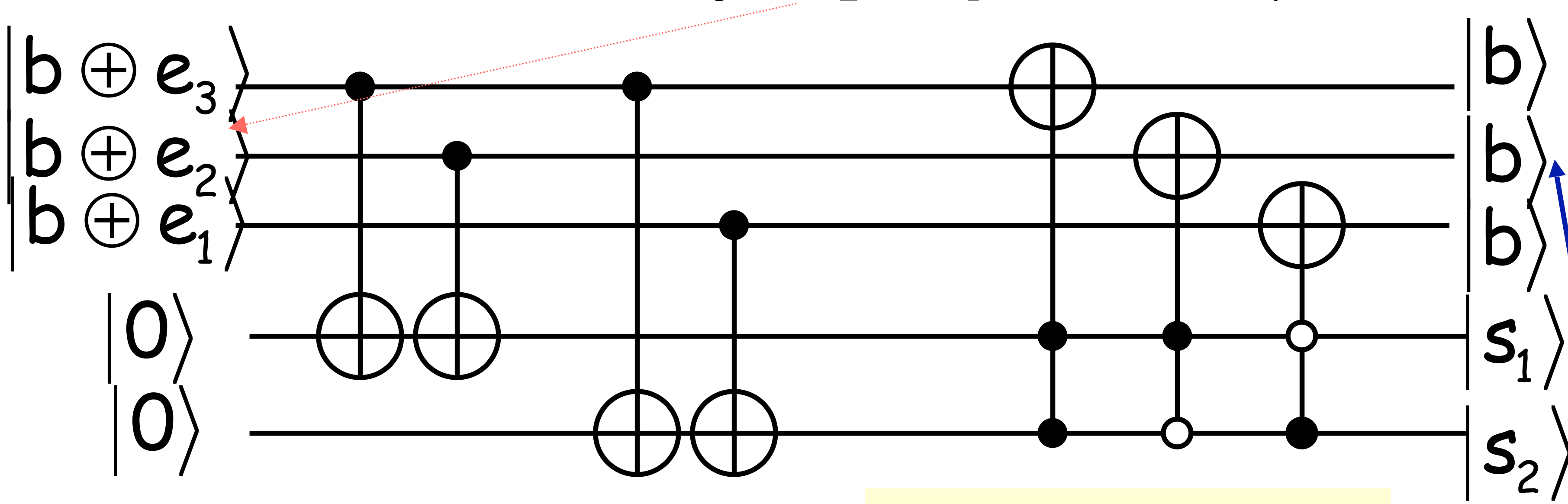
$$E = X \otimes I \otimes Z \otimes X \otimes Y = X_1 Z_3 X_4 Y_5$$

qubit	operation
1	Bit flip
2	No error
3	Phase flip
4	Bit flip
5	Bit and phase flip

This will be our error model from now

Quantum network for correcting errors

- Assume that $e_3 + e_2 + e_1 \leq 1 \qquad e_i \in \{0,1\}$



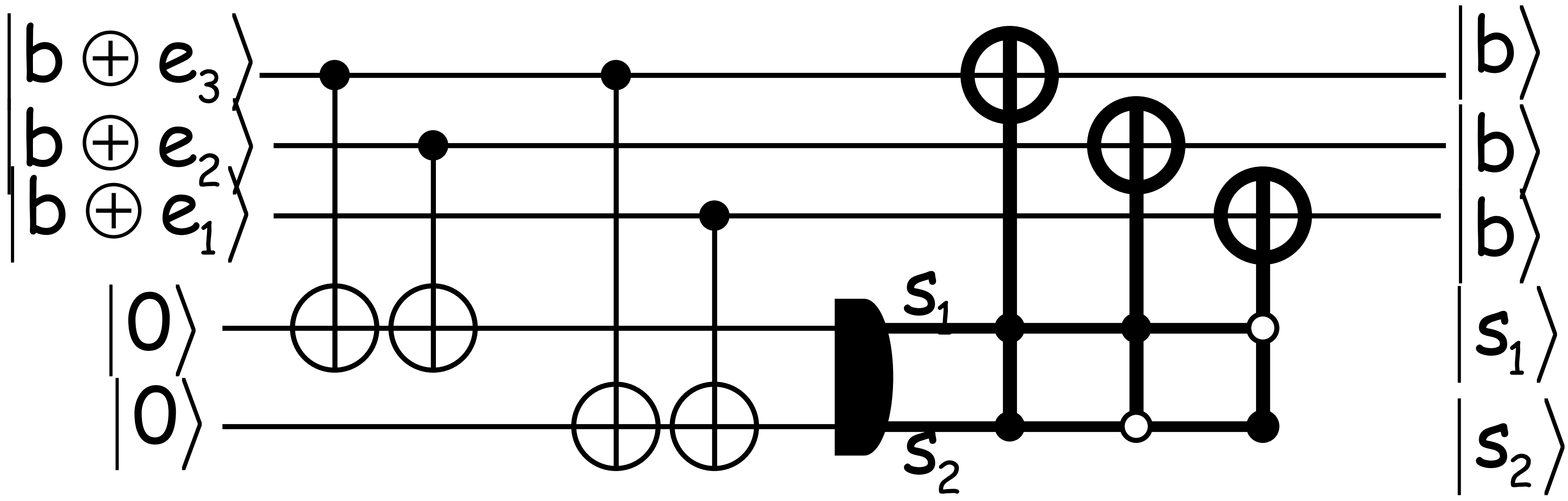
Input signal with error

$$\alpha|e_3\rangle|e_2\rangle|e_1\rangle + \beta|1 \oplus e_3\rangle|1 \oplus e_2\rangle|1 \oplus e_1\rangle \rightarrow$$

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

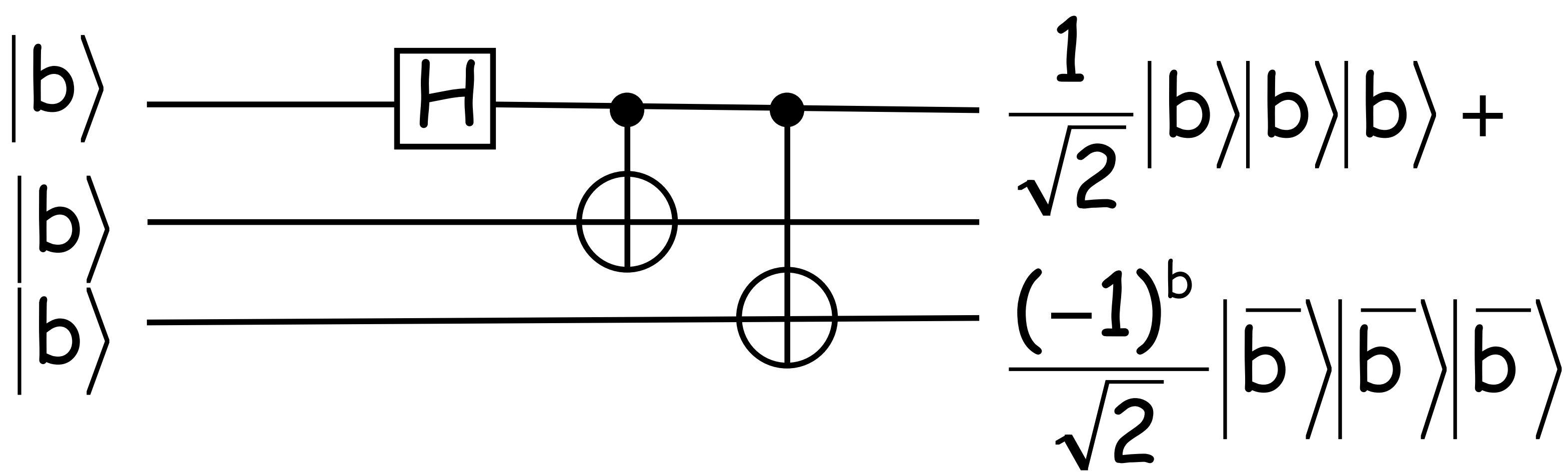
Input signal after error correcting

Equivalently



Perform operations on logical bits

- e.g. Hadamard gate



Quantum Error Correcting by Peter Shor

- In 1995, Peter Shor developed an improved procedure using 9 qubits to encode a single qubit of information
- His algorithm was a **majority vote type** of system that allowed **all single qubit errors to be *detected and corrected***

This was a starting point to great research area, although his paper had many bugs