Single Shot Fault Tolerance

I don't know much about it but I don't think it works

David Poulin

Équipe de Recherche sur la Physique de l'Information Quantique
Département de Physique
Université de Sherbrooke

Coogee Quantum Information Theory Workshop

Outline

Outline

An $[n, k, d]$ binary linear codes

- An $(n-k) \times n$ parity check matrix H with binary entries.
- Codewords are the n-bit strings x which live in the kernel of H:

$$\mathcal{C} = \left\{ x \in \{0,1\}^n : Hx = 0 \right\}$$

  (arithmetic is binary throughout).
- Assuming that the rows of H are linearly independent, there are 2^k strings in $\mathcal{C}$.
  - The code encodes k bits.
- The shortest non-zero string in $\mathcal{C}$ has weight d, and this defines the minimum distance of the code.
- A bit flip error on codeword x produces $y = x + e$.
- The syndrome of y is $s \equiv Hy = H(x+e) = He$.
- The syndrome $s = He$ gives us partial information about e and decoding consists in inferring e from this partial information.

An $[n, k, d]$ binary linear codes

- An $(n - k) \times n$ parity check matrix H with binary entries.
- Codewords are the n-bit strings x which live in the kernel of H:

$$\mathcal{C} = \left\{ x \in \{0,1\}^n : Hx = 0 \right\}$$

(arithmetic is binary throughout).

- Assuming that the rows of H are linearly independent, there are 2^k strings in $\mathcal{C}$.
    - The code encodes k bits.
- The shortest non-zero string in $\mathcal{C}$ has weight d, and this defines the minimum distance of the code.
- A bit flip error on codeword x produces $y = x + e$.
- The syndrome of y is $s \equiv Hy = H(x + e) = He$.
- The syndrome $s = He$ gives us partial information about e and decoding consists in inferring e from this partial information.

An $[n, k, d]$ binary linear codes

- An $(n-k) \times n$ parity check matrix H with binary entries.
- Codewords are the n-bit strings x which live in the kernel of H:

$$\mathcal{C} = \left\{ x \in \{0,1\}^n : Hx = 0 \right\}$$

  (arithmetic is binary throughout).
- Assuming that the rows of H are linearly independent, there are 2^k strings in $\mathcal{C}$.
  - The code encodes k bits.
- The shortest non-zero string in $\mathcal{C}$ has weight d, and this defines the minimum distance of the code.
- A bit flip error on codeword x produces $y = x + e$.
- The syndrome of y is $s \equiv Hy = H(x+e) = He$.
- The syndrome $s = He$ gives us partial information about e and decoding consists in inferring e from this partial information.

An $[n, k, d]$ binary linear codes

- An $(n-k) \times n$ parity check matrix H with binary entries.
- Codewords are the n-bit strings x which live in the kernel of H:

$$\mathcal{C} = \left\{ x \in \{0,1\}^n : Hx = 0 \right\}$$

(arithmetic is binary throughout).
- Assuming that the rows of H are linearly independent, there are 2^k strings in $\mathcal{C}$.
  - The code encodes k bits.
- The shortest non-zero string in $\mathcal{C}$ has weight d, and this defines the minimum distance of the code.
- A bit flip error on codeword x produces $y = x + e$.
- The syndrome of y is $s \equiv Hy = H(x+e) = He$.
- The syndrome $s = He$ gives us partial information about e and decoding consists in inferring e from this partial information.

An $[n, k, d]$ binary linear codes

- An $(n-k) \times n$ parity check matrix H with binary entries.
- Codewords are the n-bit strings x which live in the kernel of H:

$$\mathcal{C} = \left\{x \in \{0,1\}^n : Hx = 0\right\}$$

  (arithmetic is binary throughout).
- Assuming that the rows of H are linearly independent, there are 2^k strings in $\mathcal{C}$.
  - The code encodes k bits.
- The shortest non-zero string in $\mathcal{C}$ has weight d, and this defines the minimum distance of the code.
- A bit flip error on codeword x produces $y = x + e$.
- The syndrome of y is $s \equiv Hy = H(x+e) = He$.
- The syndrome $s = He$ gives us partial information about e and decoding consists in inferring e from this partial information.

An $[n, k, d]$ binary linear codes

- An $(n-k) \times n$ parity check matrix H with binary entries.
- Codewords are the n-bit strings x which live in the kernel of H:

$$\mathcal{C} = \left\{ x \in \{0,1\}^n : Hx = 0 \right\}$$

  (arithmetic is binary throughout).
- Assuming that the rows of H are linearly independent, there are 2^k strings in $\mathcal{C}$.
  - The code encodes k bits.
- The shortest non-zero string in $\mathcal{C}$ has weight d, and this defines the minimum distance of the code.
- A bit flip error on codeword x produces $y = x + e$.
- The syndrome of y is $s \equiv Hy = H(x+e) = He$.
- The syndrome $s = He$ gives us partial information about e and decoding consists in inferring e from this partial information.

An $[n, k, d]$ binary linear codes

- An $(n-k) \times n$ parity check matrix H with binary entries.
- Codewords are the n-bit strings x which live in the kernel of H:

$$\mathcal{C} = \left\{ x \in \{0,1\}^n : Hx = 0 \right\}$$

  (arithmetic is binary throughout).
- Assuming that the rows of H are linearly independent, there are 2^k strings in $\mathcal{C}$.
  - The code encodes k bits.
- The shortest non-zero string in $\mathcal{C}$ has weight d, and this defines the minimum distance of the code.
- A bit flip error on codeword x produces $y = x + e$.
- The syndrome of y is $s \equiv Hy = H(x+e) = He$.
- The syndrome $s = He$ gives us partial information about e and decoding consists in inferring e from this partial information.

An $[n, k, d]$ binary linear codes

- An $(n-k) \times n$ parity check matrix H with binary entries.
- Codewords are the n-bit strings x which live in the kernel of H:

$$\mathcal{C} = \left\{ x \in \{0,1\}^n : Hx = 0 \right\}$$

  (arithmetic is binary throughout).
- Assuming that the rows of H are linearly independent, there are 2^k strings in $\mathcal{C}$.
  - The code encodes k bits.
- The shortest non-zero string in $\mathcal{C}$ has weight d, and this defines the minimum distance of the code.
- A bit flip error on codeword x produces $y = x + e$.
- The syndrome of y is $s \equiv Hy = H(x+e) = He$.
- The syndrome $s = He$ gives us partial information about e and decoding consists in inferring e from this partial information.

Tanner graphs

Repetition code $[7, 1, 7]$

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Check nodes
Bit nodes

- Is 6×7 so $k = 1$
- $C = \{0000000, 1111111\}$.
- We see that $d = 7$.

Tanner graphs

Repetition code $[7, 1, 7]$

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Check nodes
Bit nodes

- Is 6×7 so $k = 1$
- $\mathcal{C} = \{0000000, 1111111\}$.
- We see that $d = 7$.

Tanner graphs

Repetition code $[7, 1, 7]$

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Check nodes
Bit nodes

- Is 6×7 so $k = 1$
- $\mathcal{C} = \{0000000, 1111111\}$.
- We see that $d = 7$.

Tanner graphs

Repetition code $[7,1,7]$

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Check nodes
Bit nodes

- Is 6×7 so $k = 1$
- $\mathcal{C} = \{0000000, 1111111\}$.
- We see that $d = 7$.

Tanner graphs

Repetition code $[7, 1, 7]$

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

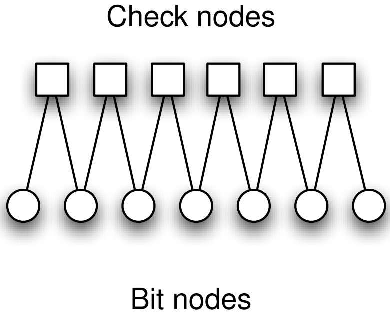
Check nodes
Bit nodes

- Is 6×7 so $k = 1$
- $\mathcal{C} = \{0000000, 1111111\}$.
- We see that $d = 7$.

Tanner graphs

Repetition code $[7, 1, 7]$

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$



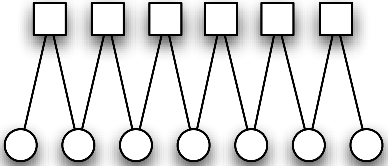
- Is 6×7 so $k = 1$
- $\mathcal{C} = \{0000000, 1111111\}$.
- We see that $d = 7$.

Tanner graphs

Repetition code

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$H \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

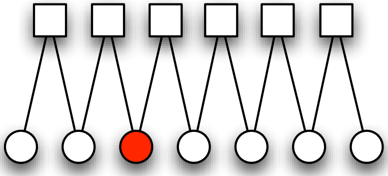


Tanner graphs

Repetition code

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$H \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

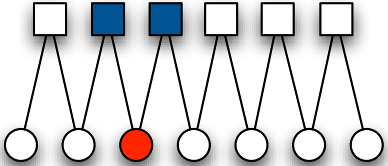


Tanner graphs

Repetition code

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

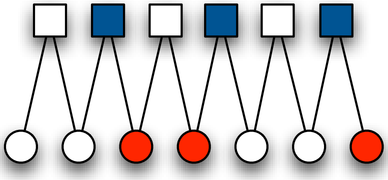
$$H \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$



Tanner graphs

Repetition code

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$



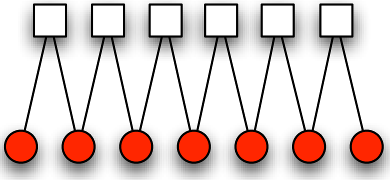
$$H \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Tanner graphs

Repetition code

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$H \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$



Tanner graphs

Hamming code $[7, 4, 3]$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- Is 3×7 so $k = 4$
- $C = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We see that $d = 3$.

Tanner graphs

Hamming code $[7, 4, 3]$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- Is 3×7 so $k = 4$
- $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We see that $d = 3$.

Tanner graphs

Hamming code $[7, 4, 3]$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- Is 3×7 so $k = 4$
- $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We see that $d = 3$.

Tanner graphs

Hamming code $[7, 4, 3]$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- Is 3×7 so $k = 4$
- $\mathcal{C} =$ $\mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We see that $d = 3$.

Tanner graphs

Hamming code $[7, 4, 3]$

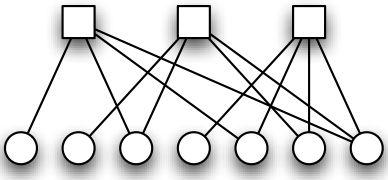
$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

- Is 3×7 so $k = 4$
- $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We see that $d = 3$.

Tanner graphs

Hamming code $[7,4,3]$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

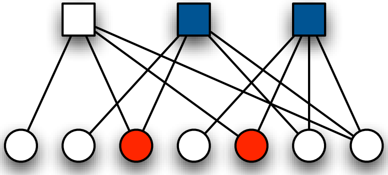


- Is 3×7 so $k = 4$
- $\mathcal{C} = \text{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We see that $d = 3$.

Tanner graphs

Hamming code $[7, 4, 3]$

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$



- Is 3×7 so $k = 4$
- $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We see that $d = 3$.

Generating matrix

- For the Hamming code, we had $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We can arrange these into the columns of a generating matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The code $\mathcal{C}$ is the image of G:
  - For any $z \in \{0,1\}^k$, Gz is a codeword.
  - $HG = 0$.

Generating matrix

- For the Hamming code, we had $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We can arrange these into the columns of a generating matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The code $\mathcal{C}$ is the image of G:
  - For any $z \in \{0,1\}^k$, Gz is a codeword.
  - $HG = 0$.

Generating matrix

- For the Hamming code, we had $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We can arrange these into the columns of a generating matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The code $\mathcal{C}$ is the image of G:
  - For any $z \in \{0,1\}^k$, Gz is a codeword.
  - $HG = 0$.

Generating matrix

- For the Hamming code, we had $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We can arrange these into the columns of a generating matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The code $\mathcal{C}$ is the image of G:
  - For any $z \in \{0, 1\}^k$, Gz is a codeword.
  - $HG = 0$.

Generating matrix

- For the Hamming code, we had $\mathcal{C} = \mathrm{span}\{1110000, 1001100, 0101010, 1101001\}$.
- We can arrange these into the columns of a generating matrix

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The code $\mathcal{C}$ is the image of G:
  - For any $z \in \{0,1\}^k$, Gz is a codeword.
  - $HG = 0$.

Sparse codes

- A sparse code, or low density parity check (LDPC) code is one whose parity check matrix is row and column sparse.
  - There are no more than λ_r nonzero entries per row.
  - There are no more than λ_c nonzero entries per colmn.
- It's easy to recognize an LDPC code from its Tanner graph.
- In a quantum setting, where extracting a syndrome implies actually measuring the corresponding checks, low weight is a blessing!

Sparse codes

- A sparse code, or low density parity check (LDPC) code is one whose parity check matrix is row and column sparse.
  - There are no more than λ_r nonzero entries per row.
  - There are no more than λ_c nonzero entries per colmn.
- It's easy to recognize an LDPC code from its Tanner graph.
- In a quantum setting, where extracting a syndrome implies actually measuring the corresponding checks, low weight is a blessing!

Sparse codes

- A sparse code, or low density parity check (LDPC) code is one whose parity check matrix is row and column sparse.
  - There are no more than λ_r nonzero entries per row.
  - There are no more than λ_c nonzero entries per colmn.
- It's easy to recognize an LDPC code from its Tanner graph.
- In a quantum setting, where extracting a syndrome implies actually measuring the corresponding checks, low weight is a blessing!

Sparse codes

- A sparse code, or low density parity check (LDPC) code is one whose parity check matrix is row and column sparse.
  - There are no more than λ_r nonzero entries per row.
  - There are no more than λ_c nonzero entries per colmn.
- It's easy to recognize an LDPC code from its Tanner graph.
- In a quantum setting, where extracting a syndrome implies actually measuring the corresponding checks, low weight is a blessing!

Sparse codes

- A sparse code, or low density parity check (LDPC) code is one whose parity check matrix is row and column sparse.
  - There are no more than λ_r nonzero entries per row.
  - There are no more than λ_c nonzero entries per colmn.
- It's easy to recognize an LDPC code from its Tanner graph.
- In a quantum setting, where extracting a syndrome implies actually measuring the corresponding checks, low weight is a blessing!

Random codes

- Define the binary entropy $h(p) = -p \log p - (1-p)\log(1-p)$. (Logarithm is base 2)
- Pick the $(n-k) \times k$ binary entries of H at random.
- With very high probability
  - The $n-k$ rows of H are linearly independent, so the code encodes k bits.
  - The code can 'correct' typical bit-flip errors provided the bit-flip probability p obeys $\frac{k}{n} < 1 - h(p)$.
- By 'correct', we mean that typical errors all have different syndromes, so they can in principle be uniquely identified from their syndrome.
- It is not known how to do this decoding efficiently for random codes.

Random codes

- Define the binary entropy $h(p) = -p \log p - (1-p) \log(1-p)$. (Logarithm is base 2)
- Pick the $(n-k) \times k$ binary entries of H at random.
- With very high probability
  - The $n-k$ rows of H are linearly independent, so the code encodes k bits.
  - The code can 'correct' typical bit-flip errors provided the bit-flip probability p obeys $\frac{k}{n} < 1 - h(p)$.
- By 'correct', we mean that typical errors all have different syndromes, so they can in principle be uniquely identified from their syndrome.
- It is not known how to do this decoding efficiently for random codes.

Random codes

- Define the binary entropy $h(p) = -p \log p - (1-p) \log(1-p)$. (Logarithm is base 2)
- Pick the $(n-k) \times k$ binary entries of H at random.
- With very high probability
  - The $n-k$ rows of H are linearly independent, so the code encodes k bits.
  - The code can 'correct' typical bit-flip errors provided the bit-flip probability p obeys $\frac{k}{n} < 1 - h(p)$.
- By 'correct', we mean that typical errors all have different syndromes, so they can in principle be uniquely identified from their syndrome.
- It is not known how to do this decoding efficiently for random codes.

Random codes

- Define the binary entropy $h(p) = -p \log p - (1-p)\log(1-p)$. (Logarithm is base 2)
- Pick the $(n-k) \times k$ binary entries of H at random.
- With very high probability
  - The $n-k$ rows of H are linearly independent, so the code encodes k bits.
  - The code can 'correct' typical bit-flip errors provided the bit-flip probability p obeys $\frac{k}{n} < 1 - h(p)$.
- By 'correct', we mean that typical errors all have different syndromes, so they can in principle be uniquely identified from their syndrome.
- It is not known how to do this decoding efficiently for random codes.

Random codes

- Define the binary entropy $h(p) = -p \log p - (1-p)\log(1-p)$. (Logarithm is base 2)
- Pick the $(n-k) \times k$ binary entries of H at random.
- With very high probability
  - The $n-k$ rows of H are linearly independent, so the code encodes k bits.
  - The code can 'correct' typical bit-flip errors provided the bit-flip probability p obeys $\frac{k}{n} < 1 - h(p)$.
- By 'correct', we mean that typical errors all have different syndromes, so they can in principle be uniquely identified from their syndrome.
- It is not known how to do this decoding efficiently for random codes.

Random codes

- Define the binary entropy $h(p) = -p \log p - (1-p)\log(1-p)$. (Logarithm is base 2)
- Pick the $(n-k) \times k$ binary entries of H at random.
- With very high probability
  - The $n-k$ rows of H are linearly independent, so the code encodes k bits.
  - The code can 'correct' typical bit-flip errors provided the bit-flip probability p obeys $\frac{k}{n} < 1 - h(p)$.
- By 'correct', we mean that typical errors all have different syndromes, so they can in principle be uniquely identified from their syndrome.
- It is not known how to do this decoding efficiently for random codes.

Random codes

- Define the binary entropy $h(p) = -p \log p - (1-p) \log(1-p)$. (Logarithm is base 2)
- Pick the $(n-k) \times k$ binary entries of H at random.
- With very high probability
  - The $n-k$ rows of H are linearly independent, so the code encodes k bits.
  - The code can 'correct' typical bit-flip errors provided the bit-flip probability p obeys $\frac{k}{n} < 1 - h(p)$.
- By 'correct', we mean that typical errors all have different syndromes, so they can in principle be uniquely identified from their syndrome.
- It is not known how to do this decoding efficiently for random codes.

Outline

Syndrome errors

- For the purpose of this talk, fault tolerance will refer to something very simple:

  The syndrome bits are not reliable.

  - Bits/qubits are subjected to errors with error-rate p.
  - The syndrome s is measured on the corrupted string.
  - The syndrome bits are subjected to a bit-flip noise with rate q.
  - We will consider the 'symmetric' case $p = q$ for simplicity.
- This is ofter referred to as the 'phenomenological noise model'.
- A more detailed noise model would consider how the syndrome bits are measured and how errors in that measuring circuit propagate.

Syndrome errors

- For the purpose of this talk, fault tolerance will refer to something very simple:

  The syndrome bits are not reliable.

  - Bits/qubits are subjected to errors with error-rate p.
  - The syndrome s is measured on the corrupted string.
  - The syndrome bits are subjected to a bit-flip noise with rate q.
  - We will consider the 'symmetric' case $p = q$ for simplicity.
- This is ofter referred to as the 'phenomenological noise model'.
- A more detailed noise model would consider how the syndrome bits are measured and how errors in that measuring circuit propagate.

Syndrome errors

- For the purpose of this talk, fault tolerance will refer to something very simple:

  The syndrome bits are not reliable.

  - Bits/qubits are subjected to errors with error-rate p.
  - The syndrome s is measured on the corrupted string.
  - The syndrome bits are subjected to a bit-flip noise with rate q.
  - We will consider the 'symmetric' case $p = q$ for simplicity.
- This is ofter referred to as the 'phenomenological noise model'.
- A more detailed noise model would consider how the syndrome bits are measured and how errors in that measuring circuit propagate.

Syndrome errors

- For the purpose of this talk, fault tolerance will refer to something very simple:

  The syndrome bits are not reliable.

  - Bits/qubits are subjected to errors with error-rate p.
  - The syndrome s is measured on the corrupted string.
  - The syndrome bits are subjected to a bit-flip noise with rate q.
  - We will consider the 'symmetric' case $p = q$ for simplicity.
- This is ofter referred to as the 'phenomenological noise model'.
- A more detailed noise model would consider how the syndrome bits are measured and how errors in that measuring circuit propagate.

Syndrome errors

- For the purpose of this talk, fault tolerance will refer to something very simple:

  The syndrome bits are not reliable.

  - Bits/qubits are subjected to errors with error-rate p.
  - The syndrome s is measured on the corrupted string.
  - The syndrome bits are subjected to a bit-flip noise with rate q.
  - We will consider the 'symmetric' case $p = q$ for simplicity.
- This is ofter referred to as the 'phenomenological noise model'.
- A more detailed noise model would consider how the syndrome bits are measured and how errors in that measuring circuit propagate.

Syndrome errors

- For the purpose of this talk, fault tolerance will refer to something very simple:

  The syndrome bits are not reliable.

  - Bits/qubits are subjected to errors with error-rate p.
  - The syndrome s is measured on the corrupted string.
  - The syndrome bits are subjected to a bit-flip noise with rate q.
  - We will consider the 'symmetric' case $p = q$ for simplicity.
- This is ofter referred to as the 'phenomenological noise model'.
- A more detailed noise model would consider how the syndrome bits are measured and how errors in that measuring circuit propagate.

Syndrome errors

- For the purpose of this talk, fault tolerance will refer to something very simple:

  The syndrome bits are not reliable.

  - Bits/qubits are subjected to errors with error-rate p.
  - The syndrome s is measured on the corrupted string.
  - The syndrome bits are subjected to a bit-flip noise with rate q.
  - We will consider the 'symmetric' case $p = q$ for simplicity.
- This is ofter referred to as the 'phenomenological noise model'.
- A more detailed noise model would consider how the syndrome bits are measured and how errors in that measuring circuit propagate.

Repeated syndrome measurements

- Repeated syndrome measurements is a good way to cope with syndrome errors.
- We can't just naively take the majority vote among the syndrome bits since additional errors can occur in between syndrome measurements, so they are not meant to agree even in the absence of syndrome errors.
- For the toric code, this leads to a picture of flux tubes in a 3D bulk with endpoints corresponding to a change of syndrome between consecutive measurements.

Repeated syndrome measurements

- Repeated syndrome measurements is a good way to cope with syndrome errors.
- We can't just naively take the majority vote among the syndrome bits since additional errors can occur in between syndrome measurements, so they are not meant to agree even in the absence of syndrome errors.
- For the toric code, this leads to a picture of flux tubes in a 3D bulk with endpoints corresponding to a change of syndrome between consecutive measurements.

Repeated syndrome measurements

- Repeated syndrome measurements is a good way to cope with syndrome errors.
- We can't just naively take the majority vote among the syndrome bits since additional errors can occur in between syndrome measurements, so they are not meant to agree even in the absence of syndrome errors.
- For the toric code, this leads to a picture of flux tubes in a 3D bulk with endpoints corresponding to a change of syndrome between consecutive measurements.

Outline

Is it possible to do fault-tolerant quantum error correction without repeating the syndrome measurements?

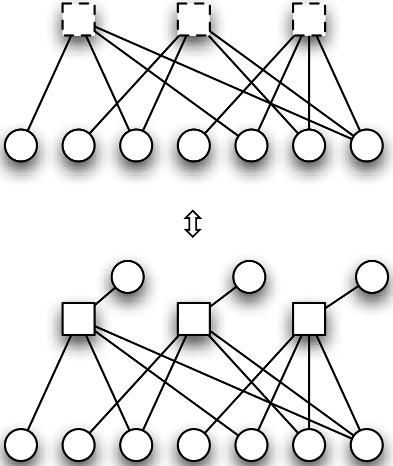
Outline

Extra Bits

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H' = \left(\begin{array}{ccc|ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

- In general, $H \rightarrow (I|H)$.
- An $[n, k, d]$ code $\rightarrow$ An $[2n-k, n, d' \leq d]$ code.

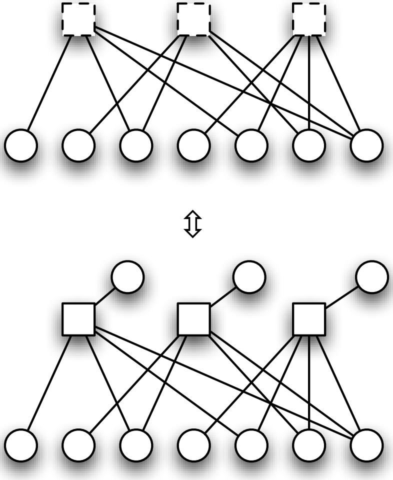


Extra Bits

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H' = \left(\begin{array}{ccc|ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

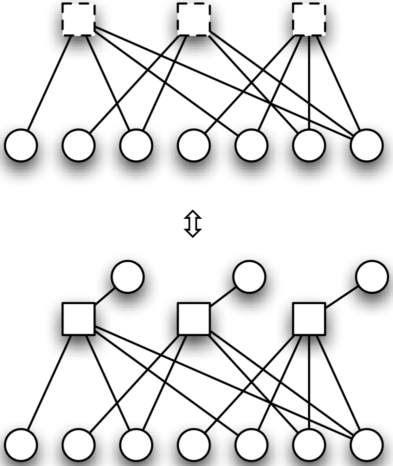
- In general, $H \rightarrow (I|H)$.
- An $[n, k, d]$ code $\rightarrow$
  An $[2n - k, n, d' \leq d]$ code.



Extra Bits

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H' = \left(\begin{array}{ccc|ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

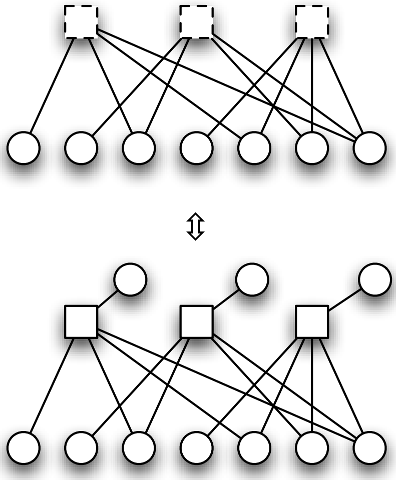


- In general, $H \rightarrow (I|H)$.
- An $[n, k, d]$ code $\rightarrow$
  An $[2n - k, n, d' \leq d]$ code.

Extra Bits

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H' = \left(\begin{array}{ccc|ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

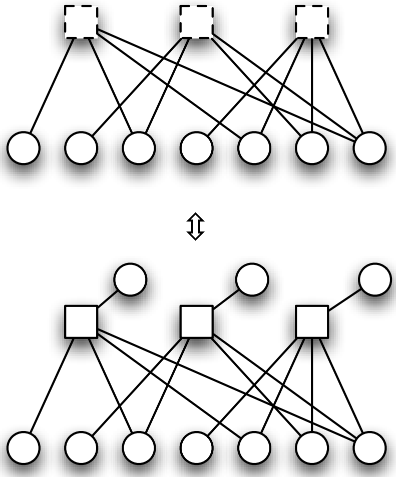


- In general, $H \to (I|H)$.
- An $[n, k, d]$ code $\to$
  An $[2n - k, n, d' \leq d]$ code.

Extra Bits

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H' = \left(\begin{array}{ccc|ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$



- In general, $H \to (I|H)$.
- An $[n, k, d]$ code $\to$
  An $[2n - k, n, d' \leq d]$ code.

Normal form

- Every parity check matrix can be put in the form $H = (I|A)$ for some matrix A:
  - Since $\mathcal{C}$ is the kernel of H, it does not change when we do row manipulations on H.
  - By row manipulations (Gaussian elimination) we can put any matrix in normal form $H = (I|A)$.
- I can think of $H' = (I|H)$ as a parity check matrix in normal form.
- Suppose I start with a random $(2n-k) \times n$ binary matrix and put it in normal form to get $H' = (I|H)$.
- The matrix H will also be randomly distributed.
- But H' being a random code can correct typical errors provided $\frac{k'}{n'} = \frac{n}{2n-k} < 1 - h(p)$, or equivalently

$$R = \frac{k}{n} < \frac{1-2h(p)}{1-h(p)}$$

Normal form

- Every parity check matrix can be put in the form $H = (I|A)$ for some matrix A:
  - Since $\mathcal{C}$ is the kernel of H, it does not change when we do row manipulations on H.
  - By row manipulations (Gaussian elimination) we can put any matrix in normal form $H = (I|A)$.
- I can think of $H' = (I|H)$ as a parity check matrix in normal form.
- Suppose I start with a random $(2n-k) \times n$ binary matrix and put it in normal form to get $H' = (I|H)$.
- The matrix H will also be randomly distributed.
- But H' being a random code can correct typical errors provided $\frac{k'}{n'} = \frac{n}{2n-k} < 1 - h(p)$, or equivalently

$$R = \frac{k}{n} < \frac{1-2h(p)}{1-h(p)}$$

Normal form

- Every parity check matrix can be put in the form $H = (I|A)$ for some matrix A:
  - Since $\mathcal{C}$ is the kernel of H, it does not change when we do row manipulations on H.
  - By row manipulations (Gaussian elimination) we can put any matrix in normal form $H = (I|A)$.
- I can think of $H' = (I|H)$ as a parity check matrix in normal form.
- Suppose I start with a random $(2n-k) \times n$ binary matrix and put it in normal form to get $H' = (I|H)$.
- The matrix H will also be randomly distributed.
- But H' being a random code can correct typical errors provided $\frac{k'}{n'} = \frac{n}{2n-k} < 1 - h(p)$, or equivalently

$$R = \frac{k}{n} < \frac{1-2h(p)}{1-h(p)}$$

Normal form

- Every parity check matrix can be put in the form $H = (I|A)$ for some matrix A:
  - Since $\mathcal{C}$ is the kernel of H, it does not change when we do row manipulations on H.
  - By row manipulations (Gaussian elimination) we can put any matrix in normal form $H = (I|A)$.
- I can think of $H' = (I|H)$ as a parity check matrix in normal form.
- Suppose I start with a random $(2n-k) \times n$ binary matrix and put it in normal form to get $H' = (I|H)$.
- The matrix H will also be randomly distributed.
- But H' being a random code can correct typical errors provided $\frac{k'}{n'} = \frac{n}{2n-k} < 1 - h(p)$, or equivalently

$$R = \frac{k}{n} < \frac{1-2h(p)}{1-h(p)}$$

Normal form

- Every parity check matrix can be put in the form $H = (I|A)$ for some matrix A:
  - Since $\mathcal{C}$ is the kernel of H, it does not change when we do row manipulations on H.
  - By row manipulations (Gaussian elimination) we can put any matrix in normal form $H = (I|A)$.
- I can think of $H' = (I|H)$ as a parity check matrix in normal form.
- Suppose I start with a random $(2n - k) \times n$ binary matrix and put it in normal form to get $H' = (I|H)$.
- The matrix H will also be randomly distributed.
- But H' being a random code can correct typical errors provided $\frac{k'}{n'} = \frac{n}{2n-k} < 1 - h(p)$, or equivalently

$$R = \frac{k}{n} < \frac{1 - 2h(p)}{1 - h(p)}$$

Normal form

- Every parity check matrix can be put in the form $H = (I|A)$ for some matrix A:
    - Since $\mathcal{C}$ is the kernel of H, it does not change when we do row manipulations on H.
    - By row manipulations (Gaussian elimination) we can put any matrix in normal form $H = (I|A)$.
- I can think of $H' = (I|H)$ as a parity check matrix in normal form.
- Suppose I start with a random $(2n-k) \times n$ binary matrix and put it in normal form to get $H' = (I|H)$.
- The matrix H will also be randomly distributed.
- But H' being a random code can correct typical errors provided $\frac{k'}{n'} = \frac{n}{2n-k} < 1 - h(p)$, or equivalently

$$R = \frac{k}{n} < \frac{1-2h(p)}{1-h(p)}$$

Normal form

- Every parity check matrix can be put in the form $H = (I|A)$ for some matrix A:
  - Since $\mathcal{C}$ is the kernel of H, it does not change when we do row manipulations on H.
  - By row manipulations (Gaussian elimination) we can put any matrix in normal form $H = (I|A)$.
- I can think of $H' = (I|H)$ as a parity check matrix in normal form.
- Suppose I start with a random $(2n - k) \times n$ binary matrix and put it in normal form to get $H' = (I|H)$.
- The matrix H will also be randomly distributed.
- But H' being a random code can correct typical errors provided $\frac{k'}{n'} = \frac{n}{2n-k} < 1 - h(p)$, or equivalently

$$R = \frac{k}{n} < \frac{1 - 2h(p)}{1 - h(p)}$$

Optimality?

- Another way of deriving this upper bound is to say that
  - the number of bits of information learned about the noise, which is $n-k$, the number of syndrome bits
  - equals the entropy produced by the noise, which is $h(p)\times$ (number of bits + number of syndromes) = $h(p)(2n-k)$.
- But this argument does not require the rows of H to be independent, since the rows of $H' = (I|H)$ are linearly independent regardless.
- Linear dependencies among the rows of H increases the rate since some of the constraints are redundant.
  - To what extent can we increase the rate of the code by making rows of H redundant while keeping H' a good code?
  - Does this count as single-shot? We would be making redundant measurements.

Optimality?

- Another way of deriving this upper bound is to say that
  - the number of bits of information learned about the noise, which is $n - k$, the number of syndrome bits
  - equals the entropy produced by the noise, which is $h(p)\times$ (number of bits + number of syndromes) = $h(p)(2n - k)$.
- But this argument does not require the rows of H to be independent, since the rows of $H' = (I|H)$ are linearly independent regardless.
- Linear dependencies among the rows of H increases the rate since some of the constraints are redundant.
  - To what extent can we increase the rate of the code by making rows of H redundant while keeping H' a good code?
  - Does this count as single-shot? We would be making redundant measurements.

Optimality?

- Another way of deriving this upper bound is to say that
  - the number of bits of information learned about the noise, which is $n - k$, the number of syndrome bits
  - equals the entropy produced by the noise, which is $h(p)\times$ (number of bits + number of syndromes) = $h(p)(2n - k)$.
- But this argument does not require the rows of H to be independent, since the rows of $H' = (I|H)$ are linearly independent regardless.
- Linear dependencies among the rows of H increases the rate since some of the constraints are redundant.
  - To what extent can we increase the rate of the code by making rows of H redundant while keeping H' a good code?
  - Does this count as single-shot? We would be making redundant measurements.

Optimality?

- Another way of deriving this upper bound is to say that
  - the number of bits of information learned about the noise, which is $n-k$, the number of syndrome bits
  - equals the entropy produced by the noise, which is $h(p)\times$ (number of bits + number of syndromes) = $h(p)(2n-k)$.
- But this argument does not require the rows of H to be independent, since the rows of $H' = (I|H)$ are linearly independent regardless.
- Linear dependencies among the rows of H increases the rate since some of the constraints are redundant.
  - To what extent can we increase the rate of the code by making rows of H redundant while keeping H' a good code?
  - Does this count as single-shot? We would be making redundant measurements.

Optimality?

- Another way of deriving this upper bound is to say that
  - the number of bits of information learned about the noise, which is $n-k$, the number of syndrome bits
  - equals the entropy produced by the noise, which is $h(p)\times$ (number of bits + number of syndromes) = $h(p)(2n-k)$.
- But this argument does not require the rows of H to be independent, since the rows of $H' = (I|H)$ are linearly independent regardless.
- Linear dependencies among the rows of H increases the rate since some of the constraints are redundant.
  - To what extent can we increase the rate of the code by making rows of H redundant while keeping H' a good code?
  - Does this count as single-shot? We would be making redundant measurements.

Optimality?

- Another way of deriving this upper bound is to say that
  - the number of bits of information learned about the noise, which is $n-k$, the number of syndrome bits
  - equals the entropy produced by the noise, which is $h(p)\times$ (number of bits + number of syndromes) = $h(p)(2n-k)$.
- But this argument does not require the rows of H to be independent, since the rows of $H' = (I|H)$ are linearly independent regardless.
- Linear dependencies among the rows of H increases the rate since some of the constraints are redundant.
  - To what extent can we increase the rate of the code by making rows of H redundant while keeping H' a good code?
  - Does this count as single-shot? We would be making redundant measurements.

Optimality?

- Another way of deriving this upper bound is to say that
  - the number of bits of information learned about the noise, which is $n - k$, the number of syndrome bits
  - equals the entropy produced by the noise, which is $h(p)\times$ (number of bits + number of syndromes) = $h(p)(2n - k)$.
- But this argument does not require the rows of H to be independent, since the rows of $H' = (I|H)$ are linearly independent regardless.
- Linear dependencies among the rows of H increases the rate since some of the constraints are redundant.
  - To what extent can we increase the rate of the code by making rows of H redundant while keeping H' a good code?
  - Does this count as single-shot? We would be making redundant measurements.

Phenomenological noise?

- In a random code, each check involves about $n/2$ bits.
- When measuring such a high-weight operator, errors will build up.
- It is not reasonable to assume that syndrome bits are accurate with probability $1-p$ whit a small p.
- This phenomenological noise model is only decent when check operators are simple, i.e., for LDPC codes.

Phenomenological noise?

- In a random code, each check involves about $n/2$ bits.
- When measuring such a high-weight operator, errors will build up.
- It is not reasonable to assume that syndrome bits are accurate with probability $1 - p$ whit a small p.
- This phenomenological noise model is only decent when check operators are simple, i.e., for LDPC codes.

Phenomenological noise?

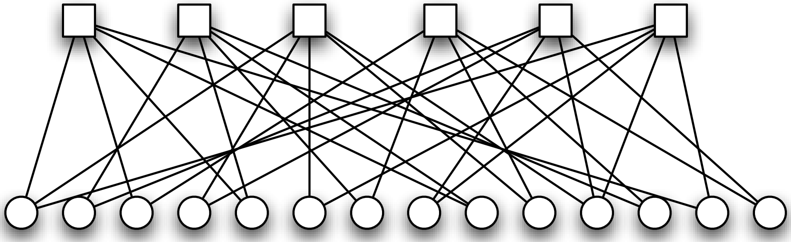
- In a random code, each check involves about $n/2$ bits.
- When measuring such a high-weight operator, errors will build up.
- It is not reasonable to assume that syndrome bits are accurate with probability $1-p$ whit a small p.
- This phenomenological noise model is only decent when check operators are simple, i.e., for LDPC codes.

Phenomenological noise?

- In a random code, each check involves about $n/2$ bits.
- When measuring such a high-weight operator, errors will build up.
- It is not reasonable to assume that syndrome bits are accurate with probability $1 - p$ whit a small p.
- This phenomenological noise model is only decent when check operators are simple, i.e., for LDPC codes.

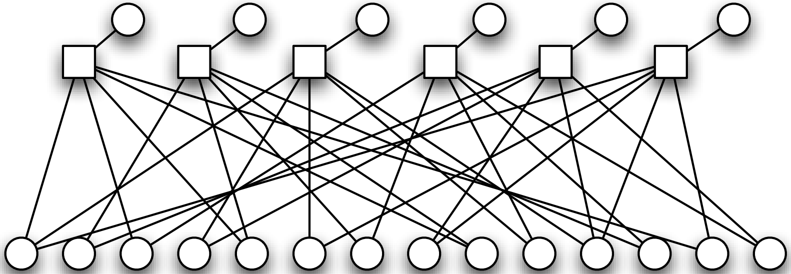
Outline

Sparse codes cannot be single-shot FT



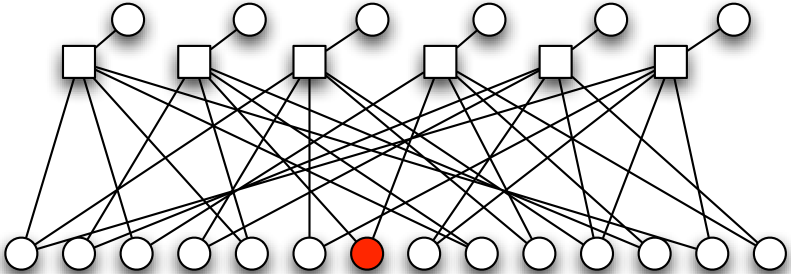
- The 'distance' of the code $H' = (I|H)$ is at most $\lambda_c + 1$
  - An error on a single bit along with an error on the $\leq \lambda_c$ connected checks goes undetected.

Sparse codes cannot be single-shot FT



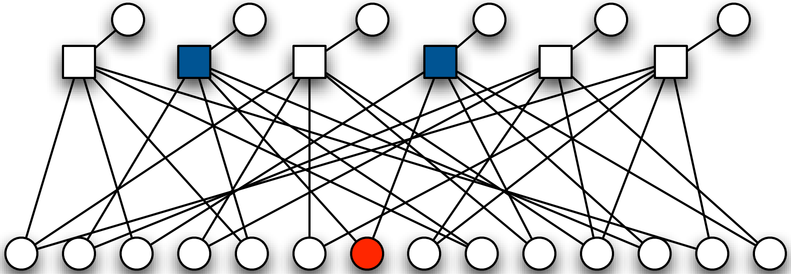
- The 'distance' of the code $H' = (I|H)$ is at most $\lambda_c + 1$
  - An error on a single bit along with an error on the $\leq \lambda_c$ connected checks goes undetected.

Sparse codes cannot be single-shot FT



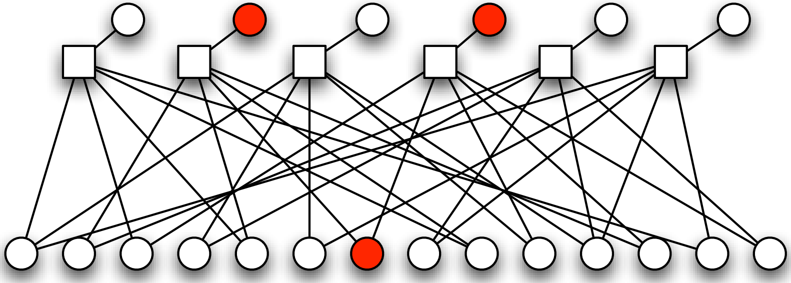
- The 'distance' of the code $H' = (I|H)$ is at most $\lambda_c + 1$
  - An error on a single bit along with an error on the $\leq \lambda_c$ connected checks goes undetected.

Sparse codes cannot be single-shot FT



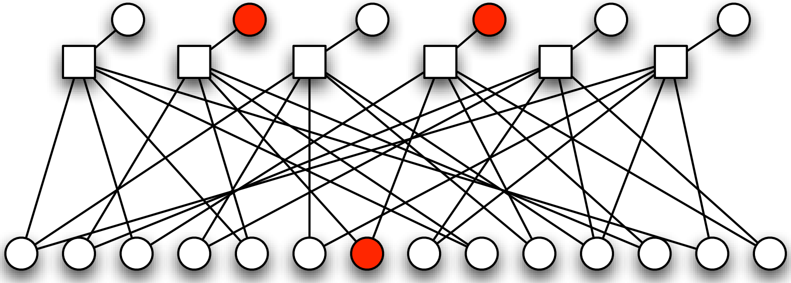
- The 'distance' of the code $H' = (I|H)$ is at most $\lambda_c + 1$
  - An error on a single bit along with an error on the $\leq \lambda_c$ connected checks goes undetected.

Sparse codes cannot be single-shot FT



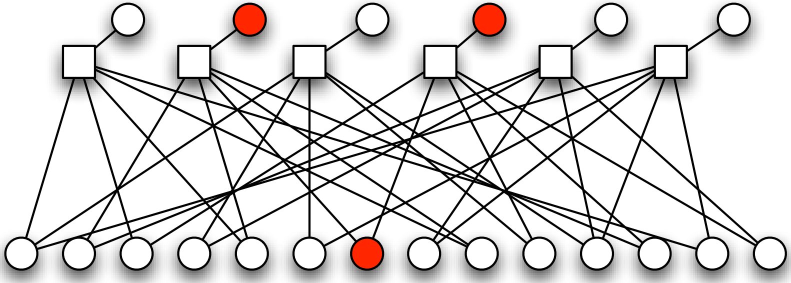
- The 'distance' of the code $H' = (I|H)$ is at most $\lambda_c + 1$
  - An error on a single bit along with an error on the $\leq \lambda_c$ connected checks goes undetected.

Sparse codes cannot be single-shot FT



- The 'distance' of the code $H' = (I|H)$ is at most $\lambda_c + 1$
  - An error on a single bit along with an error on the $\leq \lambda_c$ connected checks goes undetected.

Sparse codes cannot be single-shot FT



- The 'distance' of the code $H' = (I|H)$ is at most $\lambda_c + 1$
  - An error on a single bit along with an error on the $\leq \lambda_c$ connected checks goes undetected.

Sparse codes cannot be single-shot FT

- The generating matrix associated to H' is $G' = \left(\frac{H}{I}\right)$

$$H'G' = (I|H)\left(\frac{H}{I}\right) = H + H = 0.$$

- If H is sparse, then so is G', so the corresponding codes has many low-weight codewords; it's a bad code.

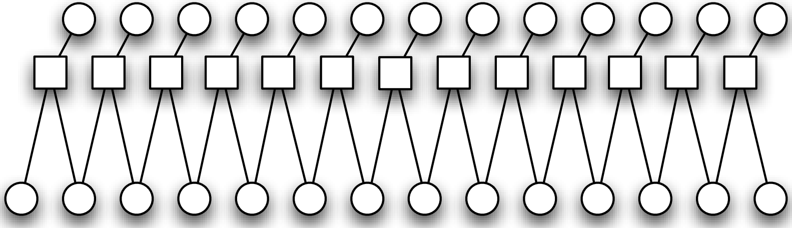
Sparse codes cannot be single-shot FT

- The generating matrix associated to H' is $G' = \left(\frac{H}{I}\right)$

$$H'G' = (I|H)\left(\frac{H}{I}\right) = H + H = 0.$$

- If H is sparse, then so is G', so the corresponding codes has many low-weight codewords; it's a bad code.

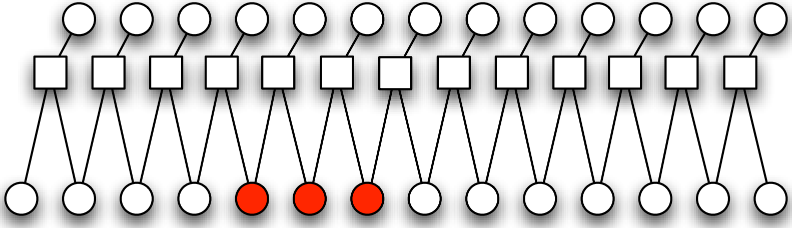
Relation to self correction



For typical errors, we need $|s(E)| > |E|$.

Self-correction requires $|s(E)| > |E|^\delta$ for $\delta > 0$?

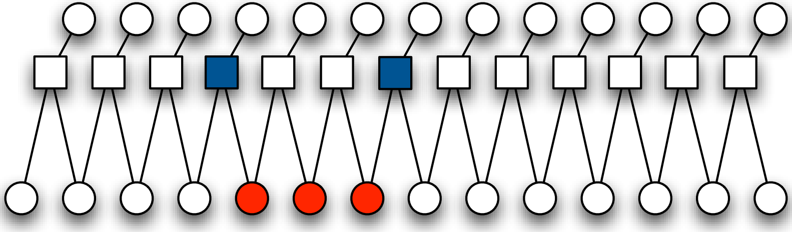
Relation to self correction



For typical errors, we need $|s(E)| > |E|$.

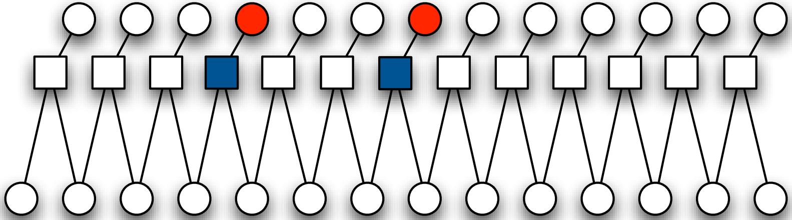
Self-correction requires $|s(E)| > |E|^\delta$ for $\delta > 0$?

Relation to self correction



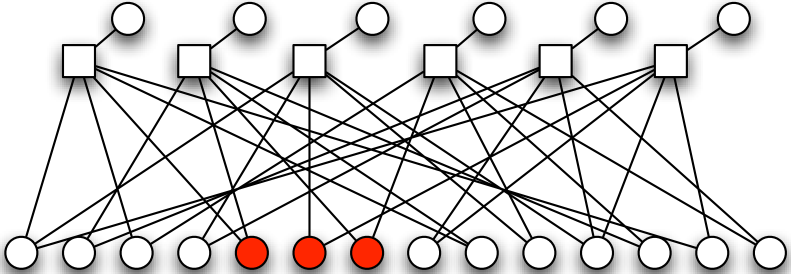
For typical errors, we need $|s(E)| > |E|$.
Self-correction requires $|s(E)| > |E|^\delta$ for $\delta > 0$?

Relation to self correction



For typical errors, we need $|s(E)| > |E|$.
Self-correction requires $|s(E)| > |E|^\delta$ for $\delta > 0$?

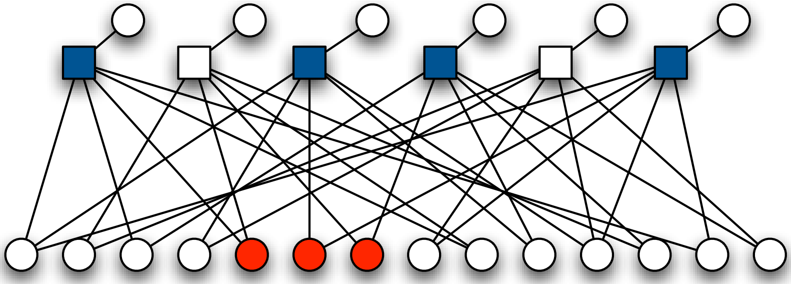
Relation to self correction



For typical errors, we need $|s(E)| > |E|$.

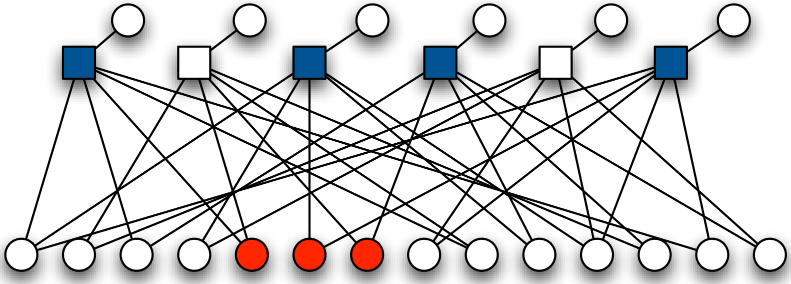
Self-correction requires $|s(E)| > |E|^\delta$ for $\delta > 0$?

Relation to self correction



For typical errors, we need $|s(E)| > |E|$.
Self-correction requires $|s(E)| > |E|^\delta$ for $\delta > 0$?

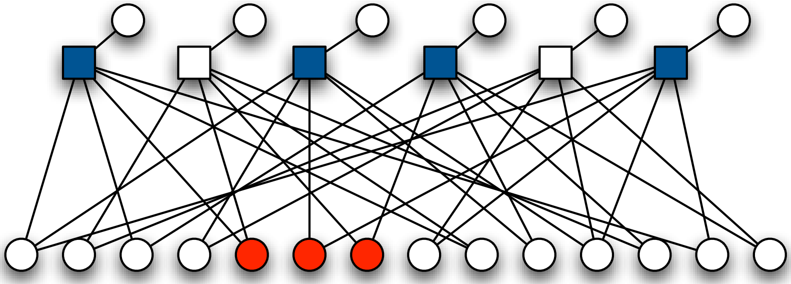
Relation to self correction



For typical errors, we need $|s(E)| > |E|$.

Self-correction requires $|s(E)| > |E|^\delta$ for $\delta > 0$?

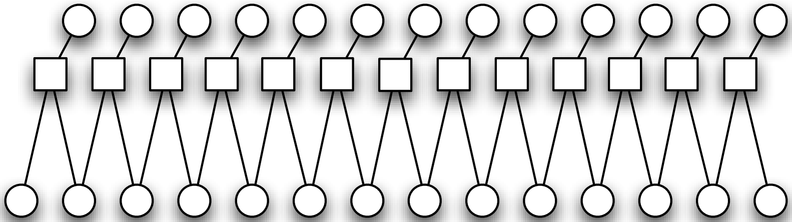
Relation to self correction



For typical errors, we need $|s(E)| > |E|$.
Self-correction requires $|s(E)| > |E|^\delta$ for $\delta > 0$?

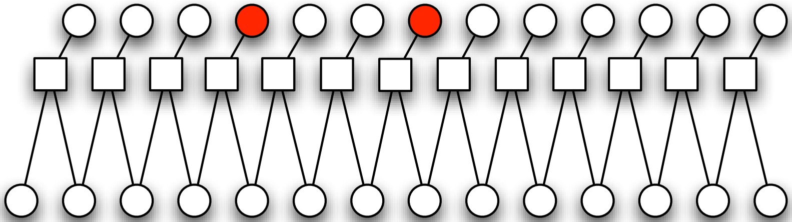
Outline

Bounded error growth



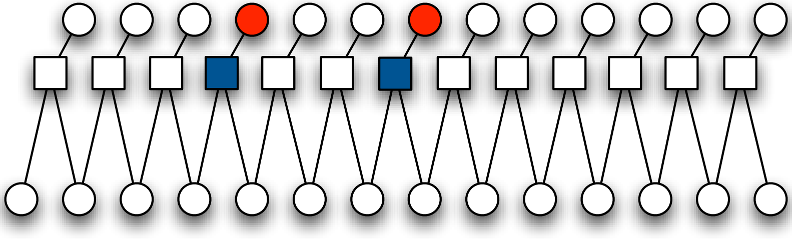
- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
  - Is the 1D Ising chain single-shot FT?

Bounded error growth



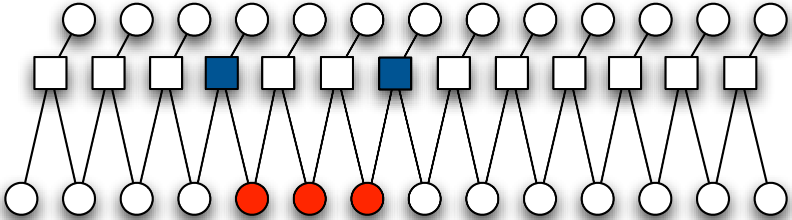
- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
  - Is the 1D Ising chain single-shot FT?

Bounded error growth



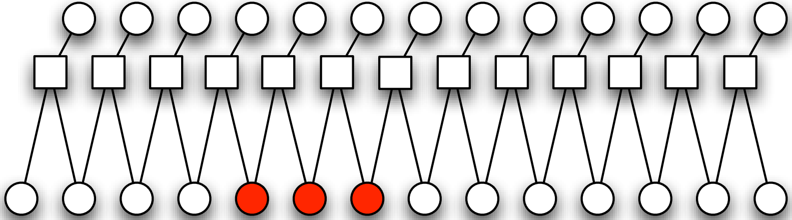
- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
  - Is the 1D Ising chain single-shot FT?

Bounded error growth



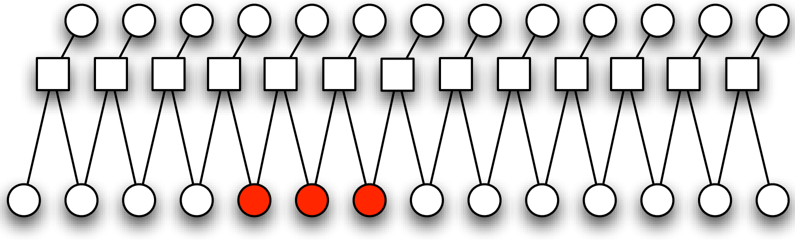
- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
  - Is the 1D Ising chain single-shot FT?

Bounded error growth



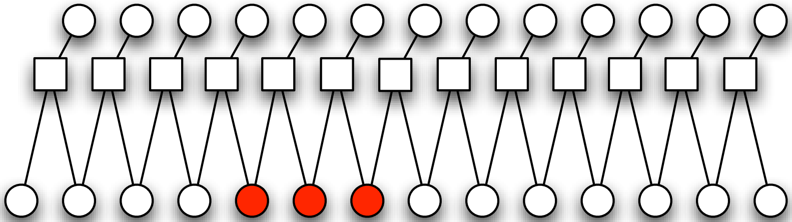
- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
  - Is the 1D Ising chain single-shot FT?

Bounded error growth



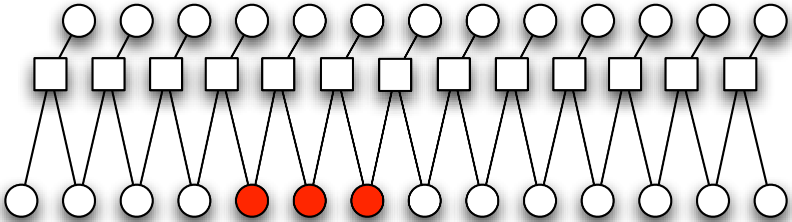
- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
    - Is the 1D Ising chain single-shot FT?

Bounded error growth



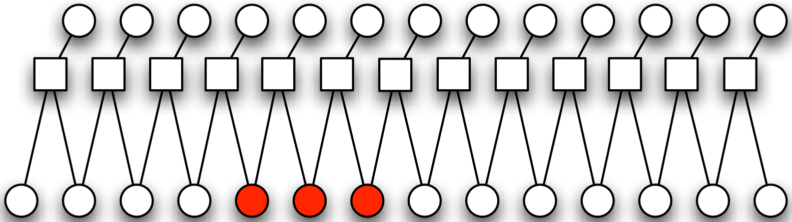
- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
  - Is the 1D Ising chain single-shot FT?

Bounded error growth



- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
  - Is the 1D Ising chain single-shot FT?

Bounded error growth



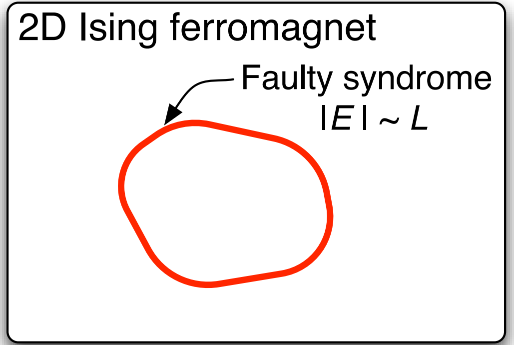
- Error correction has increased the size of the error.
- Condition: $|EC(E)| < |E|$?
- A decoder which takes into account the fact that syndrome can be faulty would not have increased the error size.
  - Is the 1D Ising chain single-shot FT?

Self-correction not enough?

2D Ising ferromagnet

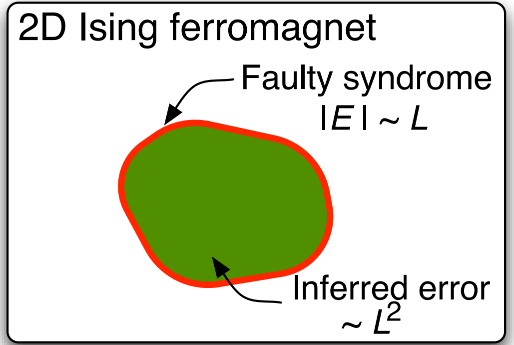
- Error correction has increased the error from $E \sim L$ to $EC(E) \sim L^2$.
- Again an informed decoder would not have done this.
- Should the right condition be $|EC(E)| < |E|^c$ for $c > 0$?
- An operational definition would be nice!

Self-correction not enough?



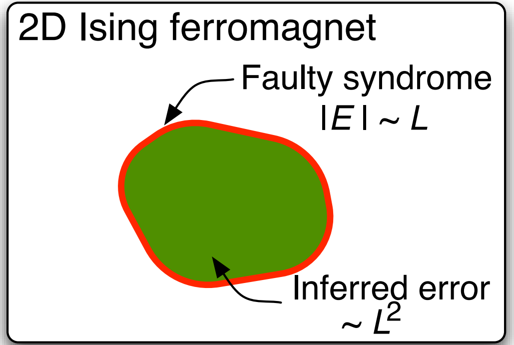
- Error correction has increased the error from $E \sim L$ to $EC(E) \sim L^2$.
- Again an informed decoder would not have done this.
- Should the right condition be $|EC(E)| < |E|^c$ for $c > 0$?
- An operational definition would be nice!

Self-correction not enough?



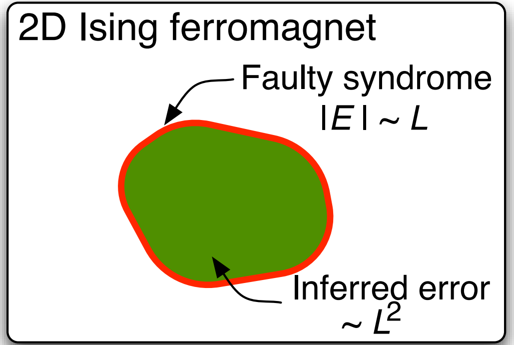
- Error correction has increased the error from $E \sim L$ to $EC(E) \sim L^2$.
- Again an informed decoder would not have done this.
- Should the right condition be $|EC(E)| < |E|^c$ for $c > 0$?
- An operational definition would be nice!

Self-correction not enough?



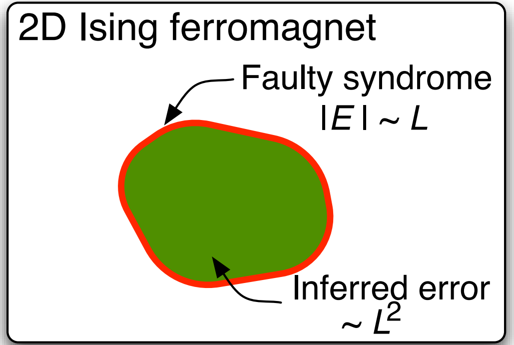
- Error correction has increased the error from $E \sim L$ to $EC(E) \sim L^2$.
- Again an informed decoder would not have done this.
- Should the right condition be $|EC(E)| < |E|^c$ for $c > 0$?
- An operational definition would be nice!

Self-correction not enough?



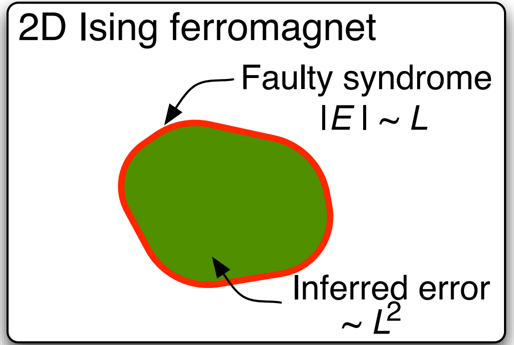
- Error correction has increased the error from $E \sim L$ to $EC(E) \sim L^2$.
- Again an informed decoder would not have done this.
- Should the right condition be $|EC(E)| < |E|^c$ for $c > 0$?
- An operational definition would be nice!

Self-correction not enough?



- Error correction has increased the error from $E \sim L$ to $EC(E) \sim L^2$.
- Again an informed decoder would not have done this.
- Should the right condition be $|EC(E)| < |E|^c$ for $c > 0$?
- An operational definition would be nice!

Self-correction not enough?



- Error correction has increased the error from $E \sim L$ to $EC(E) \sim L^2$.
- Again an informed decoder would not have done this.
- Should the right condition be $|EC(E)| < |E|^c$ for $c > 0$?
- An operational definition would be nice!

Bombin's definition

- $\tau =$ Syndrome temperature, how much it's confined.
- $\epsilon =$ Logical error rate.
- $\eta =$ Syndrome error rate.

$$EC_\eta \circ \mathcal{E}_{\tau,\epsilon} \sim \mathcal{E}_{\tau',\epsilon+\delta}$$

- $\eta \to 0 \quad \Rightarrow \quad \tau' \to 0$.
- $L \to \infty \quad \Rightarrow \quad \delta \to 0$.

Bombin's definition

- $\tau =$ Syndrome temperature, how much it's confined.
- $\epsilon =$ Logical error rate.
- $\eta =$ Syndrome error rate.

$$EC_\eta \circ \mathcal{E}_{\tau,\epsilon} \sim \mathcal{E}_{\tau',\epsilon+\delta}$$

- $\eta \to 0 \quad \Rightarrow \quad \tau' \to 0$.
- $L \to \infty \quad \Rightarrow \quad \delta \to 0$.

Bombin's definition

- $\tau =$ Syndrome temperature, how much it's confined.
- $\epsilon =$ Logical error rate.
- $\eta =$ Syndrome error rate.

$$EC_\eta \circ \mathcal{E}_{\tau,\epsilon} \sim \mathcal{E}_{\tau',\epsilon+\delta}$$

- $\eta \to 0 \quad \Rightarrow \quad \tau' \to 0$.
- $L \to \infty \quad \Rightarrow \quad \delta \to 0$.

Bombin's definition

- $\tau =$ Syndrome temperature, how much it's confined.
- $\epsilon =$ Logical error rate.
- $\eta =$ Syndrome error rate.

$$EC_\eta \circ \mathcal{E}_{\tau,\epsilon} \sim \mathcal{E}_{\tau',\epsilon+\delta}$$

- $\eta \to 0 \quad \Rightarrow \quad \tau' \to 0$.
- $L \to \infty \quad \Rightarrow \quad \delta \to 0$.

Bombin's definition

- $\tau =$ Syndrome temperature, how much it's confined.
- $\epsilon =$ Logical error rate.
- $\eta =$ Syndrome error rate.

$$EC_\eta \circ \mathcal{E}_{\tau,\epsilon} \sim \mathcal{E}_{\tau',\epsilon+\delta}$$

- $\eta \to 0 \quad \Rightarrow \quad \tau' \to 0$.
- $L \to \infty \quad \Rightarrow \quad \delta \to 0$.

Bombin's definition

- $\tau =$ Syndrome temperature, how much it's confined.
- $\epsilon =$ Logical error rate.
- $\eta =$ Syndrome error rate.

$$EC_\eta \circ \mathcal{E}_{\tau,\epsilon} \sim \mathcal{E}_{\tau',\epsilon+\delta}$$

- $\eta \to 0 \quad \Rightarrow \quad \tau' \to 0$.
- $L \to \infty \quad \Rightarrow \quad \delta \to 0$.

Bombin's definition

- $\tau =$ Syndrome temperature, how much it's confined.
- $\epsilon =$ Logical error rate.
- $\eta =$ Syndrome error rate.

$$EC_\eta \circ \mathcal{E}_{\tau,\epsilon} \sim \mathcal{E}_{\tau',\epsilon+\delta}$$

- $\eta \to 0 \quad \Rightarrow \quad \tau' \to 0$.
- $L \to \infty \quad \Rightarrow \quad \delta \to 0$.

I need to sober up to think about that one!

Outline

Summary

- Strict single-shot error correction with a phenomenological noise model is possible using random codes.
  - Derived a lower bound on achievable rate.
  - Upper bound depends on what counts as single-shot.
  - Phenomenological noise model not justified in this setting.
- Strict single-shot error correction is not possible with sparse codes, where the phenomenological noise model is justified.
- What is the right notion of single-shot error-correction and what does it imply operationally?

Summary

- Strict single-shot error correction with a phenomenological noise model is possible using random codes.
  - Derived a lower bound on achievable rate.
  - Upper bound depends on what counts as single-shot.
  - Phenomenological noise model not justified in this setting.
- Strict single-shot error correction is not possible with sparse codes, where the phenomenological noise model is justified.
- What is the right notion of single-shot error-correction and what does it imply operationally?

Summary

- Strict single-shot error correction with a phenomenological noise model is possible using random codes.
  - Derived a lower bound on achievable rate.
  - Upper bound depends on what counts as single-shot.
  - Phenomenological noise model not justified in this setting.
- Strict single-shot error correction is not possible with sparse codes, where the phenomenological noise model is justified.
- What is the right notion of single-shot error-correction and what does it imply operationally?

Summary

- Strict single-shot error correction with a phenomenological noise model is possible using random codes.
  - Derived a lower bound on achievable rate.
  - Upper bound depends on what counts as single-shot.
  - Phenomenological noise model not justified in this setting.
- Strict single-shot error correction is not possible with sparse codes, where the phenomenological noise model is justified.
- What is the right notion of single-shot error-correction and what does it imply operationally?

Summary

- Strict single-shot error correction with a phenomenological noise model is possible using random codes.
  - Derived a lower bound on achievable rate.
  - Upper bound depends on what counts as single-shot.
  - Phenomenological noise model not justified in this setting.
- Strict single-shot error correction is not possible with sparse codes, where the phenomenological noise model is justified.
- What is the right notion of single-shot error-correction and what does it imply operationally?

Summary

- Strict single-shot error correction with a phenomenological noise model is possible using random codes.
  - Derived a lower bound on achievable rate.
  - Upper bound depends on what counts as single-shot.
  - Phenomenological noise model not justified in this setting.
- Strict single-shot error correction is not possible with sparse codes, where the phenomenological noise model is justified.
- What is the right notion of single-shot error-correction and what does it imply operationally?