

Title: Quantum Error Correction 5A

Date: Feb 06, 2007 03:30 PM

URL: <http://pirsa.org/07020017>

Abstract: Generators of symplectic group, quantum Gilbert-Varshamov bound, quantum Hamming bound, quantum Singleton bound

Thm.: $Sp(2n, \mathbb{Z}_2)$ is generated by $H, R, CNOT$.

Thm.: $Sp(2n, \mathbb{Z}_2)$ is generated by $H, R, CNOT$.

CNOT:



Thm.: $Sp(2n, \mathbb{Z}_2)$ is generated by $H, R, CNOT$. $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in Sp(2n)$

CNOT:



Thm.: $\text{Sp}(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT . $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathcal{S}(2n)$

On left: $\text{CNOT}(i, j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

Thm.: $\text{Sp}(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT . $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \text{Sp}(2n)$

On left: $\text{CNOT}(i,j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H()$:

Thm.: $\text{Sp}(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT . $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathbb{M}_{2n}$

$\text{On left: CNOT } (i, j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(\cdot)$: Switches i th row from top half with i th row in bottom half

$R(\cdot)$:

Thm.: $\text{Sp}(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT . $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \text{Sp}(2n)$

On left: $\text{CNOT}(i, j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(i)$: Switches i th row from top half with i th row in bottom half

$R(i)$: Adds i th row from top half to i th row in bottom half

Thm.: $\text{Sp}(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT . $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \text{Sp}(2n)$

$\text{On left: CNOT } (i, j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(i)$: Switches i th row from top half with i th row in bottom half

$R(i)$: Adds i th row from top half to i th row in bottom half

$\text{III} = \text{SWAP}(i, j)$: Swap i th & j th rows in both top & bottom half

$\text{Controlled-Z } (i, j)$:

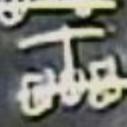
Thm.: $Sp(2n, \mathbb{Z}_2)$ is generated by $H, R, CNOT$. $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in Sp(2n)$

On left: $CNOT(i,j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(i)$: Switches i th row from top half with i th row in bottom half

$R(i)$: Adds i th row from top half to i th row in bottom half

 $= SHAP(i,j)$: Swap i th & j th rows in both top & bottom half

 $= CNOT(i,j)$: Adds i th row of top half to j th row of bottom half
Add j th row of top half to i th row of bottom half

Thm.: $S_P(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT . $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in S_P(2n)$

On Left: CNOT (i, j): Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(i)$: Switches i th row from top half with i th row in bottom half

$R(i)$: Adds i th row from top half to i th row in bottom half

$\begin{array}{c} \text{I} \\ \text{II} \\ \text{III} \end{array} = \text{SHAP}(i, j)$: Swap i th & j th rows in both top & bottom half

$\begin{array}{c} \text{I} \\ \text{II} \\ \text{III} \end{array} = \text{Controlled-Z}(i, j)$: Adds i th row of top half to j th row of bottom half
Add j th row of top half to i th row of bottom half

-~~Q~~-~~Q~~-~~Q~~-~~Q~~

Thm: $\text{Sp}(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT . $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \text{Sp}(2n)$

On left: $\text{CNOT}(i,j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(i)$: Switches i th row from top half with i th row in bottom half

$R(i)$: Adds i th row from top half to i th row in bottom half

~~$\prod_{i=1}^n \prod_{j=1}^{2i-1} = \text{SHAP}(i,j)$~~ : swaps i th & j th rows in both top & bottom half

~~$\prod_{i=1}^n \prod_{j=1}^{2i-1} = \text{Control-2}(i,j)$~~ : Adds i th row of top half to j th row of bottom half
 Adds j th row of top half to i th row of bottom half

~~$-HRB-Q(i)$~~ : Adds i th row of bottom half to i th of top half

On right:

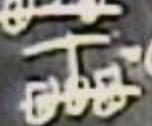
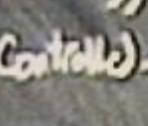
Thm: $Sp(2n, \mathbb{Z}_2)$ is generated by $H, R, CNOT$. $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in Sp(2n)$

On left: $CNOT(i,j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(i)$: Switches i th row from top half with i th row in bottom half

$R(i)$: Adds i th row from top half to i th row in bottom half

 - $SHAP(i,j)$: swaps i th & j th rows in both top & bottom half

 - $Cont(i,j) = Z(i,j)$: Adds i th row of top half to j th row of bottom half
 Adds j th row of top half to i th row of bottom half

~~- $HRB = Q(i)$~~ : Adds i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow left, bottom \rightarrow right, opposite direction

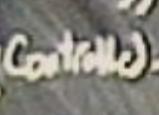
Thm: $\text{Sp}(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT . $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \text{Sp}(2n)$

On left: $\text{CNOT}(i,j)$: Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(i)$: Switches i th row from top half with i th row in bottom half

$R(i)$: Adds i th row from top half to i th row in bottom half

 - $\text{SHAP}(i,j)$: swaps i th & j th rows in both top & bottom half

 - $\text{Control-2}(i,j)$: Adds i th row of top half to j th row of bottom half
~~Control-2~~  Adds j th row of top half to i th row of bottom half

~~-HRB-Q(i)~~: Adds i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

$\Rightarrow \text{NHT}(i)$: Add row i of bottom half to top half

T-Controller- $T(i,j)$: Add j th row of top left to j th row of bottom half
Add j th row of top half to j th row of bottom half

Q-Controller- $Q(i)$: Add i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put I in upper left corner of A

$\rightarrow \text{NHT}(i,j)$: Add row i of left half to row j of bottom half

~~Top Controller~~- $T(i,j)$: Add i th row of top half to j th row of bottom half
Add j th row of top half to i th row of bottom half

~~Bottom Controller~~- $Q(i)$: Add all rows of bottom half to rows of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put I in upper left corner of A using

$\rightarrow \text{H}(i)$: Add i th row of top half to i th row of bottom half

~~Top Controller~~- $C(i,j)$: Add i th row of top half to j th row of bottom half
Add j th row of top half to i th row of bottom half

~~Bottom~~- $Q(i)$: Add i th row of bottom half to i th row of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP H.
2. Use CNOT($1,i$) to make rest of 1st column of A = 0

~~-SWAP(1,j)~~: Swap rows in both bottom half

~~-T-CNOT(i,j)~~: Add i th row of top half to j th row of bottom half
~~-CNOT(j,i)~~: Add j th row of top half to i th row of bottom half

~~-TQFT-Q(i)~~: Add i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.
2. Use CNOT(1,j) to make rest of 1st column of A 0

$$A = \begin{pmatrix} 1 & ? \\ 0 & ? \end{pmatrix}$$

~~-SWAP(i,j)~~: Swap rows in same top/bottom half

~~-CNOT(i,j)~~: Add i th row of top half to j th row of bottom half
Add j th row of top half to i th row of bottom half

~~-Q(i)~~: Add i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP H.
2. Use CNOT(1,i) to make rest of 1st column of A = 0

3.

$$A = \begin{pmatrix} 1 & ? \\ 0 & ? \end{pmatrix},$$

$\rightarrow \text{NOT}(i,j)$: Add row i to column j in bottom right half

$\rightarrow \text{CNOT-Controller-2}(i,j)$: Add i th row of top-left to j th row of bottom half
Add j th row of top half to i th row of bottom half

$\rightarrow \text{CNOT-Q}(i)$: Add i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.

2. Use CNOT(i,i) to make rest of 1st column of A = 0

3. Use R(i) & CNOT(i,i) to make 1st column of C = 0 $A = \begin{pmatrix} 1 & * \\ 0 & 0 \end{pmatrix},$

$\rightarrow \text{SWAP}(i,j)$: Swap rows in each representation half

$\rightarrow \text{CNOT}(\text{Control}-i, j)$: Add i th row of top half to j th row of bottom half
 $\rightarrow \text{CNOT}(\text{Control}-j, i)$: Add j th row of top half to i th row of bottom half

$\rightarrow \text{CNOT}(i, j)$: Add i th row of bottom half to j th row of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.
2. Use CNOT($1, i$) to make rest of 1st column of A = 0
- 3.
4. Use $R(1) \otimes \text{Control}-i, j$ to make 1st column of C = 0 $A = \begin{pmatrix} 1 & ? \\ 0 & ? \end{pmatrix}, C = \begin{pmatrix} 1 & ? \\ 0 & ? \end{pmatrix}$

$\rightarrow \text{NOT}(1)$: Add row 1 to rest of bottom half

$\rightarrow \text{CNOT}(\text{Control})(1,j)$: Add row 1 of top half to j-th row of bottom half
Add j-th row of top half to 1st row of bottom half

$\rightarrow \text{QCD}(Q(1))$: Add row 1 of bottom half to 1st of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.
2. Use CNOT(1,i) to make rest of 1st column of A = 0
3. Use CNOT on right to eliminate 1st row of A
4. Use $R(1) \& \text{CNOT}(\text{Control})(1,i)$ to make 1st column of C = 0 $A = \begin{pmatrix} 1 & ? \\ 0 & ? \end{pmatrix}, C = \begin{pmatrix} ? & ? \\ 0 & ? \end{pmatrix}$

-SWAP(1, j): Swap rows in left top-left half

-T_{Controlled}-Z(1, j): Add 1st row of top half to j-th row of bottom half
Add j-th row of top half to 1st row of bottom half

-T_{Controlled}-Q(1): Add 1st row of bottom half to 1st of top half

On right: row \rightarrow column, top = r, c, bottom = l, f

1. Put 1 in upper left corner of A using SWAP, H.

2. Use CNOT(1, i) to make rest of 1st column of A = 0

3. Use CNOT on right to eliminate 1st row of A

4. Use R(1) & Control_z-Z(1, i) to make 1st column of C = 0 $A = \begin{bmatrix} 1 & * & * \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$,

5. 1st row of C is now 0 too!

~~-SWAP(1,2)~~: Swap rows in left-top-left half

~~-CNOT-Controller-2(1,2)~~: Add 1st row of top half to 2nd row of bottom half
Add 2nd row of top half to 1st row of bottom half

~~-CNOT-Q(1)~~: Add 1st row of bottom half to 1st of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.

2. Use CNOT(1,i) to make rest of 1st column of A = 0
3. Use CNOT on right to clean 1st row of A
4. Use R(1) & Controller-2(1,2) to make 1st column of C = 0 $A = \begin{pmatrix} 1 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$,
5. 1st row of C is now 0 too! Columns \uparrow \downarrow count

$$(\bar{a}, |\bar{c}\rangle) \cdot (\bar{a}, |\bar{c}\rangle) =$$

$\rightarrow \text{HAT}(i,j)$: Add row i of top half to j th row of bottom half

$\rightarrow \text{CNOT}(\text{Controlled}-\bar{c}(i,j))$: Add i th row of top half to j th row of bottom half
Add j th row of top half to i th row of bottom half

$\rightarrow \text{QD}(i)$: Add i th row of bottom half to i th row of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.

2. Use CNOT(1,i) to make rest of 1st column of A = 0

3. Use CNOT on right to eliminate 1st row of A

4. Use R(1) & Controlled- $\bar{c}(1,:)$ to make 1st column of C = 0 $A = \begin{pmatrix} 1 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$

5. 1st row of C is now 0 too! Columns on left come to $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

~~-SWAP(1,2)~~: Swap rows in left top-left half

~~-CNOT_{Control}(1,2)~~: Add 1st row of top half to 2nd row of bottom half
~~-CNOT_{Control}(1,2)~~: Add 2nd row of top half to 1st row of bottom half

~~-Q(1)~~: Add 1st row of bottom half to 1st of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.

2. Use CNOT(1,i) to make rest of 1st column of A = 0

3. Use CNOT on right to eliminate 1st row of A

4. Use R(1) & CNOT_{Control}(1,2) to make 1st column of C = 0 $A = \begin{pmatrix} 1 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$, $C = \begin{pmatrix} 0 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$

5. 1st row of C is now 0 too! Columns on left come to $C = \begin{pmatrix} 0 & * & * \\ 0 & 1 & * \\ 0 & * & * \end{pmatrix}$

\rightarrow SWAP(1,2) : Swap rows in left top-left half

\rightarrow CNOT_{Controlled-2(1,2)}: Add 1st row of top-left to 2nd row of bottom half
Add 2nd row of top-left to 1st row of bottom half

\rightarrow CNOT_{=Q(1)}: Add 1st row of bottom half to 1st of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.

2. Use CNOT(1,i) to make rest of 1st column of A = 0

3. Use CNOT on right to eliminate 1st row of A

4. Use R(1) & CNOT_{Controlled-2(1,1)} to make 1st column of C = 0 $A = \begin{pmatrix} 1 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$, $C = \begin{pmatrix} 0 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$

5. 1st row of C is now 0 too! Columns on left come to $C = \begin{pmatrix} 0 & * & * \\ 0 & 1 & * \\ 0 & * & * \end{pmatrix}$

6. Repeat 1-5 with 2nd row & column, etc. to get $A = I, C = 0$

$\text{SWAP}(1,2)$: Swap rows in each half

$\text{CNOT}_{\text{Control}}(2,1)$: Add 2nd row of top half to 1st row of bottom half
 $\text{CNOT}_{\text{Control}}(2,3)$: Add 2nd row of top half to 3rd row of bottom half

$\text{CNOT}_{\text{Control}}(3,1)$: Add 3rd row of bottom half to 1st of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.

2. Use CNOT(1,i) to make rest of 1st column of A = 0

3. Use CNOT on right to eliminate 1st row of A

4. Use $R(1)$ & $\text{CNOT}_{\text{Control}}(2,1)$ to make 1st column of C = 0 $A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 \\ 0 & ? \end{pmatrix}$

5. 1st row of C is now 0 too! Columns on right come to $C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

6. Repeat 1-5 with 2nd row & column, etc. to get $A = I, C = O$

?

-SWAP(1,j): Swap rows in both top/bottom half

-CNOT_{Control}-Z(1,j): Add 1st row of top half to jth row of bottom half
Add jth row of top half to 1st row of bottom half

-Q_b(j): Add 1st row of bottom half to 1st of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.

2. Use CNOT(1,i) to make rest of 1st column of A = 0

3. Use CNOT on right to eliminate 1st row of A

4. Use R(1) & Control_b-Z(1,:) to make 1st column of C=0 $A = \begin{pmatrix} 1 & * & * \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, $C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

5. 1st row of C is now 0 too! Columns on left come to $C = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
 $(\bar{c}_1 | \bar{c}_1) \cdot (\bar{c}_1 | \bar{c}_1) = c_{1,1} = 0$

6. Repeat 1-5 with 2nd row/column, etc. to get $A=I, C=0$

7. Use Q_b Control_b-Z on R

Thm: $Sp(2n, \mathbb{Z}_2)$ is generated by $H, R, CNOT$. $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in Sp(2n)$

On left: CNOT(i,j): Add the i th row in top half to j th row in top half
Add the j th row in bottom half to i th row in bottom half

$H(i)$: Switches i th row from top half with i th row in bottom half

$R(i)$: Adds i th row from top half to i th row in bottom half

$\text{SWAP}(i,j)$: Swaps i th & j th rows in both top & bottom half

$\text{Controlled-Z}(i,j)$: Adds i th row of top half to j th row of bottom half
Adds j th row of top half to i th row of bottom half

$$\begin{pmatrix} 1 & 0 & p & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$\text{-HGB-Q}(i)$: Adds i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

$\text{CNOT}_{\text{Control}}(1,1)$: Adds 1st row of top half to 1st row of bottom half
 इसका अर्थ है कि 1st row of top half to 1st row of bottom half

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

$R(1) \otimes Q(1)$: Adds 1st row of bottom half to 1st row of top half

On right: row \rightarrow column, top \rightarrow left, bottom \rightarrow right

1. Put 1 in upper left corner of A using SWAP, H.
2. Use CNOT(1,i) to make rest of 1st column of A = 0
3. Use CNOT on right to eliminate 1st row of A
4. Use $R(1) \otimes \text{Control}(1,1)$ to make 1st column of C = 0 $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
5. 1st row of C is now 0 too! Columns on left commute $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
- ($a_i | \bar{c}_j \rangle - (a_j | \bar{c}_i \rangle) = c_{ij} = 0$)
6. Repeat 1-5 with 2nd row & column, etc. to get $A = I, C = 0$
7. Use $Q(\text{Control}(1,1))$ on R

$\text{CNOT}_{\text{Controlled}-Z}(i,j)$: Adds i th row of top half to j th row of bottom half
 इसका अर्थ है कि यह टॉप हाफ की i वीं रोड को बॉटम हाफ की j वीं रोड में जोड़ देता है।

$$\begin{pmatrix} \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \\ \vdots & \vdots \\ \alpha_n & \beta_n \end{pmatrix}$$

$\text{-HSEU} = Q(i)$: Adds i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.
2. Use CNOT($1,i$) to make rest of 1st column of A = 0
3. Use CNOT on right to eliminate 1st row of A
4. Use R(1) & $\text{CNOT}_{\text{Controlled}-Z}(1,:)$ to make 1st column of C = 0
5. 1st row of C is now 0 too! Columns on the right to C = $\begin{pmatrix} 1 & 0 \\ 0 & \ddots \\ 0 & 0 \end{pmatrix}$
- ($a_i | \bar{c}_i$) - ($\bar{a}_i | \bar{c}_i$) = $c_{1i} = 0$
6. Repeat 1-5 with 2nd row & column, etc. to get $A = I$
7. Use $Q^{\text{Controlled}-Z}$ on R to make $B = 0$
8. This forces $D = I$: $(\bar{a}_j | \bar{d}_j) / (\bar{b}_j | \bar{d}_j) =$

$\text{CNOT}_{\text{Control}}(j,i)$: Adds i th row of top half to j th row of bottom half
 $\text{CNOT}_{\text{Control}}(i,j)$: Adds j th row of top half to i th row of bottom half

$$\begin{pmatrix} \bar{a}_1 & \bar{c}_2 \\ \bar{a}_2 & \bar{c}_1 \end{pmatrix}$$

$\text{H} \otimes \text{I} = Q(1)$: Adds i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow right, bottom \rightarrow left

1. Put 1 in upper left corner of A using SWAP, H.
2. Use CNOT $(1,i)$ to make rest of 1st column of A = 0
3. Use CNOT on right to eliminate 1st row of A
4. Use R $(1) \otimes \text{Control}(j-i)$ to make 1st column of C = 0 $A = \begin{pmatrix} 1 & * & * \\ 0 & * & * \end{pmatrix}$
5. 1st row of C is now 0 too! Columns on left commute $C = \begin{pmatrix} 1 & 0 \\ 0 & * \end{pmatrix}$
6. Repeat 1-5 with 2nd row & column, etc. to get $A = I, C = 0$
7. Use $Q \otimes \text{Control}(j-i)$ on R to make $B = 0$
8. This forces $D = I$: $(\bar{a}_i | \bar{c}_j) \otimes (\bar{b}_j | \bar{d}_i) = \delta_{ij} = d_{ij} \Rightarrow U = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$

CNOT(i, j): Add the i th row in top half to j th row in top half
 Add the j th row in bottom half to i th row in bottom half
 H(i): Switches i th row from top half with i th row in bottom half
 R(i): Adds i th row from top half to i th row in bottom half
 $\begin{array}{c} \text{I} \\ \text{II} \\ \text{III} \end{array} = \text{SWAP}(i, j)$: swaps i, j th rows in both top/bottom half
 $\begin{array}{c} \text{I} \\ \text{II} \\ \text{III} \end{array} - \text{Control}-\text{CNOT}(i, j)$: Adds i th row of top half to j th row of bottom half
 $\begin{array}{c} \text{I} \\ \text{II} \\ \text{III} \end{array} - \text{Control}-\text{CNOT}(j, i)$: Adds j th row of top half to i th row of bottom half
 $\underline{-H \otimes I \otimes I - Q(1)}$: Adds i th row of bottom half to i th of top half

On right: row \rightarrow column, top \rightarrow left, bottom right

1. Put 1 in upper left corner of A using SWAP, H.
2. Use CNOT($1, i$) to make rest of 1st column of A = 0
3. Use CNOT on ~~right~~ to eliminate 1st row of A
4. Use R(1) & Control-CNOT($1, i$) to make 1st column of C = 0 $A = \begin{pmatrix} 1 & * & * \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$, $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{pmatrix}$
5. 1st row of C is now 0 too! Columns on ~~left~~ commute $C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & * \end{pmatrix}$
6. Repeat 1-5 with 2nd row/column/etc. to get $A = I, C = 0$
7. Use $Q(\text{Control}-\text{CNOT})$ on P make $B = 0$
8. This forces $D = I$: $(\bar{B}; |\bar{E}_i) \times (\bar{P}; |\bar{T}_j) = \delta_{ij} = \delta_{ij} \Rightarrow U = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}$

Thm. [Quantum Gilbert-Varshamov bound]:

Suppose n, k, d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j \leq 2^{n-k}$.

Thm. [Quantum Gilbert-Varshamov bound]:

Suppose n, k, d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j \leq 2^{n-k}$.

Then $\exists [n, k, d]$ stabilizer code.



Thm. [Quantum Gilbert-Varshamov bound]:

Suppose n, k, d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j \leq 2^{n-k}$.

Then $\exists [n, k, d]$ stabilizer code.

Note: # Pauli errors of at j:



Thm. [Quantum Gilbert-Varshamov bound]:

Suppose n, k, d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j \leq 2^{n-k}$.

Then $\exists [n, k, d]$ stabilizer code.

Note: # Pauli errors of wt $j = \binom{n}{j} 3^j$



Proof:

Proof: We make a list of all $[n,k]$ codes,
cross off any code



Proof: We make a list of all $[n,k]$ codes,
cross off any code with distance $< d$.

There are $E_d = \sum_{j=0}^d \binom{n}{j} 3^j$ errors of weight $< d$.

Proof: We make a list of all $[n,k]$ codes,
cross off any code with distance $< d$.

There are $E_{nk} = \sum_{j=0}^k \binom{n}{j} 3^j$ errors of weight $< d$.

Proof: We make a list of all $[n,k]$ codes,
cross off any code with distance $< d$.

There are $E_d = \sum_{j=0}^d \binom{n}{j} 3^j$ errors of weight $< d$. Cross off
all codes for E ($\exists i : E_i < d$) is in $N(s) \setminus S$.

How many codes can we cross off for E ?

Proof: We make a list of all $[n,k]$ codes,
cross off any code with distance $< d$.

There are $E_d = \sum_{j=0}^{d-1} \binom{n}{j} 3^j$ errors of weight $< d$. Cross off
all codes for E_d ($\neq E < d$) is in $N(s) \setminus S$.

How many codes can we cross off for E ? All E s have same #
of codes for which they are in $N(s) \setminus S$

Proof: We make a list of all $[[n,k]]$ codes,
cross off any code with distance $\leq d$.

There are $E = \sum_{j=0}^d \binom{n}{j} 3^j$ errors of weight $\leq d$. Cross off
all codes for E (if $E \leq d$) is in $N(S) \setminus S$.

How many codes can we cross off for E ? All E s have same #
of codes for which they are in $N(S) \setminus S$, regardless of wt E (by Clifford sp.)

Proof: We make a list of all $[[n,k]]$ codes,
cross off any code with distance $< d$.

There are $E_d = \sum_{j=0}^d \binom{n}{j} 3^j$ errors of weight $< d$. Cross off
all codes for $E_d + E < d$ is in $N(s) \setminus S$.

How many codes can we cross off for E ? All E s have same #
of codes for which they are in $N(s) \setminus S$, regardless of wt E (by Clifford sp.)

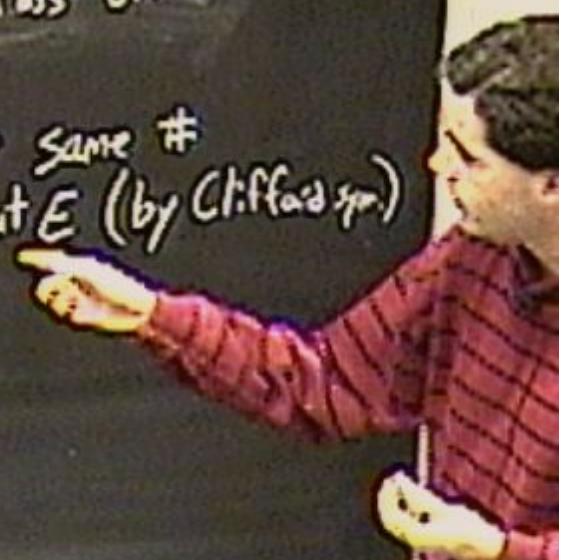
$N_{n,k}$ codes, each code has

Proof: We make a list of all $[\![n,k]\!]$ codes,
cross off any code with distance $< d$.

There are $E_d = \sum_{j=0}^d \binom{n}{j} 3^j$ errors of weight $< d$. Cross off
all codes for E ($\forall E < d$) is in $N(S) \setminus S$.

How many codes can we cross off for E ? All E s have same #
of codes for which they are in $N(S) \setminus S$, regardless of wt E (by Clifford's p.)

$N_{n,k}$ codes, each code has $|N(S) \setminus S| = 2^{n-k} - 2^{n-k}$



Proof: We make a list of all $[(n,k)]$ codes,
cross off any code with distance $< d$.

There are $E_d = \sum_{j=0}^d \binom{n}{j} 3^j$ errors of weight $< d$. Cross off
all codes for E_d (at $E < d$) is in $N(S) \setminus S$.

How many codes can we cross off for E ? All E_s have same #
of codes for which they are in $N(S) \setminus S$, regardless of wt E (by Clifford's p.)

$N_{n,k}$ codes, each code has $|N(S) \setminus S| = 2^{n-k} - 2^{n-k}$
 $\Rightarrow E_{n,k} = N_{n,k}(2^{n-k} - 2^{n-k})$

$N_{n,k}$ codes, each appears k times
End errors, each appears k times

Proof: We make a list of all $[(n,k)]$ codes,
cross off any code with distance $< d$.

There are $E_d = \sum_{j=0}^d \binom{n}{j} 3^j$ errors of weight $< d$. Cross off
all codes for E ($\forall E < d$) is in $N(S) \setminus S$.

How many codes can we cross off for E ? All E 's have same #
of codes for which they are in $N(S) \setminus S$, regardless of $\forall E$ (by Clifford grp.)
 $N_{n,k}$ codes, each code has $|N(S) \setminus S| = 2^{n-k} - 2^{n-k}$
 4^{n-1} errors, each appears K times

There are $E_{n,k} = \sum_{j=0}^{n-k} \binom{n}{j} 3^j$ errors of weight $< d$. Cross off all codes for $E_{n,k} < E$ is in $N(S) \setminus S$.

How many codes can we cross off for E ? All E s have same # of codes for which they are in $N(S) \setminus S$, regardless of $\text{wt } E$ (by Clifford's ^{parity})

$$N_{n,k} \text{ codes, each code has } |N(S) \setminus S| = 2^{n-k} - 2^{n-k} \Rightarrow (q-1)K = N_{n,k}(2^{n-k} - 2^{n-k})$$

4^{n-1} errors, each appears K times

$$K = N_{n,k} \left(\frac{2^{n-k} - 2^{n-k}}{4^{n-1}} \right) \leq N_{n,k} \frac{2^{n-k}}{4^n} = \frac{N_{n,k}}{2^{n-k}}$$

Can at most cross off
End K codes

How many codes can we cross off for E ? All E s have same #
of codes for which they are in $N(E) \setminus S$, regardless of w/ E (by Clifford sp.)

$$N_{n,k} \text{ codes, each code has } |N(E) \setminus S| = 2^{n-k} - 2^{n-k}$$

q^{n-1} errors, each appears k times

$$K = N_{n,k} \left(\frac{2^{n-k} - 2^{n-k}}{q^{n-1}} \right) \leq N_{n,k} \frac{2^{n-k}}{q^n} = \frac{N_{n,k}}{2^{n-k}}$$

Can at most cross off
End K codes $\leq \frac{q^n}{2^{n-k}} N_{n,k}$
If $< N_{n,k}$, a code is left!

Thm. [Quantum Gilbert-Varshamov bound]:

Suppose n, k, d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j < 2^{n-k}$.

Then $\exists [n, k, d]$ stabilizer code.

Note: # Pauli errors of wt $j = \binom{n}{j} 3^j$

Note: # rank errors of $\text{wt}(j) = k$

all coders for $E(\oplus E^{\perp d})$ is in $N(S)^{\perp}$.
How many coders can we cross off for E ? All E s have same #
of coders for which they are in $N(S)$, regardless of $\text{wt } E$ (by Clifford's)
 $N_{n,k}$ coders, each code has $|N(S)| = 2^n - 2^{n-k}$ $\Rightarrow (q-1)K = N_{n,k}(2^n - 2^{n-k})$
 4^{n-1} errors, each appears K times
 $K = \frac{N_{n,k}(2^n - 2^{n-k})}{4^{n-1}} \leq N_{n,k} \frac{2^n}{4^{n-1}} = \frac{N_{n,k}}{2^{n-k}}$ Can at most cross off
End K codes & $\leq N_{n,k}$
If $< N_{n,k}$, a code is left!

Proof: We make a list of all $[(n,k)]$ codes,
cross off any code with distance $< d$.

The [Quantum] Gilbert-Varshamov bound:

Suppose n, k, d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j < 2^{n-k}$.

Then $\exists [n,k,d]$ stabilizer code.

Note: # Pauli errors of $\pm j = \binom{j}{0} 3^j$

Proof: We make a list of all $[n,k]$ codes,
cross off any code with distance $< d$.

Thm. [Quantum Gilbert-Varshamov bound]:

Suppose n, k, d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j < 2^{n-k}$.

Then $\exists [n,k,d]$ stabilizer code.

Note: # Pauli errors of $|t\rangle^{\otimes j} = \binom{n}{j} 3^j$

Pauli errors of $|t\rangle^{\otimes j} = \binom{n}{j} 3^j$ is constant (fixed)

Cor: Let $n \rightarrow \infty, k \rightarrow \infty, d \rightarrow \infty, \frac{k}{n} = R$ as $\frac{d}{n} \rightarrow$ constant

$$\binom{n}{j} \rightarrow 2^{n h(\frac{j}{n})} \quad h(x) = -x \log_2 x - (1-x) \log_2(1-x)$$

Proof: We make a list of all $[\![n,k]\!]$ codes

Thm. [Quantum Gilbert-Varshamov bound]:

Suppose n, k, d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j < 2^{n-k}$.

Then $\exists [\![n,k,d]\!]$ stabilizer code.

Note: # Pauli errors of wt $j = \binom{n}{j} 3^j$

Cor.: Let $n \rightarrow \infty$, $k \rightarrow \infty$, $d \rightarrow \infty$, $\frac{k}{n} = R$ as $\frac{d}{n}$ constant (fixed d/k)

$$\binom{n}{j} \rightarrow 2^{nh(\frac{j}{n})} \quad h(x) = -x \log_2 x - (1-x) \log_2 (1-x)$$

$$\log \sum \binom{n}{j} 3^j \leq nh\left(\frac{d}{n}\right) \quad \text{If } R < 1 - h\left(\frac{d}{n}\right) - \frac{d}{n} \log 3$$

$n-k > nh\left(\frac{d}{n}\right) + d \log 3 \Leftrightarrow$ A code exists with $\frac{d}{n}, \frac{k}{n} = R$

Cor.: Let $n \rightarrow \infty$, $k \rightarrow \infty$, $d \rightarrow \infty$, $\frac{k}{n} = R$ and $\frac{d}{n}$ constant (good code)

$$(\text{?}) \rightarrow \sum_{x=0}^{nh(\frac{d}{n})} h(x) = -x \log_2 x - (1-x) \log_2 (1-x)$$

$$\log \sum_{j=0}^{n-k} \binom{n}{j} 3^j \approx nh\left(\frac{d}{n}\right) \quad \text{If } R < 1 - h\left(\frac{d}{n}\right) - \frac{d}{n} \lg 3$$

$$n-k > nh\left(\frac{d}{n}\right) + d \lg 3 \Leftrightarrow R < 1 - h\left(\frac{d}{n}\right) - \frac{d}{n} \lg 3$$

A code exists with $\frac{d}{n}, \frac{k}{n} = R$

Proof: We make a list of all $[n, k]$ codes, cross off any code with distance $< d$.

There are $E_d = \sum_{j=0}^{d-1} \binom{n}{j} 3^j$ errors of weight $< d$. Cross off all codes for $E \cap E_d$ is in $N(S) \setminus S$.

How many codes can we cross off for E ? All E s have same # of codes for which they are in $N(S) \setminus S$, regardless of wt E (by Clifford's sp.)

$N_{n,k}$ codes, each code has $|N(S) \setminus S| = 2^{n-k} - 2^{n-k}$

4^{n-1} errors, each appears K times

$$K = N_{n,k} \left(\frac{2^{n-k} - 2^{n-k}}{4^{n-1}} \right) \leq N_{n,k} \frac{2^{n-k}}{4^{n-1}} = \frac{N_{n,k}}{2^{n-k}}$$

Can at most cross off $\frac{E_d}{2^{n-k}} N_{n,k}$ End K codes $\leq \frac{E_d}{2^{n-k}} N_{n,k}$

If $K < N_{n,k}$, a code is left!

Thm. [quantum Hamming bound]:



Thm. [quantum Hamming bound]: An $(n, 2^k, d)$



Thm. [quantum Hamming bound]: An $((n, 2^k, d))$ non-degenerate code must satisfy $\left[\sum_{j=0}^{t-1} \binom{n}{j} 3^j \right] 2^k \leq 2^n \quad (d = 2t+1)$

Proof: For a non-degenerate code, different errors produce linearly independent states $\Rightarrow (\# \text{ errors})(\# \text{ basis codewords}) \leq \dim \text{Hilbert space}$.

Thm. [quantum Hamming bound]: An $((n, 2^k, d))$ non-degenerate code must satisfy $\left[\sum_{j=0}^t \binom{n}{j} 2^j \right] 2^k \leq 2^n \quad (d = 2t+1)$

Proof: For a non-degenerate code, different errors are linearly independent states $\Rightarrow (\# \text{ errors})(\# \text{ basis codewords}) \leq \# \text{ states}$.

Asymptotic limit: $\frac{d}{n} = P$

Thm. [quantum Hamming bound]: An $((n, \sum^k, d))$ non-degenerate code must satisfy $\left[\sum_{j=0}^t \binom{n}{j} 3^j \right] 3^k \leq 3^n \quad (d = 2t+1)$

Proof: For a non-degenerate code, different errors produce linearly independent states $\Rightarrow (\# \text{ errors})(\# \text{ basis codewords}) \leq d$ in Hilbert space.

$$\frac{t}{n} = p \quad R < 1 - h(p) - p \log_3$$

Asymptotic

Thm:

Thm. (quantum Hamming bound): An $((n, \sum^k, d))$ non-degenerate code must satisfy $\left[\sum_{j=0}^t \binom{n}{j} 3^j \right] 3^k \leq 3^n \quad (d=2t+1)$

Proof: For a non-degenerate code, different errors produce linearly independent states $\Rightarrow (\# \text{ errors})(\# \text{ basis codewords}) \leq d$ in Hilbert space.

Asymptotic limit: $\frac{k}{n} = p \quad R < 1 - h(p) - p \log 3$

Thm. (

Thm. [quantum Hamming bound]: An $((n, \sum^k, d))$ non-degenerate code must satisfy $\left[\sum_{j=0}^t \binom{n}{j} 3^j \right] 3^k \leq 3^n \quad (d=2t+1)$

Proof: For a non-degenerate code, different errors produce linearly independent states $\Rightarrow (\# \text{ errors})(\# \text{ basis codewords}) \leq \dim \text{ Hilbert space}$.

$$\text{Asymptotic limit: } \frac{k}{n} = p \quad R < 1 - h(p) - p \log 3$$

Thm. [quantum Singleton bound] (Kull-Lalanne bound): $n-k \geq \lceil d-1 \rceil$ (Note: this applies to non-binary registers)

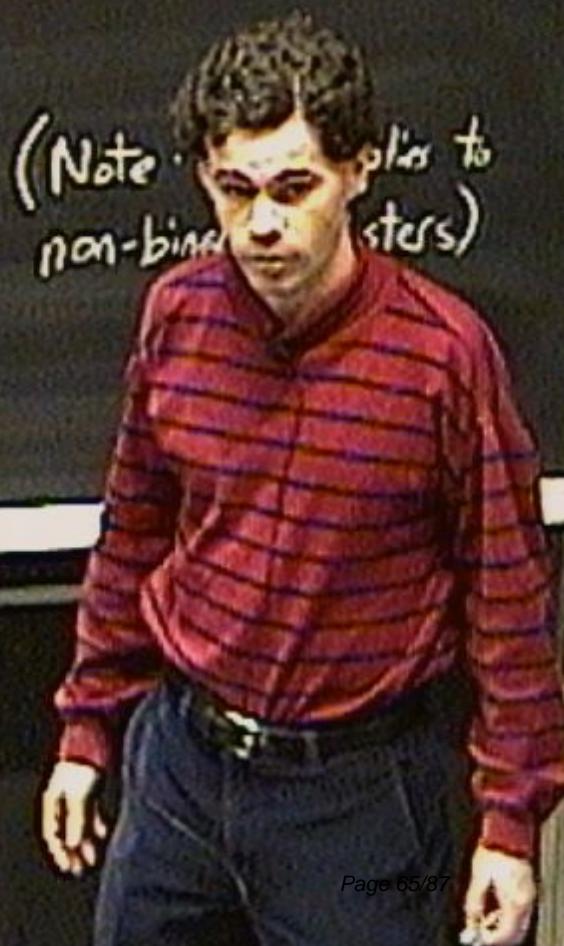
Thm. [quantum Hamming bound]: An (n, \sum^k, d) non-degenerate code must satisfy $\left[\sum_{j=0}^t \binom{n}{j} 3^j \right] 3^k \leq 3^n \quad (d=2t+1)$

Proof: For a non-degenerate code, different errors produce linearly independent states $\Rightarrow (\# \text{ errors})(\# \text{ basis codewords}) \leq \dim \text{ Hilbert space}$.

$$\text{Asymptotic limit: } \frac{k}{n} = p \quad R < 1 - h(p) - p \log_3$$

Thm. [quantum Singleton bound] (Foll-Lafleur bound): $n-k \geq 2(d-1)$

Note: Saturated by $([5, 1, 3])$ and $([4, 2, 2])$



(Note: applies to non-binary states)

Thm. [quantum Hamming bound]: An (n, \sum^k, d) non-degenerate code must satisfy $\left[\sum_{j=0}^t \binom{n}{j} 3^j \right] 3^k \leq 3^n \quad (d=2t+1)$

Proof: For a non-degenerate code, different errors produce linearly independent states $\Rightarrow (\# \text{ errors})(\# \text{ basis codewords}) \leq \dim \text{Hilbert space}$.

$$\text{Asymptotic limit: } \frac{k}{n} = p \quad R < 1 - h(p) - p \log 3$$

Thm. [quantum Singleton bound]: $n-k \geq 2(d-1)$ (Note: this applies to non-binary registers)
 (Kull-Laflanne bound)

Note: Saturated by $[[5, 1, 3]]$ and $[[4, 2, 2]]$ ($\ell[[2n, 2n-2, 2]]$)

Note: Saturated by $([-5, 1, 3])$ and $(0, 7) \rightarrow (800, 1, 3)$

Proof: $k=1$ case

Note: Saturated by $([5,1,3])$ and $(0,7) \rightarrow (8000, \dots)$

Proof: $k=1$ case
Distance d code can correct $d-1$ erasures

Note: Saturated by $([5,1,3])$ and $(0,9) \rightarrow (800,1,1)$

Proof: $k=1$ case

Distance d code can correct $d-1$ erasures

نیز

Note: Saturated by $([5,1,3])$ and $(6,7) \rightarrow (8,6,1,1)$

Proof: $k=1$ case

Distance d code can correct $d-1$ erasures



Note: Saturated by $([5,1,3])$ and $(0,7 \rightarrow (800,1,1,1))$

Proof: $k=1$ case

Distance d code can correct $d-1$ erasures



Given $n-(d-1)$ parities, can recover state
If $d-1 \geq n-(d-1)$, we get 2 copies -

Note: Saturated by $([5,1,3])$ and $([6,7,3]) \rightarrow ([8,6,3])$

Proof: $k=1$ case

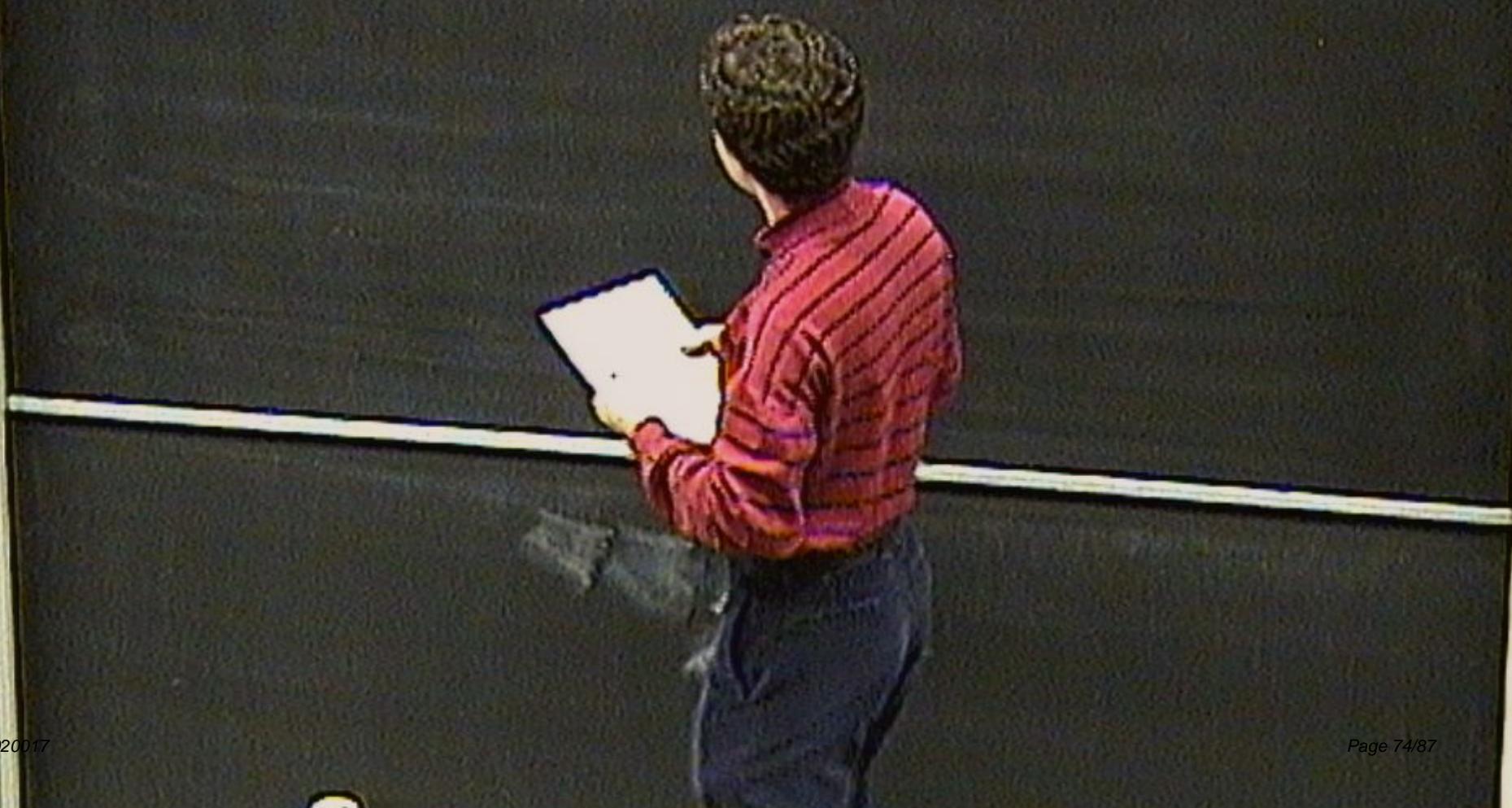
Distance d code can correct $d-1$ erasures



Given $n-(d-1)$ parities, can recover state
If $d-1 \geq n-(d-1)$, we get 2 copies - violates uniqueness
 $n-1 \geq 2(d-1)$

Von Neumann entropy: $S(\rho) = -\text{tr } \rho \log \rho$

Von Neumann Entropy: $S(\rho) = -\text{tr } \rho \log \rho$
 $S(|\psi\rangle\langle\psi|) = 0$



Von Neumann entropy: $S(\rho) = -\text{tr } \rho \log \rho$

$$S(|\psi\rangle\langle\psi|) = 0, \quad \rho_{AB} = |\psi\rangle\langle\psi| \Rightarrow S(\rho_A) = S(\rho_B)$$

Von Neumann entropy: $S(\rho) = -\text{tr } \rho \log \rho$

$$S(|\psi\rangle\langle\psi|) = 0 \quad \rightarrow \quad \rho_{AB} = |\psi\rangle\langle\psi| \Rightarrow S(\rho_A) = S(\rho_B)$$

$$S(\rho) \leq \log \text{dimension (i.e. \# qubits)} \quad = \text{ if } \rho = I/d^n$$

Von Neumann entropy: $S(\rho) = -\text{tr } \rho \log \rho$

$$S(|\psi\rangle\langle\psi|) = 0 \rightarrow \rho_{AB} = |\psi\rangle\langle\psi| \Rightarrow S(\rho_A) = S(\rho_B)$$

$$S(\rho) \leq \log \text{dimension (i.e. # qubits)} = \text{if } \rho = I/d^n.$$

Subadditivity $S(\rho_{AB}) \leq S(\rho_A) + S(\rho_B)$

Von Neumann entropy: $S(\rho) = -\text{tr } \rho \log \rho$

$$S(|\psi\rangle\langle\psi|) = 0 \quad \rightarrow \quad \rho_{AB} = |\psi\rangle\langle\psi| \Rightarrow S(\rho_A) = S(\rho_B)$$

$$S(\rho) \leq \log \text{dimension (i.e. # qubits)} \quad = \text{ if } \rho = I/d^n$$

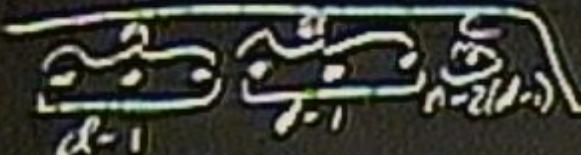
Subadditivity $S(\rho_{AB}) \leq S(\rho_A) + S(\rho_B)$

Strong subadditivity $S(\rho_{ABC}) + S(\rho_B) \leq S(\rho_{AB}) + S(\rho_{BC})$

Proof: $k=1$ case

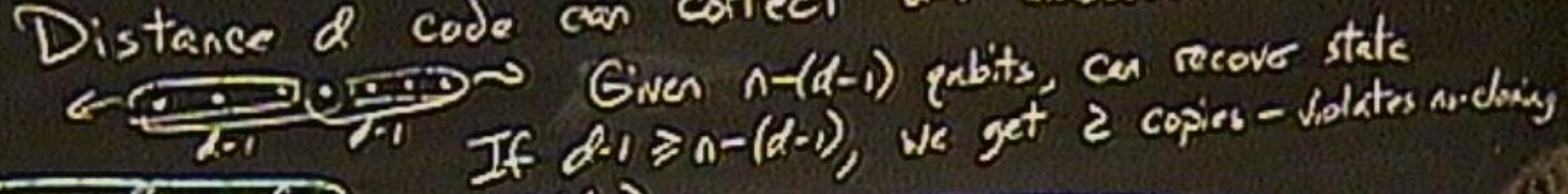
Distance d code can correct $d-1$ erasures

 Given $n-(d-1)$ qubits, can recover state
If $d-1 \geq n-(d-1)$ get 2 copies - violates no-cloning

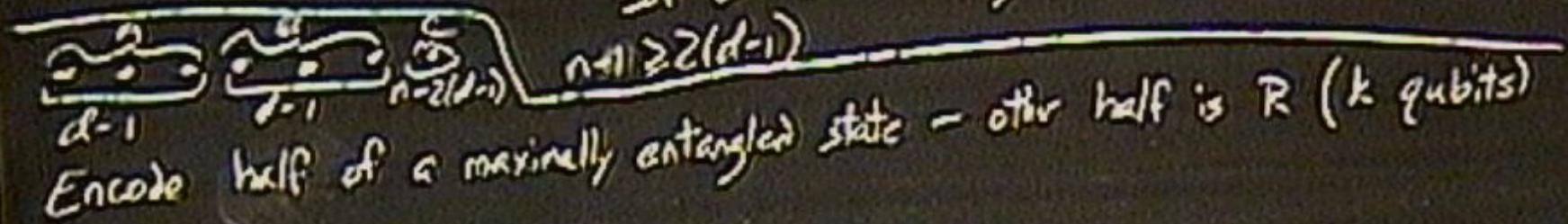
 $n-1 \geq 2(d-1)$

Proof: $k=1$ case

Distance d code can correct $d-1$ erasures

 Given $n-(d-1)$ qubits, can recover state

If $d-1 \geq n-(d-1)$, we get 2 copies - violates no-cloning

 $n-1 \geq 2(d-1)$

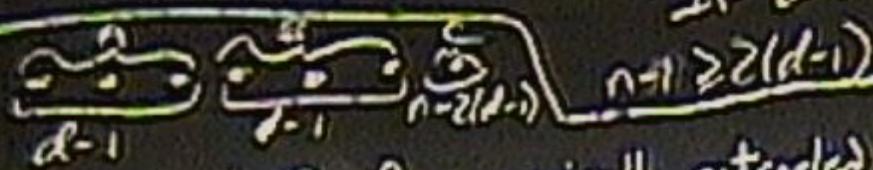
Encode half of a maximally entangled state - other half is R (k qubits)

Distance d code can correct



Given $n-(d-1)$ qubits, can recover state

If $d-1 \geq n-(d-1)$, we get 2 copies - violates no-cloning



Encode half of a maximally entangled state - other half is R (k qubits)

Global pure state $\Rightarrow S(RA) = S(BC)$, $S(RB) = S(AC)$.

R is completely mixed $\Rightarrow S(R) = k$



Von Neumann entropy: $S(\rho) = -\text{tr } \rho \log \rho$

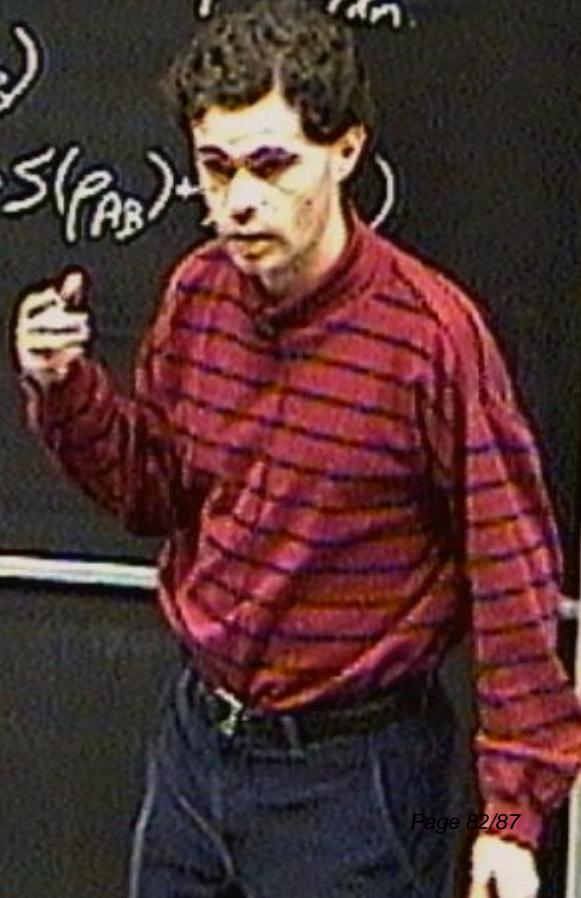
$$S(|\psi\rangle\langle\psi|) = 0 \quad \rightarrow \quad \rho_{AB} = |\psi\rangle\langle\psi| \Rightarrow S(\rho_A) = S(\rho_B)$$

$S(\rho) \leq \log \text{dimension}$ (i.e. # qubits) = if $\rho = I/m$.

Subadditivity $S(\rho_{AB}) \leq S(\rho_A) + S(\rho_B)$

Strong subadditivity $S(\rho_{ABC}) + S(\rho_B) \leq S(\rho_{AB}) + S(\rho_C)$

If $\rho_{AB} = \rho_A \otimes \rho_B$, then $S(\rho_{AB}) = S(\rho_A) + S(\rho_B)$



Von Neumann entropy: $S(\rho) = -\text{tr } \rho \log \rho$

$$S(|\psi\rangle\langle\psi|) = 0 \quad , \quad \rho_{AB} = |\psi\rangle\langle\psi| \Rightarrow S(\rho_A) = S(\rho_B)$$

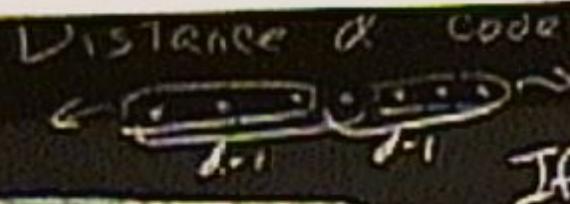
$S(\rho) \leq \log \text{dimension}$ (i.e. # qubits) = if $\rho = I/m$.

Subadditivity $S(\rho_{AB}) \leq S(\rho_A) + S(\rho_B)$

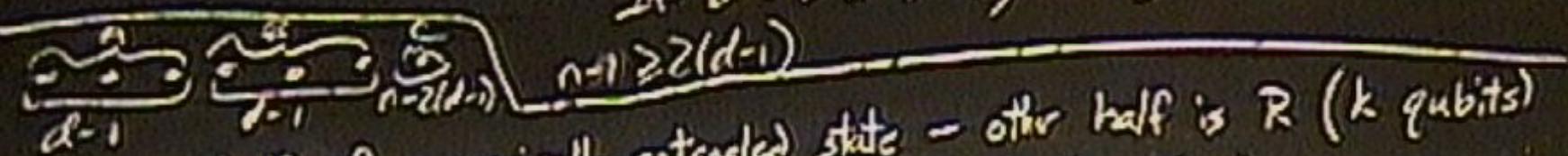
Strong subadditivity $S(\rho_{ABC}) + S(\rho_B) \leq S(\rho_{AB}) + S(\rho_{BC})$

If $\rho_{AB} = \rho_A \otimes \rho_B$, then $S(\rho_{AB}) = S(\rho_A) + S(\rho_B)$

Distance of code can be



Given $n-(d-1)$ qubits, can receive state
If $d-1 \geq n-(d-1)$, we get ≥ 2 copies - violates no-cloning



$n-1 \geq 2(d-1)$

Encode half of a maximally entangled state - other half is R (k qubits)

Global pure state $\Rightarrow S(RA) = S(BC)$, $S(RB) = S(AC)$.

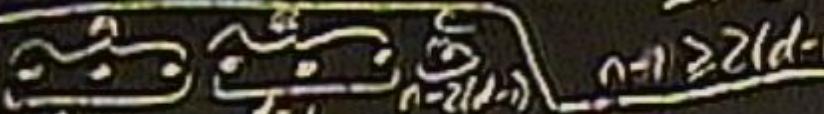
R is completely mixed $\Rightarrow S(R) = k$

Can correct $d-1$ erasures $\Rightarrow A \otimes R$ are tensor product $S(AR) = S(A) + S(R)$

Distance of code can write



Given $n-(d-1)$ qubits, can receive state
If $d-1 \geq n-(d-1)$, we get ≥ 2 copies - violates no-cloning



$n-1 \geq 2(d-1)$

Encode half of a maximally entangled state - other half is R (k qubits)

Global pure state $\Rightarrow S(RA) = S(BC)$, $S(RB) = S(AC)$.

R is completely mixed $\Rightarrow S(R) = k$

R is tensor product $S(AR) = S(A) + S(R)$

Can correct $d-1$ erasures $\Rightarrow A \otimes R$ are tensor product $S(FA) = S(BC) \leq S(B) + S(C)$

By subadditivity, $S(A) + S(R) = S(FA) = S(BC) \leq S(B) + S(C)$

Distance of code considered

Given $n-(d-1)$ qubits, can receive state
If $d-1 \geq n-(d-1)$, we get ≥ 2 copies - violates no-cloning

$n-1 \geq 2(d-1)$

Encode half of a maximally entangled state - other half is R (k qubits)

Global pure state $\Rightarrow S(RA) = S(BC)$, $S(RB) = S(AC)$.

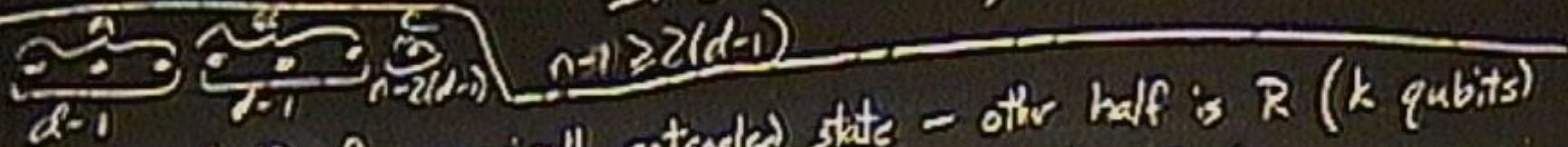
R is completely mixed $\Rightarrow S(R) = k$

Can correct $d-1$ erasures $\Rightarrow A \in R$ are tensor product $S(AR) = S(A) + S(R)$

By subadditivity, $S(A) + S(R) = S(AR) = S(BC) \leq S(B) + S(C) \Rightarrow k \leq S(C) \cdot [S(B) - S(A)]$

Similarly, $k \leq S(C) \cdot [S(A) - S(B)]$

Distance of code can't be more than $n-(d-1)$. Given $n-(d-1)$ qubits, can receive state
If $d-1 \geq n-(d-1)$, we get \geq copies - violates no-cloning

 $n-1 \geq 2(d-1)$

Encode half of a maximally entangled state - other half is R (k qubits)

Global pure state $\Rightarrow S(RA) = S(BC)$, $S(RB) = S(AC)$.

R is completely mixed $\Rightarrow S(R) = k$

Can correct $d-1$ erasures $\Rightarrow A \& R$ are tensor product $S(AR) = S(A) + S(R)$

By subadditivity, $S(A) + S(R) = S(FA) = S(BC) \leq S(B) + S(C) \Rightarrow k \leq S(C) \cdot [S(B) - S(A)]$

Similarly, $k \leq S(C) \cdot [S(A) - S(B)] \Rightarrow k \leq S(C) \leq n-2(d-1)$