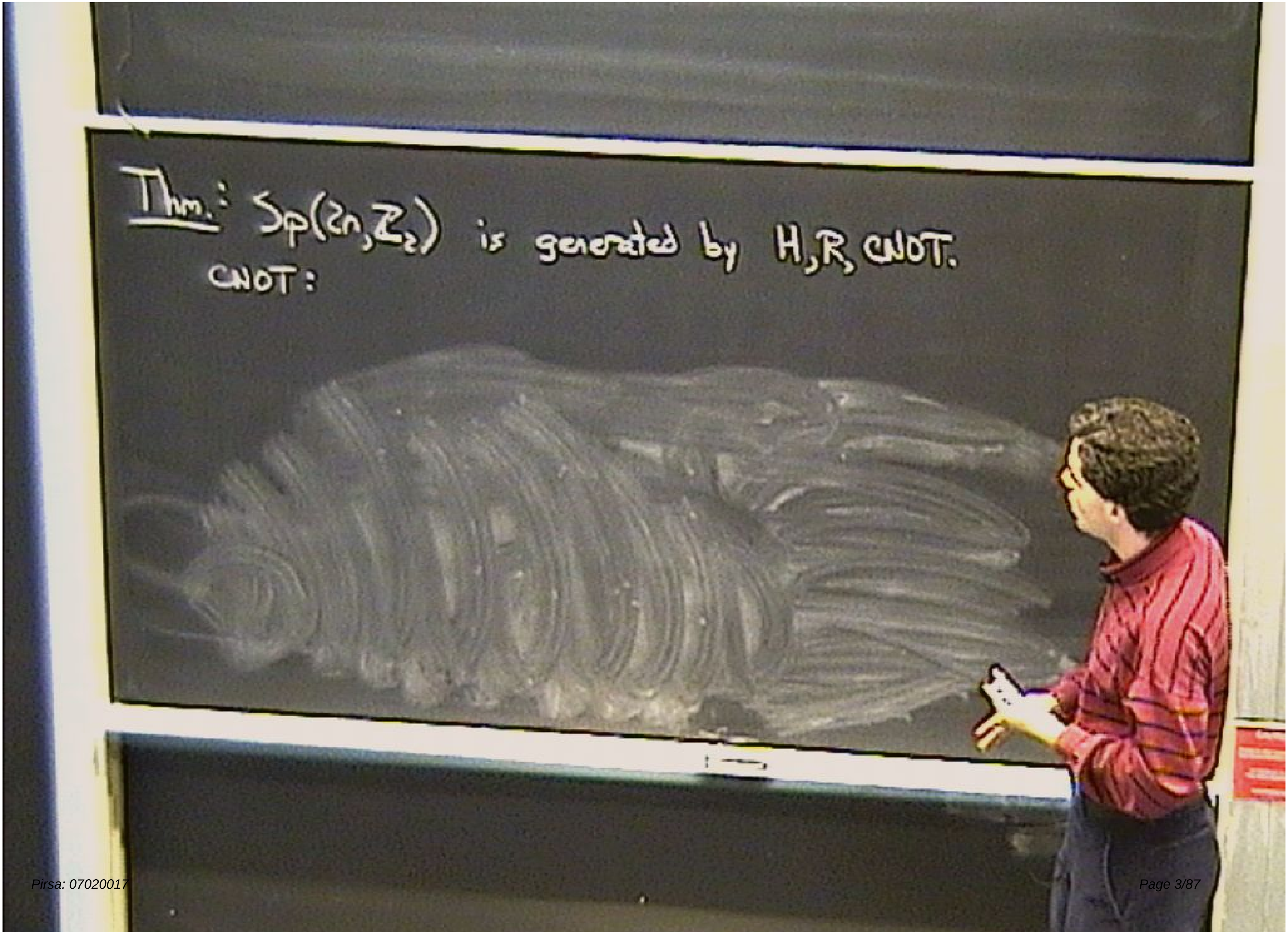
Title: Quantum Error Correction 5A

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.

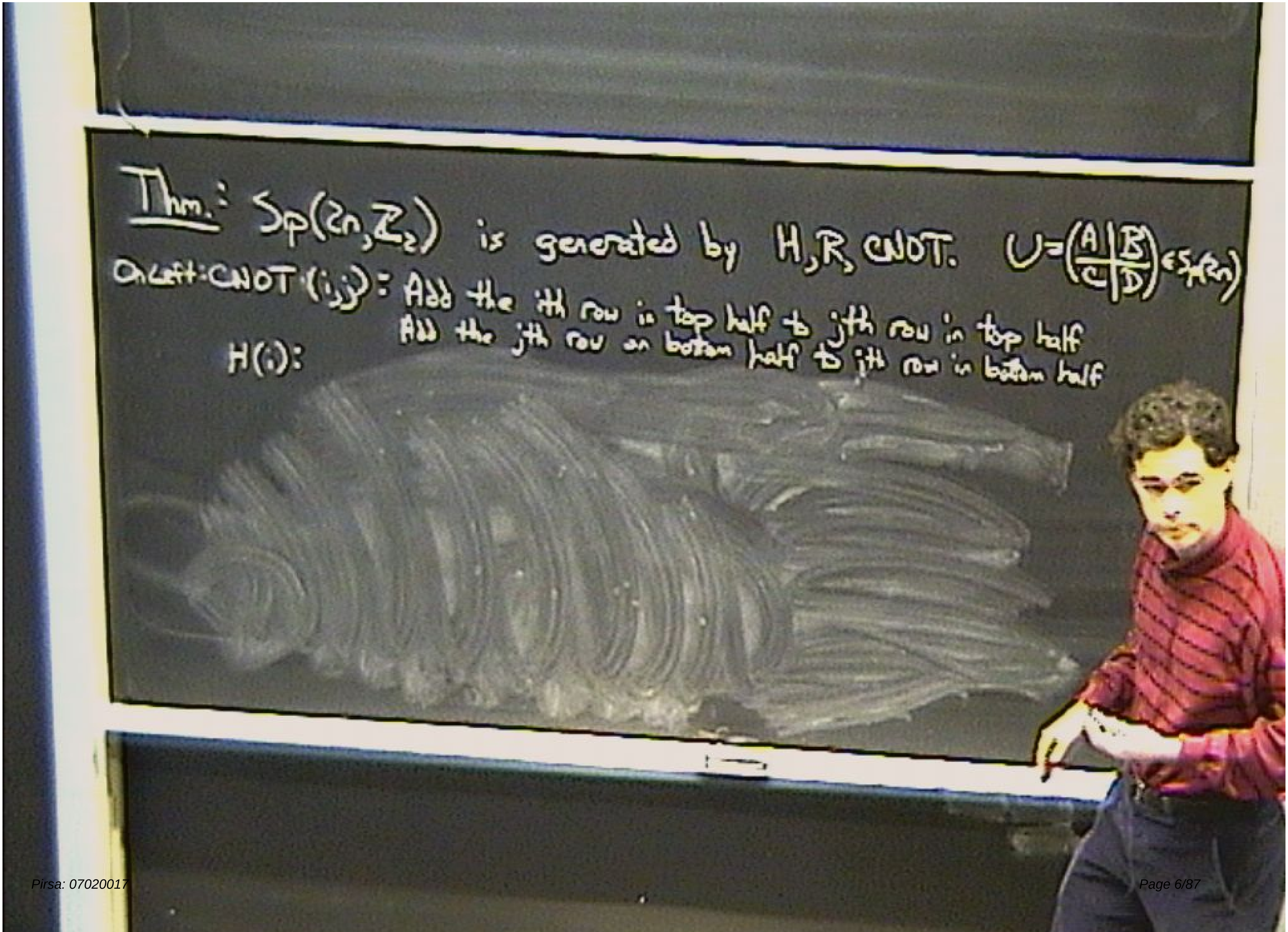

Thm.: $Sp(2n, \mathbb{Z}_2)$ is generated by H, R, CNOT.
CNOT:

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

CNOT:

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

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


Thm.: Sp(2n, Z2) is generated by H, R, CNOT. U = (A|B / C|D) ∈ Sp(2n)
On Left: CNOT(i,j): Add the ith row in top half to jth row in top half
Add the jth row on bottom half to ith row in bottom half
H(i):

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

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

Add the jth row in bottom half to ith row in bottom half

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

$R(i)$:

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

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

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

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

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

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

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

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

= SWAP(i,j): Swaps ith & jth rows in both top & bottom half

Controlled-Z(i,j):

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

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

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

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

= SWAP(i,j): Swaps ith & jth rows in both top & bottom half

Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

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

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

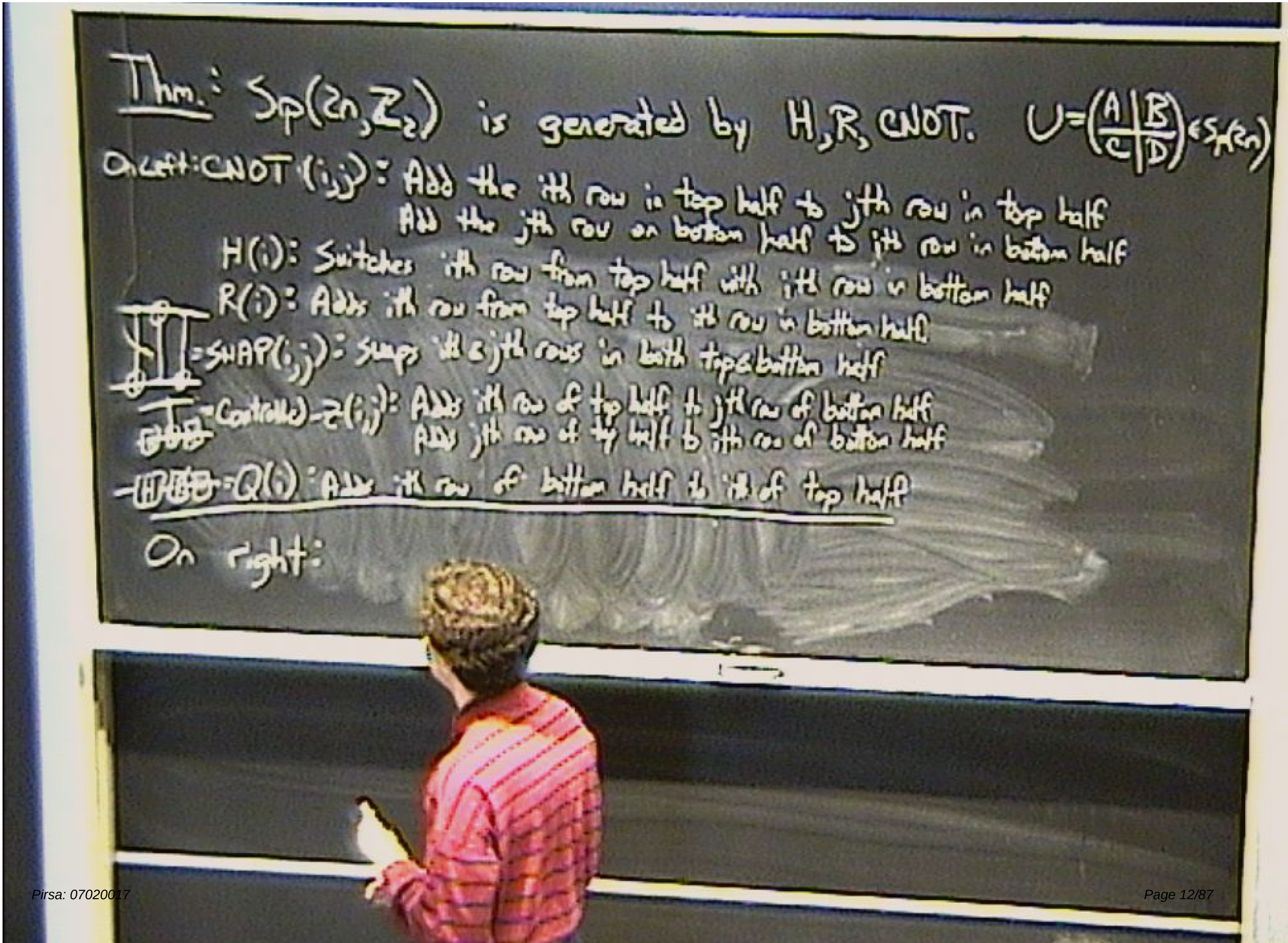
H(i): Switches ith row from top half with ith row in bottom half

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

= SWAP(i,j): Swaps ith & jth rows in both top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

$-H-R-H- = Q(i)$


Thm.: Sp(2n, Z2) is generated by H, R, CNOT. U=(A|B / C|D) ∈ Sp(2n)
On Left: CNOT(i,j): Add the ith row in top half to jth row in top half
Add the jth row on bottom half to ith row in bottom half
H(i): Switches ith row from top half with ith row in bottom half
R(i): Adds ith row from top half to ith row in bottom half
=SWAP(i,j): Swaps ith & jth rows in both top & bottom half
=Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half
=Q(i): Adds ith row of bottom half to ith of top half
On right:

<u>Thm.</u>: $Sp(2n,\mathbb{Z}_2)$ is generated by H, R, CNOT. $U=\left(\begin{array}{c|c}A&B\\\hline C&D\end{array}\right)\in Sp(2n)$

On Left: CNOT(i,j): Add the ith row in top half to jth row in top half
Add the jth row on bottom half to ith row in bottom half
$H(i)$: Switches ith row from top half with ith row in bottom half
$R(i)$: Adds ith row from top half to ith row in bottom half
= SWAP(i,j): Swaps ith & jth rows in both top & bottom half
= Controlled-$Z(i,j)$: Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half
= $Q(i)$: Adds ith row of bottom half to ith of top half

On right: row → column, top → left, bottom → right, opposite direction

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

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

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

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

= SWAP(i,j): Swaps ith & jth rows in both top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= $Q(i)$: Adds ith row of bottom half to ith of top half

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

= SWAP(i,j): swaps [illegible] in both top&bottom half

= Controlled-$Z(i,j)$: Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= $Q(i)$: Adds ith row of bottom half to ith of top half

On right: row $\to$ column, top $\to$ right, bottom $\to$ left

1. Put 1 in upper left corner of A

= SWAP(i,j): Swaps ith & jth rows in each top & bottom half

= Controlled-$Z(i,j)$: Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= $Q(i)$: Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → left

1. Put 1 in upper left corner of A using

= SWAP(i,j): swaps i,j entries in both top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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(i,j): Swaps ith,jth rows in both top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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

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

= SWAP(i,j): Swaps ... entries in each top/bottom half

Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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.

$$A = \underbrace{\begin{pmatrix} 1 & ? \\ 0 & \\ \vdots & \\ 0 & \end{pmatrix}}_{n} \Big\}\, n$$

= SWAP(i,j): Swaps i,j rows in both top&bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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 R(1) & Controlled-Z(1,i) to make 1st column of C = 0

$$A = \underbrace{\left(\begin{array}{c|c} 1 & \\ 0 & ? \\ \vdots & \\ 0 & \end{array} \right)}_{n} \Big\}{n}$$

= SWAP(i,j): Swaps ith, jth rows in each top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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) & Controlled-Z(1,i) to make 1st column of C = 0

$A = \underbrace{\left(\begin{array}{c|c} 1 & ? \\ 0 & \vdots \\ \vdots & \\ 0 & \end{array}\right)}_{n} \Big\} n$

= SWAP(i,j): swaps ith, jth rows in both top & bottom half

= Controlled-$Z(i,j)$: Adds ith row of top half to jth row of bottom half

Adds jth row of top half to ith row of bottom half

= $Q(i)$: Adds ith row of bottom half to ith of top half

On right: row $\to$ column, top $\to$ right, bottom $\to$ 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-$Z(1,i)$ to make 1st column of $C = 0$

$$A = \underbrace{\left(\begin{array}{c|c} 1 & ? \\ 0 & \\ \vdots & \vdots \\ 0 & \end{array}\right)}_{n} \Big\}n$$

= SWAP(i,j): Swaps ith, jth ... of both top & bottom half

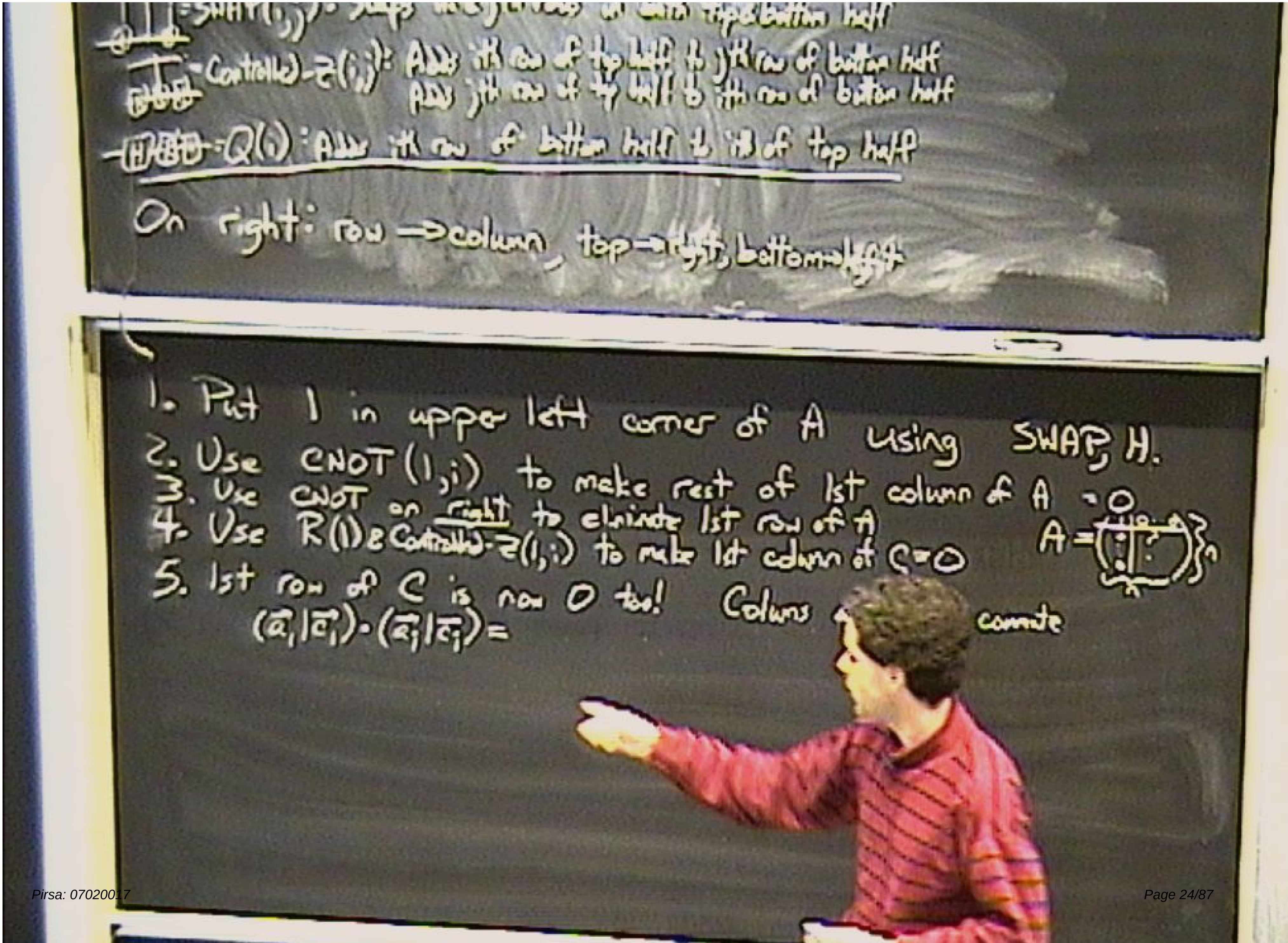
= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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-Z(1,i) to make 1st column of C = 0
5. 1st row of C is now 0 too!

$A = \underbrace{\begin{pmatrix} 1 & 0 \cdots 0 \\ 0 & \\ \vdots & \\ 0 & \end{pmatrix}}_{n} \Big\} n$



Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half
Q(i): Adds ith row of bottom half to ith of top half
On right: row → column, top → right, bottom → 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-Z(1,i) to make 1st column of C = 0
5. 1st row of C is now 0 too!
commute

= SWAP(i,j): swaps ith & jth rows in both top & bottom half

= Controlled-$Z(i,j)$: Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= $Q(i)$: Adds ith row of bottom half to ith 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-$Z(1,i)$ to make 1st column of $C = 0$
5. 1st row of C is now 0 too! Columns on left commute

$$(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_j|\vec{c}_j) = c_{1j} = 0$$

$$A = \underbrace{\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & ? \end{array}\right)}_{n}\Big\}n \qquad C = \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & ? \end{array}\right)$$

= SWAP(i,j): swaps ... in each top/bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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-Z(1,i) to make 1st column of C = 0
5. 1st row of C is now 0 too! Columns on left commute

$(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_j|\vec{c}_j) = c_{1j} = 0$

$A = \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & ? \end{array}\right)\}n$

$C = \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & ? \end{array}\right)\}n$

SWAP(i,j): swaps ith & jth rows in both top & bottom half

Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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-Z(1,i) to make 1st column of C = 0
5. 1st row of C is now 0 too! Columns on left commute

$$(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_i|\vec{c}_i) = c_{1i} = 0$$

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

$A = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & ? \end{pmatrix}}_{n}\Big\}n$

$C = \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & ? \end{pmatrix}}_{n}\Big\}n$

= SWAP(i,j): Swaps ith & jth rows in both top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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-Z(1,i) to make 1st column of C = 0
5. 1st row of C is now 0 too! Columns on left commute

$$(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_j|\vec{c}_j) = c_{1j} = 0$$

$$A = \underbrace{\left(\begin{array}{c|c} 1 & 0 \\ \hline \vdots & \vdots \end{array}\right)}_{n}\Big\}n \qquad C = \underbrace{\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & ? \\ \vdots & \end{array}\right)}_{n}\Big\}n$$

6. Repeat 1-5 with 2nd row & column, etc. to get $A = I$, $C = 0$
7.

= SWAP(i,j): swaps ith & jth rows in both top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → 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-Z(1,i) to make 1st column of C = 0
5. 1st row of C is now 0 too! Columns on left commute

$(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_j|\vec{c}_j) = c_{1j} = 0$

6. Repeat 1-5 with 2nd row & column, etc. to get $A = I$, $C = 0$
7. Use Q & Controlled-Z on R

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

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

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

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

= SWAP(i,j): Swaps ith & jth rows in both top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

$$\left(\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right)$$

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → left

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= $Q(i)$: Adds ith row of bottom half to ith of top half

On right: row $\to$ column, top $\to$ right, bottom $\to$ 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-Z$(1,i)$ to make 1st column of $C = 0$
5. 1st row of C is now 0 too! Columns on left commute
   $(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_j|\vec{c}_j) = c_{1j} = 0$
6. Repeat 1–5 with 2nd row & column, etc. to get $A = I$, $C = 0$
7. Use $Q^{(new)}$ & (Controlled-Z)$^{(new)}$ on R

Controlled-$Z(i,j)$: Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

$-H-$ $= Q(i)$: Adds ith row of bottom half to ith 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-$Z(1,i)$ to make 1st column of $C = 0$
5. 1st row of C is now 0 too! Columns on ... te
   $(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_i|\vec{c}_i) = c_{1i} = 0$
6. Repeat 1-5 with 2nd row & column, etc. to get $A = I$
7. Use $Q^{(new)}$ & (Controlled-$Z)^{(new)}$ on R make $B = 0$
8. This forces $D = I$: $(\vec{a}_i|\vec{c}_i)\cdot(\vec{b}_j|\vec{d}_j) =$

$A = \left(\begin{array}{c|c} 1 & 0 \\ 0 & ? \end{array}\right)\Big\}n$

$C = \left(\begin{array}{c|c} 0 & 0 \\ 0 & ? \end{array}\right)\Big\}n$

Controlled-$Z(i,j)$: Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

$H = Q(i)$: Adds ith row of bottom half to ith 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-$Z(1,i)$ to make 1st column of $C = 0$
5. 1st row of C is now 0 too! Columns on left commute

   $(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_i|\vec{c}_i) = c_{1i} = 0$
6. Repeat 1-5 with 2nd row & column, etc. to get $A = I$, $C = 0$
7. Use $Q^{(row)}$ & (Controlled-$Z)^{(row)}$ on R make $B = 0$
8. This forces $D = I$: $(\vec{a}_i|\vec{c}_i)\cdot(\vec{b}_j|\vec{d}_j) = \delta_{ij} = d_{ij}$ $\Rightarrow U = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & I \end{array}\right)$

$A = \underbrace{\left(\begin{array}{c|c} 1 & 0 \cdots 0 \\ \hline 0 & \\ \vdots & ? \\ 0 & \end{array}\right)}_{n} \Big\} n$

$C = \underbrace{\left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & ? \end{array}\right)}_{n-1} \Big\} n-1$

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

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

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

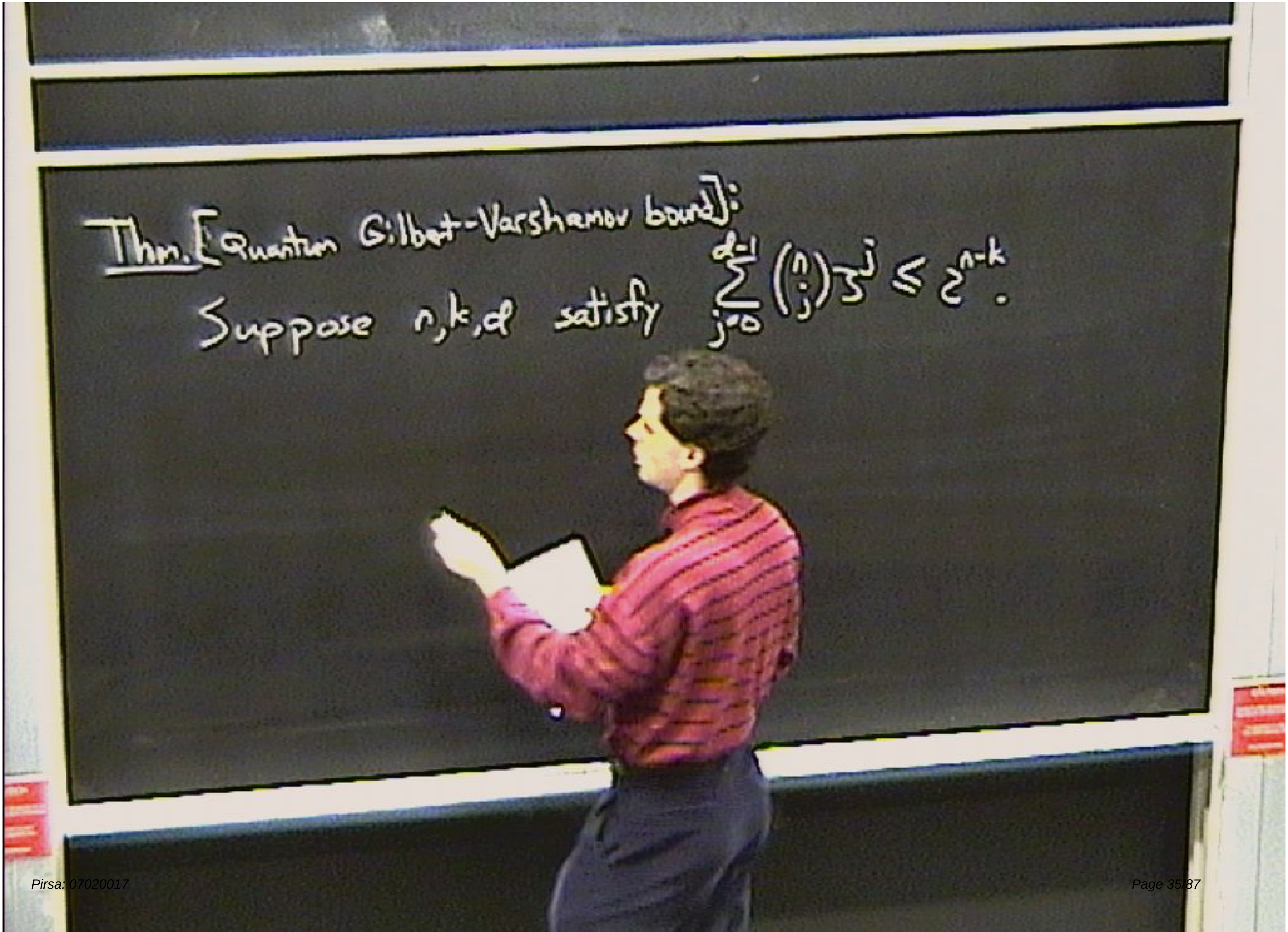
= SWAP(i,j): Swaps the i,jth rows in both top & bottom half

= Controlled-Z(i,j): Adds ith row of top half to jth row of bottom half
Adds jth row of top half to ith row of bottom half

= Q(i): Adds ith row of bottom half to ith of top half

On right: row → column, top → right, bottom → left

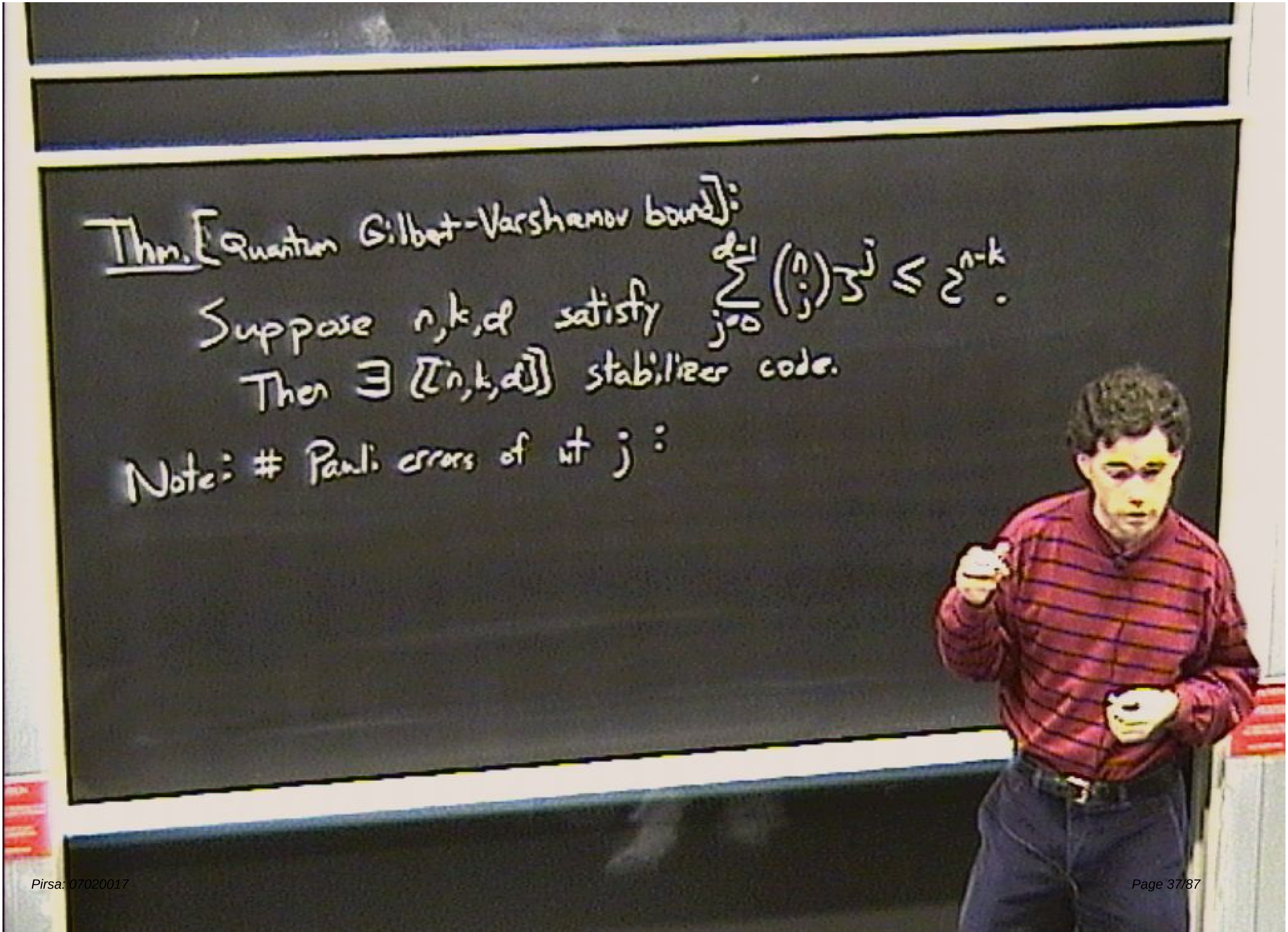
1. Put 1 in upper left corner of A using SWAP, H.
2. Use CNOT$(1,i)$ to make rest of 1st column of A = 0
3. Use CNOT on right to eliminate 1st row of A
4. Use R(1) & Controlled-Z$(1,i)$ to make 1st column of $C=0$
5. 1st row of C is now 0 too! Columns on left commute
$(\vec{a}_1|\vec{c}_1)\cdot(\vec{a}_j|\vec{c}_j) = c_{1j} = 0$
6. Repeat 1-5 with 2nd row/column, etc. to get $A=I$, $C=0$
7. Use Q (row) & Controlled-Z (pair) on R make $B=0$
8. This forces $D=I$: $(\vec{a}_i|\vec{c}_i)(\vec{b}_j|\vec{d}_j) = \delta_{ij} = d_{ij}$ $\Rightarrow U = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & I \end{array}\right)$


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.

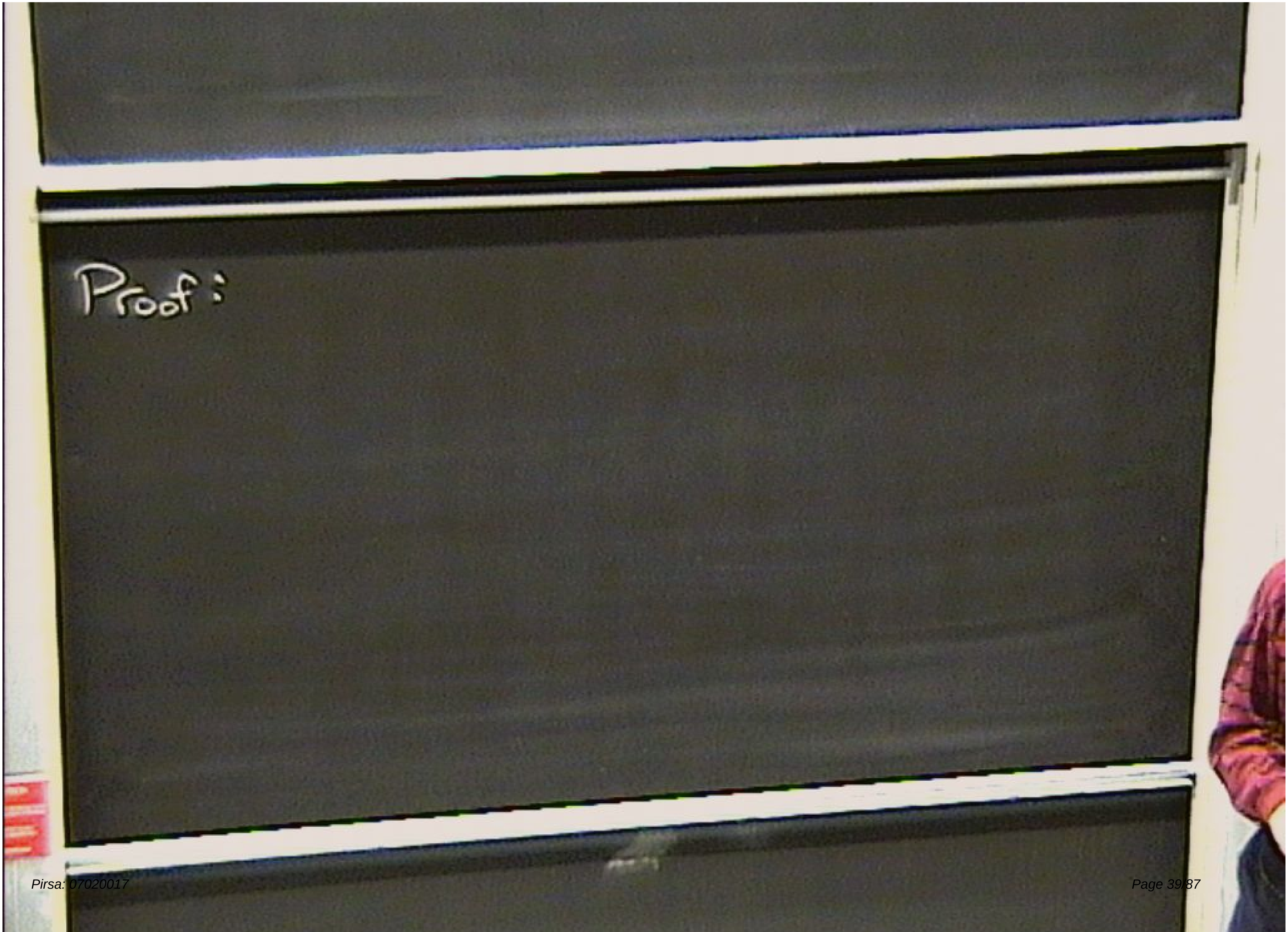

Thm. [Quantum Gilbert-Varshamov bound]:
Suppose n,k,d satisfy $\sum_{j=0}^{d-1} \binom{n}{j} 3^j \leq 2^{n-k}$.
Then $\exists$ [[n,k,d]] stabilizer code.
Note: # Pauli errors of wt j:

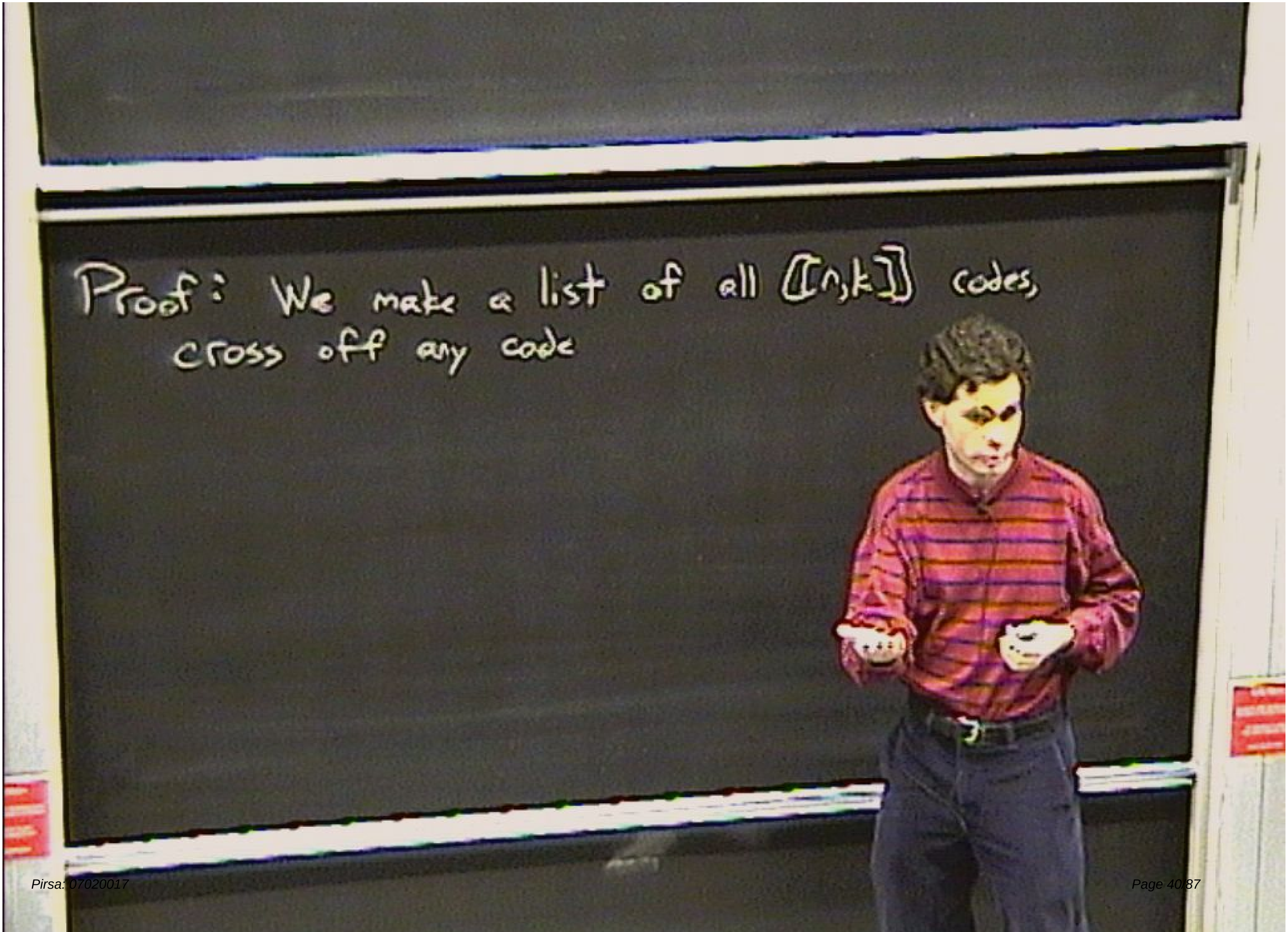
Thm. [Quantum Gilbert-Varshamov bound]:

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

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 $E_{n,d} = \sum_{j=0}^{d-1} \binom{n}{j} 3^j$ errors of weight $<d$.

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

There are $E_{n,d} = \sum_{j=0}^{d-1} \binom{n}{j} 3^j$ errors of weight $<d$.

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

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

How may codes can we cross off for E? All Es 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 $< d$.

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

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

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

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

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

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

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

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

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

$N_{n,k}$ 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 $E_{n,d} = \sum_{j=0}^{d-1} \binom{n}{j} 3^j$ errors of weight $<d$. Cross off all codes for E (wt $E<d$) is in $N(S)\setminus S$.

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

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

$E_{n,d}$ errors, each appears K times

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

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

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

$N_{n,k}$ codes, each code has $|N(S)\setminus S| = 2^{n+k} - 2^{n-k}$ $\Rightarrow (4^n-1)K = N_{n,k}(2^{n+k} - 2^{n-k})$

$4^n - 1$ errors, each appears K times

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

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

$N_{n,k}$ codes, each code has $|N(S)\setminus S| = 2^{n+k} - 2^{n-k}$ $\Rightarrow (4^n - 1)K = N_{n,k}(2^{n+k} - 2^{n-k})$

$4^n - 1$ errors, each appears K times

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

Can at most cross off $E_{n,d} K$ codes

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

$N_{n,k}$ codes, each code has $|N(S)\setminus S| = 2^{n+k} - 2^{n-k}$ $\Rightarrow (4^n-1)K = N_{n,k}(2^{n+k}-2^{n-k})$

4^n-1 errors, each appears K times

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

$$\boxed{\begin{array}{l}\text{Can at most cross off}\\ E_{n,d}K \text{ codes} \leq \frac{E_{n,d}}{2^{n-k}}N_{n,k}\\ \text{If} < N_{n,k}\text{, a code is left!}\end{array}}$$

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

all codes for E (wt $E < d$) is in $N(S) \setminus S$.

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

$N_{n,k}$ codes, each code has $|N(S) \setminus S| = 2^{n+k} - 2^{n-k}$ $\Rightarrow (4^n - 1) K = N_{n,k} (2^{n+k} - 2^{n-k})$

$4^n - 1$ 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 $E_{n,d} K$ codes $\leq \frac{E_{n,d}}{2^{n-k}} N_{n,k}$

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

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

<u>Thm.</u> [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 $j = \binom{n}{j} 3^j$

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 wt $j = \binom{n}{j} 3^j$

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

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

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 $n \to \infty$, $k \to \infty$, $d \to \infty$, $\frac{k}{n} = R$ and $\frac{d}{n}$ constant (good code)

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

$\log \sum \binom{n}{j} 3^j \lessapprox n h(\frac{d}{n})$ + $d \log 3$

$n-k > n h(\frac{d}{n}) + d \log 3 \Leftrightarrow$ If $R < 1 - h(\frac{d}{n}) - \frac{d}{n} \log 3$

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

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

$\binom{n}{j} \to 2^{nh(d/n)}$ $\quad h(x) = -x\log_2 x - (1-x)\log_2(1-x)$

$\log \sum \binom{n}{j} 3^j \approx nh(\frac{d}{n})$

If $R < 1 - h(\frac{d}{n}) - \frac{d}{n}\log 3$

$n-k > nh(\frac{d}{n}) + d\log 3 \iff$

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

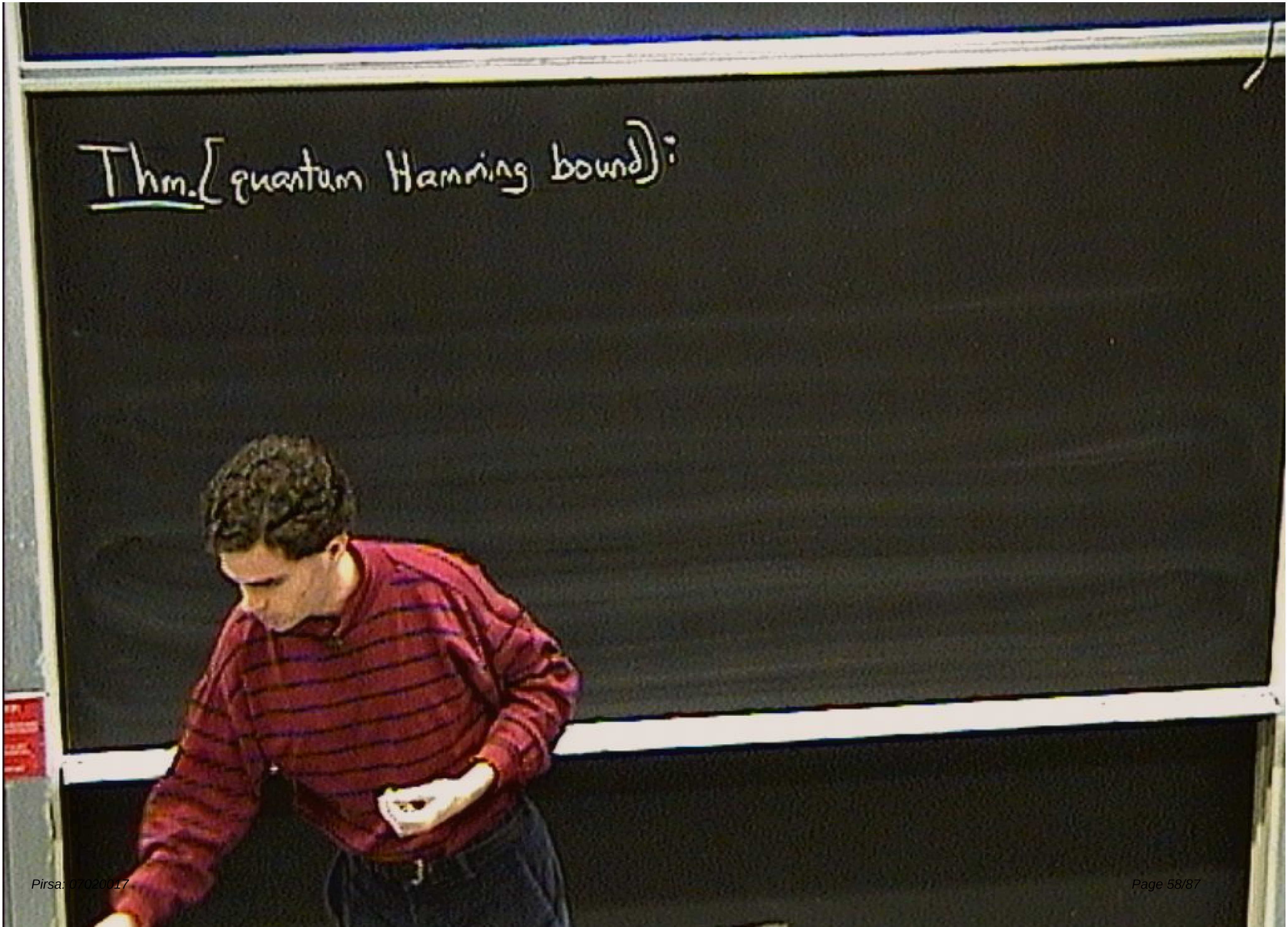
How many codes can we cross off for E? All non-identity E's have same # of codes for which they are in $N(S)\setminus S$, regardless of wt E (by Clifford gp.)

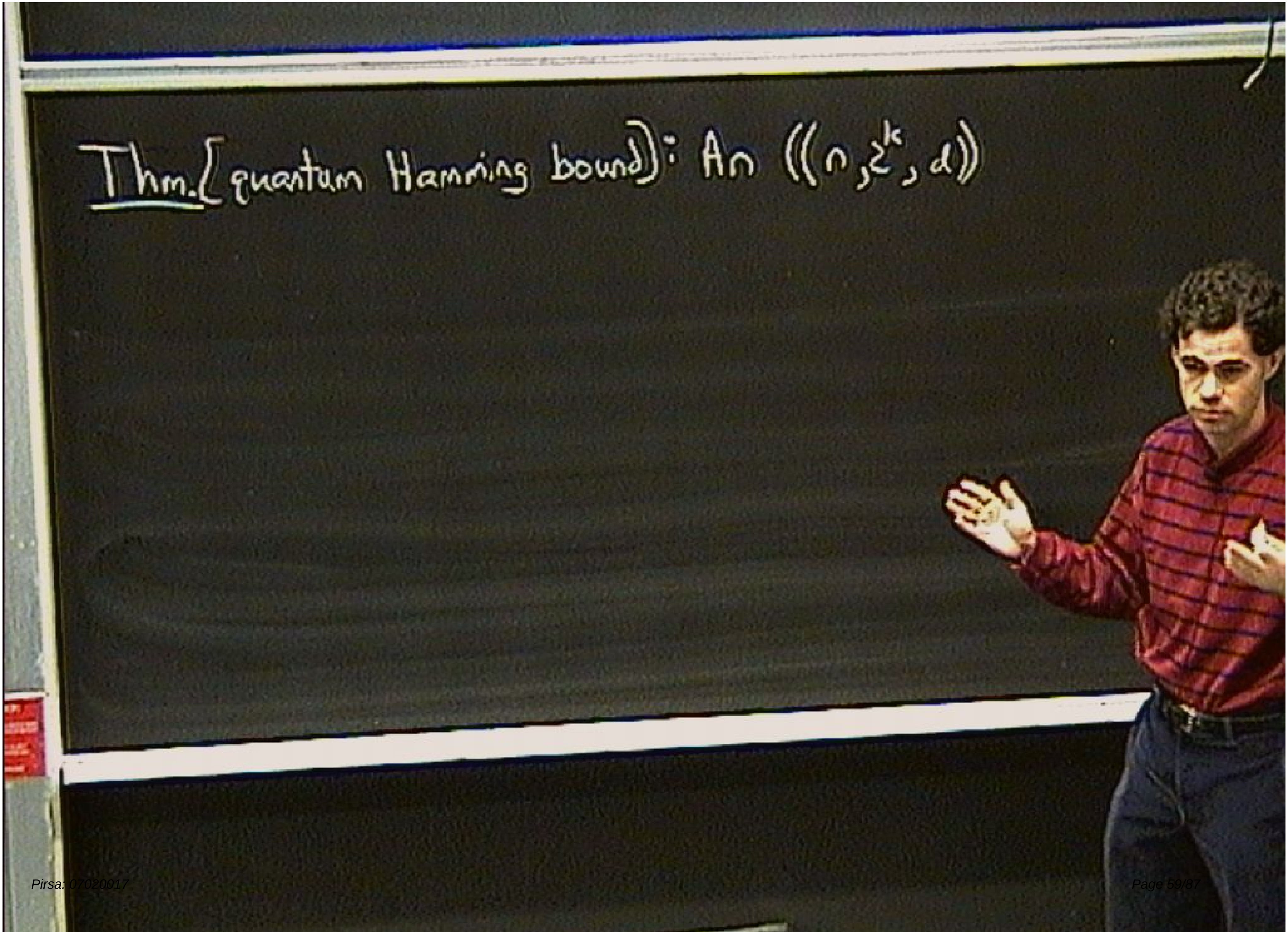
$N_{n,k}$ codes, each code has $|N(S)\setminus S| = 2^{n+k} - 2^{n-k}$ $\Rightarrow (4^n-1)K = N_{n,k}(2^{n+k}-2^{n-k})$

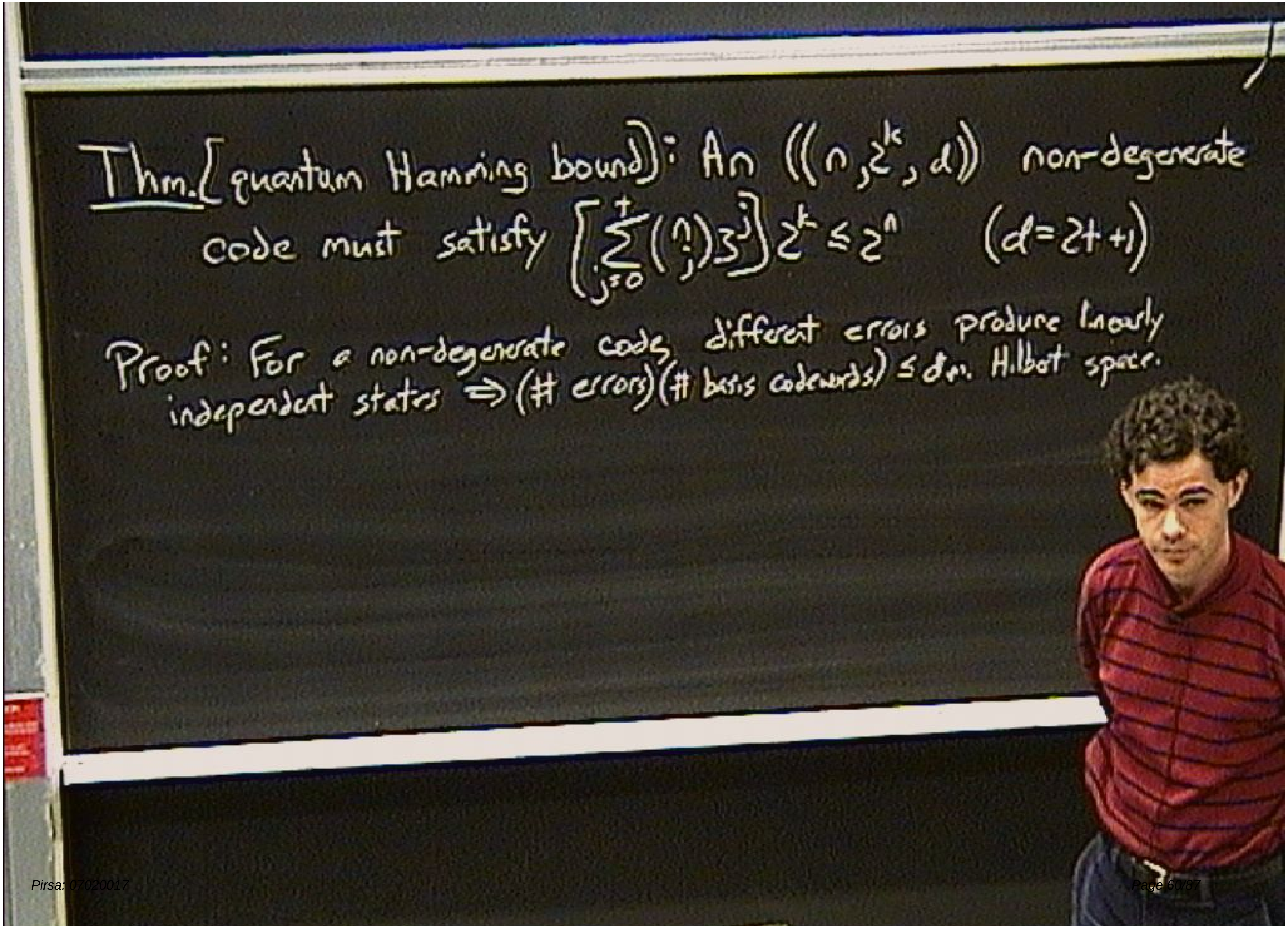
4^n-1 errors, each appears K times

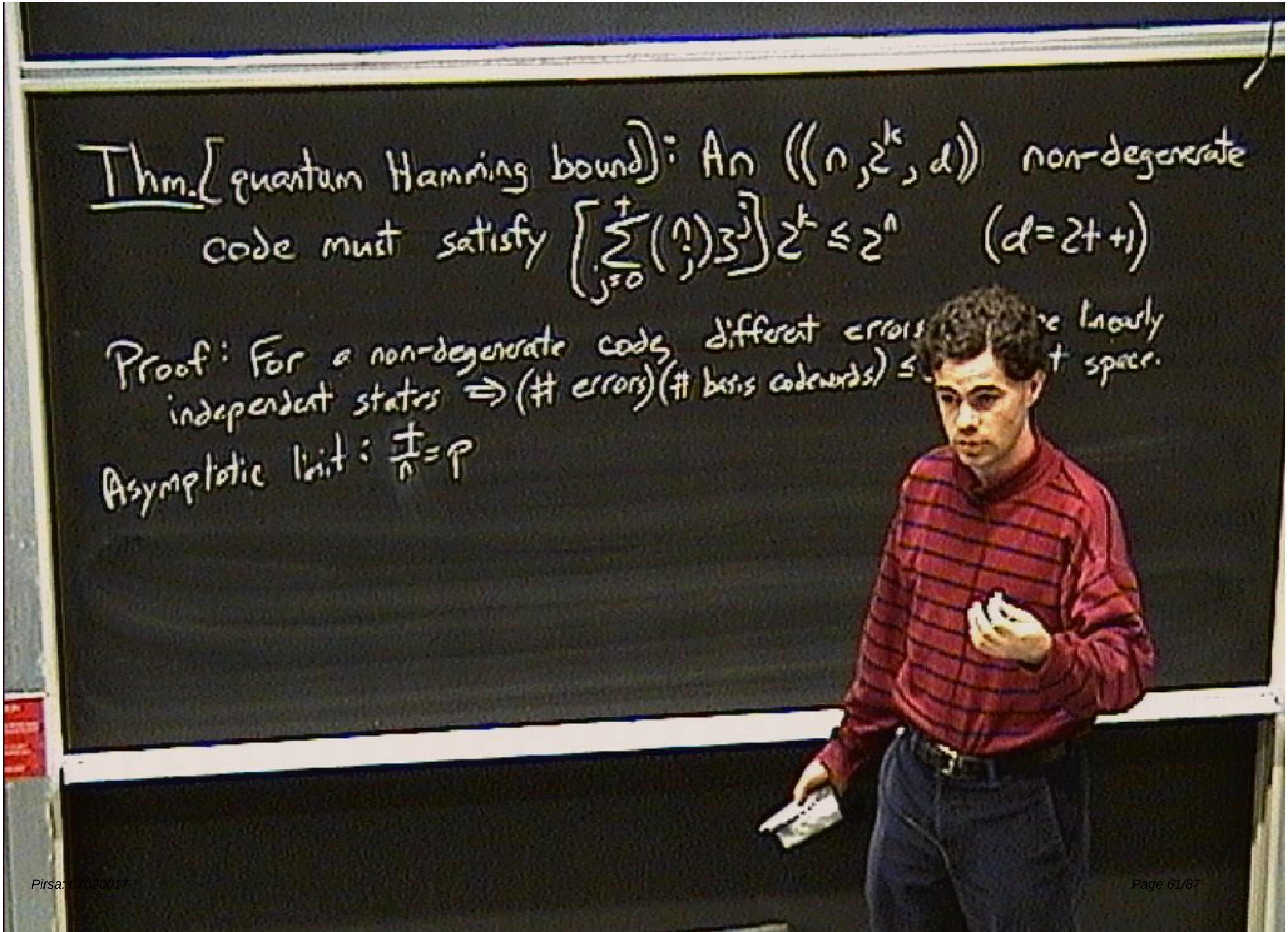
$$K = N_{n,k}\left(\frac{2^{n+k}-2^{n-k}}{4^n-1}\right) \le N_{n,k}\frac{2^{n+k}}{4^n} = \frac{N_{n,k}}{2^{n-k}}$$

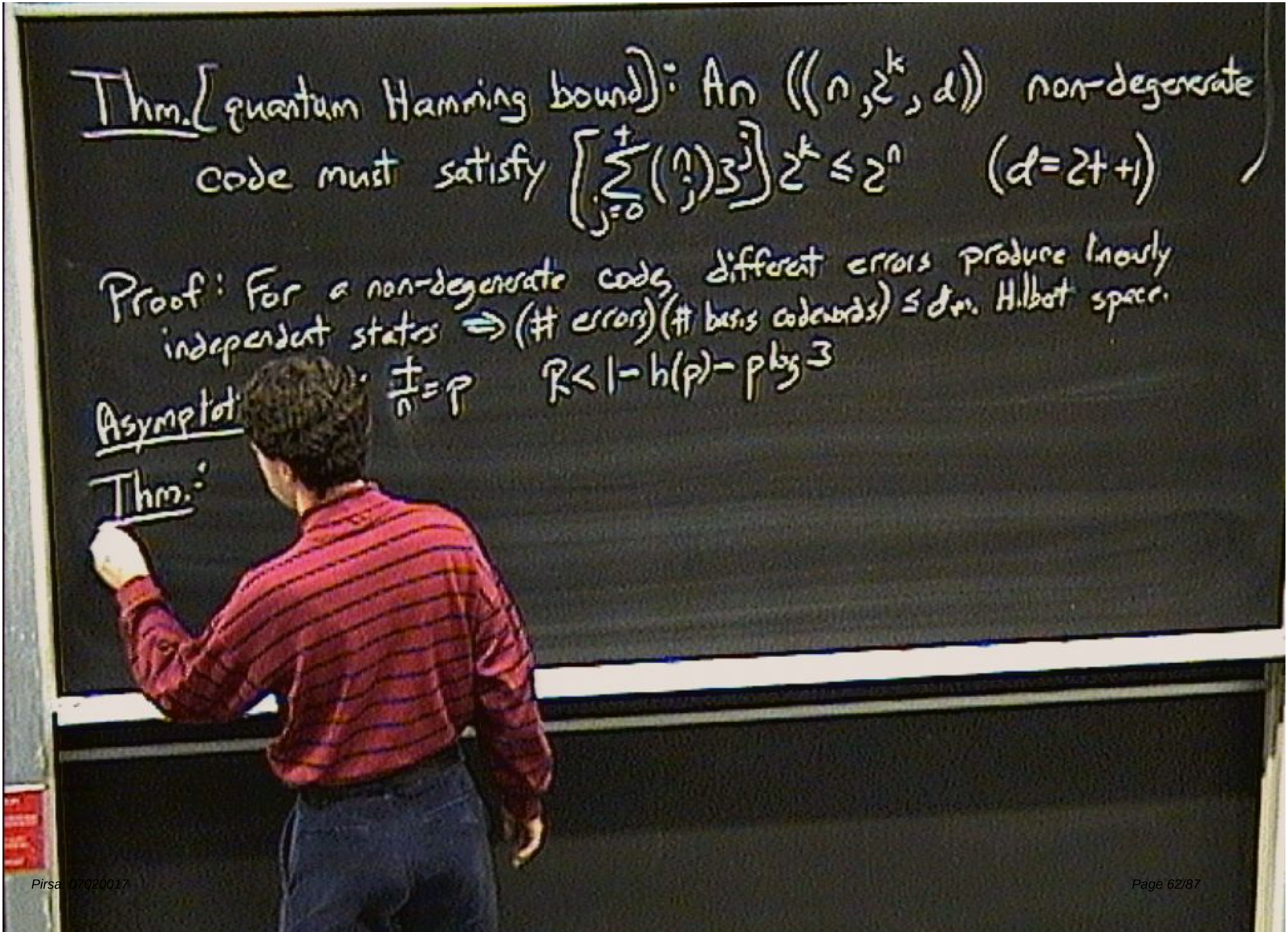
$\boxed{\text{Can at most cross off } E_{n,d}K \text{ codes} \le \frac{E_{n,d}}{2^{n-k}} N_{n,k}. \text{ If } < N_{n,k}, \text{ a code is left!}}$

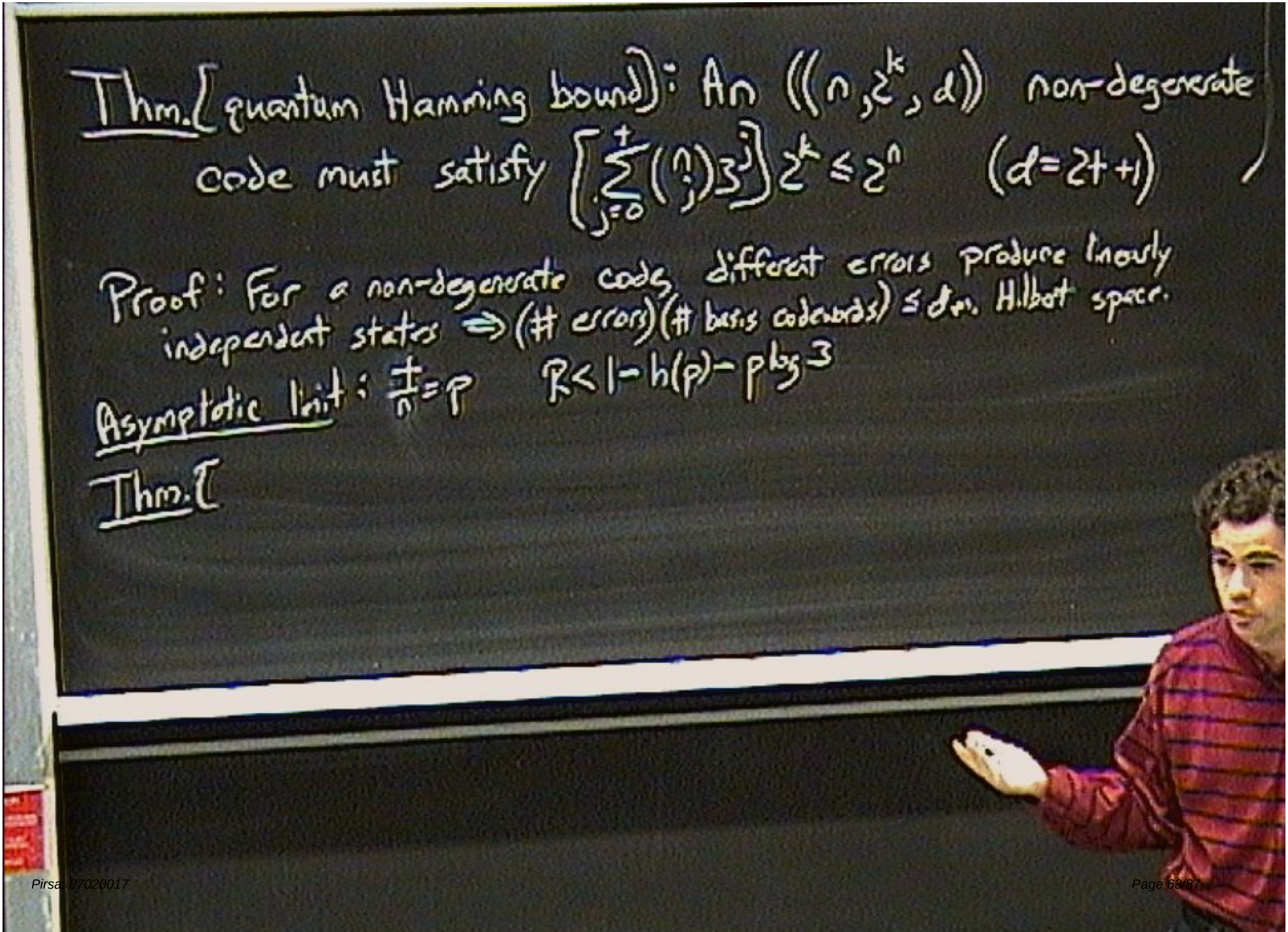

Thm. (quantum Hamming bound):


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


Thm. [quantum Hamming bound]: An $((n, 2^k, d))$ non-degenerate code must satisfy $\left[\sum_{j=0}^{t} \binom{n}{j} 3^j\right] 2^k \leq 2^n$ $(d = 2t+1)$
Proof: For a non-degenerate code, different errors produce linearly independent states $\Rightarrow$ (# errors)(# basis codewords) $\leq$ dim. Hilbert space.


Thm. [quantum Hamming bound]: An $((n, 2^k, d))$ non-degenerate
code must satisfy $\left[\sum_{j=0}^{t} \binom{n}{j} 3^j\right] 2^k \leq 2^n$ $(d = 2t+1)$
Proof: For a non-degenerate code, different errors
e linearly
independent states ⇒ (# errors)(# basis codewords) ≤
t space.
Asymptotic limit: $\frac{t}{n} = p$


Thm. (quantum Hamming bound): An $((n, 2^k, d))$ non-degenerate
code must satisfy $\left[\sum_{j=0}^{t} \binom{n}{j} 3^j\right] 2^k \le 2^n$ $(d = 2t+1)$
Proof: For a non-degenerate code, different errors produce linearly
independent states $\Rightarrow$ (# errors)(# basis codewords) $\le$ dim. Hilbert space.
Asymptot
$\frac{t}{n} = p$ $R < 1 - h(p) - p \log 3$
Thm.:


Thm.[quantum Hamming bound]: An $((n, 2^k, d))$ non-degenerate
code must satisfy $\left[\sum_{j=0}^{t} \binom{n}{j} 3^j\right] 2^k \le 2^n$ $(d = 2t+1)$
Proof: For a non-degenerate code, different errors produce linearly
independent states ⇒ (# errors)(# basis codewords) ≤ dim. Hilbert space.
Asymptotic limit: $\frac{t}{n} = p$ $R < 1 - h(p) - p \log 3$
Thm.[

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

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

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

Thm. [quantum Singleton bound]: $n - k \ge 2(d-1)$ $\quad$ ((Note: this applies to non-binary registers)
(Knill-Laflamme bound)

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

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

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

Thm. [quantum Singleton bound]: $n - k \ge 2(d-1)$ (Note: … lies to non-bin… …sters)
(Knill-Laflamme bound)

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

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

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

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

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

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

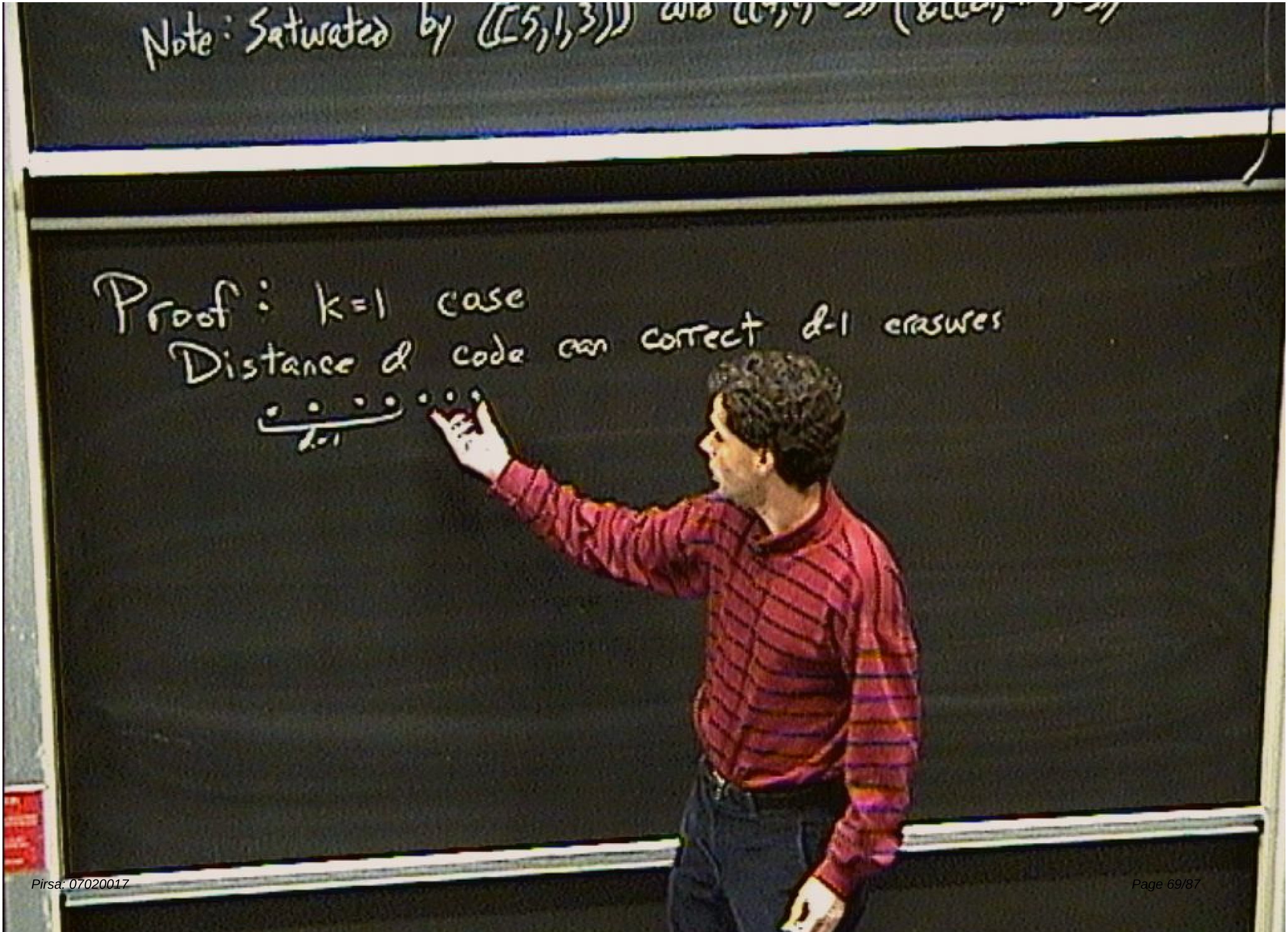
Note: Saturated by $[[5,1,3]]$ and [illegible]

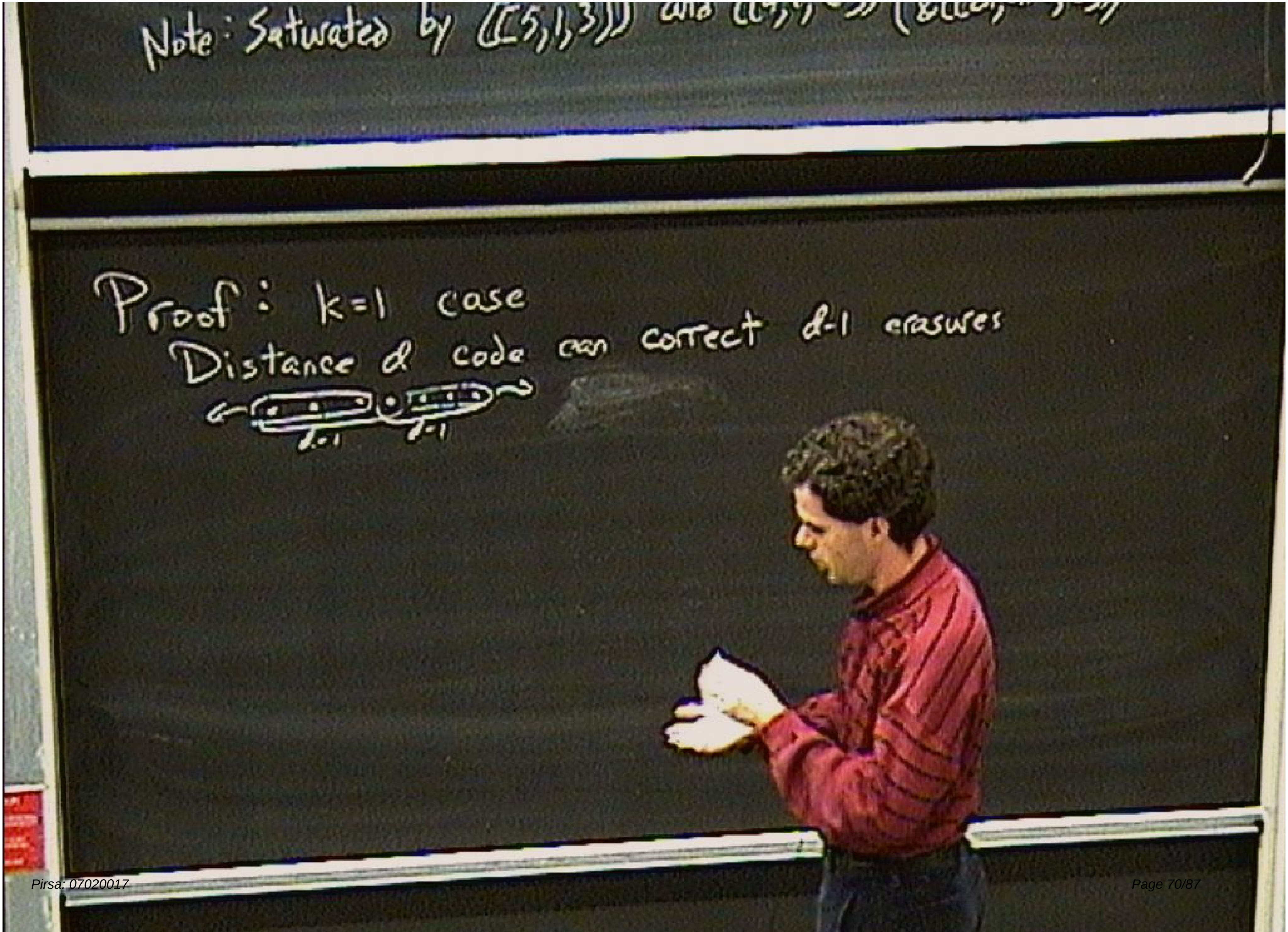
Proof: $k=1$ case

Note: Saturated by $[[5,1,3]]$ and ...

Proof: $k=1$ case

Distance d code can correct $d-1$ erasures


Note: Saturated by [[5,1,3]]
Proof: k=1 case
Distance d code can correct d-1 erasures
d-1


Note: Saturated by [[5,1,3]]
Proof: k=1 case
Distance d code can correct d-1 erasures
d-1
d-1

Note: Saturated by $[[5,1,3]]$ and

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 –

Note: Saturated by $[[5,1,3]]$ and $[[\ldots$

Proof: $k=1$ case

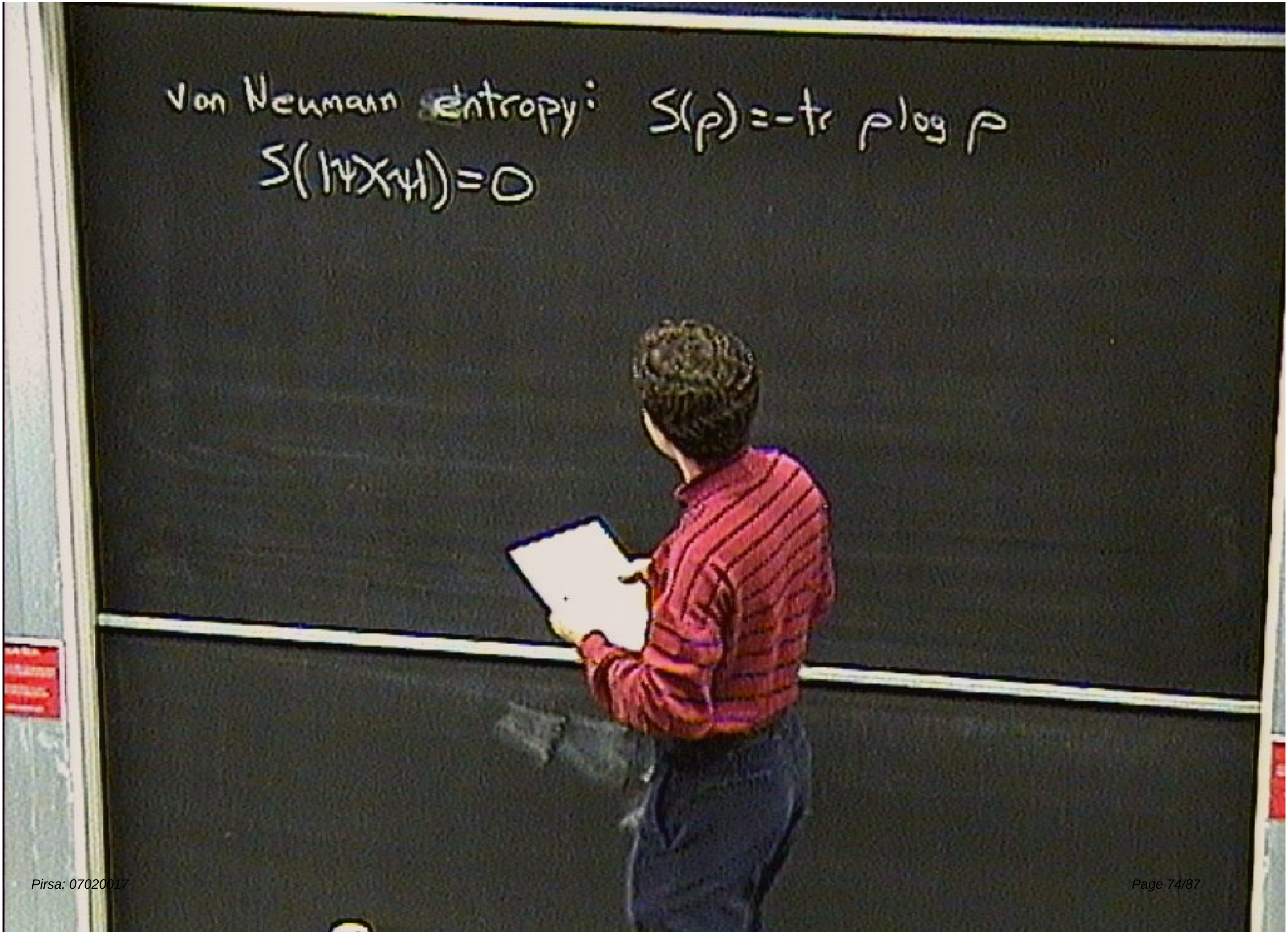
Distance d code can correct $d-1$ erasures

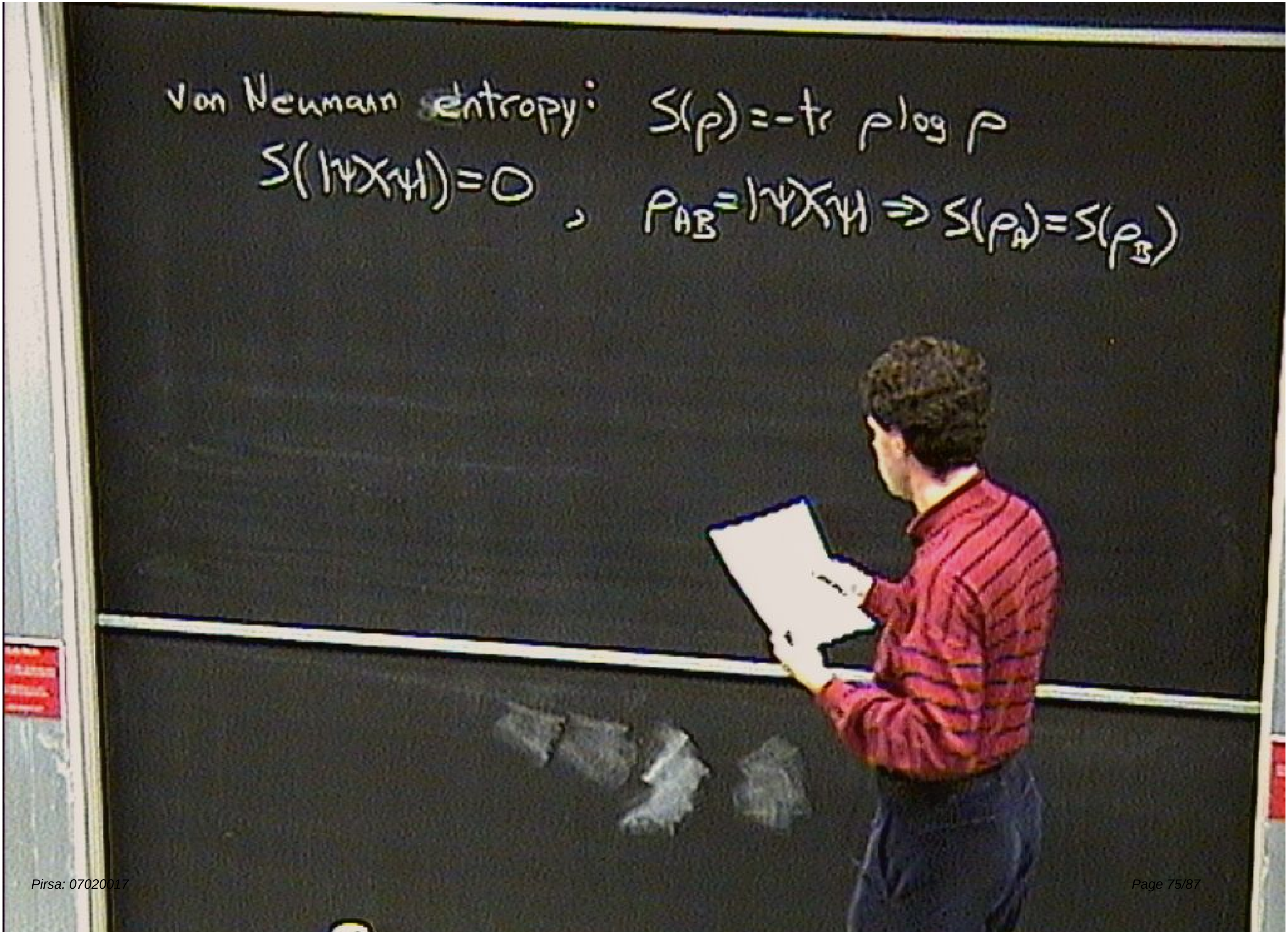
$\underbrace{\bullet\ \bullet\ \bullet}_{d-1}\ \bullet\ \underbrace{\bullet\ \bullet\ \bullet}_{d-1}$ Given $n-(d-1)$ qubits, can recover state

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

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

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


von Neumann entropy: $S(\rho) = -\mathrm{tr}\, \rho \log \rho$
$S(|\psi\rangle\langle\psi|) = 0$


von Neumann entropy: $S(\rho) = -\mathrm{tr}\, \rho \log \rho$
$S(|\psi\rangle\langle\psi|) = 0$, $\rho_{AB} = |\psi\rangle\langle\psi| \Rightarrow S(\rho_A) = S(\rho_B)$

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

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

$S(\rho) \leq \log$ dimension (i.e. # qubits) $=$ if $\rho = I/d_n$

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

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

$S(\rho) \leq \log$ dimension (i.e. # qubits) $=$ if $\rho = I/d_n$

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

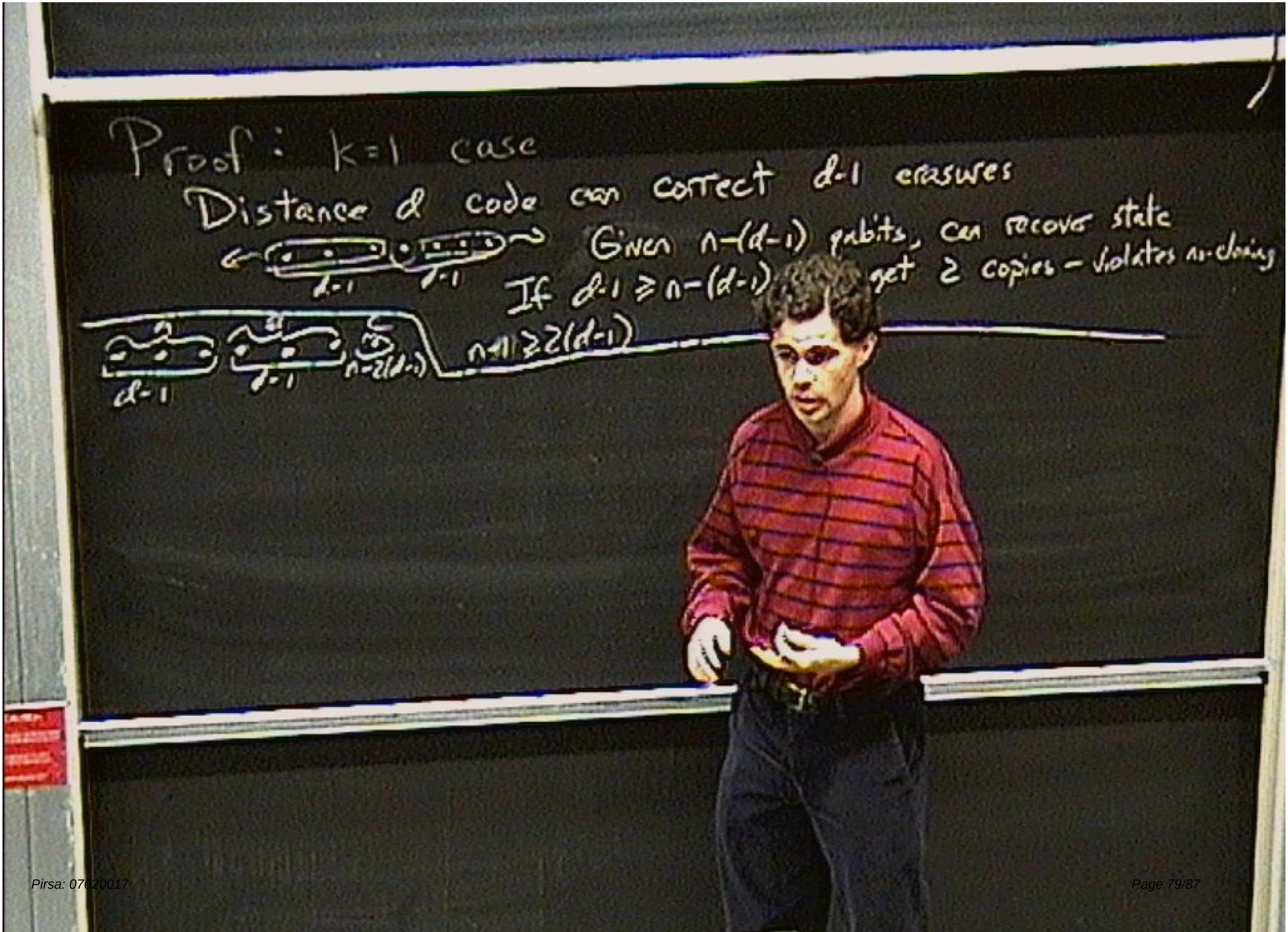
Von Neumann entropy: $S(\rho) := -\mathrm{tr}\, \rho \log \rho$

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

$S(\rho) \leq \log$ dimension (i.e. # qubits) $=$ if $\rho = I/\dim$

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 ≥ n-(d-1), get 2 copies - violates no-cloning
d-1
d-1
n+1 ≥ 2(d-1)
n-2(d-1)
d-1
d-1

Proof: $k=1$ case

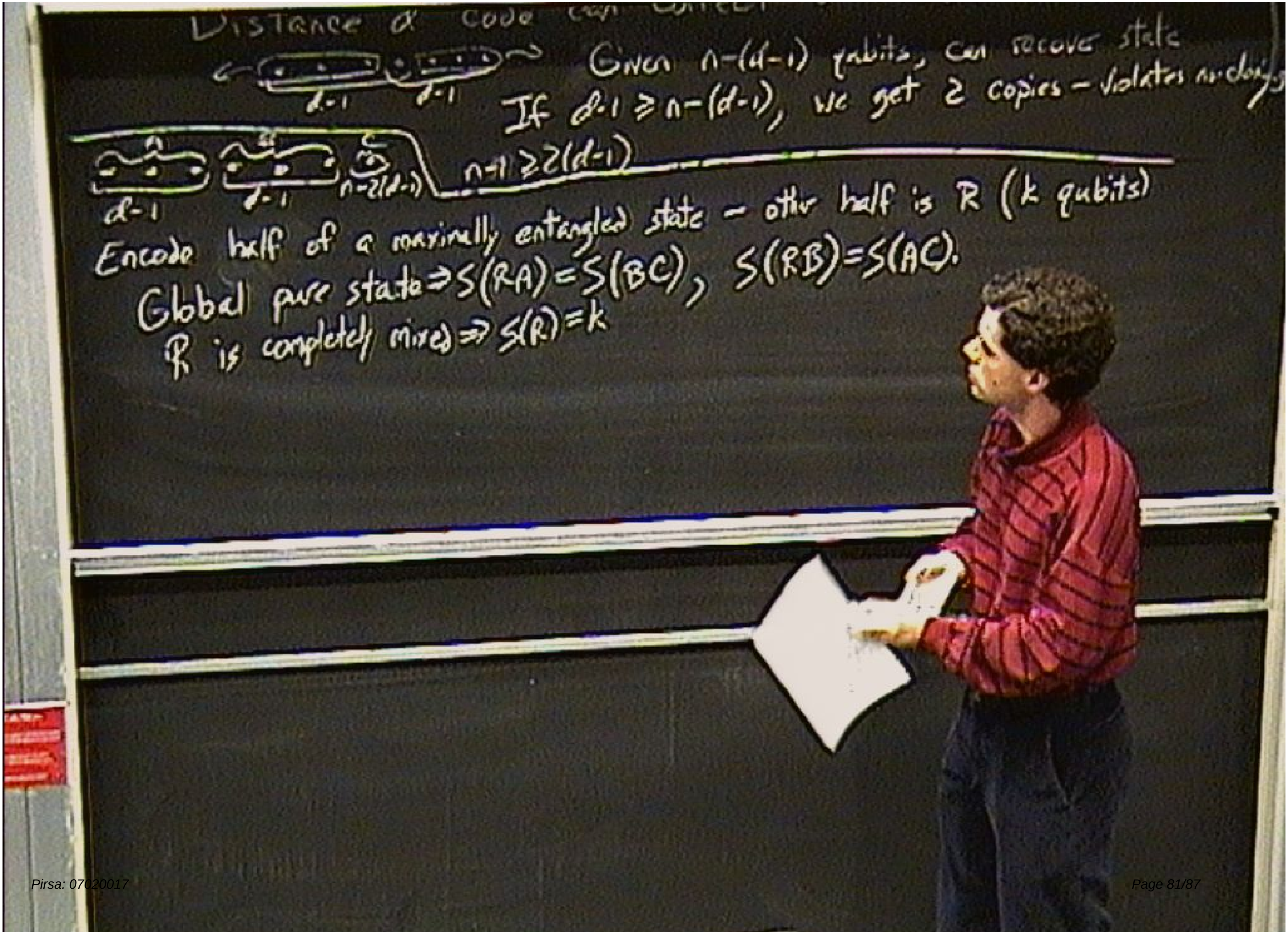
Distance d code can correct $d-1$ erasures

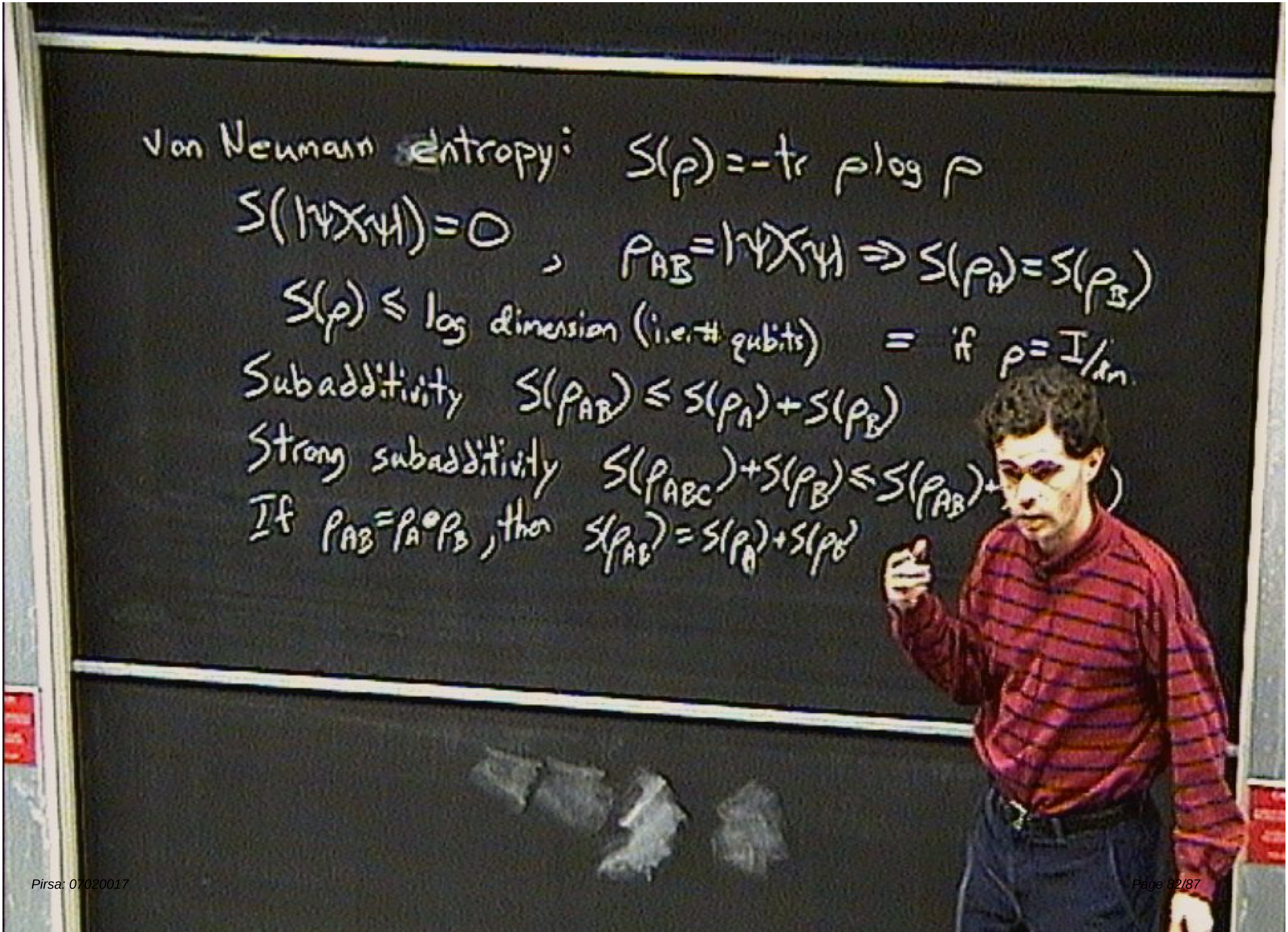
$d-1$ $d-1$ Given $n-(d-1)$ qubits, can recover state

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

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

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


Distance d code
Given n-(d-1) qubits, can recover state
d-1
d-1
If d-1 ≥ n-(d-1), we get 2 copies - violates
n-1 ≥ 2(d-1)
d-1
d-1
n-2(d-1)
Encode half of a maximally entangled state - other half is R (k qubits)
Global pure state ⇒ S(RA)=S(BC), S(RB)=S(AC).
R is completely mixed ⇒ S(R)=k


von Neumann entropy: $S(\rho) = -\mathrm{tr}\, \rho \log \rho$
$S(|\psi\rangle\langle\psi|) = 0$, $\rho_{AB} = |\psi\rangle\langle\psi| \Rightarrow S(\rho_A) = S(\rho_B)$
$S(\rho) \leq \log$ dimension (i.e. # qubits) = if $\rho = I/\dim$
Subadditivity $S(\rho_{AB}) \leq S(\rho_A) + S(\rho_B)$
Strong subadditivity $S(\rho_{ABC}) + S(\rho_B) \leq S(\rho_{AB}) +$
If $\rho_{AB} = \rho_A \otimes \rho_B$, then $S(\rho_{AB}) = S(\rho_A) + S(\rho_B)$

von Neumann entropy: $S(\rho) = -\mathrm{tr}\,\rho \log \rho$

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

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

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

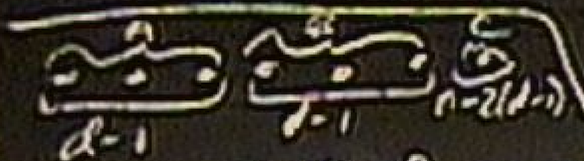
Strong subadditivity $S(\rho_{ABC}) + S(\rho_B) \leq S(\rho_{AB}) + S(\rho_{BC})$

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

Distance 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



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

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 $S(AR)=S(A)+S(R)$

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

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

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, $S(A)+S(R)=S(RA)=S(BC)\leq S(B)+S(C)$

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)

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, $S(A)+S(R)=S(RA)=S(BC)\leq S(B)+S(C) \Rightarrow k \leq S(C)+[S(B)-S(A)]$

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

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

A, B, C; $d-1$, $d-1$, $n-2(d-1)$

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

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, $S(A)+S(R)=S(RA)=S(BC)\leq S(B)+S(C) \Rightarrow k \leq S(C)+[S(B)-S(A)]$

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