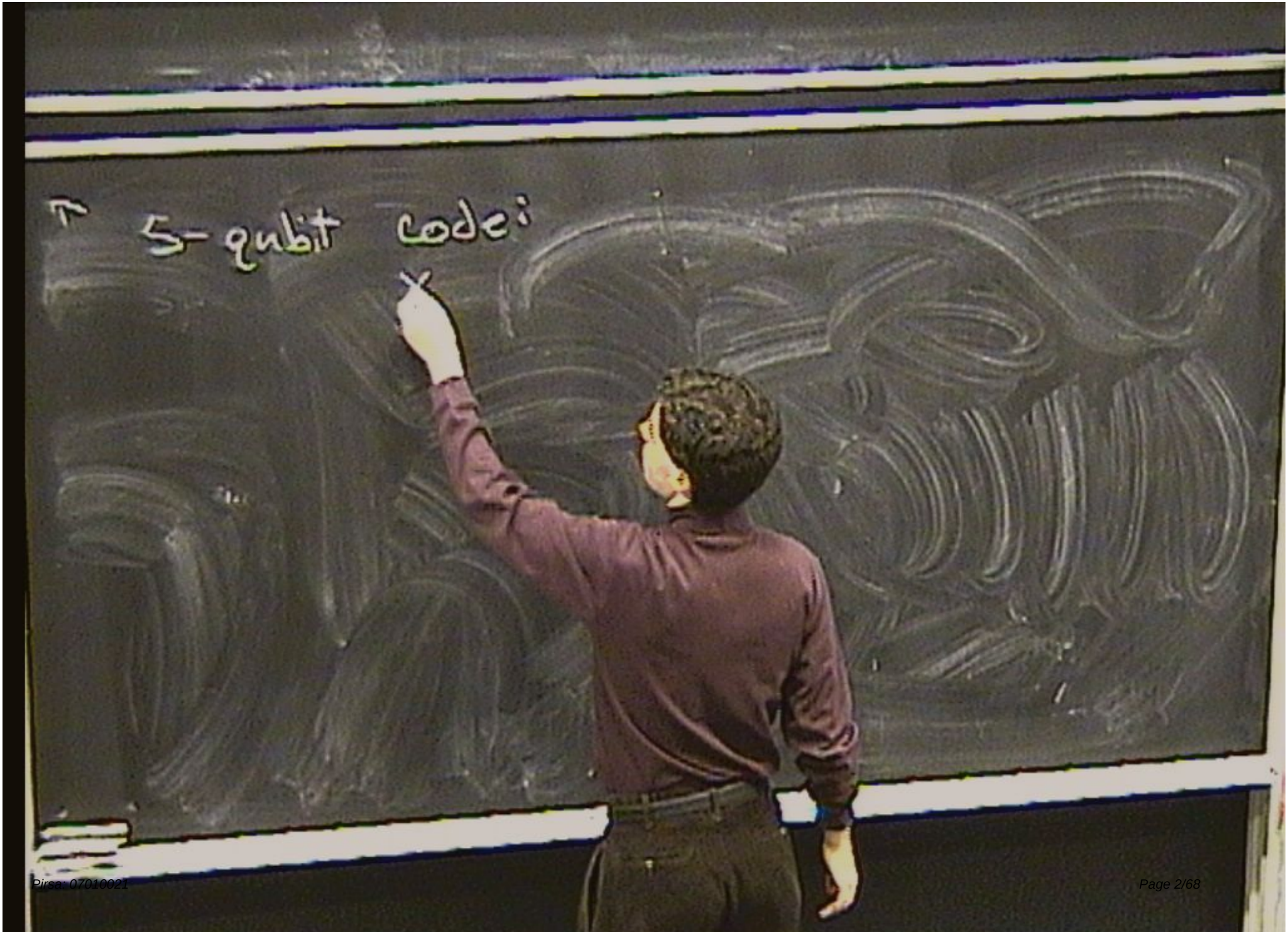
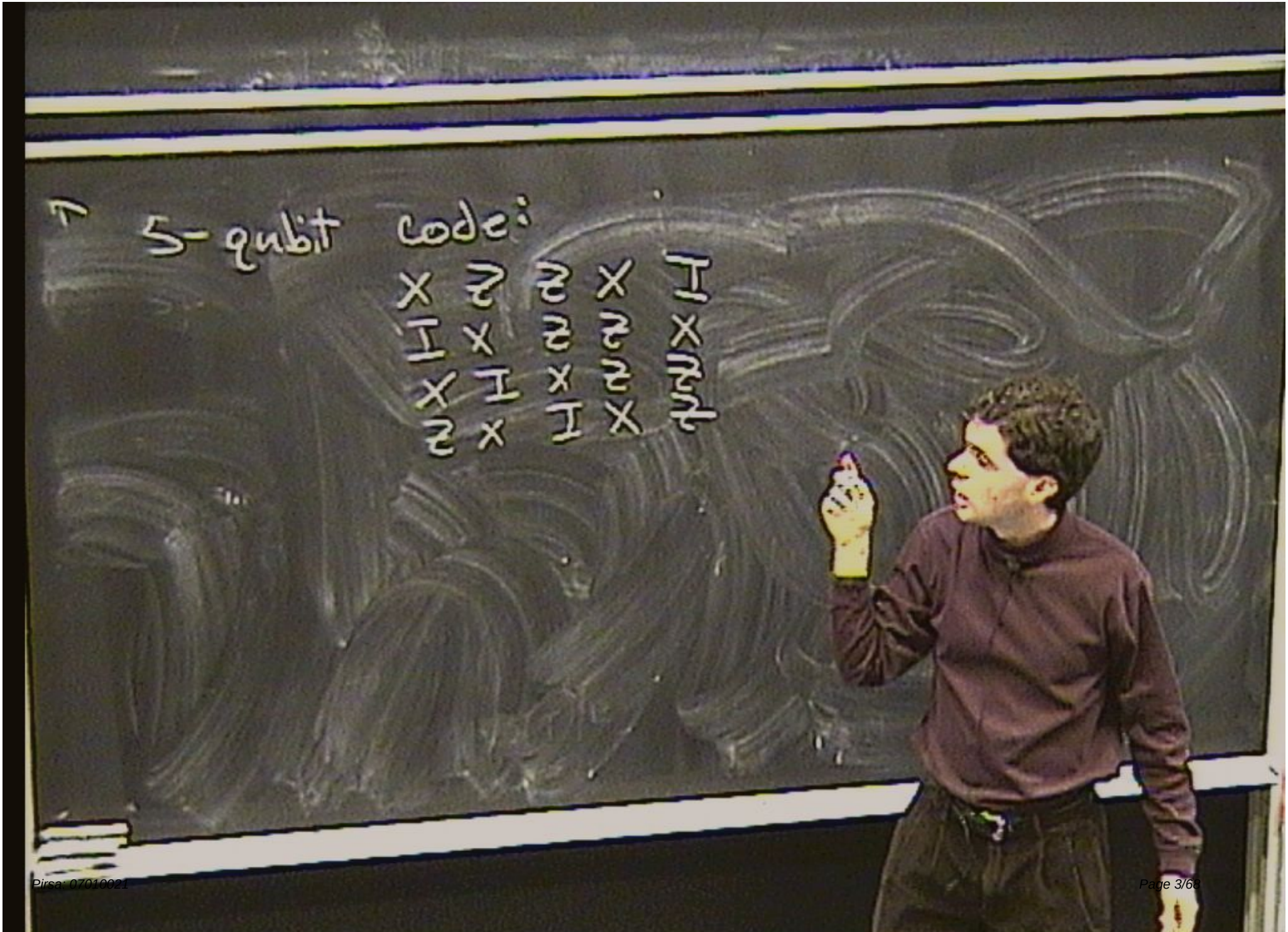
Title: Quantum Error Correction 2B

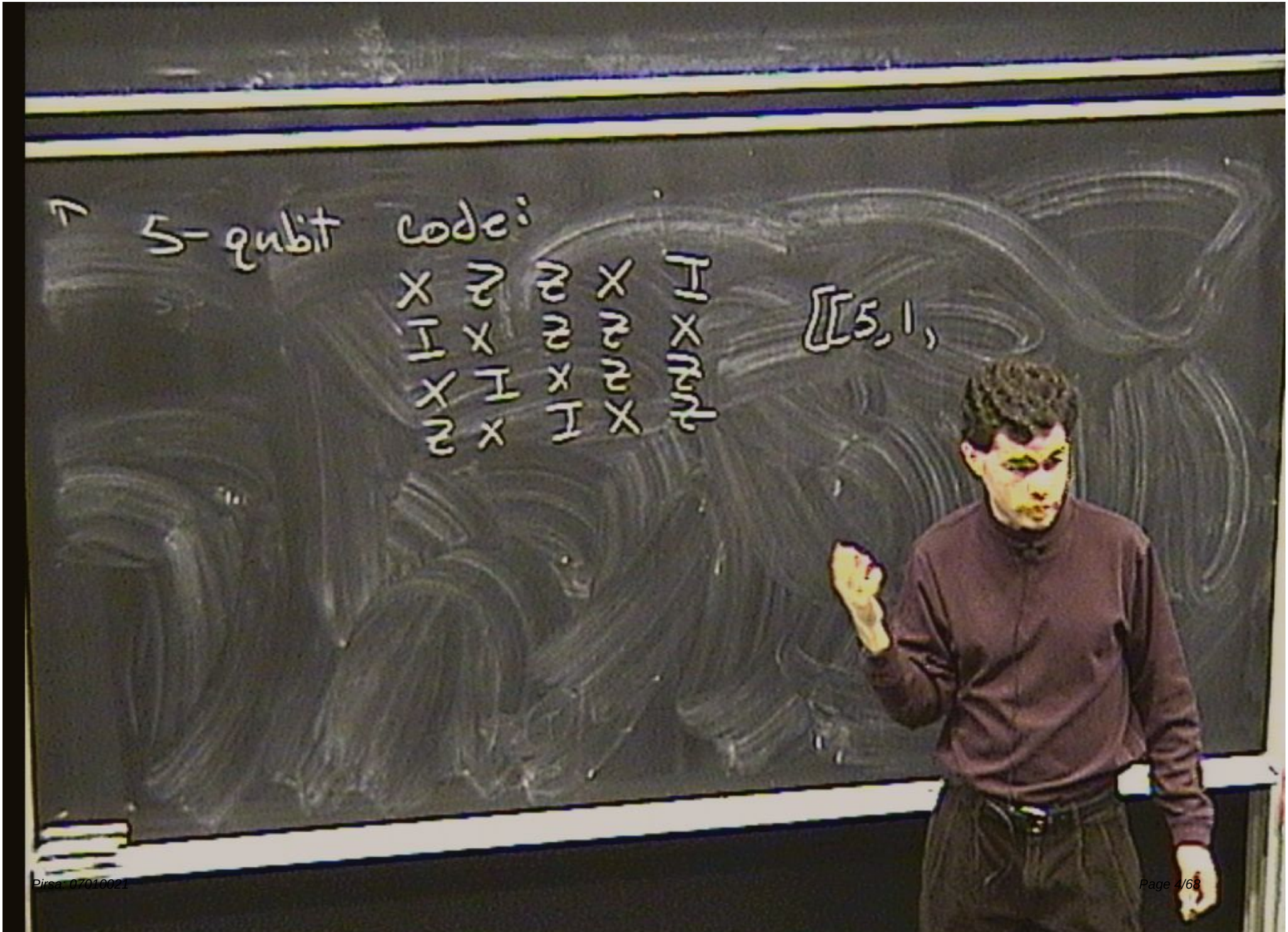
Date: Jan 16, 2007 05:00 PM

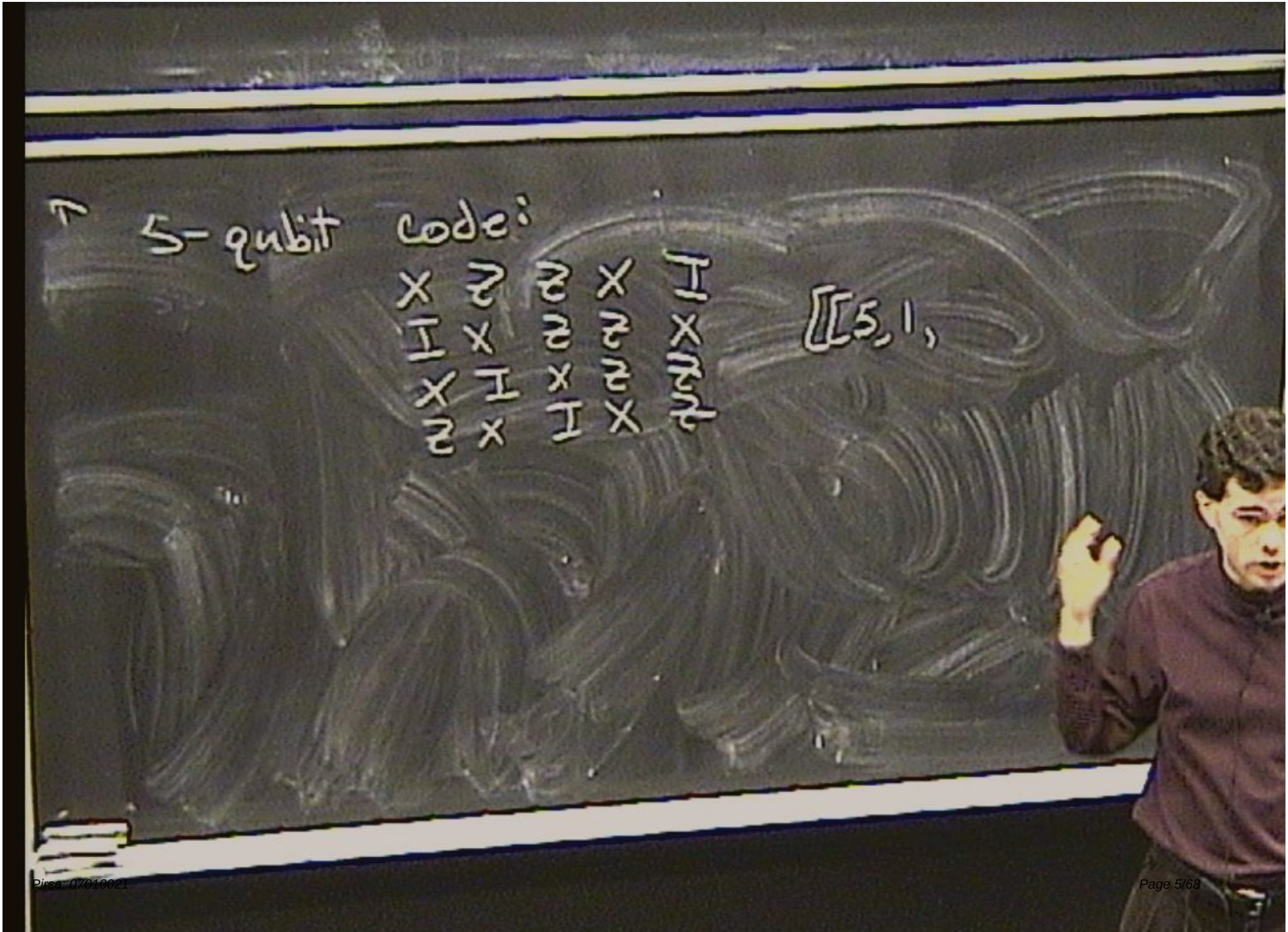
URL: http://pirsa.org/07010021

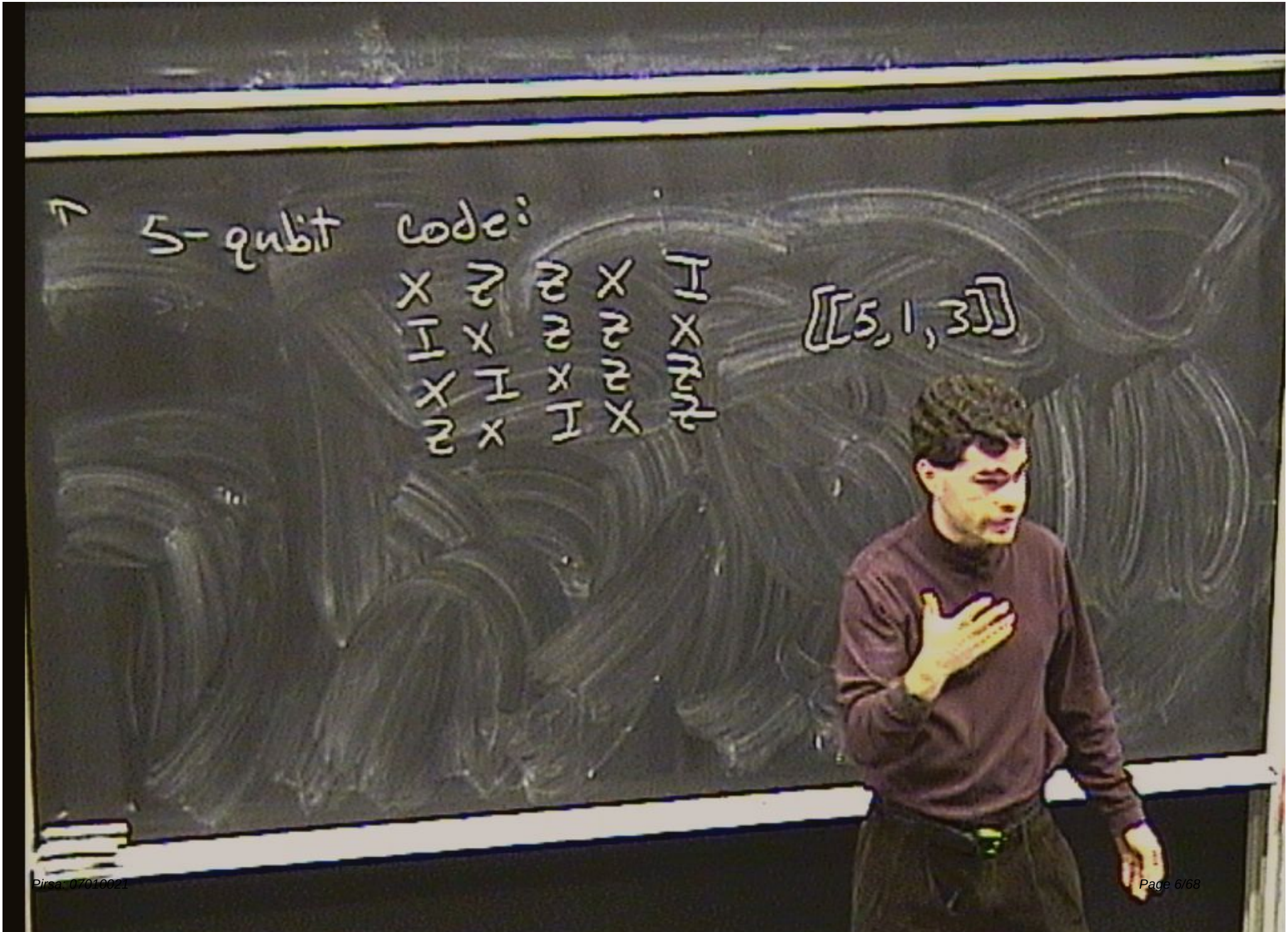
Abstract: 5-qubit code, logical Pauli group for stabilizer codes, classical linear codes (generator and parity check matrices, Hamming codes), CSS codes (definition, 7-qubit code)

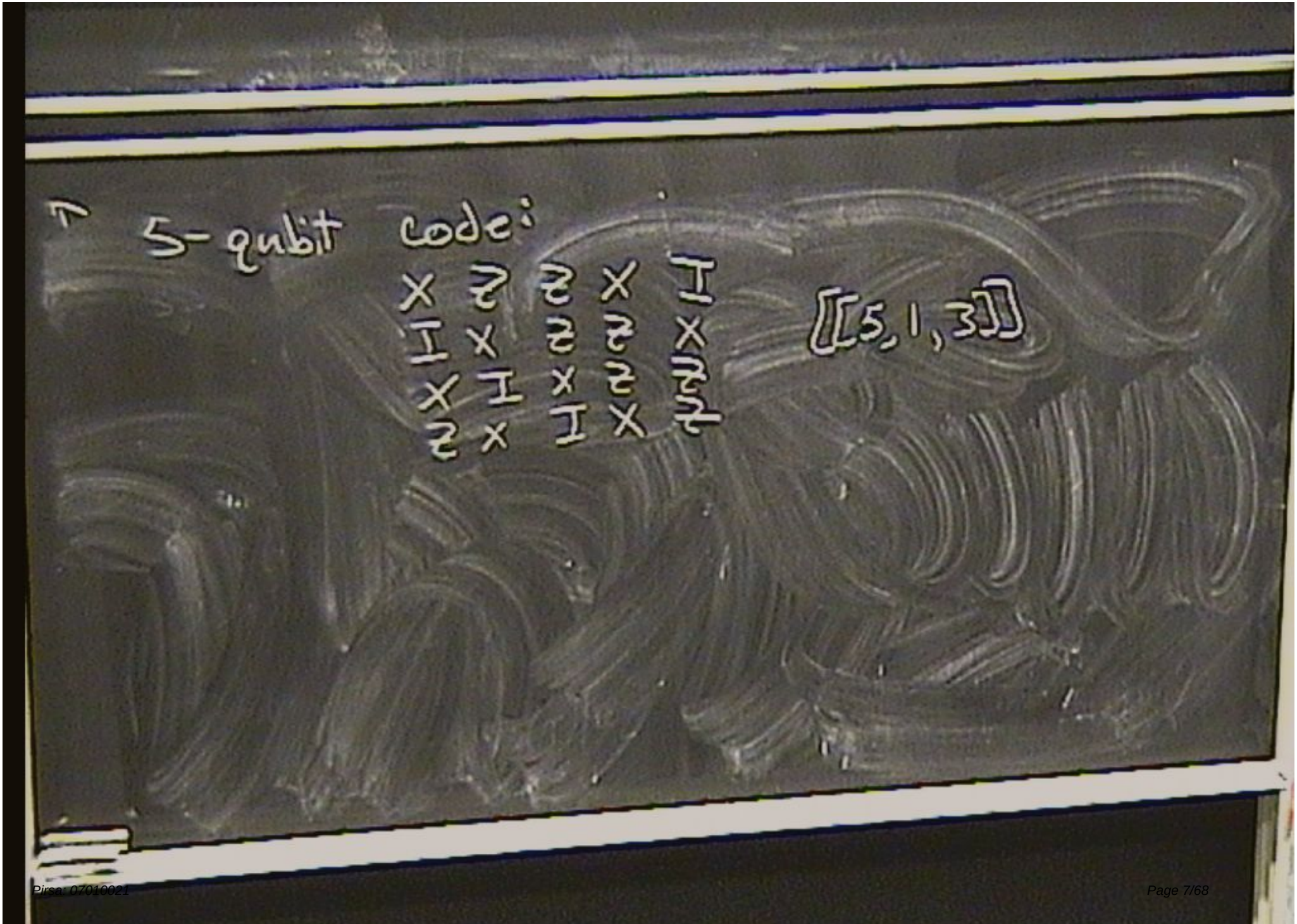

5-qubit code:


5-qubit code:
X Z Z X I
I X Z Z X
X I X Z Z
Z X I X Z


5-qubit code:
X Z Z X I
I X Z Z X
X I X Z Z
Z X I X Z
[[5, 1,


5-qubit code:
X Z Z X I
I X Z Z X
X I X Z Z
Z X I X Z
[[5, 1,


5-qubit code:
X Z Z X I
I X Z Z X
X I X Z Z
Z X I X Z
[[5,1,3]]


5-qubit code:
X Z Z X I
I X Z Z X
X I X Z Z
Z X I X Z
[[5,1,3]]

5-qubit code:

X	Z	Z	X	I
I	X	Z	Z	X
X	I	X	Z	Z
Z	X	I	X	Z

$[[5,1,3]]$

$|S| = 2^{n-k}$, $|N(S)|$

5-qubit code:

X	Z	Z	X	I
I	X	Z	Z	X
X	I	X	Z	Z
Z	X	I	X	Z

$[[5,1,3]]$

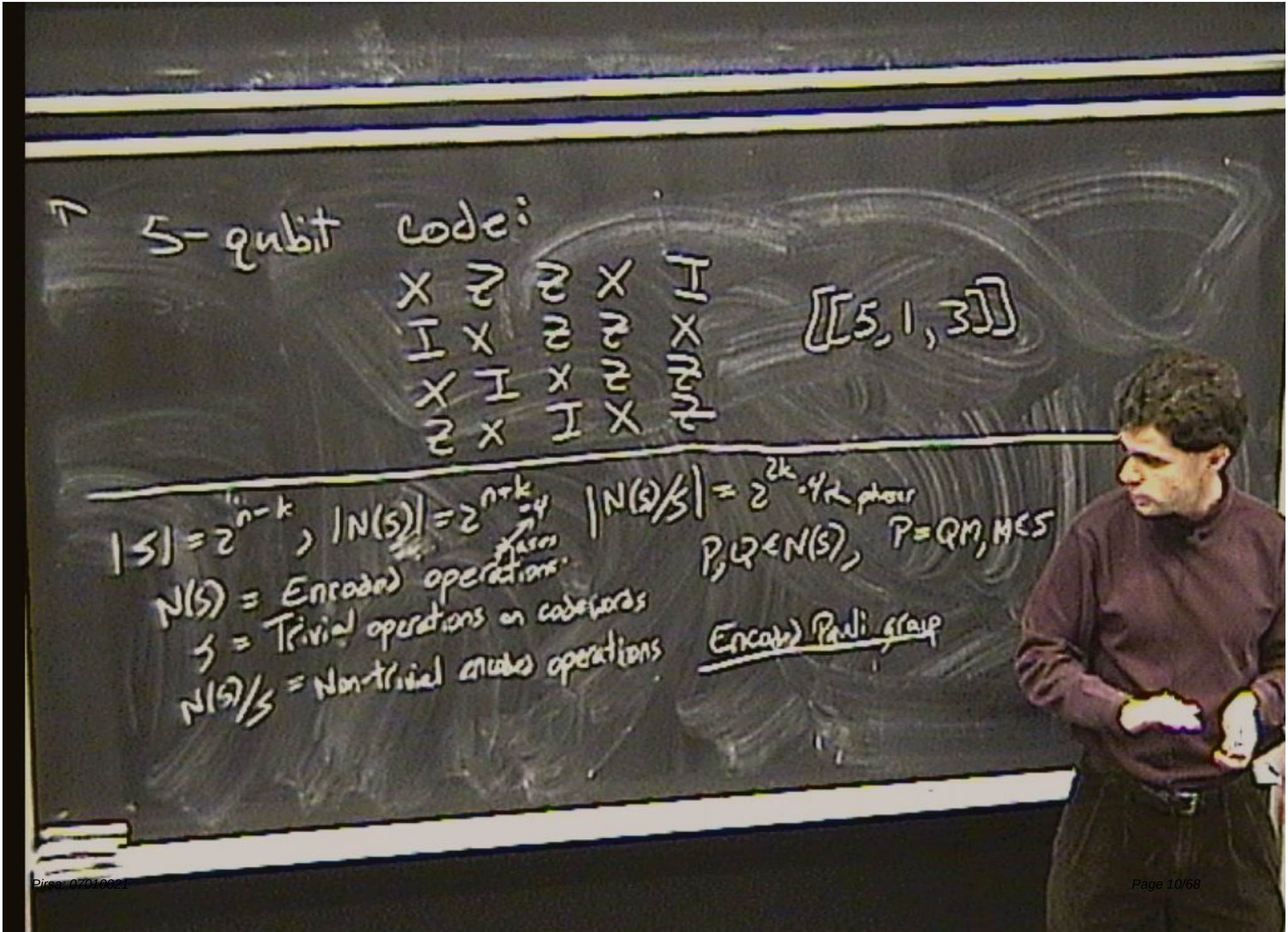
$|S| = 2^{n-k}$, $|N(S)| = 2^{n+k}$ $|N(S)/S| = 2^{2k}$,

$N(S)$ = Encoded operations.

S = Trivial operations on codewords

$N(S)/S$ = Non-trivial encoded operations

$P, Q \in N(S)$, $P = QM, M \in S$


5-qubit code:
X Z Z X I
I X Z Z X
X I X Z Z
Z X I X Z
[[5, 1, 3]]
|S| = 2^{n-k}, |N(S)| = 2^{n+k}, |N(S)/S| = 2^{2k} · 4^k phases
N(S) = Encoded operations
S = Trivial operations on codewords
N(S)/S = Non-trivial encoded operations
P, Q ∈ N(S), P = QM, M ∈ S
Encoded Pauli group

5-qubit code:

X	Z	Z	X	I
I	X	Z	Z	X
X	I	X	Z	Z
Z	X	I	X	Z

$[[5,1,3]]$

logical X: $\overline{X} = X \otimes X \otimes X \otimes X \otimes X$

logical Z: $\overline{Z} = Z \otimes Z \otimes Z \otimes Z \otimes Z$

$|S| = 2^{n-k}$, $|N(S)| = 2^{n+k}$ $\quad |N(S)/S| = 2^{2k}$ · 4^k phases

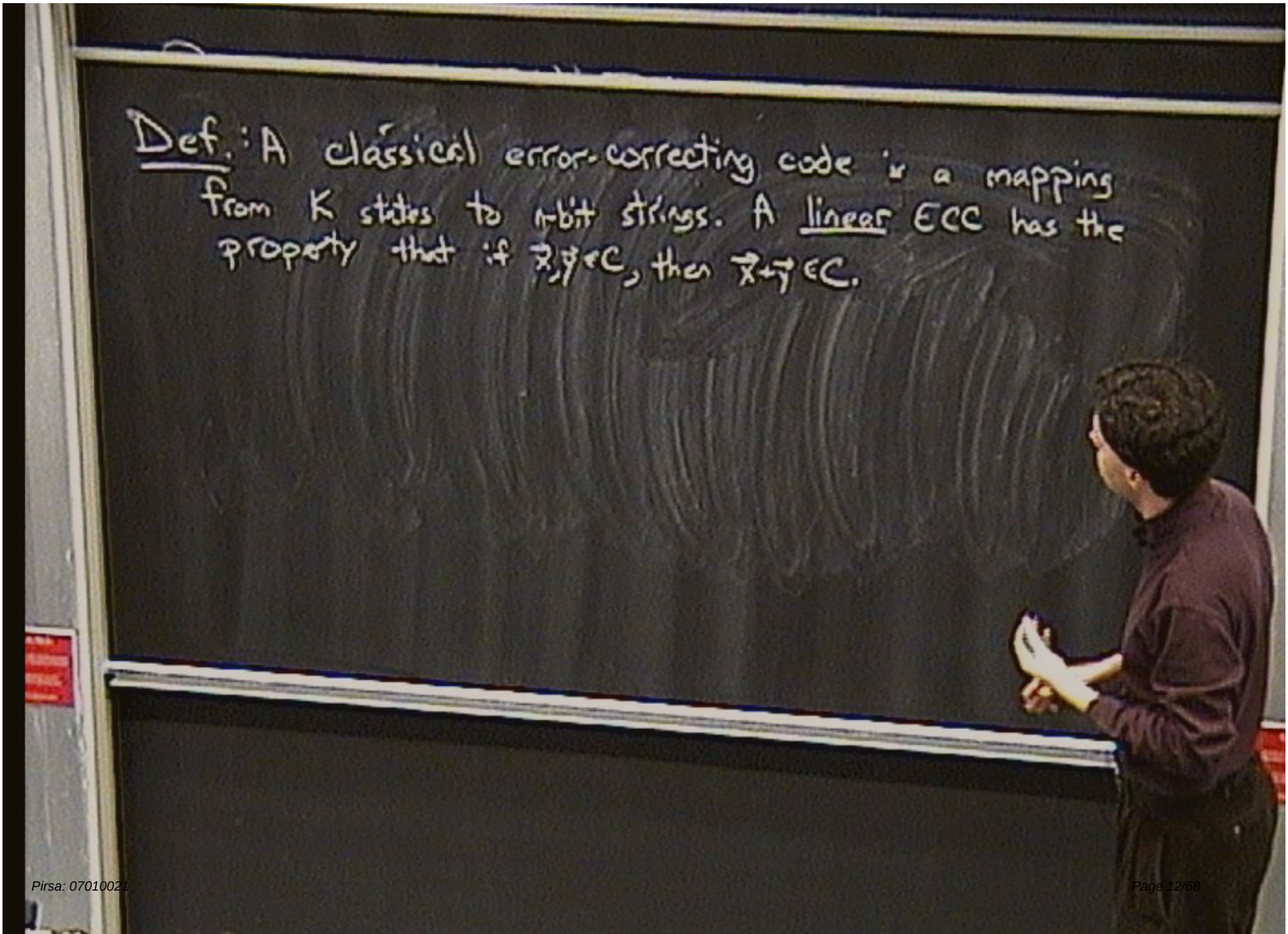
$N(S)$ = Encoded operations

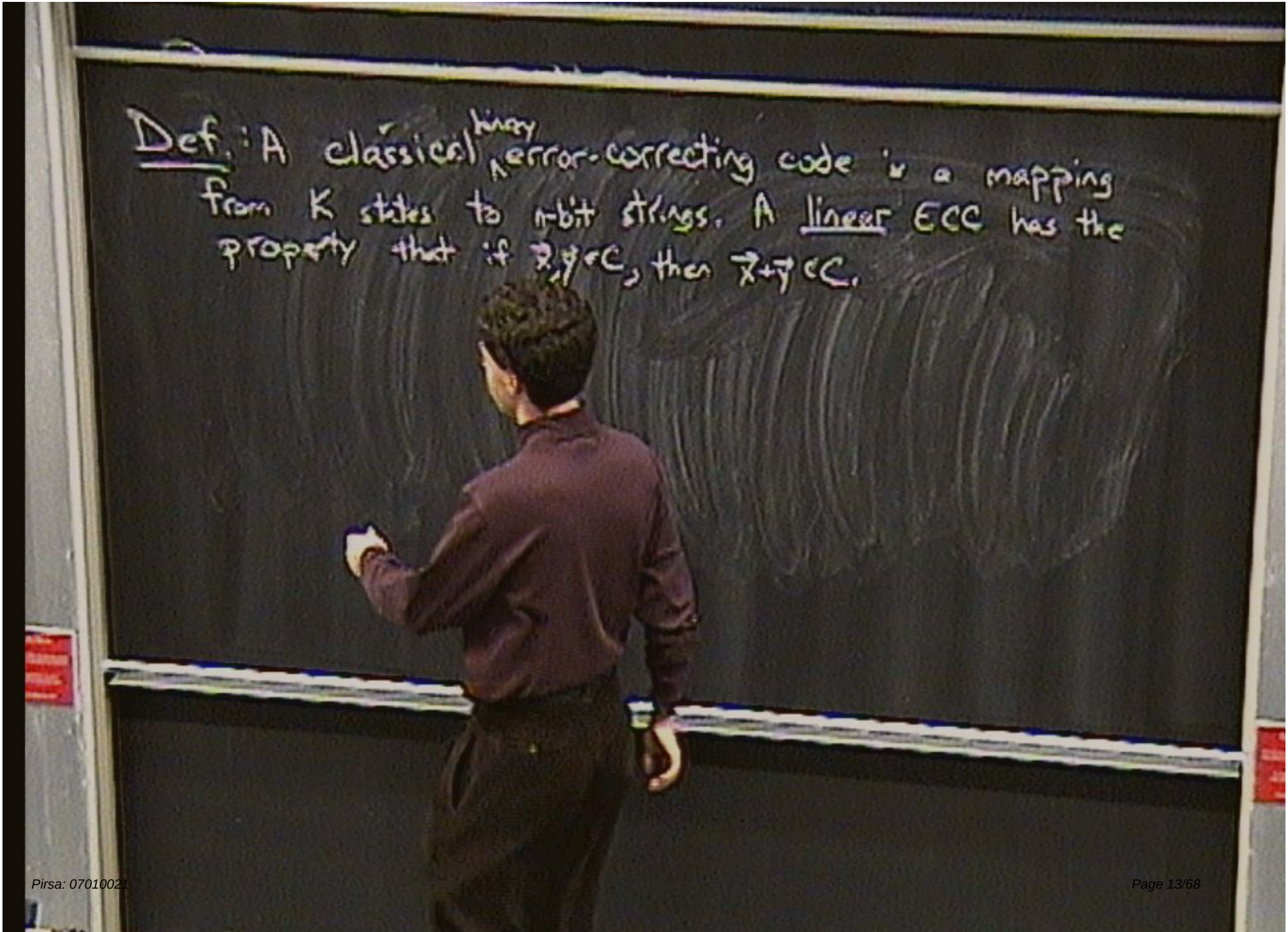
S = Trivial operations on codewords

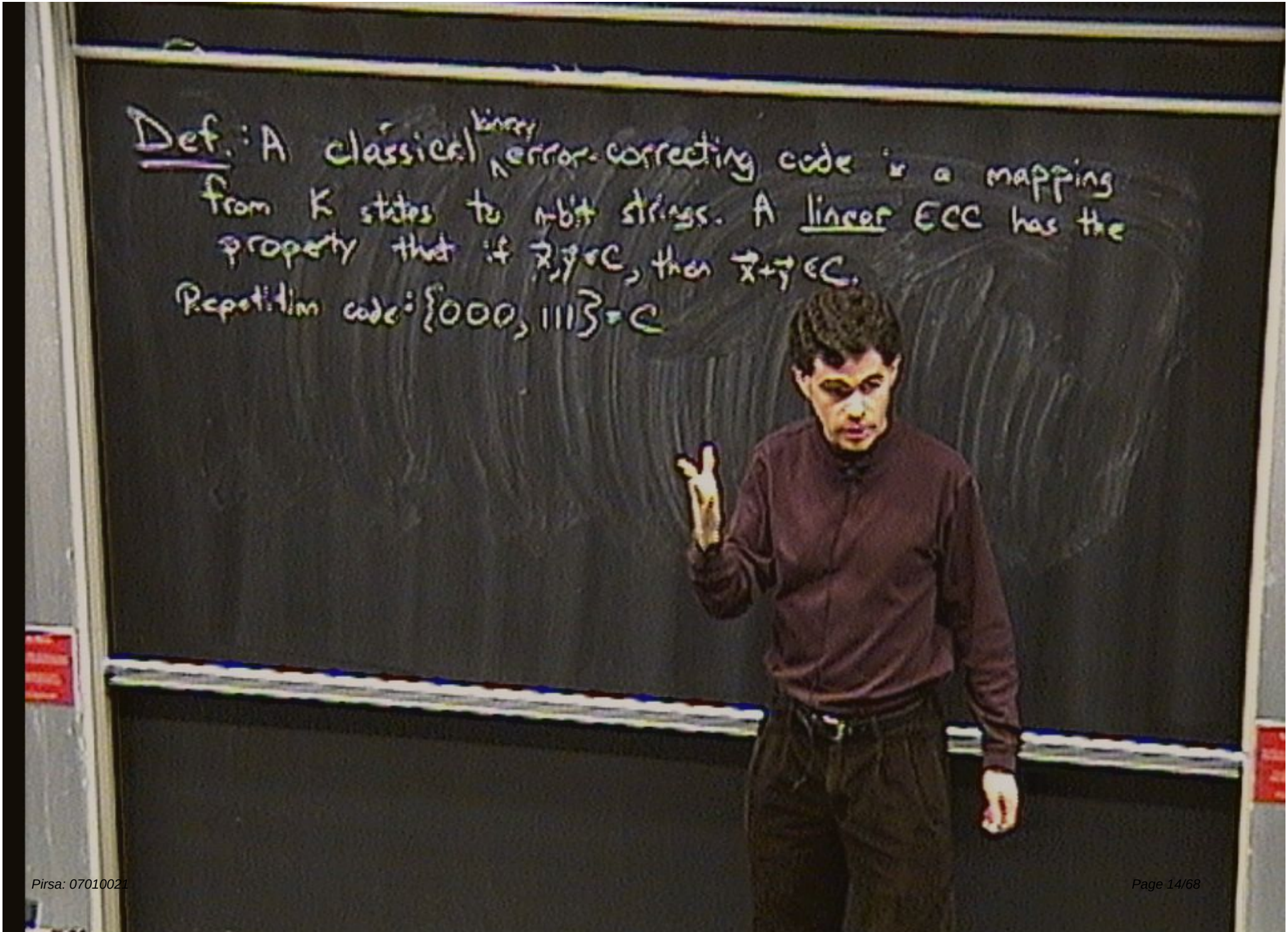
$N(S)/S$ = Nontrivial encoded operations

$P, Q \in N(S)$, $P = QM$, $M \in S$

Encoded Pauli group


Def: A classical error-correcting code is a mapping from K states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x}+\vec{y} \in C$.

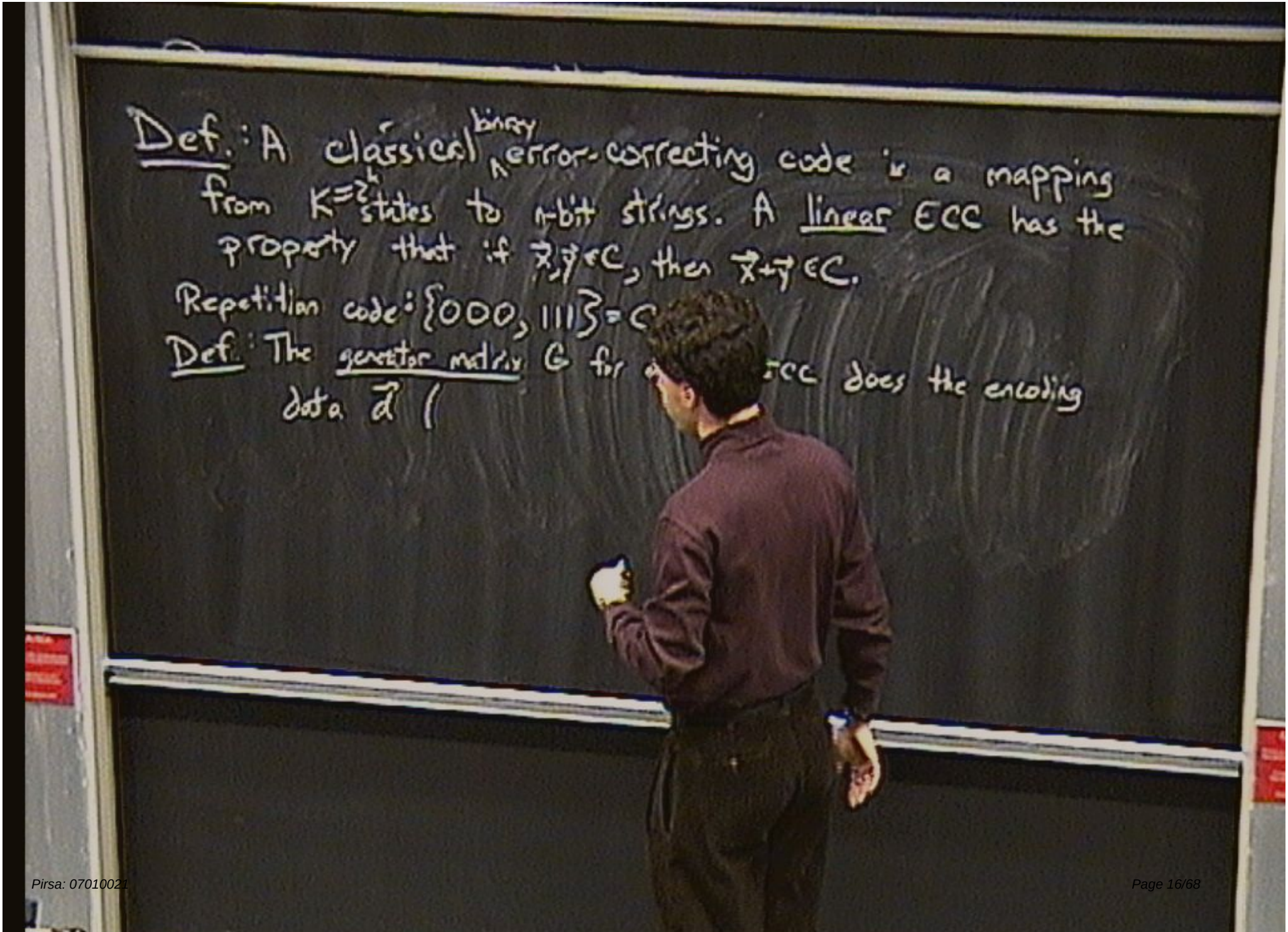

Def: A classical binary error-correcting code is a mapping
from K states to n-bit strings. A linear ECC has the
property that if x,y∈C, then x+y∈C.


Def: A classical (binary) error-correcting code is a mapping from K states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x}+\vec{y} \in C$.
Repetition code: $\{000, 111\} = C$

Def.: A classical (binary) error-correcting code is a mapping from K states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x}+\vec{y} \in C$.

Repetition code: $\{000, 111\} = C$

Def.: The generator matrix G for a linear ECC

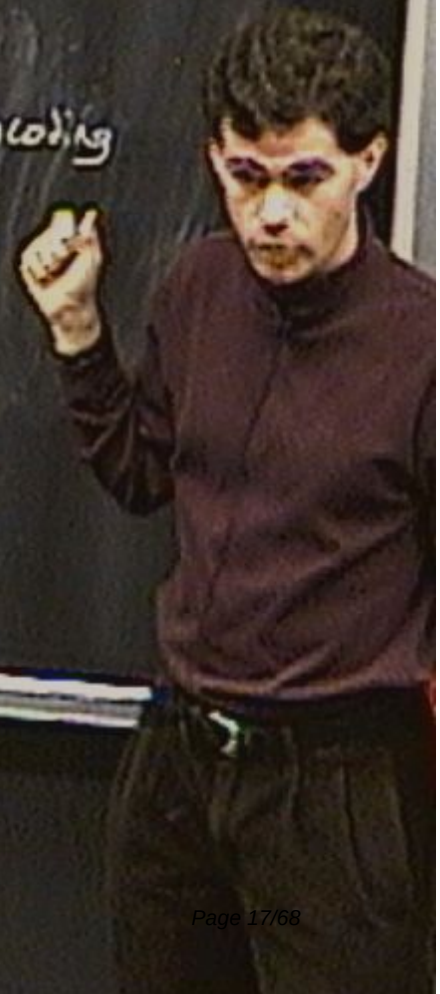

Def: A classical binary error-correcting code is a mapping
from $K=2^k$ states to n-bit strings. A linear ECC has the
property that if $\vec{x},\vec{y}\in C$, then $\vec{x}+\vec{y}\in C$.
Repetition code: {000, 111} = C
Def: The generator matrix G for
ECC does the encoding
data $\vec{d}$ (

Def.: A classical (binary) error-correcting code is a mapping from $K = 2^k$ states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x} + \vec{y} \in C$.

Repetition code: $\{000, 111\} = C$ $\quad G = (1\ 1\ 1)$

Def.: The generator matrix G for a linear ECC does the encoding data $\vec{d}$ (k-bit vector) to codeword $\vec{x} = G^T \vec{d}$

Rows of G are basis codewords



Def.: A classical (binary) error-correcting code is a mapping from $K = 2^k$ states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x} + \vec{y} \in C$.

Repetition code: $\{000, 111\} = C$ $\quad G = (1\ 1\ 1)$

Def. The generator matrix G for a linear ECC does the encoding data $\vec{d}$ (k-bit vector) to codeword $\vec{x} = G^T \vec{d}$

Rows of G are basis codewords

Def. The parity check matrix H is a maximal matrix s.t.

Def.: A classical binary error-correcting code is a mapping from $K = 2^k$ states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x}+\vec{y} \in C$.

Repetition code: $\{000, 111\} = C$ $\quad G = (1\ 1\ 1)$

Def. The generator matrix G for a linear ECC does the encoding data $\vec{d}$ (k-bit vector) to codeword $\vec{x} = G^T\vec{d}$

Rows of G are basis codewords

Def. The parity check matrix H is a maximal matrix s.t. $HG^T = 0$

$\vec{x} = G^T\vec{d} \Rightarrow H\vec{x} = HG^T\vec{d} = 0$

Def: A classical error-correcting code is a mapping from $K = 2^k$ states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x}+\vec{y} \in C$.

Repetition code: $\{000, 111\} = C$ $\quad G = (1\ 1\ 1)$

Def: The generator matrix G for a linear ECC does the encoding data $\vec{d}$ (k-bit vector) to codeword $\vec{x} = G^T\vec{d}$

Rows of G are basis codewords

Def: The parity check matrix H is a maximal matrix s.t. $HG^T = 0$

$\vec{x} = G^T\vec{d} \Rightarrow H\vec{x} = HG^T\vec{d} = 0$

For repetition code, $H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$

Def: A classical (binary) error-correcting code is a mapping from $K = 2^k$ states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x} + \vec{y} \in C$.

Repetition code: $\{000, 111\} = C$ $\quad G = (1\ 1\ 1)$

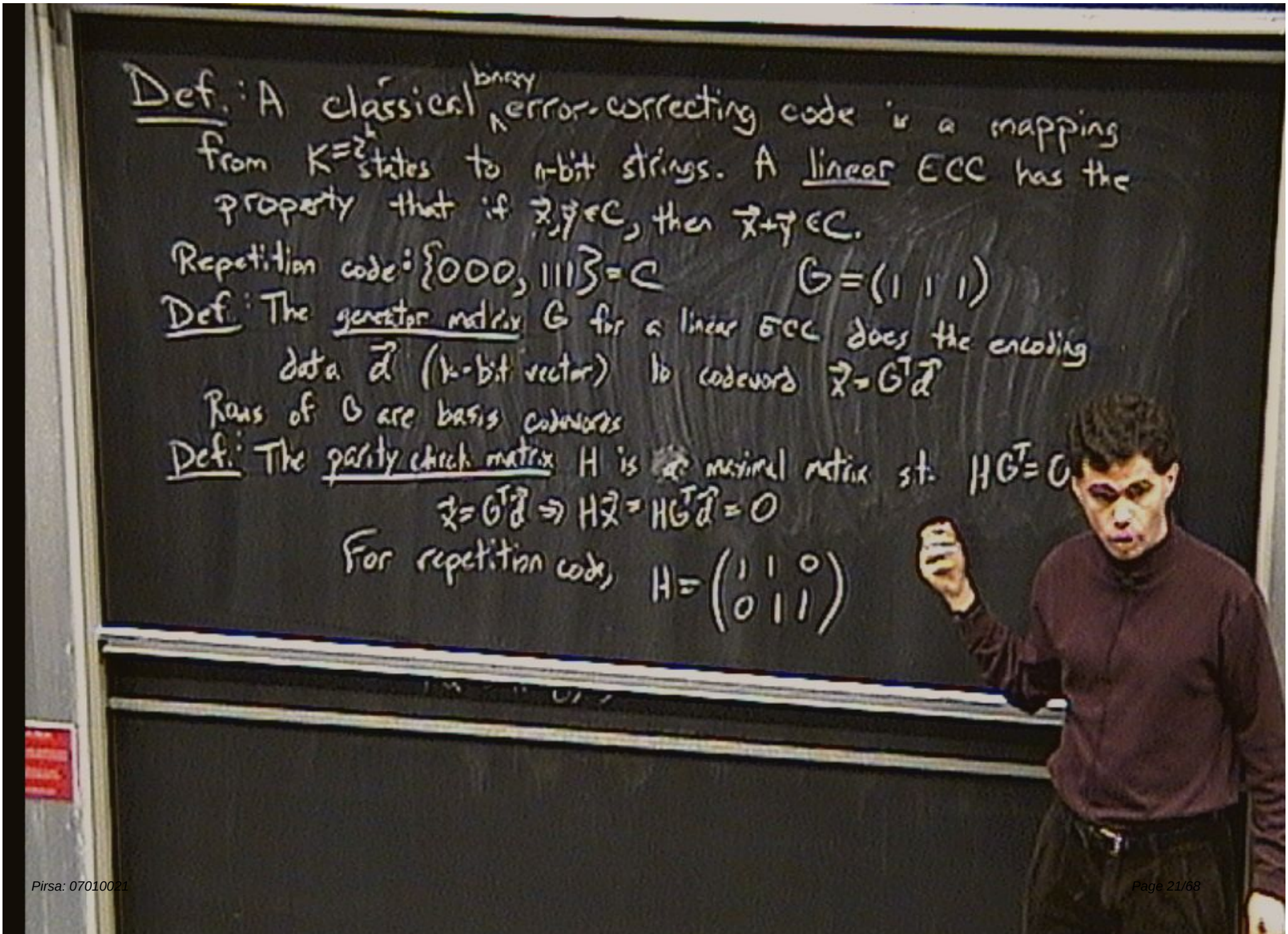
Def: The generator matrix G for a linear ECC does the encoding data $\vec{d}$ (k-bit vector) to codeword $\vec{x} = G^T \vec{d}$

Rows of G are basis codewords

Def: The parity check matrix H is a maximal matrix s.t. $HG^T = 0$

$\vec{x} = G^T \vec{d} \Rightarrow H\vec{x} = HG^T \vec{d} = 0$

For repetition code, $H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$



Def.: A classical (binary) error-correcting code is a mapping from $K = 2^k$ states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x}+\vec{y} \in C$.

Repetition code: $\{000, 111\} = C$ $\quad G = (1\ 1\ 1)$

Def: The generator matrix G for a linear ECC does the encoding data $\vec{d}$ (k-bit vector) to codeword $\vec{x} = G^T\vec{d}$

Rows of G are basis codewords

Def: The parity check matrix H is a maximal matrix $H\vec{x}^T = 0$

$$\vec{x} = G^T\vec{d} \Rightarrow H\vec{x} = HG^T\vec{d} = 0$$

For repetition code, $H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$

By linear algebra

from $K=2^k$ states to n-bit strings. A linear ECC has the property that if $\vec{x},\vec{y}\in C$, then $\vec{x}+\vec{y}\in C$.

Repetition code: $\{000, 111\}=C$ $\quad G=(1\ 1\ 1)$

Def: The generator matrix G for a linear ECC does the encoding data $\vec{d}$ (k-bit vector) to codeword $\vec{x}=G^T\vec{d}$

Rows of G are basis codewords

Def: The parity check matrix H is a maximal matrix s.t. $HG^T=0$

$\vec{x}=G^T\vec{d} \Rightarrow H\vec{x}=HG^T\vec{d}=0$

For repetition code, $H=\begin{pmatrix}1&1&0\\0&1&1\end{pmatrix}$

By linear algebra, G is $k\times n$ matrix, H is $(n-k)\times n$ matrix

Def: Symplectic inner product $P\cdot Q=P_x\cdot Q_z+Q_x\cdot P_z$ (mod 2 arithmetic) $\quad (Q_x|Q_z)$

$P\cdot Q=0$ iff $[P,Q]=0$

Def.: A classical binary error-correcting code is a mapping from $K = 2^k$ states to n-bit strings. A linear ECC has the property that if $\vec{x}, \vec{y} \in C$, then $\vec{x}+\vec{y} \in C$.

Repetition code: $\{000, 111\} = C$ $\quad G = (1\ 1\ 1)$

Def.: The generator matrix G for a linear ECC does the encoding data $\vec{d}$ (k-bit vector) to codeword $\vec{x} = G^T\vec{d}$

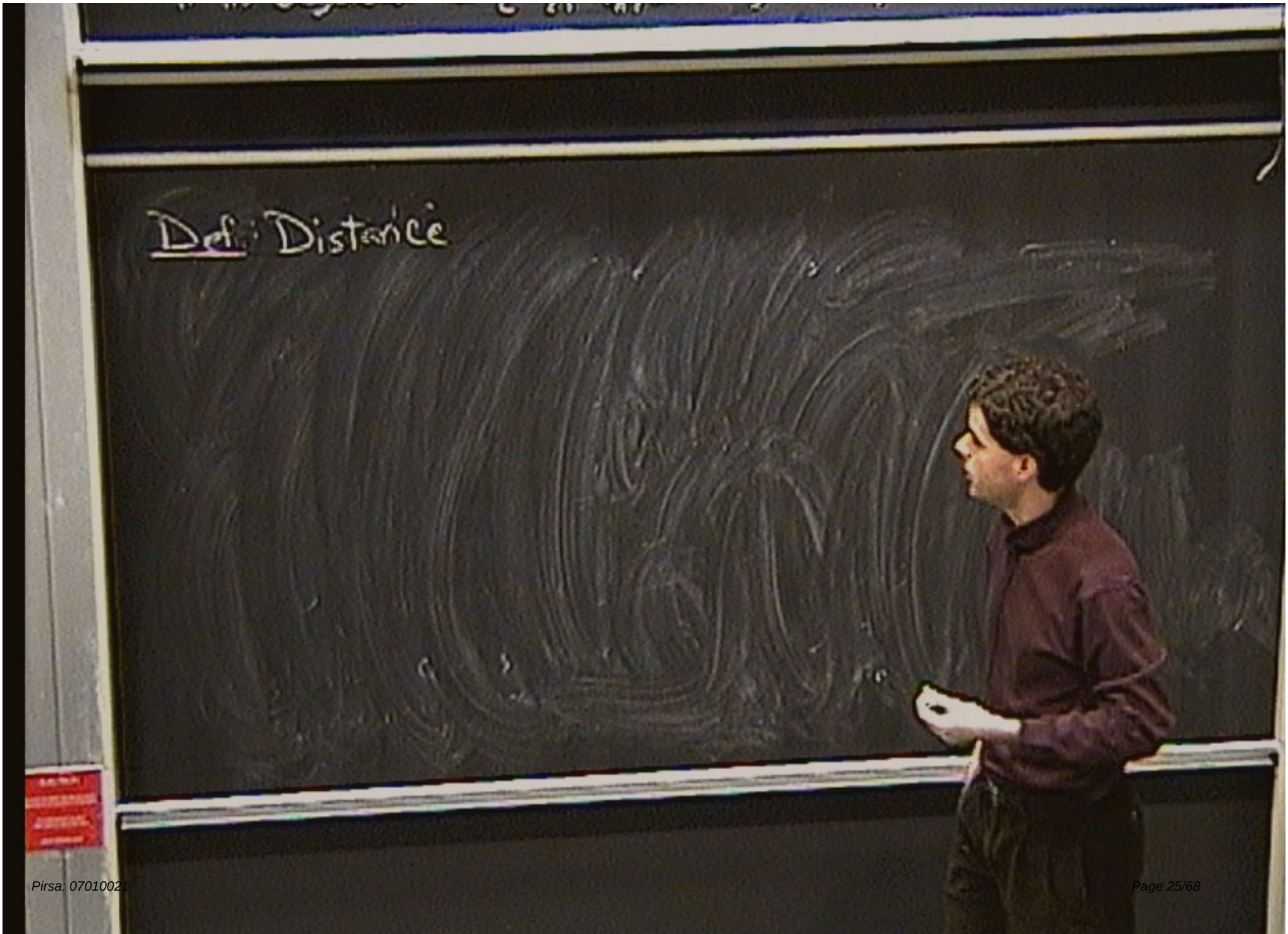
Rows of G are basis codewords

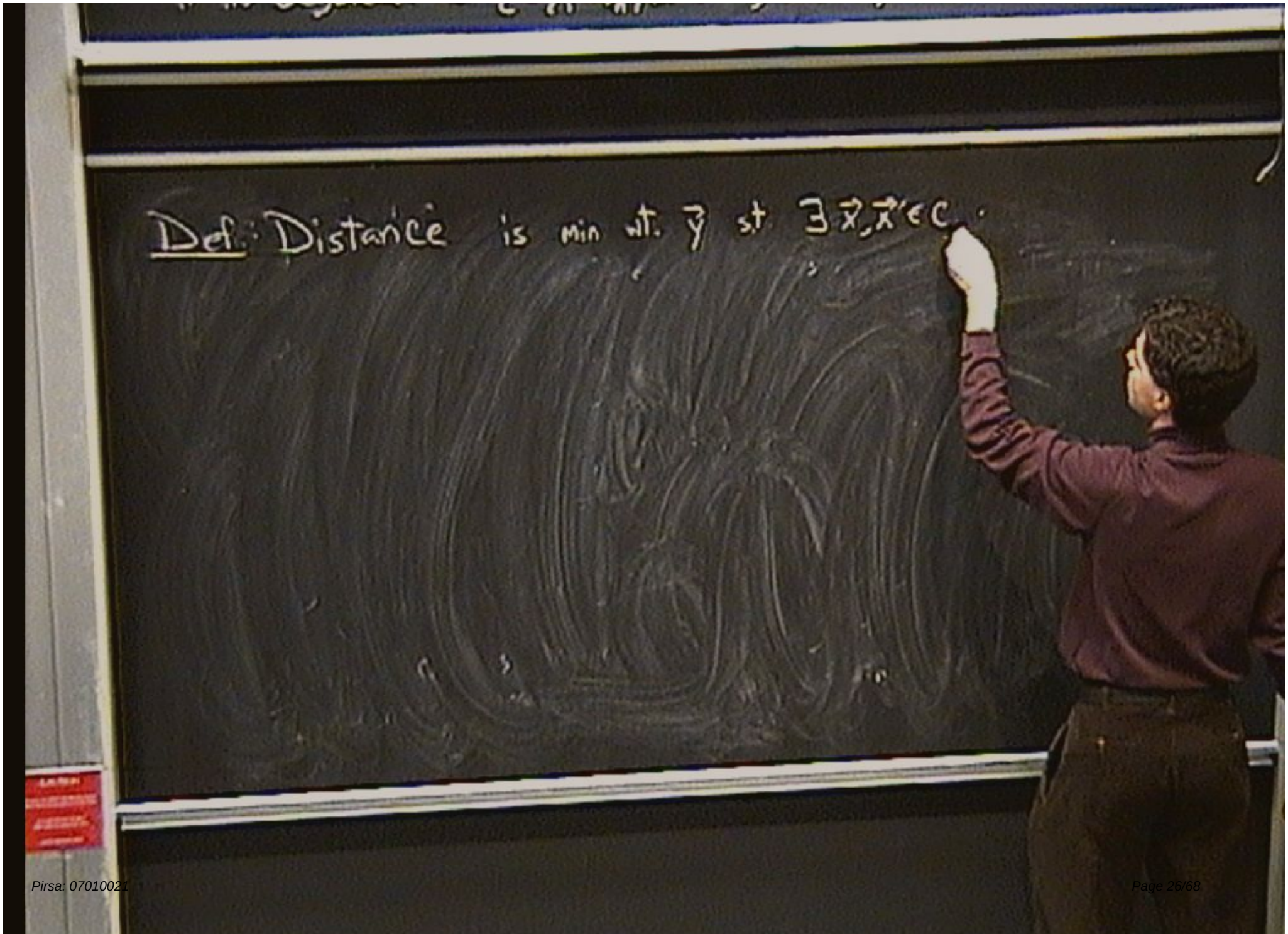
Def.: The parity check matrix H is a maximal matrix s.t. $HG^T = 0$

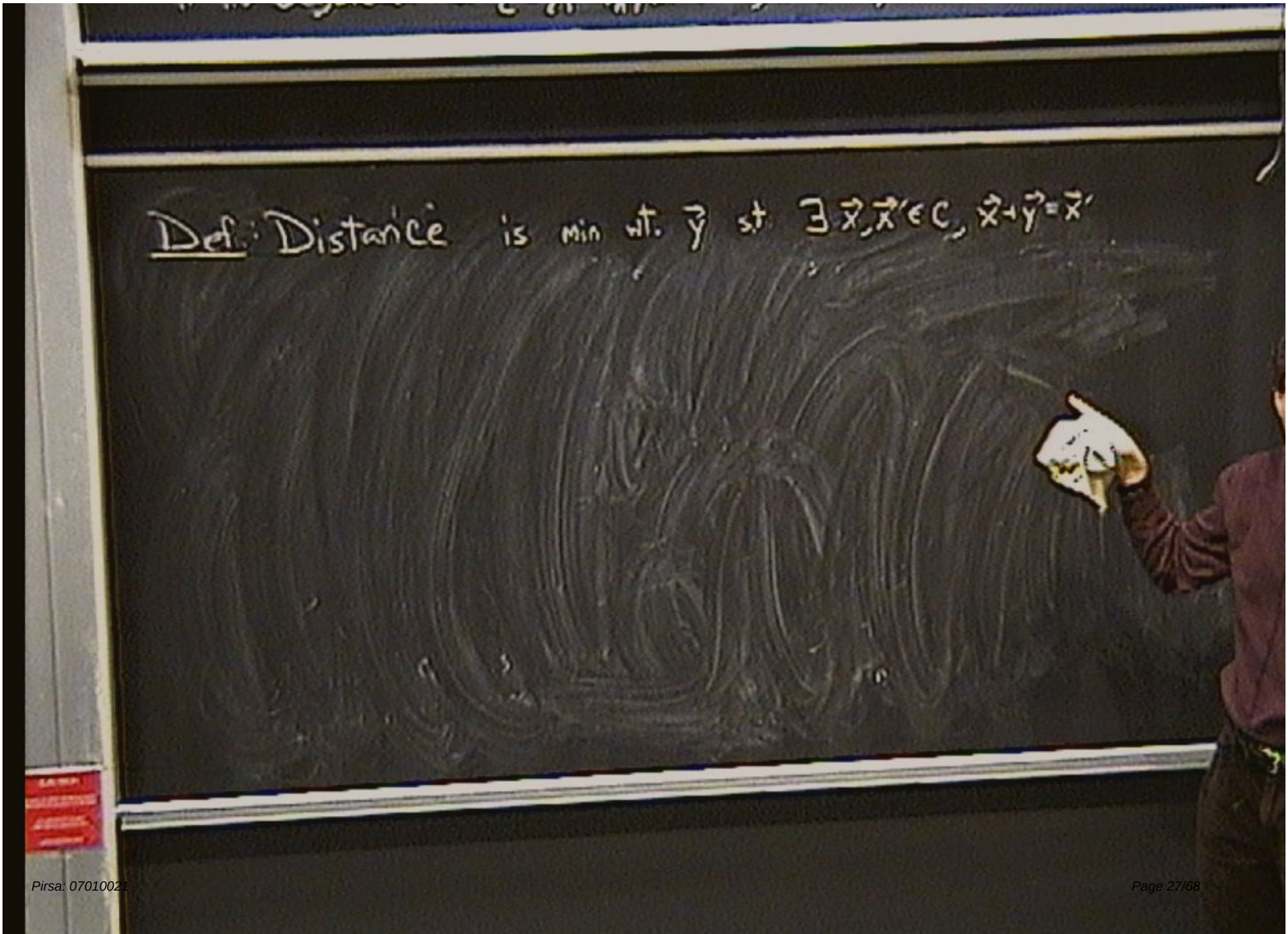
$$\vec{x} = G^T\vec{d} \Rightarrow H\vec{x} = HG^T\vec{d} = 0$$

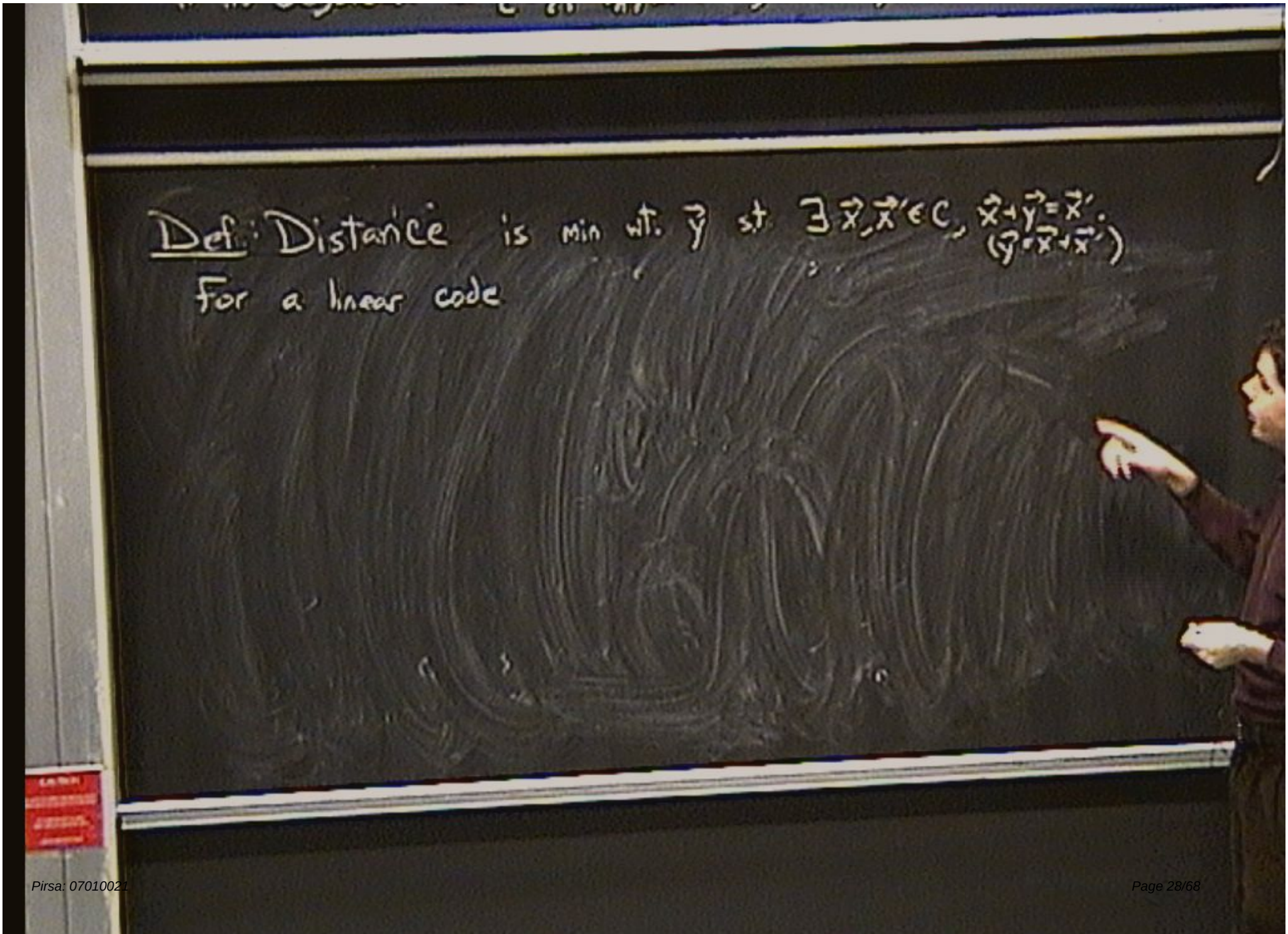
For repetition code, $H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$

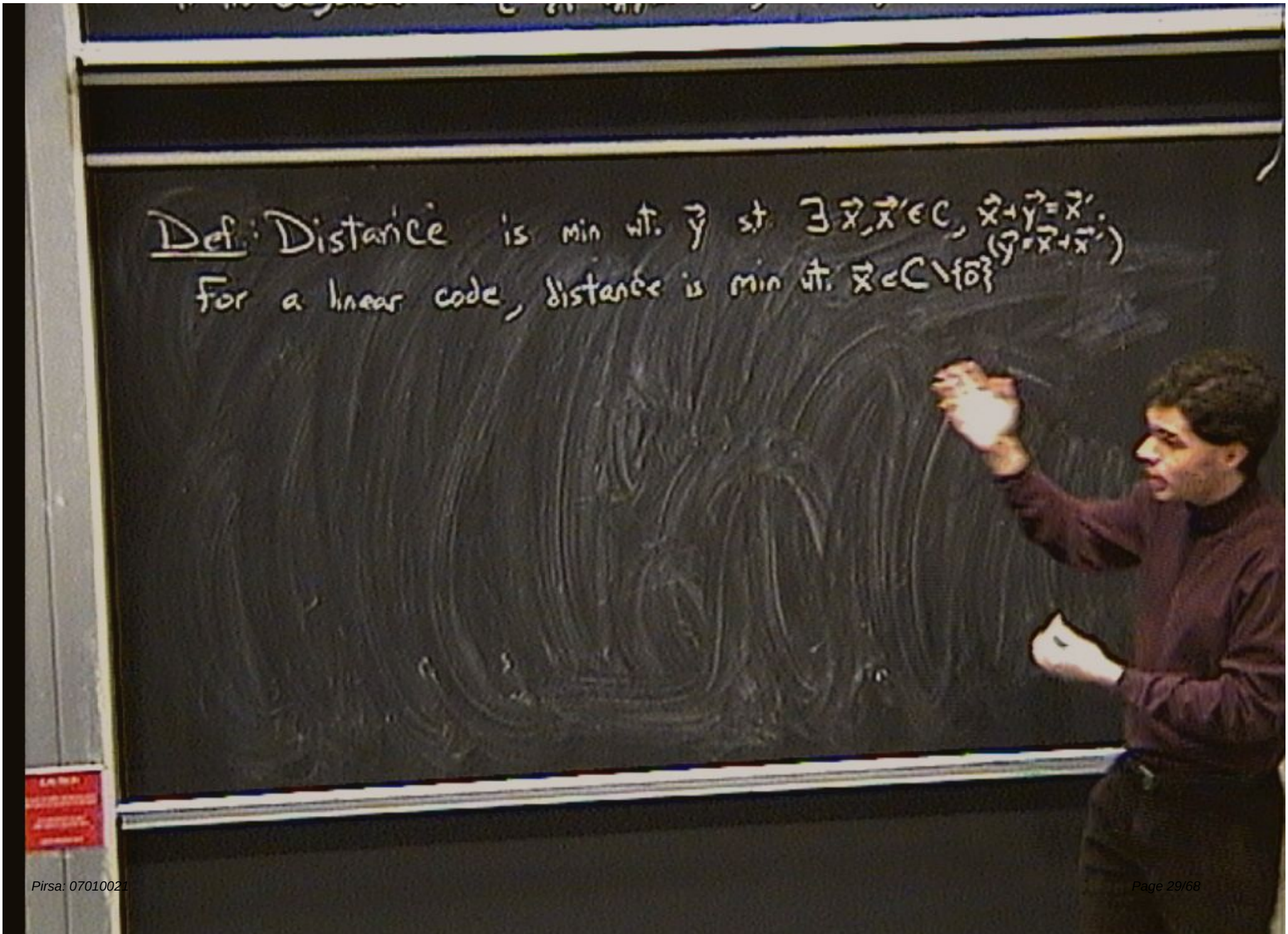
By linear algebra, G is $k \times n$ matrix, H is $(n-k) \times n$ matrix

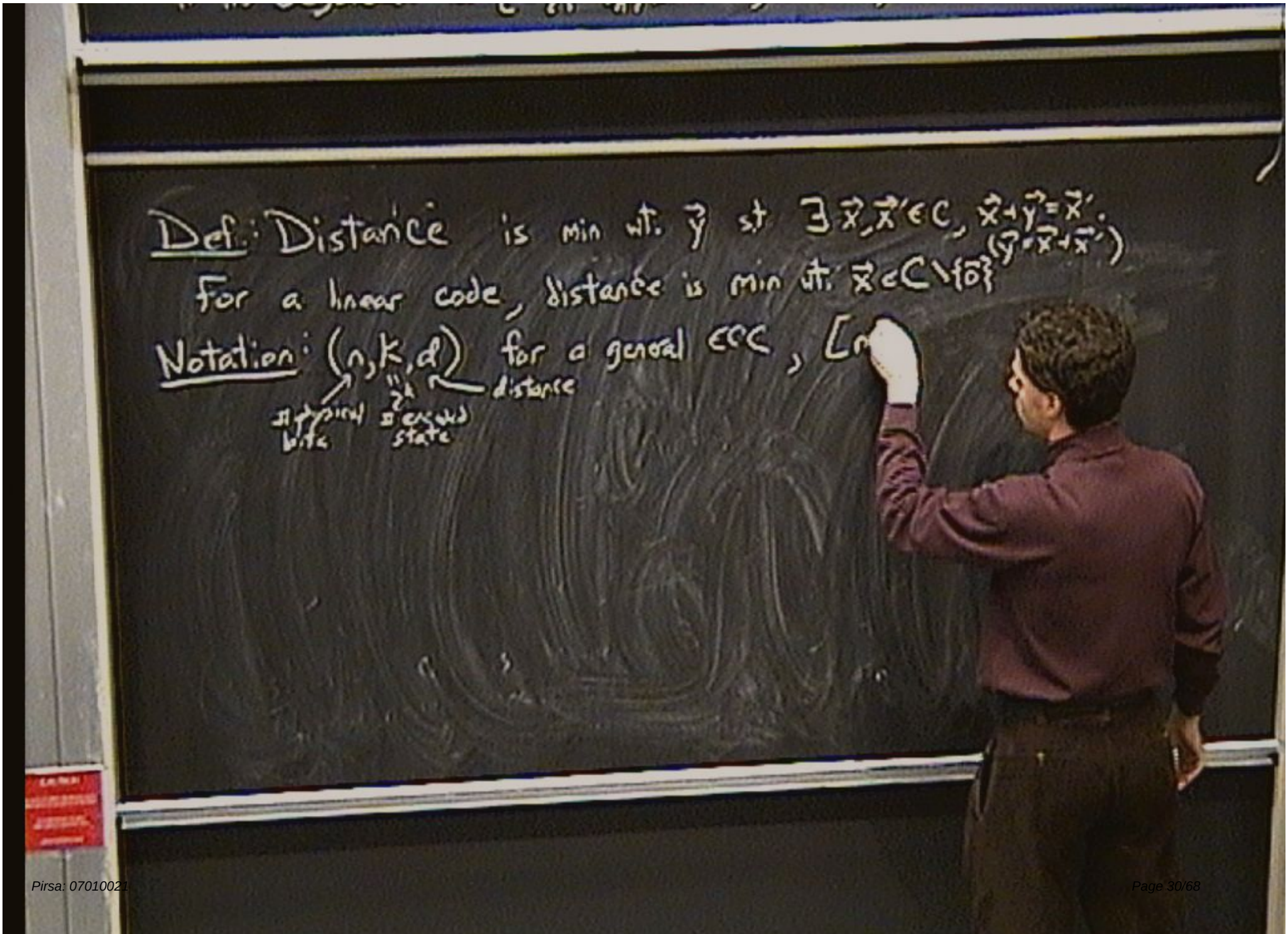

Def: Distance


Def: Distance is min wt. γ st ∃ x⃗, x⃗' ∈ C.


Def: Distance is min wt. $\vec{y}$ st $\exists \vec{x}, \vec{x}' \in C$, $\vec{x}+\vec{y}=\vec{x}'$


Def: Distance is min wt. $\vec{y}$ st. $\exists \vec{x}, \vec{x}' \in C$, $\vec{x}+\vec{y}=\vec{x}'$.
$(\vec{y}=\vec{x}+\vec{x}')$
For a linear code


Def: Distance is min wt. $\vec{y}$ s.t. $\exists \vec{x}, \vec{x}' \in C$, $\vec{x} + \vec{y} = \vec{x}'$.
$(\vec{y} = \vec{x} + \vec{x}')$
For a linear code, distance is min wt. $\vec{x} \in C \setminus \{0\}$


Def: Distance is min wt. $\vec{y}$ s.t. $\exists \vec{x}, \vec{x}' \in C$, $\vec{x} + \vec{y} = \vec{x}'$.
$(\vec{y} = \vec{x} + \vec{x}')$
For a linear code, distance is min wt. $\vec{x} \in C \setminus \{0\}$
Notation: (n, k, d) for a general ecc, [n
n = # physical bits
2^k = # encoded state
d = distance

Def: Distance is min wt. $\vec{y}$ s.t. $\exists \vec{x}, \vec{x}' \in C$, $\vec{x} + \vec{y} = \vec{x}'$. $(\vec{y} = \vec{x} + \vec{x}')$

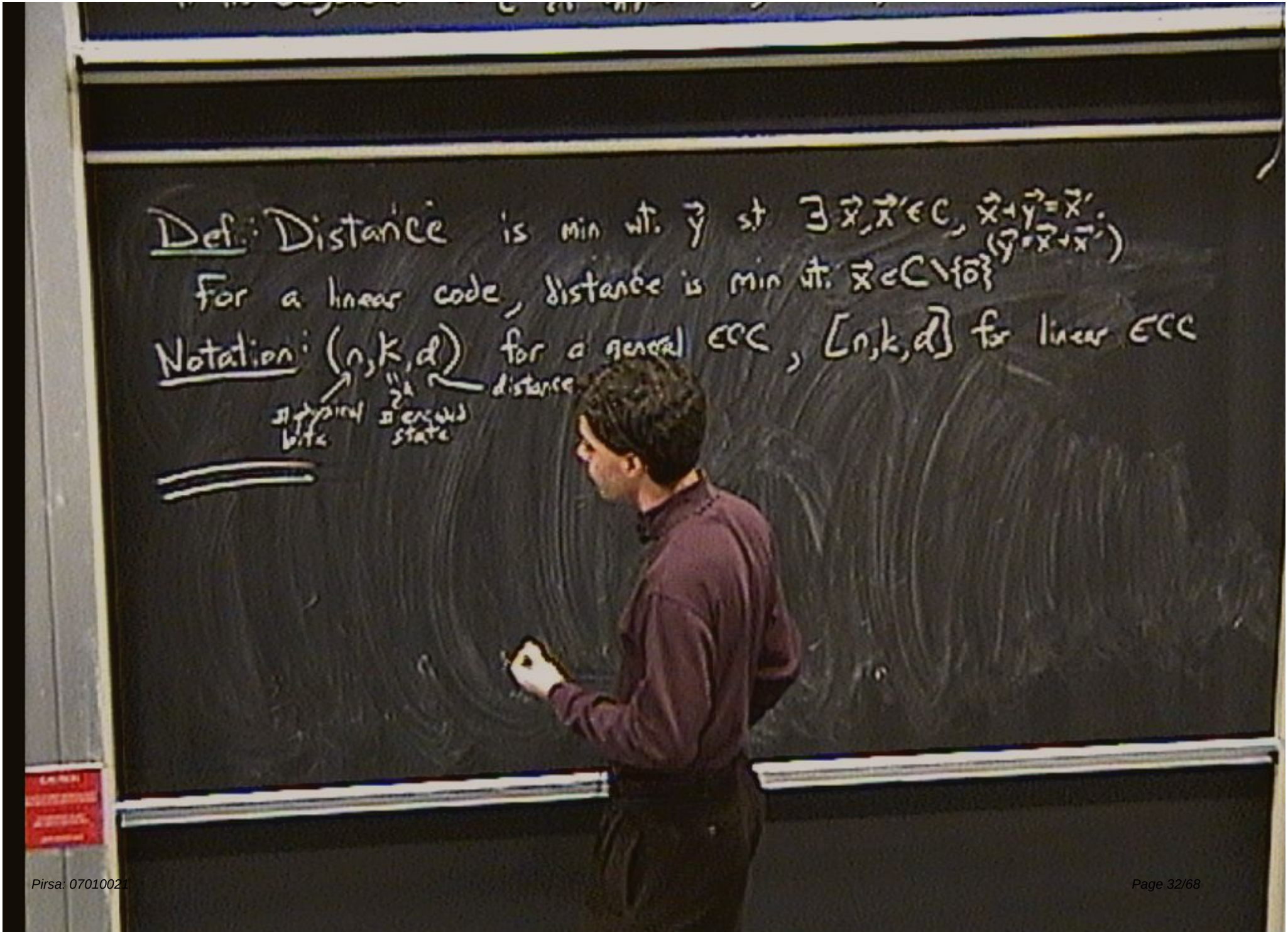
For a linear code, distance is min wt. $\vec{x} \in C \setminus \{0\}$

Notation: (n, K, d) for a general ECC, $[n, k, d]$ for linear ECC

n = # physical bits

$K = 2^k$ = # encoded states

d = distance


Def: Distance is min wt. $\vec{y}$ st $\exists \vec{x}, \vec{x}' \in C$, $\vec{x} + \vec{y} = \vec{x}'$; ($\vec{y} = \vec{x} + \vec{x}'$)
For a linear code, distance is min wt. $\vec{x} \in C \setminus \{0\}$
Notation: (n, K, d) for a general ECC, $[n, k, d]$ for linear ECC
n = # physical bits
$K = 2^k$ encoded states
d = distance

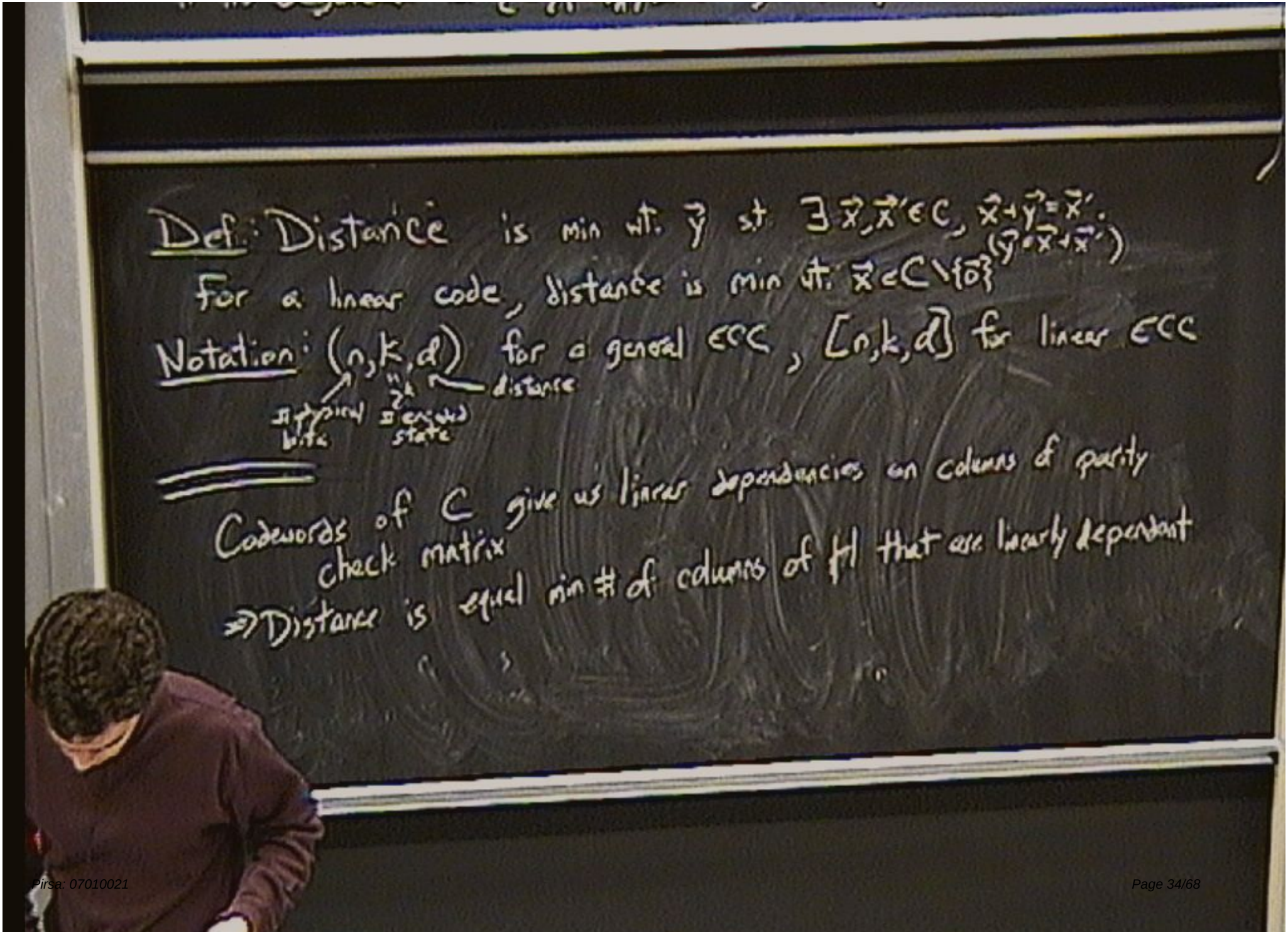
Def: Distance is min wt. $\vec{y}$ s.t. $\exists \vec{x}, \vec{x}' \in C$, $\vec{x} + \vec{y} = \vec{x}'$. ($\vec{y} = \vec{x} + \vec{x}'$)

For a linear code, distance is min wt. $\vec{x} \in C \setminus \{0\}$

Notation: (n, k, d) for a general ECC, $[n, k, d]$ for linear ECC

n: # physical bits; k: # encoded states; d: distance

Codewords of C give us linear dependencies on columns of parity check matrix


Def: Distance is min wt. $\vec{y}$ s.t. $\exists \vec{x}, \vec{x}' \in C$, $\vec{x} + \vec{y} = \vec{x}'$. ($\vec{y} \neq \vec{x}, \vec{x}'$)
For a linear code, distance is min wt. $\vec{x} \in C \setminus \{0\}$
Notation: (n, k, d) for a general $\in CC$, $[n, k, d]$ for linear $\in CC$
physical bits
encoded states
distance
Codewords of C give us linear dependencies on columns of parity check matrix
$\Rightarrow$ Distance is equal min # of columns of H that are linearly dependent

Def: Distance is min wt. $\vec{y}$ s.t. $\exists \vec{x}, \vec{x}' \in C$, $\vec{x} + \vec{y} = \vec{x}'$. ($\vec{y} = \vec{x} + \vec{x}'$)

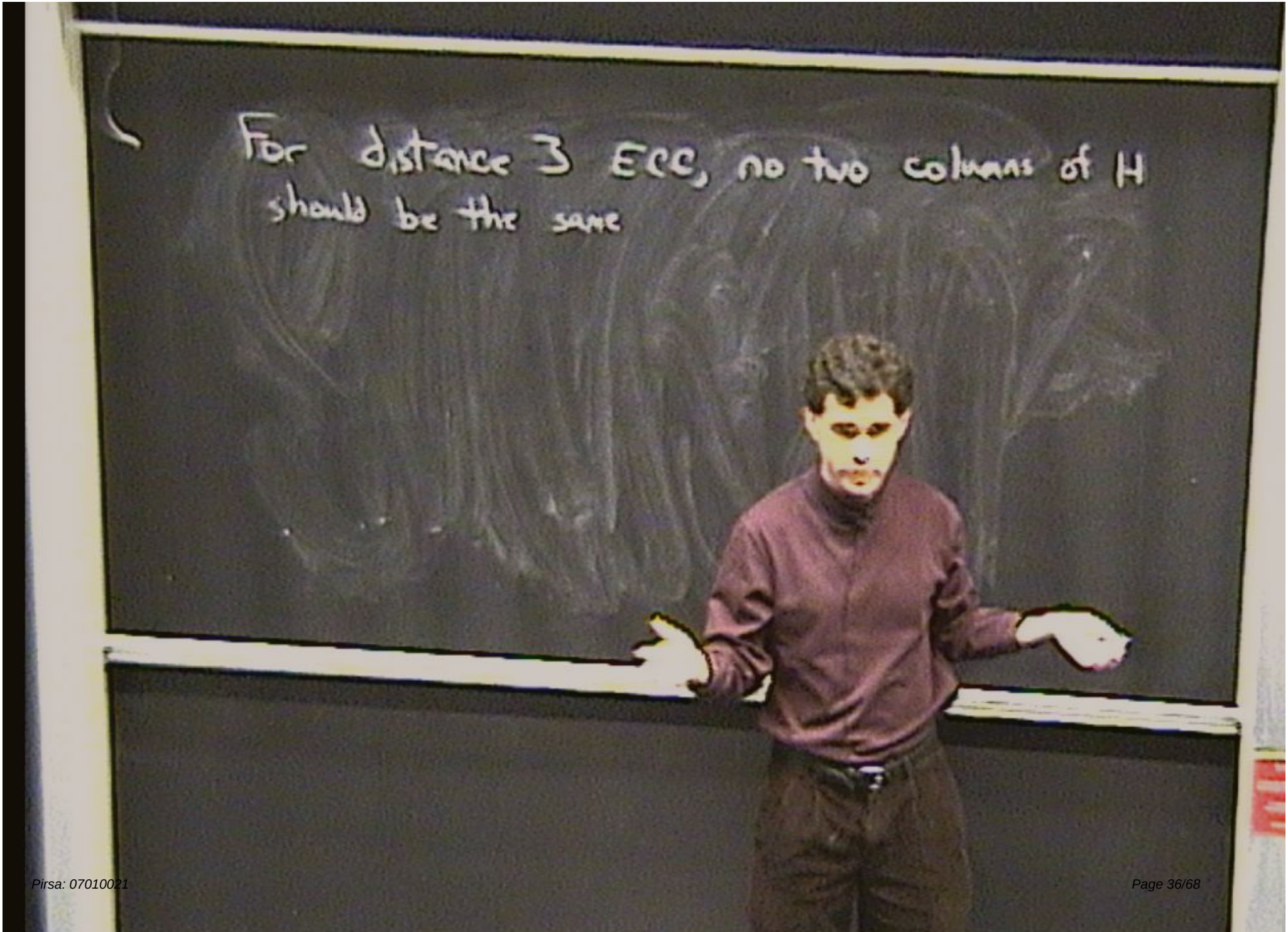
For a linear code, distance is min wt. $\vec{x} \in C \setminus \{0\}$

Notation: (n, K, d) for a general ECC, $[n, k, d]$ for linear ECC

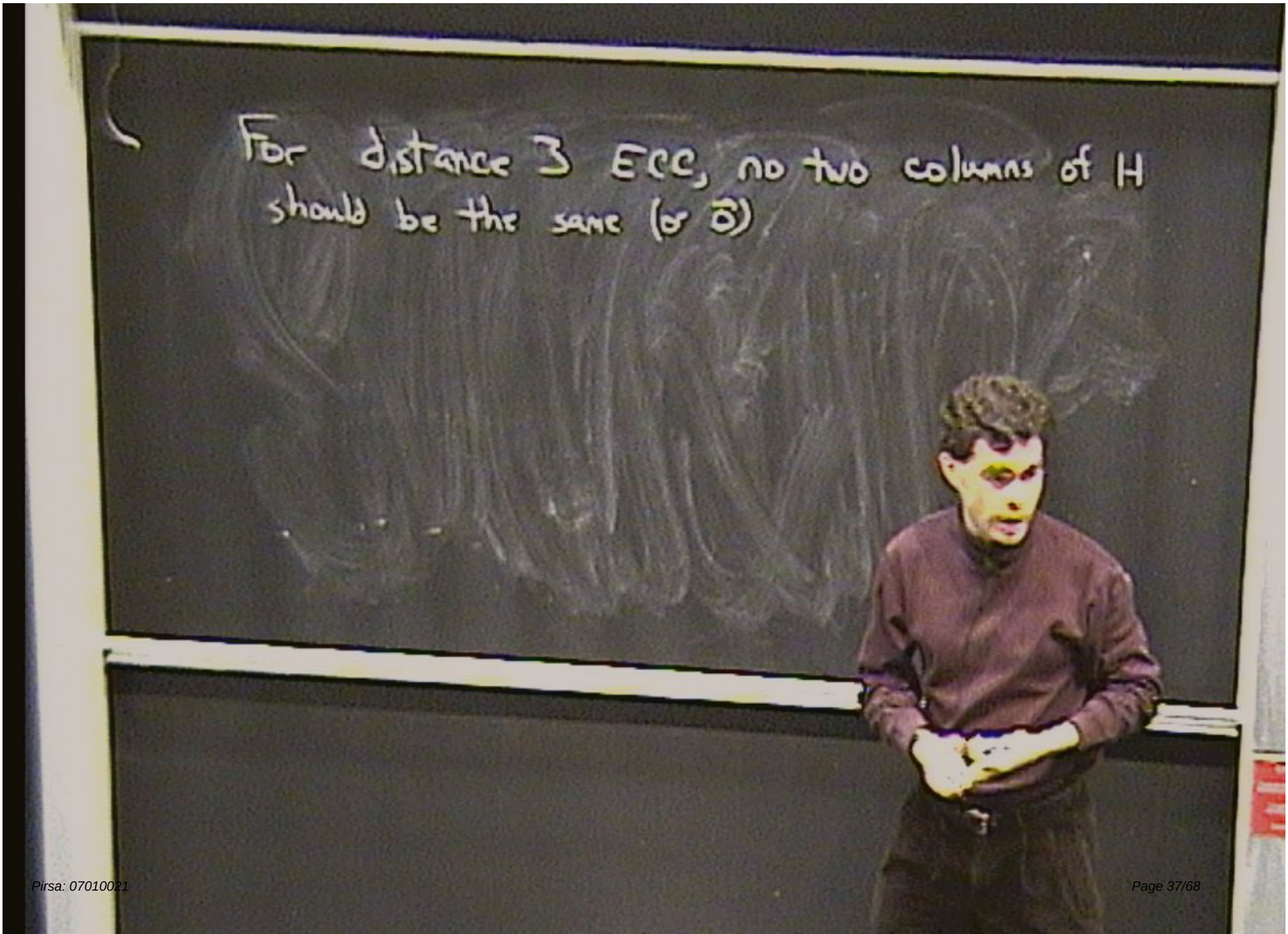
n = # physical bits; K = # encoded states; d = distance

Codewords of C give us linear dependencies on columns of parity check matrix

$\Rightarrow$ Distance is equal min # of columns of H that are linearly dependent


For distance 3 ECC, no two columns of H
should be the same

For distance 3 ECC, no two columns of H should be the same (or 0)

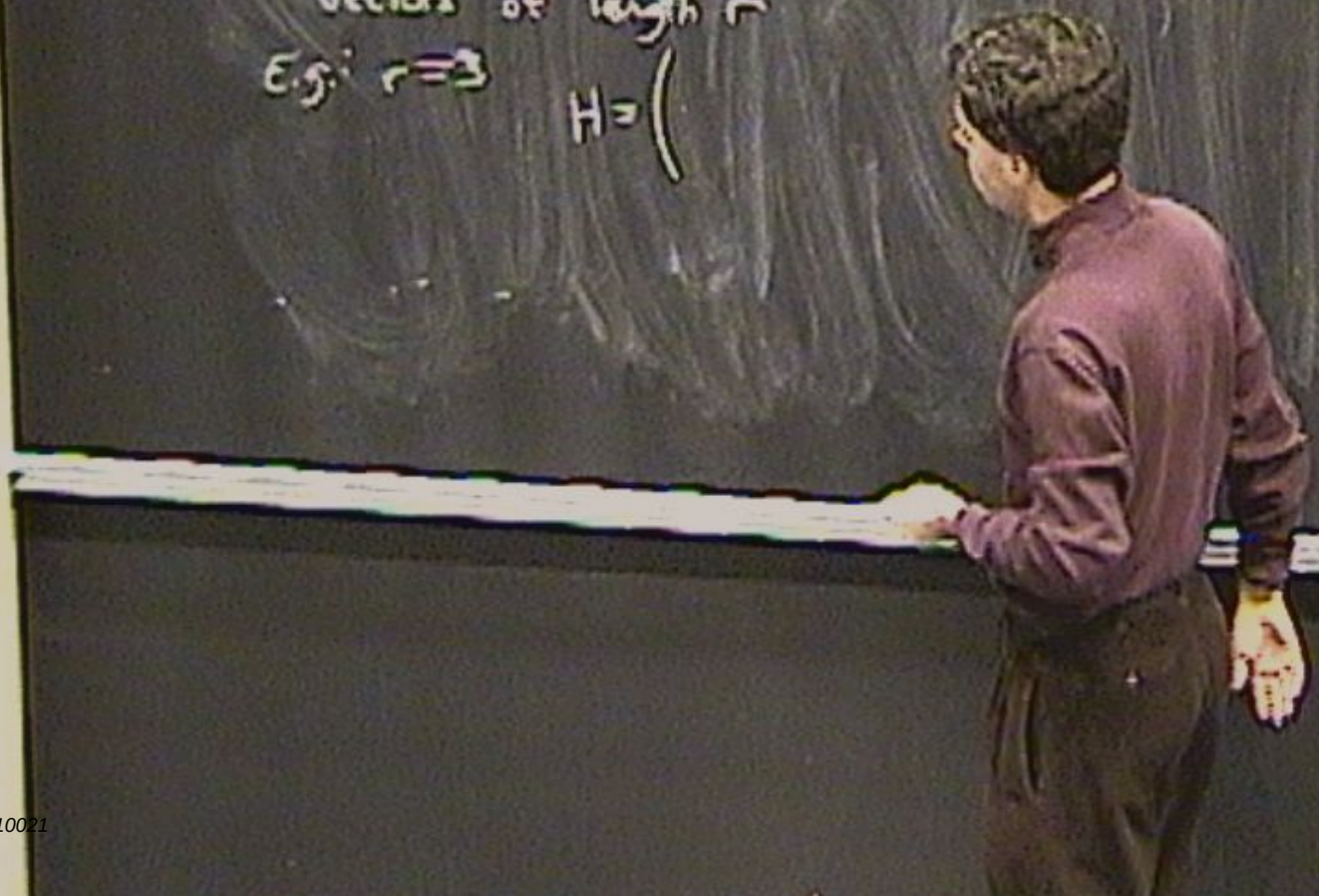


For distance 3 ECC, no two columns of H should be the same (or 0)

Hamming codes: Choose r, let columns be all distinct vectors of length r

E.g.: $r=3$

$H=($



For distance 3 ECC, no two columns of H should be the same (or 0).

Hamming codes: Choose r, let columns be all distinct vectors of length r

E.g.: $r=3$

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

For distance 3 ECC, no two columns of H should be the same (or $\vec{0}$).

Hamming codes: Choose r, let columns be all distinct vectors of length r

$n = 2^r - 1$

$k = n - r$

E.g.: $r = 3$

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

For distance 3 ECC, no two columns of H should be the same (or $\vec{0}$).

Hamming codes: Choose r, let columns be all distinct vectors of length r

$n = 2^r - 1$

$k = n - r$

$[7,4,3]$

Ex: $r = 3$

$$H = \left(\begin{array}{c|c|c|c} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{array}\right.\left.\begin{array}{c|c|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right)$$

$G = ($

For distance 3 ECC, no two columns of H should be the same (or $\vec{0}$)

Hamming codes: Choose r, let columns be all distinct vectors of length r

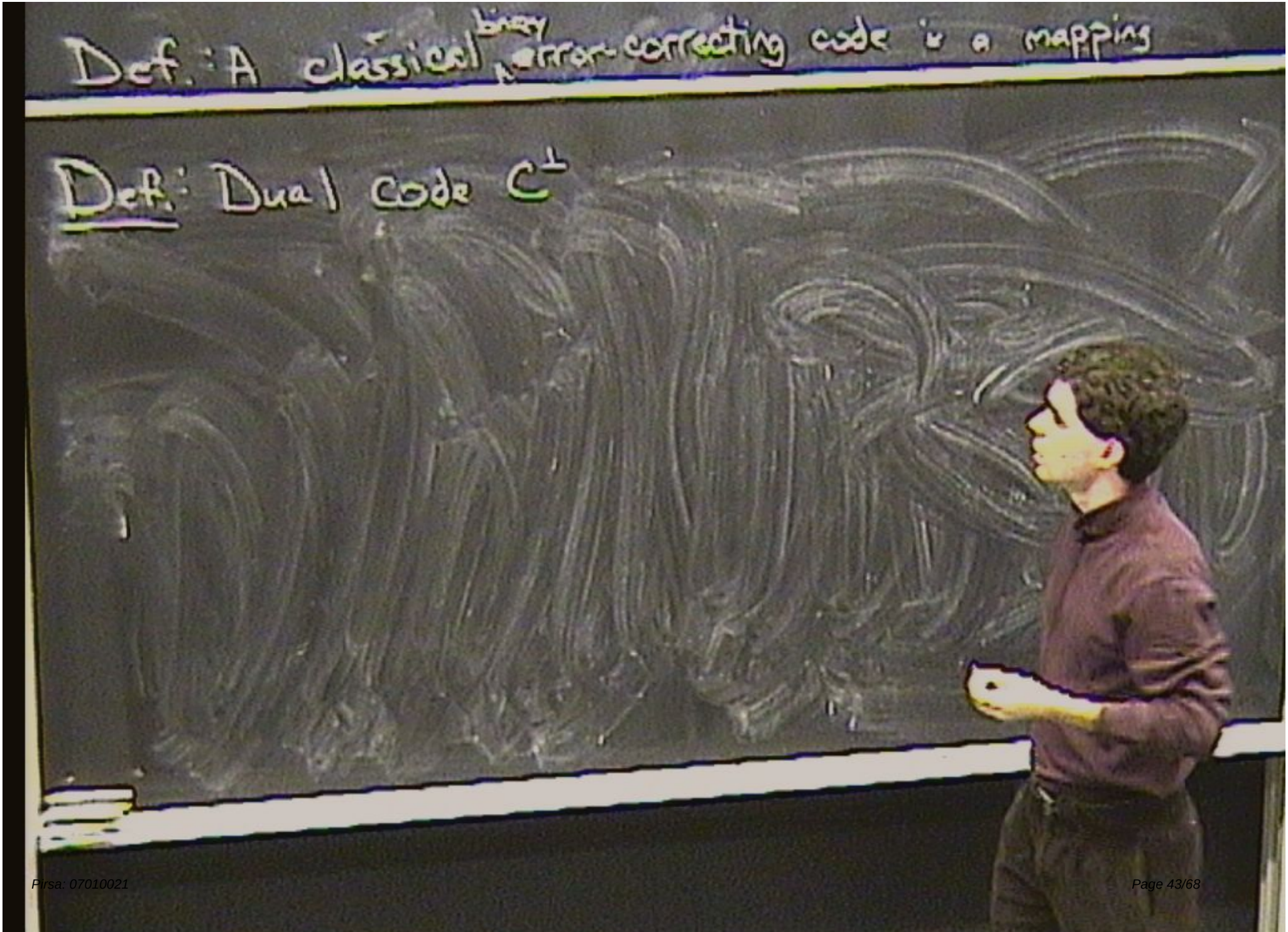
$n = 2^r - 1$

$k = n - r$

E.g.: $r = 3$

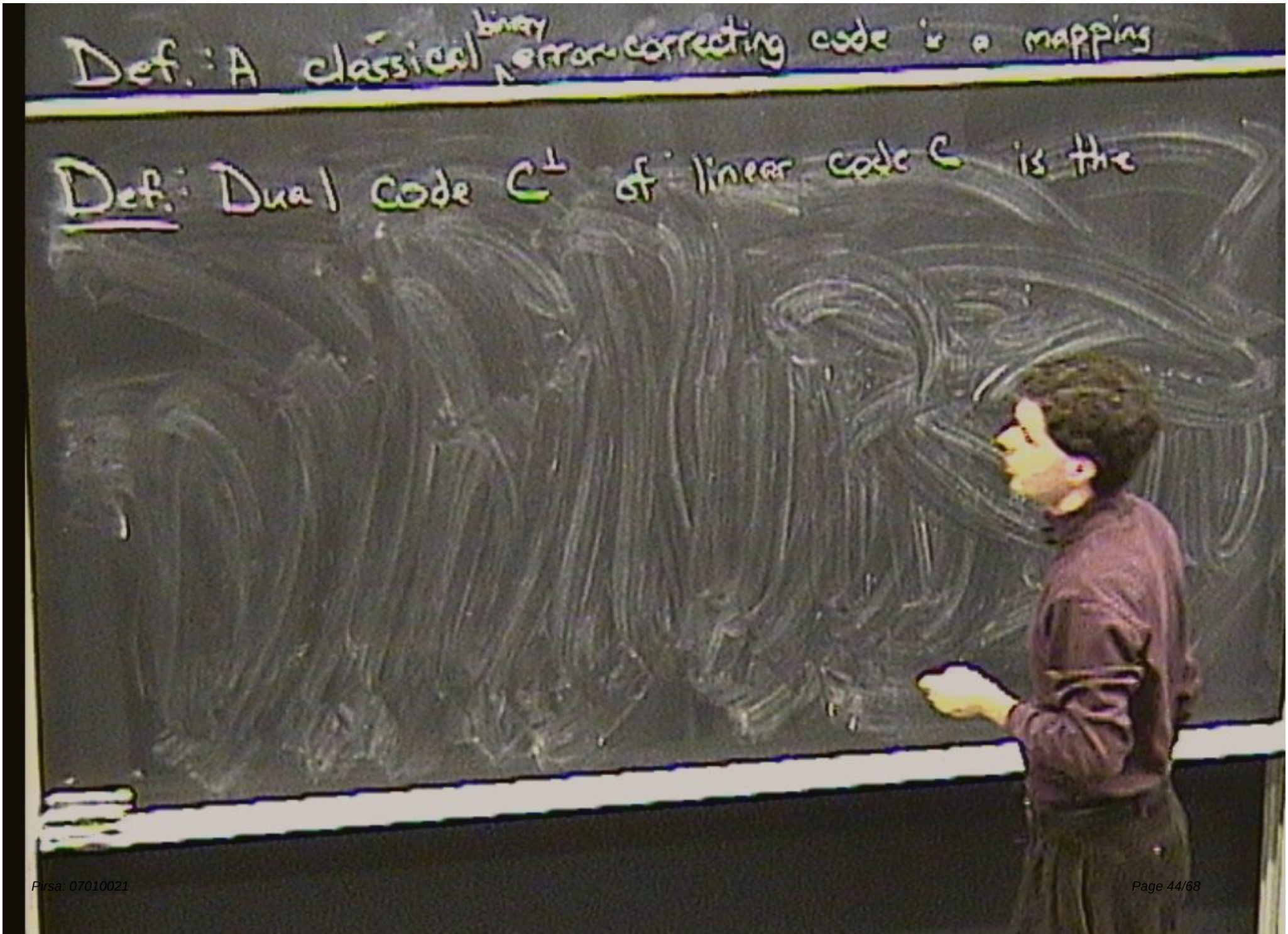
$$H = \left(\begin{array}{c|c|c|c|c|c|c} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right) \quad [7,4,3]$$

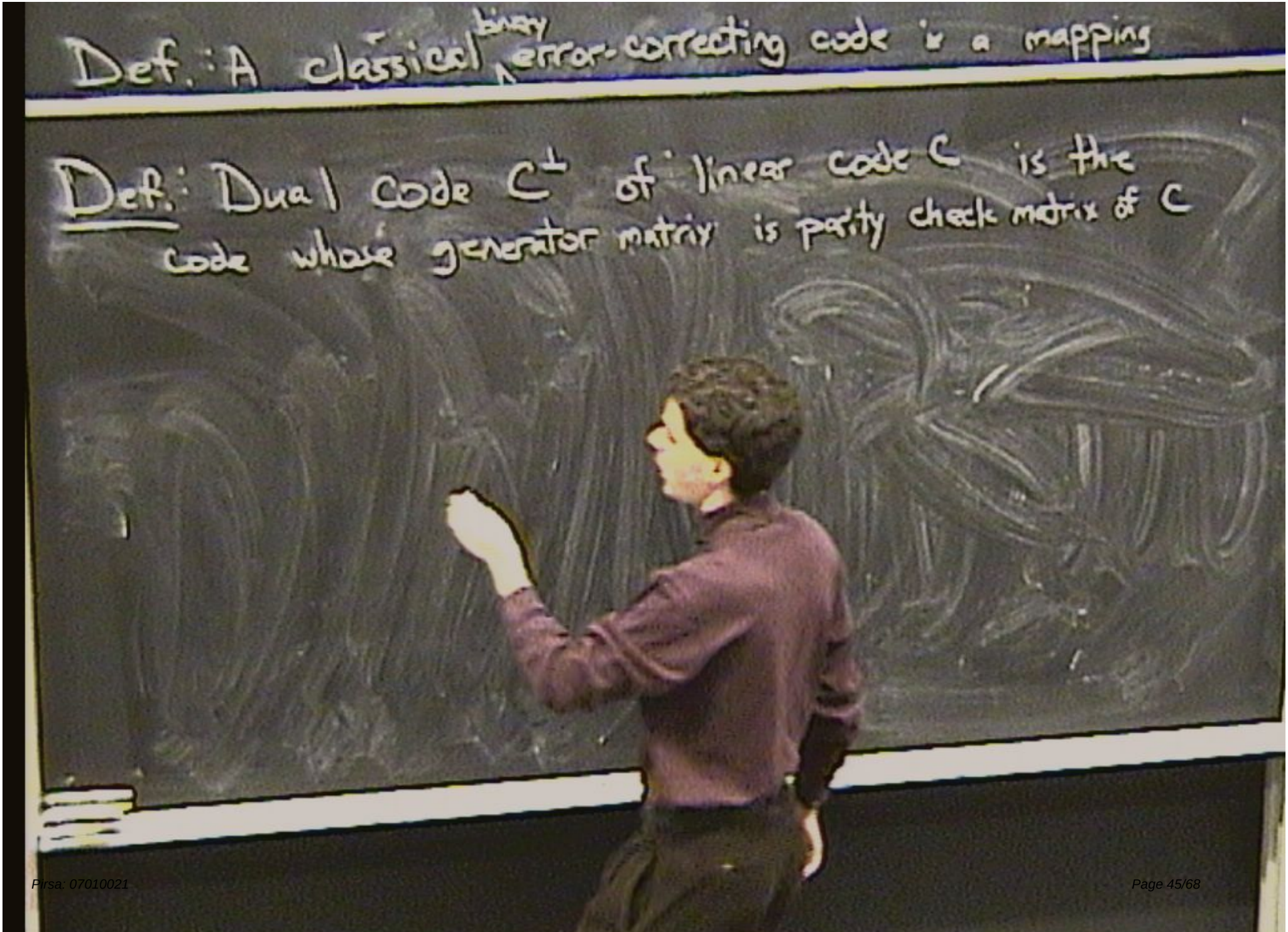
$$G = \left(\begin{array}{c|c|c|c|c|c|c} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right)$$


Def: A classical binary error-correcting code is a mapping
Def: Dual code $C^{\perp}$

Def: A classical binary error-correcting code is a mapping

Def: Dual code $C^{\perp}$ of linear code C is the




Def: A classical binary error-correcting code is a mapping
Def: Dual code $C^\perp$ of linear code C is the code whose generator matrix is parity check matrix of C

Def.: A classical binary error-correcting code is a mapping

Def.: Dual code $C^\perp$ of linear code C is the code whose generator matrix is parity check matrix of C

$(C^\perp)^\perp = C$

Def: A classical binary error-correcting code is a mapping

Def: Dual code $C^\perp$ of linear code C is the code whose generator matrix is parity check matrix of C

$(C^\perp)^\perp = C$

Linear codes as stabilizer codes

Def.: A classical (binary) error-correcting code is a mapping

Def.: Dual code $C^\perp$ of linear code C is the code whose generator matrix is parity check matrix of C

$(C^\perp)^\perp = C$

Linear codes as stabilizer codes:

Parity check matrix → stabilizer
rows of H → generators of S
1's in each row → Z's in that generator

Def: Dual code $C^\perp$ of linear code C is the code whose generator matrix is parity check matrix of C

$(C^\perp)^\perp = C$

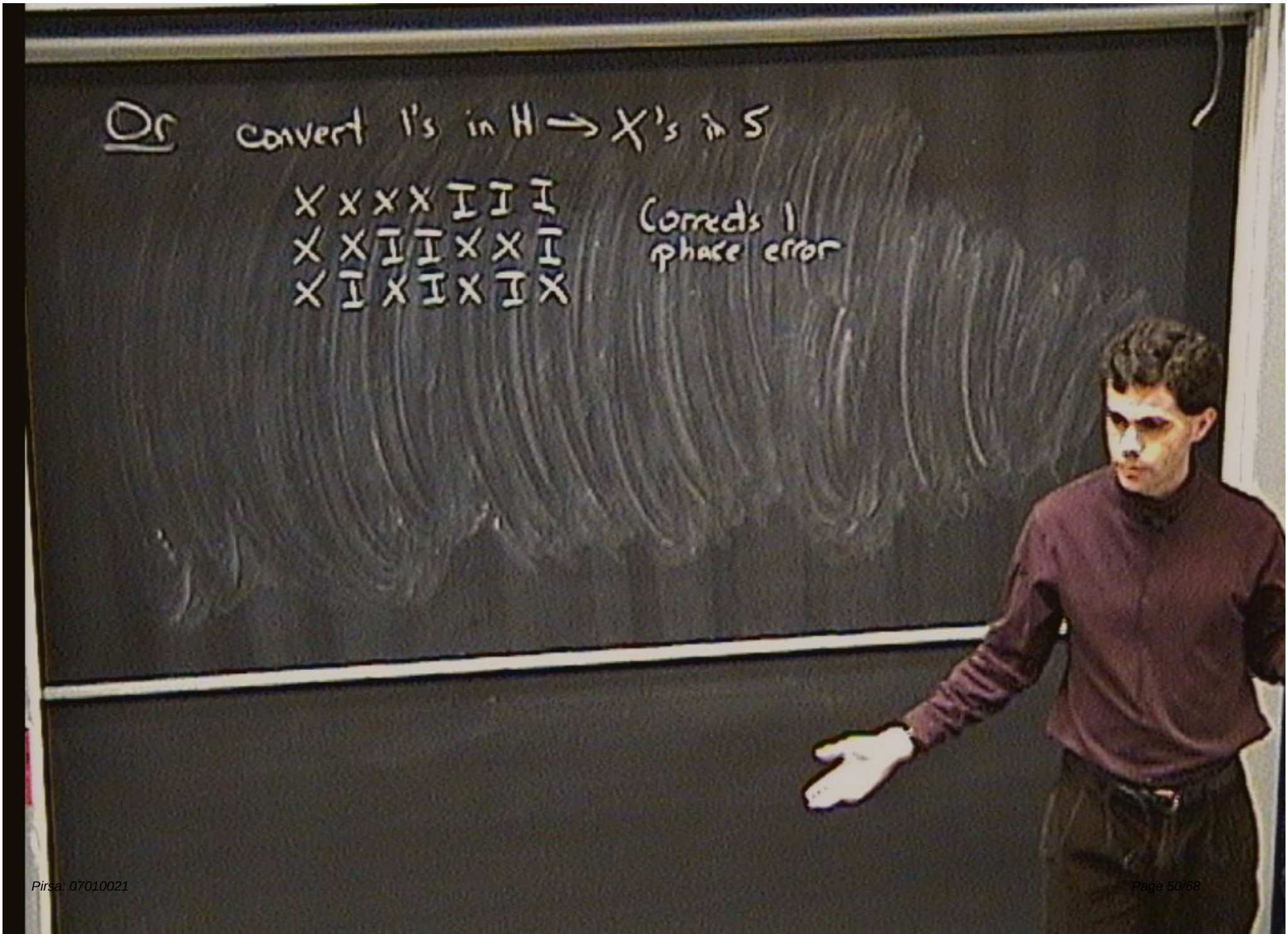
Linear codes as stabilizer codes

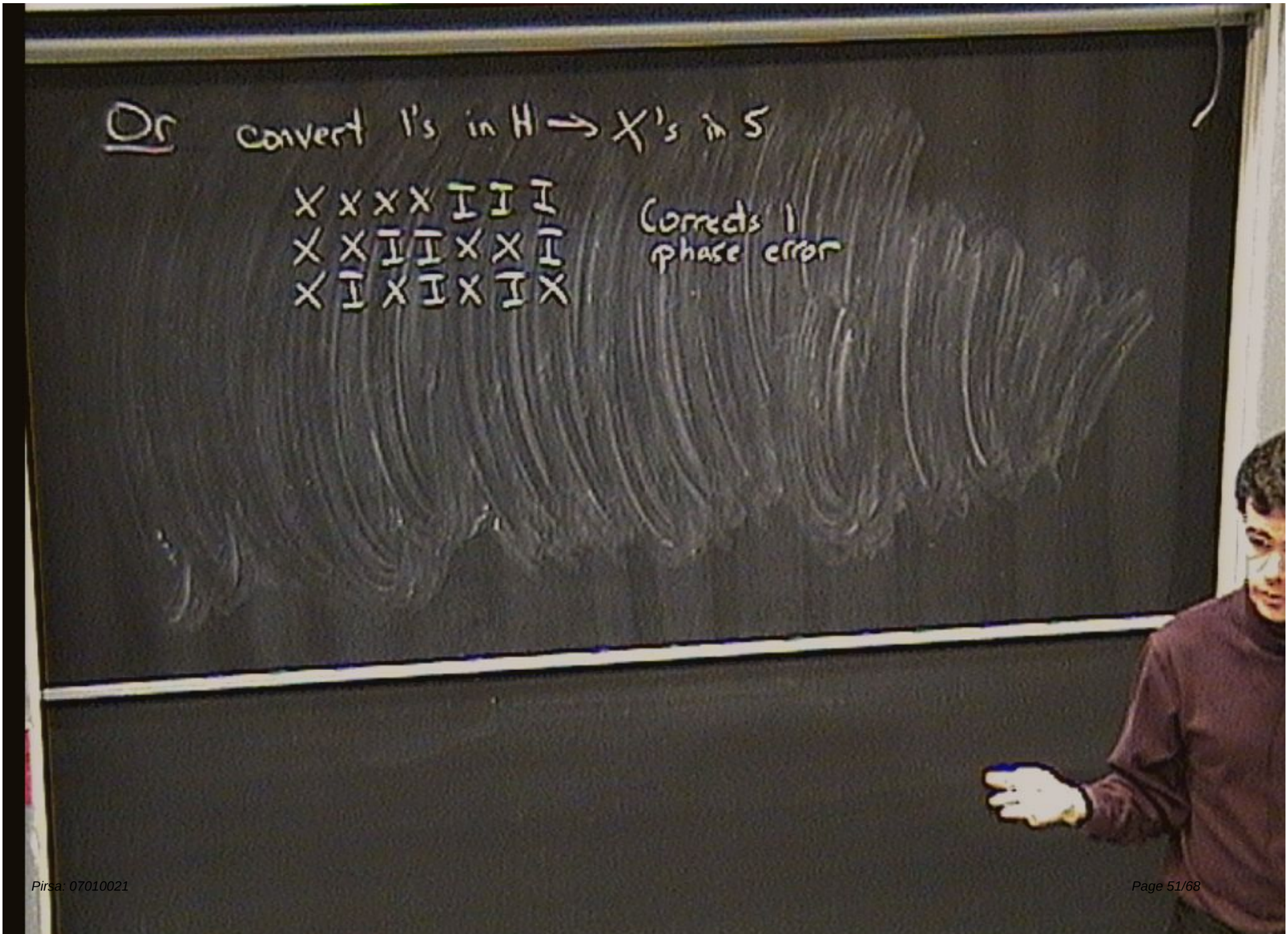
Parity check matrix $\rightarrow$ stabilizer

rows of H $\rightarrow$ generators of S

1's in each row $\rightarrow$ Z's in that generator

$$\begin{matrix} Z & Z & Z & Z & I & I & I \\ Z & Z & I & I & Z & Z & I \\ Z & I & Z & I & Z & I & Z \end{matrix}$$


Or convert I's in H → X's in S
X X X X I I I
X X I I X X I
X I X I X I X
Corrects 1 phase error


Or convert I's in H → X's in 5
X X X X I I I
X X I I X X I
X I X I X I X
Corrects 1 phase error

Or convert 1's in H → X's in S

$$\left.\begin{array}{l} XXXXIII \\ XXIIXXI \\ XIXIXIX \end{array}\right\} C_2 \quad \text{Corrects 1 phase error}$$

$$\left.\begin{array}{l} ZZZZIII \\ ZZIIZZI \\ ZIZIZIZ \end{array}\right\} C_1$$

Use C_1 to correct bit flips
C_2 to correct phase errors

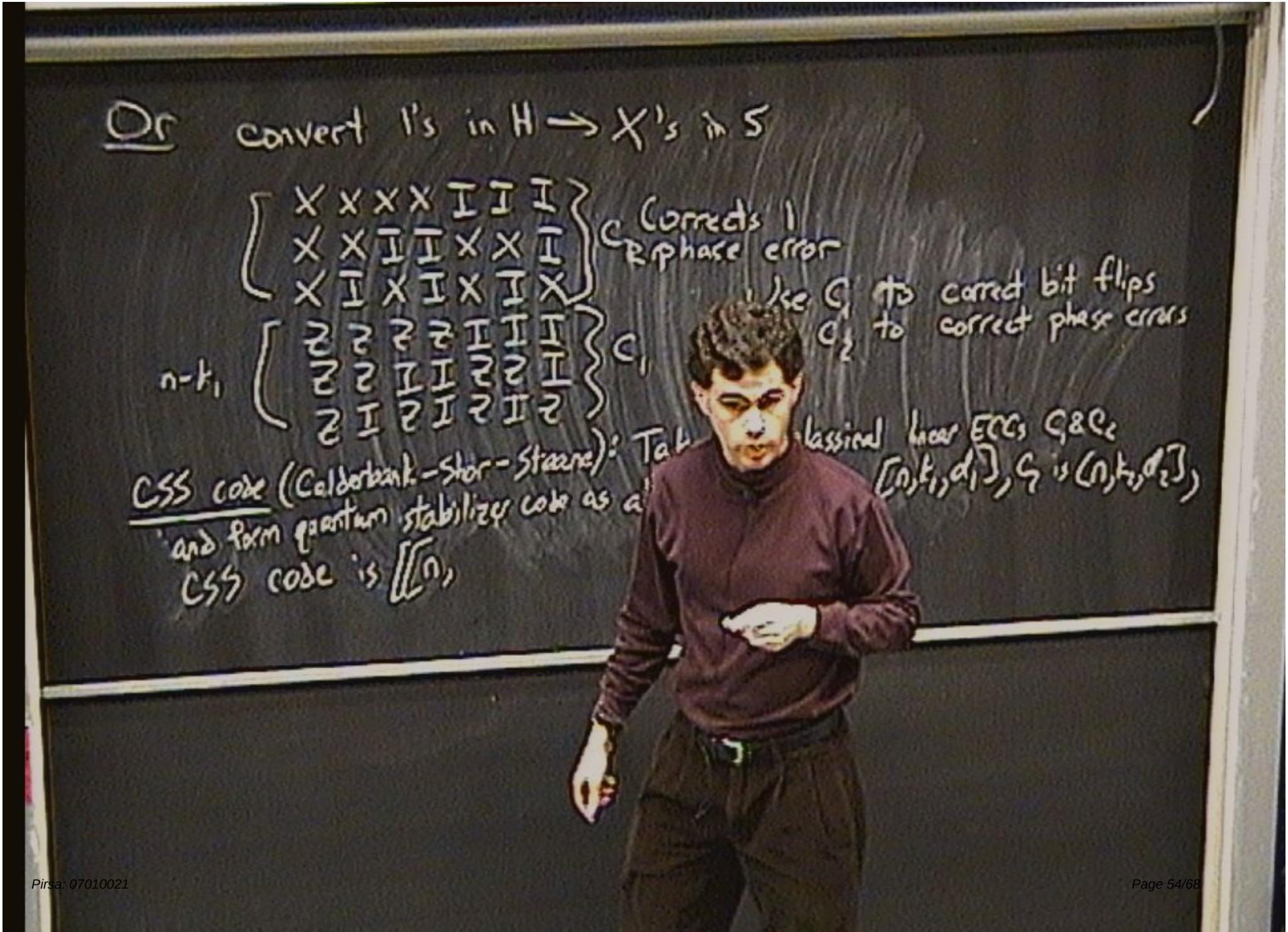
Or convert I's in $H \rightarrow X$'s in S

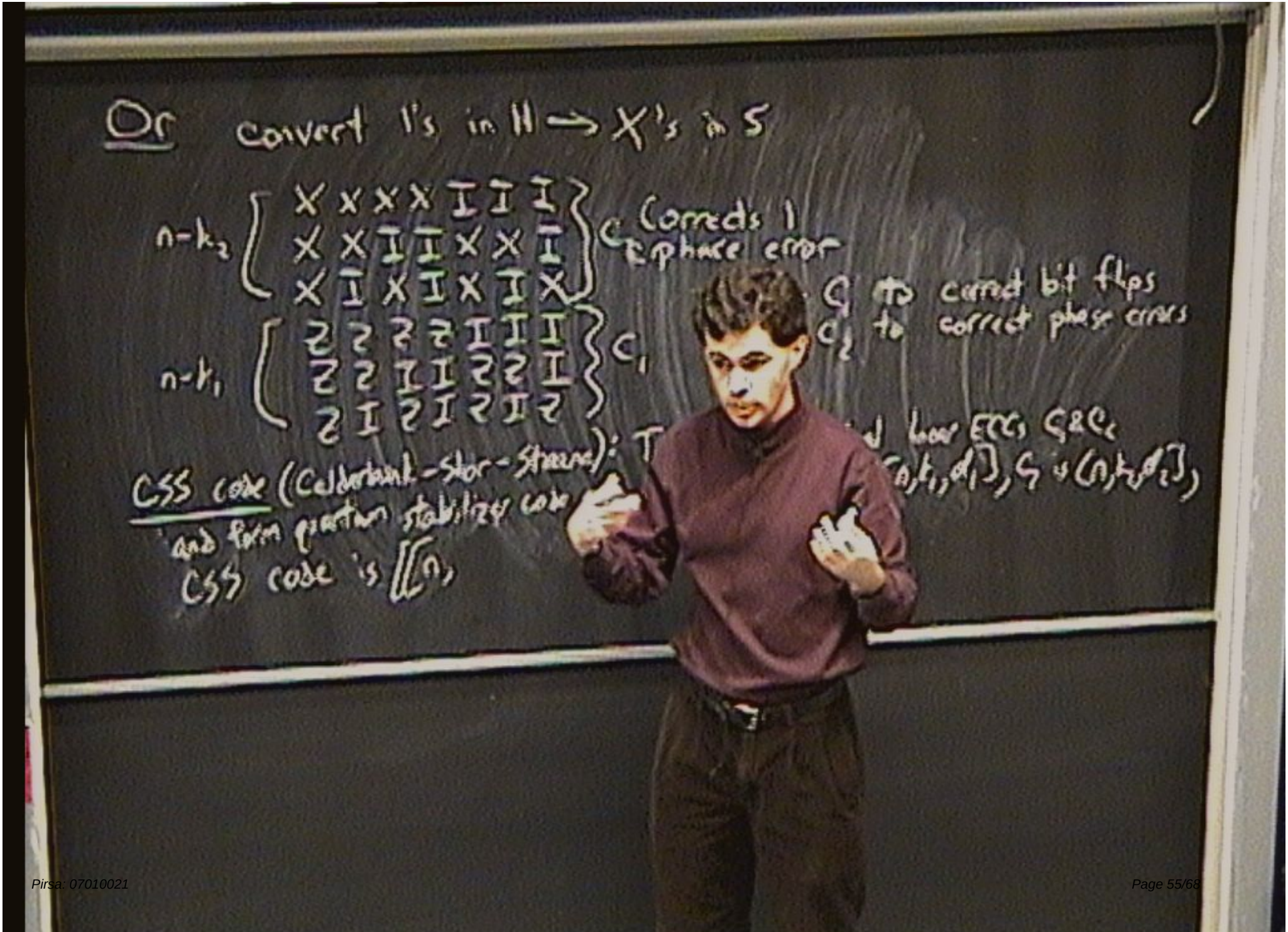
$$\left.\begin{matrix} X\,X\,X\,X\,I\,I\,I \\ X\,X\,I\,I\,X\,X\,I \\ X\,I\,X\,I\,X\,I\,X \end{matrix}\right\} C_2 \quad \text{Corrects 1 phase error}$$

$$\left.\begin{matrix} Z\,Z\,Z\,Z\,I\,I\,I \\ Z\,Z\,I\,I\,Z\,Z\,I \\ Z\,I\,Z\,I\,Z\,I\,Z \end{matrix}\right\} C_1$$

Use C_1 to correct bit flips
C_2 to correct phase

CSS code (Calderbank-Shor-Steane): Take two classical linear ECCs $C_1 \& C_2$ and form quantum stabilizer code as above.


Or convert I's in H → X's in S
X X X X I I I
X X I I X X I
X I X I X I X
Corrects 1 phase error
n-k1
Z Z Z Z I I I
Z Z I I Z Z I
Z I Z I Z I Z
C1
Use C1 to correct bit flips
C2 to correct phase errors
CSS code (Calderbank-Shor-Steane): Tak
lassical linear ECCs C1&C2
[n,k1,d1], C2 is [n,k2,d2])
and form quantum stabilizer code as a
CSS code is [[n,


Or convert I's in H → X's in S
n-k2
n-k1
C2
C1
Corrects 1 phase error
C1 to correct bit flips
C2 to correct phase errors
CSS code (Calderbank-Shor-Steane):
and form quantum stabilizer code
CSS code is [[n,

Or convert 1's in $H \rightarrow X$'s in S

$n-k_2$ $\left\{ \begin{array}{c} X\,X\,X\,X\,I\,I\,I \\ X\,X\,I\,I\,X\,X\,I \\ X\,I\,X\,I\,X\,I\,X \end{array} \right\} C_2$ Corrects 1 phase error

$n-k_1$ $\left\{ \begin{array}{c} Z\,Z\,Z\,Z\,I\,I\,I \\ Z\,Z\,I\,I\,Z\,Z\,I \\ Z\,I\,Z\,I\,Z\,I\,Z \end{array} \right\} C_1$

Use C_1 to c... it flips
C_2 to c... se errors

CSS code (Calderbank-Shor-Steane): Take two classical ... and form quantum stabilizer code as above. If C_1 is [...]
CSS code is $[[n, k_1+k_2-n, d]]$, do

Or convert 1's in $H \rightarrow X$'s in S

$$\left.\begin{matrix} X\,X\,X\,X\,I\,I\,I \\ X\,X\,I\,I\,X\,X\,I \\ X\,I\,X\,I\,X\,I\,X \end{matrix}\right\} C_2 \quad \text{Corrects 1 phase error}$$

$$\left.\begin{matrix} Z\,Z\,Z\,Z\,I\,I\,I \\ Z\,Z\,I\,I\,Z\,Z\,I \\ Z\,I\,Z\,I\,Z\,I\,Z \end{matrix}\right\} C_1$$

Use C_1 to correct bit flips
C_2 to correct phase errors

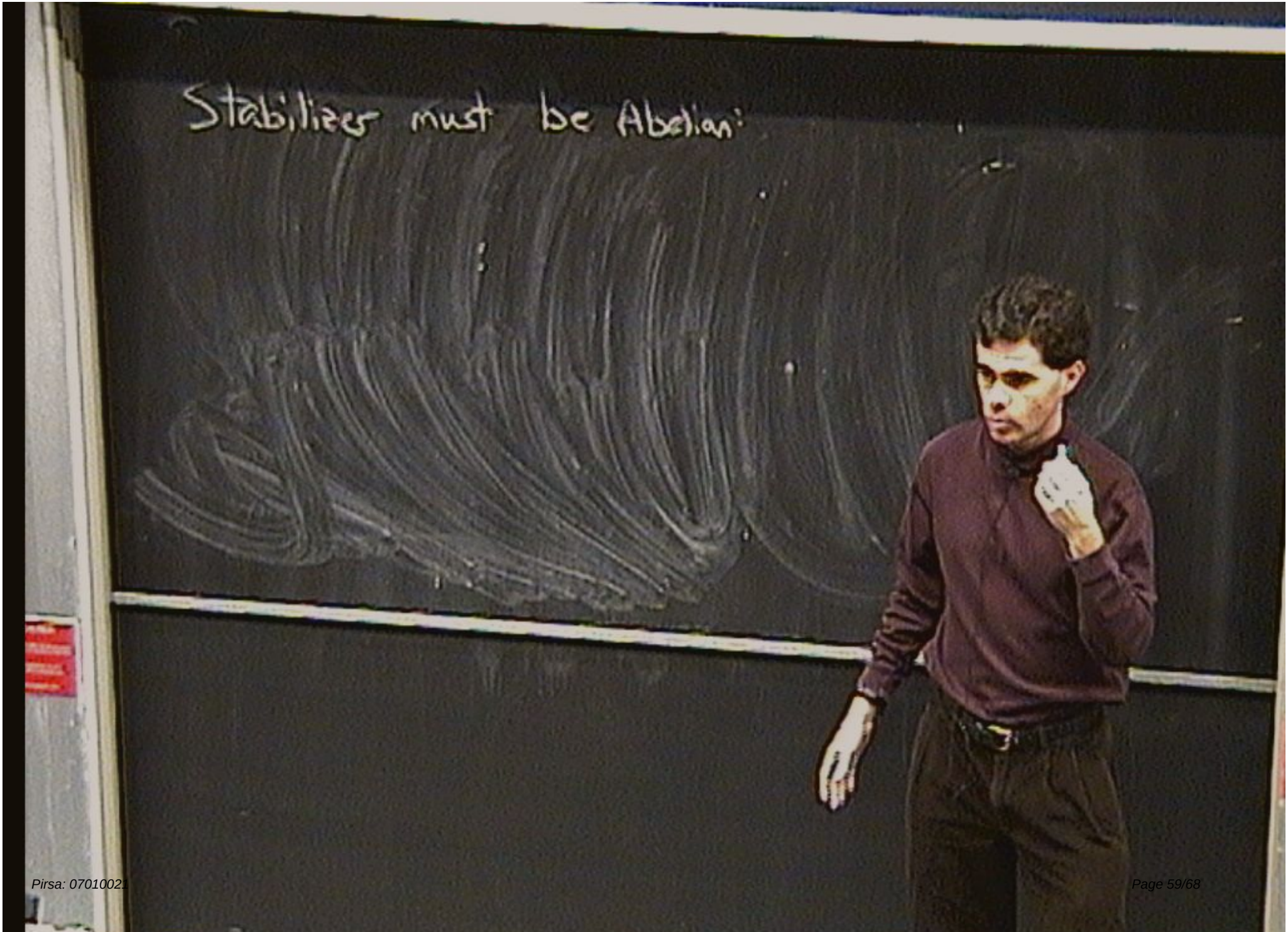
CSS code (Calderbank-Shor-Steane): Take two classical linear ECCs C_1 & C_2 and form quantum stabilizer code as above. If C_1 is $[n, k_1, d_1]$, C_2 is $(n, k_2, d_2]$)
CSS code is $[[n, k_1 + k_2 - n, d]]$, $d \geq \min(d_1, d_2)$

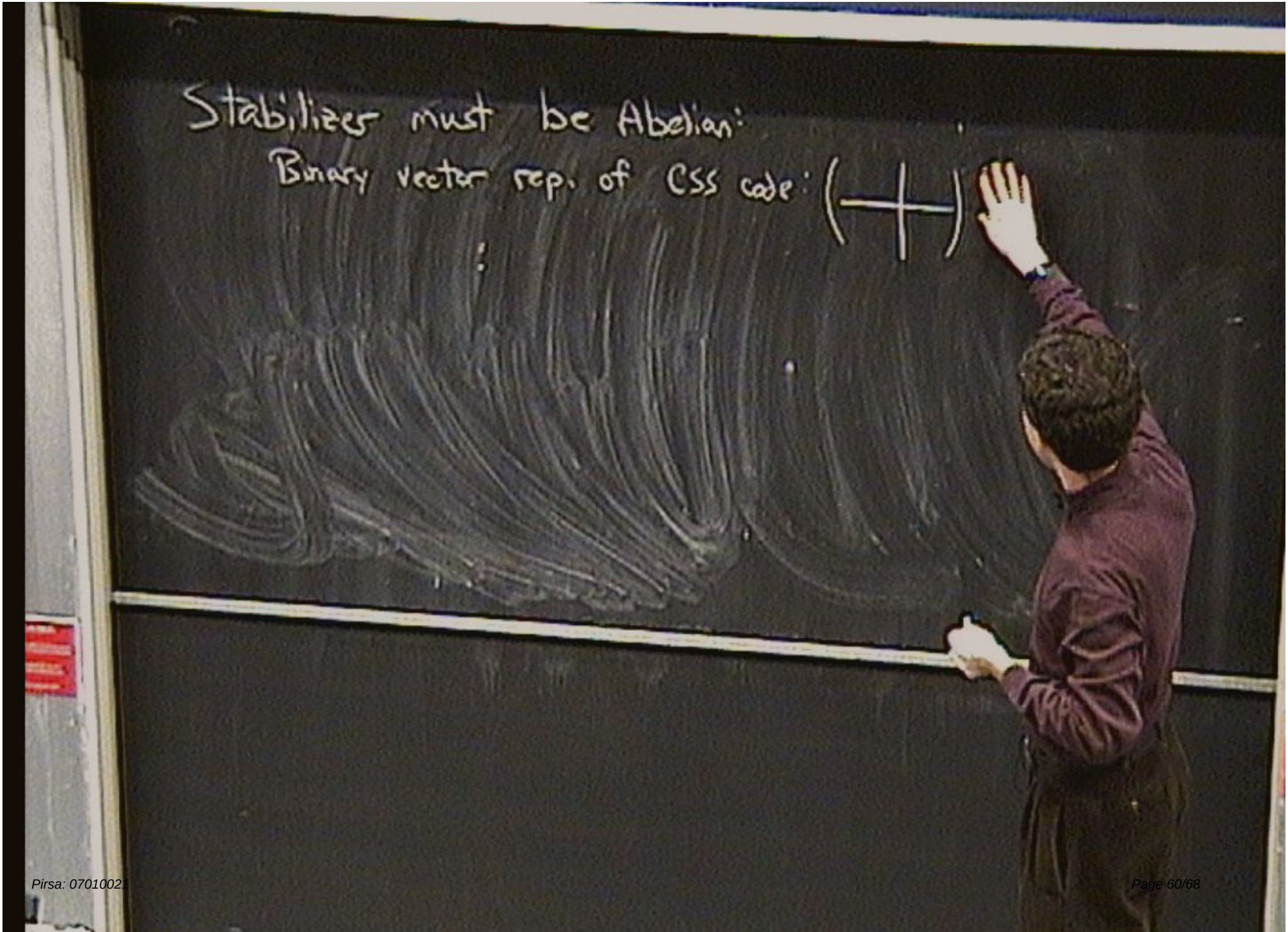
Or convert 1's in $H \to X$'s in S

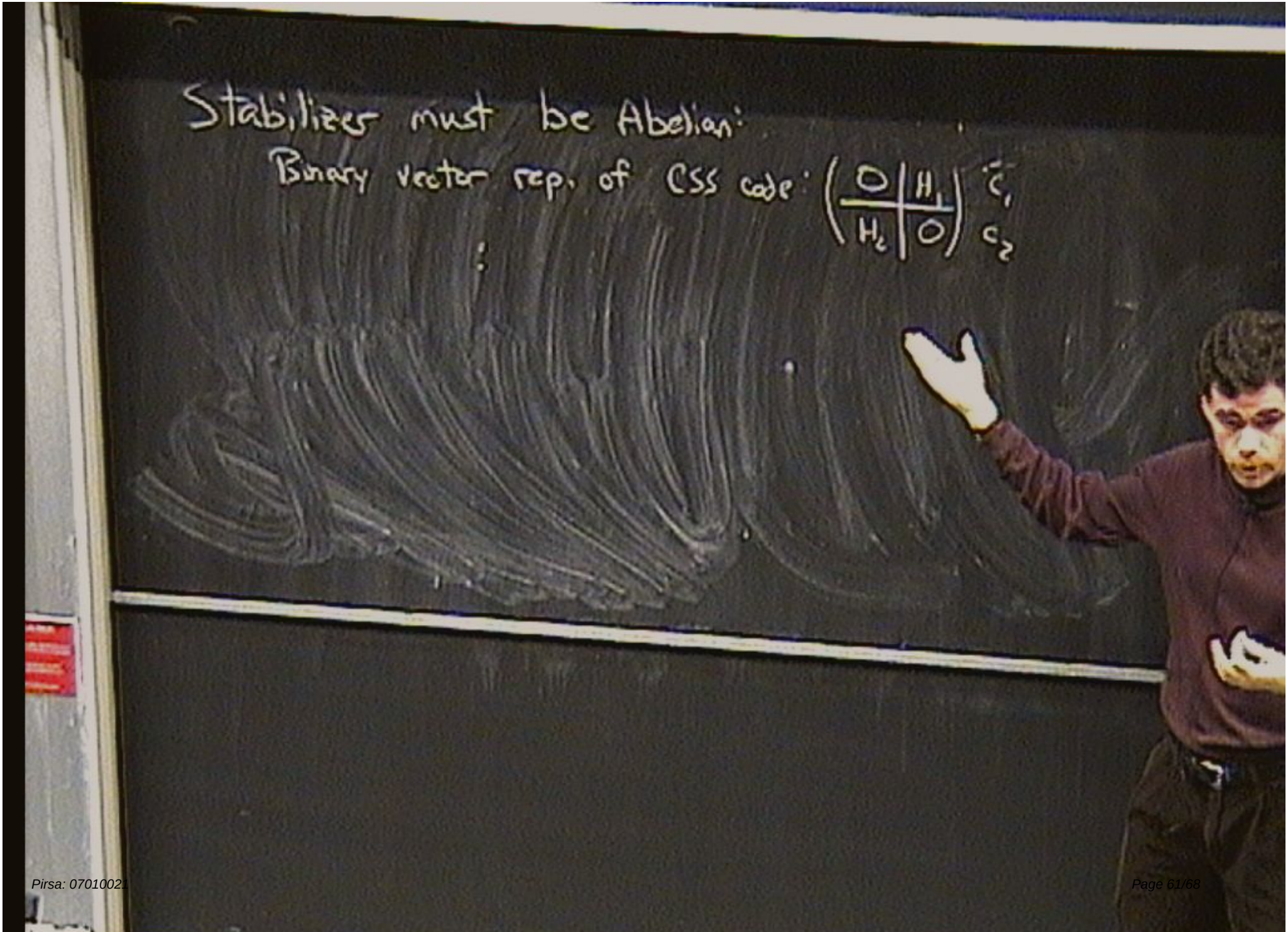
$$\begin{array}{l} n-k_2 \left\{ \begin{array}{c} X\,X\,X\,X\,I\,I\,I \\ X\,X\,I\,I\,X\,X\,I \\ X\,I\,X\,I\,X\,I\,X \end{array} \right\} C_2 \text{ Corrects 1 phase error} \\ n-k_1 \left\{ \begin{array}{c} Z\,Z\,Z\,Z\,I\,I\,I \\ Z\,Z\,I\,I\,Z\,Z\,I \\ Z\,I\,Z\,I\,Z\,I\,Z \end{array} \right\} C_1 \end{array}$$

Use C_1 to correct bit flips
C_2 to correct phase errors

CSS code (Calderbank-Shor-Steane): Take two classical linear ECCs C_1 & C_2 and form quantum stabilizer code as above. If C_1 is $[n, k_1, d_1]$, C_2 is $(n, k_2, d_2]$)
CSS code is $[[n, k_1 + k_2 - n, d]]$, $d \geq \min(d_1, d_2)$


Stabilizer must be Abelian:


Stabilizer must be Abelian:
Binary vector rep. of CSS code:


Stabilizer must be Abelian:
Binary vector rep. of CSS code:
$\left(\begin{array}{c|c} 0 & H_1 \\ \hline H_2 & 0 \end{array}\right)$
c_1
c_2

Stabilizer must be Abelian:

Binary vector rep. of CSS code: $\left(\begin{array}{c|c} 0 & H_1 \\ \hline H_2 & 0 \end{array}\right)$ $\begin{matrix} \vec{c}_1 \\ c_2 \end{matrix}$

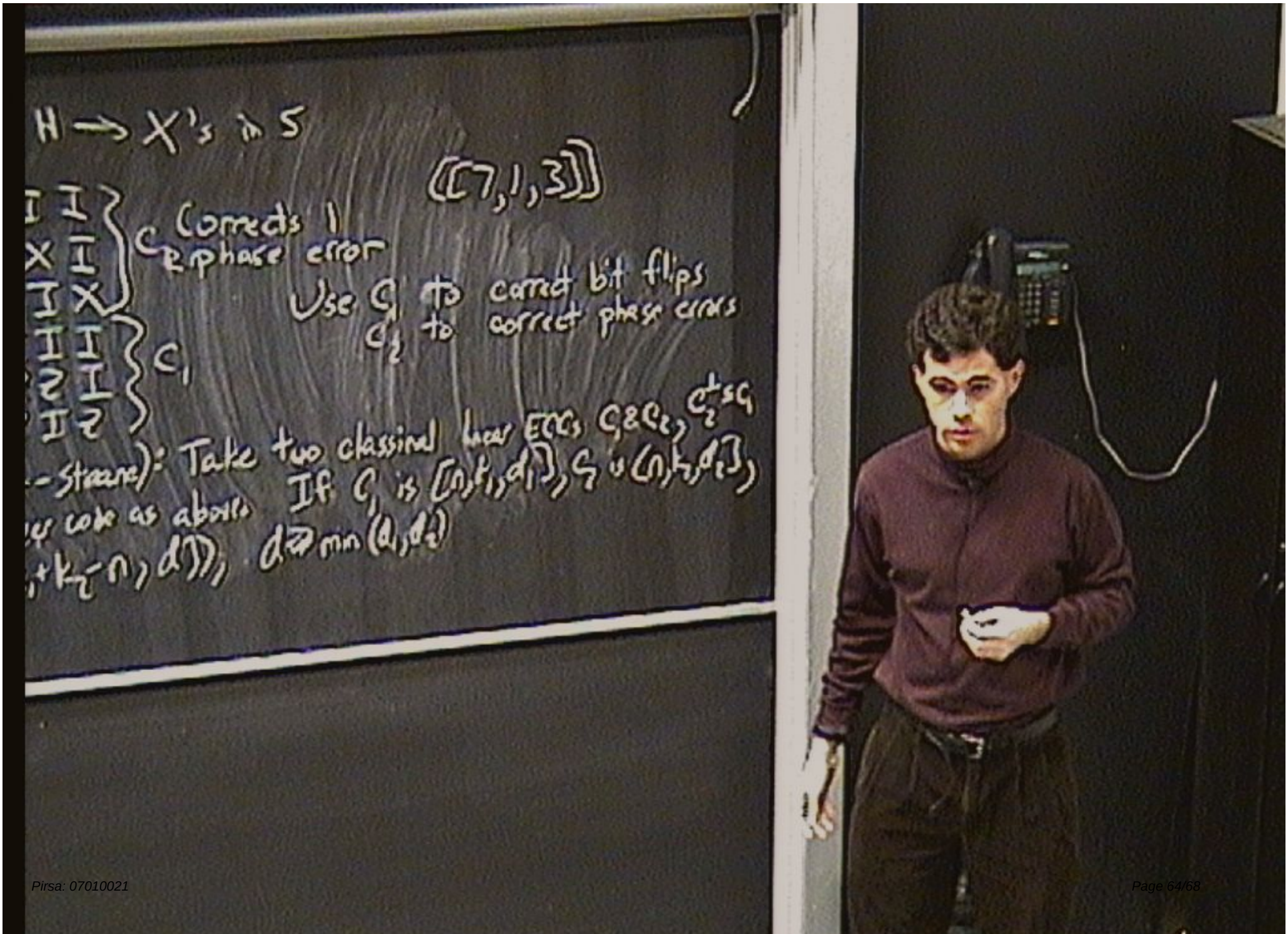
Generators commute iff rows of H_1 are orthogonal rows of H_2

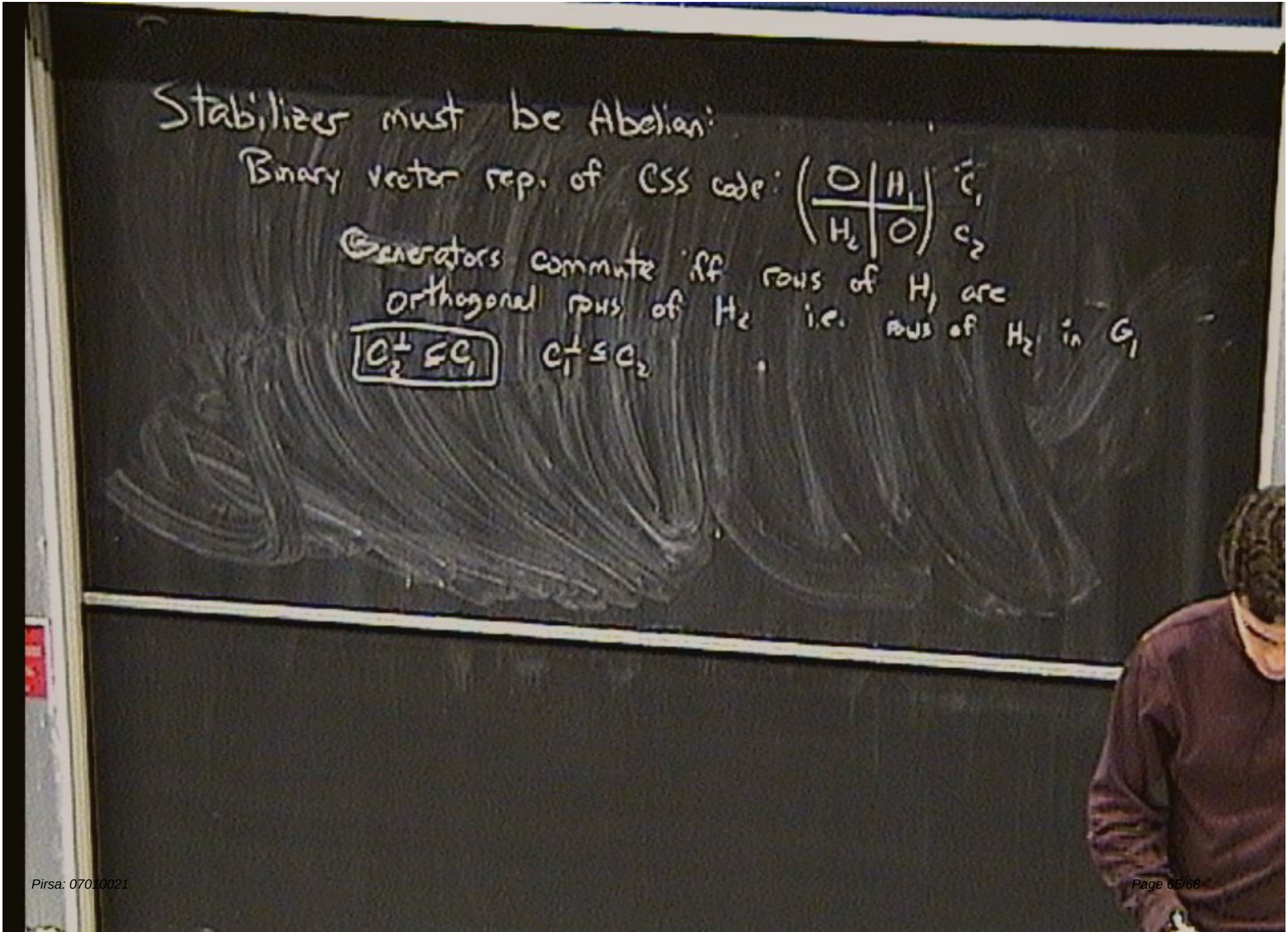
Stabilizers must be Abelian:

Binary vector rep. of CSS code: $\left(\begin{array}{c|c} 0 & H_1 \\ \hline H_2 & 0 \end{array}\right)$ $\begin{matrix} \bar{C}_1 \\ C_2 \end{matrix}$

Generators commute iff rows of H_1 are orthogonal rows of H_2 i.e. rows of H_2 in G_1

$\boxed{C_2^{\perp} \subset C_1}$


H → X's in S
[[7,1,3]]
Corrects 1 phase error
C2
C1
Use C1 to correct bit flips
C2 to correct phase errors
Take two classical linear ECCs C1 ⊇ C2
C2⊥ sq
If C1 is [n, k1, d1], C2 is [n, k2, d2]
d = min(d1, d2)


Stabilizer must be Abelian:
Binary vector rep. of CSS code: $\left(\begin{array}{c|c} 0 & H_1 \\ \hline H_2 & 0 \end{array}\right)$ $\bar{c}_1$ c_2
Generators commute iff rows of H_1 are orthogonal rows of H_2 i.e. rows of H_2 in G_1
$C_2^\perp \subseteq C_1$ $C_1^\perp \subseteq C_2$

Or convert I's in H → X's in S

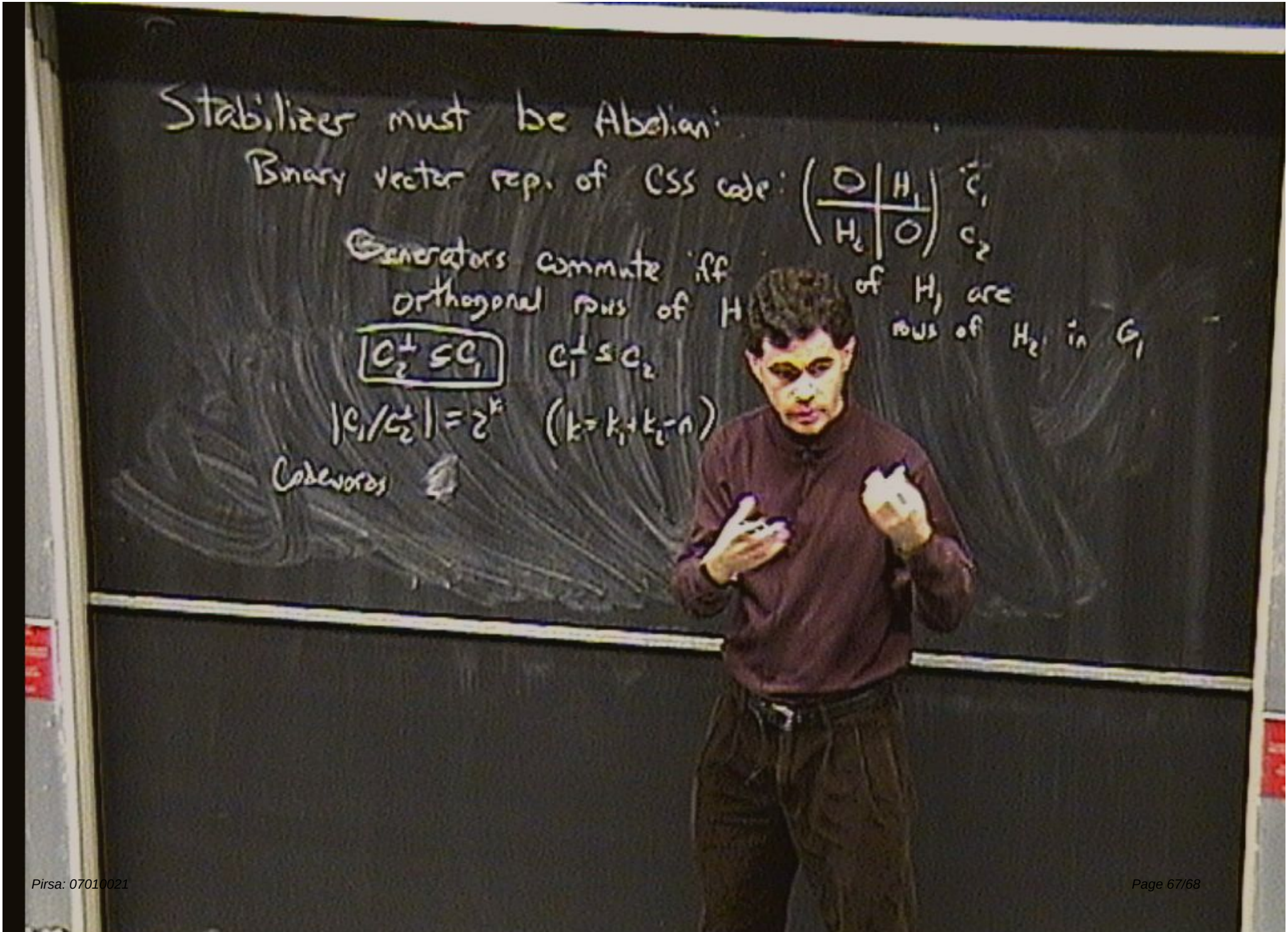
$$\begin{array}{ll} n-k_2 & \left\{\begin{array}{ccccccc} X & X & X & X & I & I & I \\ X & X & I & I & X & X & I \\ X & I & X & I & X & I & X \end{array}\right\} C_2 \\ n-k_1 & \left\{\begin{array}{ccccccc} Z & Z & Z & Z & I & I & I \\ Z & Z & I & I & Z & Z & I \\ Z & I & Z & I & Z & I & Z \end{array}\right\} C_1 \end{array}$$

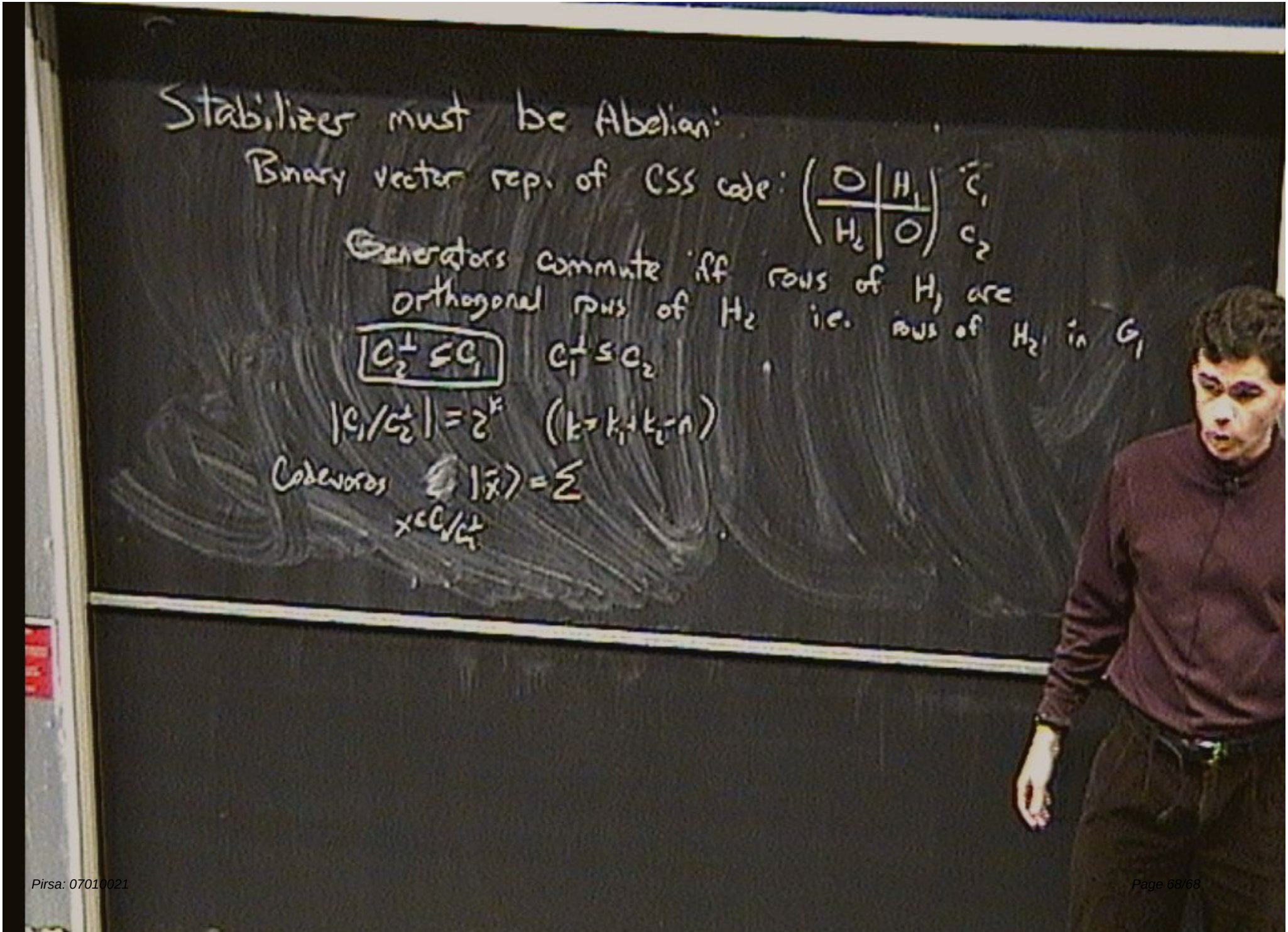
Corrects 1 phase error

([7,1,3])

Use C_1 to correct
C_2 to correct

CSS code (Calderbank-Shor-Steane): Take two classical linear ECCs and form quantum stabilizer code as above. If C_1 is $[n, k_1, d_1]$, C_2 CSS code is $[[n, k_1+k_2-n, d]]$; $d \geq \min(d_1, d_2)$


Stabilizer must be Abelian:
Binary vector rep. of CSS code:
$\left(\begin{array}{c|c} 0 & H_1 \\ \hline H_2 & 0 \end{array}\right)$ $\vec{c}_1$ c_2
Generators commute iff
orthogonal rows of H
of H_1 are
rows of H_2 in G_1
$C_2^\perp \subseteq C_1$
$C_1^\perp \subseteq C_2$
$|C_1/C_2^\perp| = 2^k$ $(k = k_1 + k_2 - n)$
Codewords


Stabilizer must be Abelian:
Binary vector rep. of CSS code: $\left(\begin{array}{c|c} 0 & H_1 \\ \hline H_2 & 0 \end{array}\right)$ $\vec{c}_1$ c_2
Generators commute iff rows of H_1 are orthogonal rows of H_2 i.e. rows of H_2 in G_1
$C_2^\perp \subseteq C_1$ $C_1^\perp \subseteq C_2$
$|C_1/C_2^\perp| = 2^k$ $(k = k_1 + k_2 - n)$
Codewords $\sum_{x \in C_1/C_2^\perp} |\vec{x}\rangle = \sum$