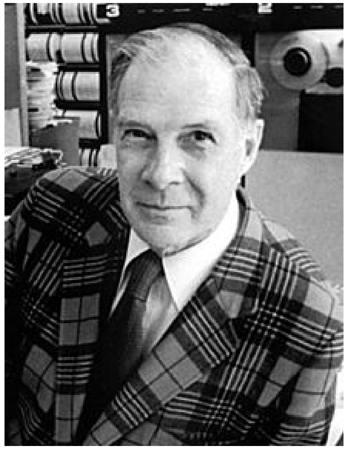
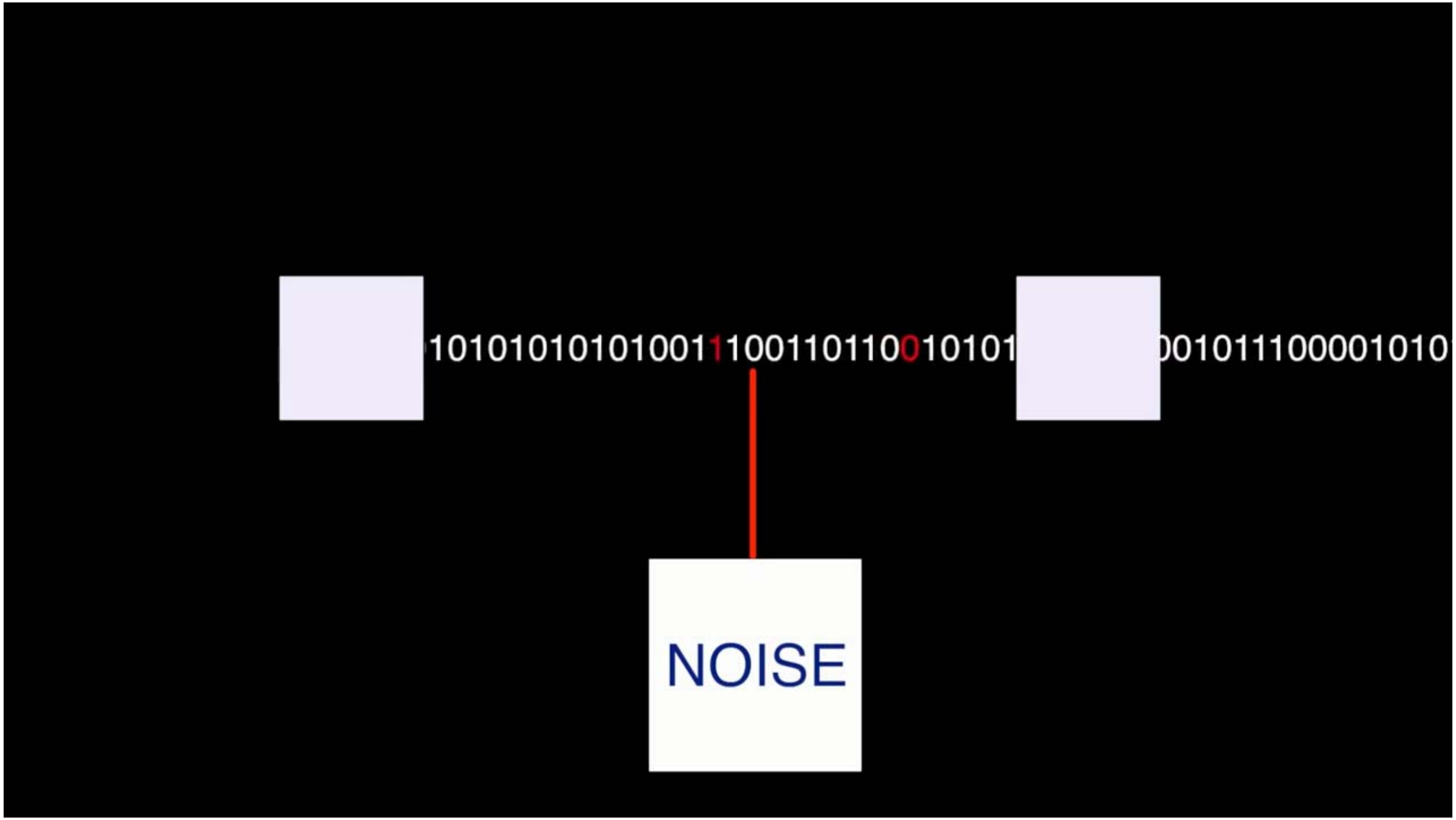
Hamming codes

- One of the earliest codes studied in coding theory.
- Named after Richard W. Hamming
  - The IEEE Richard W. **Hamming Medal**, named after him, is an award given annually by Institute of Electrical and Electronics Engineers (IEEE), for "exceptional contributions to information sciences, systems and technology“.
    - Sponsored by Qualcomm, Inc
    - Some Recipients:
      - 1988 - Richard W. Hamming
      - 1997 - Thomas M. Cover
      - 1999 - David A. Huffman
      - 2011 - Toby Berger



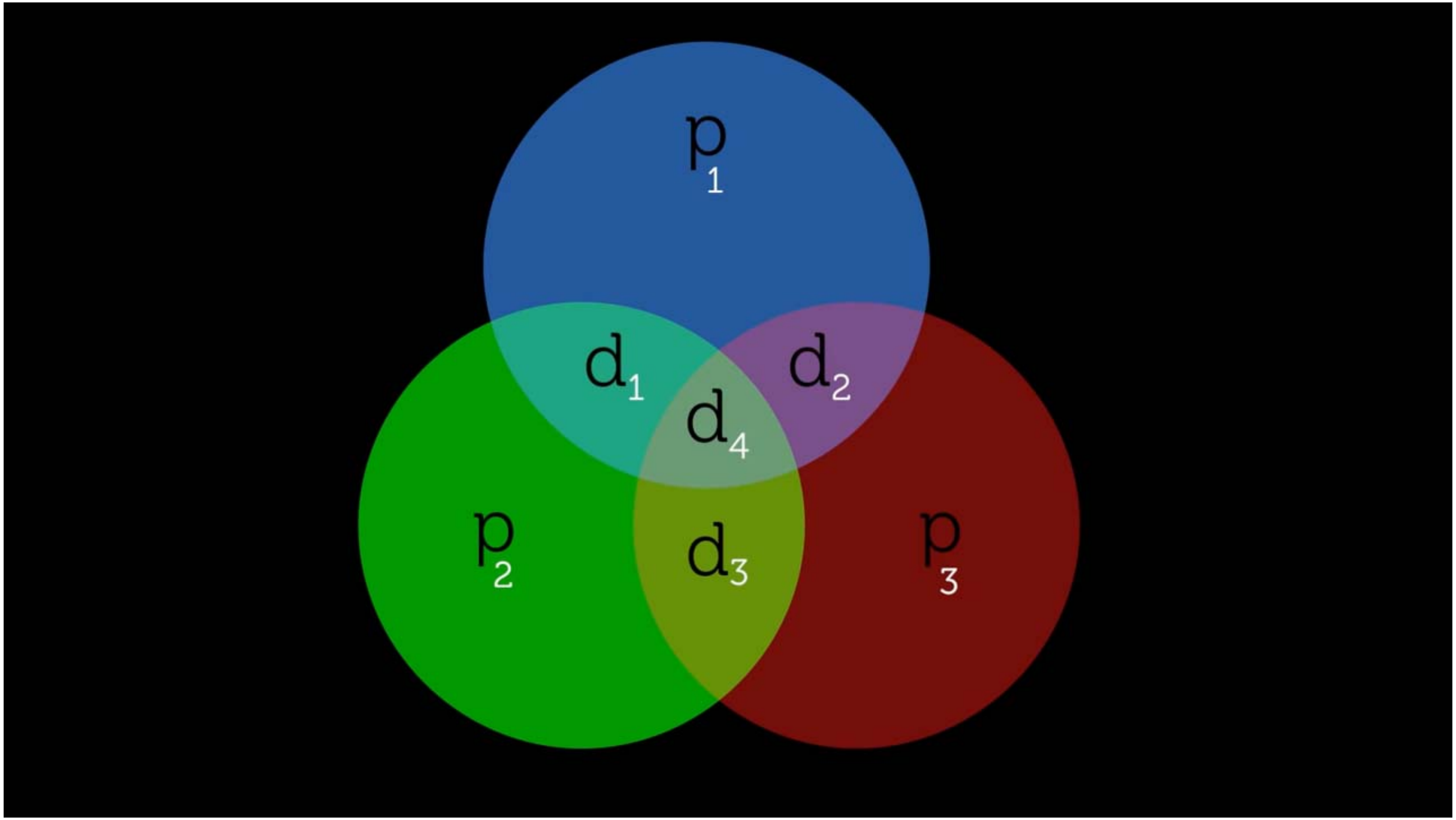
- The simplest of a class of (algebraic) error correcting codes that **can *correct* one error in a block of bits**

Hamming codes



[https://www.youtube.com/watch?v=cBBTWcHkVVY]

Hamming codes: Ex. 1



[https://www.youtube.com/watch?v=cBBTWcHkVVY]

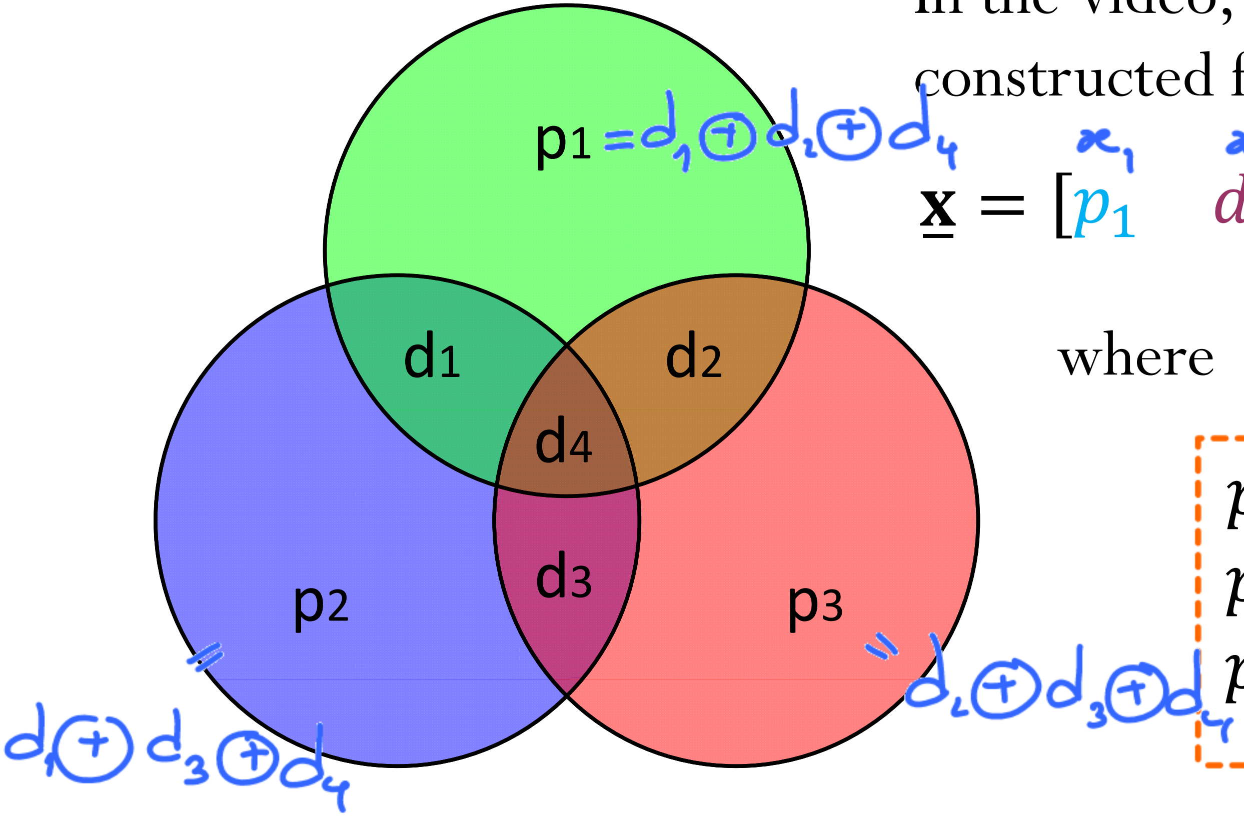
Hamming codes: Ex. 1

$n=7$

$k=4$

code rate $= \frac{k}{n} = \frac{4}{7}$

This is an example of Hamming (7,4) code



In the video, the codeword is constructed from the data by

$$\underline{\mathbf{x}} = [\overset{x_1}{p_1} \quad \overset{x_2}{d_1} \quad \overset{x_3}{p_2} \quad \overset{x_4}{d_2} \quad \overset{x_5}{p_3} \quad \overset{x_6}{d_3} \quad \overset{x_7}{d_4}]$$

where

structure

$$\boxed{\begin{aligned} p_1 &= d_1 \oplus d_2 \oplus d_4 \\ p_2 &= d_1 \oplus d_3 \oplus d_4 \\ p_3 &= d_2 \oplus d_3 \oplus d_4 \end{aligned}}$$

- The message bits are also referred to as the data bits or information bits.
- The non-message bits are also referred to as parity check bits, checksum bits, parity bits, or check bits.

Writing the generator matrix from the code “structure”

Generator matrix: a revisit

- Fact: The 1s and 0s in the j^{th} column of **G** tells which positions of the data bits are combined ($\oplus$) to produce the j^{th} bit in the codeword.
- For the Hamming code in the previous slide,

$$\underline{\mathbf{x}} = \begin{bmatrix} p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \end{bmatrix}$$

$$p_1 = d_1 \oplus d_2 \oplus d_4$$
$$p_2 = d_1 \oplus d_3 \oplus d_4$$
$$p_3 = d_2 \oplus d_3 \oplus d_4$$

$$= \begin{bmatrix} d_1 & d_2 & d_3 & d_4 \end{bmatrix} \underbrace{\left[\begin{array}{ccccccc} p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right]}_{\mathbf{G}}$$

Generator matrix: a revisit

- From $\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = \sum_{j=1}^{k} b_j \underline{\mathbf{g}}^{(j)}$, we see that the j element of the codeword $\underline{\mathbf{x}}$ of a linear code is constructed from a linear combination of the bits in the message:

$$x_j = \sum_{i=1}^{k} b_i g_{ij} .$$

- The elements in the j^{th} column of the generator matrix become the weights for the combination.
  - Because we are working in GF(2), g_{ij} has only two values: 0 or 1.
    - When it is 1, we use b_i in the sum.
    - When it is 0, we don't use b_i in the sum.
- Conclusion: For the j^{th} column, the i^{th} element is determined from whether the i^{th} message bit is used in the sum that produces the j^{th} element of the codeword $\underline{\mathbf{x}}$.

Codebook of a linear block code

d				**x**						
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	1	0	1
0	0	1	0	0	0	1	0	1	1	0
0	0	1	1	1	0	0	0	0	1	1
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	0	1	1	0	0	1
0	1	1	0	1	0	1	1	0	1	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	1	1	0	0	0	0
1	0	0	1	0	1	0	0	1	0	1
1	0	1	0	1	1	0	0	1	1	0
1	0	1	1	0	1	1	0	0	1	1
1	1	0	0	0	1	1	1	1	0	0
1	1	0	1	1	1	0	1	0	0	1
1	1	1	0	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1	1	1	1

- Now that we have a sufficiently-large example of a codebook, let's consider some important types of problems.
- Given a codebook, how can we check that the code is linear?
- Given a codebook, how can we find the corresponding generator matrix?

Codebook of a linear block code

d				**x**						
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	1	0	1
0	0	1	0	0	0	1	0	1	1	0
0	0	1	1	1	0	0	0	0	1	1
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	0	1	1	0	0	1
0	1	1	0	1	0	1	1	0	1	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	1	1	0	0	0	0
1	0	0	1	0	1	0	0	1	0	1
1	0	1	0	1	1	0	0	1	1	0
1	0	1	1	0	1	1	0	0	1	1
1	1	0	0	0	1	1	1	1	0	0
1	1	0	1	1	1	0	1	0	0	1
1	1	1	0	0	1	0	1	0	1	0
1	1	1	1	1	1	1	1	1	1	1

Note that

- Each bit of the codeword for linear code is either
  - the same as one of the message bits
    - Here, the second bit (x_2) of the codeword is the same as the first bit (b_1) of the message
  - the sum of some bits from the message
    - Here, the first bit (x_1) of the codeword is the sum of the first, second and fourth bits of the message.
- So, each column in the codebook should also satisfy the above structure (relationship).

“Reading” the structure from the codebook.

	$\underline{\mathbf{d}}$				$\underline{\mathbf{x}}$						
	0	0	0	0	0	0	0	0	0	0	0
d_4	0	0	0	1	1	0	1	0	1	0	1
d_3	0	0	1	0	0	0	1	0	1	1	0
	0	0	1	1	1	0	0	0	0	1	1
d_2	0	1	0	0	1	0	0	1	1	0	0
	0	1	0	1	0	0	1	1	0	0	1
	0	1	1	0	1	0	1	1	0	1	0
	0	1	1	1	0	0	0	1	1	1	1
d_1	1	0	0	0	1	1	1	0	0	0	0
	1	0	0	1	0	1	0	0	1	0	1
	1	0	1	0	1	1	0	0	1	1	0
	1	0	1	1	0	1	1	0	0	1	1
	1	1	0	0	0	1	1	1	1	0	0
	1	1	0	1	1	1	0	1	0	0	1
	1	1	1	0	0	1	0	1	0	1	0
	1	1	1	1	1	1	1	1	1	1	1

- One can “read” the structure (relationship) from the codebook.
- From $x_j = \sum_{i=1}^{k} d_i g_{ij}$, when we look at the message block with a single 1 at position i, then
  - the value of x_j in the corresponding codeword gives g_{ij}
  - $x_1 = d_1 \oplus d_2 \oplus d_4$
  - $x_3 = d_1 \oplus d_3 \oplus d_4$

“Reading” the generator matrix from the codebook.

	$\underline{\mathbf{d}}$				$\underline{\mathbf{x}}$							
	0	0	0	0	0	0	0	0	0	0	0	
d_4	0	0	0	1	1	0	1	0	1	0	1	$\underline{\mathbf{g}}^{(4)}$
d_3	0	0	1	0	0	0	1	0	1	1	0	$\underline{\mathbf{g}}^{(3)}$
	0	0	1	1	1	0	0	0	0	1	1	
d_2	0	1	0	0	1	0	0	1	1	0	0	$\underline{\mathbf{g}}^{(2)}$
	0	1	0	1	0	0	1	1	0	0	1	
	0	1	1	0	1	0	1	1	0	1	0	
	0	1	1	1	0	0	0	1	1	1	1	
d_1	1	0	0	0	1	1	1	0	0	0	0	$\underline{\mathbf{g}}^{(1)}$
	1	0	0	1	0	1	0	0	1	0	1	
	1	0	1	0	1	1	0	0	1	1	0	
	1	0	1	1	0	1	1	0	0	1	1	
	1	1	0	0	0	1	1	1	1	0	0	
	1	1	0	1	1	1	0	1	0	0	1	
	1	1	1	0	0	1	0	1	0	1	0	
	1	1	1	1	1	1	1	1	1	1	1	

$\uparrow$ $d_1 \oplus d_2 \oplus d_4$

- One can also “read” **G** from the codebook.
- From $\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = \sum_{j=1}^{k} b_j \underline{\mathbf{g}}^{(j)}$, when we look at the message block with a single 1 at position i, then the corresponding codeword is the same as $\underline{\mathbf{g}}^{(j)}$.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} = \mathbf{G}$$

Checking linearity of a code

	$\underline{\mathbf{d}}$				$\underline{\mathbf{x}}$						
	0	0	0	0	0	0	0	0	0	0	0
d_4	0	0	0	1	1	0	1	0	1	0	1
d_3	0	0	1	0	0	0	1	0	1	1	0
	0	0	1	1	1	0	0	0	0	1	1
d_2	0	1	0	0	1	0	0	1	1	0	0
	0	1	0	1	0	0	1	1	0	0	1
	0	1	1	0	1	0	1	1	0	1	0
	0	1	1	1	0	0	0	1	1	1	1
d_1	1	0	0	0	1	1	1	0	0	0	0
	1	0	0	1	0	1	0	0	1	0	1
	1	0	1	0	1	1	0	0	1	1	0
	1	0	1	1	0	1	1	0	0	1	1
	1	1	0	0	0	1	1	1	1	0	0
	1	1	0	1	1	1	0	1	0	0	1
	1	1	1	0	0	1	0	1	0	1	0
	1	1	1	1	1	1	1	1	1	1	1

- Another technique for checking linearity of a code when the codebook is provided is to look at each column of the codeword part.
- Write down the equation by reading the structure from appropriate row discussed earlier.
    - For example, here, we read $x_1 = d_1 \oplus d_2 \oplus d_4$.
- Then, we add the corresponding columns of the message part and check whether the sum is the same as the corresponding codeword column.
- So, we need to check n summations.
    - Direct checking that we discussed earlier consider $\binom{M-1}{2}$ summations.

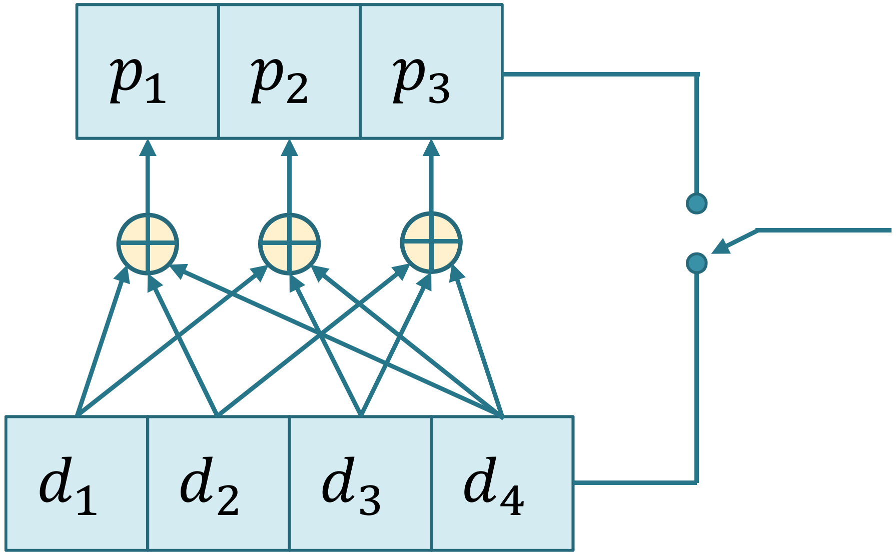
Checking linearity of a code

	$\underline{\mathbf{d}}$				$\underline{\mathbf{x}}$							
	0	0	0	0	0	0	0	0	0	0	0	
d_4	0	0	0	1	1	0	1	0	1	0	1	$\underline{\mathbf{g}}^{(4)}$
d_3	0	0	1	0	0	0	1	0	1	1	0	$\underline{\mathbf{g}}^{(3)}$
	0	0	1	1	1	0	0	0	0	1	1	
d_2	0	1	0	0	1	0	0	1	1	0	0	$\underline{\mathbf{g}}^{(2)}$
	0	1	0	1	0	0	1	1	0	0	1	
	0	1	1	0	1	0	1	1	0	1	0	
	0	1	1	1	0	0	0	1	1	1	1	
d_1	1	0	0	0	1	1	1	0	0	0	0	$\underline{\mathbf{g}}^{(1)}$
	1	0	0	1	0	1	0	0	1	0	1	
	1	0	1	0	1	1	0	0	1	1	0	
	1	0	1	1	0	1	1	0	0	1	1	
	1	1	0	0	1	1	1	1	1	0	0	
	1	1	0	1	1	1	0	1	0	0	1	
	1	1	1	0	0	1	0	1	0	1	0	
	1	1	1	1	1	1	1	1	1	1	1	

- Here is an example of non-linear code.
- Again, we read $x_1 = d_1 \oplus d_2 \oplus d_4$.
- We add the message columns corresponding to d_1, d_2, d_4,
  - We see that the first bit of the 13th codeword does not conform with the structure above.
  - The corresponding message is 1100.
  - We see that $\underline{\mathbf{g}}^{(1)}$ and $\underline{\mathbf{g}}^{(2)}$ are codewords but $\underline{\mathbf{g}}^{(1)} \oplus \underline{\mathbf{g}}^{(2)} =$ 0111100 is not one of the codewords.

Implementation

- Linear block codes are typically implemented with modulo-2 adders tied to the appropriate stages of a shift register.



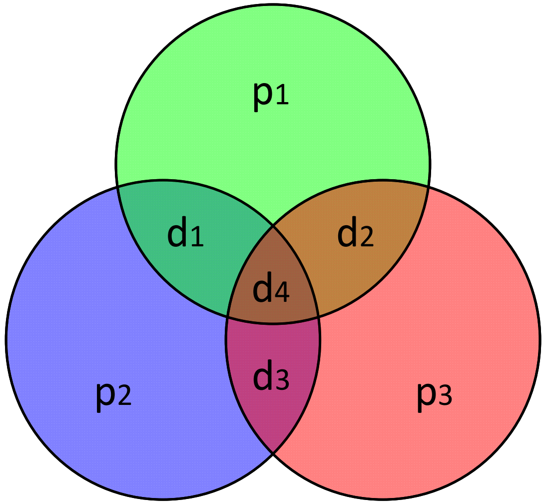
$$\underline{\mathbf{x}} = [p_1 \quad d_1 \quad p_2 \quad d_2 \quad p_3 \quad d_3 \quad d_4]$$

where

$$p_1 = d_1 \oplus d_2 \oplus d_4$$
$$p_2 = d_1 \oplus d_3 \oplus d_4$$
$$p_3 = d_2 \oplus d_3 \oplus d_4$$

Back to

Hamming codes: Ex. 1



$$\begin{array}{cccccccc} & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ \underline{\mathbf{x}} = [& p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4] \end{array}$$

Structure in the codeword:

$$\begin{aligned} p_1 &= d_1 \oplus d_2 \oplus d_4 \\ p_2 &= d_1 \oplus d_3 \oplus d_4 \\ p_3 &= d_2 \oplus d_3 \oplus d_4 \end{aligned} \Longleftrightarrow \begin{aligned} p_1 \oplus d_1 \oplus d_2 \oplus d_4 &= 0 \\ p_2 \oplus d_1 \oplus d_3 \oplus d_4 &= 0 \\ p_3 \oplus d_2 \oplus d_3 \oplus d_4 &= 0 \end{aligned}$$

At the receiver, we check whether the received vector $\underline{\mathbf{y}}$ still satisfies these conditions via computing the **syndrome vector**:

$$\underline{\mathbf{s}} = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \underline{0}?$$

$$\begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \end{array}$$

Parity Check Matrix: Ex 1

- Intuitively, the **parity check matrix** $\mathbf{H}$, as the name suggests, tells which bits in the observed vector $\underline{\mathbf{y}}$ are used to "check" for validity of $\underline{\mathbf{y}}$.
- The number of rows is the same as the number of conditions to check (which is the same as the number of parity check bits).
- For each row, a one indicates that the bits (including the bits in the parity positions) are used in the validity check calculation.

Structure in the codeword:

$$p_1 \oplus d_1 \oplus d_2 \oplus d_4 = 0$$
$$p_2 \oplus d_1 \oplus d_3 \oplus d_4 = 0$$
$$p_3 \oplus d_2 \oplus d_3 \oplus d_4 = 0$$

$\Longleftrightarrow$

$$\begin{array}{cccccccc} & y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 \\ & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ & p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \\ \mathbf{H} = & \multicolumn{7}{c}{\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}} \end{array}$$

Parity Check Matrix: Ex 1

Relationship between **G** and **H**.

$$\begin{array}{cc} & \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \end{array} \\ \mathbf{G} = & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array} \Longleftrightarrow \begin{array}{cc} & \begin{array}{ccccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \end{array} \\ \mathbf{H} = & \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Parity Check Matrix: Ex 1

Relationship between **G** and **H**.

$$\mathbf{G} = \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array} \Longleftrightarrow \mathbf{H} = \begin{array}{ccccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

Parity Check Matrix: Ex 1

Relationship between **G** and **H**.

$$\mathbf{G} = \begin{array}{c} \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array} \Longleftrightarrow \mathbf{H} = \begin{array}{c} \begin{array}{ccccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

(columns of) identity matrix
in the data positions

(columns of) identity matrix
in the parity check positions

Parity Check Matrix: Ex 1

Relationship between **G** and **H**.

$$\mathbf{G} = \begin{array}{ccccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array} \Longleftrightarrow \mathbf{H} = \begin{array}{ccccccc} y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ p_1 & d_1 & p_2 & d_2 & p_3 & d_3 & d_4 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

Parity Check Matrix

Key property:

$$\boxed{\mathbf{G}\mathbf{H}^T = \mathbf{0}_{k\times(n-k)}}$$

Proof:

- When there is no error $(\underline{\mathbf{e}} = \underline{\mathbf{0}})$, the syndrome vector calculation should give $\underline{\mathbf{s}} = \underline{\mathbf{0}}$.
- By definition,

$$\underline{\mathbf{s}} = \underline{\mathbf{y}}\mathbf{H}^T = (\underline{\mathbf{x}} \oplus \underline{\mathbf{e}})\mathbf{H}^T = \underline{\mathbf{x}}\mathbf{H}^T \oplus \underline{\mathbf{e}}\mathbf{H}^T = \underline{\mathbf{b}}\mathbf{G}\mathbf{H}^T \oplus \underline{\mathbf{e}}\mathbf{H}^T.$$

- Therefore, when $\underline{\mathbf{e}} = \underline{\mathbf{0}}$, we have $\underline{\mathbf{s}} = \underline{\mathbf{b}}\mathbf{G}\mathbf{H}^T$.
- To have $\underline{\mathbf{s}} = \underline{\mathbf{0}}$ for any $\underline{\mathbf{b}}$, we must have $\mathbf{G}\mathbf{H}^T = \underline{\mathbf{0}}$.

Systematic Encoding

- Code constructed with distinct information bits and check bits in each codeword are called **systematic codes**.
  - **Message bits are “visible” in the codeword**.
- Popular forms of **G**:

$$\boxed{\mathbf{G} = \left[\mathbf{P}_{k\times(n-k)} \;\vdots\; \mathbf{I}_k\right]}$$

$$\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = \left[b_1 \quad b_2 \quad \cdots \quad b_k\right]\left[\mathbf{P}_{k\times(n-k)} \;\vdots\; \mathbf{I}_k\right]$$

$$= \left[x_1 \quad x_2 \quad \cdots \quad x_{n-k} \;\vdots\; \underset{x_{n-k+1}}{b_1} \quad \underset{x_{n-k+2}}{b_2} \quad \cdots \quad \underset{x_n}{b_k}\right]$$

$$\boxed{\mathbf{G} = \left[\mathbf{I}_k \;\vdots\; \mathbf{P}_{k\times(n-k)}\right]}$$

$$\underline{\mathbf{x}} = \underline{\mathbf{b}}\mathbf{G} = \left[b_1 \quad b_2 \quad \cdots \quad b_k\right]\left[\mathbf{I}_k \;\vdots\; \mathbf{P}_{k\times(n-k)}\right]$$

$$= \left[\underset{x_1}{b_1} \quad \underset{x_2}{b_2} \quad \cdots \quad \underset{x_k}{b_k} \;\vdots\; x_{k+1} \quad x_{k+2} \quad \cdots \quad x_n\right]$$

Parity check matrix

Ex. single-parity-check code

$$\underline{x} = [\, b_1 \;\; b_2 \;\; b_3 \;\; \underbrace{b_1 \oplus b_2 \oplus b_3}_{P} \,]$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \Rightarrow [\, 1 \;\; 1 \;\; 1 \;\; 1 \,]$$

- For the generators matrices we discussed in the previous slide, the corresponding **parity check matrix** can be found easily:

$$\mathbf{G} = \left[\mathbf{P}_{k\times(n-k)} \;\vdots\; \mathbf{I}_k \right] \Rightarrow \mathbf{H} = \left[\mathbf{I}_{n-k} \;\vdots\; -\mathbf{P}^T \right]$$

$$\text{Check: } \mathbf{GH}^T = [\mathbf{P} \;\vdots\; \mathbf{I}] \begin{bmatrix} \mathbf{I} \\ \textbf{-P} \end{bmatrix} = \mathbf{P} \oplus (-\mathbf{P}) = \mathbf{0}_{k\times(n-k)}$$

$$\mathbf{G} = \left[\mathbf{I}_k \;\vdots\; \mathbf{P}_{k\times(n-k)} \right] \Rightarrow \mathbf{H} = \left[-\mathbf{P}^T \;\vdots\; \mathbf{I}_{n-k} \right]$$

Hamming codes: Ex. 2

- Systematic (7,4) Hamming Codes

$$\mathbf{G} = \left[\begin{array}{ccc:cccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathbf{H} = \left[\begin{array}{ccc:cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right]$$

Hamming codes

Now, we will gives a general recipe for constructing Hamming codes.

Parameters:

- $m = n - k =$ number of parity bits
- $n = 2^m - 1 \in \{3,7,15,31,63,127,\ldots\}$
- $k = n - m = 2^m - m - 1$

Example

2, 3, 4, ...

3, 7, 15, ...

1, 4, 11, ...

It can be shown that, for Hamming codes,

- $d_{\min} = 3$.
- Error correcting capability: $t = 1$

Rate : $\frac{k}{n} = \frac{n-m}{2^m-1} = \frac{2^m-1-m}{2^m-1} \longrightarrow 1$ when m is large

Construction of Hamming Codes

Ex. $m=2$ $\quad \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}$

- Start with m.

1. Parity check matrix **H**:
   - Construct a matrix whose columns consist of *all* nonzero binary m-tuples.
   - The ordering of the columns is arbitrary.
     However, next step is easy when the columns are arranged so that $\mathbf{H} = \left[\mathbf{I}_m \mid \mathbf{P}\right]$.

$$H = \begin{bmatrix} \boxed{\begin{matrix}1 & 0\\ 0 & 1\end{matrix}} & \boxed{\begin{matrix}1\\ 1\end{matrix}} \end{bmatrix}$$

2. Generator matrix **G**:
   - When $\mathbf{H} = \left[\mathbf{I}_m \mid \mathbf{P}\right]$, we have $\mathbf{G} = \left[-\mathbf{P}^T \mid \mathbf{I}_k\right] = \left[\mathbf{P}^T \mid \mathbf{I}_k\right]$.

$$G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Hamming codes: Ex. 2

$m = 3$

$n = 2^m - 1 = 2^3 - 1 = 7$

- Systematic (7,4) Hamming Codes

$-\mathbf{P}^T$

$$\mathbf{H} = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right]$$

- Columns are all possible 3-bit vectors
- We arrange the columns so that $\mathbf{I}_3$ is on the left to make the code systematic. (One can also put $\mathbf{I}_3$ on the right.)

$$\mathbf{G} = \left[\begin{array}{ccc|cccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$\mathbf{P}$

- Note that the size of the identity matrices in **G** and **H** are not the same.

Minimum Distance Decoding

- At the decoder, suppose we want to use minimum distance decoding, then
  - The decoder needs to have the list of all the possible codewords so that it can compare their distances to the received vector $\underline{\mathbf{y}}$.
  - There are 2^k codewords each having n bits.
    Therefore, saving these takes $2^k \times n$ bits.
  - Also, we will need to perform the comparison 2^k times.
- Alternatively, we can utilize the syndrome vector (which is computed from the parity-check matrix).
  - The syndrome vector is computed from the parity-check matrix $\mathbf{H}$.
  - Therefore, saving $\mathbf{H}$ takes $(n-k) \times n$ bits.

Minimum Distance Decoding

- Observe that

$$d(\underline{\mathbf{x}},\underline{\mathbf{y}}) = \boldsymbol{w}(\underline{\mathbf{x}} \oplus \underline{\mathbf{y}}) = \boldsymbol{w}(\underline{\mathbf{e}})$$

- Therefore, minimizing the distance is the same as minimizing the weight of the error pattern.
- New goal:
  - find the decoded error pattern $\hat{\underline{\mathbf{e}}}$ with the minimum weight
  - then, the decoded codeword is $\hat{\underline{\mathbf{x}}} = \underline{\mathbf{y}} \oplus \hat{\underline{\mathbf{e}}}$
- Once we know $\hat{\underline{\mathbf{x}}}$ we can directly extract the message part from the decoded codeword if we are using systematic code.
- For example, consider

$$\mathbf{G} = \left[\begin{array}{ccc:cccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

Suppose $\hat{\underline{\mathbf{x}}} = 1011010$, then we know that the decoded message is $\hat{\underline{\mathbf{b}}} = 1010$.

Properties of Syndrome Vector

- From $\mathbf{G}\mathbf{H}^T = \mathbf{0}$, we have

$$\underline{\mathbf{s}} = \underline{\mathbf{y}}\mathbf{H}^T = \left(\underline{\mathbf{x}} \oplus \underline{\mathbf{e}}\right)\mathbf{H}^T = \left(\underline{\mathbf{b}}\mathbf{G} \oplus \underline{\mathbf{e}}\right)\mathbf{H}^T = \underline{\mathbf{e}}\mathbf{H}^T$$

- Thinking of **H** as a matrix with many columns inside,

$$\mathbf{H} = \begin{bmatrix} \underline{\mathbf{h}}_1 \\ \boxed{\quad \underline{\mathbf{h}}_2 \quad} \\ \vdots \\ \underline{\mathbf{h}}_{n-k} \end{bmatrix}_{(n-k)\times n} = \begin{bmatrix} \underline{\mathbf{v}}_1^T & \boxed{\underline{\mathbf{v}}_2^T} & \cdots & \underline{\mathbf{v}}_n^T \end{bmatrix}$$

$$\underline{\mathbf{s}} = \underline{\mathbf{e}}\mathbf{H}^T = \sum_{j=1}^{n} e_j \underline{\mathbf{v}}_j$$

- Therefore, $\underline{\mathbf{s}}$ is a $($linear combination of the columns of **H**.$)^T$

Hamming Codes: Ex. 2

$$\mathbf{H} = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right]$$

$$\underline{\mathbf{s}} = \underline{\mathbf{e}}\mathbf{H}^T = \underbrace{\sum_{j=1}^{n} e_j \underline{\mathbf{v}}_j}_{\text{Linear combination of the columns of } \mathbf{H}}$$

Note that for an error pattern with a single one in the j^{th} coordinate position, the syndrome $\underline{\mathbf{s}} = \underline{\mathbf{y}}\mathbf{H}^T$ is the same as the j^{th} column of $\mathbf{H}$.

Error pattern $\underline{\mathbf{e}}$	Syndrome = $\underline{\mathbf{e}}\mathbf{H}^T$
(0,0,0,0,0,0,0)	(0,0,0)
(0,0,0,0,0,0,1)	(1,1,1)
(0,0,0,0,0,1,0)	(1,1,0)
(0,0,0,0,1,0,0)	(1,0,1)
(0,0,0,1,0,0,0)	(0,1,1)
(0,0,1,0,0,0,0)	(0,0,1)
(0,1,0,0,0,0,0)	(0,1,0)
(1,0,0,0,0,0,0)	(1,0,0)

Properties of Syndrome Vector

- We will assume that the columns of **H** are nonzero and distinct.
  - This is automatically satisfied for Hamming codes constructed from our recipe.
- Case 1: When $\underline{\mathbf{e}} = \underline{\mathbf{0}}$, we have $\underline{\mathbf{s}} = \underline{\mathbf{0}}$.
  - When $\underline{\mathbf{s}} = \underline{\mathbf{0}}$, we can conclude that $\hat{\underline{\mathbf{e}}} = \underline{\mathbf{0}}$.
    - There can also be $\underline{\mathbf{e}} \neq \underline{\mathbf{0}}$ that gives $\underline{\mathbf{s}} = \underline{\mathbf{0}}$.
      - For example, any nonzero $\tilde{\underline{\mathbf{e}}} \in \mathcal{C}$, will also give $\underline{\mathbf{s}} = \underline{\mathbf{0}}$.
      - However, they have larger weight than $\underline{\mathbf{e}} = \underline{\mathbf{0}}$.
    - The decoded codeword is the same as the received vector.
- Case 2: When, $e_i = \begin{cases} 0, & i = j, \\ 1, & i \neq j, \end{cases}$ (a pattern with a single one in the j^{th} position)

  we have $\underline{\mathbf{s}} = \underline{\mathbf{v}}_j$ = the j^{th} column of **H**.
  - When $\underline{\mathbf{s}}$ = the j^{th} column of **H**, we can conclude that $\hat{e}_i = \begin{cases} 0, & i = j, \\ 1, & i \neq j, \end{cases}$
    - There can also be other $\underline{\mathbf{e}}$ that give $\underline{\mathbf{s}} = \underline{\mathbf{v}}_j$. However, their weights
      - can not be 0 (because, if so, we would have $\underline{\mathbf{s}} = \underline{\mathbf{0}}$ but the columns of **H** are nonzero)
      - nor 1 (because the columns of **H** are distinct).
    - We flip the j^{th} bit of the received vector to get the decoded codeword.

Decoding Algorithm

- Assumption: the columns of **H** are nonzero and distinct.
- Compute the **syndrome** $\underline{\mathbf{s}} = \underline{\mathbf{y}}\mathbf{H}^T$ for the received vector.
- Case 1: If $\underline{\mathbf{s}} = \underline{\mathbf{0}}$, set $\underline{\hat{\mathbf{x}}} = \underline{\mathbf{y}}$.
- Case 2: If $\underline{\mathbf{s}} \neq \underline{\mathbf{0}}$,
  - determine the position j of the column of **H** that is the same as (the transposition) of the syndrome,
  - set $\underline{\hat{\mathbf{x}}} = \underline{\mathbf{y}}$ but with the j^{th} bit complemented.
- For Hamming codes, because the columns are constructed from all possible non-zero *m*-tuples, the syndrome vectors must fall into one of the two cases considered.
- For general linear block codes, the two cases above may not cover every cases.

Hamming Codes: Ex. 1

- Consider the Hamming code with

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & \underline{0} & 1 & \underline{0} & 1 & \underline{0} & \underline{1} \end{bmatrix} \Longleftrightarrow \mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

- Suppose we observe $\underline{\mathbf{y}} = [0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1]$ at the receiver. Find the decoded codeword and the decoded message.

$\underline{s} = \underline{y} H^T = (1 \ 1 \ 0)$

Same as the second column of H.

So, we correct 2nd bit of $\underline{y}$:

$\underline{\hat{x}} = [0 \ \underline{0} \ 0 \ \underline{1} \ 1 \ \underline{1} \ \underline{1}]$

$\underline{\hat{b}} = [0 \ 1 \ 1 \ 1]$

[To be explored in the HW]

Hamming Codes: The original method

- Encoding
  - The bit positions that are powers of 2 (1, 2, 4, 8, 16, etc.) are check bits.
  - The rest (3, 5, 6, 7, 9, etc.) are filled up with the *k* data bits.
  - Each check bit forces the parity of some collection of bits, including itself, to be even (or odd).
    - To see which check bits the data bit in position *i* contributes to, rewrite *i* as a sum of powers of 2. A bit is checked by just those check bits occurring in its expansion
- Decoding
  - When a codeword arrives, the receiver initializes a counter to zero. It then examines each check bit at position *i* ($i = 1, 2, 4, 8, \ldots$) to see if it has the correct parity.
  - If not, the receiver adds *i* to the counter. If the counter is zero after all the check bits have been examined (i.e., if they were all correct), the codeword is accepted as valid. If the counter is nonzero, it contains the position of the incorrect bit.

Interleaving

- Conventional error-control methods such as parity checking are designed for errors that are isolated or statistically independent events.
- Some errors occur in bursts that span several successive bits.
  - Errors tend to group together in bursts. Thus, errors are no longer independent
  - Examples
    - impulse noise produced by lightning and switching transients
    - fading or in wireless systems
    - channel with memory
- Such multiple errors wreak havoc on the performance of conventional codes and must be combated by special techniques.
- One solution is to spread out the transmitted codewords.
- We consider a type of interleaving called **block interleaving**.

Interleave as a verb

- To interleave = to combine different things so that parts of one thing are put between parts of another thing
- Ex. To interleave two books together:

Interleaving: Example

Consider a sequence of *m* blocks of coded data:

$$\left(x_1^{(1)}x_2^{(1)}\cdots x_n^{(1)}\right)\left(x_1^{(2)}x_2^{(2)}\cdots x_n^{(2)}\right)\cdots\left(x_1^{(\ell)}x_2^{(\ell)}\cdots x_n^{(\ell)}\right)$$

$$\Big\downarrow$$

$$\begin{array}{cccc} x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(\ell)} & x_2^{(\ell)} & \cdots & x_n^{(\ell)} \end{array}$$

$$\Big\downarrow$$

$$\left(x_1^{(1)}x_1^{(2)}\cdots x_1^{(\ell)}\right)\left(x_2^{(1)}x_2^{(2)}\cdots x_2^{(\ell)}\right)\cdots\left(x_n^{(1)}x_n^{(2)}\cdots x_n^{(\ell)}\right)$$

- Arrange these blocks as rows of a table.
- Normally, we get the bit sequence simply by reading the table by rows.
- With interleaving (by an interleaver), transmission is accomplished by reading out of this table by columns.
- Here, ℓ blocks each of length *n* are interleaved to form a sequence of length ℓn.

The received symbols must be deinterleaved (by a deinterleaver) prior to decoding.

Interleaving: Advantage

- Consider the case of a system that can only correct single errors.
- If an error burst happens to the original bit sequence, the system would be overwhelmed and unable to correct the problem.

original bit sequence $\left(x_1^{(1)} x_2^{(1)} \cdots x_n^{(1)}\right)\left(x_1^{(2)} x_2^{(2)} \cdots x_n^{(2)}\right)\cdots\left(x_1^{(\ell)} x_2^{(\ell)} \cdots x_n^{(\ell)}\right)$

interleaved transmission $\left(x_1^{(1)} x_1^{(2)} \cdots x_1^{(\ell)}\right)\left(x_2^{(1)} x_2^{(2)} \cdots x_2^{(\ell)}\right)\cdots\left(x_n^{(1)} x_n^{(2)} \cdots x_n^{(\ell)}\right)$

- However, in the interleaved transmission,
  - successive bits which come from ***different*** original blocks have been corrupted
  - when received, the bit sequence is reordered to its original form and then the FEC can correct the faulty bits
  - Therefore, single error-correction system is able to fix several errors.

Interleaving: Advantage

- If a burst of errors affects at most ℓ consecutive bits, then each original block will have at most one error.
- If a burst of errors affects at most $r\ell$ consecutive bits (assume $r < n$),
  then each original block will have at most r errors.
- Assume that there are no other errors in the transmitted stream of ℓn bits.
  - A single error-correcting code can be used to correct a single burst spanning upto ℓ symbols.
  - A double error-correcting code can be used to correct a single burst spanning upto 2ℓ symbols.

References: Linear Codes

- Lathi and Ding, ***Modern Digital and Analog Communication Systems***, 2009
  - [TK5101 L333 2009]
  - Chapter 15 p. 907-918
- Carlson and Crilly, ***Communication Systems: An Introduction to Signals and Noise in Electrical Communication***, 2010
  - [TK5102.5 C3 2010]
  - Chapter 13 p. 591-597, 604-611
- Cover and Thomas, ***Elements of Information Theory***, 2006
  - 1991 Version: [Q360 C68 1991]
  - Section 7.11 p. 210-215