SYNDROME DECODING

Definitions of the components/Keywords:

1

• A block code of length n and 2^k codewords is called a ***linear (n,k)*** code if and only if its 2^k codewords form a k- dimensional subspace of the vector space of all the n- tuples over the field *GF(2).*

2

• Any codeword **v = uG**

where **u** is the message and **G** is the generator matrix

The dimension of **v** is 1 x n, **u** is 1 x k and **G** is k x n

3

• The *error vector or error pattern* **e** is the difference between the received n-tuple **r** and the transmitted codeword **v**:

hence the received vector **r** is the vector sum of the transmitted codeword and the error vector.

r = v + e

4

• When **r** is received, the decoder computes the following:

$$s = r.H^T$$

s is called the ***Syndrome*** of ***r***

The dimension of **s** is 1 x n-k, **r** is 1 x n and H^T is n x n-k

5

• Addition of any two codewords results in another codeword and the addition is Modulo – 2 addition

Master Layout

1

2

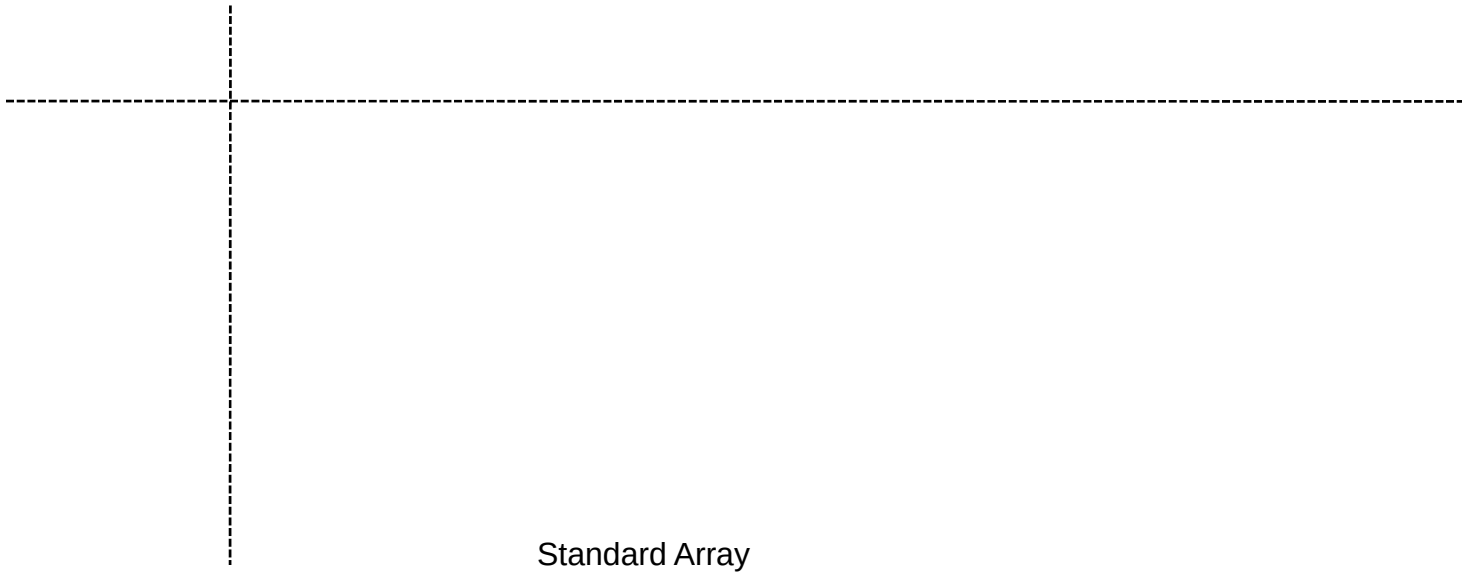
3

4

5

(6,3) linear block code

- Provide a box to enter the received vector that is to be decoded
- The vector is sequence of 1s and 0s



Standard Array

Step 1: Standard Array Decoding

The generator matrix for (6,3) linear code is :

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

Generator Matrix

Instruction for the animator	Text to be displayed in the working area (DT)
• Initially keep the page blank • Then show the first two sentences and show the matrix	• Consider a (6,3) linear code • The generator matrix for the code is shown

Step 2:

All the code words of (6,3) are

C = { 000 000, 001 011, 010 110, 100 101, 011 101, 101 110, 110 011, 111 000}

Instruction for the animator	Text to be displayed in the working area (DT)
• Retain the generator matrix and show the statement and the bit sequences of 1s and 0s given above • Then show the text in DT	• These are the codewords of (6,3) code. • These are obtained by multiplying the message bits with the generator matrix

1

2

3

4

5

Step 3:

000 000 | 001 011 | 010 110 | 100 101 | 011 101 | 101 110 | 110 011 | 111 000

Instruction for the animator	Text to be displayed in the working area (DT)
• First show the text in DT • Then show the bit sequences arranged in the above fashion	• To construct a standard array, place all the codewords in the first row starting with the all -zeros codeword. • Number of rows present in a standard array = 2^{n-k} = 2^{6-3} = 8 rows

1

2

3

4

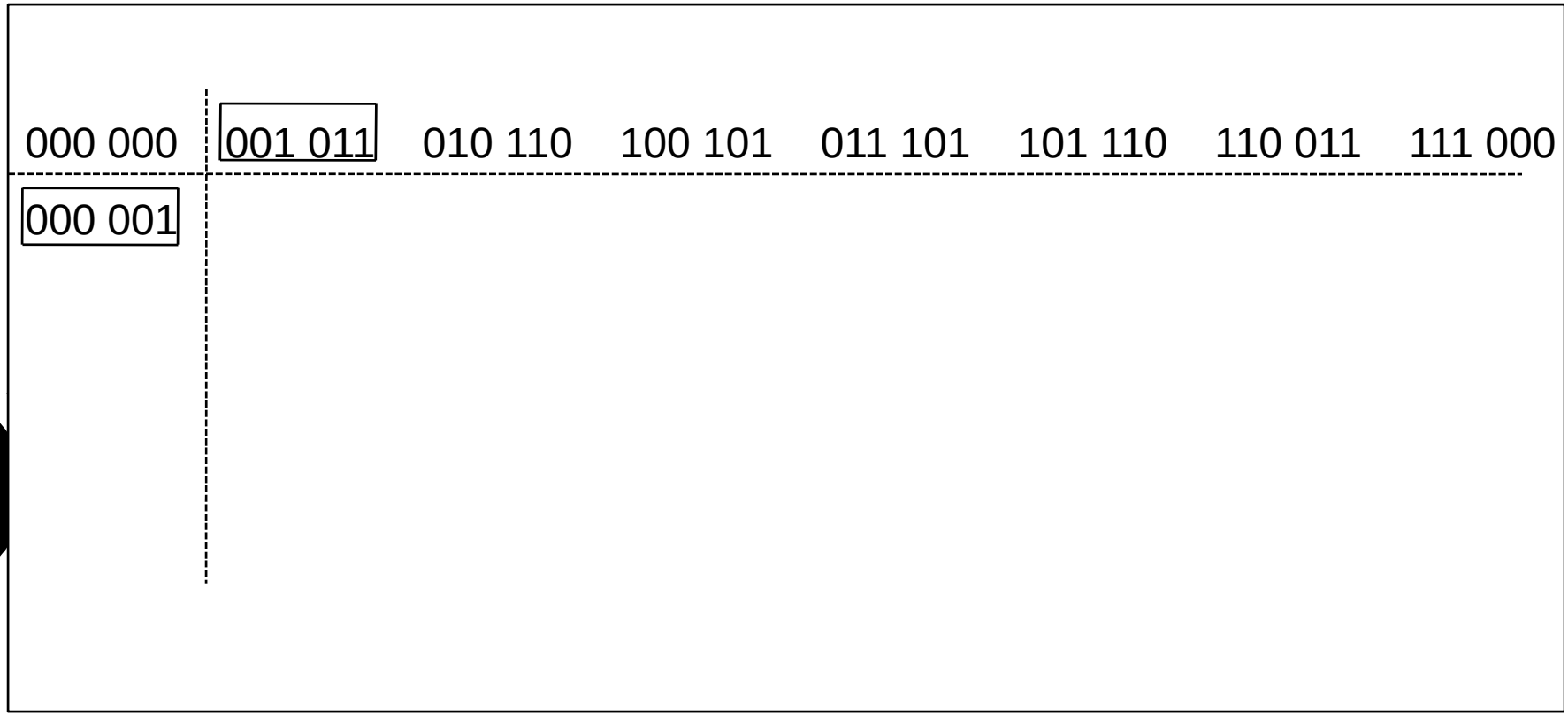
5

Step 4:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001							

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the text in DT	• Choose a minimum weight n-tuple from the remaining n-tuples • Place the n-tuple of weight 1 in the first column of second row • This n-tuple is called the error pattern • Add the error pattern to each of the codewords in the first row to form the second row

Step 5:



Instruction for the animator	Text to be displayed in the working area (DT)
• Show rectangles as shown in the figure • Show the sentences in DT	• Add the error pattern to the first codeword • Here the addition is modulo – 2 addition

Step 6:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010						

Instruction for the animator	Text to be displayed in the working area (DT)
• Then show the result of addition in red colour	• The result of modulo – 2 addition

1

2

3

4

5

Step 7:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010						

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the sentence in DT	• Add the error pattern to the second codeword

Step 8:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111					

Instruction for the animator	Text to be displayed in the working area (DT)
• Then show the result of addition in red colour	• The result of modulo – 2 addition

Step 9:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111					

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the sentence in DT	• Add the error pattern to the second codeword

Step 10:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100				

Instruction for the animator	Text to be displayed in the working area (DT)
• Then show the result of addition in red colour	• The result of modulo – 2 addition

Step 11:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111001

Instruction for the animator	Text to be displayed in the working area (DT)
• Similarly show for entire second row	

Step 12:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111001
000 010							

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the sentence in DT • Show the bit sequence of 1s and 0s in the third row	• Choose another n-tuple of minimum weight from the remaining n-tuples

Step 13:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111 001
000 010	001 001	010 100	100 111	011 111	101 100	110 001	111 010

Instruction for the animator	Text to be displayed in the working area (DT)
• Show rectangles as shown in the figure • Show the sentences in DT • Show the addition for all the codewords similar to the first row (repeat steps 5 and 6)	• Add the error pattern to the first codeword • Here the addition is modulo – 2 addition

Step 14:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111 001
000 010	001 001	010 100	100 111	011 111	101 100	110 001	111 010
000 100	001 111	010 010	100 001	011 001	101 010	110 111	111 100
001 000	000 011	011 110	101 101	010 101	100 110	111 011	110 000
010 000	011 011	000 110	110 101	001 101	111 110	100 011	101 000
100 000	101 011	110 110	000 101	111 101	001 110	010 011	011 000

Instruction for the animator	Text to be displayed in the working area (DT)
• Similarly repeat for all the other 4 bit sequences	

Step 15:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111 001
000 010	001 001	010 100	100 111	011 111	101 100	110 001	111 010
000 100	001 111	010 010	100 001	011 001	101 010	110 111	111 100
001 000	000 011	011 110	101 101	010 101	100 110	111 011	110 000
010 000	011 011	000 110	110 101	001 101	111 110	100 011	101 000
100 000	101 011	110 110	000 101	111 101	001 110	010 011	011 000

Instruction for the animator	Text to be displayed in the working area (DT)
• Then show the text in DT	• All the n-tuples or vectors of weight 1 are exhausted • Now choose another n-tuple of weight 2 from the remaining n-tuples • There are $\binom{6}{2}$ =15 n-tuples of weight 2

Step 16:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111 001
000 010	001 001	010 100	100 111	011 111	101 100	110 001	111 010
000 100	001 111	010 010	100 001	011 001	101 010	110 111	111 100
001 000	000 011	011 110	101 101	010 101	100 110	111 011	110 000
010 000	011 011	000 110	110 101	001 101	111 110	100 011	101 000
100 000	101 011	110 110	000 101	111 101	001 110	010 011	011 000

Instruction for the animator	Text to be displayed in the working area (DT)
• Then show the lines at given places • Also change the colour of those codewords • After this show the text in DT	• Out of 15 n-tuples 12 are present in the array, so take any one of the remaining three n-tuples of weight 2 and place it in the last row

Step 17:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111 001
000 010	001 001	010 100	100 111	011 111	101 100	110 001	111 010
000 100	001 111	010 010	100 001	011 001	101 010	110 111	111 100
001 000	000 011	011 110	101 101	010 101	100 110	111 011	110 000
010 000	011 011	000 110	110 101	001 101	111 110	100 011	101 000
100 000	101 011	110 110	000 101	111 101	001 110	010 011	011 000
100 010							

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the bit sequence in the first column as shown • Then repeat steps 5 and 6	

Step 18:

000 000	001 011	010 110	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111 001
000 010	001 001	010 100	100 111	011 111	101 100	110 001	111 010
000 100	001 111	010 010	100 001	011 001	101 010	110 111	111 100
001 000	000 011	011 110	101 101	010 101	100 110	111 011	110 000
010 000	011 011	000 110	110 101	001 101	111 110	100 011	101 000
100 000	101 011	110 110	000 101	111 101	001 110	010 011	011 000
100 010	101 001	110 100	000 111	111 111	001 100	010 001	011 010

Instruction for the animator	Text to be displayed in the working area (DT)
• Atlast show this entire table and the text in DT	• This is the Standard array for a (6,3) code • The rows in a standard array are called Cosets • The first element in a row of the standard array is called the Coset leader • Coset leaders are exactly the correctable error patterns

Step 19:

Let the received vector **r** be 000 110

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the array and down the array show the above statements	

Step 20:

000 000	001 011	**010 110**	100 101	011 101	101 110	110 011	111 000
000 001	001 010	010 111	100 100	011 100	101 111	110 010	111 001
000 010	001 001	010 100	100 111	011 111	101 100	110 001	111 010
000 100	001 111	010 010	100 001	011 001	101 010	110 111	111 100
001 000	000 011	011 110	101 101	010 101	100 110	111 011	110 000
010 000	011 011	**000 110**	110 101	001 101	111 110	100 011	101 000
100 000	101 011	110 110	000 101	111 101	001 110	010 011	011 000
100 010	101 001	110 100	000 111	111 111	001 100	010 001	011 010

Then the received vector **r** is decoded to be the 010 110 from the Standard array

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the received vector **r** in different colour(blue) • Then show the bit sequence in purple colour and give the statement “the **r** is decoded as “	

Step 21: Syndrome decoding

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the text in DT	• Huge storage memory (and searching time) is required by standard array decoding. • Hence another method called Syndrome decoding is used. • The syndrome depends only on the error pattern and not on the transmitted codeword. • Therefore, each coset in the array is associated with a unique syndrome. • All the n-tuples in a coset have the same syndrome and different cosets have different syndromes. • Syndrome decoding reduces storage memory from $n \times 2^n$ to $2^{n-k}(2n-k)$. Also, It reduces the searching time considerably.

Step 22:

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

- We know **r = v + e** and $s = r.H^T$
- Therefore $s = e.H^T$ since $v.H^T$ **= 0**

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the text in DT along with the matrix • Also show the text below the matrix	• H is the parity check matrix

Step 23:

$$S = e.H^T = \begin{bmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} e_1 + e_2 + e_4 & e_2 + e_3 + e_5 & e_1 + e_3 + e_6 \end{bmatrix}$$

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the above text	

Step 24:

1
2
3
4
5

Coset Leader/ error pattern	Syndrome
000 000	000
000 001	001
000 010	010
000 100	100
001 000	011
010 000	110
100 000	101
100 010	111

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the above text	•

Step 25:

Suppose **r =** 000 110

$$s = r.H^T = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

= 1 1 0

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the above text	•

Step 25:

1

2

3

4

5

Then compute $\hat{v} = r + e$

where $\hat{v}$ is the estimated codeword

r is the received vector

e is the error pattern corresponding to the syndrome S

Therefore $\hat{v}$ = 000 110 + 010 000 = 010 110

Hence **r** is decode as 010 110

Instruction for the animator	Text to be displayed in the working area (DT)
• Show the above text	•

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

The user should be able to provide the remaining 9 elements with 1s and 0s

Interactivity:

Try it yourself

- Provide a box to enter the received vector

Project OSCAR
Open Source Courseware Animations Repository

Summary

- **Syndrome decoding** is a more efficient method of decoding a linear code over a noisy channel. Syndrome decoding is *minimum distance decoding* using a reduced lookup table.
- A block code of length n and 2^k codewords is called a ***linear (n,k)*** code if and only if its 2^k codewords form a k - dimensional subspace of the vector space of all the n- tuples over the field *GF(2).*
- Any codeword **v = uG**

  where **u** is the message and **G** is the generator matrix

  The dimension of **v** is 1 x n, **u** is 1 x k and **G** is k x n
- The *error vector or error pattern* **e** is the difference between the received n-tuple **r** and the transmitted codeword **v**:

hence the received vector **r** is the vector sum of the transmitted codeword and the error vector.

r = v + e

- When **r** is received, the decoder computes the following:

$$s = r.H^T$$

s is called the ***Syndrome*** of ***r***

The dimension of **s** is 1 x n-k, **r** is 1 x n and H^T is n x n-k