Universidade de Lisboa

IST

MEEC

DIGITAL TRANSMISSION

2nd Practical Work

- Linear Binary Code –

-1st Phase –

Class: *TDig11* **Group:** **Date:** *2025/2026*

Number: *ist1115593* **Name:** *Raj Maharjan*

Number: *ist1118696* **Name:** *Kristine Saralidze*

Work Evaluation: _____ **Professor: Dr. Marko Beko**

Accademic Year 2024/2025

1. OBJECTIVES

In a digital telecommunication system, the existence of errors is the genesis of the use of error detection and correction codes. Although it increases redundancy, that is, reducing the useful binary throughput, it makes communication more robust against errors. A linear error correction code can be adjusted by concatenating the symbols to transmit a certain sequence of symbols that depends on the symbols to be transmitted.

The receiver operates on the received data and determines the sequence of data that originated it.
Throughout the second part of the statement, variable names are mentioned and highlighted in **Bold**. These variables, and only these, are grouped in the *workspace* and must be sent (**mandatorily**) along with the code file of the assignment and the duly completed statement.

Consider that the information bits are organized into blocks of size and, before being sent to the channel, are transformed into codewords (c_l) of size n. This transformation is performed through the concatenation (systematic form) of $n-k$ bits test bits (b_i) to the k information bits (m_i). **Figure 1** summarizes the aforementioned linear code.

$$k\ bits \rightarrow \boxed{linear\ code(n,k)} \rightarrow n\ bits$$

data blocks — code word

Figure 1

In matrix/vector terms, the relationship outlined in the previous figure is expressed as follows: $\underset{1\times k}{\boldsymbol{m}} \rightarrow \boxed{linear\ code(n,k)} \rightarrow \underset{1\times n}{\boldsymbol{c}}$

Figure 2

As mentioned earlier,

$$\mathbf{c} = \left[\mathbf{b} \,\vdots\, \mathbf{m}\right]$$

The parity test bits ($\boldsymbol{b}$) are obtained through the parity matrix ($\overline{\boldsymbol{P}}$)

$$\mathbf{b} = \mathbf{m}\,\overline{\mathbf{P}}$$

$$\mathbf{b} \rightarrow 1 \times n-k$$
$$\mathbf{m} \rightarrow 1 \times k$$
$$\overline{\mathbf{P}} \rightarrow k \times (n-k)$$

Through **Figure 2,** and the two previous relationships, the generator matrix $\overline{\mathbf{G}}$

$$\boldsymbol{c} = \boldsymbol{m}\ \overline{\boldsymbol{G}}$$
$$\overline{\boldsymbol{G}} \rightarrow k \times n$$

Which is written in the systematic form as follows:

$$\overline{\mathbf{G}} = \left[\ \overline{\mathbf{P}}\ \vdots\ \overline{\mathbf{I}}_k\ \right]$$

In the previous expression, $\overline{\boldsymbol{I}}_k$ is the identity matrix of size k. N Note that if the matrix $\overline{\boldsymbol{P}}$ is inverted with $\overline{\boldsymbol{I}}_k$, the codeword $\boldsymbol{c}$ is also inverted. $\mathbf{c} = \left[\mathbf{m}\ \vdots\ \mathbf{b}\right]$

At the receiver, after the transmission of the codeword through the channel, it is necessary to determine which information word $\boldsymbol{m}$ originated $\boldsymbol{c}$. Since $\overline{\mathbf{G}}$ is in systematic form, in the absence of errors, recovering the word $\boldsymbol{m}$ is trivial, as it is sufficient to isolate the bits corresponding to the information word.

In general, at the receiver, the received word ($\boldsymbol{r}$) is the sum of the transmitted word and the error vector or error pattern ($\boldsymbol{e}$).

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}$$

The error vector will have "1" in the positions where an error occurred. With this in mind, it is evident that when errors occur, the received word may not be a valid codeword. However, there is a certain number of errors for which the received word $\boldsymbol{r}$ becomes another valid codeword, making it impossible to determine whether an error occurred or not.

The procedure used by the receiver to determine the transmitted word begins with the calculation of the syndrome vector, which is obtained by multiplying the received word by the parity-check matrix.

$$\boldsymbol{s} = \boldsymbol{r}\ \overline{\boldsymbol{H}}$$
$$\boldsymbol{s} \rightarrow\ 1 \times (n-k)$$

The parity-check matrix is such that:

$$\mathbf{c}\overline{\mathbf{H}} = 0$$

Thus, it is demonstrated that:

$$\overline{\mathbf{H}} = \begin{bmatrix} \mathbf{I}_{n-k} \\ \hline \mathbf{P} \end{bmatrix}$$

$$\overline{\mathbf{H}} \rightarrow n \times (n-k)$$

Note that:

$$\overline{\mathbf{G}}\,\overline{\mathbf{H}} = \mathbf{0}$$

Considering the previous properties:

$$\mathbf{s} = \mathbf{r}\,\overline{\mathbf{H}} = (\mathbf{c}+\mathbf{e})\overline{\mathbf{H}} = \mathbf{c}\overline{\mathbf{H}} + \mathbf{e}\overline{\mathbf{H}} = 0 + \mathbf{e}\overline{\mathbf{H}}$$

It is verified that in the absence of errors:

$$\mathbf{e} = \mathbf{0} \Rightarrow \mathbf{s} = \mathbf{0}$$

As demonstrated in the theoretical class, the performance of the code depends on the characteristics of the matrix $\overline{\boldsymbol{H}}$, specifically on its rows. The minimum distance of the code is equal to the smallest number of rows of $\overline{\boldsymbol{H}}$ that, when summed, produce the null vector.

The distance between two binary vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of length l is defined as:

$$d(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{l} \left(|a_i - b_i| \right)$$

A Hamming code must satisfy the following relationship:

$$k = 2^p - p - 1$$

where:

$$p = n - k$$

In this assignment, the goal is to implement a linear Hamming error correction code (n, k). Where the parity-check matrix is in systematic form.

An algorithm that allows generating the parity matrices $\overline{\boldsymbol{P}}$, the generator matrix $\overline{\mathbf{G}}$, and the parity-check matrix $\overline{\boldsymbol{H}}$ will be described below. Once the length of the encoded word $\mathbf{c}$ is defined, the positions of the bits are numbered.

$$\mathbf{c} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & \cdots & 255 \end{bmatrix}$$

Next, the positions are written in binary notation.

$$\mathbf{c} = \begin{bmatrix} 1 & 10 & 11 & 100 & 101 & 110 & 111 & \cdots & 11111111 \end{bmatrix}$$

Thus, the parity bits are those that have only one digit set to 1 in their binary representation, corresponding to positions 1, 2, 4, 8, 16, 32, 64, 128, and so on. All other positions will be data bits. The method to obtain the parity bits from the data bits is as follows:

1st parity bit: Sum of all data bits whose position has a 1 in the 1st least significant bit.

2nd parity bit: Sum of all data bits whose position has a 1 in the 2nd least significant bit.

3rd parity bit: Sum of all data bits whose position has a 1 in the 3rd least significant bit.

…

8th parity bit: Sum of all data bits whose position has a 1 in the 8th least significant bit.

To ensure that you have correctly understood the procedure, verify if for Hamming (7,4) you obtain the expected result.

2. REPORT AND EXECUTION OF THE WORK IN MATLAB.

2.1) **(Theoretical)** What is the minimum distance of a Hamming code? Justify.

The minimum distance of any Hamming code is **3**.

Hamming codes are designed to correct exactly one bit-error. For a code to correct **t** error, the minimum distance must be at least: $d_{min} >= 2*t + 1$. i.e for t = 1, $d_{min} = 3$.

For example, take the (7,4) Hamming code: all non-zero codewords have at least 3 bits different from each other, so the d_{min} between any two codewords is 3. The reason is that Hamming codes are designed to correct one error and detect two errors. A d_{min} of 3 means that if 1 bit flips, it's still closer to the original codeword than to any other (can correct it) and if 2 bits flip, we can detect an error (because it won't be a valid codeword), but can't be sure which original codeword it came from.

So, for any Hamming code, no matter its length, the d_{min} is always 3.

2.2) (**Theoretical**) What is the error detection capability of a Hamming code? Justify.

The error-detection capability of a Hamming code is 2 errors.

As established above, a hamming code has $d_{min} = 3$. Hamming code can detect t errors if $d_{min} > t$. This is true for t=1 and t=2. i.e. can detect 2 errors

Example: in the (7,4) Hamming code, if we transmit 0000000 and errors flip bits 1, 2, and 4, we might accidentally get another valid codeword like 1110001, so we wouldn't detect an error.

2.3) (**Theoretical**) A Hamming code is a perfect code for correcting single-bit errors. Why?

Hamming code is perfect for correcting single-bit errors as for even k = 1, we have n = 3, with d(k) = 3, hence for the code to be corrected the bounds specified by , is always satisfied and hence we're able to get the correct codeword even when it consists of an error bit.

Ex. 000 transmitted but receiver got , but using the maximum likelihood decoding, we know that the corruption in the code and easly resolve it.

2.4) Consider $p = n - k = 8$

2.5) (**Theoretical**) Calculate all dimensions of relevant vectors and matrices

Given p = 8, then we have 8 homogeneous equations (parity check equations) with n unknowns.

b row vector of test bits = n−k = 8

Codeword = n also, p = n-k → n = 8 + k ⇒ (8+k)

Information vector = k also p = n-k → k = n - 8 ⇒ (n-8)

Parity Matrix = k × (n-k) = k × 8

Generation Matrix = k × n = (n-8) × n

Parity Check Matrix = n × 8 = (8+k) × 8

Also, we know Codeword × (Parity Check Matrix)^T = 1 × 8

2.6) Determine the Parity-Check Matrix. Comment on its properties.

The most important property of a Parity-Check matrix is that when multiplied with a legitimate codeword c,

$$cH = [m \mid b] \times [\frac{P}{I_{n-k}}] = 0$$

Also since cH = 0 then elementary operations on the rows of H do not change the associated system of homogeneous linear equations.

- Matrix Name: **H**

2.7) Calculate the minimum number of rows of the Parity-Check Matrix that must be summed to result in the null vector. Indicate the indices of the rows of the first set you found. Comment on the results.

min_col_sum_zero=3

index_min_col_sum_zero=1,2,26

The results speak for themselves as the minimum hamming distance is 3, which is what we've got from the result.

- Variable name: **min_col_sum_zero**

- Variable name with the indices of the rows: **index_min_col_sum_zero** (in one line)

2.8) Import the syndrome variable into MATLAB using the function: load('Sindroma_G0y_T1_F1').

Note: **y** corresponds to the number of your laboratory group.

Example: The student a21401268 **imports the file Sindroma_G06_T1_F1.wav (2+1+4+0+1+2+6+8 = 24 = 2+4 = 6)**

2.9) Which sets of rows of **H** need to be summed, with a maximum of 2 terms, to equal the indicated syndrome?

The single row solution is {242}. And there are many 2 row solutions: {1,211},{2,20},{3,132},{4,55},{5,227},…,{232,253}. Total number of solutions = 128

Variable Name: **col_sum2_sind** (It has the same number of columns as the sets of rows summed. The content is the index of the row of H if it involves only one term, place zero in the second column.)

2.10) (**Theoretical**) What is the relationship between the row indices of **H** from **2.9)** and the vector $\boldsymbol{e}$?

The row indices of the matrix H, (i.e. some h_i=i^{th} row of the matrix H) when multiplied with e → error pattern (corresponding to the value for h_i) would produce the syndrome.

$$e'H = [e_1 \cdots e_n][h_1 \cdots h_n]^T = e_1[h_1] + \cdots + e_n[h_n] = s$$

where $s = [h_j] + [h_k] + \cdots$ such that $e_j = e_k = \cdots = 1$; i.e. j, k have errors.

Syndrome is the sum of the rows of H corresponding to the erroneous bits.

2.11) (**Theoretical**) Within the sets calculated in **2.9)**, which one is the most likely? Why?

The most likely ones corresponds to the eH = s

where s matches exactly one row of H .

Single bit errors are much more probable, than multi-bit error as the probability decreases by a factor of P. i.e. for Pe = 0.001 the probability of 1 error bit is Pe

while the probability for 2 error bit is P2 e . Hence the (242, 0) is the more probable within the set calculated, which is the only single error value.

2.12) **(Theoretical)** For this code, is there more than one error pattern that generates the same syndrome?

Yes, we found 128 combinations of 2 rows when summed up generates the same syndrome.

2.13) Save only the variables highlighted in **nigrito** in a workspace with the following name:

G0z_**y**_T1_F1

2.14) Save the MATLAB code file with the following name:

G0_**y**_T1_F1

where **y** corresponds to the student number