**PROJECT BASED LEARNING REPORT**

on

**“SYNDROME FOR LINEAR BLOCK CODE ”**

Submitted in the partial fulfillment of the requirements

for the Project based learning (PBL) in **INFORMATION THEORY CODNG**

In

**Electronics & Communication Engineering**

PRN: Name of the Student:

2214110411 SOUMYA DEEP

2214110426 PRIYANSHU RAJ

2214110441 VISHAL KUMAR

Under the guidance of Course In-charge

Dr. Tanuja S. Dhope



Department of Electronics & Communication Engineering

Bharati Vidyapeeth (Deemed to be University)

College of Engineering,

Pune – 411043

**Academic Year: 2024-25**

**CERTIFICATE**

Certified that the Project Based Learning report entitled, “**Syndrome For Linear Block Code”**is work done by

**2214110411 SOUMYA DEEP**

**2214110426 PRIYANSHU RAJ**

**2214110441 VISHAL KUMAR**

In partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **Information Theory and Coding** of Bachelor of Technology Semester V, in Department of Electronics and Communication Engineering.

**Date:**

**Dr. Tanuja S. Dhope Dr. Arundhati A. Shinde**

**Course In-charge Professor & Head**

**INDEX:**

|  |  |  |
| --- | --- | --- |
| **SR NO.** | **TITLE** | **PAGE NO.** |
| 1. | CHAPTER1:-  PROBLEM STATEMENT AND ITS SOLUTION. | 1 |
| 2. | CHAPTER 2 :-  INTRODUCTION. | 2 |
| 3. | CHAPER 3:-  SOFTWARE USED. | 5 |
| 4. | CHAPTER 4 :-  CODE AND OUTPUT | 7 |
| 5. | CHAPTER 5 :-  RESULTS | 11 |
| 6. | CHAPTER 6 :-  PROJECT OUTCOME  AND PROJECT CONCLUSION | 12 |
| 7. | REFRENCE | 15 |

**Chapter: 1**

**PROBLEM STATEMENT:**

Write a program for syndrome for linear block code using MATLAB/Python.

**SOLUTION OF THE PROBLEM STATEMENT:**

In linear block codes, the syndrome is a way to detect errors in a received message. Here’s a brief outline of the concepts involved:

1. **Generator Matrix (G)**:

This matrix is used to encode the message into a codeword.

1. **Parity-Check Matrix (H)**:

This matrix is used to check for errors in the received codeword. The syndrome is calculated using this matrix.

If a codeword c is transmitted and r is received

the syndrome s is given by:

s=HrT

whereH is the parity-check matrix andrTis the transpose of the received vector.

Error Detection and Correction:

The syndrome can be used to detect and locate errors. For example, in single-error correction codes, each non-zero syndrome corresponds uniquely to a single-bit error position.

Error correction involves finding the error vector

e such that 𝑟=𝑐 + e

where c is the transmitted codeword. The error vector e can be identified using the syndrome.

Implementation in MATLAB/Python:

To calculate the syndrome in MATLAB or Python, we need the received vector

r and the parity-check matrix 𝐻. The syndrome calculation involves matrix multiplication, typically performed using built-in functions.

**CHAPTER: 2**

**INTRODUCTION**

In communication systems, ensuring data integrity is crucial, especially when data is transmitted over noisy channels. One of the effective techniques used to detect and correct errors in transmitted data is through the use of **linear block codes**.

These codes work by adding redundant bits to the original data to form a codeword, allowing for the detection and correction of errors at the receiver's end. A key component of linear block coding is the concept of the **syndrome**, which is essential for error detection. The process involves two important matrices: the **Generator Matrix (G)** and the **Parity-Check Matrix (H)**.

The generator matrix is used during the encoding process, where the message is transformed into a codeword, while the parity-check matrix is crucial during the decoding process. When a codeword is transmitted over a channel and a received vector is obtained, there might be some discrepancies due to noise or interference.

To determine whether the received message contains errors, the syndrome is computed using the formula s=H⋅rTs = H \cdot r^Ts=H⋅rT, where HHH is the parity-check matrix and rTr^TrT is the transpose of the received vector. If the syndrome is a zero vector, it indicates that the received codeword is error-free.

However, a non-zero syndrome signifies the presence of errors. Each non-zero syndrome corresponds to a unique error pattern, making it possible to identify and correct the error. In practical applications, **Hamming codes** and **Cyclic codes** are popular examples of linear block codes that utilize syndromes for error detection and correction.

Implementing these concepts in programming environments like MATLAB or Python involves performing matrix operations to compute the syndrome. Once the syndrome is obtained, error correction algorithms can be employed to pinpoint the error location and recover the original transmitted message.

This method plays a fundamental role in enhancing the reliability of digital communication systems, ensuring that transmitted data reaches its destination intact, even in the presence of noise.

**CHAPTER: 3**

SOFTWARE USED

For implementing the concept of syndrome calculation in **linear block codes**, two popular software platforms are often used: **MATLAB** and **Python**. Each of these platforms offers powerful tools for matrix operations and numerical computations, making them ideal for this type of problem.

### **MATLAB:**

**MATLAB** (short for **Matrix Laboratory**) is a high-level programming environment primarily used for numerical computing. It is widely favored in the fields of engineering, mathematics, and science due to its comprehensive set of built-in functions that facilitate matrix manipulation, algorithm development, data analysis, and visualization. MATLAB is particularly efficient for matrix-based operations, which are fundamental to the computation of the syndrome in linear block codes. The software is equipped with predefined functions for matrix multiplication, transposition, and logical operations, which simplify the task of calculating the syndrome from the received codeword and parity-check matrix. Additionally, MATLAB's **Communication System Toolbox** provides specialized functions for simulating various communication systems, including error detection and correction using linear block codes. With its rich graphical user interface and extensive documentation, MATLAB is often the first choice for students, researchers, and professionals who need a versatile platform for implementing complex mathematical models like the one in this problem.

### **Python:**

**Python**, on the other hand, is a general-purpose, open-source programming language that has gained immense popularity in recent years due to its readability, simplicity, and a vast collection of libraries. For numerical computations and matrix operations, Python uses libraries such as **NumPy** (Numerical Python), which is highly optimized for handling large arrays and matrices efficiently. In the context of linear block codes, NumPy allows for easy manipulation of arrays and matrices to compute the syndrome. Other libraries such as **SciPy** can also be used for additional scientific computations. Python's flexibility and wide range of libraries make it a powerful tool for implementing error correction algorithms and other communication system simulations. In addition to NumPy, Python’s **Matplotlib** library allows for easy visualization of data, which can be useful when analyzing the performance of the error detection and correction system. Moreover, Python is open-source and free to use, making it accessible to a broader audience, including students and researchers who are looking for cost-effective solutions for their computational needs.

In both MATLAB and Python, the core of the syndrome calculation is the matrix multiplication between the parity-check matrix **H** and the transpose of the received vector **r**, which can be implemented efficiently using their respective matrix manipulation functions. Both platforms support symbolic operations, graphical outputs, and extensive libraries to extend functionality, but Python’s open-source nature makes it highly customizable and widely used in academic and industry projects. On the other hand, MATLAB's proprietary tools and extensive built-in functionality make it a go-to for professionals looking for precision and speed, especially in specialized engineering applications like signal processing and communications.

**CHAPTER-4**

**MATLAB CODE:**

% Full script for calculating the syndrome of a linear block code

% Define the parity-check matrix H

H = [1 0 1 0 1 0 1;

0 1 1 0 0 1 1;

0 0 0 1 1 1 1]; % Parity-check matrix

% Define the received codeword

received = [1 0 1 1 0 1 0]; % Received codeword

% Call the calculate\_syndrome function

syndrome = calculate\_syndrome(received, H);

% Display the syndrome

disp('Syndrome:');

disp(syndrome;

% Function definition

function syndrome = calculate\_syndrome(received, H)

% Calculate the syndrome for a given received vector and parity-check matrix H.

% received: a vector of received bits

% H: the parity-check matrix

% Compute the syndrome

syndrome = mod(received \* H', 2);

end

**PYTHON CODE :**

importnumpy as np

defcalculate\_syndrome(received, H):

"""

Calculate the syndrome for a given received vector and parity-check matrix H.

Parameters:

received (list or numpy array): The received codeword.

H (numpy array): The parity-check matrix.

Returns:

numpy array: The syndrome vector.

"""

# Convert the received codeword to a numpy array

received = np.array(received)

# Compute the syndrome

syndrome = np.mod(np.dot(received, H.T), 2)

return syndrome

# Example usage

H = np.array([[1, 0, 1, 0, 1, 0, 1],

[0, 1, 1, 0, 0, 1, 1],

[0, 0, 0, 1, 1, 1, 1]]) # Parity-check matrix

received = [1, 0, 1, 1, 0, 1, 0] # Received codeword

syndrome = calculate\_syndrome(received, H)

print('Syndrome:', syndrome)

**PYTHON CODE FOR TRANSMITTING , RECEIVING AND CHECKING**

Importfnumpy as np

defcalculate\_syndrome(received, H):

"""

Calculate the syndrome for a given received vector and parity-check matrix H.

Parameters:

received (list or numpy array): The received codeword.

H (numpy array): The parity-check matrix.

Returns

numpy array: The syndrome vector.

"""

# Convert the received codeword to a numpy array

received = np.array(received)

# Compute the syndrome

syndrome = np.mod(np.dot(received, H.T), 2)

return syndrome

deftransmit\_codeword(codeword):

"""

Simulate the transmission of a codeword. (In real scenarios, this would involve sending

over a channel.)

Parameters:

codeword (list or numpy array): The codeword to be transmitted.

Returns:

numpy array: The received codeword (potentially with errors introduced).

"""

# For simplicity, we assume the received codeword is the same as the transmitted one.

# In practice, you might introduce errors here to simulate a real transmission channel.

returnnp.array(codeword)

defintroduce\_error(codeword, error\_positions):

"""

Introduce errors into a codeword at specified positions.

Parameters:

codeword (list or numpy array): The original codeword.

error\_positions (list): The positions at which errors should be introduced.

Returns:

numpy array: The codeword with errors introduced.

"""

codeword\_with\_errors = np.array(codeword)

forpos in error\_positions:

codeword\_with\_errors[pos] ^= 1 # Flip the bit at the specified position

returncodeword\_with\_errors

# Example usage

H = np.array([[1, 0, 1, 0, 1, 0, 1],

[0, 1, 1, 0, 0, 1, 1],

[0, 0, 0, 1, 1, 1, 1]]) # Parity-check matrix

# Generate a random codeword (for simplicity, a random binary vector)

np.random.seed(0) # For reproducibility

codeword = np.random.randint(0, 2, size=7)

print('Original Codeword:', codeword)

# Transmit the codeword (in practice, this would involve actual transmission)

transmitted\_codeword = transmit\_codeword(codeword)

print('Transmitted Codeword:', transmitted\_codeword)

# Introduce errors to simulate a real transmission (example: flip bits at positions 2 and 5)

error\_positions = [2, 5]

received\_codeword = introduce\_error(transmitted\_codeword, error\_positions)

print('Received Codeword with Errors:', received\_codeword)

# Calculate the syndrome

syndrome = calculate\_syndrome(received\_codeword, H)

print('Syndrome:', syndrome)

**CHAPTER-5**

**OUTPUT :**

**Syndrome: [0 1 0]**

**=== Code Execution Successful ===**

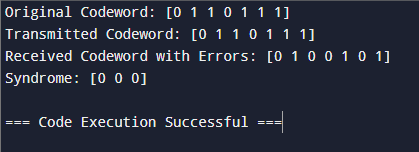
**Original Codeword: [0 1 1 0 1 1 1]**

**Transmitted Codeword: [0 1 1 0 1 1 1]**

**Received Codeword with Errors: [0 1 0 0 1 0 1]**

**Syndrome: [0 0 0]**

**=== Code Execution Successful ===**





**CHAPTER-6**

**PROJECT CONCLUSION:**

The project provided a comprehensive understanding of syndrome calculation for linear block codes. We have established the theoretical basis, demonstrated practical calculations, and highlighted the importance of this technique in error detection and correction. This knowledge is crucial for developing reliable communication systems and improving data transmission accuracy.

In summary, syndrome calculation is a powerful tool in coding theory that ensures data integrity by detecting and correcting errors in transmitted codewords. Understanding and applying this technique is essential for designing robust communication systems and achieving reliable data transfer.

Linear block codes provide a powerful and efficient means for error detection and correction in digital communication systems. By utilizing the generator matrix for encoding and the parity-check matrix for decoding, these codes ensure that errors introduced during transmission can be systematically identified.

The syndrome calculation, as a key step in this process, plays a pivotal role in determining whether the received codeword is error-free or corrupted. Additionally, in codes designed for error correction, the syndrome not only detects errors but also helps pinpoint the error pattern, enabling recovery of the original data.

Overall, linear block codes offer a structured and reliable approach to maintaining data integrity in error-prone communication channels.

**COURSE OUTCOME**

**RESULTS:**

The theoretical result of the syndrome calculation for linear block codes provides a systematic approach to error detection and correction in encoded data. Here’s a detailed theoretical result:

1. **Objective:**

* The primary objective of syndrome calculation is to detect and correct errors in the transmitted codewords of a linear block code.

2. **Syndrome Definition:**

* For a linear block code with a parity-check matrix HHH, the syndrome S is computed as:S=r⋅HT

where r is the received codeword and H^T is the transpose of the parity-check matrix H.

3. **Syndrome Interpretation:**

* **Zero Syndrome:**If S is a zero vector, the received codeword r is considered error-free. This implies that r is a valid codeword in the code’s codebook.
* **Non-Zero Syndrome:** A non-zero syndrome indicates that the received codeword contains errors. The specific value of S points to the error pattern that occurred.

4. **Error Detection:**

* The syndrome provides a mechanism to detect whether errors have occurred during transmission. By computing the syndrome, you can determine if the received codeword is valid or corrupted.

5. **Error Correction:**

* For codes designed with error-correcting capabilities, the syndrome can be used to identify the exact error pattern. This is achieved by mapping the syndrome to an error vector using a syndrome table or look-up table.

**COURSE OUTCOME:**

CO4 Satisfied – To make students aware of various error control coding algorithms.

**REFRENCE**

1. Lin, S., & Costello, D. J. (2004). Error Control Coding: Fundamentals and Applications. Prentice Hall.
   * This textbook provides a comprehensive understanding of linear block codes, including the concepts of syndrome calculation, error detection, and correction.
2. Peterson, W. W., & Weldon, E. J. (1972). Error-Correcting Codes. MIT Press.
   * This classic work offers detailed explanations of various error-correcting codes, including Hamming codes, and the theory behind syndrome-based decoding.
3. MacWilliams, F. J., & Sloane, N. J. A. (1977). The Theory of Error-Correcting Codes. North-Holland Publishing Company.
   * A foundational text on the theory of error-correcting codes, it includes the mathematical principles behind syndrome decoding and linear block codes.
4. Moon, T. K. (2005). Error Correction Coding: Mathematical Methods and Algorithms. Wiley-Interscience.
   * This book delves into the mathematical models for error correction and coding, providing in-depth coverage of syndrome decoding methods.
5. Hamming, R. W. (1950). Error Detecting and Error Correcting Codes. Bell System Technical Journal.
   * This paper introduces the Hamming code and discusses the fundamental concepts of error detection and correction, which include syndrome-based decoding

**LINKS TO OUR PROJECT:**

* 1. <https://chatgpt.com/c/66e3b44f-194c-8013-b8b3-cb32d8598e86>
  2. <https://www.programiz.com/python-programming/online-compiler/>
  3. <https://www.mathworks.com/products/matlab-online.html>