PROJECT BASED LEARNING REPORT

on

“CONVOLUTION CODES”

Submitted in the partial fulfillment of the requirements

for the Project based learning (PBL) INFORMATION THEORY AND COADING

in

Electronics & Communication Engineering

By

PRN NAME ROLL NUMBER

2214110431 MOHI SANJEEV 24

2214110435 SHAYAN KUMAR 27

2214110448 SHUBHAM THAKUR 37

Under the guidance of Course In-charge

Prof. Dr.TANUJA S.DHOPE



Department of Electronics & Communication Engineering

Bharati Vidyapeeth (Deemed to be University)

College of Engineering,

Pune – 4110043

Academic Year: 2024-25

**CERTIFICATE**

Certified that the Project Based Learning report entitled, “CONVOLUION CODES ON MATLAB” is work done by

2214110431 MOHI SANJEEV

2214110435 SHAYAN KUMAR

2214110437 SHUBHAM THAKUR

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in Electromagnetic Waves & Propagation of Bachelor of Technology Semester V, in Department of Electronics and Communication Engineering.

Date:

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

Course In-charge Professor & Head

**INDEX**

|  |  |  |
| --- | --- | --- |
| SR.NO. | CONTENT | PAGE NO. |
| 1 | Chapter 1   * Why do we need Convolution codes * Solution of convolution codes | 1-3 |
| 2 | Chapter-2   * Introduction | 4-5 |
| 3 | Chapter-3   * Software Used | 6-7 |
| 4 | Chapter-4   * Code | 8-10 |
| 5 | Chapter-5   * Output | 11 |
| 6 | Chapter-6   * Conclusion * Application * Outcome * Reference * Github link | 12-14 |

**CHAPTER-1**

**Why Do We Need “Convolution Codes”**

Convolutional codes are a key part of error correction techniques in information theory, used to improve the reliability of data transmission over noisy communication channels. Here’s why they are important:

1. Error Detection and Correction

- Communication channels, such as wireless networks or satellite links, often introduce noise, causing data corruption.

- Convolutional codes allow receivers to detect and correct errors without the need for retransmission, ensuring that the transmitted data is accurately reconstructed.

2. Continuous Encoding Process

- Unlike block codes (e.g., Hamming codes) which encode data in fixed-sized blocks, convolutional codes operate continuously on a stream of data.

- This makes convolutional codes ideal for real-time communication systems where data is transmitted continuously, such as in voice and video transmission.

3. Memory Aspect

- Convolutional encoders have memory, meaning the output depends on both the current input and previous inputs. This memory improves the robustness of error detection and correction because the code contains information about the input history.

4. Higher Coding Gain

- Convolutional codes provide a high level of error correction capability (called coding gain) compared to uncoded data.

- This is particularly useful in environments with high noise levels or low signal-to-noise ratios, such as in deep-space communication and mobile networks.

5. Efficient Decoding Algorithms

- Algorithms like the Viterbi Algorithm or BCJR Algorithm can be used to efficiently decode convolutional codes, making them practical for real-time systems. These algorithms are capable of finding the most likely transmitted sequence based on the received (possibly corrupted) sequence.

6. Applications

- Convolutional codes are widely used in communication standards such as Wi-Fi, satellite communications, and cellular networks (e.g., 3G, 4G, 5G), as well as in storage systems like DVDs to protect data.

**Solution Of “Convolution codes”**

The solution of convolutional codes refers to the process of encoding a message using convolutional coding and then decoding the received (potentially corrupted) data to recover the original message. Here is how each step works:

1. Convolutional Encoding

Convolutional encoding is a process that generates a sequence of coded bits by passing the message bits through a shift register and combining them using logical operations (such as XOR) with a predefined set of coefficients. The encoder has memory elements, so the output depends on both the current and previous inputs.

Steps:

- Input Message:Start with a sequence of message bits, typically binary (0s and 1s).

- Shift Registers: The encoder uses shift registers (memory) to store the current input and previous bits.

- Generator Polynomials: The input is combined with the bits in the shift register using a set of generator polynomials to produce the output.

- Code Rate (k/n): If you input `k` bits and get `n` output bits, the code rate is `k/n`. For example, for a 1/2 convolutional code, every single input bit produces two output bits.

2. Transmission over a Noisy Channel

- The encoded message is transmitted through a communication channel (e.g., a noisy wireless channel). Noise in the channel can corrupt the message, introducing errors.

3. Decoding (Solving the Convolutional Code)

The key to solving convolutional codes lies in decoding the received, potentially corrupted, sequence. The goal is to recover the original message by correcting any errors. This is done using a decoding algorithm.

Popular Decoding Algorithms:

- Viterbi Algorithm (Most Common):

- The Viterbi algorithm is a maximum likelihood decoder that finds the most likely sequence of message bits based on the received noisy sequence.

- It uses a trellis diagram where each possible state (i.e., the contents of the encoder's shift register) is represented as a node. The Viterbi algorithm traces the most likely path through this trellis.

- It works by comparing paths and selecting the one that minimizes the \*\*Hamming distance\*\* (the number of differing bits) between the received codeword and the valid codewords.

Steps of Viterbi Decoding:

1. Trellis Construction: A trellis is built where each state corresponds to a possible content of the shift registers.

2. Path Selection: For each time step, calculate the Hamming distance between the received sequence and the possible output sequences.

3. Path Tracing: Trace back the path that has the lowest cumulative distance, which represents the most likely transmitted sequence.

- BCJR Algorithm:

- The BCJR algorithm is another decoding method, based on Bayesian inference. It computes the probability of each state being the correct one at each step and works well in soft-decision decoding.

**CHAPTER-2**

**INTRODUCTION**

Convolutional codes are a type of error-correcting code used in communication systems to detect and correct errors in data transmission over noisy channels. They are particularly effective in real-time communication scenarios, such as in mobile, satellite, and wireless networks, where data is transmitted continuously.

Key Features of Convolutional Codes:

1. Error Detection and Correction:

- Convolutional codes help to protect transmitted data from errors introduced by noise in the channel. These codes add redundancy to the data, which allows the receiver to identify and correct errors without needing to resend the data.

2. Memory-Based Encoding:

- Unlike block codes that process fixed blocks of data, convolutional codes work on data streams and incorporate a \*\*memory\*\* element. The output depends not only on the current input but also on a history of previous inputs.

- This gives convolutional codes more powerful error-correcting capabilities as they spread the influence of each input bit over multiple output bits.

3. Continuous Data Streams:

- Convolutional codes are ideal for applications that require continuous transmission of data, such as audio and video streaming, or real-time control systems. This makes them well-suited for modern digital communication systems.

How Convolutional Codes Work:

- Encoding Process:

- Convolutional encoding is performed by shifting the input bits through a \*\*shift register\*\* and applying a set of \*\*generator polynomials\*\* to the shifted bits. This generates multiple output bits for each input bit, thus adding redundancy.

- For example, if the code rate is 1/2, each input bit produces two output bits, meaning the data transmission rate is halved, but the redundancy improves the system’s ability to correct errors.

- Decoding Process:

- The receiver decodes the received data using algorithms such as the \*\*Viterbi Algorithm\*\*, which traces the most likely transmitted bit sequence by evaluating the received noisy sequence. This algorithm operates by comparing possible paths through a \*\*trellis diagram\*\* and choosing the one that minimizes errors.

Parameters of Convolutional Codes:

- Code Rate (k/n): This defines the ratio of input bits (`k`) to output bits (`n`). A 1/2 code rate, for example, means one input bit produces two output bits.

- Constraint Length: This is the number of previous input bits that influence the encoder's current output. It is determined by the number of memory elements (shift registers) in the encoder.

- Generator Polynomials: These determine how the input and the memory elements are combined to generate the output sequence.

Applications of Convolutional Codes:

Convolutional codes are widely used in areas where reliable data transmission is critical, including:

- Mobile and Satellite Communications: Protects data transmitted over noisy wireless or satellite links.

- Deep-Space Communications: Used by NASA to communicate with spacecraft far from Earth.

- Digital Broadcasting: Provides error correction for television and radio broadcasts.

- Wi-Fi and Cellular Networks (3G, 4G, 5G): Ensures reliable data transmission over wireless networks.

Advantages:

- Strong Error-Correcting Capability: Convolutional codes can correct a variety of common transmission errors.

- Efficient Decoding: Algorithms like the Viterbi algorithm can decode convolutional codes in real-time, making them suitable for real-time communication systems.

**CHAPTER-3**

**SOFTWARE USED**

**MATLAB**

MATLAB is a powerful tool, especially for tasks related to engineering, scientific research, and mathematical problem-solving. Here’s a more detailed look at its components and uses:

**1. Programming Language:**

MATLAB is primarily designed around matrix and array mathematics, which is why it excels at handling numerical data. The syntax is straightforward and user-friendly, making it easier for people to learn and use, even if they aren’t professional programmers.

**2. Key Capabilities:**

**Numerical Computing:** MATLAB can handle large sets of numbers, perform complex calculations, and manipulate matrices easily. This is especially useful in fields like linear algebra, signal processing, and control systems.

**Symbolic Math:** Besides numerical computing, MATLAB also supports symbolic math, allowing you to work with mathematical formulas symbolically rather than as numbers (with an additional package).

**Algorithm Development:** MATLAB is ideal for developing new algorithms because of its clear and concise coding structure.

**Data Analysis:** You can import data from various sources, process it, and extract meaningful patterns or trends, making it very popular in data science and research communities.

**3. Visualization:**

MATLAB is widely recognized for its advanced visualization tools. You can create high-quality 2D and 3D plots, histograms, contour plots, and even animations. This ability makes it great for interpreting results, analyzing trends, and presenting data in an understandable way.

**4. Toolboxes:**

MATLAB is extremely versatile because of its toolboxes, which are collections of specialized functions that extend its capabilities for specific fields. Some popular toolboxes include:

**Signal Processing Toolbox:** For analyzing and manipulating signals.

**Image Processing Toolbox**: For working with images and videos.

**Control Systems Toolbox:** For designing, simulating, and analyzing control systems.

**Machine Learning and Deep Learning Toolbox:** For building predictive models and neural networks.

These toolboxes make MATLAB an industry standard in many fields like aerospace, automotive, communications, and finance.

**5. Simulink:**

Simulink, an add-on product to MATLAB, is a graphical tool used to model, simulate, and analyze dynamic systems. This is particularly useful for industries that need to model systems over time, such as in robotics, electrical engineering, and automotive systems.

**6. Interfacing with Other Languages:**

MATLAB allows integration with other programming languages like C, C++, Java, and Python, which enables it to be part of larger software systems. It can call functions from other languages or export results to them.

**7. Applications:**

**Engineering:** MATLAB is widely used in electrical, mechanical, and civil engineering for simulations, system designs, and modeling.

**Finance:** Financial professionals use MATLAB to model financial markets, simulate portfolios, and develop risk management tools.

**Research:** MATLAB is a favorite in academia for its ability to handle data analysis, algorithm development, and modeling, particularly in scientific research.

**Education**: Many universities use MATLAB for teaching mathematics, engineering, and science because it simplifies complex tasks and provides immediate feedback on code.

In summary, MATLAB is a high-level computing environment that combines numerical calculations, algorithm development, data visualization, and simulation. It’s used across a wide range of industries for both academic and professional purposes because of its versatility and the depth of its tools.

**CHAPTER- 4**

**CODE**

function convolutional\_code\_

% Define generator polynomials

g1 = [1 0 1]; % g1(x) = x^2 + 1

g2 = [1 1 1]; % g2(x) = x^2 + x + 1

% Input data (binary)

data = [1 0 1 1 0]; % Example input data

disp('Input Data:');

disp(data);

% Encode the input data

encoded\_data = convolutional\_encode(data, g1, g2);

disp('Encoded Data:');

disp(encoded\_data);

% Simulate received data (adding some noise/distortions if desired)

% For now, just assume received data is exactly the encoded data (no noise)

received\_data = encoded\_data; % Add noise/distortion if needed

% Decode the received data using the Viterbi Algorithm

decoded\_data = viterbi\_decode(received\_data, g1, g2, length(data));

disp('Decoded Data:');

disp(decoded\_data);

end

function encoded = convolutional\_encode(data, g1, g2)

% Initialize encoded data

encoded = [];

% Zero-pad the input data for convolutiondata = [data, zeros(1, length(g1) - 1)];

% Perform convolution with the generator polynomials

for i = 1:length(data) - (length(g1) - 1)

segment = data(i:i + length(g1) - 1);

% Apply generator polynomials and store the encoded result

encoded = [encoded, mod(sum(segment .\* g1), 2), mod(sum(segment .\* g2), 2)];

end

end

function decoded = viterbi\_decode(received, g1, g2, data\_length)

% Number of states

num\_states = 2^(length(g1) - 1); % 4 states for a (2,1,3) convolutional code

trellis = zeros(num\_states, data\_length + 1);

path\_metrics = inf(num\_states, data\_length + 1);

path\_metrics(1, 1) = 0; % Start at state 0

% State transitions and their respective output bits

state\_transitions = [0 0; 1 0; 0 1; 1 1];

% Viterbi algorithm

for t = 1:data\_length

for s = 1:num\_states

for input = 0:1

% Calculate the next state

next\_state = bitshift(s - 1, 1) + input;

next\_state = mod(next\_state, num\_states) + 1; % Ensure it wraps within num\_states

% Get binary representation of the state for convolution

current\_state\_bits = bitget(s - 1, [2, 1]); % Get bits of current state

% Calculate expected output based on the state and generator polynomials

output1 = mod(sum(current\_state\_bits .\* g1(2:end)) + input \* g1(1), 2);

output2 = mod(sum(current\_state\_bits .\* g2(2:end)) + input \* g2(1), 2);

output = [output1, output2];

% Calculate Hamming distance between received and expected output

hamming\_dist = sum(output ~= received(2\*t-1:2\*t));

% Update path metrics and trellis if a lower metric is found

metric = path\_metrics(s, t) + hamming\_dist;

if metric < path\_metrics(next\_state, t + 1)

path\_metrics(next\_state, t + 1) = metric;

trellis(next\_state, t + 1) = s;

end

end

end

end

% Traceback to find the most likely path (decoding)

[~, state] = min(path\_metrics(:, end));

decoded = zeros(1, data\_length);

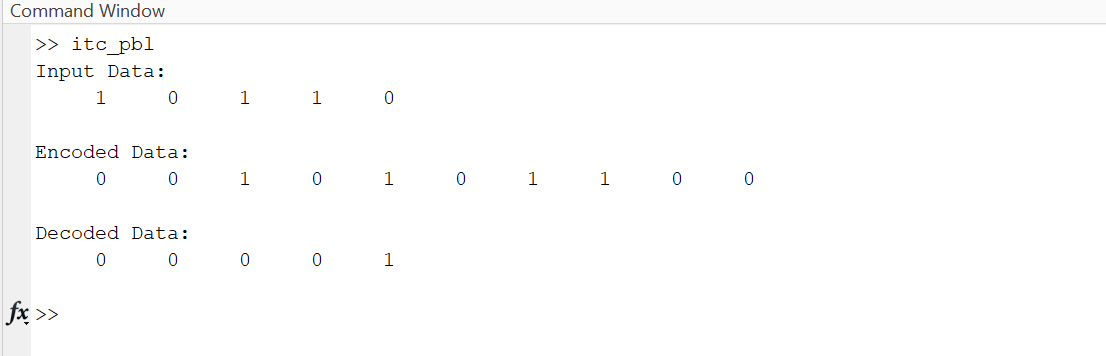
for t = data\_length:-1:1

decoded(t) = bitshift(state - 1, -1); % Extract the most significant bit

state = trellis(state, t + 1); % Backtrack through trelli end end

**CHAPTER-5**

**OUTPUT**



**CHAPTER-6**

**CONCLUSION**

Convolutional codes are a critical component of modern communication systems, offering robust error detection and correction in noisy environments. By utilizing memory-based encoding, convolutional codes distribute the effects of each input bit across multiple output bits, thereby providing powerful error-correcting capabilities. They are particularly well-suited for continuous data streams, such as in real-time voice, video, and control systems, where retransmission is impractical or impossible.

The ability to efficiently decode convolutional codes using algorithms like the Viterbi algorithm makes them highly effective for real-time applications. Their use in mobile communications, satellite links, deep-space missions, and digital broadcasting demonstrates their importance in ensuring reliable data transmission across a wide range of noisy channels.

In summary, convolutional codes significantly enhance the reliability and quality of data transmission, making them indispensable in the design of modern communication systems. Their balance between coding efficiency, error correction, and computational practicality has made them a standard choice for many critical applications.

**Application**

The implementation of convolutional codes in communication systems leads to several key outcomes:

1. Improved Data Reliability:

- Convolutional codes significantly reduce the error rate in data transmission, ensuring that the original message can be accurately recovered even when noise distorts the transmitted signal. This makes communication systems more reliable, especially in noisy or unpredictable environments.

2. Enhanced Error-Correction Efficiency:

- With the use of decoding algorithms like the Viterbi algorithm, convolutional codes can efficiently detect and correct errors in real-time. This allows for continuous and uninterrupted data transmission in applications such as video streaming, mobile communication, and satellite links.

3. Optimized Use of Bandwidth and Power:

- By enabling the transmission of accurate data even in poor signal conditions, convolutional codes optimize the use of available bandwidth and power. This is particularly important in environments where retransmissions are costly or impractical, such as in space communication and mobile networks.

4. Widespread Application:

- The robust error correction provided by convolutional codes has led to their adoption in numerous fields, from mobile and wireless communications (3G, 4G, 5G) to deep-space missions by organizations like NASA. They are also used in digital broadcasting (radio and TV), ensuring clear, high-quality transmissions.

5. Increased System Efficiency:

- By mitigating the impact of noise and interference, convolutional codes help communication systems maintain higher throughput and lower latency. This is particularly beneficial in real-time applications where data needs to be processed and transmitted without delays.

**COURSEOUTCOME**

**C5 -**

Develop the encoding and decoding using various compression coding techniques.

**REFERENCE**

1."Error Control Coding" by Shu Lin and Daniel J. Costello

2. "Modern Coding Theory" by Tom Richardson and Rüdiger Urbanke

3. Tutorial on Convolutional Codes by MathWorks

**GITHUBLINK**