EN2091 Laboratory Practice and Projects
Communication Systems and Networks - Workshop 2
Task Sheet

Index No.: 220735E **Group No.:** C-17

Date: 08/10/2024

Task 1.

Editor - D:\sem 3\Labs\week 5\N2\encode1.m

encode1.m

```
function [final_codeword] = encode1(data_input, divisor_input)
% encode1 takes a data_input and divisor_input as inputs
% encode1 outputs the CRC value final_codeword

data_len = length(data_input);
div_len = length(divisor_input) - 1;
data_input = bin2dec(data_input); % Convert binary string to decimal
divisor_input = bin2dec(divisor_input);
divisor_input = bitshift(divisor_input, data_len - div_len - 1); % Align divisor with data_input
divisor_input = bitshift(divisor_input, div_len); % Shift divisor to the appropriate degree
remainder = bitshift(data_input, div_len); % Shift data_input for division

% Perform division to calculate the remainder
for i = 1:data_len
    if bitget(remainder, data_len + div_len)
        remainder = bitxor(remainder, divisor_input); % XOR the remainder with the divisor
    end
    remainder = bitshift(remainder, 1); % Shift remainder to the left by 1
end

% Calculate the CRC check value
crc_value = bitshift(remainder, -data_len);
crc_value = dec2bin(crc_value); % Convert remainder back to binary
crc_value = pad(crc_value, div_len, 'left', '0'); % Pad the CRC value with leading zeros if necessary
data_input = dec2bin(data_input); % Convert data_input back to binary

% Append the CRC value to the original data to form the final codeword
final_codeword = append(data_input, crc_value);
end
```

Command Window

```
ans =

    '1010011110101'
```

Task 2.

D: ▸ sem 3 ▸ Labs ▸ week 5 ▸ N1

Editor - D:\sem 3\Labs\week 5\N1\decode1.m

encode1.m decode1.m

```
function [error_check] = decode1(encoded_word, divisor_input)
% decode1 takes an encoded_word and divisor_input as inputs
% and outputs if there is an error (syndrome) or not

word_len = length(encoded_word); % Length of the encoded word
div_len = length(divisor_input) - 1; % Degree of the divisor
encoded_word = bin2dec(encoded_word); % Convert encoded word to decimal
divisor_input = bin2dec(divisor_input); % Convert divisor to decimal
divisor_input = bitshift(divisor_input, word_len - div_len - 1); % Align divisor with encoded word
divisor_input = bitshift(divisor_input, div_len); % Shift divisor to the appropriate degree
remainder = bitshift(encoded_word, div_len); % Shift encoded word for division

% Perform division to check for errors
for i = 1:word_len
    if bitget(remainder, word_len + div_len)
        remainder = bitxor(remainder, divisor_input); % XOR remainder with divisor if necessary
    end
    remainder = bitshift(remainder, 1); % Shift remainder to the left by 1
end

% Calculate the remainder to check for errors
remainder = bitshift(remainder, -word_len); % Shift back the remainder
error_check = dec2bin(remainder); % Convert remainder to binary to output error status

end
```

Command Window

```
>> decode1('1010011110101','10111')

ans =

    '0'
```

Task 3.

```
p = 0.5; % Probability of bit flip in the binary symmetric channel (BSC)
encodedData = encode1('101001111', '10111'); % Encode the data using the given divisor

% Convert the encodedData (binary string) to a logical array
encodedDataLogical = encodedData - '0'; % This converts the character array to a logical array

% Transmit the encoded data through a binary symmetric channel
receivedData = bsc(encodedDataLogical, p);

% Convert the received data back to a binary string for decoding
receivedDataString = char(receivedData + '0'); % Convert logical array back to string

% Decode the received data and get the syndrome value
syndrome = decode1(receivedDataString, '10111');

% Display the syndrome to check if there is an error
disp('Syndrome (Error Check Value):');
disp(syndrome);
```

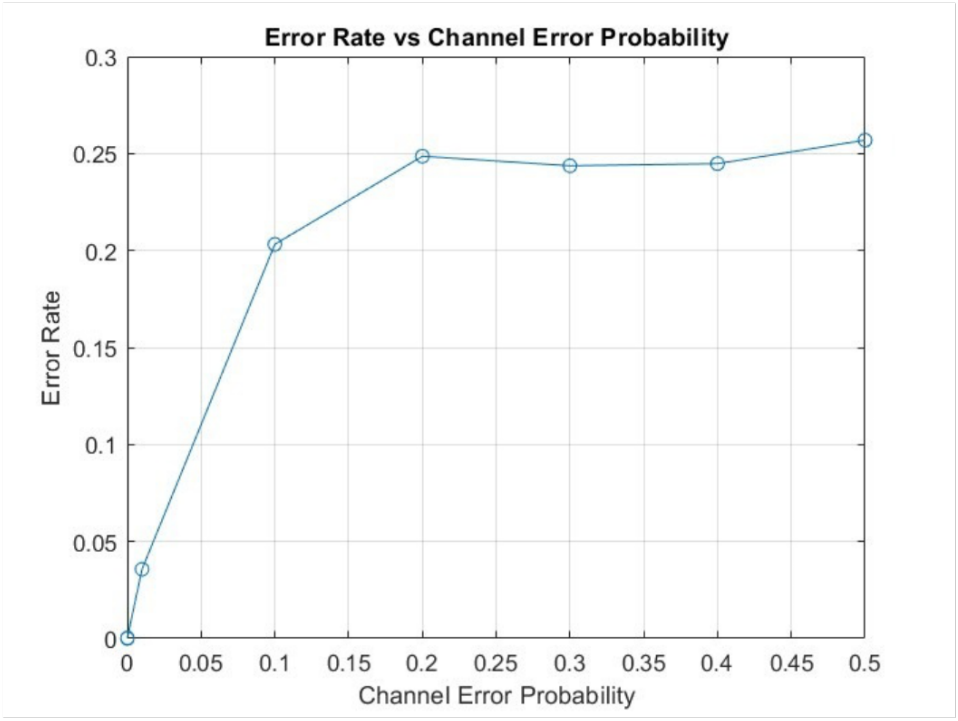
```
>> untitled3
Syndrome (Error Check Value):
1110
>> untitled3
Syndrome (Error Check Value):
10
>>
```

Syndrome is not equal to 0000 and it varies from time to time.

Task 4.

```
% Generate 10000 random data values
randomData = randi([0 2^9-1],1,10^4,"uint16");
errorRates = zeros(1,8);
probabilityOfError = [0.5 0.4 0.3 0.2 0.1 0.01 0.00 0.0001];
for m = 1:8
    errorCount = 0;
    for n = 1:10^4
        % Add zeros to make the binary representation 9 bits long
        binaryData = pad(num2str(dec2bin(randomData(n))),9,'left','0');
        % Call the encode function
        encodedWord = encode1(binaryData,'10111');
        codeWordArray = zeros(1,13);
        % Store the encoded bits in an array
        for p = 1:length(encodedWord)
            codeWordArray(p) = str2double(encodedWord(p));
        end
        % Pass the codeword through a binary symmetric channel
        modifiedCodewordArray = bsc(codeWordArray,probabilityOfError(m));
        modifiedCodeword = '';
        % Convert the modified codeword back to a string
        for q = 1:length(encodedWord)
            modifiedCodeword = append(modifiedCodeword, num2str(modifiedCodewordArray(q)));
        end
        % Decode the modified codeword
        decodedSyndrome = decode1(modifiedCodeword,'10111');
        % Check if an error occurred
        if decodedSyndrome ~= '0'
            errorCount = errorCount + 1;
        end
    end
    % Calculate and store the error rate
    newErrorRate = errorCount / 10000;
    errorRates(m) = newErrorRate;
end

% Plot error rate vs. channel error probability
figure; % Open a new figure window
plot(probabilityOfError, errorRates, '-o'); % Plot with line and markers
xlabel('Channel Error Probability'); % X-axis label
ylabel('Error Rate'); % Y-axis label
title('Error Rate vs Channel Error Probability'); % Plot title
grid on; % Enable grid
```


Error Rate vs Channel Error Probability
Error Rate
Channel Error Probability
0.3
0.25
0.2
0.15
0.1
0.05
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5