**EC-431** **Digital Communication**

**AMINA QADEER**

**MUHAMMAD SALMAN**

**Project**

Simulate a complete digital communication system with different modulation schemes in MATLAB for transmitting and receiving text messages.

### Basic Overview

In digital communication, information (such as text messages) is transmitted from one place to another through a series of steps: encoding (or modulation), transmission, and decoding (or demodulation).

### Steps in the Communication System

1. **Text to Binary Conversion**: The text message is first converted into binary data because communication systems transmit signals in binary form. Each character of the text is represented as a series of 1s and 0s.
2. **Modulation**:
   * **BPSK (Binary Phase Shift Keying)**: Each binary bit (1 or 0) is converted into a signal with a specific phase. For instance, '0' might be a signal phase of 0 degrees, and '1' might be a signal phase of 180 degrees.
   * **QAM (Quadrature Amplitude Modulation)**: Binary bits are grouped together into symbols (like pairs or quads of bits), and each symbol is represented by a unique combination of signal amplitude and phase. This allows more bits to be transmitted with each symbol. QPSK maps pairs of bits to one of four phases.
   * Adds noise and demodulates by calculating the angle of the noisy signal.

**16 QAM**:  Maps groups of 4 bits to one of 16 symbols.

* +  Adds noise and demodulates by using custom QAM functions to get back the symbols.

1. **Transmission Over a Channel**:
   * The modulated signal is sent over a communication channel (like radio waves, Fiber optics, etc.).
   * During this phase, the signal may be corrupted by noise. A common type of noise is **AWGN (Additive White Gaussian Noise)**, which is random noise that affects all frequencies of the signal.
2. **Demodulation**:
   * At the receiver end, the received signal (which may now include noise) undergoes demodulation.
   * **BPSK Demodulation**: The phases of the received signals are detected and converted back into binary bits.
   * **QAM Demodulation**: The amplitude and phase of the received symbols are detected and converted back into the group of binary bits that each symbol represents.
3. **Binary to Text Conversion**: Finally, the binary data is reassembled back into text, reconstructing the original message.

**CODE:**

% Prompt user for a text message

message = input('Enter a text message to transmit: ', 's');

% Modulation schemes

modulationSchemes = {'BPSK', 'QPSK', '16QAM'};

snr = 20; % Signal-to-Noise Ratio in dB

% Convert message to binary

binaryMessage = reshape(dec2bin(message, 8).' - '0', 1, []);

% Helper function to convert decimal symbols to binary arrays

function binaryArray = decimalToBinaryArray(decimalValues, numBits)

binaryArray = zeros(length(decimalValues), numBits);

for idx = 1:length(decimalValues)

binaryString = dec2bin(decimalValues(idx), numBits);

binaryArray(idx, :) = binaryString - '0'; % Convert char array to numeric

end

end

% Custom QAM Modulation Function

function modulatedSignal = customQAMMod(symbols, M)

k = sqrt(M);

realPart = 2\*mod(symbols, k) - k + 1;

imagPart = 2\*floor(symbols/k) - k + 1;

modulatedSignal = (realPart + 1i\*imagPart) / sqrt(2);

end

% Custom QAM Demodulation Function

function demodulatedSymbols = customQAMDemod(signal, M)

k = sqrt(M);

realPart = round(real(signal) \* sqrt(2) + k - 1);

imagPart = round(imag(signal) \* sqrt(2) + k - 1);

demodulatedSymbols = realPart + k \* imagPart;

demodulatedSymbols = min(max(demodulatedSymbols, 0), M-1);

end

for i = 1:length(modulationSchemes)

scheme = modulationSchemes{i};

fprintf('Simulating %s modulation\n', scheme);

switch scheme

case 'BPSK'

% BPSK Modulation

modulatedSignal = 2 \* binaryMessage - 1; % Map 0 to -1 and 1 to 1

% Add AWGN

noiseStdDev = sqrt(mean(abs(modulatedSignal).^2) / (10^(snr/10)));

noisySignal = modulatedSignal + noiseStdDev \* randn(size(modulatedSignal));

% BPSK Demodulation

demodulatedSignal = noisySignal > 0;

case 'QPSK'

% QPSK Modulation

reshapedBinaryMessage = reshape(binaryMessage, 2, []).';

binaryStrings = reshape(sprintf('%d', reshapedBinaryMessage.'), 2, []).';

symbols = arrayfun(@(x) bin2dec(binaryStrings(x,:)), 1:size(binaryStrings,1))';

modulatedSignal = exp(1j \* pi/4 \* (2\*symbols + 1));

% Add AWGN

noiseStdDev = sqrt(mean(abs(modulatedSignal).^2) / (10^(snr/10)));

noisySignal = modulatedSignal + noiseStdDev \* (randn(size(modulatedSignal)) + 1j \* randn(size(modulatedSignal)));

% QPSK Demodulation

demodulatedSymbols = round((angle(noisySignal ./ abs(noisySignal)) / (pi/4) - 1) / 2);

demodulatedSymbols = mod(demodulatedSymbols, 4);

demodulatedBinary = decimalToBinaryArray(demodulatedSymbols, 2);

demodulatedSignal = reshape(demodulatedBinary.', 1, []);

case '16QAM'

% 16QAM Modulation

M = 16;

paddedBinaryMessage = [binaryMessage, zeros(1, mod(-length(binaryMessage), log2(M)))];

reshapedBinaryMessage = reshape(paddedBinaryMessage, log2(M), []).';

binaryStrings = reshape(sprintf('%d', reshapedBinaryMessage.'), log2(M), []).';

symbols = arrayfun(@(x) bin2dec(binaryStrings(x,:)), 1:size(binaryStrings,1))';

modulatedSignal = customQAMMod(symbols, M);

% Add AWGN

noiseStdDev = sqrt(mean(abs(modulatedSignal).^2) / (10^(snr/10)));

noisySignal = modulatedSignal + noiseStdDev \* (randn(size(modulatedSignal)) + 1j \* randn(size(modulatedSignal)));

% 16QAM Demodulation

demodulatedSymbols = customQAMDemod(noisySignal, M);

demodulatedBinary = decimalToBinaryArray(demodulatedSymbols, log2(M));

demodulatedSignal = reshape(demodulatedBinary.', 1, []);

% Remove padding

demodulatedSignal = demodulatedSignal(1:length(binaryMessage));

end

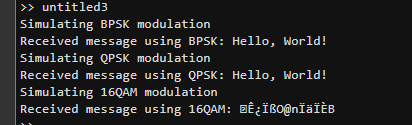
% Convert binary to text

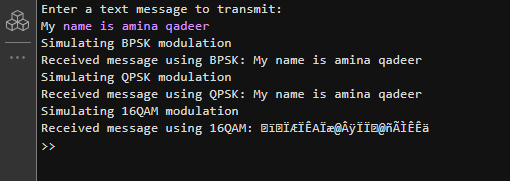
receivedMessage = char(bin2dec(reshape(sprintf('%d', demodulatedSignal), 8, []).'))';

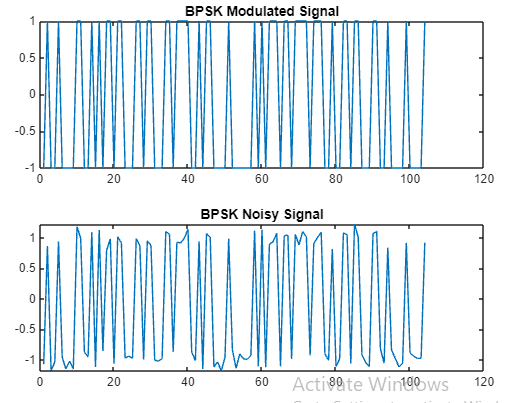
fprintf('Received message using %s: %s\n', scheme, receivedMessage);

end

**Output the Results:**

****

****

****

**A graph of a function

Description automatically generated with medium confidence**

**A screen shot of a graph

Description automatically generated**