Data Generation: Modulation Classification

EE18BTECH11014

Krishna Srikar Durbha

Signal and Channel Parameters

```
% No.of Samples
N = 10000;

% SNR(in dBW)
SNR = [5,10,15,20,25,30];

% Channel Length for Rayleigh Fading
L = [2,3];

% Modulation Schemes
modulationTypes = categorical(["QPSK", "16-QAM", "64-QAM"]);

% Channels
channelTypes = categorical(["AWGN", "Rayleigh"]);

% Normalise Data
Norm = false;

% Ignoring Warnings
warning('off','all');
```

Data Generation

Generating Signal

```
numModulationTypes = length(modulationTypes);
numChannelTypes = length(channelTypes);
for i = 1:numChannelTypes
    for j = 1:numModulationTypes
        if channelTypes(i) == "Rayleigh"
            DataGeneration(N*50,modulationTypes(j),channelTypes(i),SNR,L,Norm)
        elseif channelTypes(i) == "AWGN"
            DataGeneration(N,modulationTypes(j),channelTypes(i),SNR,L,Norm)
        end
        fprintf('-------------------------\n');
    end
end
```

```
Saved AWGN QPSK Data
-------------------------
Saved AWGN 16-QAM Data
-------------------------
Saved AWGN 64-QAM Data
-------------------------
Saved Rayleigh QPSK Data
-------------------------
Saved Rayleigh 16-QAM Data
```

```
-------------------------
Saved Rayleigh 64-QAM Data
-------------------------
```

Functions

Transmit Power:

Transmit-Power (SignalPower) = $\frac{\sum_i^M ||s_i||^2}{M}$

The above is Transmit Power of Signal where no.of Symbols are "M".

For Normalising Symbols, we divide each Symbol by $\sqrt{SignalPower}$.

SNR Ratio:

$$SNR = \frac{P}{N\sigma^2}$$

where P: Transmit Power, "N" is Dimensional Space and and σ^2: Variance of Gaussian Channel

Bit Error Rate Calculation for a Signal

```
function DataGeneration(N,Modulation,Channel,SNR,L,Normalise)

    S = size(SNR,2);
    C = size(L,2);

    % For AWGN Channel
    if Channel == "AWGN"
        % File Path
        dataDirectory = fullfile("../Data/" + string(Channel) + "/" + string(Modulatior
        mkdir(dataDirectory);

        for i = 1:S
            % Generating Modulated Data
            [tx,txModulated,SignalPower] = TransmissionData(N,Modulation,Normalise);

            snr = SNR(i);
            snrW = 10^(snr/10);
            % disp("SNR(in dB):"  + string(SNR(i)) + " SNR:" + string(snr))

            % Transmission: Adding Noise
            rx = txModulated + (randn(N,1)+1i*randn(N,1))*sqrt(1/2)*sqrt(SignalPower/sr
            % disp(string(Channel) + " " + string(Modulation) + " " + string(snr))

            % Saving File
            fileName = fullfile(dataDirectory,sprintf("%sdB-SNR",string(snr)));
            save(fileName,"tx","txModulated","rx","snr");
        end
    end


    % For Rayleigh Channel
    if Channel == "Rayleigh"
        % Channel Lengths
        for l = 1:C
            % ChannelModel: Modelling
            ChannelLength = L(l);
            if ChannelLength == 2
                ChannelCoeff = [sqrt(0.8), sqrt(0.2)];
                ChannelModel = ChannelCoeff.*(randn(1,ChannelLength)+1i*randn(1,Channel
            elseif ChannelLength == 3
                ChannelCoeff = [sqrt(0.75), sqrt(0.2), sqrt(0.05)];
                ChannelModel = ChannelCoeff.*(randn(1,ChannelLength)+1i*randn(1,Channel
            end


            % File Path
            dataDirectory = fullfile("../Data/" + string(Channel) + "/" + string(Channe
            mkdir(dataDirectory);

            for i = 1:S
                % Generating Modulated Data
                [tx,txModulated,SignalPower] = TransmissionData(N,Modulation,Normalise

                snr = SNR(i);
                snrW = 10^(snr/10);
                % disp("SNR(in dB):"  + string(SNR(i)) + " SNR:" + string(snr))

                % Transmisssion: Fading Effect of Channel and Adding Noise
                rx = conv(txModulated,ChannelModel,'same') + (randn(N,1)+1i*randn(N,1)
                % disp(string(Channel) + " " + string(Modulation) + " " + string(snr))

                % Saving File
                fileName = fullfile(dataDirectory,sprintf("%sdB-SNR",string(snr)));
                save(fileName,"tx","txModulated","rx","snr","ChannelLength");
            end
        end
    end
    disp("Saved " + string(Channel) + " " + string(Modulation) + " Data")
end
```

Transmission Data Generation

```
function [tx,txModulated,SignalPower] = TransmissionData(N,Modulation,Normalise)
    % Generating Transmitter Signal
    if Modulation == "QPSK"
        tx = randi([0 3], N, 1);
    elseif Modulation == "16-QAM"
        tx = randi([0 15], N, 1);
```

```matlab
    elseif Modulation == "64-QAM"
        tx = randi([0 63], N, 1);
    end

    % Modulation: Modulating Data and Scatter Plotting it
    if Modulation == "QPSK"
        qpskmod = comm.QPSKModulator();
        txModulated = qpskmod(tx);
        %scatterplot(txModulated);
        %grid on;
    elseif Modulation == "16-QAM"
        txModulated = qammod(tx,16);
        %scatterplot(txModulated);
        %grid on;
    elseif Modulation == "64-QAM"
        txModulated = qammod(tx,64);
        %scatterplot(txModulated);
        %grid on;
    end

    % Normalising Data
    % This makes Power of Signal = 1
    if Normalise == true
        SignalPower = sqrt(mean(abs(txModulated).^2));
        txModulated = txModulated/SignalPower;
    end

    SignalPower = mean(abs(txModulated).^2);
    % disp("Power of Signal = " + string(SignalPower));
end
```