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;

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

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

Data Generation

Generating Signal

```
numModulationTypes = length(modulationTypes);
numChannelTypes = length(channelTypes);
for i = 1:numChannelTypes
    for j = 1:numModulationTypes
        if channelTypes(i) == "Rayleigh"
            DataGeneration(N*100,modulationTypes(j),channelTypes(i),SNR,L)
        elseif channelTypes(i) == "AWGN"
            DataGeneration(N,modulationTypes(j),channelTypes(i),SNR,L)
        end
    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

Bit Error Rate Calculation for a Signal

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

```
    if Modulation == "QPSK"
        tx = randi([0 3], N, 1);
    elseif Modulation == "16-QAM"
        tx = randi([0 15], N, 1);
    elseif Modulation == "64-QAM"
        tx = randi([0 63], N, 1);
    end

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

    % Modulation: Modulating Data and Scatter Plotting it
    if Modulation == "QPSK"
        qpskmod = comm.QPSKModulator("PhaseOffset",0);
        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
    NormCoeff = sqrt(mean(abs(txModulated).^2));
    txModulated = txModulated/NormCoeff;


    % Transmission: Transmission of Data through Channel and
    % Decoding: Decoding the Received Data

    % Channel Coefficients
    % This makes Power of Signal = 1
    if L == 2
        ChannelCoeff = [sqrt(0.8), sqrt(0.2)];
    elseif L == 3
        ChannelCoeff = [sqrt(0.75), sqrt(0.2), sqrt(0.05)];
    end

    S = size(SNR,2);

    for i = 1:S
        snr = 10^(SNR(i)/10);
        % disp("SNR(in dB):"  + string(SNR(i)) + " SNR:" + string(snr))
        % Fading and Noise
        if Channel == "Rayleigh"
            % disp(string(Channel) + " " + string(Modulation) + " " + string(snr))
            rx = RayleighFading(txModulated,ChannelCoeff) + (randn(N,1)+1i*randn(N,1))/
        elseif Channel == "AWGN"
            % disp(string(Channel) + " " + string(Modulation) + " " + string(snr))
            rx = txModulated + (randn(N,1)+1i*randn(N,1))/sqrt(2*snr);
        end

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

Rayleigh Fading

```
function rx = RayleighFading(tx,ChannelCoeff)
    ChannelLength = size(ChannelCoeff,2);
    Channel = ChannelCoeff.*(randn(1,ChannelLength)+1i*randn(1,ChannelLength)/sqrt(2));
    rx = sum(tx.*Channel,2);
end
```