Data Generation for Modulation Classification

Reference: https://in.mathworks.com/help/deeplearning/ug/modulation-classification-with-deep-learning.html

Modulation Schemes:

1. QPSK
2. 16-QAM
3. 64-QAM

Channels:

1. AWGN
2. Multi-Path Fading
3. ARMA

Training Data is generated with a SNR ratio of 30dB.

Testing Data is generated with a SNR ratio of [5,10,15,20,25] dB.

Modulation and Channel Schemes

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

% Channels
channelTypes = categorical(["Rician"]);

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

Waveform Specifications

```
numFramesPerModType = 1000;  % No.of frames per Modulation Scheme
sps = 8;                    % Samples per symbol
spf = 1024;                 % Samples per frame
symbolsPerFrame = spf / sps;
fs = 200e3;                 % Sample rate
fc = [902e6 100e6];         % Center frequencies
transDelay = 50;

SNRs = [5,10,15,20,25,30];  % SNR Ratio

channel = helperModClassTestChannel(...
  'SampleRate', fs, ...
  'SNR', 30, ...
  'PathDelays', [0 1.8 3.4] / fs, ...
  'AveragePathGains', [0 -2 -10], ...
  'KFactor', 4, ...
  'MaximumDopplerShift', 4, ...
  'MaximumClockOffset', 5, ...
  'CenterFrequency', 902e6);
```

Data-Generation

```
numModulationTypes = length(modulationTypes);
numChannelTypes = length(channelTypes);
for i = 1:numChannelTypes
    for j = 1:numModulationTypes
        if channelTypes(i) == "Rician"
            DataGeneration(modulationTypes(j),channelTypes(i),SNRs,numFramesPerModType,
        end
        fprintf('-------------------------\n');
    end
end
```

```
Saved Rician QPSK Data
-------------------------
Saved Rician 16QAM Data
-------------------------
Saved Rician 64QAM Data
-------------------------
```

Data Generation for Multi-Path Fading Channel

```
function DataGeneration(Modulation,Channel,SNRs,numFramesPerModType,sps,spf,fs,transDel
    if Channel == "Rician"
        % File Path
        dataDirectory = fullfile("../Data/" + string(Channel) + "/" + string(Modulation
        mkdir(dataDirectory);

        dataSrc = helperModClassGetSource(Modulation, sps, 2*spf, fs);
        modulator = helperModClassGetModulator(Modulation, sps, fs);

        if contains(char(Modulation), {'B-FM','DSB-AM','SSB-AM'})
            % Analog modulation types use a center frequency of 100 MHz
            channel.CenterFrequency = 100e6;
        else
            % Digital modulation types use a center frequency of 902 MHz
            channel.CenterFrequency = 902e6;
        end

        S = size(SNRs,2);
        for i = 1:S
            snr = SNRs(i);
            channel.SNR = snr;
            rx = [];

            for p = 1:numFramesPerModType
                % Generate Random Data
                x = dataSrc();

                % Modulate
                y = modulator(x);

                % Pass through independent channels
```

```
                 rxSamples = channel(y);

                 % Remove transients from the beginning, trim to size, and normalize
                 rx_perFrame = helperModClassFrameGenerator(rxSamples, spf, spf, transDe
                 rx_frameComponents = [real(rx_perFrame),imag(rx_perFrame)];
                 rx = [rx,rx_frameComponents];
             end

             % Saving File
             rx = reshape(rx,[spf,2,numFramesPerModType]);
             rx = permute(rx,[3,1,2]);
             rx = reshape(rx,[numFramesPerModType,1,spf,2]);
             fileName = fullfile(dataDirectory,sprintf("%sdB-SNR",string(snr)));
             save(fileName,"rx","snr");
         end
     end
     disp("Saved " + string(Channel) + " " + string(Modulation) + " Data")
 end
```