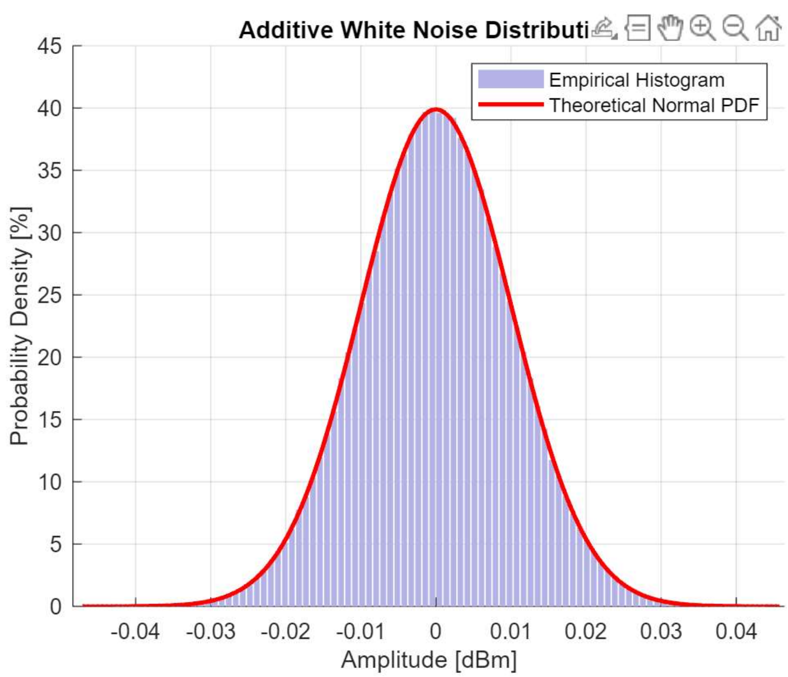
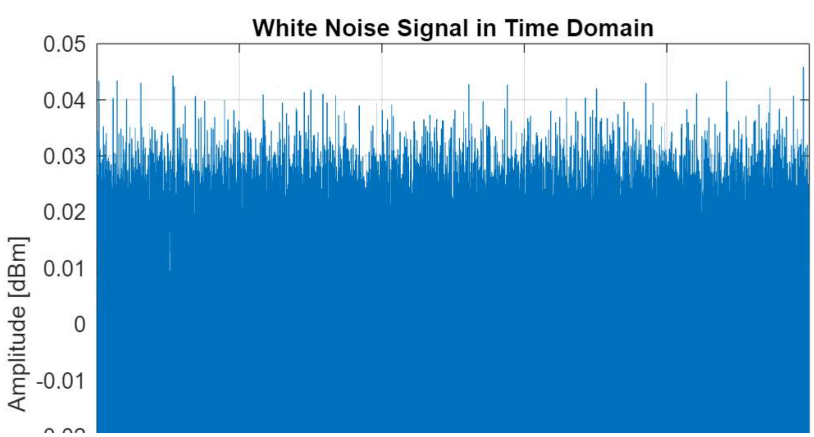
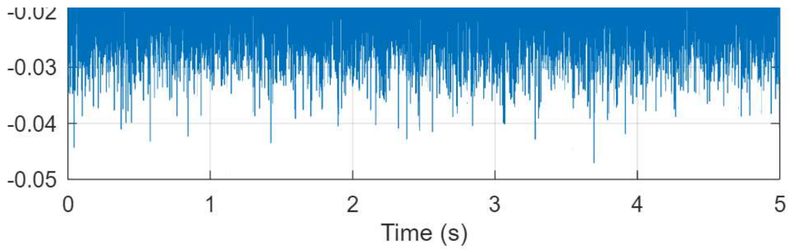
```
% This script is meant to provide a simulation of generating a white noise
% signal that can be added to the recieved signal at the input of the
% demodulator

% Parameters
fs = 100000;          % Sampling frequency in Hz
duration = 5;       % Duration in seconds
noise_std = 0.01;   % Standard deviation of the noise
noise_mean = 0;     % Mean of the noise
nbins = 100;         % Number of histogram bins

% Generate white noise
white_noise = generate_white_noise(fs, duration, noise_std, noise_mean);

% Define x range for PDF
x_range = linspace(min(white_noise), max(white_noise), 1000);

% Compute the probability density function (PDF) of the normal distribution
theoretical_pdf = (1 / (noise_std * sqrt(2*pi))) * exp(-0.5 * ((x_range - noise_mean) / noise_std).^2);

% Plot the empirical and theoretical PDF
plot_pdf(white_noise, theoretical_pdf, x_range, nbins);
```




```

% Plot the white noise signal in the time domain
plot_time_domain(white_noise, fs)
```






```
function white_noise = generate_white_noise(fs, duration, std, mean)
    % This function generates an additive white noise signal with a given
    % duration in seconds and sampling rate. The mean and std can be specified
    % in the input parameters.

    % Calculate the total number of samples
    num_samples = round(fs * duration);

    % Generate white noise with specified mean and standard deviation
    white_noise = std * randn(1, num_samples) + mean;
end

%-----------------------------------------------------------------------------------------------

function plot_pdf(empirical_data, theoretical_data, x_range, num_bins)
    % This function plots an empirical data set for a pdf versus its
    % theoretical pdf

    % Histogram of the data
    [counts, edges] = histcounts(empirical_data, num_bins, 'Normalization', 'pdf');
    bin_centers = (edges(1:end-1) + edges(2:end)) / 2;

    % Plot
    figure;
    hold on;
    bar(bin_centers, counts, 'FaceColor', [0.7 0.7 0.9], 'EdgeColor', 'none'); % Histogram
    plot(x_range, theoretical_data, 'r', 'LineWidth', 2); % Theoretical PDF
    xlabel('Amplitude [dBm]');
    ylabel('Probability Density [%]');
    title('Additive White Noise Distribution');
    legend('Empirical Histogram', 'Theoretical Normal PDF');
    grid on;
    hold off;
end

%-----------------------------------------------------------------------------------------------

function plot_time_domain(signal, fs)
    % This function plots a signal in the time-domain

    % define time axis
    t = linspace(0, length(signal)/fs, length(signal));

    figure;
    plot(t, signal);
    xlabel('Time (s)');
    ylabel('Amplitude [dBm]');
    title('White Noise Signal in Time Domain');
    grid on;
end

%-----------------------------------------------------------------------------------------------
```