ENEL 529 - Lab 3

Kyle Derby MacInnis - 10053959

November 3, 2014

Abstract

The purpose of this lab is to simulate a Rayleigh fading channel using the Inverse Discrete Fourier Transform method, and upon simulating the channel, the lab will focus on calculating the outage probability of the Rayleigh faded signal.

Exercise 1: Simulating the Rayleigh Fading Channel using IDFT

Exercise Objective

The objective of this exercise will be to simulate a Rayleigh Fading Channel using the Inverse Discrete Fourier Transform method, and then upon simulating the received signal, calculate the simulated and theoretical outage probabilities associated with Rayleigh Faded Signals.

Overview

The IDFT method works under the principle that the signal is made up a many NLOS transmissions. Due to the Central Limit Theorem, these individual components can be described in terms of Gaussian Random variables split into two groups: *in-phase* and *quadrature*. In the frequency domain, these two i.i.d Gaussian random variables are multiplied with the Magnitude response of the system, then sent through the IDFT and scaled by a correction factor. Once scaled, the received signal is the square root of the sum of the squares of the *in-phase* and *quadrature* components. This will cover steps 1-10 of the following procedure.

Procedure

1. Setup the System Variables

```
%% Step 1 - Setup System Variables
% Operating Frequency - fc = 900 MHz
fc = 900 * 10^6;
% Sample Interval - Ts = 1 msec
Ts = 1 * 10^(-3);
% Vehicle Speed - v = 90Km/hr (25 m/s)
v = 90 * (1000/3600);
% Variance of Gaussian Variables X and Y
sigma2 = 1.0;
% Speed of Light
c = 3.0 * 10^8;
```

2. Calculate the Normalized Maximum Doppler Frequency

```
%% Step 2 - Calculate Normalized Doppler Frequency
% Maximum Dopple Frequency - fd_max
fd_max = v*fc/c;
% Calculate Normalized Doppler Frequency - fd_norm
fd_norm = fd_max * Ts;
```

3. Calculate the # of Points Affected by the Doppler Magnitude Spectrum

```
%% Step 3 - Calculate # of points, where Shaping Filter = 0
% Assume # of Points in Time Domain, M = 1000
M = 1000;
% Vector Index - m
m = 1:M;
% Find # of point, k_m
k_m = floor(fd_norm*M);
```

4. Calculate the Sampling Interval of the Frequency Domain

```
%% Step 4 - Calculate Sampling Interval in Frequency
% Sampling Frequency - delta_f
delta_f = 1/(M*Ts);
```

5. Generate two Gaussian Random Variable Vectors, **X** and **Y**

```
%% Step 5 - Generate 2 Vectors X and Y
% Random Variable X
X = sqrt(sigma2)*randn(1, M);
% Random Variable Y
Y = sqrt(sigma2)*randn(1, M);
```

6. Generate the Magnitude Reponse Function of the Doppler Fading Filter. The following function was added in a separate file to define the Magnitude Response:

```
%% For Step 6
% Create Piecewise Magnitude Response - F(m)
function F = F(m, km, fdmax, Ts_, M_)
    if ( m == 1 )
        F = 0;
    elseif (( m >= 2 ) && ( m <= km ))
        F = (sqrt(0.5/(sqrt(1 - ((m-1)/(M_*Ts_*fdmax))^2))));
    elseif ( m == (km + 1) )
        F = (sqrt((km/2)*((pi/2) - atan((km-1)/(sqrt(2*km -1))))));
    elseif (( m >= (km + 2) ) && ( m <= (M_ - km) ))
        F = 0;
    elseif ( m == (M_ - km + 1) )
        F = (sqrt((km/2)*((pi/2) - atan((km -1)/(sqrt(2*km -1))))));
    elseif (( m >= (M_ - km + 2) ) && ( m <= M_ ))
        F = (sqrt(0.5/(sqrt(1 - ((M_ - (m-1))/(M_*Ts_*fdmax))^2))));
    else
        F = NaN;
    end
end
```

Following the definition of the piecewise function, the response was calculated in the main file using the following:

```
%% Step 6 - Generate the Magnitude Response of Shaping Filter
% Define Empty Vector for Values
Fm = zeros(M,1);
% Generate Filter
for i=1:M
    Fm(i,1) = F(i, k_m, fd_max, Ts, M);
end
```

7. Calculate the Output of the Shaping Filter

```
%% Step 7 - Calculate the Output of the Shaping Filter
% Generate Empty Vector
filt_out = zeros(M,1);
% Calculate Output
for i=1:M
    filt_out(i,1) = X(1,i)*F(i, k_m, fd_max, Ts, M) - j*Y(1,i)*F(i, k_m, fd_max, Ts, M)
    ;
end
```

8. Calculate the IDFT of the Output

```
%% Step 8 - Take Invesre Discrete Fourier Transform of Output
% Calculate IDFT of filt_out
inv_filt_out = ifft(filt_out, M);
```

9. Scale the IDFT result by Correction Factor, and Calculate *in-phase* and *quadrature* power components

```
%% Step 9 - Multiply IDFT by Scaling Factor
% Initialize sigma2_2 to 0
sigma2_2 = 0;
% Calculate sigma2_2
for i = 1:M
    sigma2_2 = sigma2_2 + (Fm(i,1)/M)^2;
end
sigma2_2 = sigma2 * sigma2_2;
% Scaling Factor - SF
SF = 1/(sigma2_2*sqrt(2));
% Multiply Output by Scaling Factor
scaled_inv_filt_out = SF * inv_filt_out;
% Take Quadrature and In-Phase Components of Scaled Signal
Zt = real(scaled_inv_filt_out);
Zq = imag(scaled_inv_filt_out);
% Calculate Power of Zt and Zq
Pt = 0;
Pq = 0;
for i = 1:M
    Pt = Pt + Zt(i)^2;
    Pq = Pq + Zq(i)^2;
end
```

10. Calculate the resulting Signal Envelope of the Received Signal

```
%% Step 10 - Calculate the Received Signal Envelope
% Create Empty Signal Envelope vector, R
R = zeros(M,1);
% Calculate the Envelope
for i=1:M
    R(i,1) = sqrt(Zt(i,1)^2 + Zq(i,1)^2);
end
% Plot the Received Signal Envelope
figure(3);
plot(m, R(:,1), 'r-');
title('Received Signal Envelope vs Time');
xlabel('Time: (msec)');
ylabel('Envelope');
```

The following graph was plotted as a result:

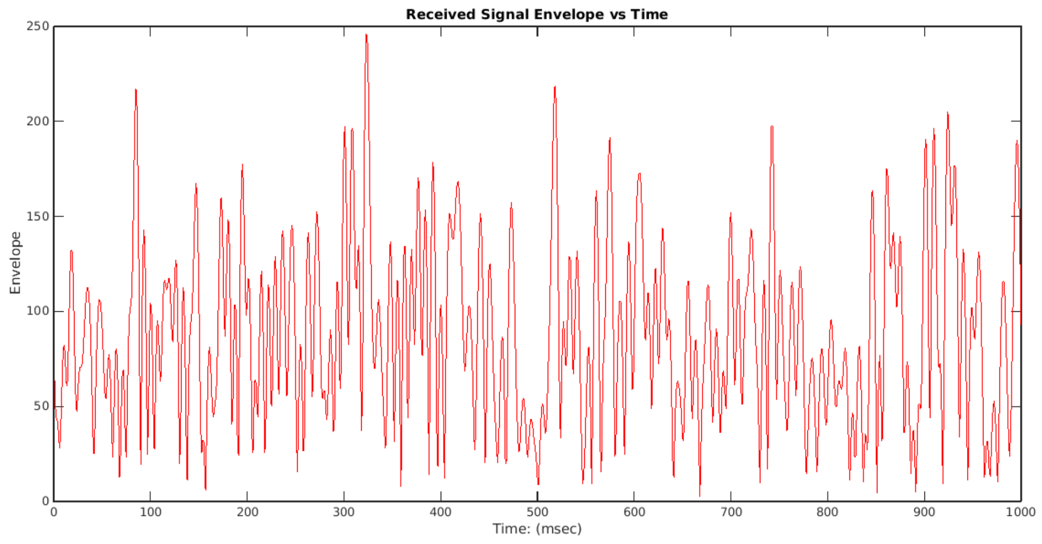

Figure 1: The graph of the Received Signal Envelope plotted against time.

11. Calculate the Average Power, $P_{av,k}$ of the Received Signal

```
%% Step 11 - Calculate the Average Power of the Received Signal
% Average Power, P_avk
P_av(k) = (Pt + Pq)/M;
```

12. Calculate the Outage Probability assuming $P_{th,k} = P_{av,k}$ by counting the number of times the signal strength falls below it

```
%% Step 12 - Calculate Outage Probability
% Set Threshold to Average Power of Iteration
P_th = P_av(k);
% Calculate Number of Outages
for i = 1:M
    if ( (Zt(i)^2 + Zq(i)^2) < P_th )
        N_out(k) = N_out(k) + 1;
    end
end
% Calculate Outage Probability
P_out(k) = N_out(k)/M;
```

13. Repeat steps 5-12 for a total of 50 runs and Calculate the Average Power over the runs. The following code was added to the previously mentioned code before and after steps 5-12:

```

%% ..... STEPS 1-4 .....

%% For Step 5-12: Loop K Times
K = 50;
% Generate Average Power Vector
P_av = zeros(K,1);
% Generate Outage Counter Vector
N_out = zeros(K,1);
```

```
% Generate Outage Probability Vector
P_out = zeros(K,1);

% Loop K Iterations
for k=1:K

%% .....................

%% ..... STEPS 5-12 .....

%% .....................

%% End Iteration Loop After 50 Times
end

%% Step 13 - After Loop, Calculate Average Power over K iterations
% Initialize Total Average Power
Pav = 0;
% Calculate Total Average Power
for k = 1:K
    Pav = Pav + P_av(k);
end
Pav = Pav/K;
```

14. Calculate the Average Outage Probability over all of the simulated runs

```
%% Step 14 - Calculate the Average Simulated Outage Probability
% Initialize Average Probability Outage
Pout = 0;
% Calculate Average Probability Outage
for k=1:K
    Pout = Pout + P_out(k);
end
Pout = Pout/K;
```

Conclusions

The main exercise of the lab was completed without much difficulty, however there was a bit of an odd error relating to the scale of the received signal envelope. It appears to be orders of magnitude higher than expected. The cause of this could not be found.

Additional Questions and Exercises

Task 1.1 - Plot Doppler Spectrum vs. Frequency and RMS Normalized Signal Envelope

Part a) - Doppler Spectrum vs Frequency

The following code was used to plot the graph of the Shaping Filter versus the frequency:

```
%% 1.1 - a) Plot Doppler Spectrum vs Frequency
% Calculate Frequency Change and Spectrum
fm = zeros(M,1);
F_m = zeros(M,1);
for m = 1:M
    fm(m,1) = m*delta_f;
    F_m(m,1) = F(fm(m,1), k_m, fd_max, Ts, M);
end
% Plot Magnitude Spectrum vs Frequency Fm
```

```
figure(1);
plot(fm, F_m, 'r-');
title('Doppler Magnitude Spectrum vs Frequency');
xlabel('Frequency: (Hz)');
ylabel('Magnitude Spectrum: F(m)');
```

The following graph was plotted as shown:

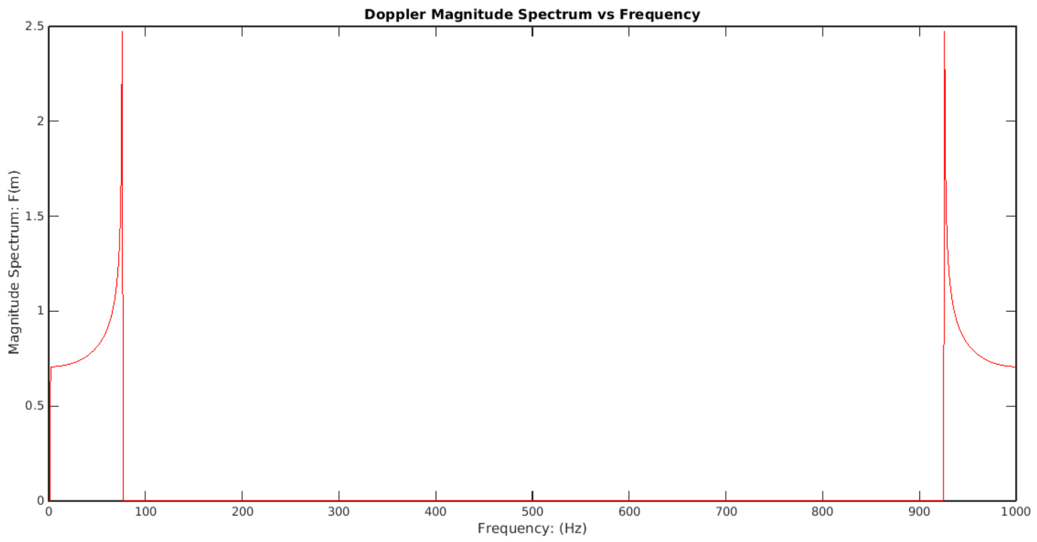

Figure 2: The graph of the Doppler Shaping Filter Magnitude Response against frequency.

Part b) - RMS Normalized Signal Envelope

The Following code was used to plot the graph of the RMS normalized Signal Envelope:

```
%% 1.1 - b) Plot RMS Normalized Signal Envelope
% Calculate RMS Value of Signal Envelope
R_rms = 0;
for i=1:M
    R_rms = R_rms + R(i,1)^2;
end
R_rms = sqrt(R_rms/M);
% Calculate the RMS Normalized Signal Envelope in dB
R_rms_dB = zeros(M,1);
for i=1:M
    R_rms_dB(i,1) = 20*log10(R(i,1)/R_rms);
end
% Calculate Time, t_m
t_m = zeros(M,1);
for i = 1:M
    t_m(i,1) = i*Ts;
end
% Plot RMS Normalized Signal vs Time
figure(2);
plot(t_m, R_rms_dB, 'r-');
title('RMS Normalized Signal Envelope vs Time');
xlabel('Time: (msec)');
ylabel('Signal Envelope Magnitude: (dB)');
```

The following Graph was plotted as a result:

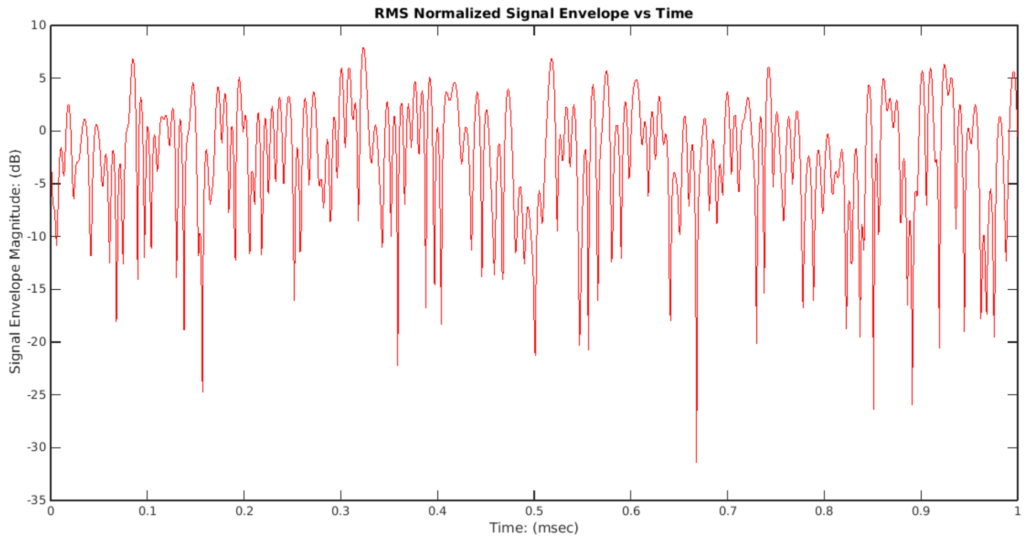

Figure 3: The Graph of the RMS Normalized Signal Envelope.

Task 1.2 - What is the Simulated Average Power?

The following code was added to the file to calculate the Simulated Average Power:

```
%% 1.2 − What is the Simulated Average Power, Pav?
% Display Average Power of Simulation
display('This Simulated Average Power after 50 Iterations is:');
display(Pav);
```

The calculated result following the run was found to be:

$$P_{av} = 8.2793 * 10^3$$

Task 1.3 - What is the Average Simulated Outage Probability over 50 runs?

The following code was added after the loop:

```
%% 1.3 − What is the Average Simulated Outage Probability, Pout?
% Display Average Outage Probability
display('The average simulated outage probability after 50 iterations is:');
display(Pout);
```

The calculated result following a successful run of the file was found to be:

$$P_{out} = 0.6316$$

Task 1.4 - What is the Theoretical Outage Probability?

The following was added to file to calculate the theoretical outage probability:

```
%% 1.4 - What is the Theoretical Outage Probability?
% Calculate Pout (Theoretical)
Pout_theory = 1 - exp(-(Pav)/(Pav));
% Display Theoretical Outage Probability
display('The theoretical outage probability is:');
display(Pout_theory);
```

The result following the run was found to be:

$$P_{out,theory} = 0.6321$$

Task 1.5 - Compare the Simulated and Theoretical Outage Probabilities.

The following was added to compare the simulated and theoretical outage probabilities:

```
%% 1.5 - Compare the Simulated and Theoretical
display('The difference between the Simulated and Theoretical Outage Probabilities are:');
display(Pout_theory-Pout);
display('The % differece between the two is:');
display((Pout_theory-Pout) * 100,'%');
```

The calculated results were found to be:

$$P_{out,theory} - P_{out} = 5.0056 * 10^{-4}$$

$$P_{out,theory} - P_{out} * 100\% = 0.0501\%$$

Remarks on the Lab

The Lab was completed successfully with little difficulty. The main source of difficulty came from writing the piecewise function for the shaping filter and getting it to run properly. The second part of difficulty showed up somewhere in the scale of the results. The final results seem to be very large for the numbers being generated, and I could not find the cause of the issue. The logic seems fine, but everything appears to be on orders of magnitude too large. Besides the above mentioned issues, the rest of the lab went off without a hitch. I enjoyed this lab, however I feel that perhaps the theory behind this lab may have been expanded on a little bit more in lieu of the procedural description.