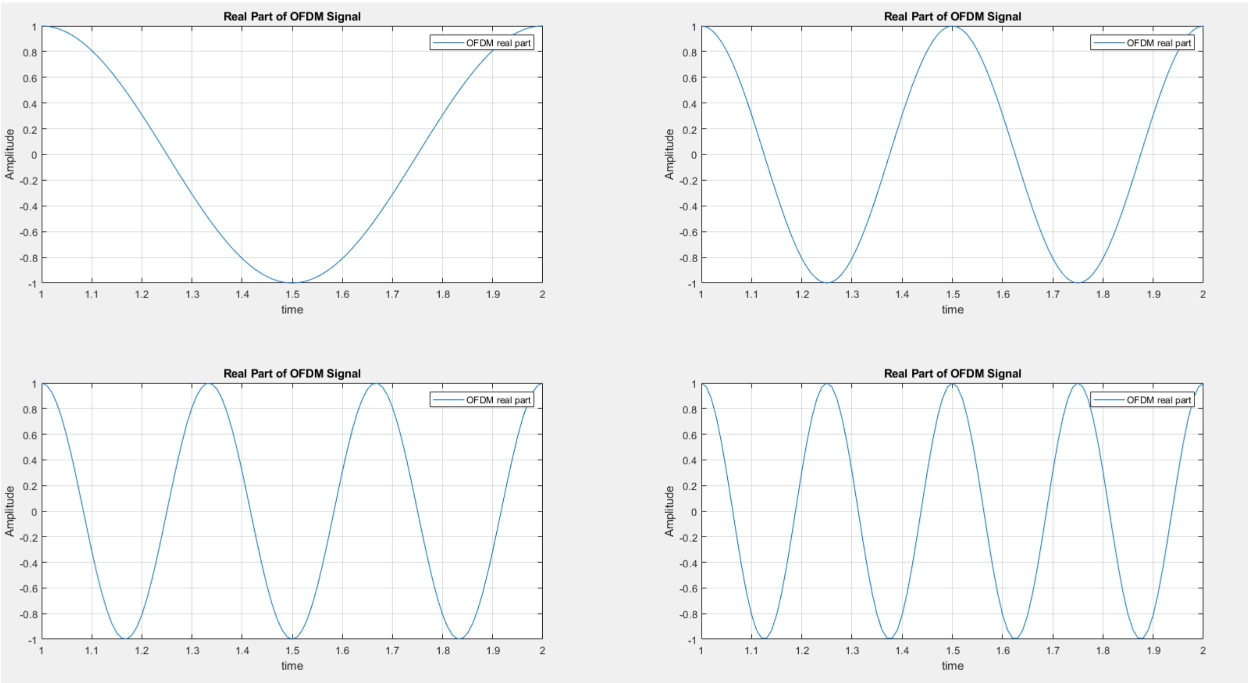
Steven Daniels
A53328625
Prelab6

Theory

1. Generation of OFDM Signals

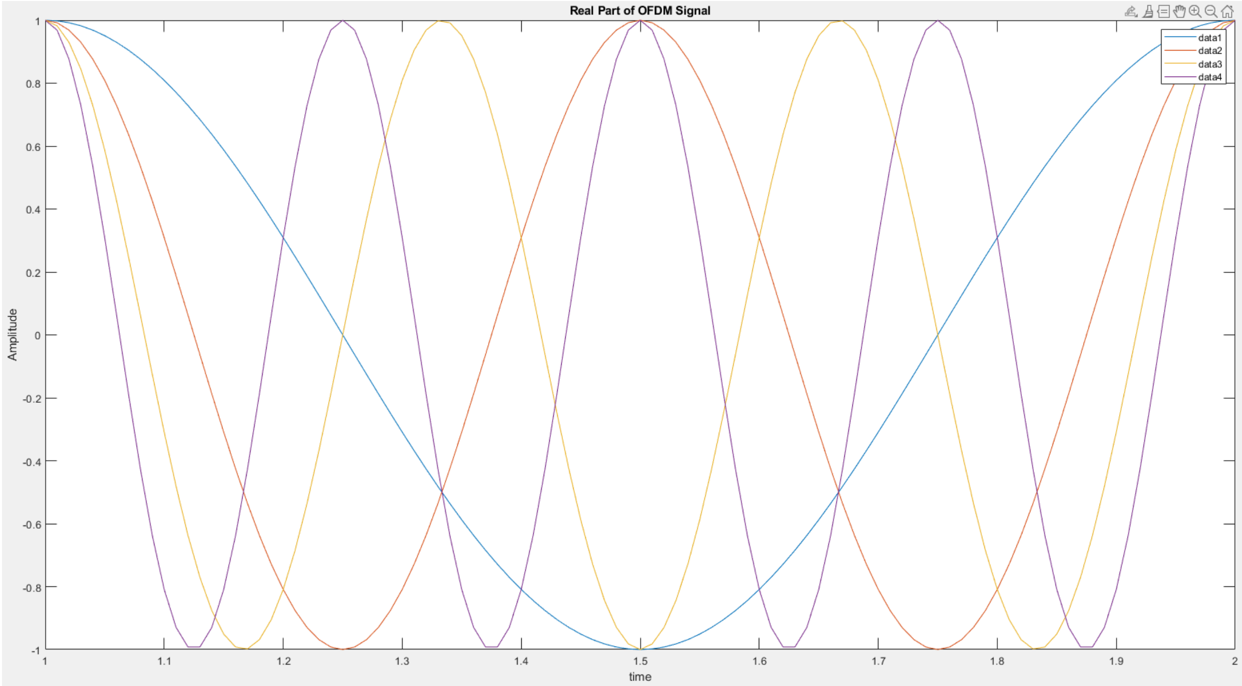
a.



- `% Parameters`
- `num_subcarriers = 4;              % Number of subcarriers`
- `f_0 = 1;                          % center frequency`
- `bw_hz = 2;                        % bandwidth`
- `delta_f  = bw_hz/num_subcarriers; % subcarrier spacing`
- `Ts_sec = 1/delta_f;               % symbol duration`

- symbol_rate_sps = 1/(Ts_sec*50); % symbol rate, i.e. the rate at which symbols are transmitted within each subcarrier
- t = 1:symbol_rate_sps:Ts_sec;   % time vector
- % Generate complex exponential carriers
- n_subcarrier = (1:num_subcarriers)';
- rows = size(n_subcarrier,1);
- cols = size(t,2);
- s_t = zeros(rows,cols);
- %% a) Plot the time waveform for each of the four subcarriers using f0 = 1.
- for nIdx = 1:size(n_subcarrier)
-    sub_carrier = n_subcarrier(nIdx);
-    s_t(nIdx,:) = exp(-1j * 2 * pi * sub_carrier * f_0 * t);
-    % Plot the generated OFDM signal
-    figure(1);
-    subplot(2, 2, nIdx);
-    plot(t,(s_t(nIdx,:)));
-    title('Real Part of OFDM Signal');
-    xlabel('time');
-    ylabel('Amplitude');
-    legend("OFDM real part")
-    grid on
-    hold on
- end
- hold off

b.



- % Parameters
- num_subcarriers = 4;   % Number of subcarriers
- f_0 = 1;   % center frequency
- bw_hz = 2;   % bandwidth
- delta_f = bw_hz/num_subcarriers;   % subcarrier spacing
- Ts_sec = 1/delta_f;   % symbol duration
- symbol_rate_sps = 1/(Ts_sec*50);   % symbol rate, i.e. the rate at which symbols are transmitted within each subcarrier
- t = 1:symbol_rate_sps:Ts_sec;   % time vector
- % Generate complex exponential carriers
- n_subcarrier = (1:num_subcarriers)';
- rows = size(n_subcarrier,1);
- cols = size(t,2);
- s_t = zeros(rows,cols);

- %% a) Plot the time waveform for each of the four subcarriers using f0 = 1.
- for nIdx = 1:size(n_subcarrier)
-     sub_carrier = n_subcarrier(nIdx);
-     s_t(nIdx,:) = exp(-1j * 2 * pi * sub_carrier * f_0 * t);
- end
- hold off
- %% b) Plot the spectrum for each of the four subcarriers on the same plot.
- for nIdx = 1:size(n_subcarrier)
-     % Plot the generated OFDM signal
-     figure(2);
-     plot(t,real(s_t(nIdx,:)));
-     title('Real Part of OFDM Signal');
-     xlabel('time');
-     ylabel('Amplitude');
-     legend
-     grid on
-     hold on
- end
- hold off
-
-

c. **Minimum subcarrier symbol duration such that subcarriers are orthogonal**
   - T = 1/2*fd

d. **Minimum subcarrier symbol duration such that subcarriers are orthogonal when each subcarrier has a random phase offset**
   - T = 1/fd

2. **OFDM and the FFT Matrix**
   a. **Show that all columns are orthogonal**
      - The orthogonality of a symmetric matrix can be determined by verifying that the dot product of each pair of different columns is zero.

prelab6_2_2_a_b.m

```matlab
%% OFDM and the FFT Matrix
%% 2.a Show that all columns are orthogonal
% Create an IDFT matrix
[twiddle_factors_mat] = calc_dft_twiddle_factors([1 1 1 1]);
disp('IDFT Matrix:');
disp(twiddle_factors_mat);
A = twiddle_factors_mat;
n = length(twiddle_factors_mat);
is_col_orthogonal = false;
% Iterate over pairs of columns
for i = 1:n-1
    for j = i+1:n
        % Calculate the dot product
        dot_product = dot(A(:,i), A(:,j));
        % Check if the dot product is close to zero (considering numerical precision)
        if abs(dot_product) > 1e-10
            is_col_orthogonal = false;
            return;
        end
    end
end
msg = sprintf('2.a: Dot product of Twiddle Factor columns: %d \n', is_col_orthogonal);
disp(msg);
identity = twiddle_factors_mat *conj(transpose(twiddle_factors_mat));
identity = identity/norm(identity);
```

ommand Window

```
>> prelab6_2_2_a_b
IDFT Matrix:
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i

2.a: Dot product of Twiddle Factor columns: 0
```

•

b. Write an expression for the inverse of the IFFt matrix. Call this matrix F

If

$$[F^{-1}] = \frac{1}{N} e^{-i2\pi k \frac{n}{N}}$$

Then,

$$[F^{-1}] = F \; = \; e^{i2\pi k \frac{n}{N}}$$

-
- **b.i) Verify that FF^-1 = I, in other words, show that the matrix F is the inverse of F^-1;**

```
%% 2.b.i Verify that FF^-1 = I, in other words, show that the matrix F is the inverse of F^-1;
[dft_twiddle_factors_mat] = calc_dft_twiddle_factors([1 1 1 1]);
[idft_twiddle_factors_mat] = calc_idft_twiddle_factors([1 1 1 1]);
disp('2.b.i: DFT Matrix * IDFT Matrix):');
disp(dft_twiddle_factors_mat * idft_twiddle_factors_mat);
```

ommand Window

```
>> prelab6_1_2
2.a: Dot product of Twiddle Factor columns: 0

2.b.i: DFT Matrix * IDFT Matrix):
   1.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
  -0.0000 - 0.0000i   1.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 - 0.0000i  -0.0000 - 0.0000i   1.0000 + 0.0000i  -0.0000 + 0.0000i
   0.0000 - 0.0000i   0.0000 - 0.0000i  -0.0000 - 0.0000i   1.0000 + 0.0000i
```

```


function [twiddle_factors_mat] = calc_dft_twiddle_factors(signal)
       N=length(signal);
    twiddle_factors_temp = zeros(N,N);
       for k=0:1:N-1
           for n=0:1:N-1
                   dft_sinusiod = exp(-1j*2*pi*n*k/N);
                   twiddle_factors_temp(k+1,n+1) = dft_sinusiod;
```

```
            end
    end
    twiddle_factors_mat = twiddle_factors_temp;
End
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
function [twiddle_factors_mat] = calc_idft_twiddle_factors(signal)
       N=length(signal);
    twiddle_factors_temp = zeros(N,N);
       for k=0:1:N-1
           for n=0:1:N-1
                   idft_sinusiod = exp(1j*2*pi*n*k/N);
                   twiddle_factors_temp(k+1,n+1) = idft_sinusiod;
            end
    end
    twiddle_factors_mat = twiddle_factors_temp * (1/N);
end


```

c. **Find a scaling constant C such that F′ = CF−1 and F′ becomes a unitary matrix**

- The scaling constant is the norm of the 1/sqrt(n)

d. **Find an input vector x such that the output vector y expressing the time series is a complex sinusoid at frequency $\omega = 2\pi$**

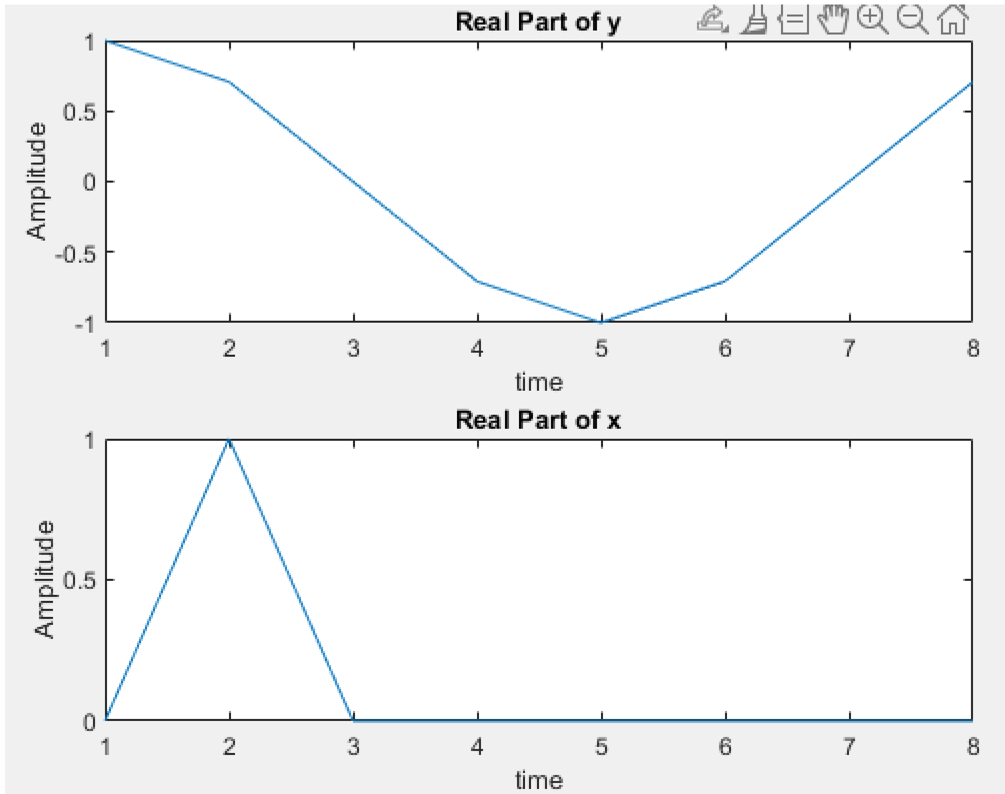
prelab6_2_2_d_e.m

```
%% Prelab6 Problem 2.2.d
% Find an input vector x such that the output vector y expressing the time
% series is a complex sinusoid at frequency ω = 2pi/N, where N = 8.
[dft_twiddle_factors_mat] = calc_dft_twiddle_factors([1 1 1 1 1 1 1 1]);
[idft_twiddle_factors_mat] = calc_idft_twiddle_factors([1 1 1 1 1 1 1 1]);
omega = 2*pi/8;
y = exp(1j*omega.*(0:7));

figure(1)
subplot(2, 1, 1);
plot(real(y))
disp(y)

disp(fft(y))
x = y * dft_twiddle_factors_mat;
subplot(2, 1, 2);
plot(real(x)/norm(x))
```

Command Window

```
>> prelab6_2_2_d_e
  Columns 1 through 5

   1.0000 + 0.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i  -0.7071 + 0.7071i  -1.0000 + 0.0000i

  Columns 6 through 8

  -0.7071 - 0.7071i  -0.0000 - 1.0000i   0.7071 - 0.7071i

  Columns 1 through 5

  -0.0000 + 0.0000i   8.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 6 through 8

  -0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

•



-

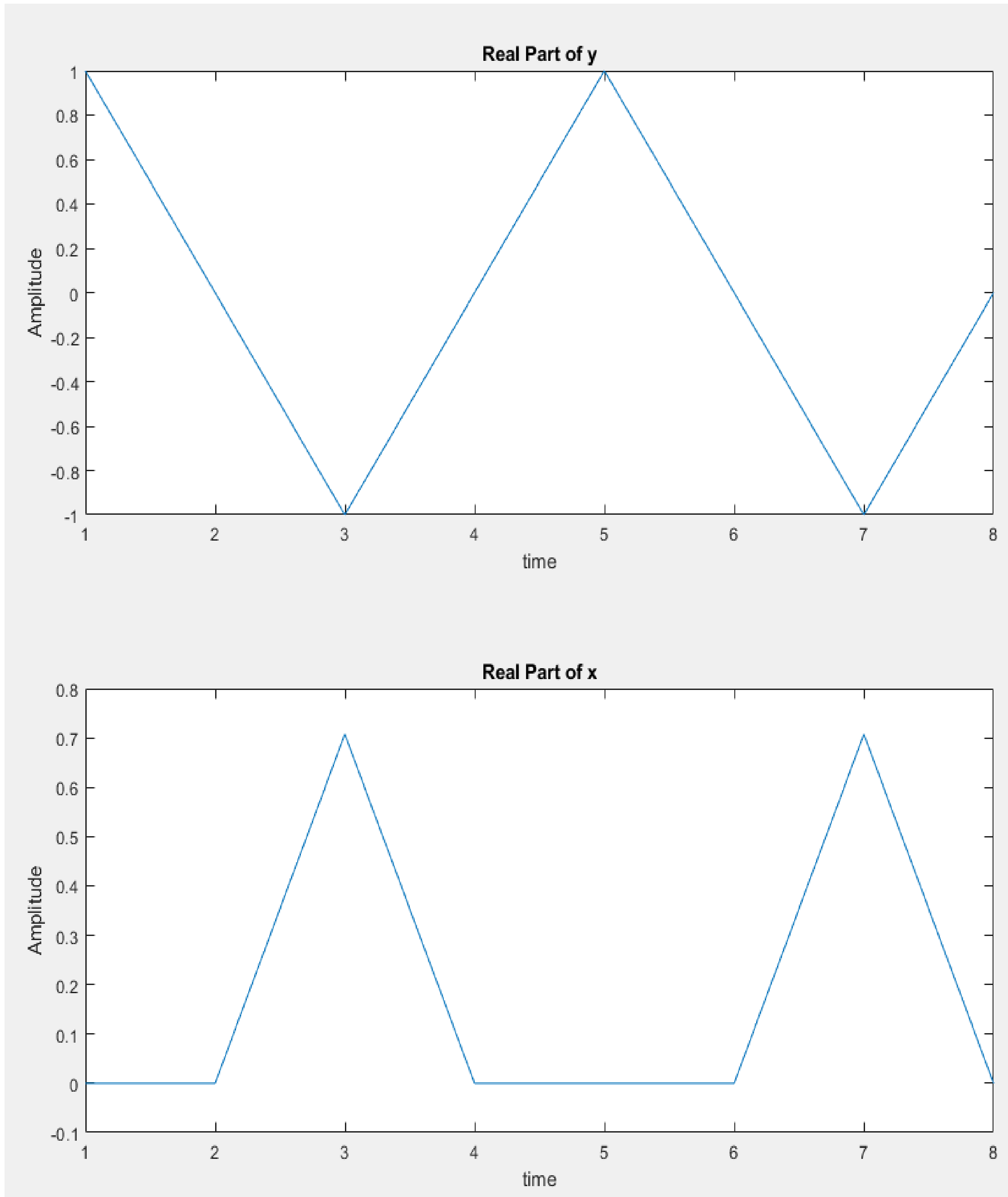
e. Find an input vector x such that the output vector y expressing the time series is a real sinusoid at frequency $\omega = 4\pi N$

prelab6_2_2_d_e.m

```
%% Problem 2.2.e
omega = 4*pi/8;
y = cos(omega.*(0:7));
figure(2)
subplot(2, 1, 1);
plot((y))
title('Real Part of y');
xlabel('time');
ylabel('Amplitude');

disp(y)
disp(fft(y))
x = y * dft_twiddle_factors_mat;
subplot(2, 1, 2);
plot(real(x)/norm(x))
title('Real Part of x');
xlabel('time');
ylabel('Amplitude');
```

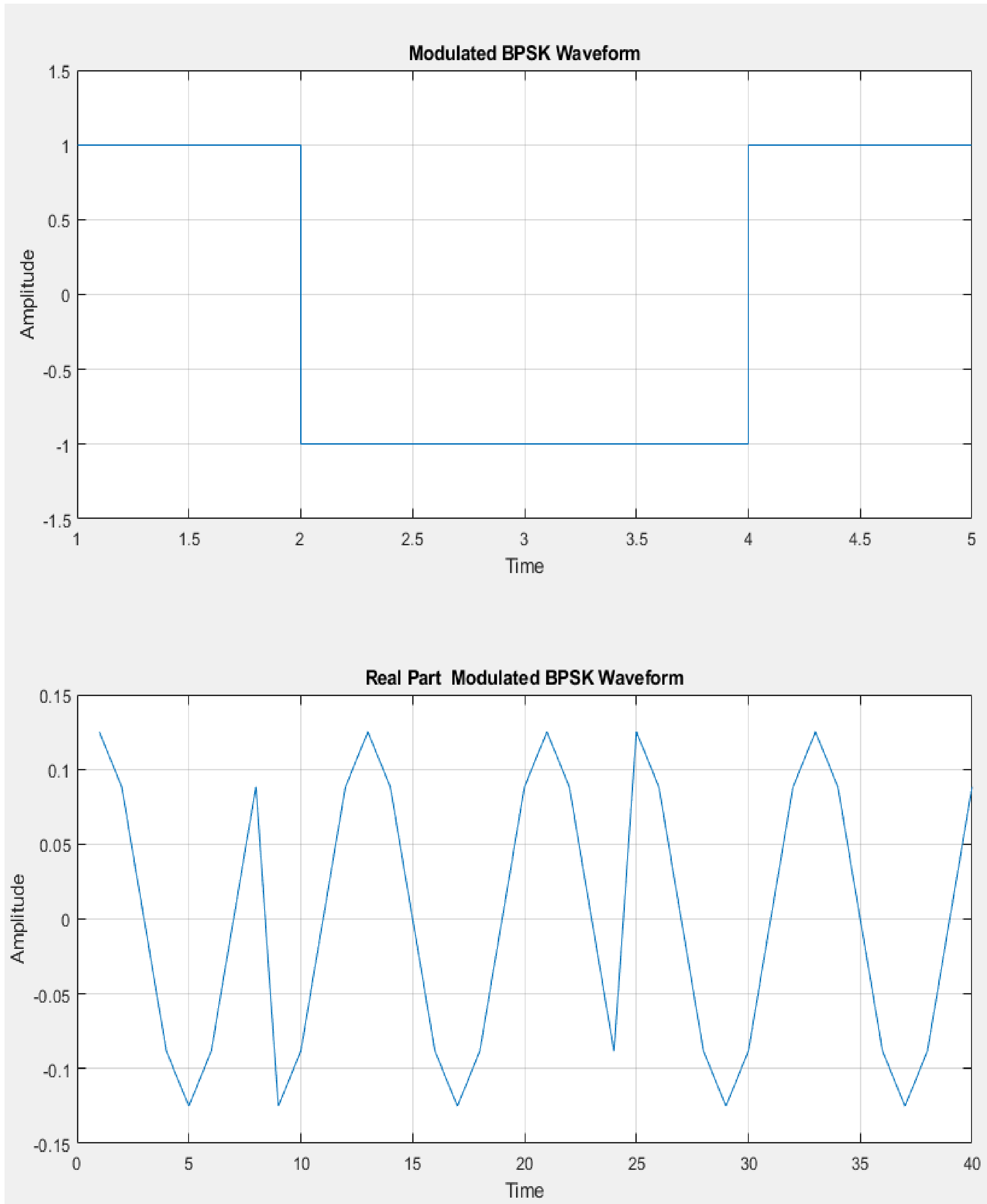
mmand Window

```
>> prelab6_2_2_d_e
  Columns 1 through 4

   1.0000 + 0.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i  -0.7071 + 0.7071i

  Columns 5 through 8

  -1.0000 + 0.0000i  -0.7071 - 0.7071i  -0.0000 - 1.0000i   0.7071 - 0.7071i

  Columns 1 through 4

  -0.0000 + 0.0000i   8.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

  Columns 5 through 8
```

- 

Real Part of y
Amplitude
time
Real Part of x
Amplitude
time

f. Create a modulated BPSK waveform at $N = 8$ samples per symbol at a carrier frequency of $\omega = 2\pi/N$ by using the IFFT matrix F−1, use the following message sequence $m = \{1, 0, 0, 1, 1\}$



Modulated BPSK Waveform
Amplitude
Time
1.5
1
0.5
0
-0.5
-1
-1.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Real Part Modulated BPSK Waveform
Amplitude
Time
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
0
5
10
15
20
25
30
35
40

•

```
%% Prelab6 Problem 2.2.f
% Define the number of samples per bit
N = 8;
% Generate a random bit sequence
bit_sequence = [1 0 0 1 1]; % Generating a sequence of 10 bits
% Modulate the bit sequence using BPSK
% Map 0 to -1 and 1 to 1
bpsk_signal = 2 * bit_sequence - 1;
% Flatten the array
bpsk_signal = bpsk_signal(:)';
% Time vector for plotting
t = 1:length(bpsk_signal);
% Plotting the modulated BPSK waveform
figure(1);
subplot(2, 1, 1);
stairs(t, bpsk_signal);
title('Modulated BPSK Waveform');
xlabel('Time');
ylabel('Amplitude');
axis([1 length(bpsk_signal) -1.5 1.5]);
grid on;
% IFFT BPSK using the F^-1 matrix
[idft_twiddle_factors_mat] = calc_idft_twiddle_factors([1 1 1 1 1 1 1 1]);
temp = zeros(8,5);
temp(2,:) = bpsk_signal;
modulated_bpsk =  idft_twiddle_factors_mat * temp ;
moduladated_bpsk_vec = reshape(modulated_bpsk,1,[]);
subplot(2, 1, 2);
plot(real(moduladated_bpsk_vec));
title('Real Part  Modulated BPSK Waveform');
xlabel('Samples');
ylabel('Amplitude');
```

- grid on;
-

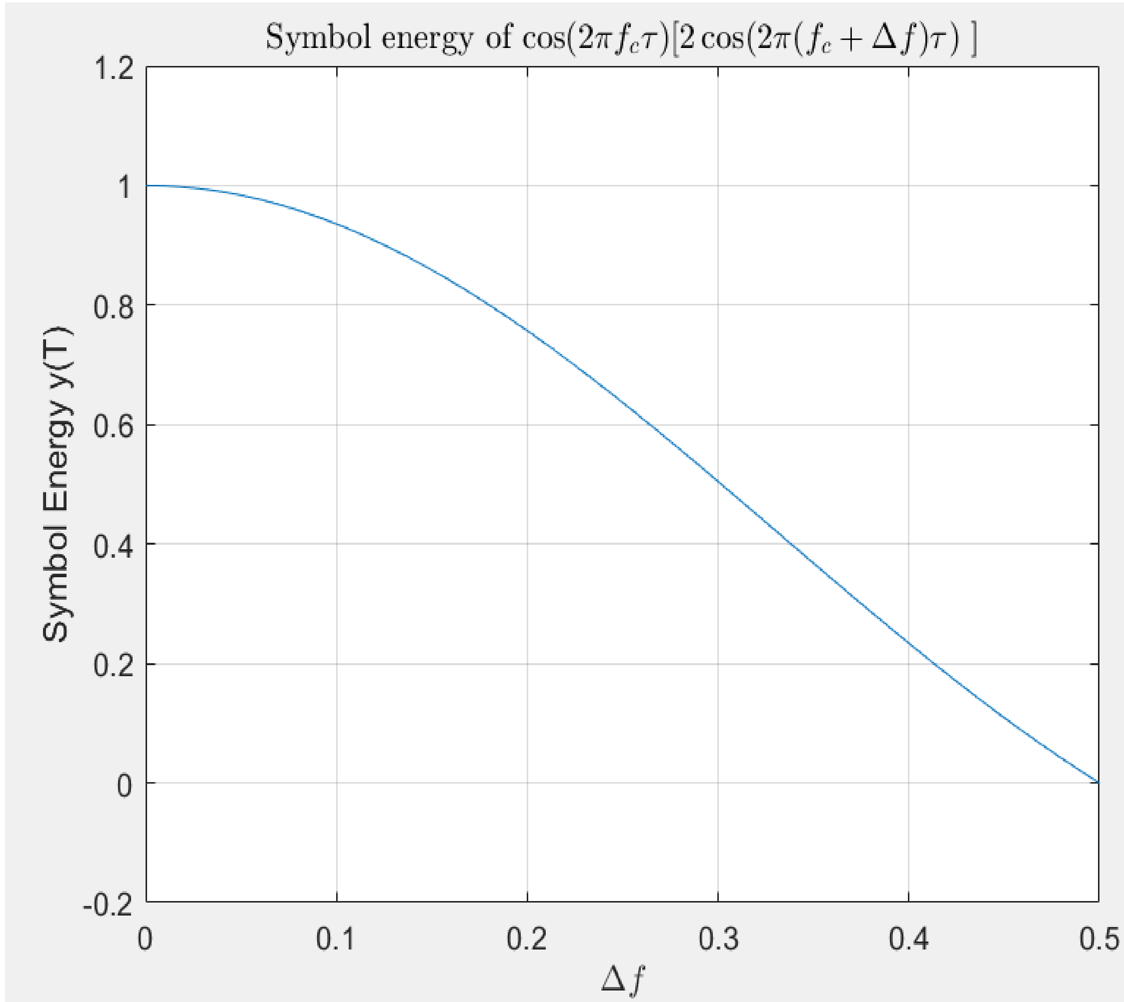
3. **Frequency and Bit Error Rate**
   a. **For a fixed symbol interval T, what is the smallest Δf that produces an orthogonal frequency signal with respect to the original frequency-modulated signal?**
      - fd=1/2T, where T is the symbol duration

   b. **Let the bit energy Eb for no frequency error (Δf = 0). Plot Eb as a function of Δf for T = 1 from Δf = 0 to the value determined in part (a) for which the local oscillator and the incident signal are orthogonal.**

prelab6_1_3_c.m

```matlab
%% Prelab6 Problem 3.b Plot the bit energy(Eb) as a function of delta_f for T = 1 from delta_f to 1/2*T
T = 1; % symbol duration
fc = 1e3;
% Define the function to integrate
func = @(t,Fc,Fd) cos(2*pi*Fc*t) .* (2*cos(2*pi*(Fc + Fd)*t));

delta_f_vec = 0:0.0005:1/(2*T);
y_t = zeros(size(delta_f_vec,2),1)';
for fdIdx = 1:size(delta_f_vec,2)
    % Perform the integration
    y_t(fdIdx) = (1/T) * integral(@(t) func(t,fc,delta_f_vec(fdIdx)), 0, T);
end

% Display the result,
disp(['Integral value: ', num2str(y_t)]);
figure(1)
plot(delta_f_vec,y_t)
title(['Symbol energy of $\cos({2} \pi f_c \tau) \lbrack 2\cos({2} \pi ' ...
    '(f_c + \Delta f) \tau$) \rbrack'],'Interpreter','latex');
xlabel('$\Delta f$','Interpreter','latex');
ylabel('Symbol Energy y(T)');
grid on;
```

-



-

c. For T = 1, determine the maximum value of Δfmax of the frequency offset such that the bit error rate does not change by more than a factor of two when Eb/N0 = 5 (linear).

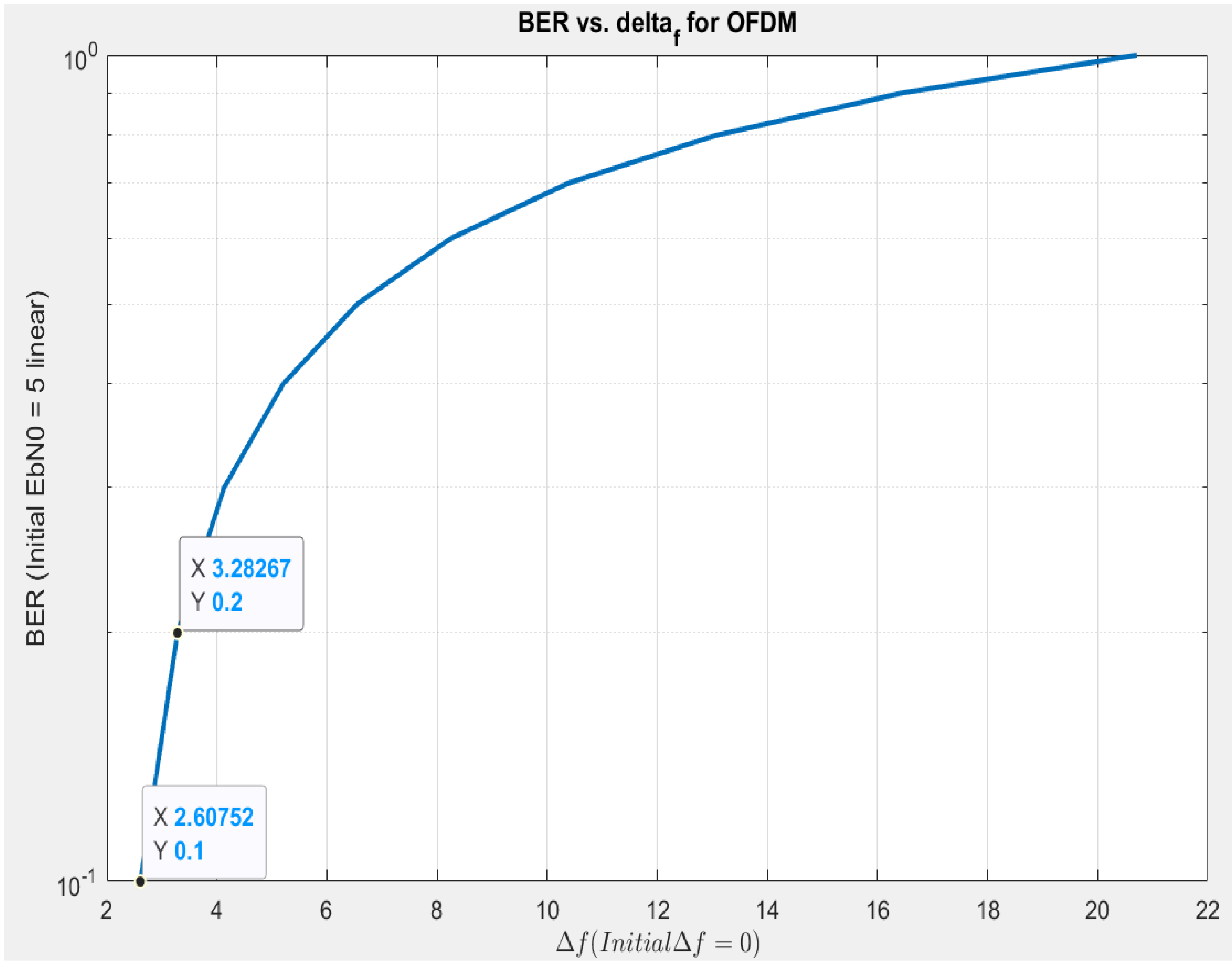
- Delta_f = 3.3

```matlab
% Given Eb/N0 in dB
EbN0_dB = 5;

% Convert Eb/N0 from dB to linear scale
EbN0 = 10^(EbN0_dB/10);

% Calculate BER
BER = qfunc(sqrt(EbN0));

disp(['Theoretical Bit Error Rate (BER): ' num2str(BER)]);
```

Command Window

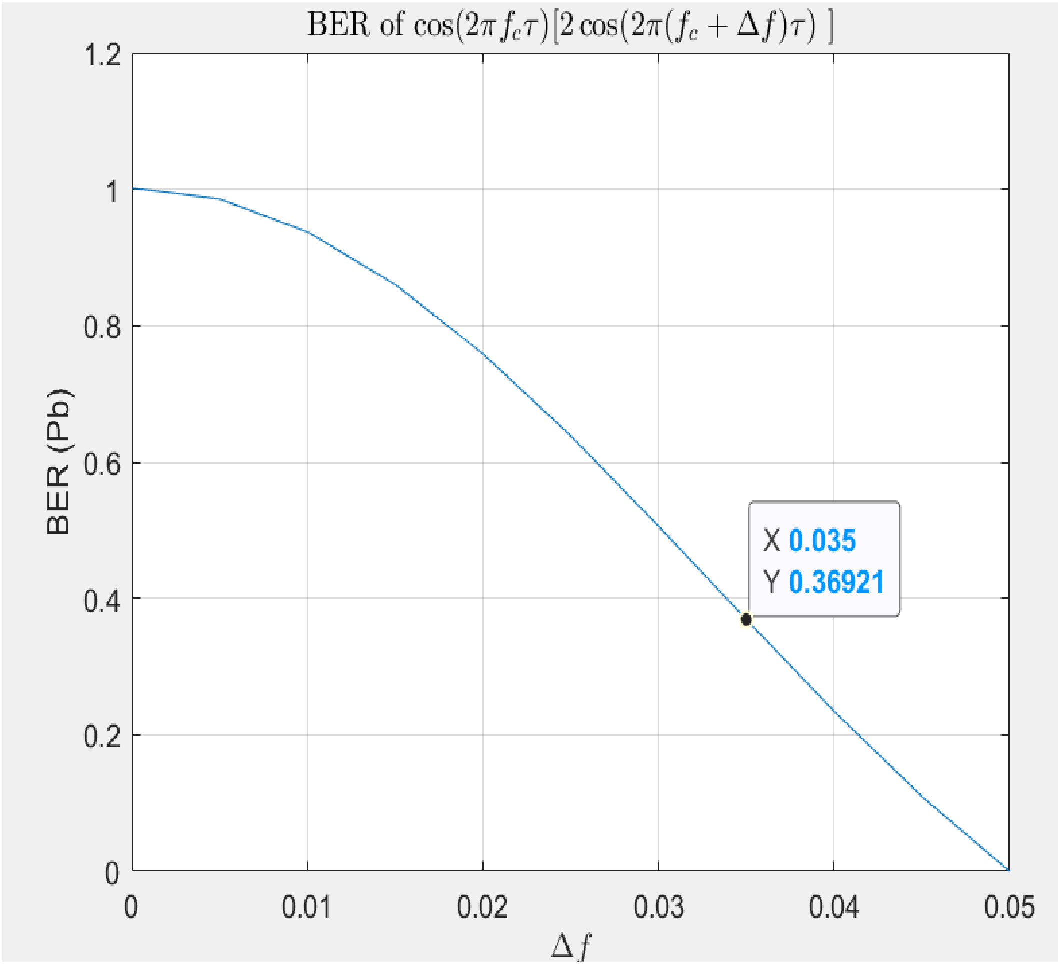
```
>> sandbox
Theoretical Bit Error Rate (BER): 0.037679
```

- B

- 

BER vs. delta_f for OFDM
BER (Initial EbN0 = 5 linear)
10^0
10^{-1}
X 3.28267
Y 0.2
X 2.60752
Y 0.1
2
4
6
8
10
12
14
16
18
20
22
$\Delta f (Initial \Delta f = 0)$

d. Using the procedure described in part (c) to find $\Delta fmax$, calculate the fractional stability $\Delta fmax/fc$ when fc = 10e9 Hz and T = 10 ms

- 

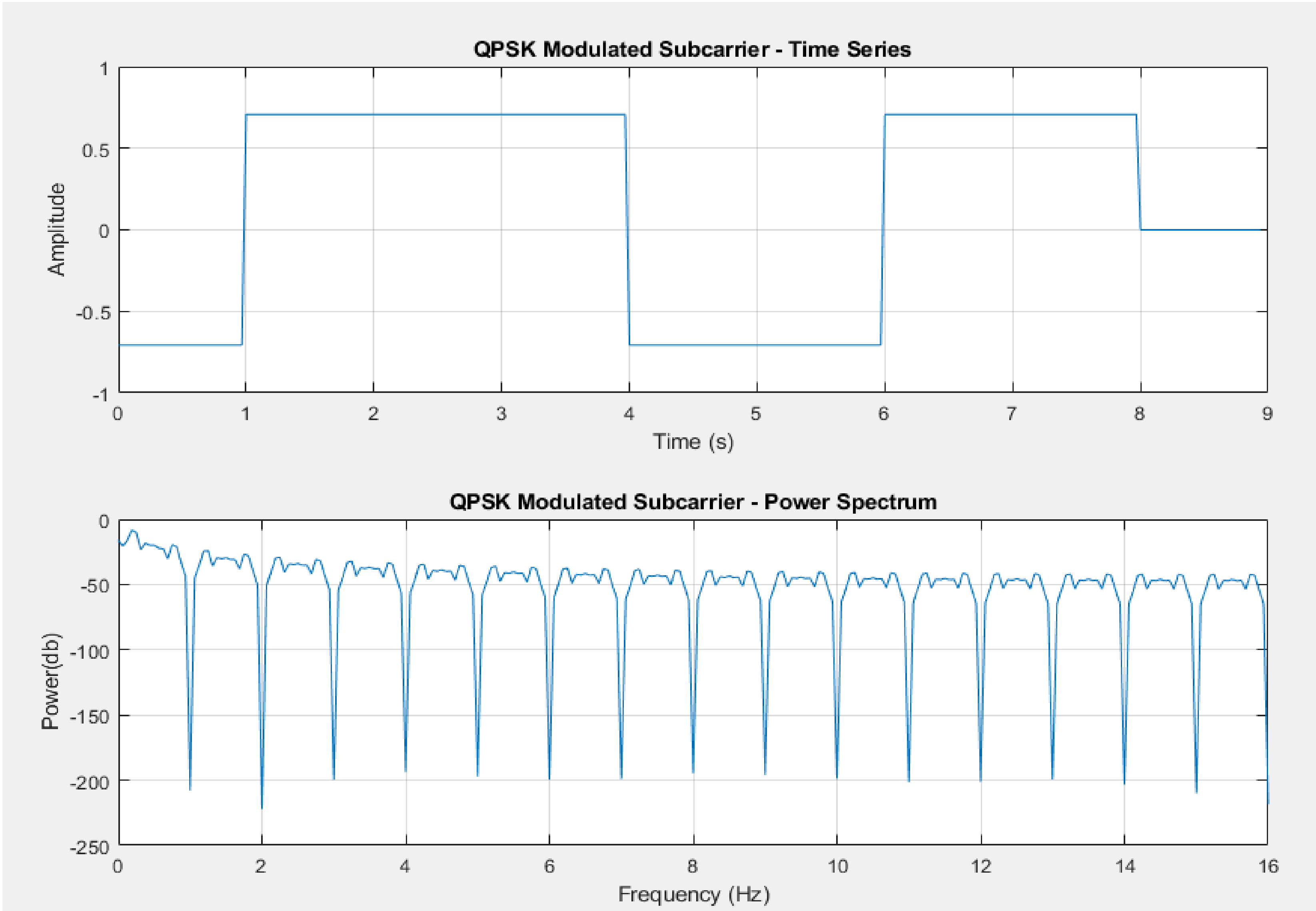
- Command Window

```
>> .035/10e9

ans =

   3.5000e-12
```

2 Simulation - Lab 6

1. **Generation and Synthesis of OFDM Signals.**

a. Generate a single, QPSK modulated subcarrier with random data using rectangular baseband pulses using 32 samples/symbols. Plot the power spectrum and the time series, and comment on each.



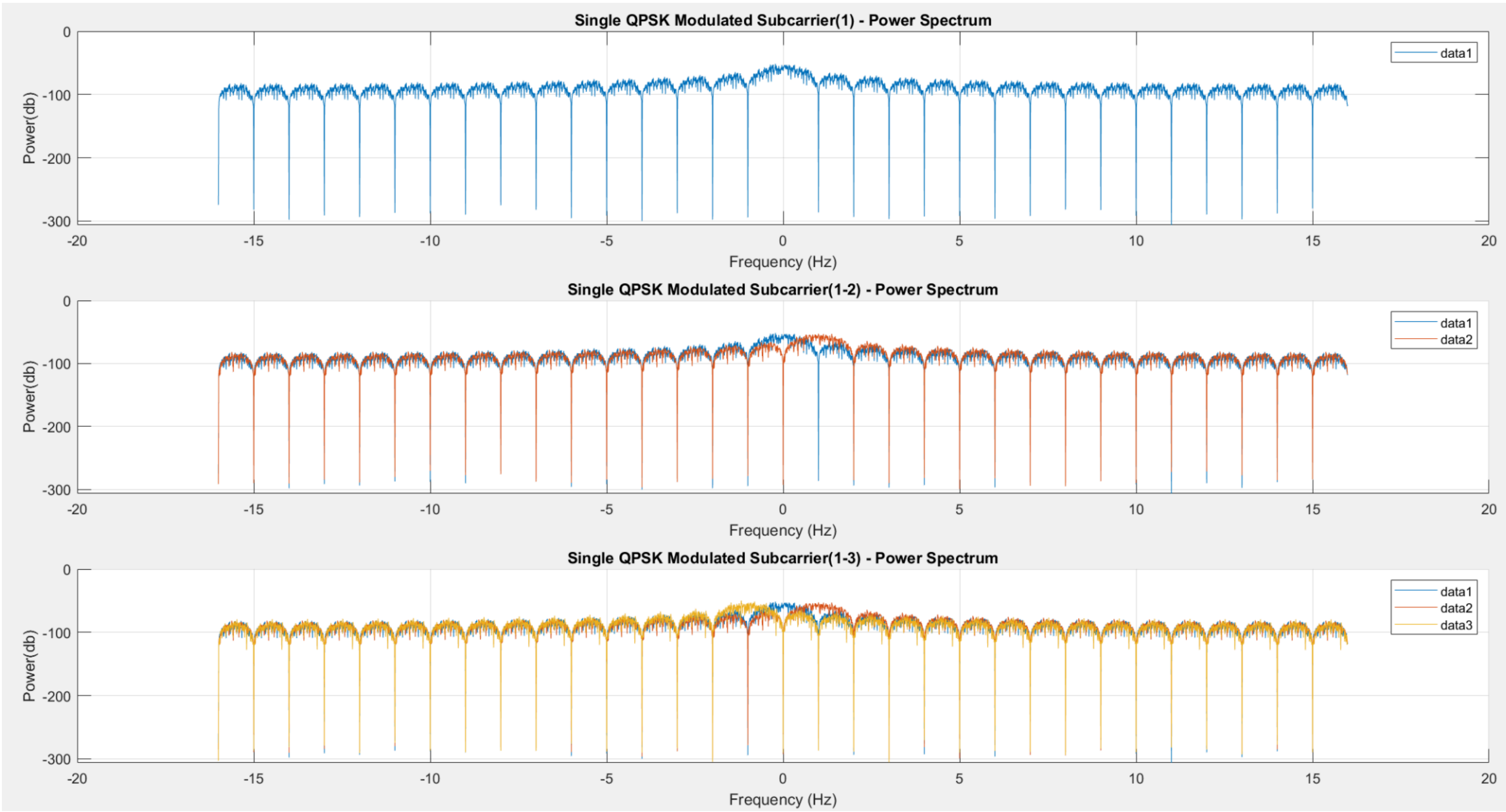
i. QPSK Modulated Subcarrier Time series Comments:

1. The time series is a rectangular pulse that rises to cover the bandwidth of each subcarrier channel.

ii. QPSK Modulated Subcarrier Power Spectrum Comments:

1. The subcarrier has the form of a sinc function with a 30dB dropoff.

b. Repeat part (a) adding two additional modulated subcarriers spaced ±fd = 1/T = 1/(NTs) apart from the original subcarrier where T is the OFDM block symbol duration.



i. QPSK Time series Comments:

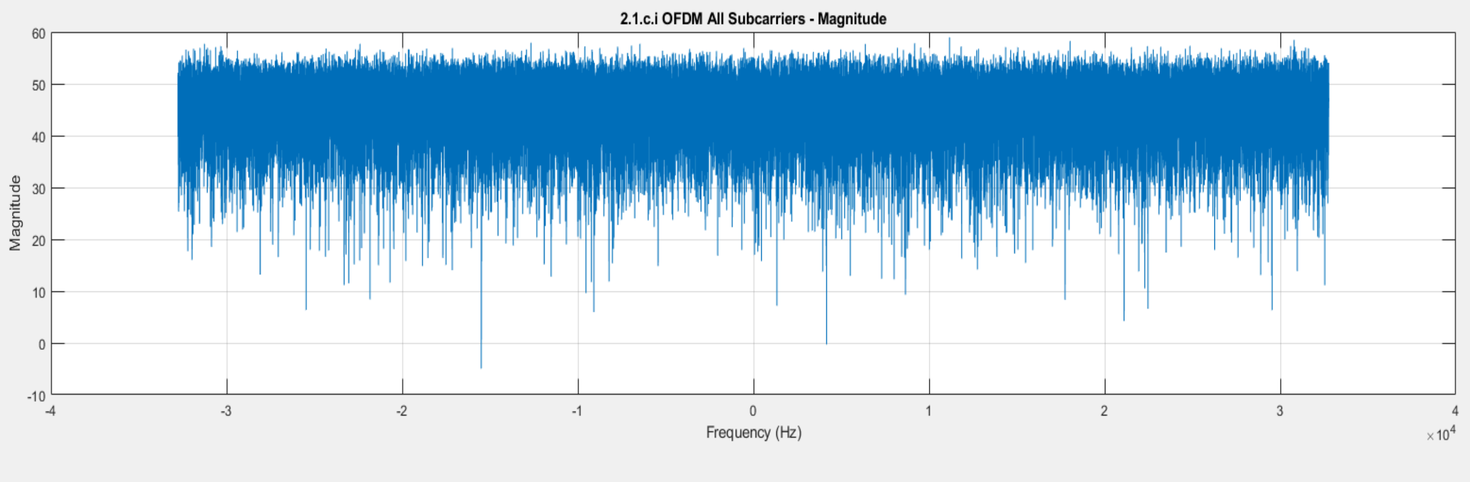
1. The time series is a rectangular pulse that rises to cover the bandwidth of each subcarrier channel.

ii. **QPSK Modulated Subcarrier Comments:**

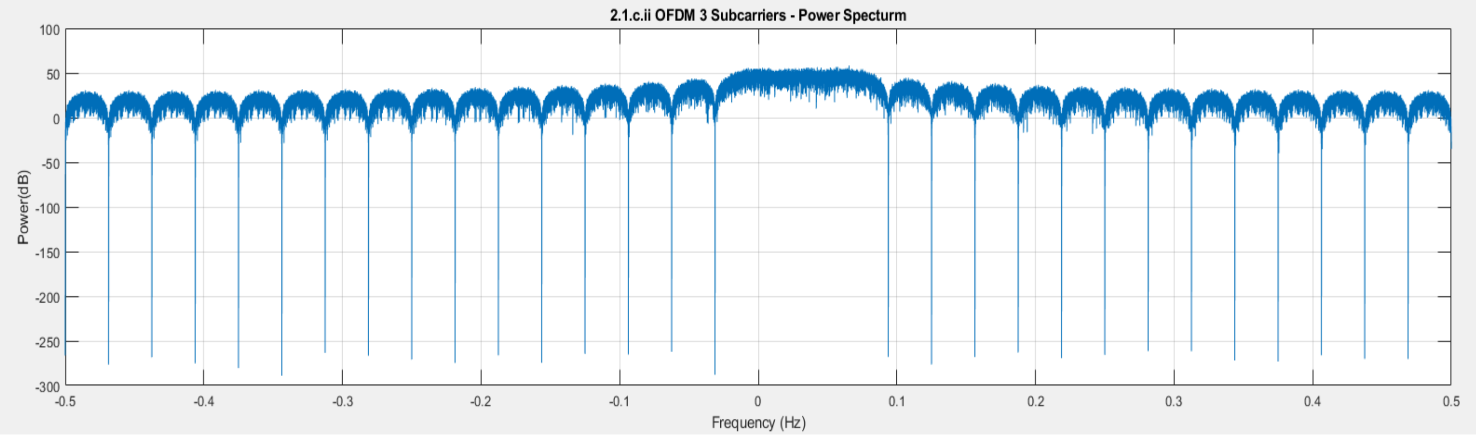
1. **The subcarrier has the form of a sinc function with a 30dB dropoff.The nulls of each subcarrier overlap with the peaks of the neighbors.**

c. Use a parallelizer to transform the serial data from a QPSK symbol mapper fed by a random bitstream.

i. Plot the magnitude of power spectrum of the total transmitted signal s(t) with all subcarriers enabled.

1. 

ii. Plot the power spectrum of s(t) with only three subcarriers enabled.

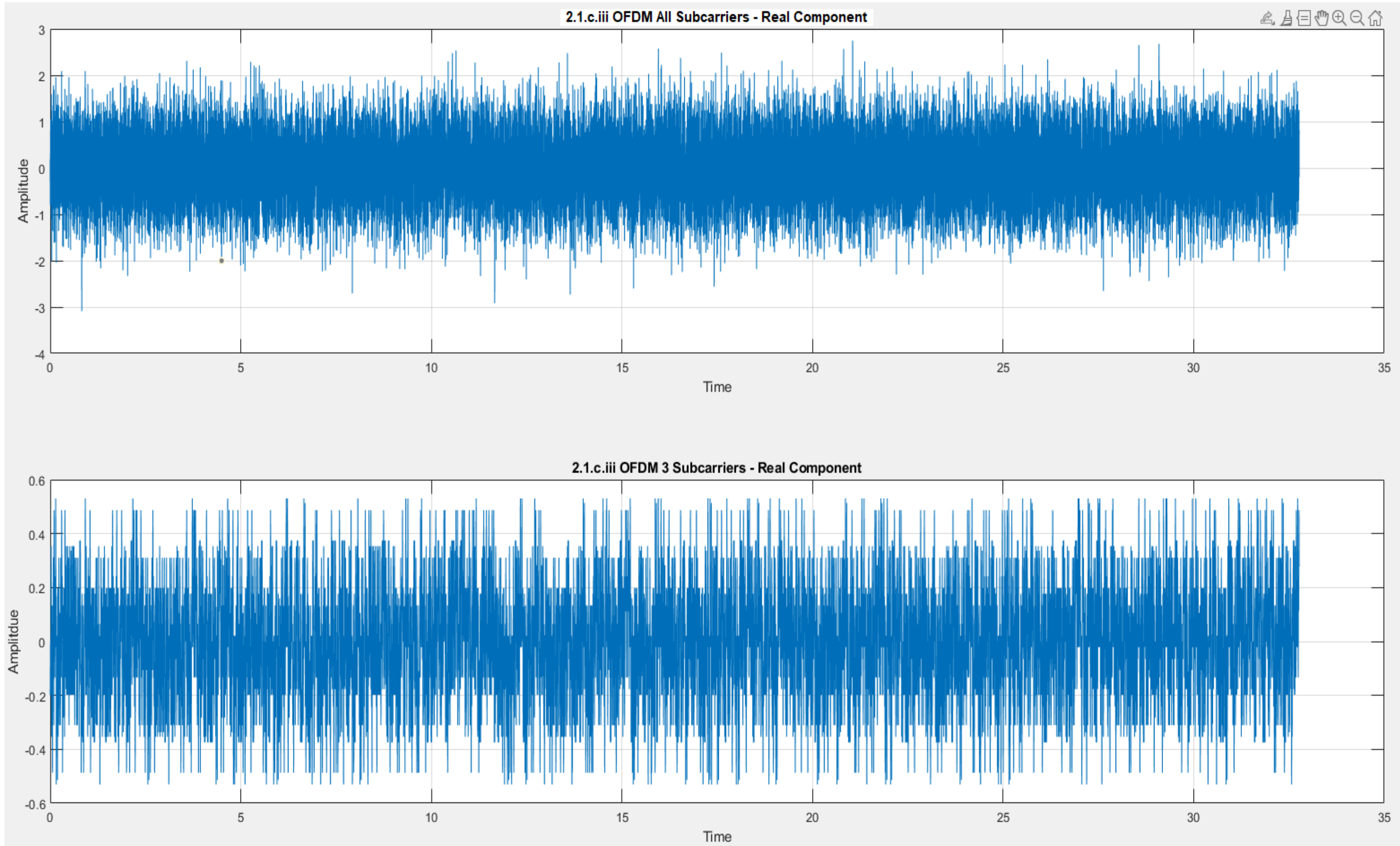
1. 

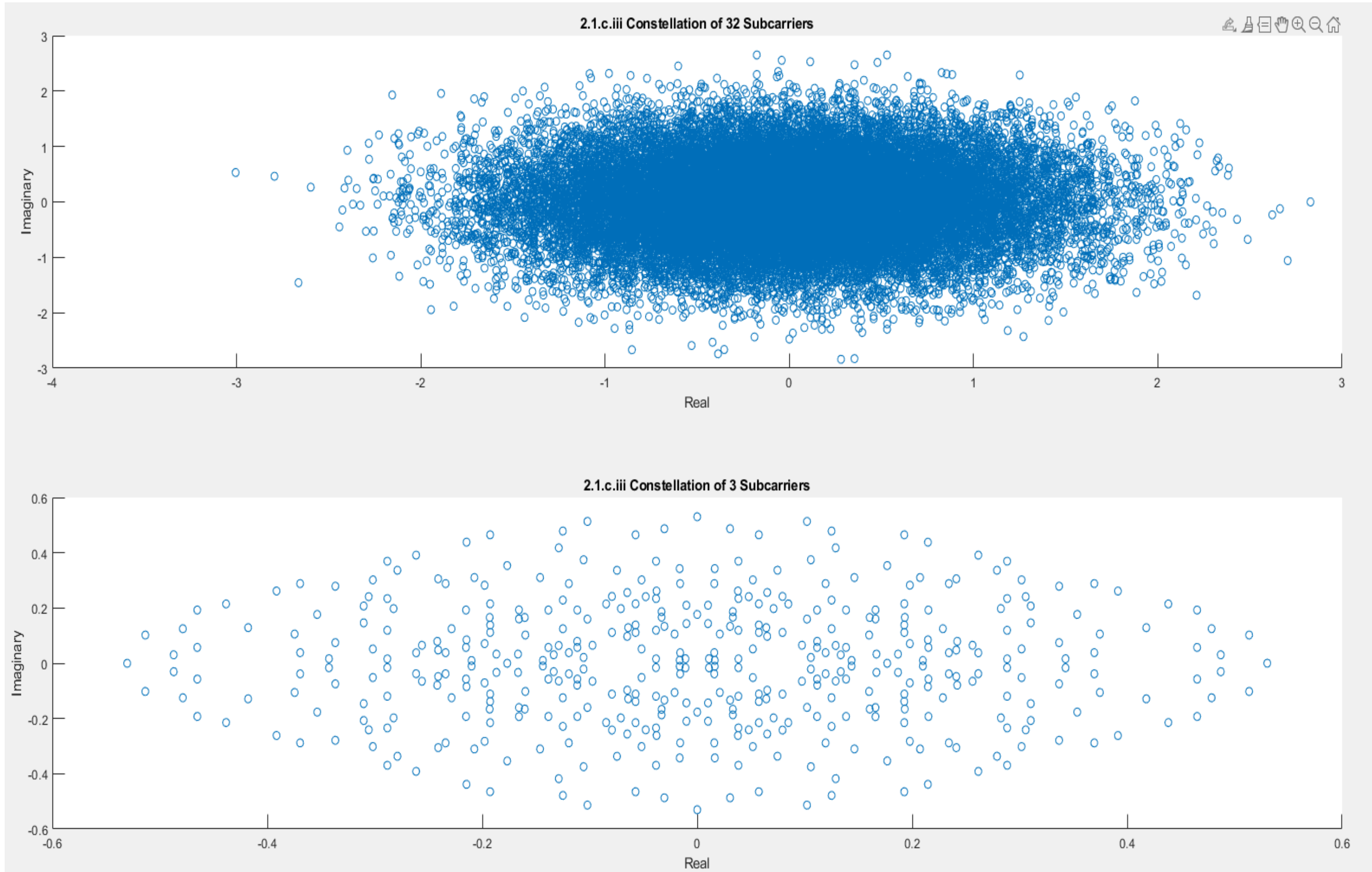
iii. Plot the real part of the corresponding time series for each case. What happens if you try and view all of the subcarriers’ constellations overlaid on top of one another?

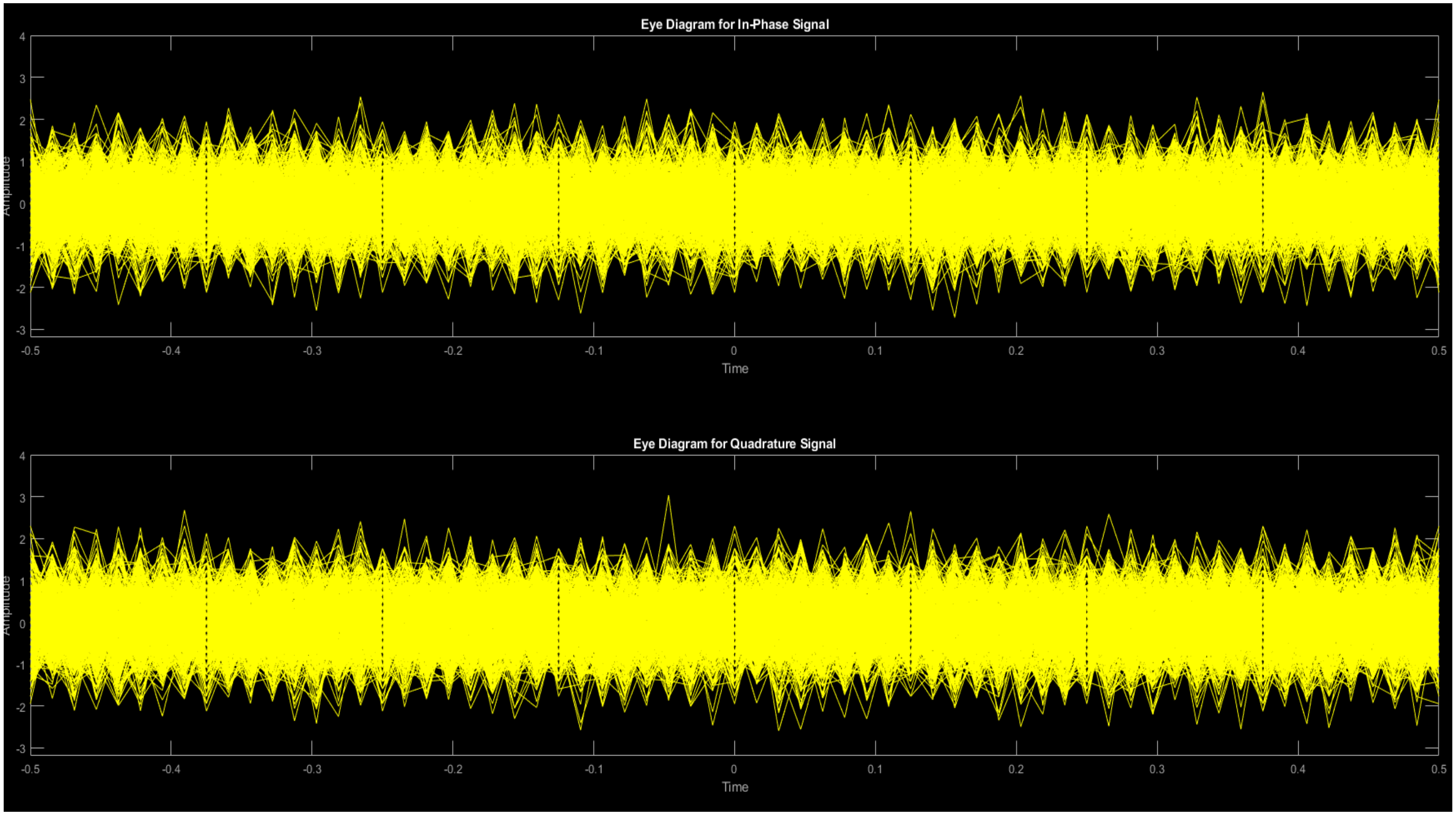
Answer: When plotting the subcarries constellations overlaid on top of one another the data is concentrated around the zero point.

What does this mean if you try to generate an eye-pattern to find the optimal sampling time?

Answer: The eye diagram will not show the optimal sampling time because the “eye” portion of the diagram is closed. This indicates that there is poor signal quality or distortion which could be due to noise, interference, or signal distortion. Essentially the eye diagram doesn’t provide any useful information about the OFDM signal. Primarily because the information is carried in the frequency domain not the time domain.


2.1.c.iii OFDM All Subcarriers - Real Component
Amplitude
Time
2.1.c.iii OFDM 3 Subcarriers - Real Component
Amplitdue
Time

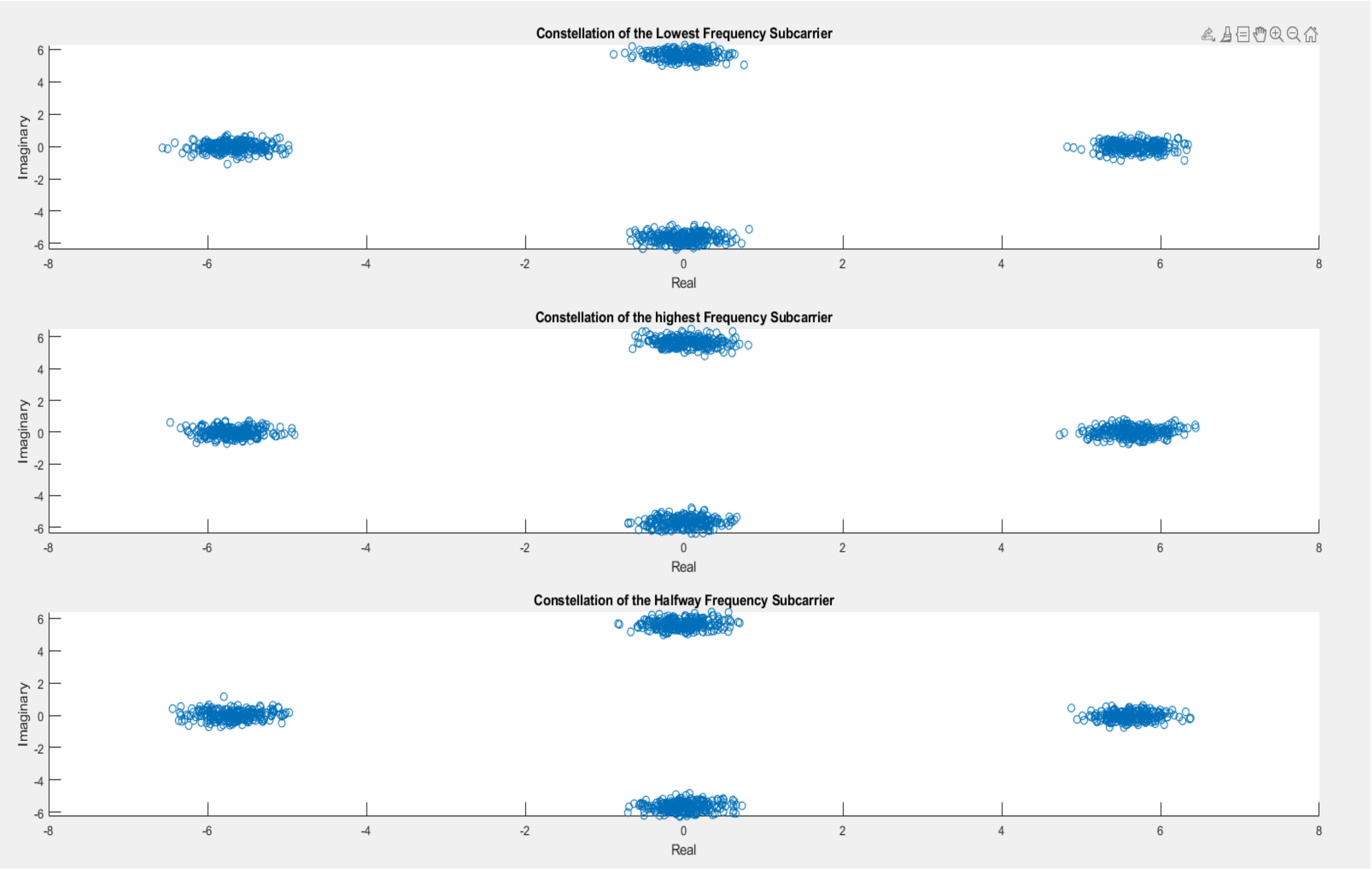

2.1.c.iii Constellation of 32 Subcarriers
Imaginary
Real
2.1.c.iii Constellation of 3 Subcarriers
Imaginary
Real



d. Take the complex-baseband time waveform generated after the IFFT and add a circularly symmetric gaussian random variable with a standard deviation $\sigma = 0.05$ per quadrature component. Note that this is equivalent to adding an independent gaussian random variable to both the in-phase and quadrature sample values.

e. Take the forward FFT of this noisy time sequence to produce the noisy received values in each subcarrier for one OFDM frame of 32 complex-valued symbols.

f. Now generate multiple OFDM frames (at least 1000) and plot the QPSK constellation for the subcarrier with the smallest frequency, the subcarrier with the largest frequency, and the subcarrier with a frequency that is halfway in between the extreme values.

i. 

g. Generate a complex-baseband time sequence of at least 1000 OFDM frames. Now, using this sequence, create a "frame synchronization" error by offsetting FFT relative to the OFDM frame as shown below

   i. Start with a one symbol offset and view the three constellations described in part (g) as a function of time as each OFDM frame is demodulated.

      1. What happens to each subcarrier constellation? Why? (Recall that a time shift of t0 produces a phase shift of 2πt0f in the frequency domain.)

   ii. Repeat for an offset of two symbols and compare the results to the one-symbol case

      1. Answer

a. The QPSK constellation looks noisy and has a phase offset in each subcarrier.