Marwin B. Alejo 2020-20221 EE274_ProgEx04

Table of Contents

Also accessible through http://www.github.com/soymarwin/ee274/EE274_ProgEx04 for history tracking.

Tutorials for DFT

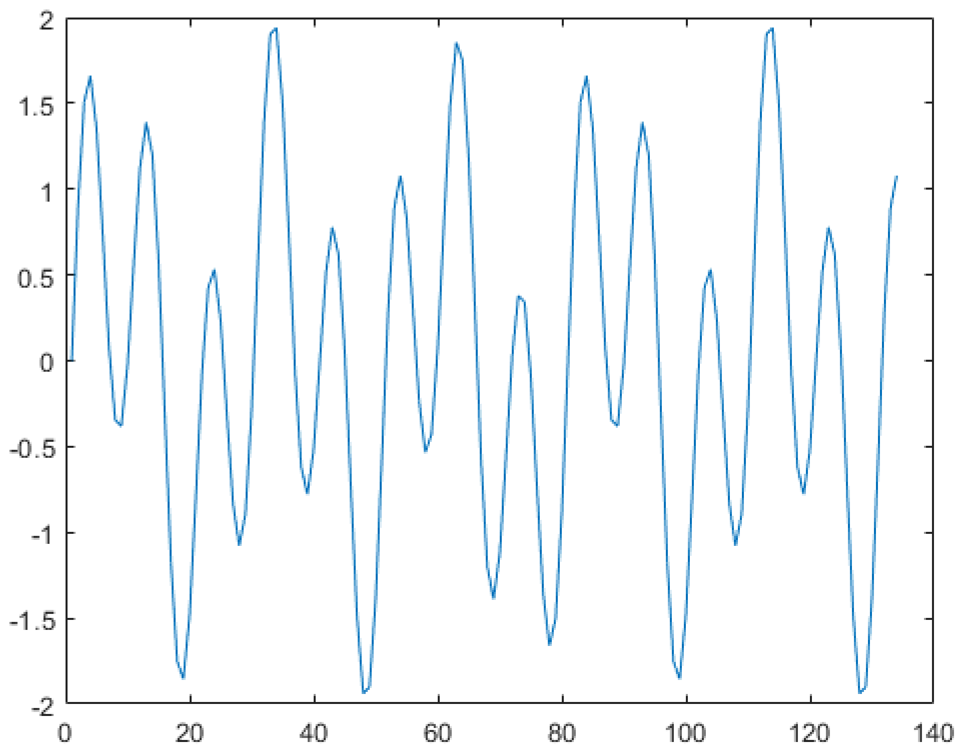
A. Resolving two signals of close frequency

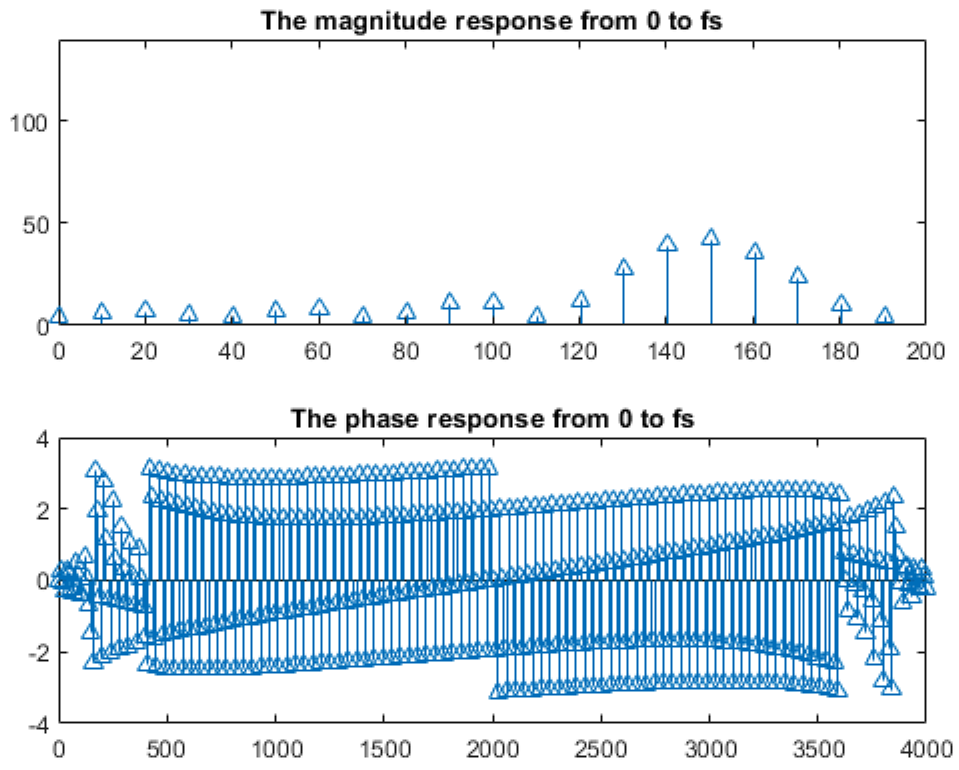
A.1-2. Pre-answered in the tuts.

A.3.a-b.

```
fs = 4000;
freq = [400 150];
```

```
amp = [1.2 0.8];
ph = [0 0];
t = [0:1/fs:5/min(freq)];
% generate the sinusoid
y
 =((amp'*ones(1,length(t))).*sin((2*pi*freq'*t)+ph'*ones(1,length(t))))';
figure(); plot(sum(y'));

ty_sum = sum(y');
y_sum = ty_sum(1:100);
Nfft = 400;
y_fft = fft(y_sum, Nfft);
% generate the frequency axis
w = linspace(0,fs,Nfft);
% Plot the magnitude and the phase response
mag = abs(y_fft);
ph1 = angle(y_fft);
figure(); subplot(2,1,1), stem(w, mag, '^');
axis ([0, 200, 0, 140]);
% use axis to zoom into plot to determine frequencies of sinusoids
title('The magnitude response from 0 to fs');
axis; % turn autoscaling back on
subplot(2,1,2), stem(w,ph1, '^');
title('The phase response from 0 to fs');
```



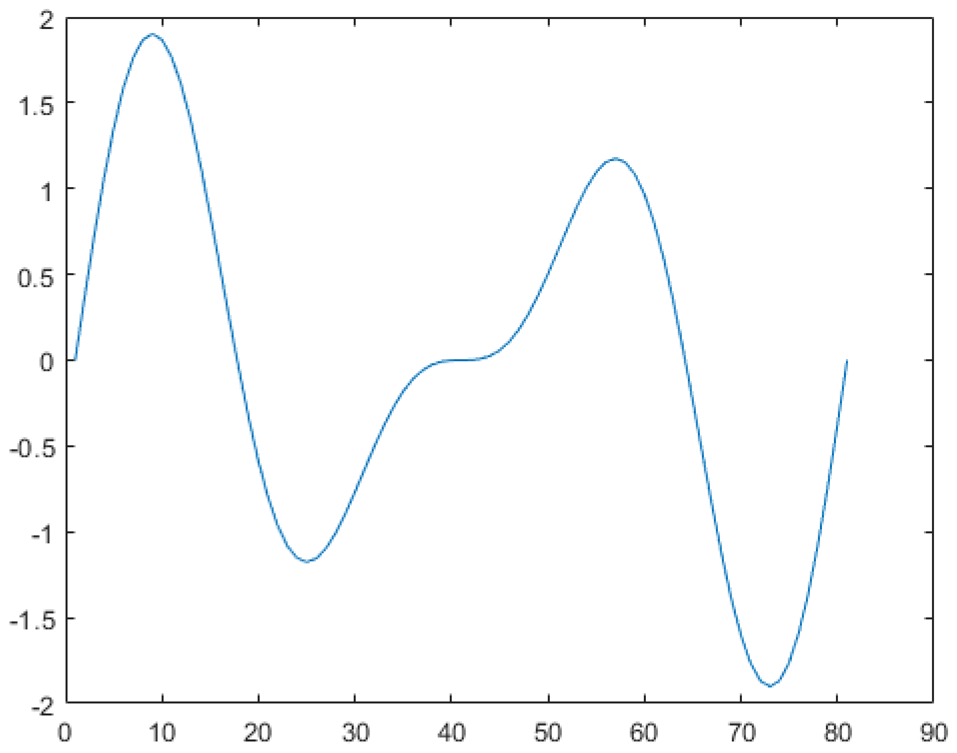


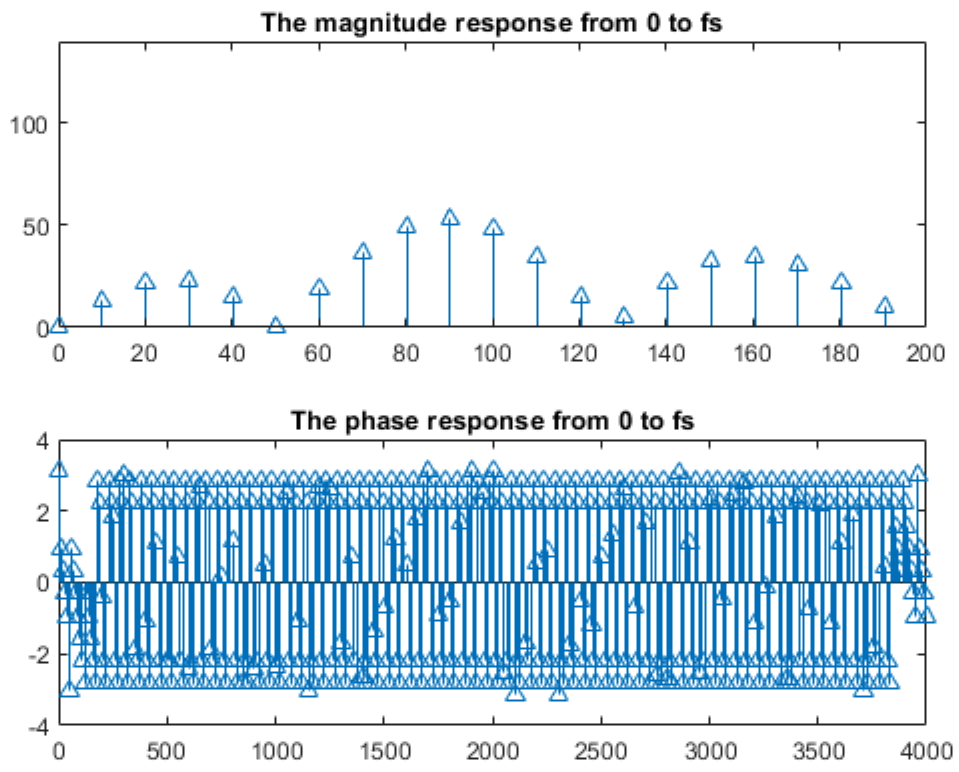
Increasing the number of samples allow us to correctly determine the frequencies of the sinusoids with more accuracy as it increases the numbers of phases and magnitudes f the DFT.

A.4.

```
fs = 4000;
freq = [100 150];
amp = [1.2 0.8];
ph = [0 0];
t = [0:1/fs:2/min(freq)];
% generate the sinusoid
y
 =((amp'*ones(1,length(t))).*sin((2*pi*freq'*t)+ph'*ones(1,length(t))))';
figure(); plot(sum(y'));

ty_sum = sum(y');
y_sum = ty_sum();
Nfft = 400;
y_fft = fft(y_sum, Nfft);
% generate the frequency axis
w = linspace(0,fs,Nfft);
% Plot the magnitude and the phase response
mag = abs(y_fft);
ph1 = angle(y_fft);
figure(); subplot(2,1,1), stem(w, mag, '^');
axis ([0, 200, 0, 140]);
```

```
% use axis to zoom into plot to determine frequencies of sinusoids
title('The magnitude response from 0 to fs');
axis; % turn autoscaling back on
subplot(2,1,2), stem(w,ph1, '^');
title('The phase response from 0 to fs');
```





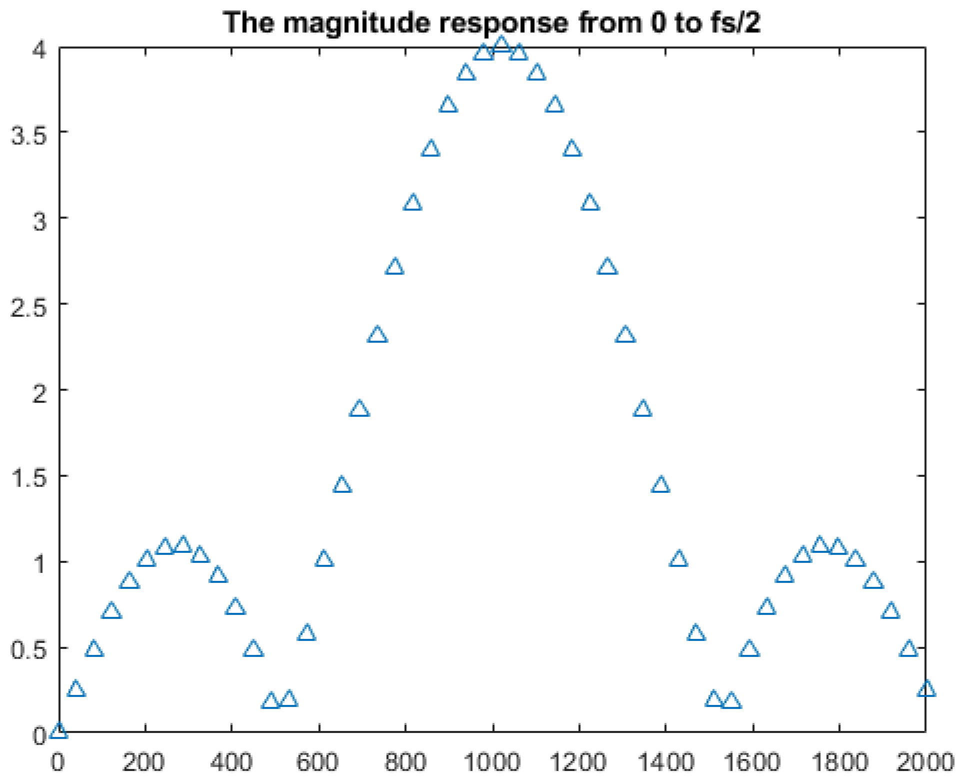
% The DFT can resolve the frequencies if the two signals however with N=100 it cannot be accurately identified due to few magnitude and phase spectrum as compared with N=400 which yielded to having more samples of phase and magnitude which shows the frequency in a more accurate plot.

B. Effects of zero padding and signal length on the DFT

B.1.

```
fs = 4000;
freq = 1000;
amp = 1;
ph = 0;
t = [0:1/fs:2/freq];
% generate the sinusoid
y = amp*sin((2*pi*freq*t)+ph)';
y_sum = y;
Nfft = 100;
y_fft = fft(y_sum, Nfft);
% generate the frequency axis
w = linspace(0, fs/2, Nfft/2);
y_fft = y_fft(1:length(y_fft)/2);
% Plot the magnitude response,
mag = abs(y_fft);
```

```
figure(); plot(w, mag, '^');
title('The magnitude response from 0 to fs/2');
```

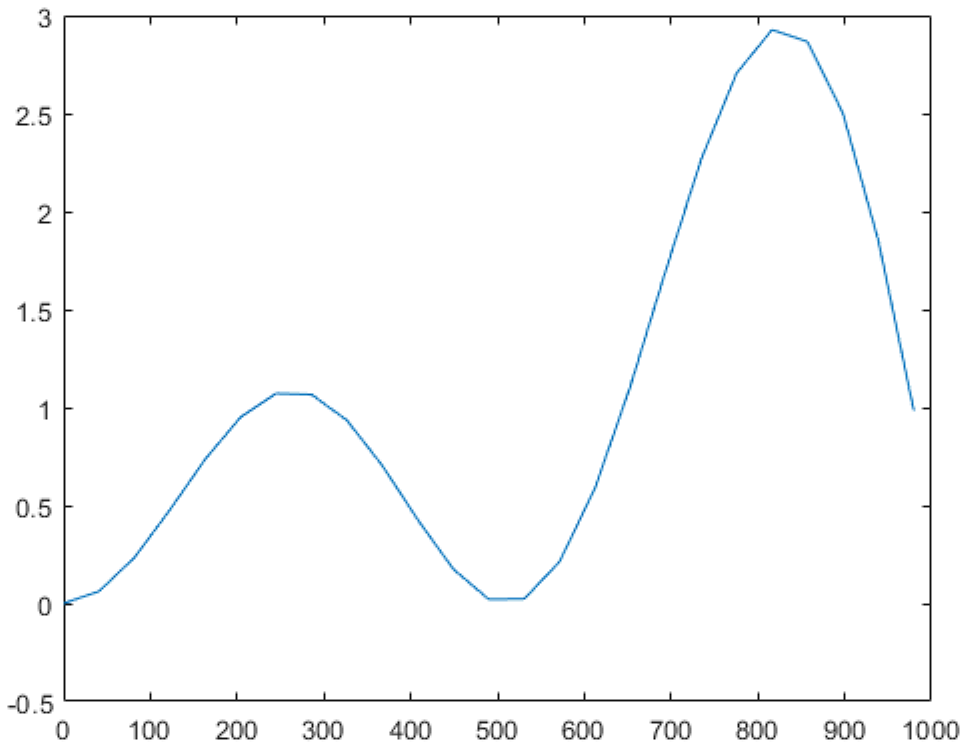


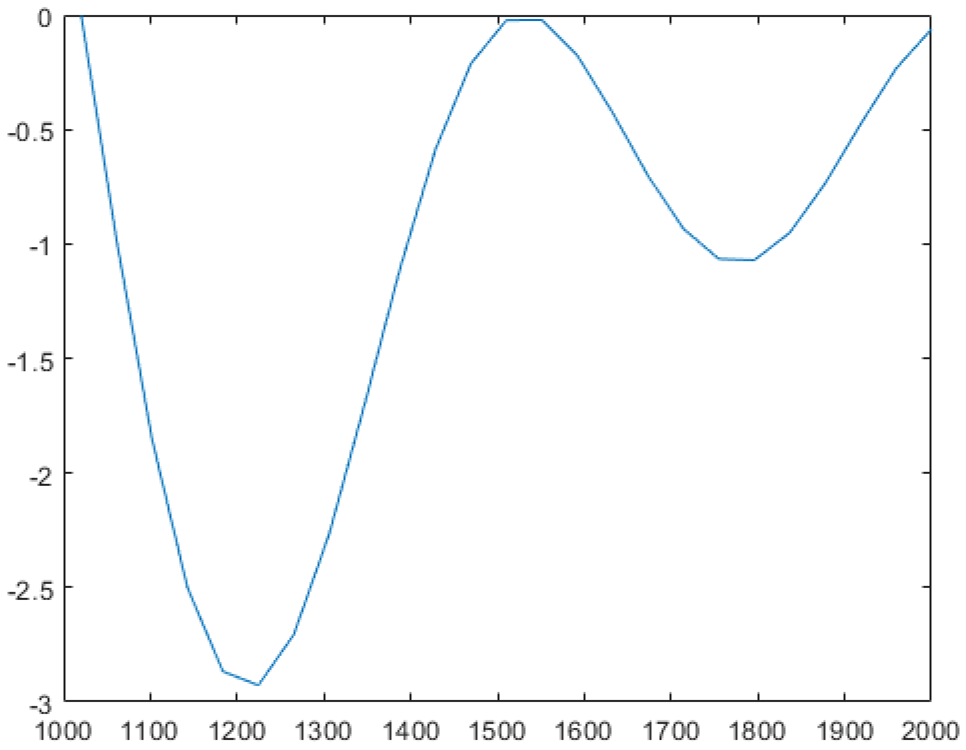
B.1.a.

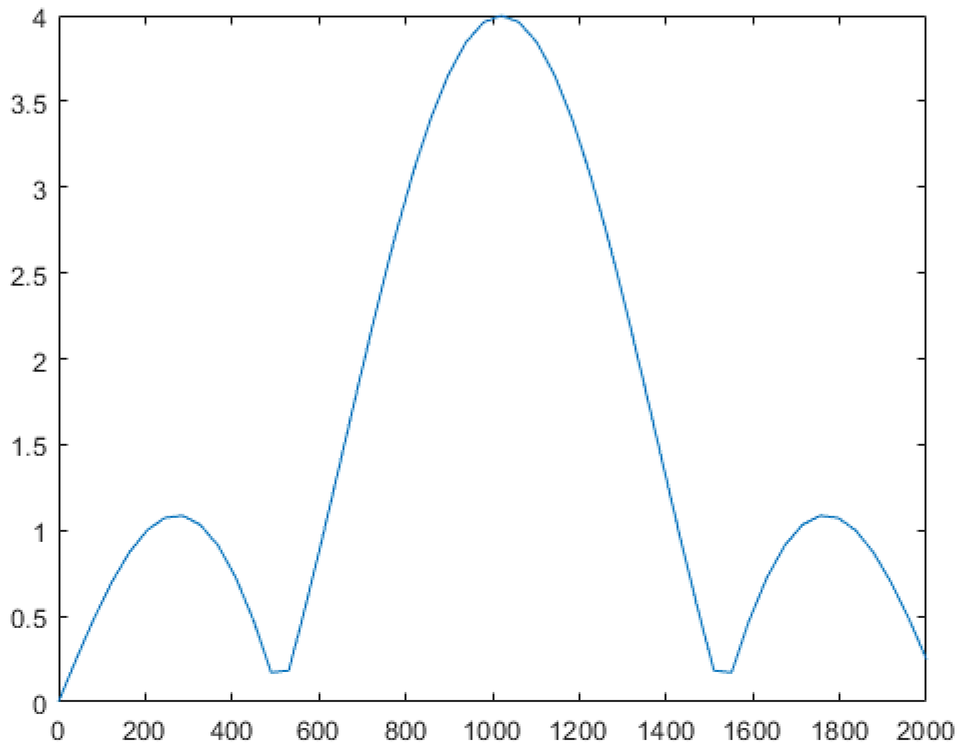
Through abs(), all the values of y_fft especially the negative values are transformed into positives as such the sum values from 500:1500 samples yields to a wave with a max magnitude of 4 as shown in the figure above(B.1) and below.

```
figure(); plot(w(1:25), y_fft(1:25));
figure(); plot(w(26:50), y_fft(26:50));
figure(); plot(w,abs(y_fft));
```

```
Warning: Imaginary
parts of complex X
and/or Y arguments
ignored.
Warning: Imaginary
parts of complex X
and/or Y arguments
ignored.
```


3
2.5
2
1.5
1
0.5
0
-0.5
0
100
200
300
400
500
600
700
800
900
1000


0
-0.5
-1
-1.5
-2
-2.5
-3
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
2000



B.1.b.

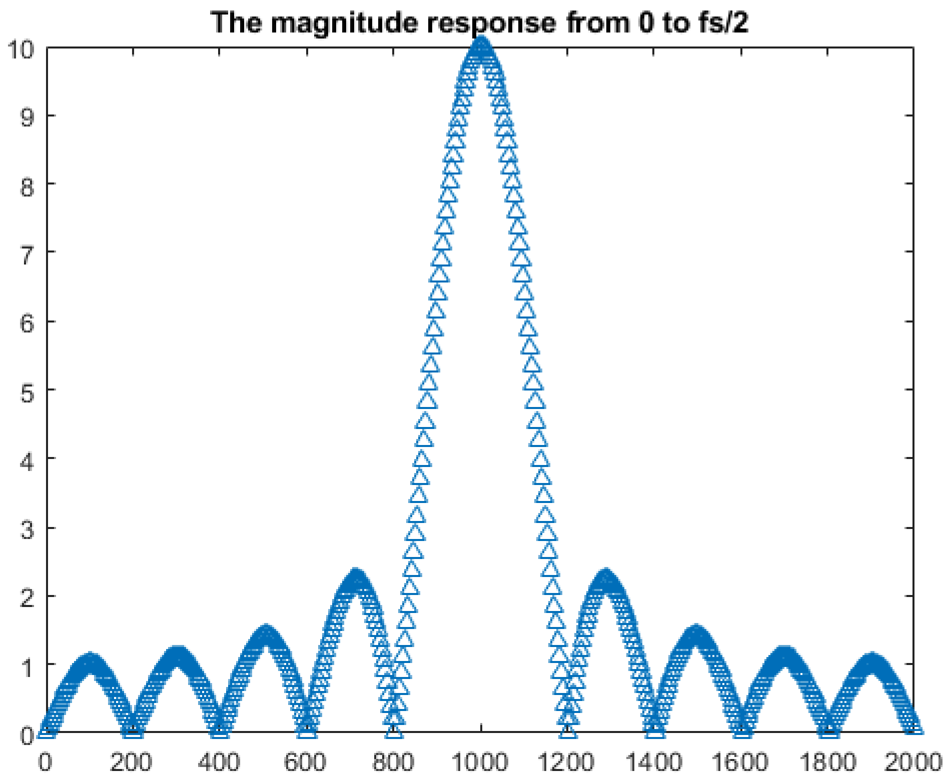
Increasing the number of frequency samples is equivalent to having a more accurate magnitude response and its plot.

B.1.c.

The same result as with B.1.b for increasing the number of samples. Furthermore, increasing the periods means increasing the magnitude's amplitude as shown in figure below.

```
fs = 4000;
freq = 1000;
amp = 1;
ph = 0;
t = [0:1/fs:5/freq]; % period inc to 5 from 2.
% generate the sinusoid
y = amp*sin((2*pi*freq*t)+ph)';
y_sum = y;
Nfft = 1000; % increased to 1k for fun
y_fft = fft(y_sum, Nfft);
% generate the frequency axis
w = linspace(0, fs/2, Nfft/2);
y_fft = y_fft(1:length(y_fft)/2);
% Plot the magnitude response,
mag = abs(y_fft);
```

```
figure(); plot(w, mag, '^');
title('The magnitude response from 0 to fs/2');
```



B.1.d.

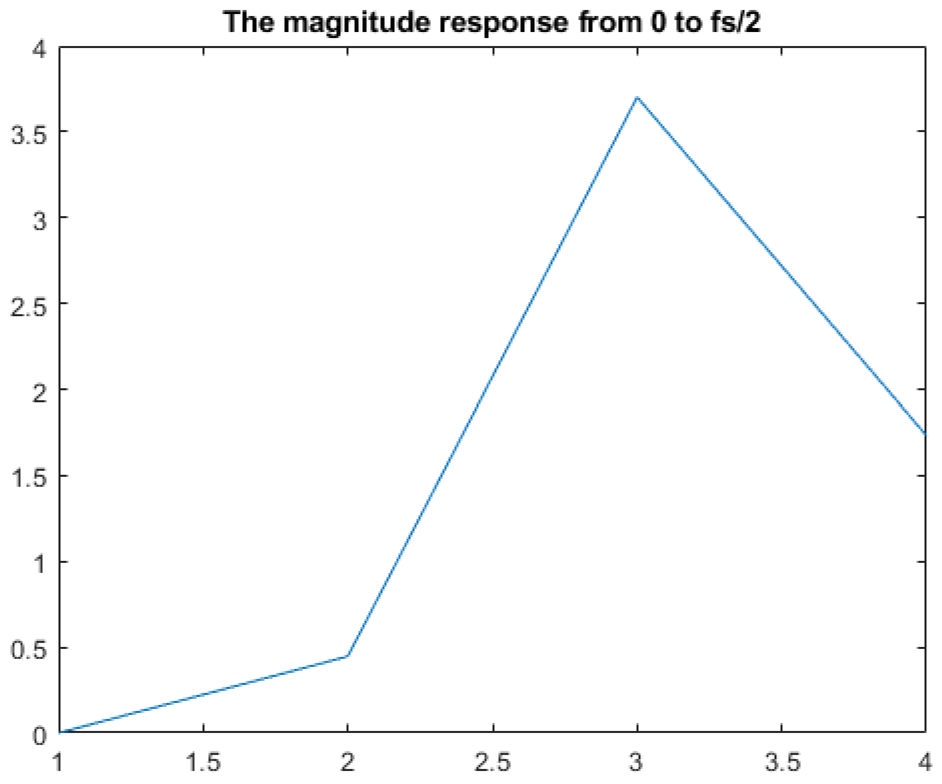
Increasing the signal length means extending the range of lobes over the time domain. Without increasing modifying the time range, the plots will not be visible in the default range.

B.2.

```
fs = 4000;
freq = 1000;
amp = 1;
ph = 0;
t = [0:1/fs:2/freq];
% generate the sinusoid
y = amp*sin((2*pi*freq*t)+ph)';
y_sum = y;
y_fft = fft(y_sum);
y_fft = y_fft(1:length(y_fft)/2);
% Plot the magnitude response,
mag = abs(y_fft);
figure(); plot(mag);
title('The magnitude response from 0 to fs/2');
```

```
Warning: Integer
```

```
operands are
required for colon
operator when used
as index.
```



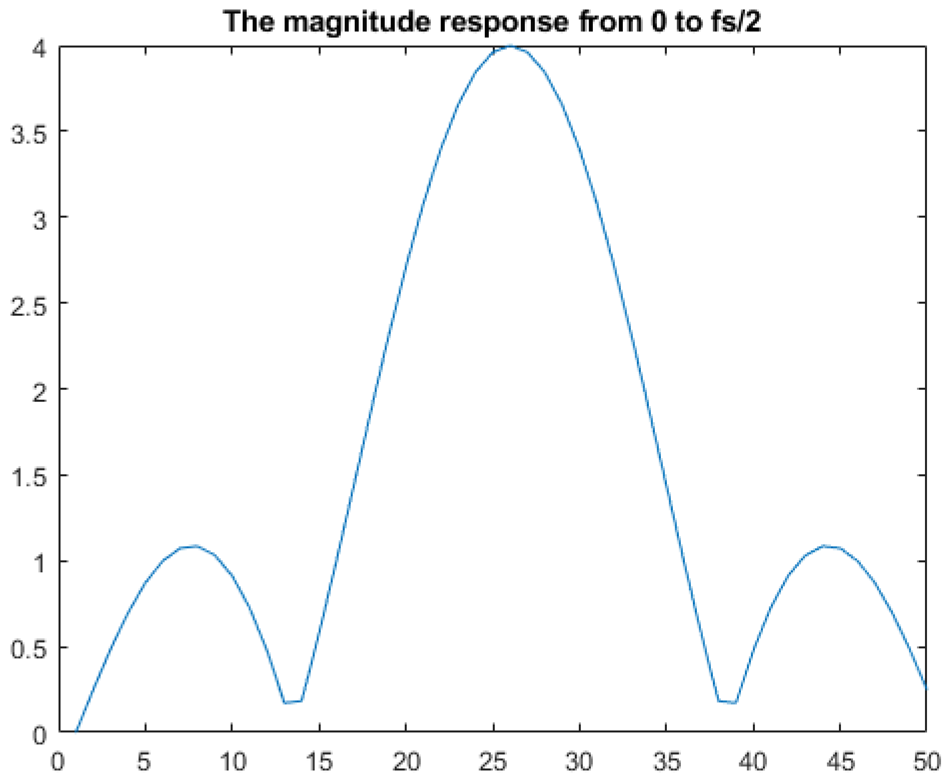
B.2.a.

As per the result of the code in this section and the fft() documatation, without specifying the sample number, function will return the fft as a vector of y_sum which is 9 in this case. As with the magnitude response, only the linear peak values of magnitude of the original plot.

B.2.b.

```
fs = 4000;
freq = 1000;
amp = 1;
ph = 0;
t = [0:1/fs:2/freq];
% generate the sinusoid
y = amp*sin((2*pi*freq*t)+ph)';
y(length(y)+1:100)=0; % 100 zero-padding
y_fft = fft(y);
y_fft = y_fft(1:length(y_fft)/2);
% Plot the magnitude response,
mag = abs(y_fft);
```

```
figure(); plot(mag);
title('The magnitude response from 0 to fs/2');
```



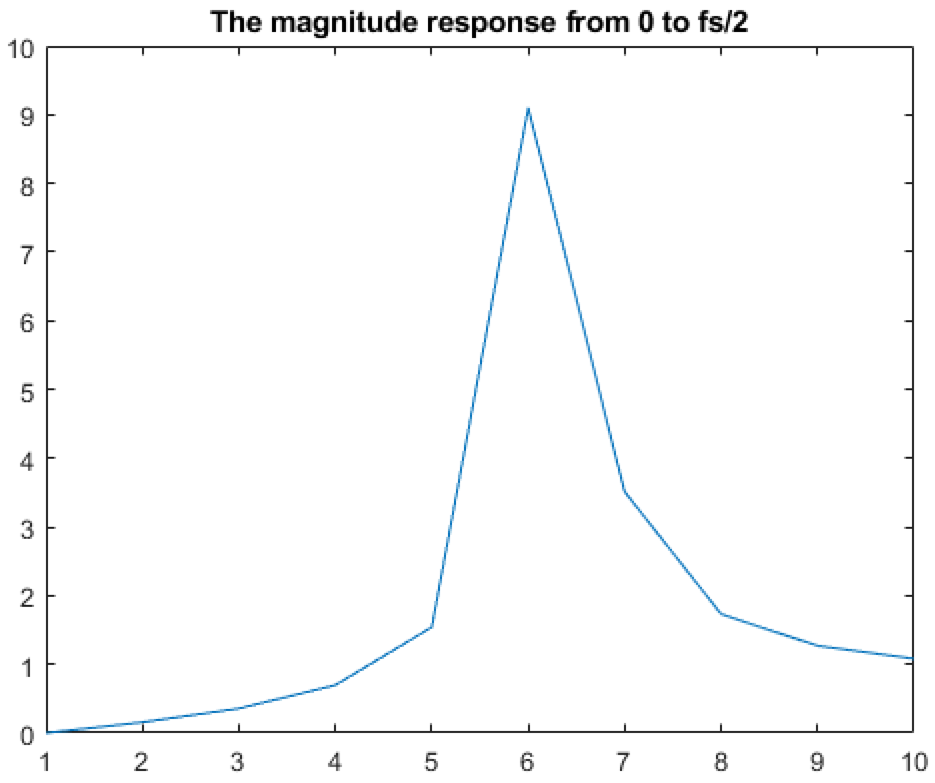
% As shown in the figure of this section, padded zeros in the DFT acts like the frequency samples in DFT thus, reverting the plot back to its original plot.

B.2.c.

```
fs = 4000;
freq = 1000;
amp = 1;
ph = 0;
t = [0:1/fs:5/freq];
% generate the sinusoid
y = amp*sin((2*pi*freq*t)+ph)';
% y(length(y)+1:100)=0; % 100 zero-padding
y_fft = fft(y);
y_fft = y_fft(1:length(y_fft)/2);
% Plot the magnitude response,
mag = abs(y_fft);
figure(); plot(mag);
title('The magnitude response from 0 to fs/2');
```

```
Warning: Integer
operands are
required for colon
```

```
operator when used
as index.
```

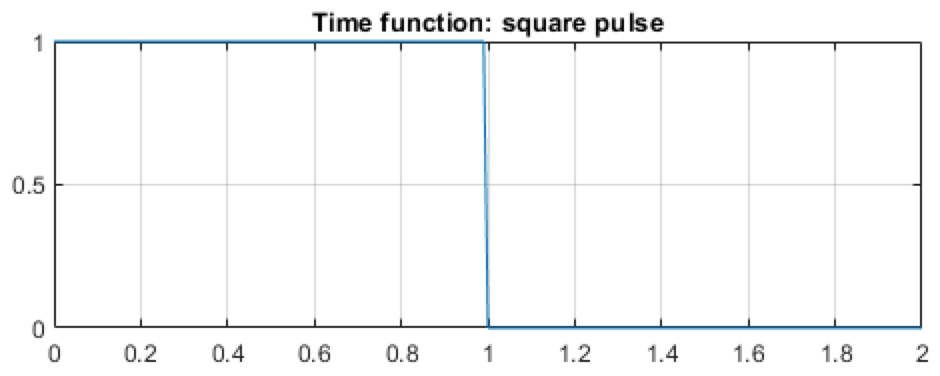


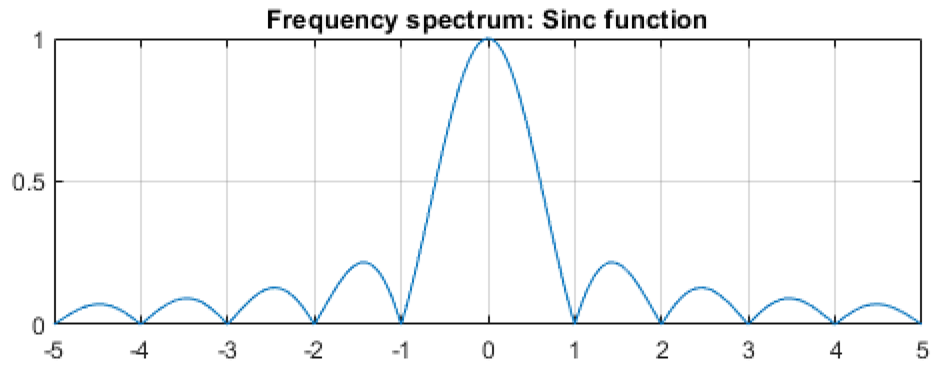
% Increasing the period means increasing the magnitude peaks of the signal furthermore, it gradually eliminate the sidelobes of the magnitude spectrum.

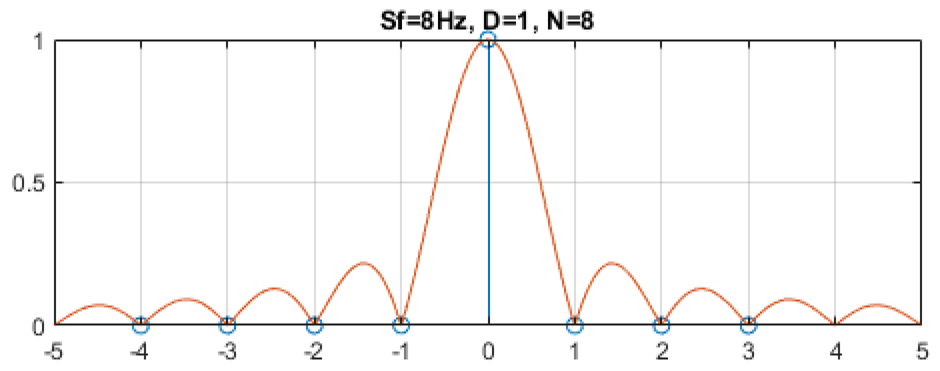
C(Cute). Graduate students

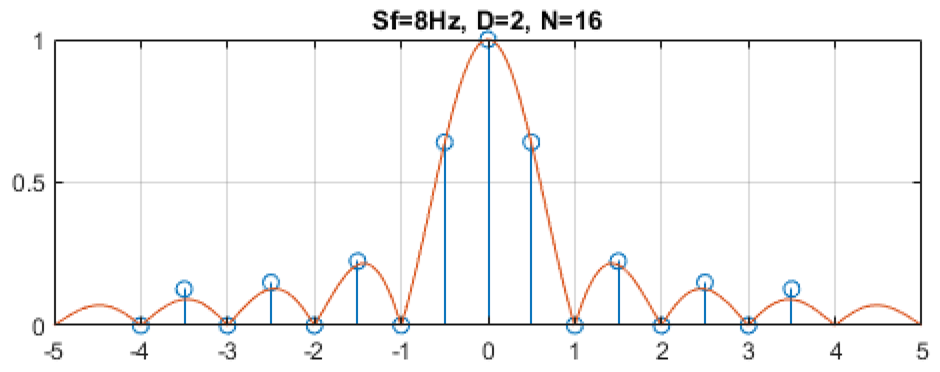
```
C1_P=2; C1_W=2; C1_t=0:0.01:C1_P;
C1_R = rectpuls(C1_t, C1_W);
C1_f=-5:0.01:5;
C1_XR=exp(-j*pi*C1_f).*sinc(C1_f);
figure(), subplot(2,1,1), plot(C1_t, C1_R);
title("Time function: square pulse"); grid on;
subplot(2,1,2), plot(C1_f,abs(C1_XR));
title("Frequency spectrum: Sinc function"); grid on;

C2_Sf=8;
C2_D=1;
C2_N=8;
C2_tn=0:1/C2_Sf:(C2_N-1)*1/C2_Sf;
C2_xn=rectpuls(C2_tn, C1_W);
C2_ya=fft(C2_xn, C2_N);
C2_ya=fftshift(C2_ya);
C2_fo=1/C2_D;
C2_fa=-(C2_N/2)*C2_fo:C2_fo:(C2_N/2-1)*C2_fo;
```

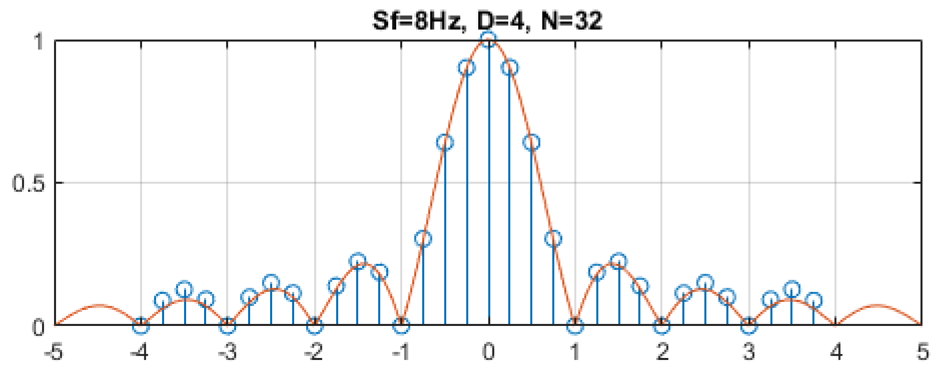
```
figure(); subplot(2,1,1); stem(C2_fa,1/C2_Sf*(abs(C2_ya)));
title('Sf=8Hz, D=1, N=8'); grid on; hold on;
plot(C1_f, abs(C1_XR)); hold off;

C3_Sf=8;
C3_D=2;
C3_N=16;
C3_tn=0:1/C3_Sf:(C3_N-1)*1/C3_Sf;
C3_xn=rectpuls(C3_tn, C1_W);
C3_ya=fft(C3_xn, C3_N);
C3_ya=fftshift(C3_ya);
C3_fo=1/C3_D;
C3_fa=-(C3_N/2)*C3_fo:C3_fo:(C3_N/2-1)*C3_fo;
subplot(2,1,2); stem(C3_fa,1/C3_Sf*(abs(C3_ya)));
title('Sf=8Hz, D=2, N=16'); grid on; hold on;
plot(C1_f, abs(C1_XR)); hold off;

C4_Sf=8;
C4_D=4;
C4_N=32;
C4_tn=0:1/C4_Sf:(C4_N-1)*1/C4_Sf;
C4_xn=rectpuls(C4_tn, C1_W);
C4_ya=fft(C4_xn, C4_N);
C4_ya=fftshift(C4_ya);
C4_fo=1/C4_D;
C4_fa=-(C4_N/2)*C4_fo:C4_fo:(C4_N/2-1)*C4_fo;
figure();subplot(2,1,1); stem(C4_fa,1/C4_Sf*(abs(C4_ya)));
title('Sf=8Hz, D=4, N=32'); grid on; hold on;
plot(C1_f, abs(C1_XR)); hold off;

C5_R = rectpuls(C1_t, C1_W);
C5_f=-8:0.01:8;
C5_XR=exp(-j*pi*C5_f).*sinc(C5_f);
C6_Sf=16;
C6_D=4;
C6_N=64;
C6_tn=0:1/C6_Sf:(C6_N-1)*1/C6_Sf;
C6_xn=rectpuls(C6_tn, C1_W);
C6_ya=fft(C6_xn, C6_N);
C6_ya=fftshift(C6_ya);
C6_fo=1/C6_D;
C6_fa=-(C6_N/2)*C6_fo:C6_fo:(C6_N/2-1)*C6_fo;
subplot(2,1,2); stem(C6_fa,1/C6_Sf*(abs(C6_ya)));
title('Sf=16Hz, D=4, N=64'); grid on; hold on;
plot(C5_f, abs(C5_XR)); hold off;
```

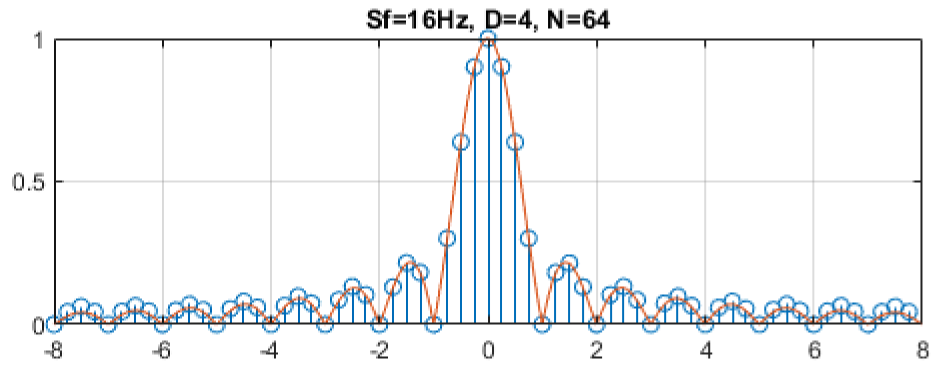

Time function: square pulse
1
0.5
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Frequency spectrum: Sinc function
1
0.5
0
-5
-4
-3
-2
-1
0
1
2
3
4
5


Sf=8Hz, D=1, N=8
1
0.5
0
-5
-4
-3
-2
-1
0
1
2
3
4
5


Sf=8Hz, D=2, N=16
1
0.5
0
-5
-4
-3
-2
-1
0
1
2
3
4
5





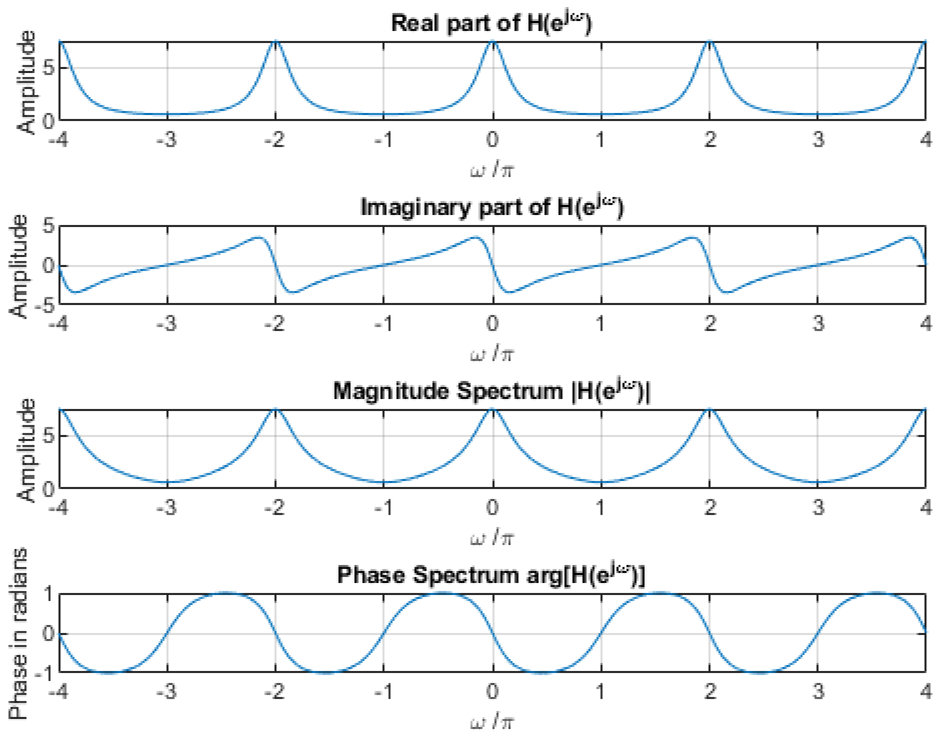
Properties of DFT

A. Computation of DFT

Original Code with single figure display modification.

```
clf;
w = -4*pi:8*pi/511:4*pi;
num = [2 1];den = [1 -0.6];
h = freqz(num, den, w);
subplot(4,1,1)
plot(w/pi,real(h));grid
title('Real part of H(e^{j\omega})')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,2)
plot(w/pi,imag(h));grid
title('Imaginary part of H(e^{j\omega})')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,3)
plot(w/pi,abs(h));grid
title('Magnitude Spectrum |H(e^{j\omega})|')
xlabel('\omega /\pi');
ylabel('Amplitude');
```

```
subplot(4,1,4)
plot(w/pi,angle(h));grid
title('Phase Spectrum arg[H(e^{j\omega})]')
xlabel('\omega /\pi');
ylabel('Phase in radians');
```



A.1. Compare and differentiate above code output and freqz(num,den).

The signal in code with a w parameter in freqz() plots every two interval in x-space (8*pi/511) while the code that does not have a w parameter plots in the entire -4:4 range.

A.2. Symmetries in real and imag parts of DFT.

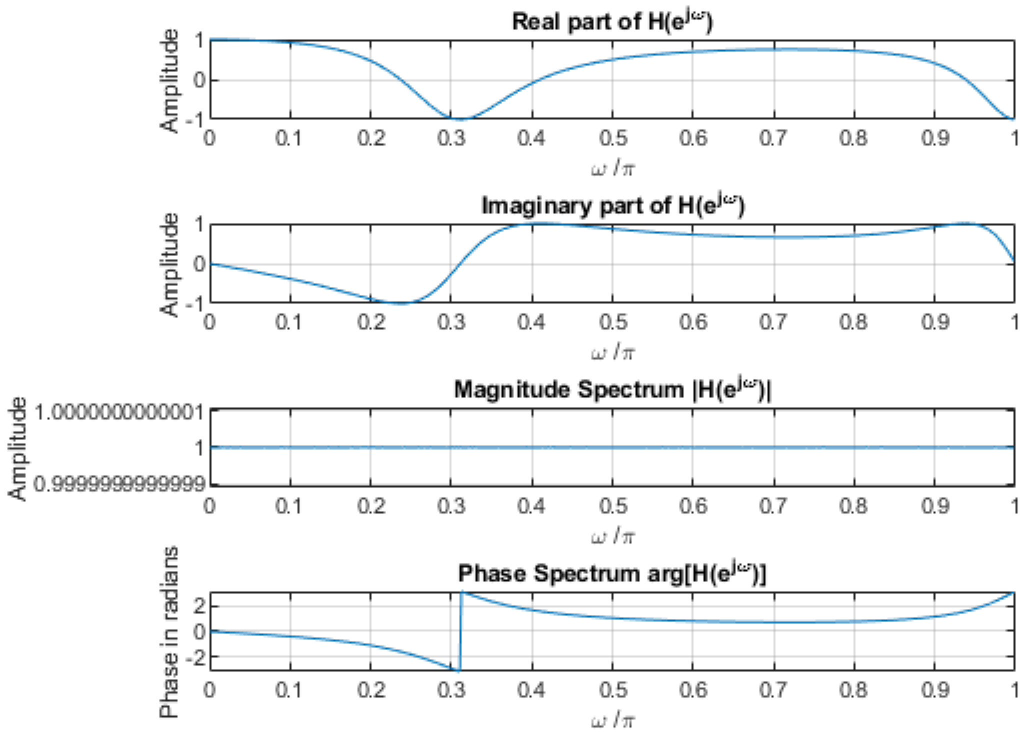
The two graphs are symmetric as they follow the same function signature only that the first one plots every 8*pi/511 in -4:4 x-space while the latter plots in the entire x-space of -4:4.

The DTFT is a periodic function of omega with a period of 2pi. All of the plots are 2pi periodic with the real part and magnitude are even symmetric while the imaginary part of phase are odd symmetric.

A.3. Modify the above code and satisfy the fxn below...

$$U(e^{j\omega}) = \frac{0.7 - 0.5e^{j\omega} + 0.3e^{-j2\omega} + e^{-j3\omega}}{1 + 0.3e^{-j\omega} - 0.5^{-j2\omega} + 0.7e^{-j3\omega}}$$

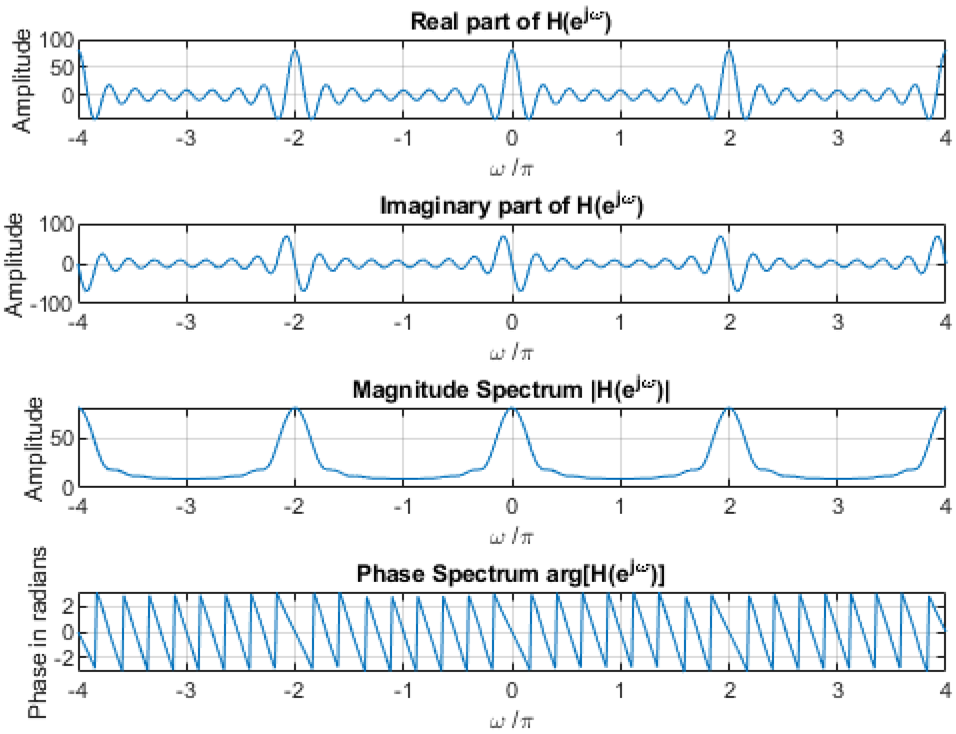
```
A3_N = 512;
A3_num = [0.7 -0.5 0.3 1]; A3_den = [1 0.3 -0.5 0.7];
[A3_h, A3_w] = freqz(A3_num, A3_den, A3_N);
figure()
subplot(4,1,1)
plot(A3_w/pi,real(A3_h));grid
title('Real part of H(e^{j\omega})')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,2)
plot(A3_w/pi,imag(A3_h));grid
title('Imaginary part of H(e^{j\omega})')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,3)
plot(A3_w/pi,abs(A3_h));grid
title('Magnitude Spectrum |H(e^{j\omega})|')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,4)
plot(A3_w/pi,angle(A3_h));grid
title('Phase Spectrum arg[H(e^{j\omega})]')
xlabel('\omega /\pi');
ylabel('Phase in radians');
```

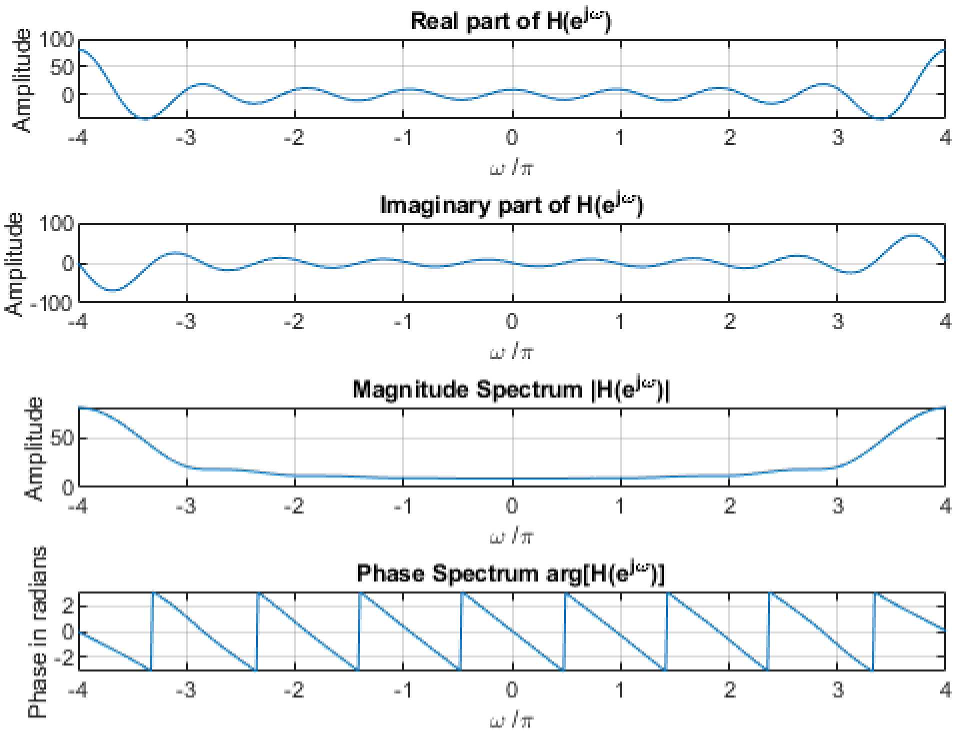


A.4. Modify the code and evaluate the sequence. Use freqz and fft. Compare.

```
A4_w = -4*pi:8*pi/511:4*pi;
A4_num = [1 3 5 7 9 11 13 15 17]; A4_den = 1;
A4_h = freqz(A4_num, A4_den, A4_w);
A4_hf = fft(A4_num, A3_N);
% freqz the way
figure()
subplot(4,1,1)
plot(A4_w/pi,real(A4_h));grid
title('Real part of H(e^{j\omega})')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,2)
plot(A4_w/pi,imag(A4_h));grid
title('Imaginary part of H(e^{j\omega})')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,3)
plot(A4_w/pi,abs(A4_h));grid
title('Magnitude Spectrum |H(e^{j\omega})|')
xlabel('\omega /\pi');
ylabel('Amplitude');
```

```
subplot(4,1,4)
plot(A4_w/pi,angle(A4_h));grid
title('Phase Spectrum arg[H(e^{j\omega})]')
xlabel('\omega /\pi');
ylabel('Phase in radians');
% fft
figure()
subplot(4,1,1)
plot(A4_w/pi,real(A4_hf));grid
title('Real part of H(e^{j\omega})')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,2)
plot(A4_w/pi,imag(A4_hf));grid
title('Imaginary part of H(e^{j\omega})')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,3)
plot(A4_w/pi,abs(A4_hf));grid
title('Magnitude Spectrum |H(e^{j\omega})|')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(4,1,4)
plot(A4_w/pi,angle(A4_hf));grid
title('Phase Spectrum arg[H(e^{j\omega})]')
xlabel('\omega /\pi');
ylabel('Phase in radians');
```

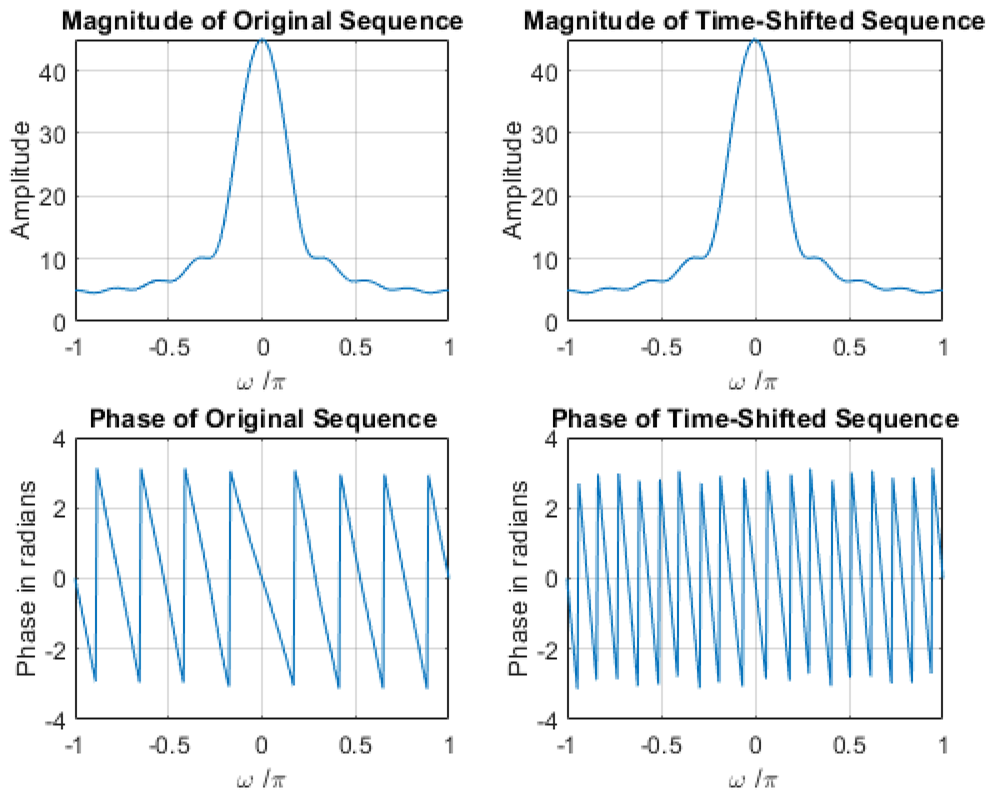

Real part of H(e^jω)
Amplitude
ω /π
Imaginary part of H(e^jω)
Magnitude Spectrum |H(e^jω)|
Phase Spectrum arg[H(e^jω)]
Phase in radians


Real part of H(e^jω)
Amplitude
ω /π
Imaginary part of H(e^jω)
Magnitude Spectrum |H(e^jω)|
Phase Spectrum arg[H(e^jω)]
Phase in radians

Although freqz() and fft() does the same for DFT only that freqz compute the phase and magnitude periorically whilst fft() does the same thing but of one period alone.

B. Time-shift property of DFT

```
clf;
B_w = -pi:2*pi/255:pi; B_D = 10;
B_num = [1 2 3 4 5 6 7 8 9];
B_h1 = freqz(B_num, 1, B_w);
B_h2 = freqz([zeros(1,B_D) B_num], 1, B_w);
subplot(2,2,1)
plot(B_w/pi,abs(B_h1));grid
title('Magnitude of Original Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(2,2,2)
plot(B_w/pi,abs(B_h2));grid
title('Magnitude of Time-Shifted Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(2,2,3)
plot(B_w/pi,angle(B_h1));grid
title('Phase of Original Sequence')
xlabel('\omega /\pi');
ylabel('Phase in radians');
subplot(2,2,4)
plot(B_w/pi,angle(B_h2));grid
title('Phase of Time-Shifted Sequence')
xlabel('\omega /\pi');
ylabel('Phase in radians');
```

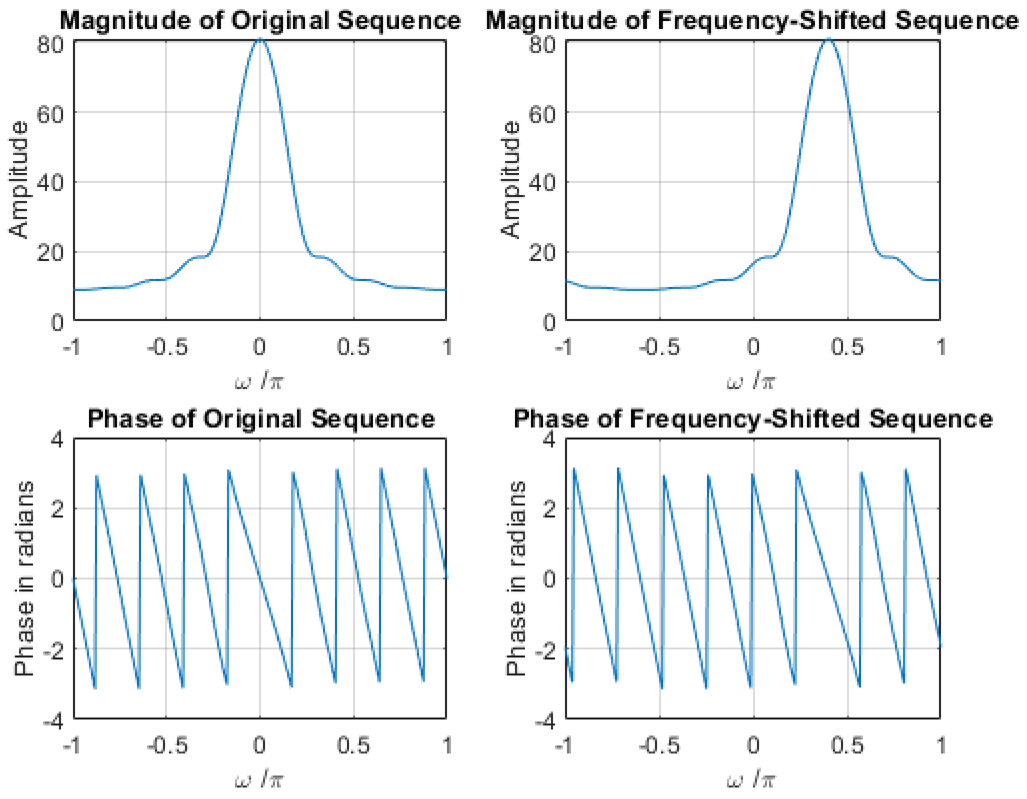


Based from the generated plots, the magnitude spectrun is not affected by the time-shift, on the phase spectrum instead. The phase function becomes steeper and more negative. B_D is the parameter that control the time shift in the program. Increasing the length makes the magnitude narrower and phase steeper as more delay are being added between the I/O system. The larger the delay (B_D), the more negative slope is added to the spectrum.

C. Frequency-shift property of DFT

```
clf;
C_w = -pi:2*pi/255:pi; C_wo = 0.4*pi;
C_num1 = [1 3 5 7 9 11 13 15 17];
C_L = length(C_num1);
C_h1 = freqz(C_num1, 1, C_w);
C_n = 0:C_L-1;
C_num2 = exp(C_wo*i*C_n).*C_num1;
C_h2 = freqz(C_num2, 1, C_w);
subplot(2,2,1)
plot(C_w/pi,abs(C_h1));grid
title('Magnitude of Original Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(2,2,2)
plot(C_w/pi,abs(C_h2));grid
title('Magnitude of Frequency-Shifted Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
```

```
subplot(2,2,3)
plot(C_w/pi,angle(C_h1));grid
title('Phase of Original Sequence')
xlabel('\omega /\pi');
ylabel('Phase in radians');
subplot(2,2,4)
plot(C_w/pi,angle(C_h2));grid
title('Phase of Frequency-Shifted Sequence')
xlabel('\omega /\pi');
ylabel('Phase in radians');
```

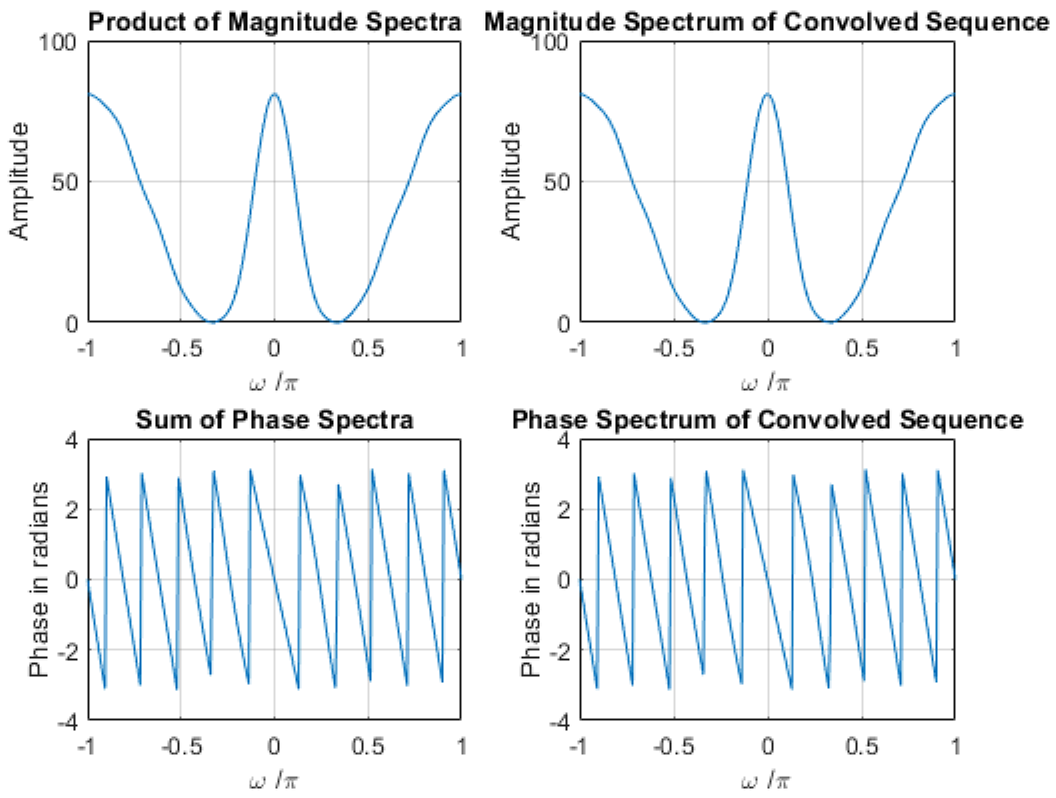


Based from the generated plots, Both the phase and magnitude are shifted to the right by wo=0.4pi. The parameter wo is the sole parameter that controls the amount of shift.

D. Convolution property of DFT

```
clf;
D_w = -pi:2*pi/255:pi;
D_x1 = [1 3 5 7 9 11 13 15 17];
D_x2 = [1 -2 3 -2 1];
D_y = conv(D_x1,D_x2);
D_h1 = freqz(D_x1, 1, D_w);
D_h2 = freqz(D_x2, 1, D_w);
D_hp = D_h1.*D_h2;
D_h3 = freqz(D_y,1,D_w);
subplot(2,2,1)
```

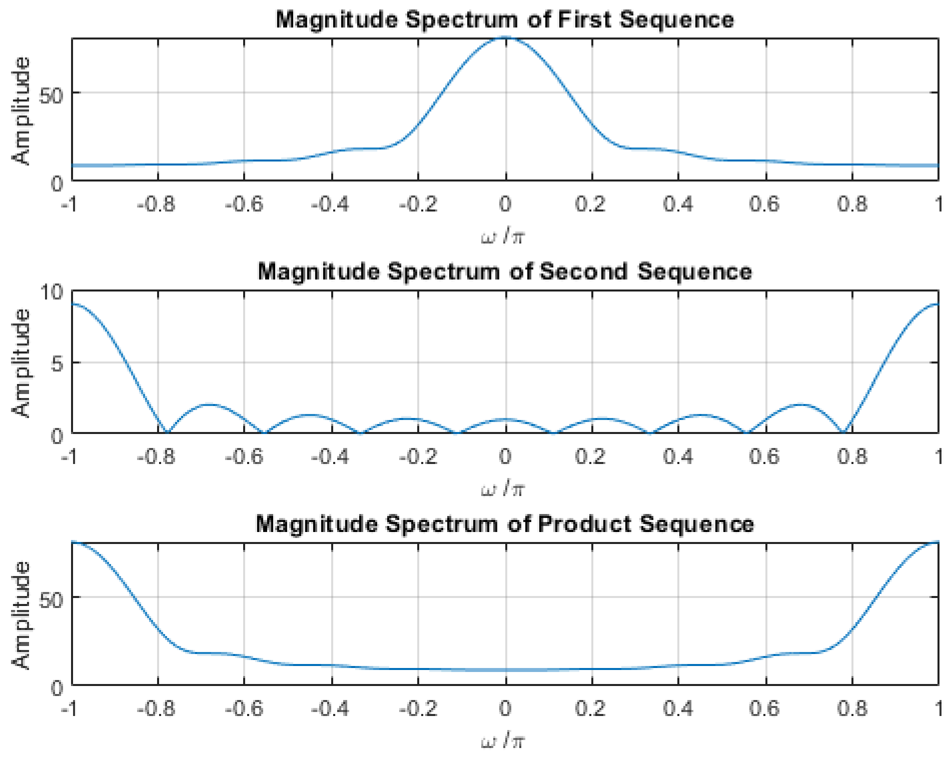
```
plot(D_w/pi,abs(D_hp));grid
title('Product of Magnitude Spectra')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(2,2,2)
plot(D_w/pi,abs(D_h3));grid
title('Magnitude Spectrum of Convolved Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(2,2,3)
plot(D_w/pi,angle(D_hp));grid
title('Sum of Phase Spectra')
xlabel('\omega /\pi');
ylabel('Phase in radians');
subplot(2,2,4)
plot(D_w/pi,angle(D_h3));grid
title('Phase Spectrum of Convolved Sequence')
xlabel('\omega /\pi');
ylabel('Phase in radians');
```



Based from the generted plots, the magnitude and phase from pointwise multiplication of two DTFT of the original sequences are equivalent to thos eobtained from time-domain convolution of two sequences hence, the convolution property of the DTFT.

E. Modulation property

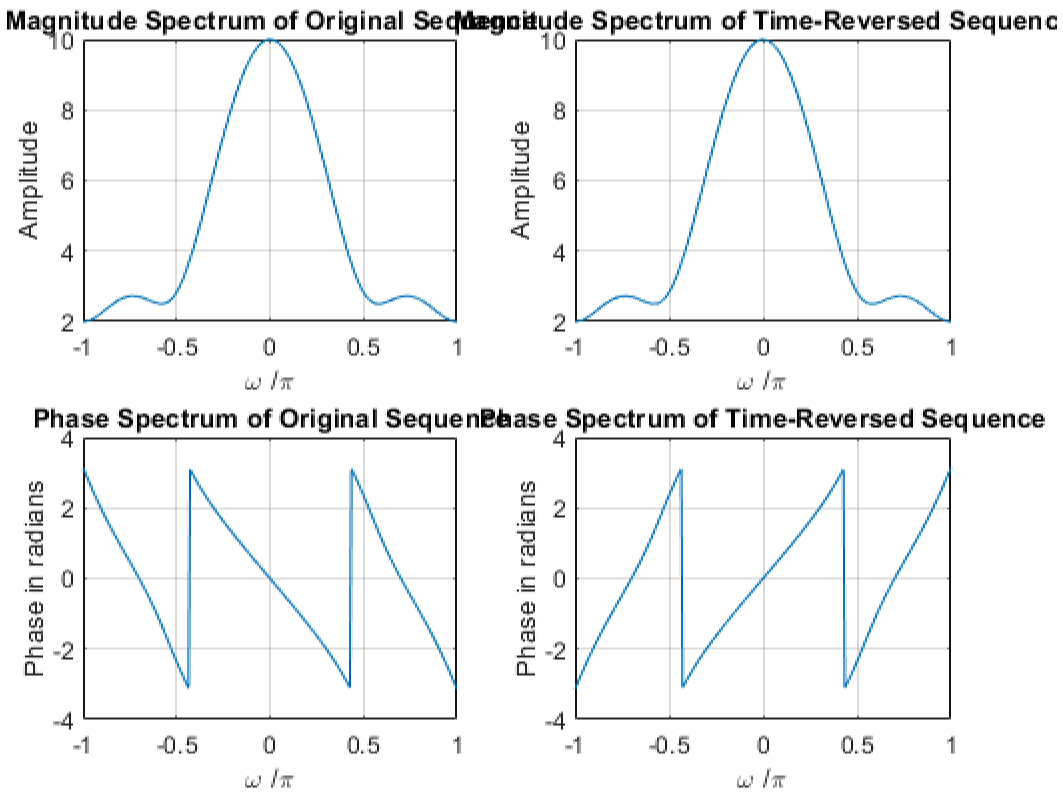
```
clf;
E_w = -pi:2*pi/255:pi;
E_x1 = [1 3 5 7 9 11 13 15 17];
E_x2 = [1 -1 1 -1 1 -1 1 -1 1];
E_y = E_x1.*E_x2;
E_h1 = freqz(E_x1, 1, E_w);
E_h2 = freqz(E_x2, 1, E_w);
E_h3 = freqz(E_y,1,E_w);
subplot(3,1,1)
plot(E_w/pi,abs(E_h1));grid
title('Magnitude Spectrum of First Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(3,1,2)
plot(E_w/pi,abs(E_h2));grid
title('Magnitude Spectrum of Second Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(3,1,3)
plot(E_w/pi,abs(E_h3));grid
title('Magnitude Spectrum of Product Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
```



Based from the generated plots, the DTFT of the sequential prouct of y is 0.5pi times the convolution of E_x1 and E_x2 sequences. The low-pass mainlobe of the DTFT of E_x1 combines with the high-pass mainlobe of DTFT E_x2 to produce a high-pass mainlobe centered at (plus/minus)pi in the magnitude of the product signal. The low-pass mainlobe of the DTFT of E_x1 combines with the low-pass sidelobes of DTFT E_x2 to produce a low-pass smooth rgion of lower gain centered at DC in the magnitude of product signal.

F. Time-reversal property

```
clf;
F_w = -pi:2*pi/255:pi;
F_num = [1 2 3 4];
F_L = length(F_num)-1;
F_h1 = freqz(F_num, 1, F_w);
F_h2 = freqz(fliplr(F_num), 1, F_w);
F_h3 = exp(F_w*F_L*i).*F_h2;
subplot(2,2,1)
plot(F_w/pi,abs(F_h1));grid
title('Magnitude Spectrum of Original Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(2,2,2)
plot(F_w/pi,abs(F_h3));grid
title('Magnitude Spectrum of Time-Reversed Sequence')
xlabel('\omega /\pi');
ylabel('Amplitude');
subplot(2,2,3)
plot(F_w/pi,angle(F_h1));grid
title('Phase Spectrum of Original Sequence')
xlabel('\omega /\pi');
ylabel('Phase in radians');
subplot(2,2,4)
plot(F_w/pi,angle(F_h3));grid
title('Phase Spectrum of Time-Reversed Sequence')
xlabel('\omega /\pi');
ylabel('Phase in radians');
```



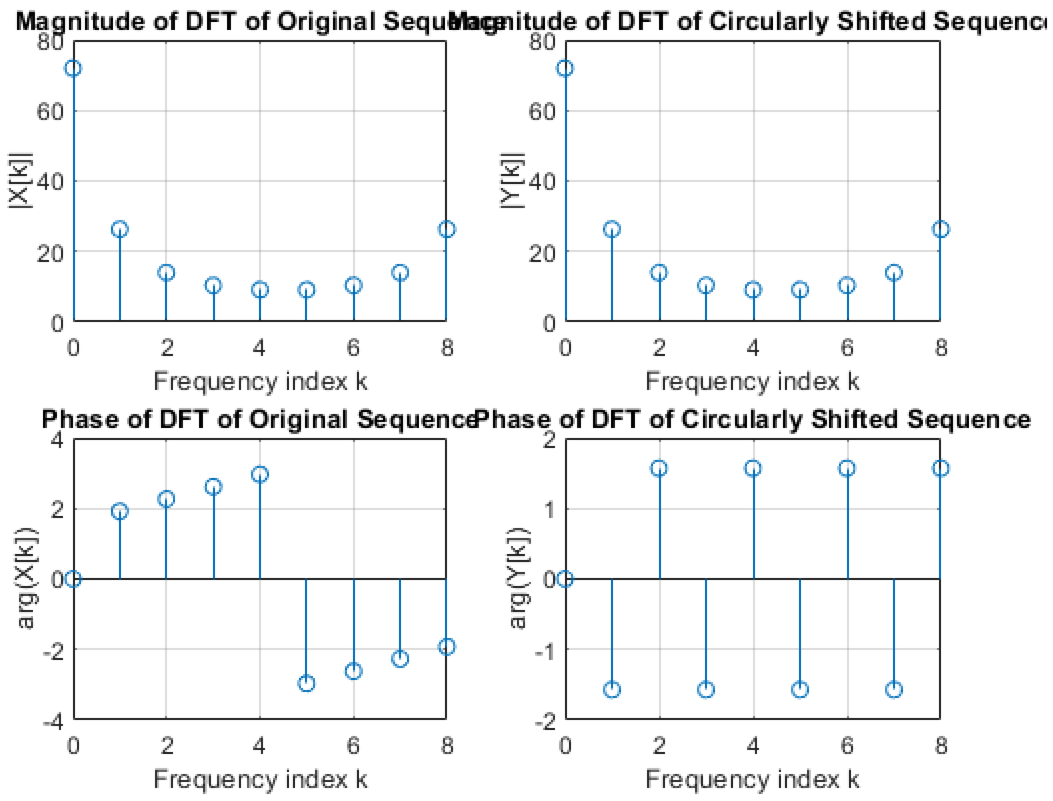
Based from the generated plots, the original ramp sequence is nonzero when n greater than or equal to zero and n is less than or equal to three. A new sequence is formed using the function fliplr which contains the samples of the original ramp sequence in reversed time order. Although the new sequence is nonzero still in the said range before, the time reversed ramp must be nonzero when n is greater than or equal to -3 and less than or equal to zero. This property required left shift in time and ccompished in frequency domain using time-shifting property of DTFT. The steps are 1) freqz() is called to set F_h2 equal to DTFT of the new sequence obtained from calling fliplr on t original ramp sequence. 2) F_h3 is set equal to the DTFT of the time reveresed ramp by multiplying F_h2 with a linear phase term to implement left shifting in time domain.

G for Graduates (1). Circular-shifting property

circshift is defined by the matlab function circshift in matlab2020a.

```
clf;
G1_x = [0 2 4 6 8 10 12 14 16];
G1_N = length(G1_x)-1; G1_n = 0:G1_N;
G1_y = circshift(G1_x,5);
G1_XF = fft(G1_x);
G1_YF = fft(G1_y);
subplot(2,2,1);
stem(G1_n,abs(G1_XF));grid;
title('Magnitude of DFT of Original Sequence');
xlabel('Frequency index k');
ylabel('|X[k]|');
subplot(2,2,2);
```

```
stem(G1_n,abs(G1_YF));grid;
title('Magnitude of DFT of Circularly Shifted Sequence');
xlabel('Frequency index k');
ylabel('|Y[k]|');
subplot(2,2,3);
stem(G1_n,angle(G1_XF));grid;
title('Phase of DFT of Original Sequence');
xlabel('Frequency index k');
ylabel('arg(X[k])');
subplot(2,2,4);
stem(G1_n,angle(G1_YF));grid;
title('Phase of DFT of Circularly Shifted Sequence');
xlabel('Frequency index k');
ylabel('arg(Y[k])');
```



circshift is a function in matlab that operates as follows: the input sequence G1_x is circularly shifted by G1_M position. If G1_M > 0, then circshift remove the leftmost G1_M elements from the vector G1_x and appends them on the right side of the remaining elements to obtain the circularly shifted sequence. If G1_M < 0, circhisft first complements G1_M by the length of G1_x. Based from the generated plots, the length of the sequence G1_n=8 and the time shift is left-advanced by five samples as established by the phase term expression W{N}{kn0}=W{N}{-k5}=exp(jk10pi/8)=exp(jk5pi/4) which increses the slope of phase, gradually. There are five cuts in phase of the shifted signal and one in the original signal.

G for Graduates (2). Circular-convolution property

circonv is defined by the matlab function cconv in matlab2020a.

```
G2_g1 = [1 2 3 4 5]; G2_g2 = [2 2 0 1 1];
G2_g1e = [G2_g1 zeros(1,length(G2_g2)-1)];
G2_g2e = [G2_g2 zeros(1,length(G2_g1)-1)];
G2_ylin = cconv(G2_g1e,G2_g2e); % decided to use cconv instead for
 quick run
disp('Linear convolution via circular convolution = ');
disp(G2_ylin(1:9));
G2_y = conv(G2_g1, G2_g2);
disp('Direct linear convolution = ');
disp(G2_y);
```

```
Linear convolution via circular convolution =
  Columns 1 through 7

    2.0000    6.0000   10.0000   15.0000   21.0000   15.0000    7.0000

  Columns 8 through 9

    9.0000    5.0000

Direct linear convolution =
     2     6    10    15    21    15     7     9     5
```

circonv or cconv is a matlab function that require two inputs G2_g1e and G2_g2e of equal length. Function cconv or circonv works like a periodic extension of G2_g2e. Let G2_g2ep be the infinite length periodic extension of G2_g2e. The elements from 1 to L of the output vector G2_ylin are obtained by taking the inner product between G2_g1e and a length L vector sh obtained by circularly shifting right the time reversed vector G2_g2etr. The output sample y[n] with n is greater than or equal to 1 and less than or equal to L, the amount of the right circulat shift is n-1 postion. Based from the generated results, zero padding onto the match length made possible for the implementation of linear convolution using circular convolution.

Published with MATLAB® R2020a