BIALYSTOK UNIVERSITY OF TECHNOLOGY

SIGNAL PROCESSING

TASK 1

TUTOR:

Dr inż. MAREK PARFIENIUK

STUDENTS:

AHMET TOREHAN HAYTOGLU

HARUN GULEC

TASK 1

1.1

Learn Matlab functions for reading, saving, and playing sounds: wavread, wavwrite, sound. Learn how mono- and stereophonic sounds are represented in Matlab as matrices. Learn Matlab functions for 1D plotting: plot, stem as well as functions for labeling axes, setting scale, line style etc.

Wavread: Read Microsoft WAVE (".wav") sound file.

[Y,FS,NBITS] = wavread(FILE); Wavread read FILE and return Y that is sample of wav file. And also it's possible to return hertz of sample (FS) and number of bits(NBITS).

Wavwrite: Write Microsoft WAVE (".wav") sound file.

wavwrite(Y,FS,NBITS,WAVEFILE) ; Wavwrite write Y to WAVEFILE with FS hertz and NBITS number of bits will be written. Default FS is 8000 Hz and default NBITS is 16 bit.

Sound: Play vector as sound.

sound(Y,FS,BITS) ; Sound play vector as sound. But vector must be between -1.0 and 1.0. If you want to play stereo music you should use N-by-2 matrice. Default FS is 8192 Hertz and NBITS is 8 or 16 it's up to platform.

In Matlab stereophonic sound represents as N-by-2 matrix and monophonic sounds represents as N-by-1 matrix.

Plot: Draw linear plot.

plot(X,Y,S); Draw vector X versus vector Y and S is style of line.

Stem: Discrete points of plot.

Stem(X,Y); plots the data sequence Y at the values specified in X.

title('Title of Graph'): Specifies title of the plot.

xlabel('Label of X axis'): Specifies label for x axis of plot.

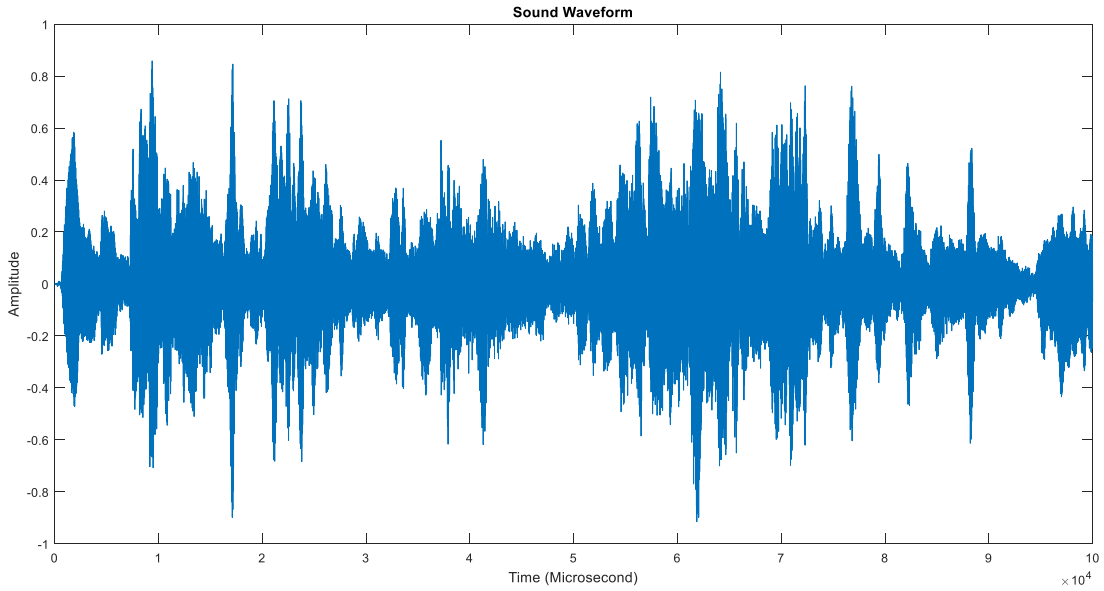
ylabel('Label of Y axis'): Specifies label for y axis of plot.

1.2

Develop a script that loads a signal from a .wav file, plots its waveform, and plays the signal as sound.

Our script for read, sound a wavefile and draw its waveform;

```
[y,Fs]=audioread('tone.wav'); %read the tone and assign data, frequency, bits
a=y(1:100000); % to get shorten data
plot(a)%plot waveform
xlabel('Time (Mikrosecond)')
ylabel('Amplitude')
title('Sound Waveform')
sound(a,Fs)%sound music
```



1.3

How does increasing/decreasing of sampling frequency affect sound You hear?

We can use sound function with 2 parameters. One of them sound data other one sampling frequency.

```
clear all
[a,Fs]=wavread('tone.wav'); %Read the tone and separate sound to frequency
signal and bits
b=a(1:100000); %cut the sound to make smaller and assign to b
sound(b,Fs)
pause(2); %Program will wait for 2 seconds between the commands
sound(b,Fs/2)%Playing the sound after the frequency drops to Fs/2.
pause(2); %Program will wait for 2 seconds between the commands
sound(b,Fs*2) %Playing the sound after the frequency increased to Fs*2
```

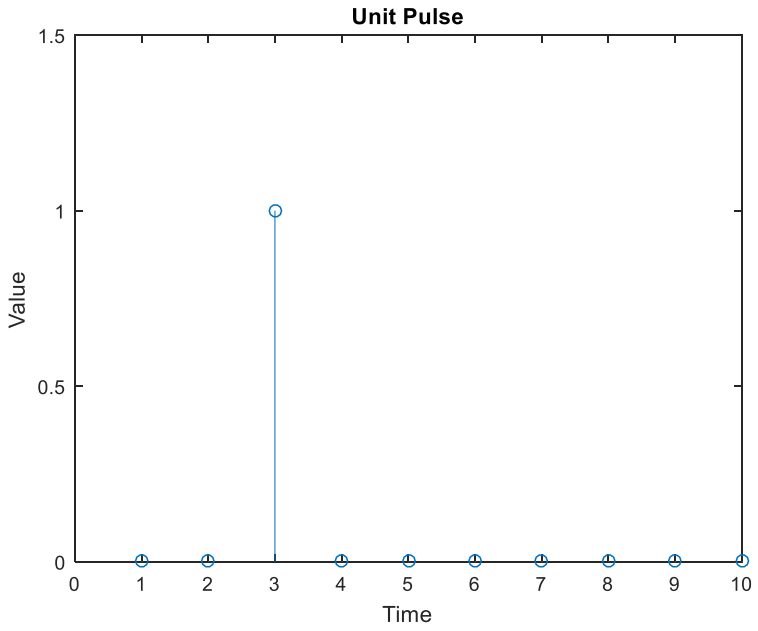
1.4

Develop a function that generates an unit pulse. Develop a function that generates a unit step signal. The functions should have the following parameters: the number of samples and sample index at which pulse/step occurs. Develop a script that demonstrates the function.

Our unit pulse function :

```
function x = unit_pulse( L,P )%define a function
    x=zeros(1,L); %Create a matrix just contain zero until to L
    x(P)=1; % Put 1 to P index of matrix
end
```

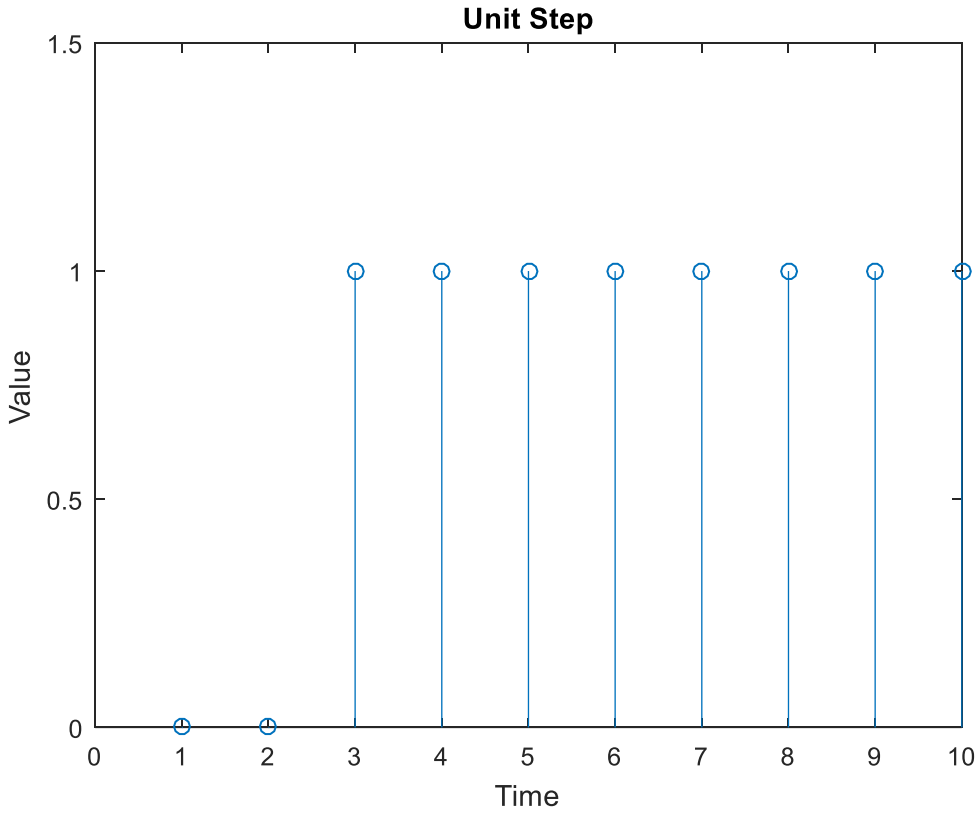
```
%to understand how working code and show graphic
x=unit_pulse(10,3);
stem(x)
axis([0 10 0 1.5])
title('Unit Pulse')
xlabel('Time')
ylabel('Value')
```



Our unit step function :

```
function x = unit_step ( L,P )%define a function
    x=zeros(1,L); %Create a matrix just contain zero until to L
    for ii=P:L
        x(ii)=1; % All points after P indis will be 1
    end
end
```

```
%to understand how working code and show graphic
x=unit_step(10,3);
stem(x)
axis([0 10 0 1.5])
title('Unit Step')
xlabel('Time')
ylabel('Value')
```



1.5

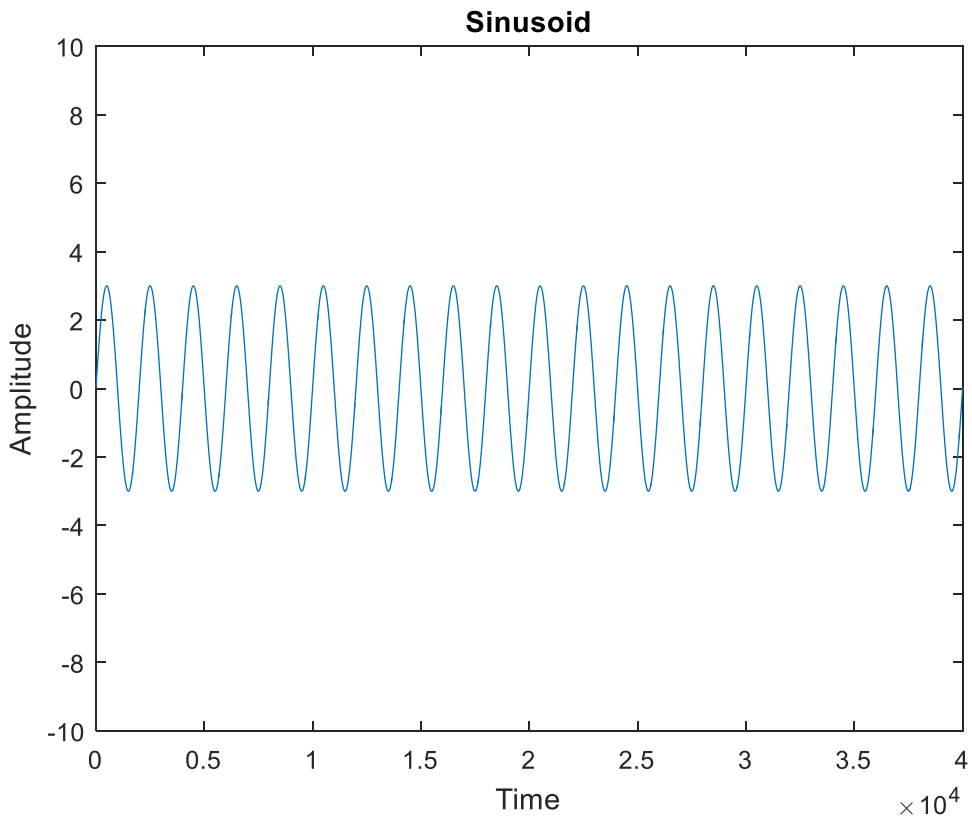
Develop a function that, for a given sampling frequency f_S, generates samples of the sinusoid with a given magnitude A, frequency f_0 [Hz], and phase shift φ [rad]. Additional parameters are the start and stop moments of signal sampling, tstart and tstop. Think on connection between time and sample index n.

$$x(n) = A\ sin(2\pi f_0 n + \varphi)$$

Develop a script that demonstrates the function.

```
function x= sinusoid (fs , f,a, t1, t2, q)

    t= t1:(1/fs ):t2 ; %define t between t1 and t2 at intervals of fs

    x= a*sin(2*pi*f*t+q);%define sinusoidal(x(n) = A*sin(2*pi*fs*n+Q))function

end
```

```
clear all %clear everything
fs=10000; %value of sample frequency
f=5; %value of frequency
a=3; %value of amplitude
t1=0;%start value of function
t2=5;%end value of function
q=0; %phase difference value
plotx = sinusoid(fs , f , a , t1 , t2 , q ); %call sinusoid function and assaign
plot(plotx) %to draw plotx
axis([0, 40000 , -10 , 10]); %describe axis range
xlabel('Time')
ylabel('Amplitude')
title('Sinusoid')
```



1.6

Generate several sinusoidal signals with various frequencies, and play them using the sound function. How does modifying the magnitude and phase shift of the signal affect sound You hear?

```
clear all

fs=1000; %value of sampling frequency

f = 5; %value of frequency

a=5; %value of amplitude

t1=0;%start value of function

t2=2;%end value of function

q=0; %phase difference value

x=sinusoid(fs , f-150 ,a , t1 , t2 , q );%Assign sinusoid to x with decreasing
150 frequency
sound(x) %play the base sound
pause(2) %wait 2 seconds

x=sinusoid(fs , f+100 , a , t1 , t2 , q );%Assign sinusoid to x with increasing
100 frequency
sound(x) %play the base sound
pause(2) %wait 2 seconds

x=sinusoid(fs , f , a , t1 , t2 , q );%Assign sinusoid to x with baseline values
sound(x) %play the base sound
pause(2) %wait 2 seconds


x=sinusoid(fs , f , a , t1 , t2 , q+150 );%Assign sunisoid to x with increasing
phase value 150
sound(x) %play the base sound
pause(2) %wait 2 seconds

x=sinusoid(fs , f , a+150 , t1 , t2 , q);%Assign sunisoid to x with increasing
amplitude value 150
sound(x) %play the base sound
```

1.7

Table 1 describes a signal whose fragments are sinusoids with various frequencies and time extents. Generate this signal, tuning the ΔT and f_{ref} parameters so as to hear a melody when playing samples as sound. For example ΔT = 3s i f_{ref} = 1 kHz.

Table 1: Dane

No. of fragment	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Time extent[a]	8	8	8	8	8	8	16	16	4	8	8	8	8	8	8	16	16	4
Frequency[b]	1	0	0	0.5	-0.5	-0.5	-1	0	1	1	0	0	0.5	-0.5	-0.5	-1	0	-1

[a] ΔT/value from the table ;
[b] $f_{ref}(1 + 0.2$ value from the table)

Do not play the signal in the chunk-by-chunk manner. Play the vector obtained by concatenating chunks.

```
clear all
fs=1000; %value of frequency
a=1; %value of amplitude
t1=0;%start value of function
t2=3;%end value of function
q=0; %phase difference value
freq=[200,0,0,100,-100,-100,-200,0,200,200,0,0,100,-100,-100,-200,0,-200];
timeEx=[8,8,8,8,8,8,16,16,4,8,8,8,8,8,8,16,16,4];
x=[];
for ii=1:length(freq)
    x=[x sinusoid(fs, 1000+freq(ii), a , t1 , t2/timeEx(ii) , q )];
end
sound(x)%Play song
```

1.8

Learn Matlab functions for reading, saving, and showing images: imread, imwrite, imagesc, colormap. Learn how images are represented as matrices Matlab.

imread : Read image from graphics file.

A = imread(FILENAME,FMT) reads a grayscale or color image from the file specified by the string FILENAME. If the file is not in the current directory, or in a directory on the MATLAB path, specify the full pathname.

imwrite : Write image to graphics file.

imwrite(A,FILENAME,FMT) writes the image A to the file specified by FILENAME in the format specified by FMT.

imagesc : Scale data and display as image.

imagesc(...) is the same as IMAGE(...) except the data is scaled to use the full colormap imagesc(...,CLIM) where CLIM = [CLOW CHIGH] can specify the scaling.

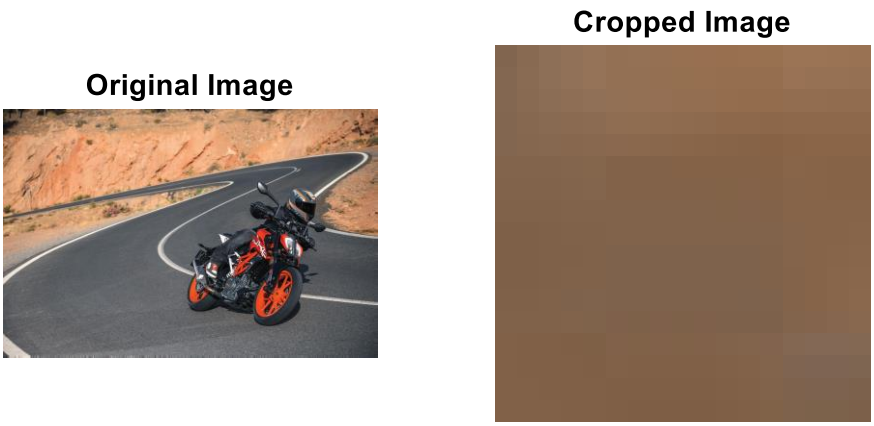
colormap : Color look-up table.

colormap(MAP) sets the current figure's colormap to MAP. colormap('default') sets the current figure's colormap to the root's default, whose setting is JET.

1.9

Develop a script that loads and shows (in a single window, in two plots next to each other) an image and its part which has the following coordinates (row: 100, column: 100, width: 16,height: 16). Additionally, values of the pixels of the image parts should be listed in the Matlab console window. Use direct indexing of matrix elements, even though the imagecrop function exists.

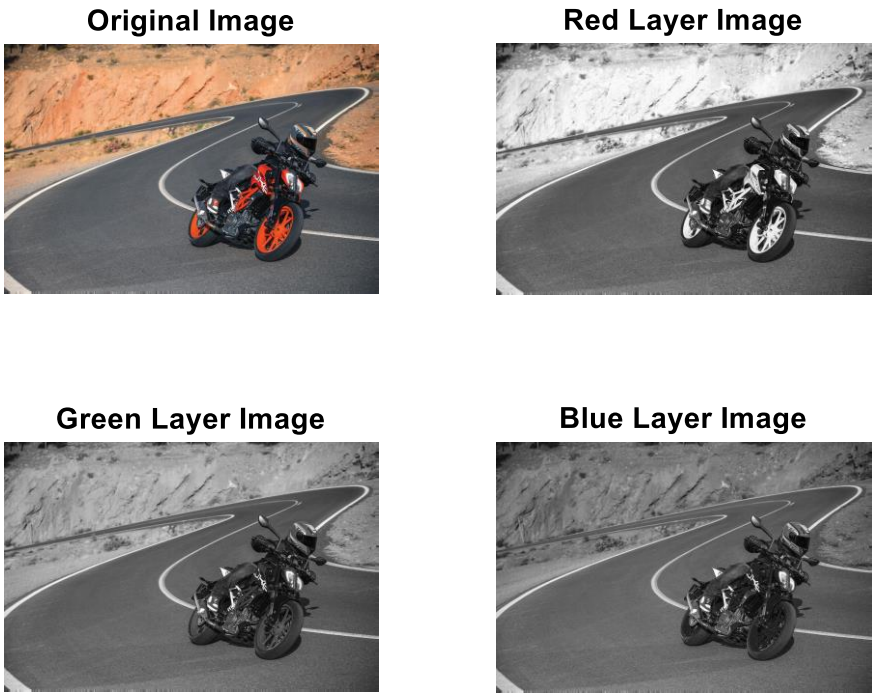
```
clear all

a=imread('Duke390.jpg');%to read image

b=a(100:116,100:116,:); %to crop a small part with size data from original
picture

subplot(1,2,1)
imshow(a) % show original picture
title('Original Image')
subplot(1,2,2)
imshow(b) % show cropped picture
title('Cropped Image')
```



1.10

Develop a script that loads a color image and shows (in a single window, in four plots next to each other) this image and its red, green, and blue components. The components should be shown as grayscale images.

```
clear all

a=imread('Duke390.jpg'); %Read image

subplot(2,2,1)
imshow(a) % Show Original image
title('Original Image')

subplot(2,2,2)
imshow(a(:,:,1))% Show red layer of image
title('Red Layer Image')

subplot(2,2,3)
imshow(a(:,:,2))% Show Green layer of image
title('Green Layer Image')

subplot(2,2,4)
imshow(a(:,:,3))% Show Blue layer of image
title('Blue Layer Image')
```



BIALYSTOK UNIVERSITY OF TECHNOLOGY

SIGNAL PROCESSING

TASK 2

TUTOR:

Dr inż. MAREK PARFIENIUK

STUDENTS:

AHMET TOREHAN HAYTOGLU

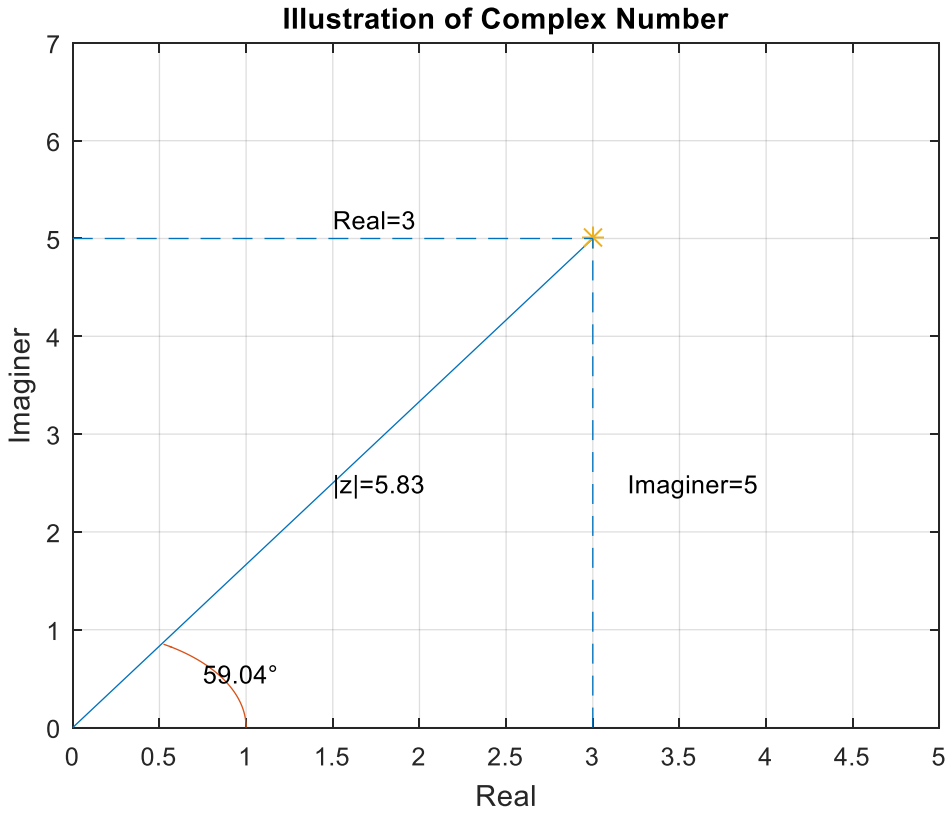
HARUN GULEC

TASK 2

2.1

Develop a plot that explains the notation of complex number. In the plot, show/mark/: an exemplary number,its modulus and arguments, as well as its both real and imaginary parts. Learn and use appropriate Matlab functions (real, imag, abs, angle, line, text, arc), as well as various colors, line styles, arrows etc.

```
clear all

cplx=3+5i; %define any complex number
x=real(cplx); %real part of complex number
y=imag(cplx); %imaginary part of complex number

modulo=abs(cplx); %modul of complex number

teta= 360/(2*pi)*atan(imag(cplx)/real(cplx));% argument of complex number in
degree form
a=0:0.02: (pi / (180/teta)); %%
Z=cos(a); %%Draw argument of complex number
T=sin(a); %%
plot([0 real(cplx)], [0 imag(cplx)],Z,T,x,y,'*','MarkerSize',8)

axis([0,x+2,0,y+2]);%length of axis

line([x x], [0 y],'LineStyle', '- -'); %to make dashed line imaginer
line([0 x], [y y],'LineStyle', '- -'); %to make dashed line imaginer

x1 = x/4;
y1 = y/9;
S=sprintf('%.2f%c',teta,char(176)); %Assign argument to variable in string
format
text(x1,y1,S) %Write argument to graphic

x1 = x/2;
y1 = y/2;
S=sprintf('|z|=%.2f',modulo);%Assign modul of compex number to variable in
string format
text(x1,y1,S )%Write modul of complex number to graphic

x1 = x/2;
y1 = y+0.2;
S=sprintf('Real=%d',x); %Assign real part of complex number in string format
text(x1,y1,S)%Write real part of complex number to graphic

x1 = x+0.2;
y1 = y/2;
S=sprintf('Imaginer=%d',y); %Assign real part of complex number in string format
text(x1,y1,S)%Write real part of complex number to graphic

xlabel('Real'); % Name of x-Label
ylabel('Imaginer'); % Name of y-Label
title('Illustration of Complex Number')
```



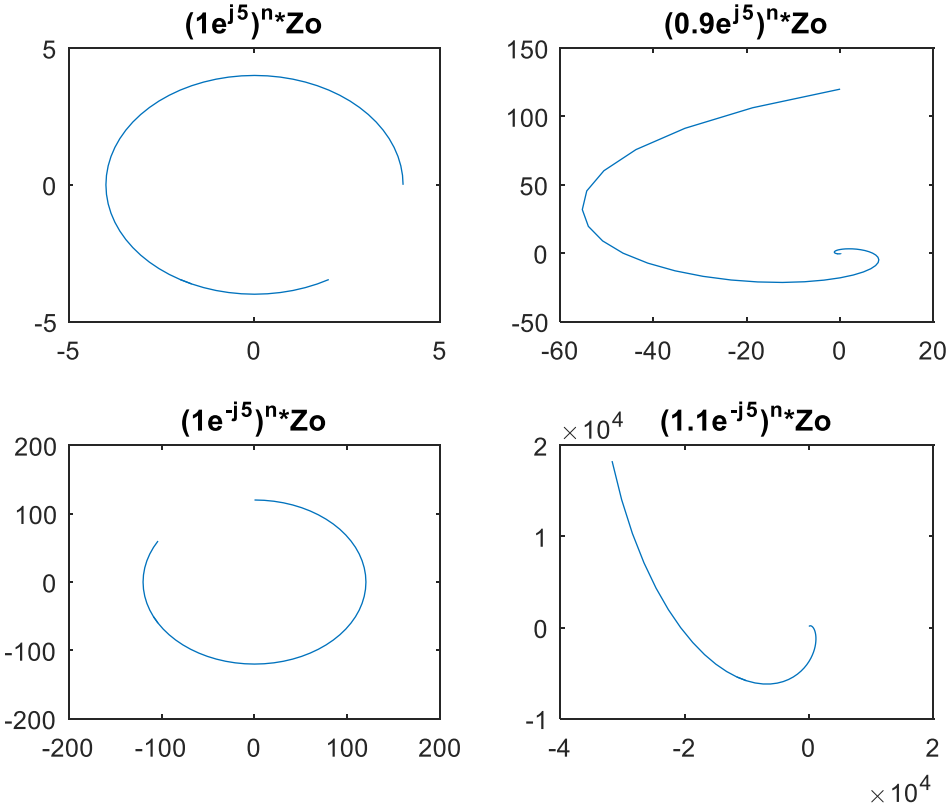
2.2

Compute values of the product

$$z_R^n z_0$$

for n = 0 ... 60, $z_0 = 4$, and $z_R = 1e^{j5^\circ}$ Illustrate your results in a plot. How does the plot look for $z_0 = 4^{j30^\circ}$ and $z_R = 0.9e^{j10^\circ}$ Experiment with other numbers, arguments and modules. For example. $z_R = 1e^{-j5^\circ}$ and $z_R = 1.1e^{-j5^\circ}$ Carefully convert angles from degrees to radians. Be careful about whether an argument is given in degrees or in radians.

```
clear all

n = 0:60 ;%define n 0 to 60

Zo = 4;

Zr=1*exp(1*i*(5*(360/(2*pi))); % define asked function

a= (Zr.^n) * Zo; %generate graphic

Zr2 =0.9*exp(1*i*10*(360/(2*pi))); %define other function
Zr3 = 1*exp(-1*i*5*(360/(2*pi))); %define other function
Zr4 = 1.1*exp(-1*i*5*(360/(2*pi))); %define other function

subplot(2 , 2 , 1)
plot(a)

subplot(2 , 2 , 2)
plot((4.^1*i*30)*(Zr2.^n))

subplot(2 , 2 , 3)
plot((4.^1*i*30)*(Zr3.^n))

subplot(2 , 2 , 4)
plot((4.^1*i*30)*(Zr4.^n))
```



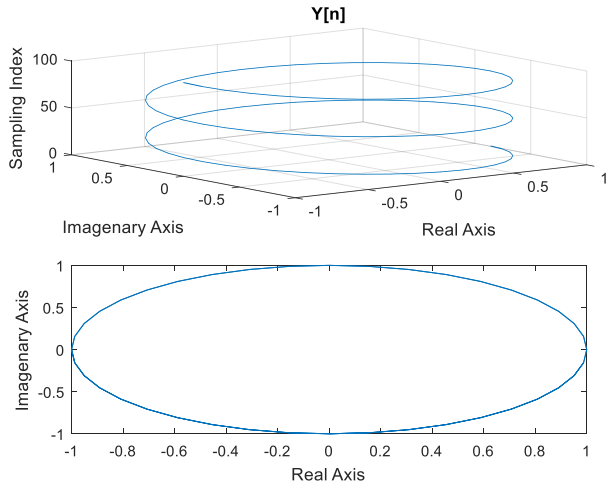
2.3

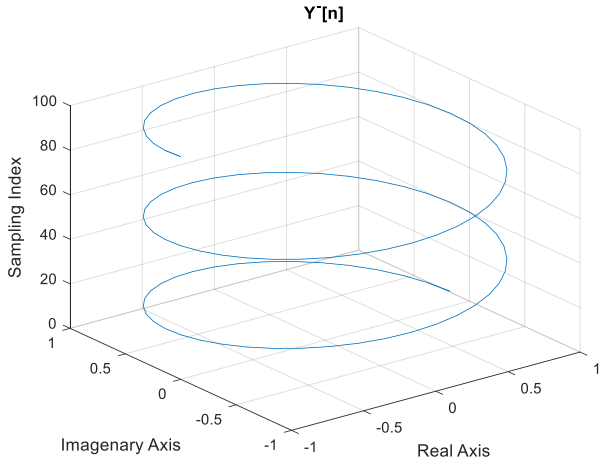
Plot in 3 dimensions (x: Re, y: Im, z: sample index) several dozens or several hundreds of samples of the signal y[n] = $e^{\frac{jn\pi}{20}}$. In other 3D plots, illustrate the conjugate signal $\bar{y}[n]$, as well as the signals

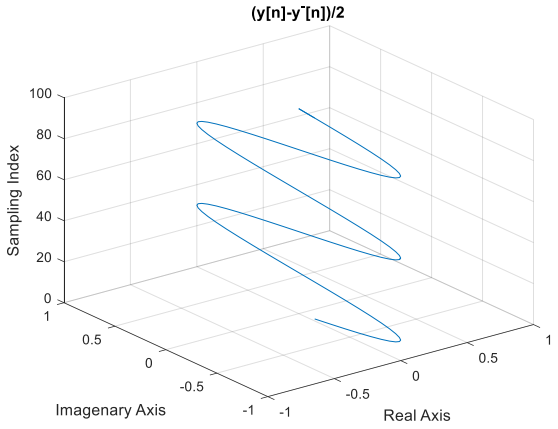
$$\frac{y[n]+\bar{y}[n]}{2} \text{ and } \frac{y[n]-\bar{y}[n]}{2}$$

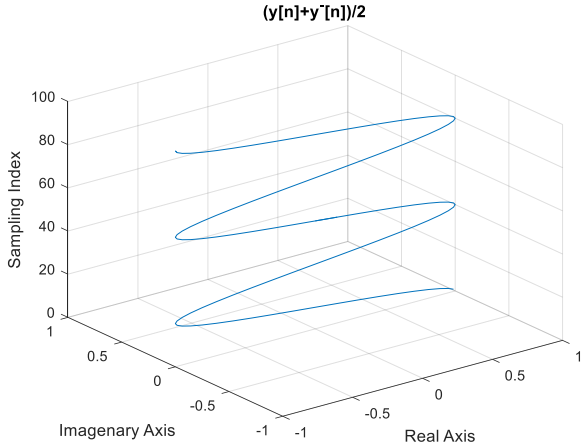
In 2 dimensions, plot both real and imaginary parts of y[n], and then relate these plots to the 3D plots You have drawn before. How to convert the first 3D plot into a picture similar to the corresponding 2D plot.

```
function result = complex( N )
clear all
n= 1 : 100 ; %n=1 to n=100 contain a hundred number
result=exp(-(pi * n * 1*i)/20 ); %assaign to result our signal
x= real (result) ; %real part of our result
y= imag (result) ; %imaginal part of our result
z=n; % z index for 3D plot
figure % show the  figure
subplot(2,1,1) %to demonstrate two plot in one figure
plot3 (x , y , z ) % Draw 3D plot of signal
grid on ; %open grid
xlabel('Real Axis');
ylabel('Imagenary Axis');
zlabel('Sampling Index');
title('Y[n]')
subplot(2,1,2)%to demonstrate two plot in one figure (For Second)
plot(x,y); %Draw 2D plot of signal

figure % show the  figure
plot3(x , -y , z )  % Draw 3D plot of conjugated signal
grid on; %open grid
title('Y^-[n]')
conjugateresult=x-j*y; %Calculate conjugated signal

c = (result + conjugateresult ) / 2; %Assign to c , first function
k=real(c);%real part of C
l=imag(c);%imaginal part of C

d = (result - conjugateresult ) / 2; %Assign to d , second function
m=real(d);%real part of D
n=imag(d);%imaginal part of D
xlabel('Real Axis');
ylabel('Imagenary Axis');
zlabel('Sampling Index');
figure
plot3(k , l , z);
grid on; %open grid
title('(y[n]+y^-[n])/2'
xlabel('Real Axis');
ylabel('Imagenary Axis');
zlabel('Sampling Index');
figure
plot3(m , n , z);
grid on; %open grid
title('(y[n]-y^-[n])/2')
xlabel('Real Axis');
ylabel('Imagenary Axis');
zlabel('Sampling Index');
end
```









2.4

There are given the following complex numbers: $z_1 = \sqrt{2} + j\sqrt{2}$, $z_2 = 3 - j4$, $z_3 = 5e^{j60°}j$, $z_4 = 2e^{-j30°}$. Determine the polar representations of z1 and z2, and the rectangular representation of z3 and z4. Calculate $z_1 z_2$ and $\frac{z_1}{z_2}$ by using the rectangular form and calculate $z_3 z_4$ and $\frac{z_3}{z_4}$ by using the polar form. Do calculations by hand, and then check results using Matlab. Subsequent stages of manual calculations and equations used should be published in Your report. Use an equation editor in a word processor, like MS Word, or prepare Your report using Latex. The complex numbers and results should be illustrated in the complex plane.

Matlab Solution :

```
clear all;

z1=sqrt(2)+1j.*sqrt(2);% define complex  number functions
z2=3-1j.*4;% define complex  number functions
z3=5*exp(1j.*pi/3);% define complex  number functions
z4=2*exp(-1j.*pi/6);% define complex  number functions

r1=abs(z1);%calculate module of complex number
tanTeta1=imag(z1)/real(z1); % calculate tangant teta of complex number
teta=atan(tanTeta1); % calculate teta in radian form
teta=360/(2*pi)*teta; % convert teta radian to degree form
S=sprintf('z1=%dcis(%d)',r1,ceil(teta)); % show polar form
disp(S);

r2=abs(z2);%calculate module of complex number
tanTeta2=imag(z2)/real(z2);% calculate tangant teta of complex number
teta2=atan(tanTeta2);% calculate teta in radian form
teta2=360/(2*pi)*teta2;% convert teta radian to degree form
S=sprintf('z2=%dcis(%d)',r2,int32(teta2));% show polar form
disp(S);

S=sprintf( 'z3=%s' , num2str(z3) );
disp(S);%show rectangular form
S=sprintf( 'z4=%s' , num2str(z4) );
disp(S);%show rectangular form

S=sprintf('z1*z2=%.2f+(%.2fj)' , real(z1*z2),imag(z1*z2));
disp(S)
S=sprintf('z1/z2=%.2f+(%.2fj)' , real(z1/z2),imag(z1/z2));
disp(S)
teta3=(360/(2*pi)*atan(imag(z3)/real(z3)));
teta4=(360/(2*pi)*atan(imag(z4)/real(z4)));
S=sprintf('z3*z4=%.2fcis(%.0f)' ,abs(z3)*abs(z4),(teta3+teta4));
disp(S)
S=sprintf('z3/z4=%.2fcis(%.0f)' ,abs(z3)/abs(z4),(teta3-teta4));
disp(S)
```

Output:

```
>> task2_4
z1=2cis(45)
z2=5cis(-53)
z3=2.5+4.3301i
z4=1.7321-1i
z1*z2=9.90+(-1.41j)
z1/z2=-0.06+(0.40j)
z3*z4=10.00cis(30)
z3/z4=2.50cis(90)
```

Theoretical Solution:

$\mathbf{Z_1}=\sqrt{2}+j\sqrt{2}$

$|Z_1|=\sqrt{\sqrt{2}^2+\sqrt{2}^2}=2$

$\tan\varphi=\frac{\sqrt{2}}{\sqrt{2}}=1$

$\arctan(1)=45°$

$Z_1=2\text{cis}(45)$

$\mathbf{Z_2}=3-j4$

$|Z_2|=\sqrt{3^2+(-4)^2}=5$

$\tan\varphi=\frac{-4}{3}$

$\arctan(\frac{-4}{3})=-53°$

$Z_2=5\text{cis}(-53)$

$\mathbf{Z_3}=5e^{j60}$

$5e^{j60}=5(\cos(60)+i\sin(60))$

$=5\ (\ 0.5+j0.86)$

$=2.5+j4.33$

$\mathbf{Z_4}=2e^{-j30}$

$2e^{-j30}=2(\cos(30)-i\sin(30))$

$=2\ (\ 0.86-j0.5)$

$=1.72-1i$

$\mathbf{Z_1}*\mathbf{Z_2}=(\sqrt{2}+\sqrt{2}i)*(3-4i)$

$3\sqrt{2}-4\sqrt{2}i+3\sqrt{2}i+4\sqrt{2}$

$Z_1*Z_2=7\sqrt{2}-\sqrt{2}i$

$\mathbf{Z_3}*\mathbf{Z_4}=(5*2)\,e^{i(60°-30°)}=10e^{i(30°)}$

$$\frac{\mathbf{Z_1}}{\mathbf{Z_2}}=\frac{\sqrt{2}+i\sqrt{2}}{3-4j}$$

$$\frac{(\sqrt{2}+\sqrt{2}i)*(3+4i)}{25}$$

$$\frac{3\sqrt{2}+4\sqrt{2}i+3\sqrt{2}i-4\sqrt{2}}{25}$$

$$\frac{Z_1}{Z_2}=\frac{-\sqrt{2}+7\sqrt{2}i}{25}$$

$$\frac{\mathbf{Z_3}}{\mathbf{Z_4}}=\frac{5e^{60°i}}{2e^{-30°i}}=2.5e^{90°}$$

BIALYSTOK UNIVERSITY OF TECHNOLOGY

SIGNAL PROCESSING

TASK 3

TUTOR:

Dr inż. MAREK PARFIENIUK

STUDENTS:

AHMET TOREHAN HAYTOGLU

HARUN GULEC

TASK 3

3.1

Recall the previous problems related to sound processing. Check scripts and functions You have developed , look through Your reports and results.

We have checked our scripts and functions about sound processing in task 1.

3.2

Develop a function that, for a given vector plots (the stem function, separate plots in a single window) real parts, imaginary parts, modules and arguments of its elements. It should be possible, as an option, to skip showing real and imaginary parts. Care about correct labeling of axes.

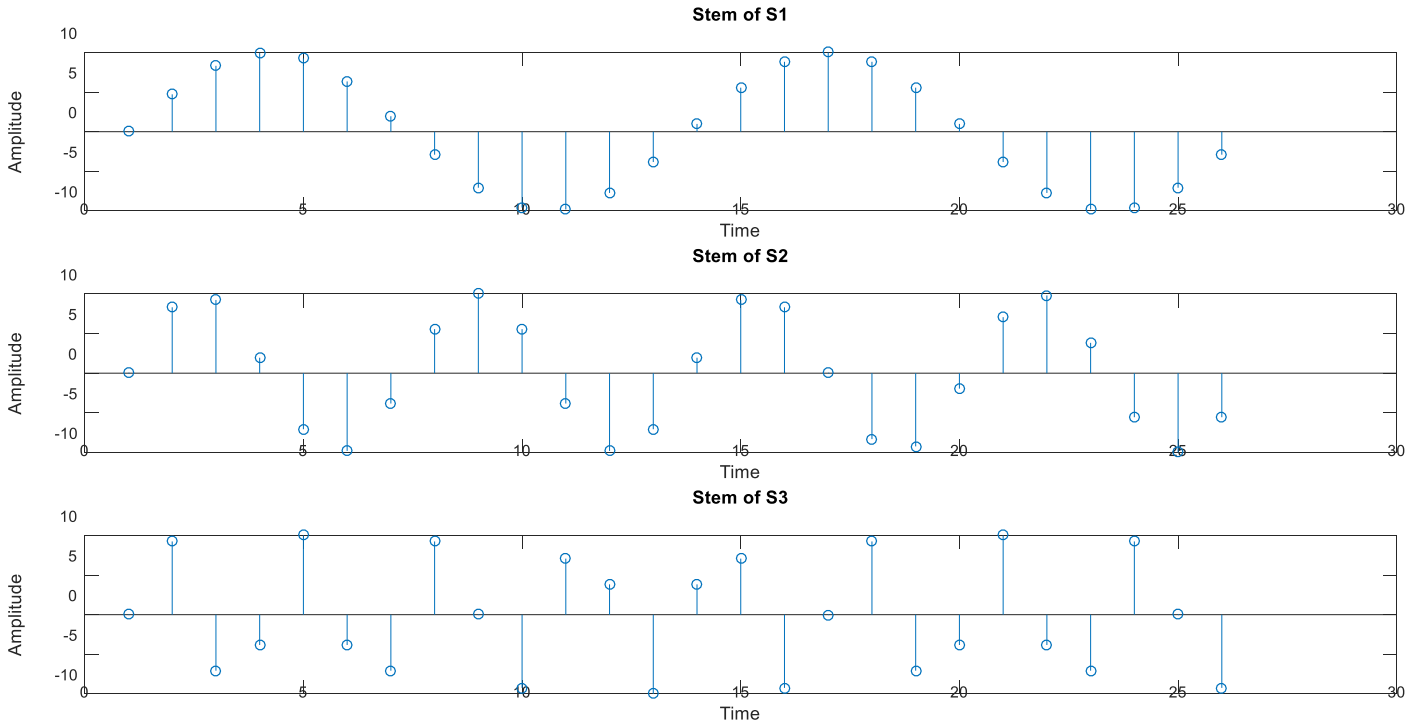
```
function [reel,imaginer,modul,teta] = drawstemofcomplex(z , option)

clc

imaginer=imag(z); %imaginer parts of z
maks=max(imaginer); %to determine maksimum value of Imaginer

reel = real(z); %real parts of z
maks1=max(reel); %to determine maksimum value of Real

modul=sqrt((reel.^2) + (imaginer.^2)); %Modules of z
maks2=max(modul); %to determine maksimum value of Modul

radian=angle(z); %Complex number's arguments angle in Radian form
teta=radian*57.2957795; %Radian to degree
maks3=max(teta); %Maksimum value of Argument

option=lower(option); %To convert to lower case option
count=0;
if size(option)==0
    msgbox('Please enter an option!','Fatal Error :)','error')
    return; %If option is not given
end
if (size(strfind(option,'none'))~=0)
    return;%If option is 'none' return without figure
end
if(size(strfind(option,'all'))~=0)
    figure%if option is all then show all graphics
    subplot(4 ,1 ,1)
    stem(reel)
    ylim([0 maks1+2 ])%to determine y-label limits
    title('Real') % Header of graphics
    subplot(4 ,1 ,2)
    stem(imaginer)
    ylim([0 maks+2])%to determine y-label limits
    title('Imaginer')% Header of graphics
    subplot(4, 1, 3)
    stem(modul)
    ylim([0 maks2+2])%to determine y-label limits
    title('Modul')% Header of graphics
    subplot(4 ,1 , 4)
    stem(teta)
    ylim([0 maks3+2])%to determine y-label limits
    title('Argument')% Header of graphics
    return;
end
```

```
if(size(strfind(option,'r')) ~=0)
  count = count+1; %to understand how many graphics we need
end
if(size(strfind(option,'i')) ~=0)
  count = count+1; %to understand how many graphics we need
end
if(size(strfind(option,'m')) ~=0)
  count = count+1; %to understand how many graphics we need
end
if(size(strfind(option,'a')) ~=0)
  count = count+1; %to understand how many graphics we need
end

figure
for n=1:count
    if size(strfind(option,'r')) ~=0
        subplot(count,1 ,n)% if option has 'r', plot real graphic
        stem(reel)
        ylim([0 maks1+2 ])
        title('Real')
        option=strrep(option,'r','');% delete 'r' from option
    elseif size(strfind(option,'i')) ~=0
        subplot(count,1 ,n)% if option has 'i', plot imaginer graphic
        stem(imaginer)
        ylim([0 maks+2])
        title('Imaginer')
        option=strrep(option,'i','');% delete 'i' from option
    elseif size(strfind(option,'m')) ~=0
        subplot(count,1, n) %if option has 'm', plot module graphic
        stem(modul)
        ylim([0 maks2+2])
        title('Modul')
        option=strrep(option,'m','');% delete 'm' from option
    elseif size(strfind(option,'a')) ~=0
        subplot(count,1 , n)%if option has 'a', plot argument graphic
        stem(teta)
        ylim([0 maks3+2])
        title('Argument')
        option=strrep(option,'a','');% delete 'a' from option
    end
end

end
```

3.3

Generate several sinusoidal signals whose frequencies are different but time extents are the same (the numbers of samples are the same e.g. 128 or 256). This code will be used in the following scripts.

```
clear all

fs=256;%Sampling frequency value
A=10;%magnitude value
t1=0;%start time
t2=0.1;%stop time
shift=0;%phase shift

f1=20
s1=sinusoid( fs  ,f1 , A , t1 ,t2 , shift);

f2=40
s2=sinusoid( fs  ,f2, A , t1 ,t2 , shift);

f3=80
s3=sinusoid( fs  ,f3, A , t1 ,t2 , shift);

subplot(3, 1 ,1)
stem(s1)
xlabel('Time');
ylabel('Amplitude');
title('Stem of S1')
subplot(3 , 1 , 2)
stem(s2)
xlabel('Time');
ylabel('Amplitude');
title('Stem of S2')
subplot(3, 1 ,3)
stem(s3)
xlabel('Time');
ylabel('Amplitude');
title('Stem of S3')
```

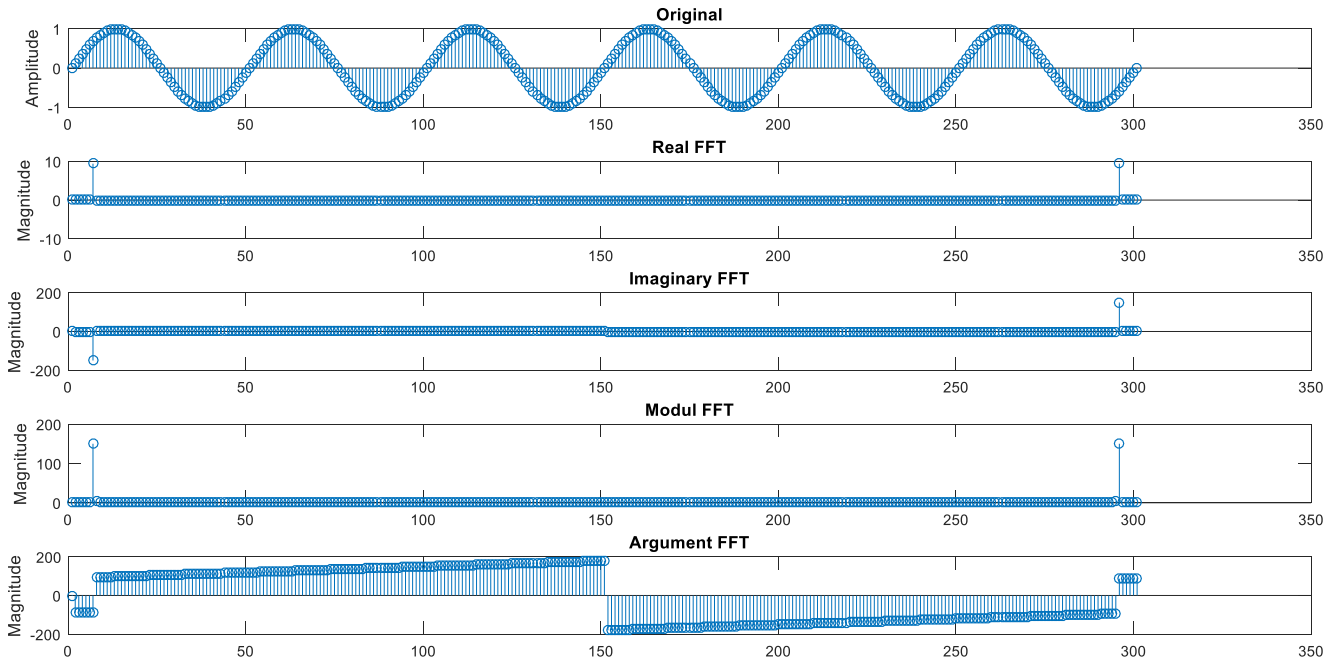


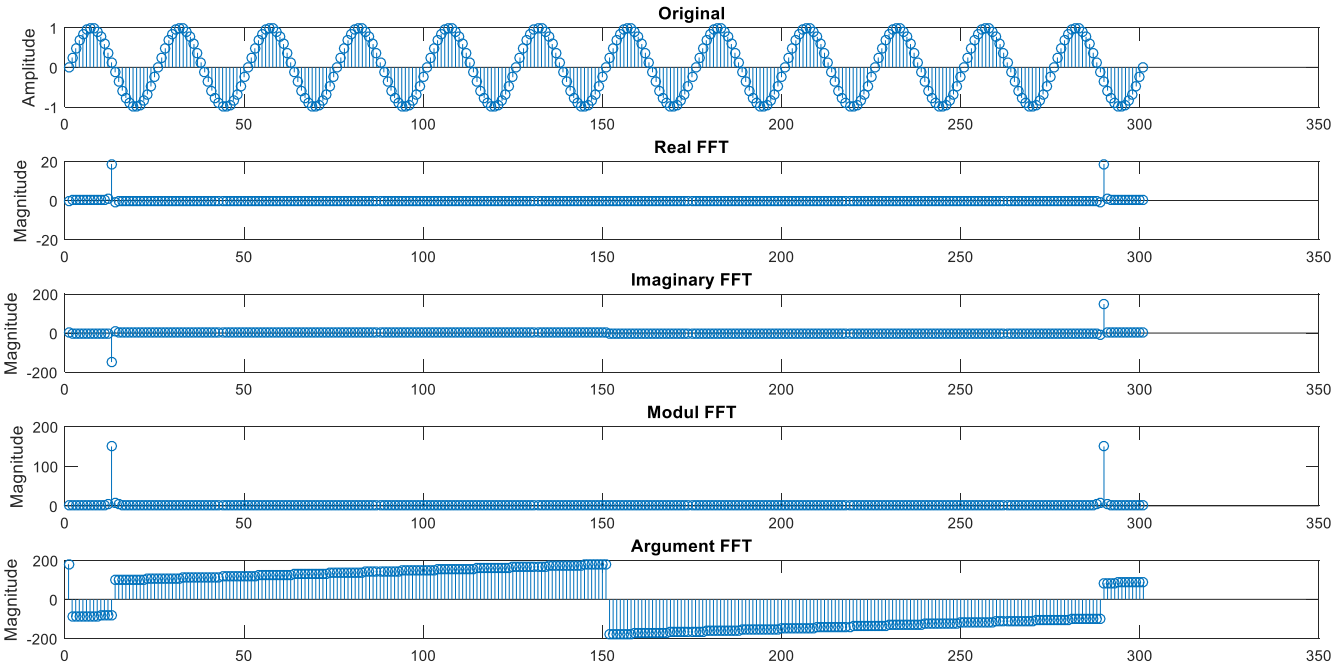
3.4

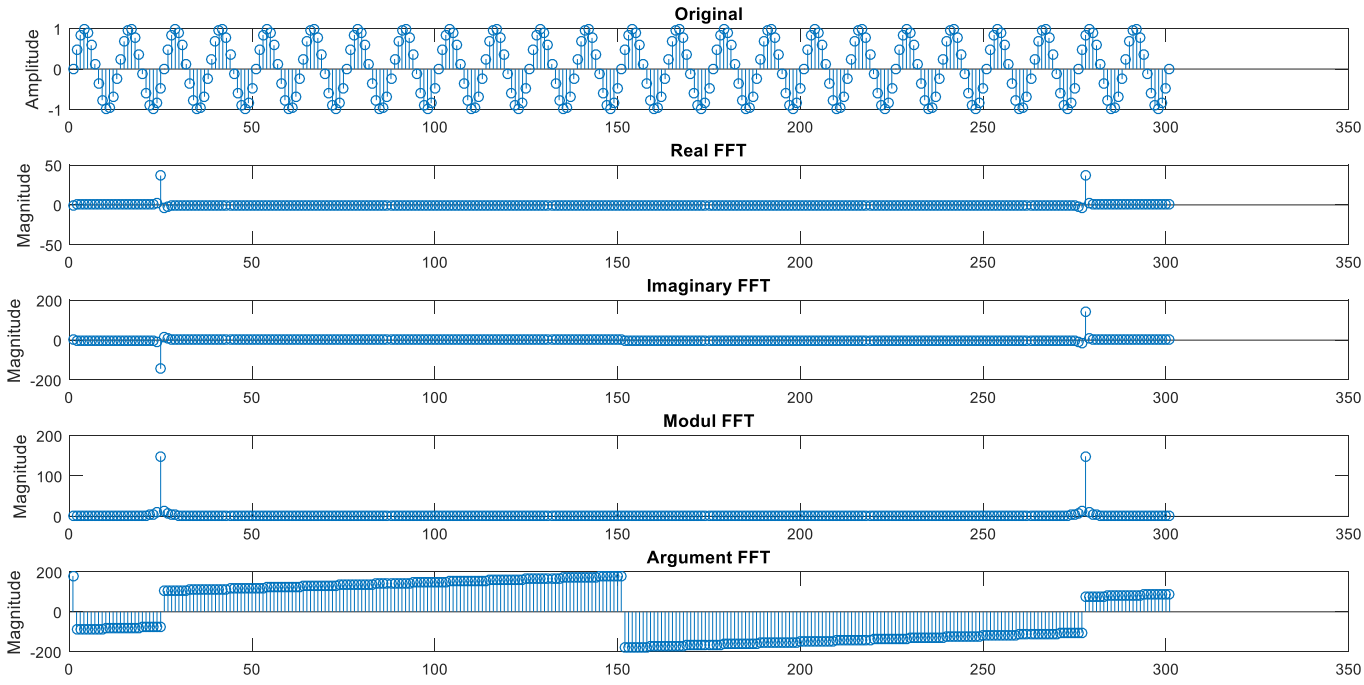
For each sinusoid, compute its Discrete Fourier Transform (DFT, the fft function) and plot the results, using the function You developed when solving the previous problem. Interpret and compare plots. Think on connections between indexes of DFT samples and frequencies.

```
clear all

fs = 1000; %value of sampling frequency
A=1; % value of magnitude
t1=0;% start time
t2 = 0.3;%stop time
shift=0;%shift phase
f1=20
s1=sinusoid( fs  ,f1 , A , t1 ,t2 , shift);
f2=40
s2=sinusoid( fs  ,f2, A , t1 ,t2 , shift);
f3=80
s3=sinusoid( fs  ,f3, A , t1 ,t2 , shift);
figure
[R,I,M,Arg]=drawstemofcomplex(fft(s1),'none');
%calculate Real(R), Imaginer(I),Module(M),Argument(Arg) of complex system
subplot(5,1,1)
stem(s1) %original stem of s1
title('Original');
ylabel('Amplitude')
subplot(5,1,2)
stem(R) %Real fft stem of s1
title('Real FFT');
ylabel('Magnitude')
subplot(5,1,3)
stem(I) %Imaginary fft stem of s1
title('Imaginary FFT');
ylabel('Magnitude')
subplot(5,1,4)
stem(M) %Module fft stem of s1
title('Modul FFT');
ylabel('Magnitude')
subplot(5,1,5)
stem(Arg) %Argument fft stem of s1
title('Argument FFT');
ylabel('Magnitude')
%stems of s2
figure
[R,I,M,Arg]=drawstemofcomplex(fft(s2),'none');
subplot(5,1,1)
stem(s2)
title('Original');
ylabel('Amplitude')
subplot(5,1,2)
stem(R)
title('Real FFT');
ylabel('Magnitude')
subplot(5,1,3)
stem(I)
title('Imaginary FFT');
ylabel('Magnitude')
subplot(5,1,4)
stem(M)
title('Modul FFT');
ylabel('Magnitude')
subplot(5,1,5)
stem(Arg)
title('Argument FFT');
ylabel('Magnitude')
```

```
%%
%stems of s3
figure
[R,I,M,Arg]=drawstemofcomplex(fft(s3),'none');
subplot(5,1,1)
stem(s3)
title('Original');
ylabel('Amplitude')
subplot(5,1,2)
stem(R)
title('Real FFT');
ylabel('Magnitude')
subplot(5,1,3)
stem(I)
title('Imaginary FFT');
ylabel('Magnitude')
subplot(5,1,4)
stem(M)
title('Modul FFT');
ylabel('Magnitude')
subplot(5,1,5)
stem(Arg)
title('Argument FFT');
ylabel('Magnitude')
%%
```





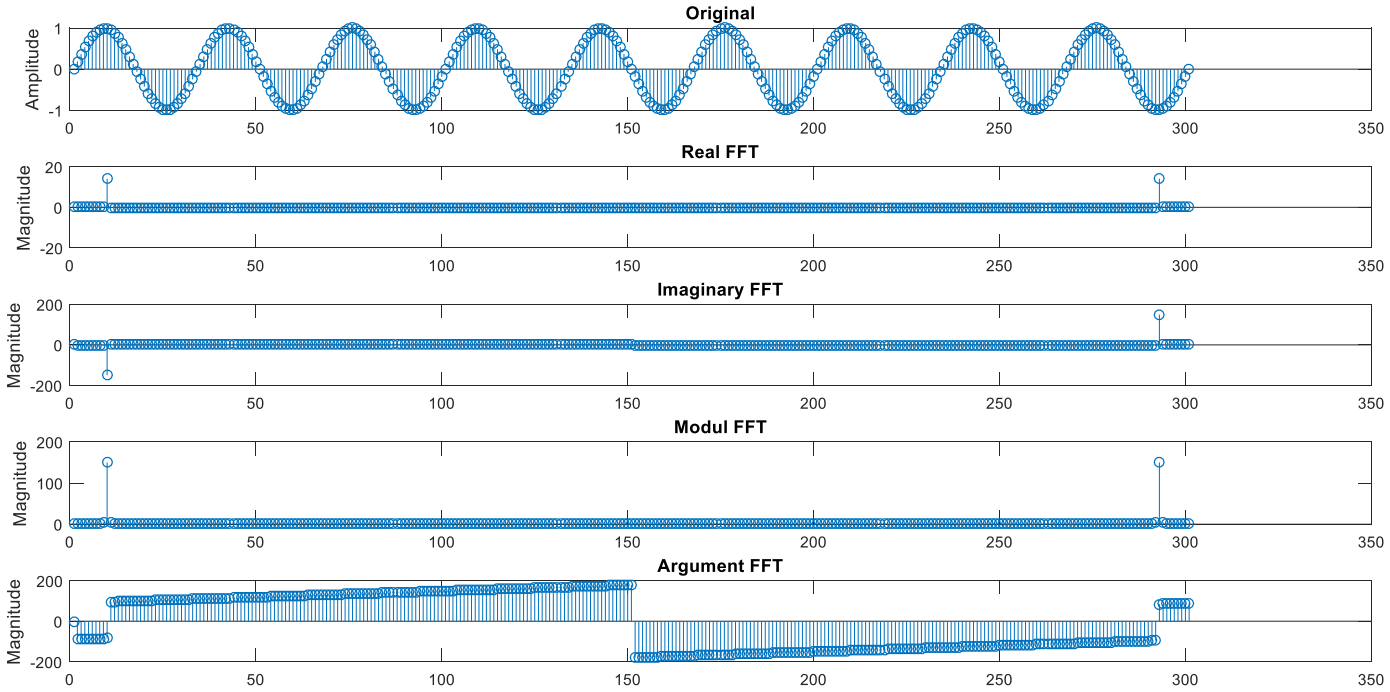


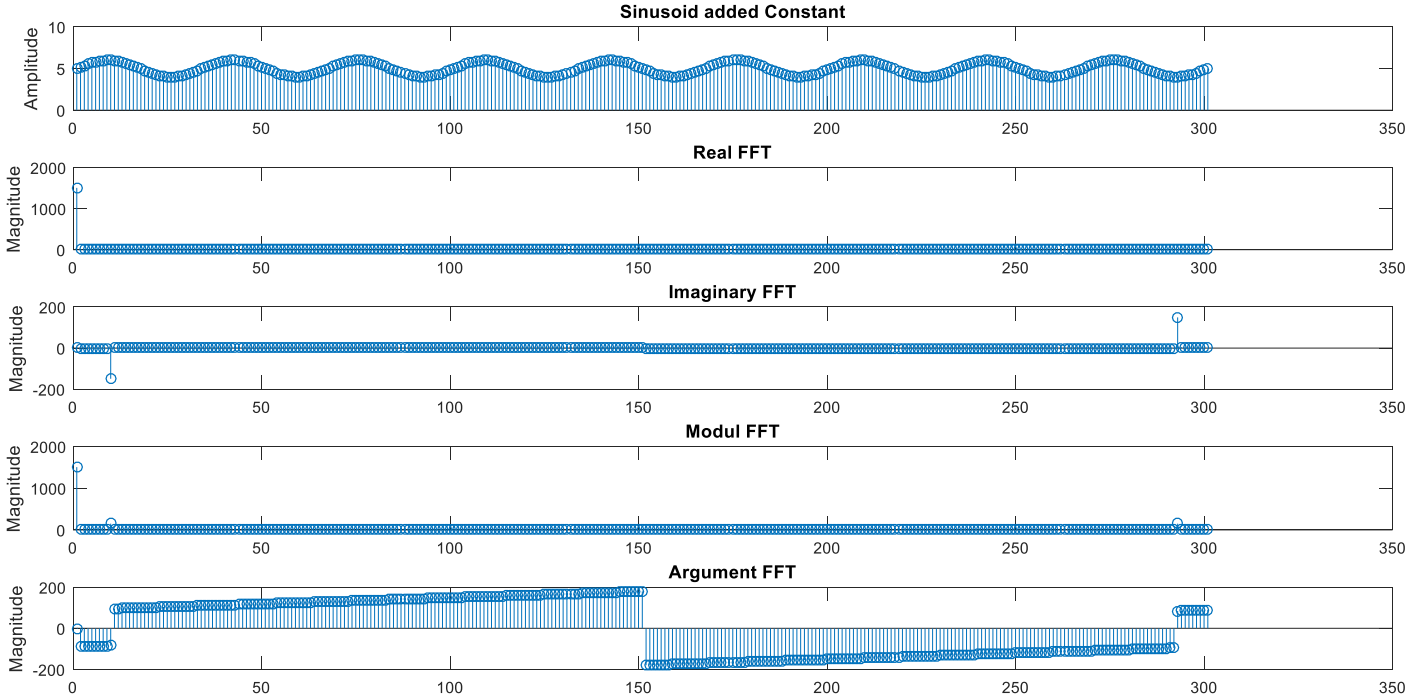
3.5

How does the result of the DFT varies when a constant is added to a sinusoid?

When we add constant(5) to sinusoid, we have seen on the graphic original sinusoid raised till value of constant. Real FFT and module FFT has changed but imaginary and argument FFT has not changed. So, constant added to a sinusoid doesn't affect argument and imaginary FFT.

```
clear all

fs = 1000; %value of sampling frequency
f= 30;%value of frequency
A=1; % value of magnitude
t1=0;% start time
t2 = 0.3;%stop time
shift=0;%shift phase
C=5;%value of constant

x=sinusoid(fs ,f, A , t1 ,t2 , shift);%generate sinusoid
xC=C+x;

figure
[R,I,M,Arg]=drawstemofcomplex(fft(x),'none');
%calculate Real(R), Imaginer(I),Module(M),Argument(Arg) of complex system
subplot(5,1,1)
stem(x) %original stem of x
title('Original');

subplot(5,1,2)
stem(R) %Real fft stem of x
title('Real FFT');

subplot(5,1,3)
stem(I) %Imaginary fft stem of x
title('Imaginary FFT');

subplot(5,1,4)
stem(M) %Module fft stem of x
title('Modul FFT');

subplot(5,1,5)
stem(Arg) %Argument fft stem of x
title('Argument FFT');

%%
%stems of xC
figure
[R,I,M,Arg]=drawstemofcomplex(fft(xC),'none');
subplot(5,1,1)
stem(xC)
title('Sinusoid added Constant');
subplot(5,1,2)
stem(R)
title('Real FFT');
subplot(5,1,3)
stem(I)
title('Imaginary FFT');
subplot(5,1,4)
stem(M)
title('Modul FFT');
subplot(5,1,5)
stem(Arg)
title('Argument FFT');
%%
```

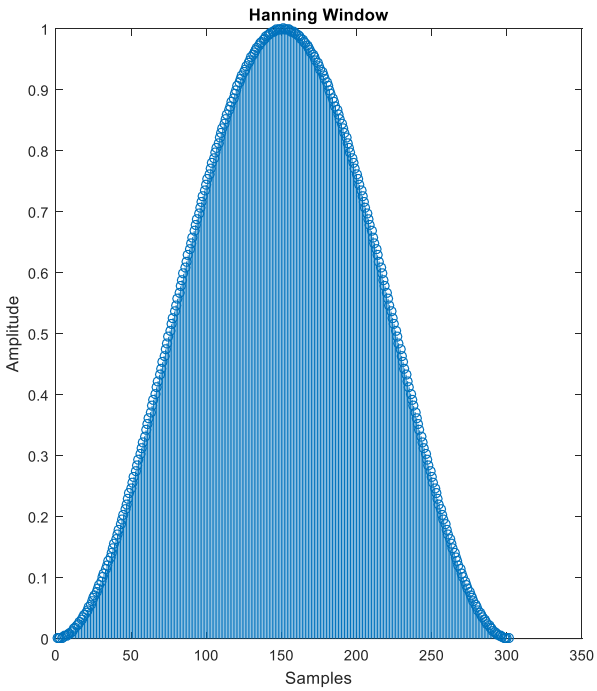

Original
Amplitude
Real FFT
Magnitude
Imaginary FFT
Magnitude
Modul FFT
Magnitude
Argument FFT
Magnitude

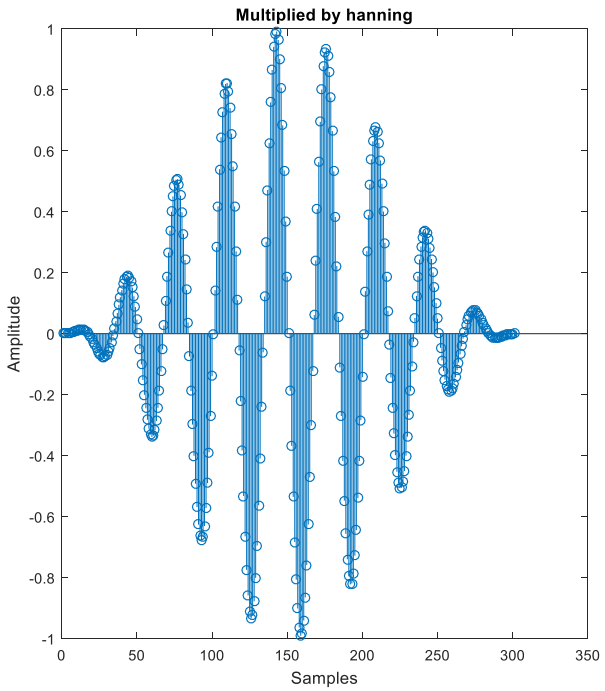

Sinusoid added Constant
Amplitude
Real FFT
Magnitude
Imaginary FFT
Magnitude
Modul FFT
Magnitude
Argument FFT
Magnitude

3.6

How do DFT results depend on whether signal encompasses integer or non-integer multiplies of the periods of the sinusoids that form this signal. For example, consider one sinusoid with and without one sample that begins an additional period. For a signal with non-integer multiply of the periods, how are DFT results affected by windowing this signal by using eg. Hamming window. Use the window function and element-by-element multiply the signal by the window?

```
clear all

fs = 1000; %value of sampling frequency
f=30;%value of frequency
A=1; % value of magnitude
t1=0;% start time
t2 = 0.3;%stop time
shift=0;%shift phase
x1=sinusoid(fs ,f, A , t1 ,t2 , shift);%generate 3 random sinusoid

h = hanning(length(x1)); %hanning of x1
figure
subplot(1,2,1)
stem(h) %%stem of Hanning window
title('Hanning Window');
x1 = x1(:) .* h(:); %% multiply elemet-by-element x1 by hanning
subplot(1,2,2)
stem(x1) % stem of x1 hanning form
title('Multiplied by hanning');
```

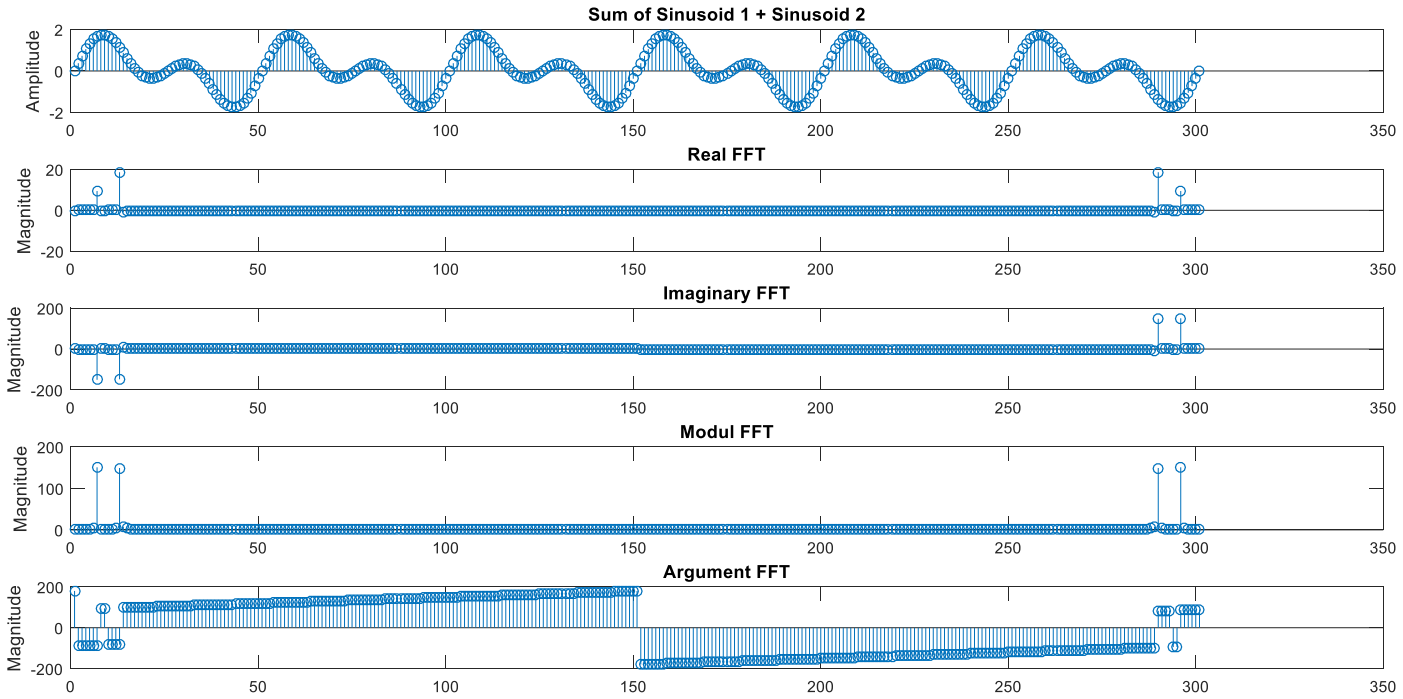


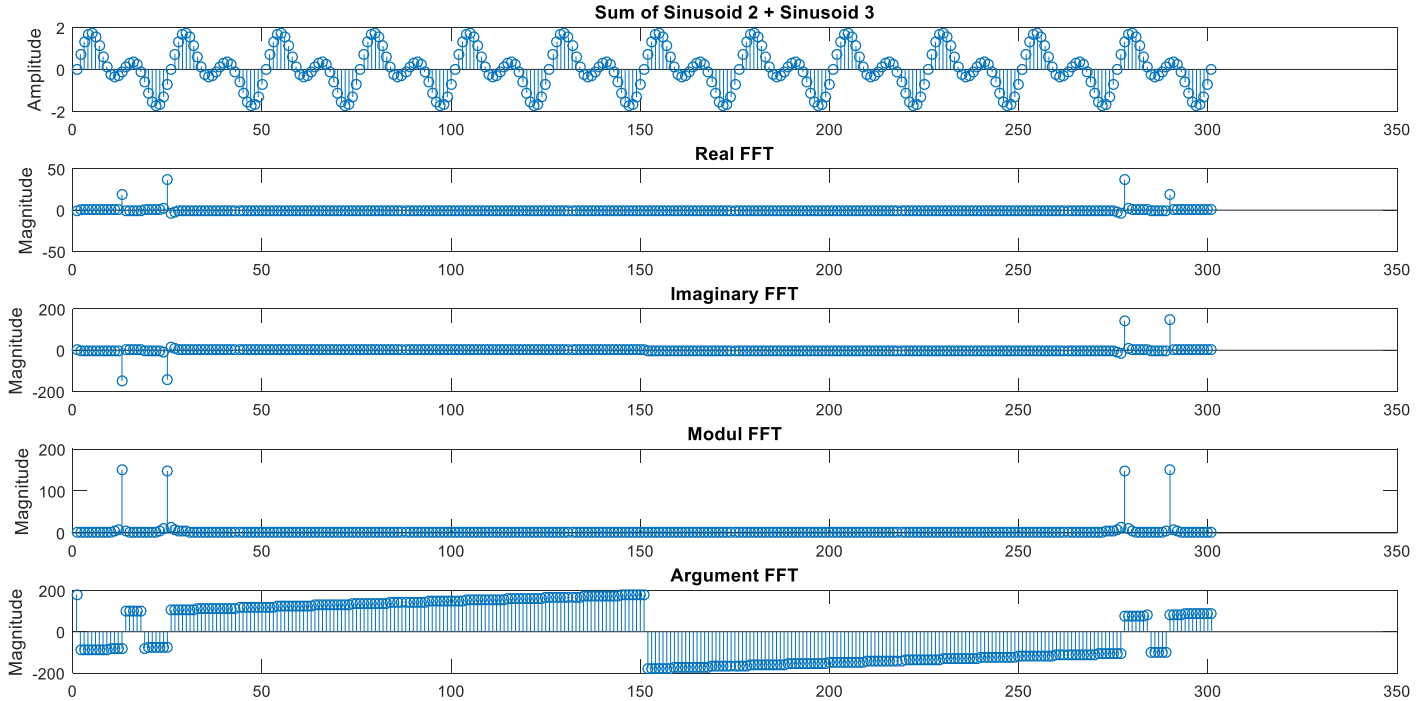


3.7

Repeat Problem 3.4 for various possible sums of sinusoids. Compare results to the results of Problem 3.4.

```
clear all
fs = 1000; %value of sampling frequency
A=1; % value of magnitude
t1=0;% start time
t2 = 0.3;%stop time
shift=0;%shift phase
f1=20
s1=sinusoid( fs  ,f1 , A , t1 ,t2 , shift);
f2=40
s2=sinusoid( fs  ,f2, A , t1 ,t2 , shift);
f3=80
s3=sinusoid( fs  ,f3, A , t1 ,t2 , shift);
%stems of s1+s2
figure
sum_s1s2 = s1+s2;
[R,I,M,Arg]=drawstemofcomplex(fft(sum_s1s2),'none');
subplot(5,1,1)
stem(sum_s1s2)
title('Sum of Sinusoid 1 + Sinusoid 2');
ylabel('Amplitude')
subplot(5,1,2)
stem(R)
title('Real FFT');
ylabel('Magnitude')
subplot(5,1,3)
stem(I)
title('Imaginary FFT');
ylabel('Magnitude')
subplot(5,1,4)
stem(M)
title('Modul FFT');
ylabel('Magnitude')
subplot(5,1,5)
stem(Arg)
title('Argument FFT');
ylabel('Magnitude')
%stems of s2+s3
figure
sum_s2s3 = s2+s3;
[R,I,M,Arg]=drawstemofcomplex(fft(sum_s2s3),'none');
subplot(5,1,1)
stem(sum_s2s3)
title('Sum of Sinusoid 2 + Sinusoid 3');
ylabel('Amplitude')
subplot(5,1,2)
stem(R)
title('Real FFT');
ylabel('Magnitude')
subplot(5,1,3)
stem(I)
title('Imaginary FFT');
ylabel('Magnitude')
subplot(5,1,4)
stem(M)
title('Modul FFT');
ylabel('Magnitude')
subplot(5,1,5)
stem(Arg)
title('Argument FFT');
ylabel('Magnitude')
```

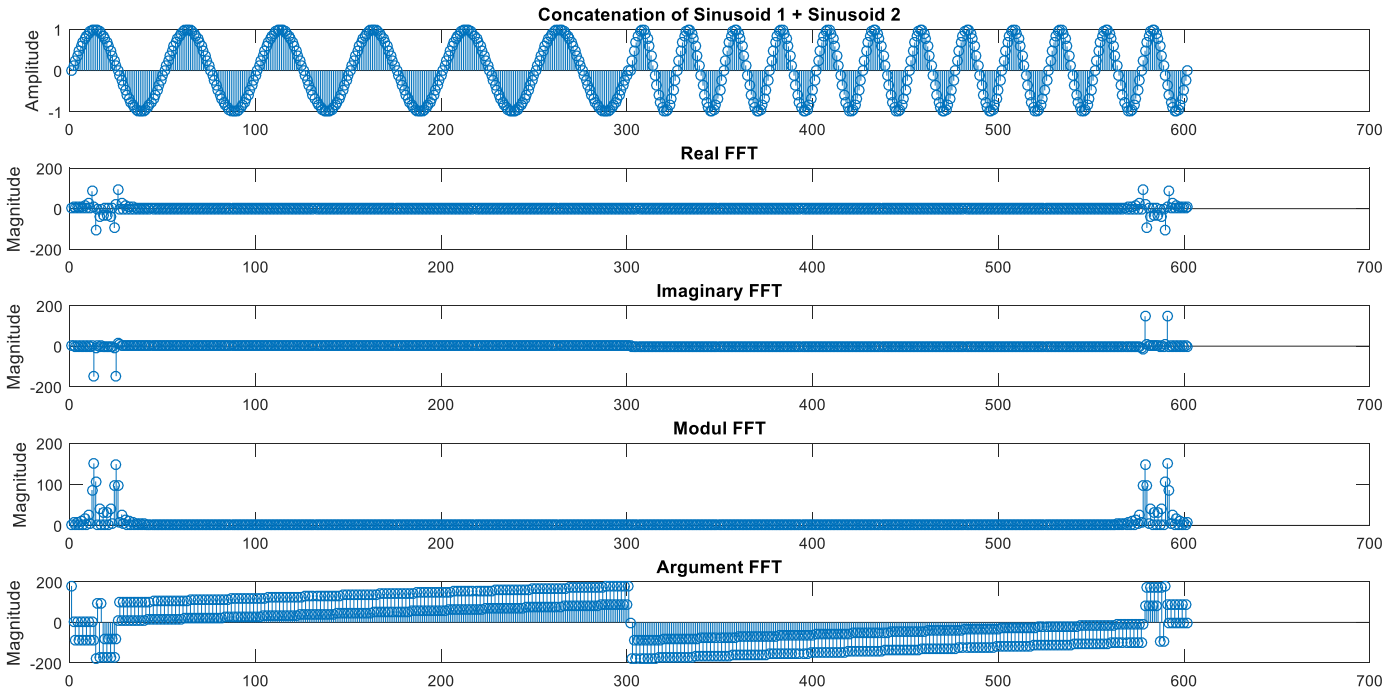

Sum of Sinusoid 1 + Sinusoid 2
Amplitude
Real FFT
Magnitude
Imaginary FFT
Magnitude
Modul FFT
Magnitude
Argument FFT
Magnitude

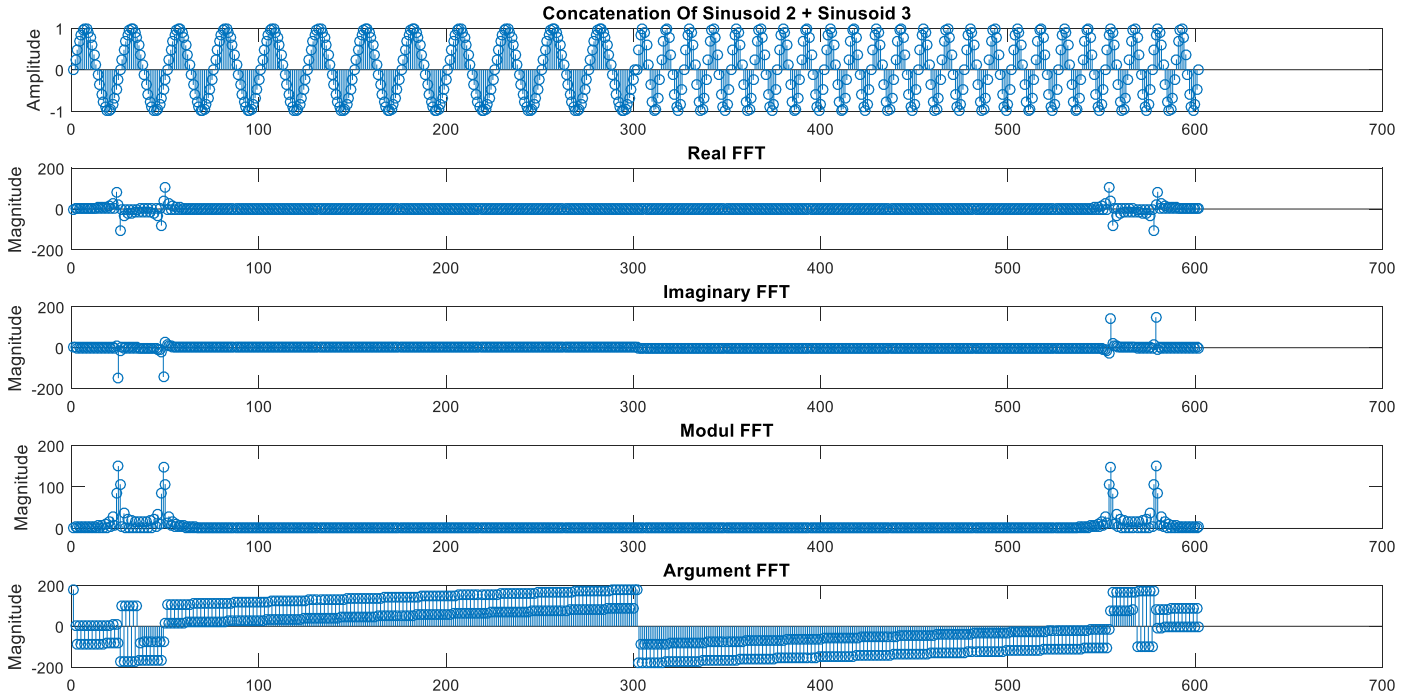

Sum of Sinusoid 2 + Sinusoid 3
Amplitude
Real FFT
Magnitude
Imaginary FFT
Magnitude
Modul FFT
Magnitude
Argument FFT
Magnitude

3.8

Repeat the previous task for various concatenations of sinusoids.

```
clear all

fs = 1000; %value of sampling frequency
A=1; % value of magnitude
t1=0;% start time
t2 = 0.3;%stop time
shift=0;%shift phase
[x1,x2,x3]=Generatesinusoid(fs , A , t1 ,t2 , shift);%generate 3 random sinusoid

%%
%stems of concatenations x1,x2
figure
con_x1x2 = [x1 x2];
[R,I,M,Arg]=drawstemofcomplex(fft(con_x1x2),'none');
subplot(5,1,1)
stem(con_x1x2)
title('Concatenation of Sinusoid 1 + Sinusoid 2');
subplot(5,1,2)
stem(R)
title('Real FFT');
subplot(5,1,3)
stem(I)
title('Imaginary FFT');
subplot(5,1,4)
stem(M)
title('Modul FFT');
subplot(5,1,5)
stem(Arg)
title('Argument FFT');
%%
%stems of concatenations x2,x3
figure
con_x2x3 = [x2 x3];
[R,I,M,Arg]=drawstemofcomplex(fft(con_x2x3),'none');
subplot(5,1,1)
stem(con_x2x3)
title('Concatenation Of Sinusoid 2 + Sinusoid 3');
subplot(5,1,2)
stem(R)
title('Real FFT');
subplot(5,1,3)
stem(I)
title('Imaginary FFT');
subplot(5,1,4)
stem(M)
title('Modul FFT');
subplot(5,1,5)
stem(Arg)
title('Argument FFT');
%%
```

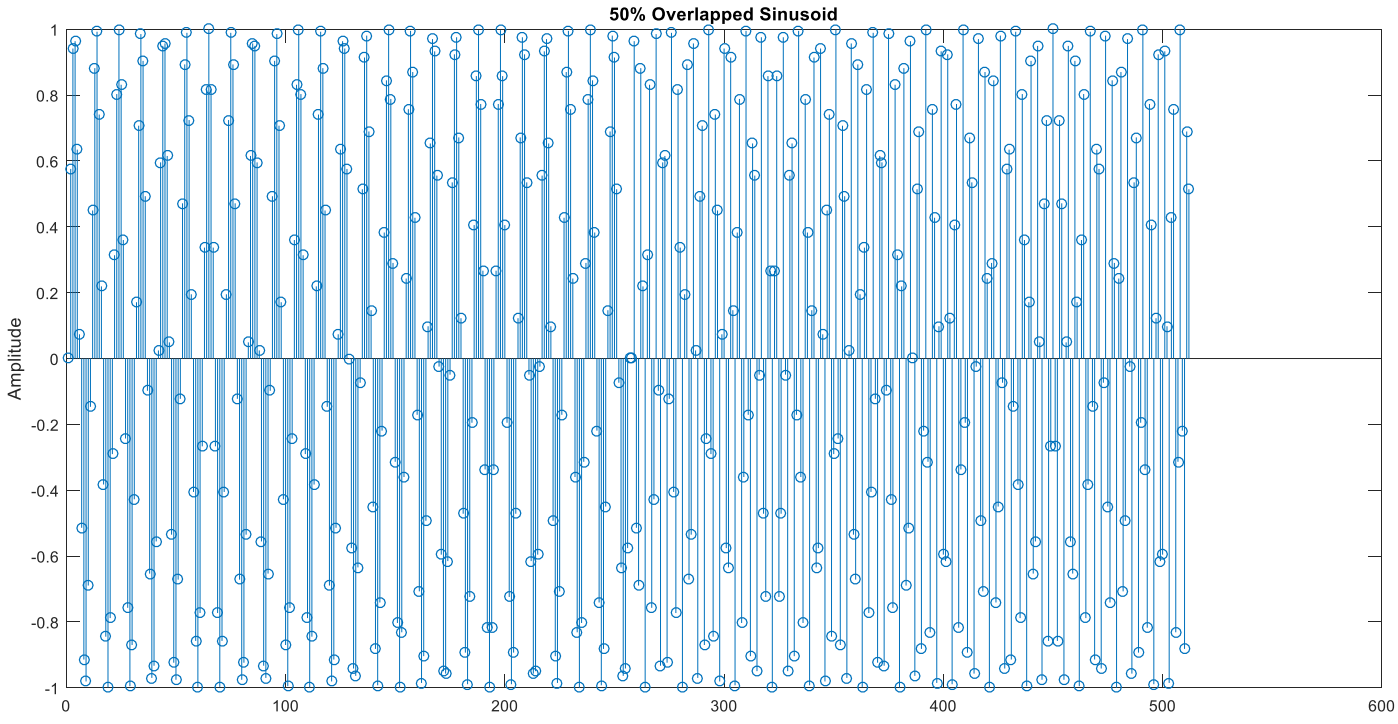

Concatenation of Sinusoid 1 + Sinusoid 2
Amplitude
Real FFT
Magnitude
Imaginary FFT
Magnitude
Modul FFT
Magnitude
Argument FFT
Magnitude

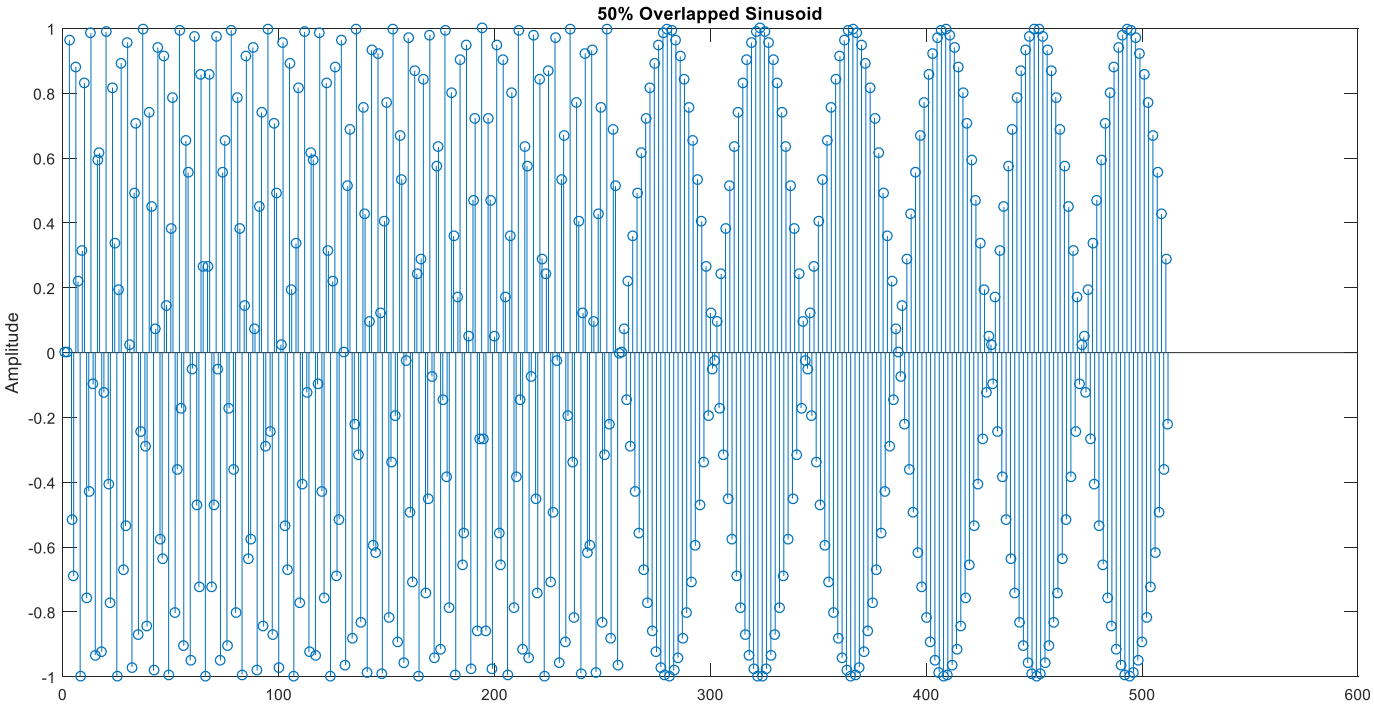

Concatenation Of Sinusoid 2 + Sinusoid 3
Amplitude
Real FFT
Magnitude
Imaginary FFT
Magnitude
Modul FFT
Magnitude
Argument FFT
Magnitude

3.9

Develop a script that extracts frames from a signal, with 50% overlap. Each frame is DFT-transformed, optionally after being windowed using the Hamming window, and the DFT results are displayed in a figure (frame-by-frame, with overwriting). Before processing a next frame, the script should wait for a confirmation by the user or should be delayed by several seconds, so as to allow the user to look at the plot of the current frame.

```
clear all

fs = 512; %value of sampling frequency
A=1; % value of magnitude
t1=0;% start time
t2 = 0.5;%stop time
shift=0;%shift phase
f1=50
s1=sinusoid( fs  ,f1 , A , t1 ,t2 , shift);

f2=150
s2=sinusoid( fs  ,f2, A , t1 ,t2 , shift);

f3=250
s3=sinusoid( fs  ,f3, A , t1 ,t2 , shift);
S=[s1,s2,s3]
lenghtofOverlap=512;
index=1:lenghtofOverlap;
for k=0:1
    f=S(index+(k.*(lenghtofOverlap/2)));
    stem(f)
    title('50% Overlapped Sinusoid')
    xlabel('Time');
    ylabel('Amplitude');
    pause(5)
end
```





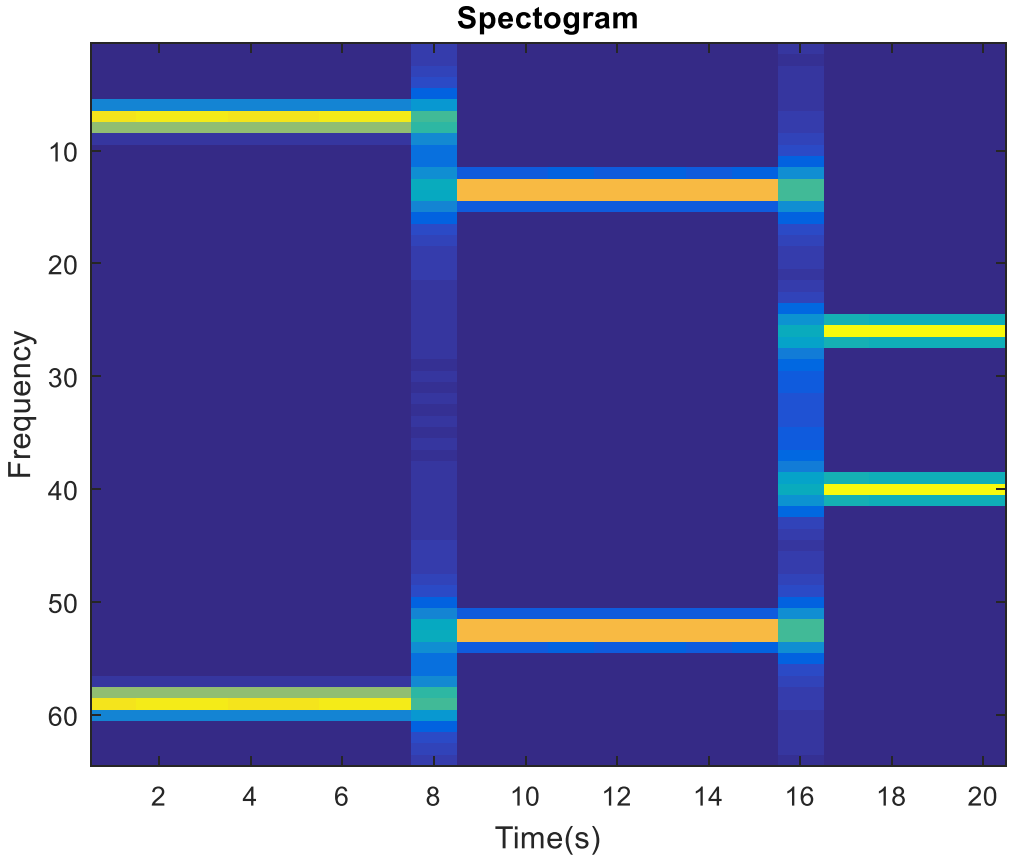
3.10

Nothing to report.

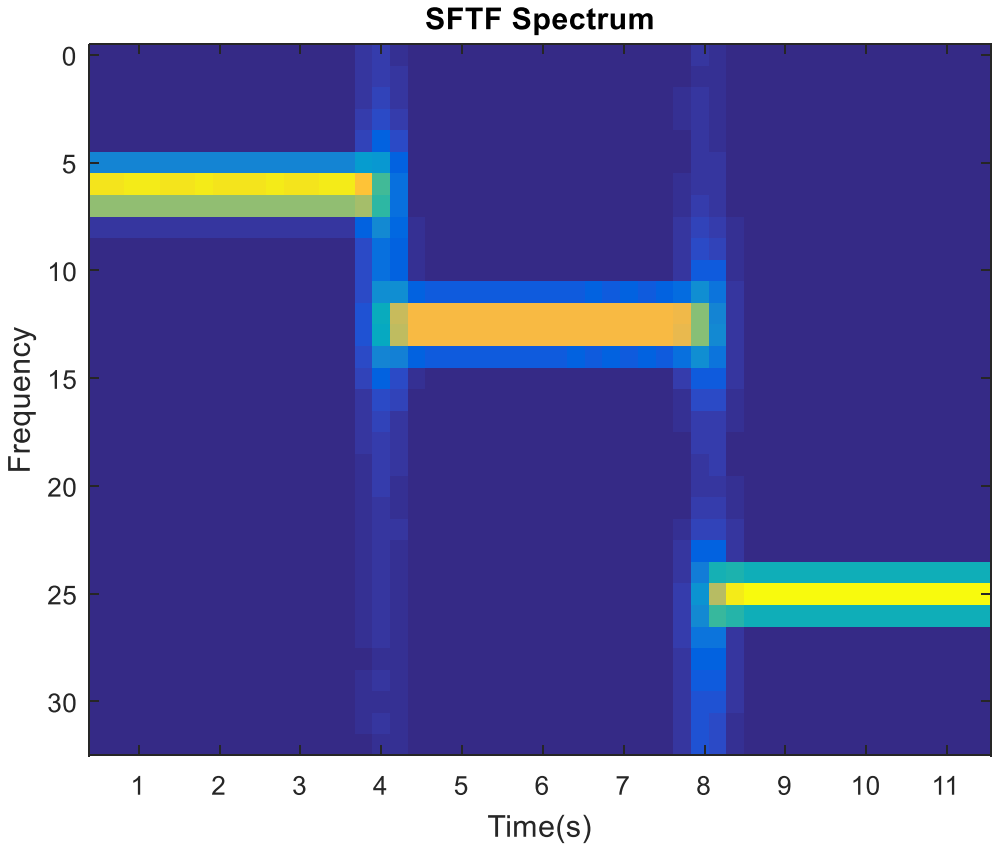
3.11

Repeat the same task after modifying the script so as that vector of DFT coefficients of subsequent frames are combined into a single matrix as its columns. Do not show partial results between processing subsequent frames. Just after processing all frames, plot the magnitudes of elements of the matrix as a image. Use the imagesc function. Interpret results and carefully assign labels to the axes of plots. Such a method for analyzing signals is called the short-time Fourier transform (STFT), and the plot You created is called spectrogram.

```
clear all

fs = 512;
A=1;
t1=0;
t2 = 0.5;
shift=0;
f1=50
s1=sinusoid( fs  ,f1 , A , t1 ,t2 , shift);

f2=100
s2=sinusoid( fs  ,f2, A , t1 ,t2 , shift);

f3=200
s3=sinusoid( fs  ,f3, A , t1 ,t2 , shift);

S=[s1,s2,s3]
lengthofOverlap=64;
index=1:lengthofOverlap;
h=hamming(lengthofOverlap)';
count=20;
spectrogram=zeros(lengthofOverlap,count);
for k=1:count
    frame=S(index+((k-1).*(lengthofOverlap/2)));
    window=frame.*h;
    spec=fft(window);
    spectrogram(:,k)=spec;
end
figure
imagesc(abs(spectrogram))
xlabel('Time(s)')
ylabel('Frequency')
title('Spectogram')
```



```
clear all

fs = 512;
A=1;
t1=0;
t2 = 0.5;
shift=0;
f1=50
s1=sinusoid( fs  ,f1 , A , t1 ,t2 , shift);

f2=100
s2=sinusoid( fs  ,f2, A , t1 ,t2 , shift);

f3=200
s3=sinusoid( fs  ,f3, A , t1 ,t2 , shift);

x=[s1,s2,s3]
lengthofOverlap=100;
w=hamming(lengthofOverlap)';

[S,F,T]=spectrogram(x,w,50,100,512)
imagesc(T,F,abs(S))
ylabel('Frequency')
xlabel('Time(s)')
title('SFTF Spectrum')
```



BIALYSTOK UNIVERSITY OF TECHNOLOGY

SIGNAL PROCESSING

TASK 4

TUTOR:

Dr inż. MAREK PARFIENIUK

STUDENTS:

AHMET TOREHAN HAYTOGLU

HARUN GULEC

TASK 4

4.1

Learn the following Matlab functions: filter, conv, impz, freqz.

filter: One-dimensional digital filter.

Y = filter(B,A,X) filters the data in vector X with the filter described by vectors A and B to create the filtered data Y. The filter is a "Direct Form II Transposed" implementation of the standard difference equation.

conv: Convolution and polynomial multiplication.

C = conv(A, B) convolves vectors A and B. The resulting vector is length MAX([LENGTH(A)+LENGTH(B)-1,LENGTH(A),LENGTH(B)]). If A and B are vectors of polynomial coefficients, convolving them is equivalent to multiplying the two polynomials.

impz: Impulse response of digital filter

[H,T] = impz(B,A) computes the impulse response of the filter.

freqz: Frequency response of digital filter

[H,W] = freqz(B,A,N) returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter.

4.2

Check if the systems described by the following difference equations

a) $y[n]=x[n-1] + x[2-n]$

b) $y[n]=2x[n-1] + 5$

are linear, time-invariant, causal, and stable. The systems can be checked using an arbitrary way (analytical calculations, reasoning, Matlab experiment), but a justification must be given in your report.

$y[n]=x[n-1] + x[2-n]$

Causal

The system is NOT causal because for any $n < 1$, the output depends on a future input, e.g. $y[0] = x[-1] + x[2]$.

Linear

The system is linear. Because;

$x[n] = \alpha x_1[n] + \beta x_2[n]$

$y[n] = x[n-1] + x[2-n]$

$= [\alpha x_1 + \beta x_2][n-1] + [\alpha x_1 + \beta x_2][2-n]$

$= \alpha x_1[n-1] + \beta x_2[n-1] + \alpha x_1[2-n] + \beta x_2[2-n]$

$= \alpha [x_1[n-1] + x_1[2-n]] + \beta [x_2[n-1] + x_2[2-n]]$

$= \alpha y_1[n] + \beta y_2[n]$

Time Invariant

The system is time invariant. To see this, we let y[n] be the output corresponding to the input x[n] and let $x_a[n] = x[n-a]$. Then the output $y_a[n]$ corresponding to the input signal $x_a[n]$ is

$$y_a[n] = x_a[n-1] + x_a[2-n]$$
$$= x[[n-a]-1] + x[2-[n-a]]$$
$$= y[n-a]$$

Stable

The system is stable because for a definite bounded input, we can get definite bounded output. E.g.

x[n−1]=2 and x[2−n]=5
y[n]=7

y[n]=2x[n−1] + 5

Causal

The system is causal because for any t < 1, the output NOT depends on a future input, e.g. y[0] = x[−1] + x[2].

Linear

The system is NOT linear. Because;

$x_a[n]=x_1[n]+x_2[n]$
$y_1[n]= 2x_1[n-1] + 5$
$y_2[n]= 2x_2[n-1] + 5$
$y_a[n] \neq y_1[n]+ y_2[n]$

Time Invariant

The system is time invariant. To see this, we let y[n] be the output corresponding to the input x[n] and let $x_a[n] = x[n-a]$. Then the output $y_a[n]$ corresponding to the input signal $x_a[n]$ is

$$y_a[n] = 2x_a[n-1] + 5$$
$$= 2x[[n-a]-1] + 5$$
$$= y[n-a]$$

Stable

The system is stable because for a definite bounded input, we can get definite bounded output. E.g.

x[n−1]=2
y[n]=9

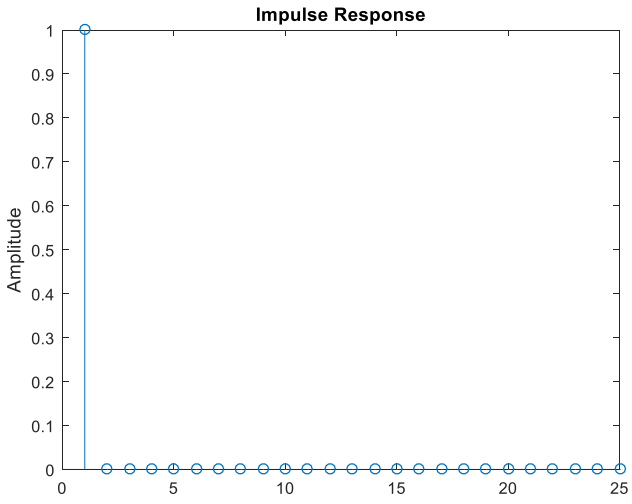
4.3

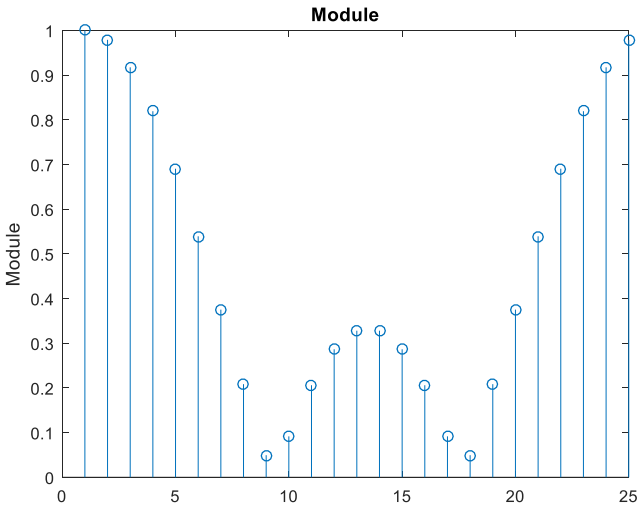
For the system described by the following difference equation

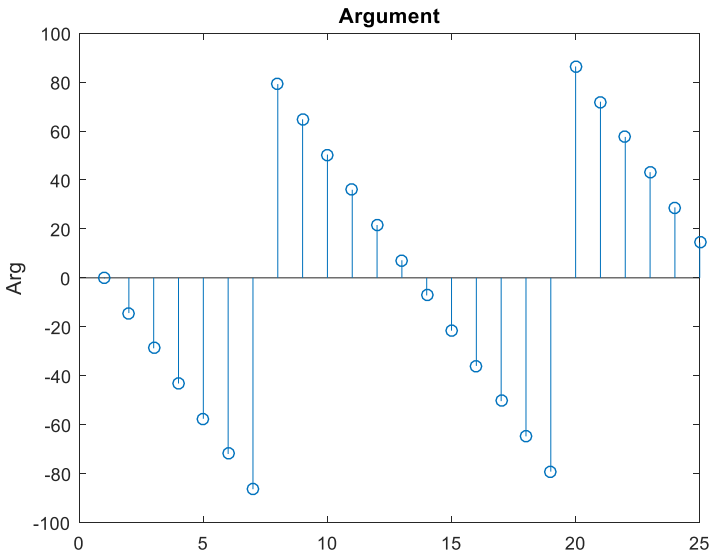
$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k]$$

show its impulse response and both modules and arguments of the coefficients of its Discrete Fourier Transform (DFT). Consider N = 2, 3, 4, and 9.

```
clear all
n=[5 6 7];
N=3;
x=1:20;
B=1/N.*ones(1,3);

x = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ];
y=filter(B,1,x);
figure
stem(x) %impulse
title('Impulse Response')
ylabel('Amplitude')
k=fft(y);
figure
stem(abs(k))
title('Module')
ylabel('Module')
figure
arg=atan(imag(k)./real(k)).*(360/(2*pi));
stem(arg)
title('Argument')
ylabel('Arg')
```



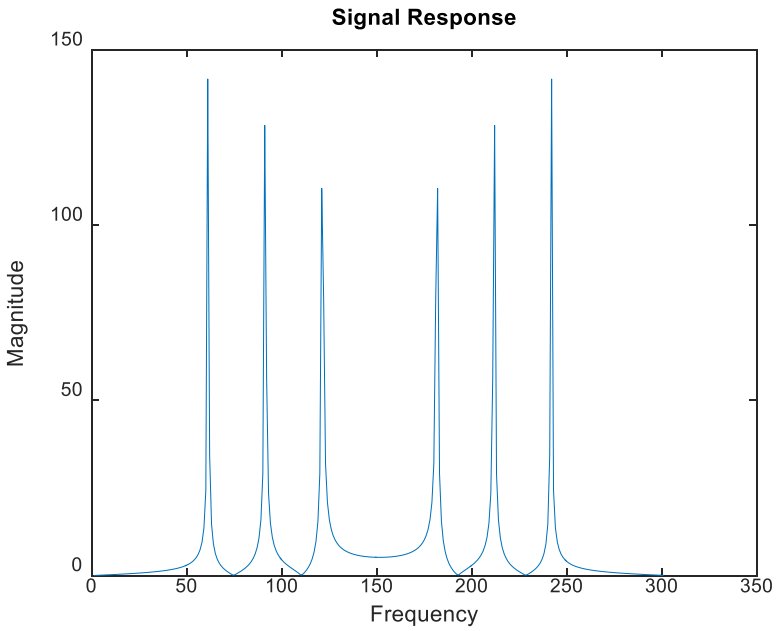


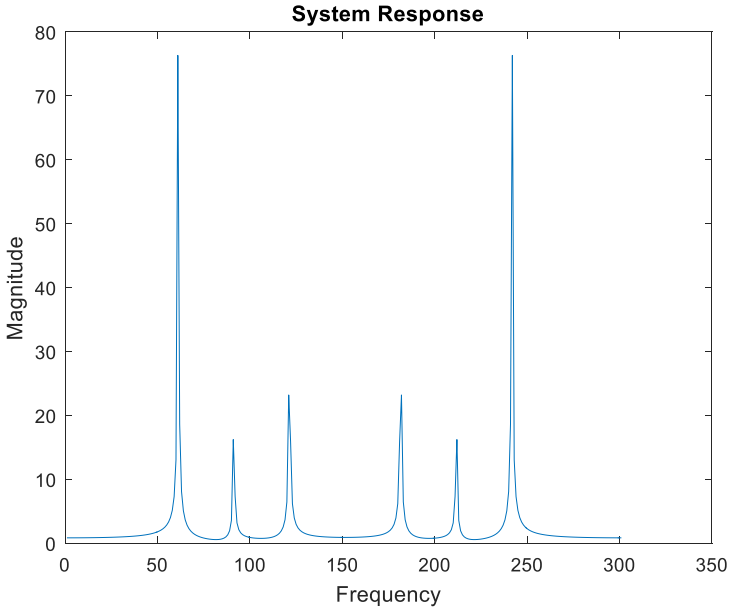


4.4

Generate several sinusoids whose frequencies are equally spaced over the range from 0 to Fs. Then, add these sinusoids so as to form a single signal. Pass this signal through (using the filter function) the system considered in the previous problem. Compare the DFT of a signal and system responses, so as to conclude how the signal spectrum has been modified by the system.

```
clear all
clc

fs=100;
N=3;
B=1/N.*ones(1,3);

f=[20 30 40];
y=sinusoid(fs ,f(1) ,1 ,0 ,3 ,0);
for ii=2:length(f)
    y=y+sinusoid(fs ,f(ii) ,1 ,0 ,3 ,0);
end

filtered=filter(B,1,y);

figure
plot(abs(fft(y)))
title('Signal Response')
ylabel('Magnitude')
xlabel('Frequency')

figure
plot(abs(fft(filtered)))
title('System Response')
ylabel('Magnitude')
xlabel('Frequency')
```





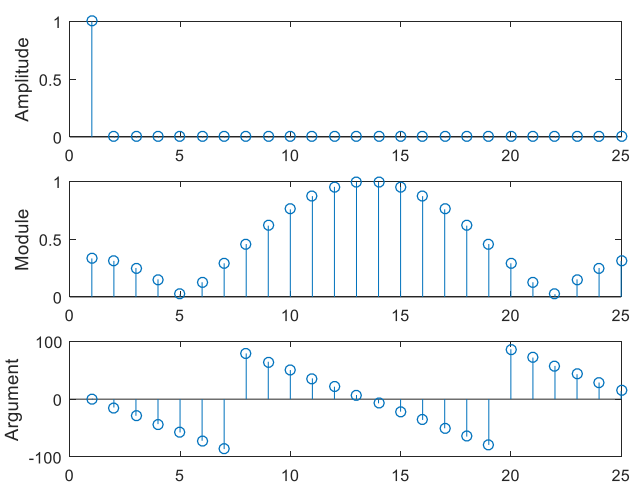
4.5

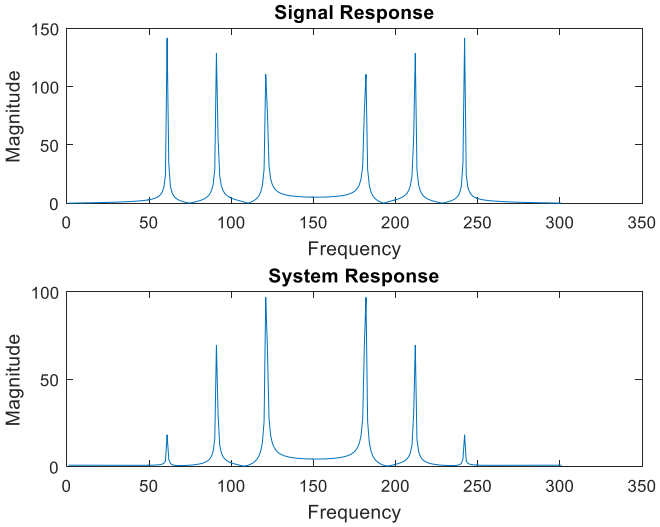
Repeat solving both previous problems after changing the difference equation for

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1}(-1)^k x[n-k]$$

Compare the results against the results of those problems.

```
clear all

N=3;
fs=100;
o=ones(1,N);
B=1/N.*o;
for ii=1:N
    B(ii)=(-1).^(ii-1).*B(ii);
end

x = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ];
y=filter(B,1,x);

figure
subplot(3,1,1)
stem(x) %impulse
ylabel('Amplitude')
k=fft(y);
subplot(3,1,2)
stem(abs(k))
ylabel('Module')
subplot(3,1,3)
arg=atan(imag(k)./real(k)).*(360/(2*pi));
stem(arg)
ylabel('Argument')

f=[20 30 40];
S=sinusoid(fs ,f(1) ,1 ,0 ,3 ,0);
for ii=2:length(f)
    S=S+sinusoid(fs ,f(ii) ,1 ,0 ,3 ,0);
end

filtered=filter(B,1,S);

figure
subplot(2,1,1)
plot(abs(fft(S)))
title('Signal Response')
ylabel('Magnitude')
xlabel('Frequency')

subplot(2,1,2)
plot(abs(fft(filtered)))
title('System Response')
ylabel('Magnitude')
xlabel('Frequency')
```





4.6

Develop Your own Matlab function for convolving 1D sequences of values/samples, by using loop-related Matlab instructions. Check if Your function is correct, by processing simple short sequences of samples.

```
function w= MyConv( u,v )
    m = length(u);
    n = length(v);
     for k=1:(n+m-1)
         w(k)=0;
         for j=max(1,k+1-n):1:min(k,m)
             w(k)=w(k)+u(j)*v(k-j+1);
         end
    end
end
```

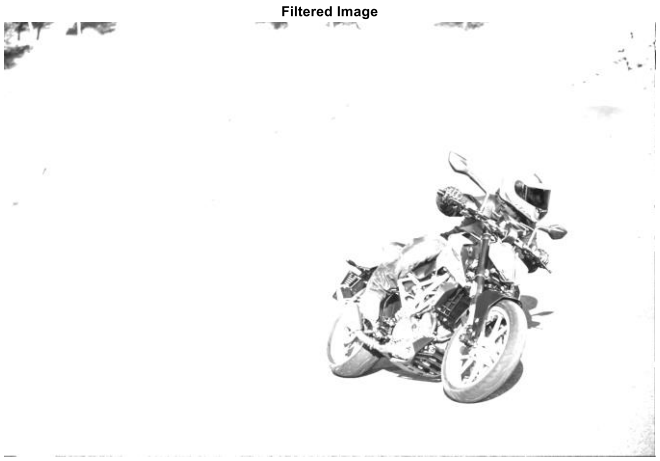
4.7

Using the function You develop when solving the previous problem, filter a gray-scale image using the 1-dimensional filter determined by the following equation

$$y[n] = x[n + 1] - 2x[n] + x[n - 1]$$

Firstly, filter rows (or columns) of the original image, and then filter columns (or rows) of the image that resulted from the first filtering. The filter should work as an edge detector.

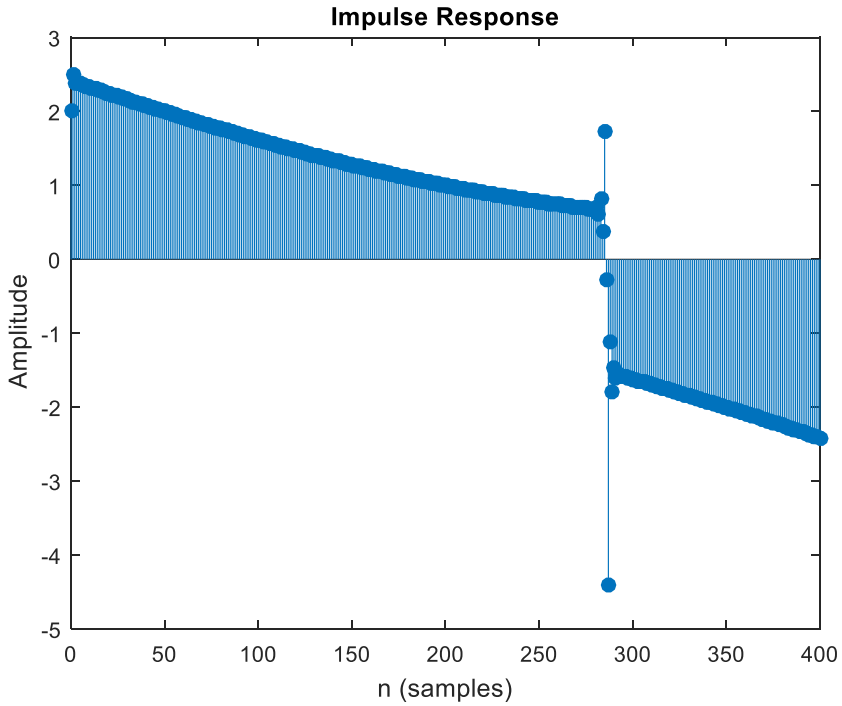
```
clear all

RGB=imread('Duke390.jpg');
gray=rgb2gray(RGB);
figure
imshow(gray)
title('Original Image')
[rows columns] = size(gray);
r=[1 -2 1];
for ii=1:rows
    y=MyConv(gray(ii,:),r);
    gray(ii,:)=y(2:length(gray(ii,:))+1);
end
for ii=1:columns
    y=MyConv(gray(:,ii),r)
    gray(:,ii)=y(2:length(gray(:,ii))+1);
end
figure
imagesc(gray);
title('Filtered Image')
```



4.8

Write the difference equation that describes the system in fig. 1. Develop a Matlab script that shows the impulse response of the system (several dozens of first coefficients) for several combinations of values of a and b in the range −2 do 2. What depends on coefficient values? Is always the impulse response infinite?

```
clear all

b=-2:0.01:2;
y(1)=b(1);
a=1;
for ii=2:size(b,2)
    z(ii)=1/y(ii-1);
    y(ii)=a*z(ii)+b(ii);
end
impz(y*-1)
```



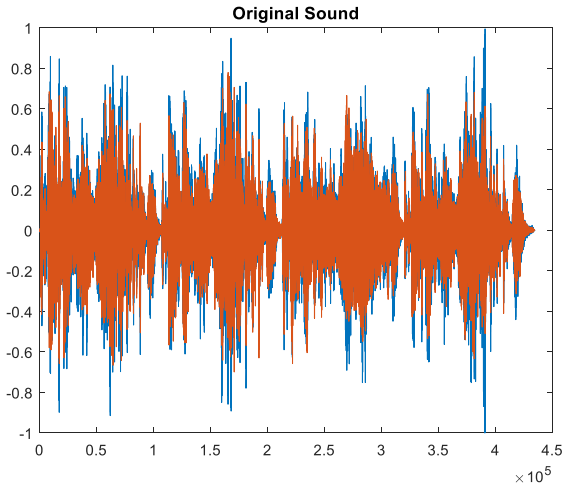
4.9

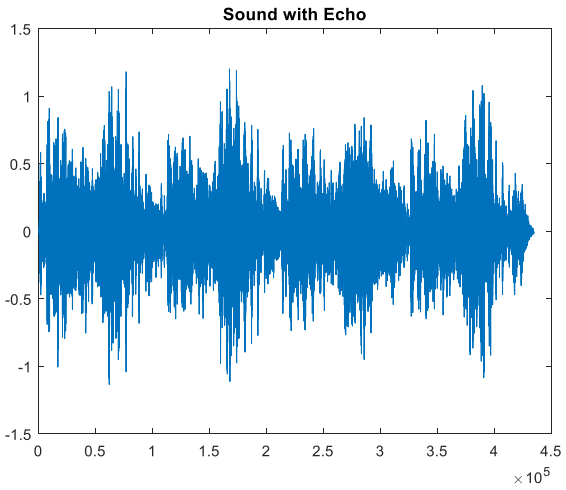
Develop a Matlab script that demonstrates how to generate echo by using the following equation

$$y[n] = x[n] + ax[n - D]$$

The D parameters determines the delay. In order to make echo audible, set the delay $\tau = D/fs$ larger than 40 ms. For large values of the sampling frequency, fs, so large delays are equivalent to values of D of order of several thousands. The value of 0 <a< 1 should not be very low. Use Your script to process an utterance with several words. Register the utterance by yourself, using a microphone, or look for utterances in WWW. Try several combinations of values of echo parameters. For one of the combinations, plot both original signal and its variant with echo.

```
clear all

[x,Fs]=audioread('tone.wav');

figure
plot(x)
D = 6000;
T=D/Fs
a=0.75;

for ii=1:length(x)
    if (ii > D)
        echo = a*x(ii-D);
    else
        echo = 0;
    end
    y(ii)=x(ii)+echo;
end
figure
plot(y)
sound(y,Fs)
```


Original Sound
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
×10^5


Sound with Echo
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
×10^5

BIALYSTOK UNIVERSITY OF TECHNOLOGY

SIGNAL PROCESSING

TASK 5

TUTOR:

Dr inż. MAREK PARFIENIUK

STUDENTS:

AHMET TOREHAN HAYTOGLU

HARUN GULEC

TASK 5

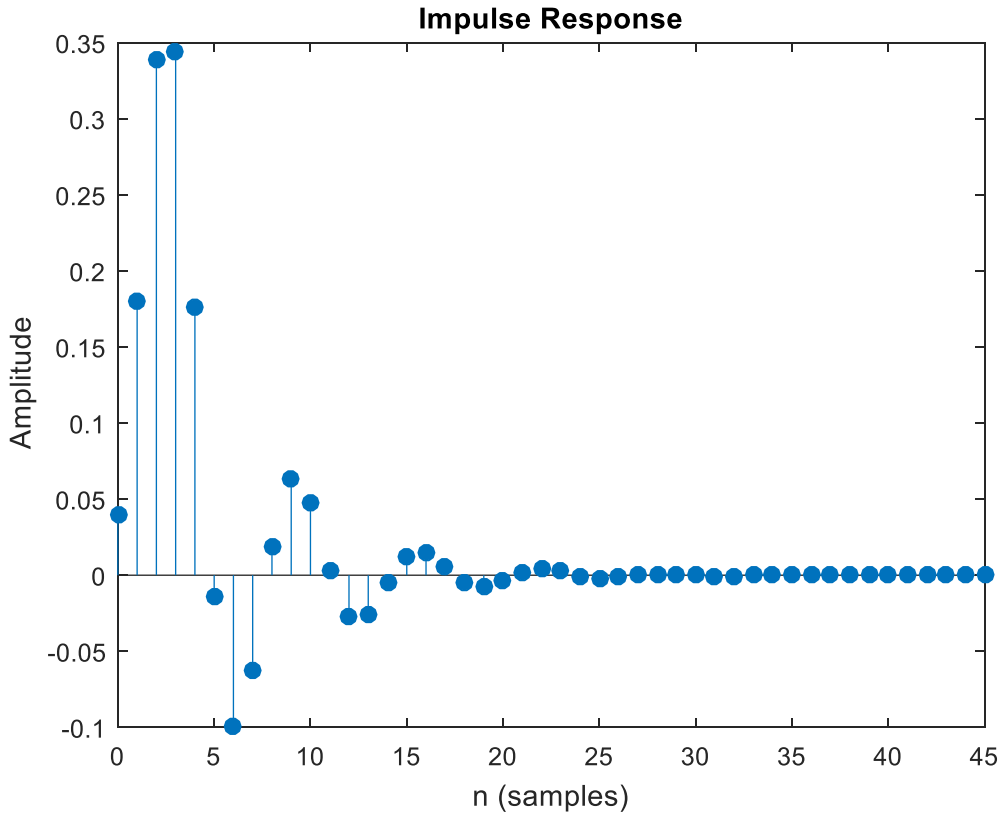
5.1

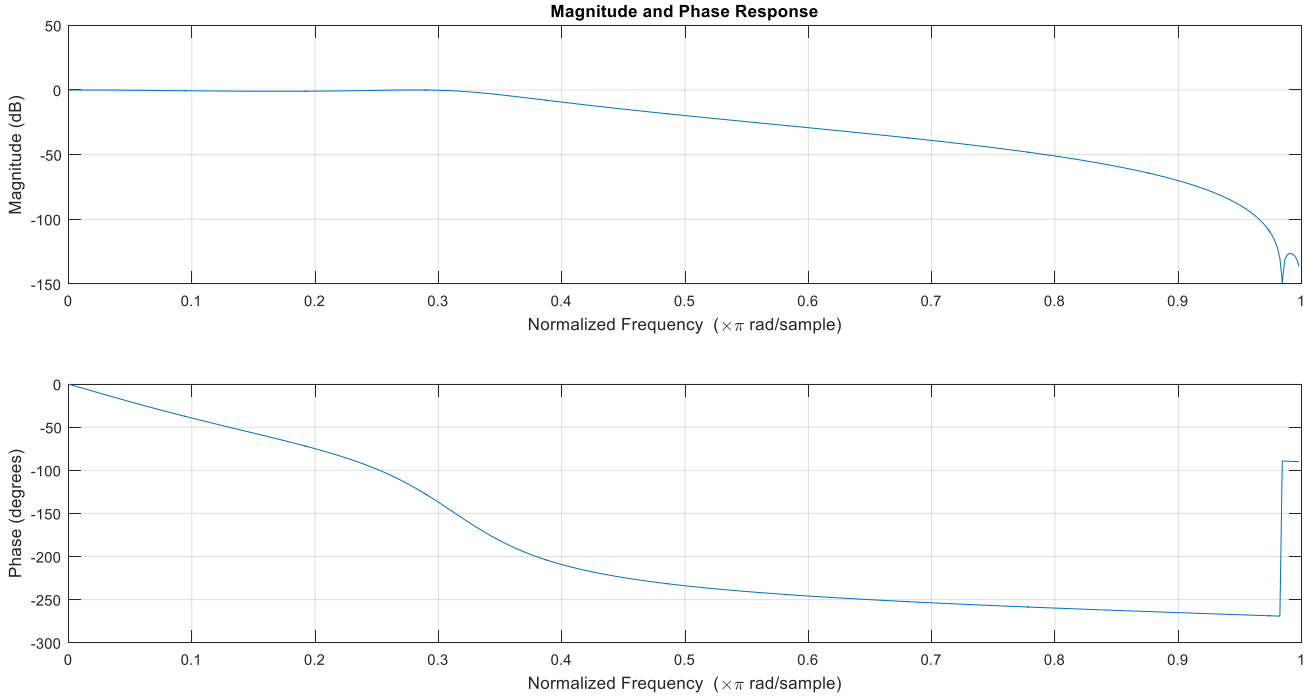
For the filter whose transfer function is

$$H(z) = \frac{0.0403 + 0.1208z^{-1} + 0.1208z^{-2} + 0.0403z^{-3}}{1 - 1.4726z^{-1} + 1.1715z^{-2} - 0.3767z^{-3}}$$

write the difference equation, and plot its impulse response, magnitude response, and phase response.

```
clear all

h_num = [0.0403 0.1208 0.1208 0.0403];
h_den = [1 -1.4726 1.1715 -0.3767];

figure
impz(h_num, h_den)

figure
freqz(h_num, h_den)
title('Magnitude and Phase Response')
```

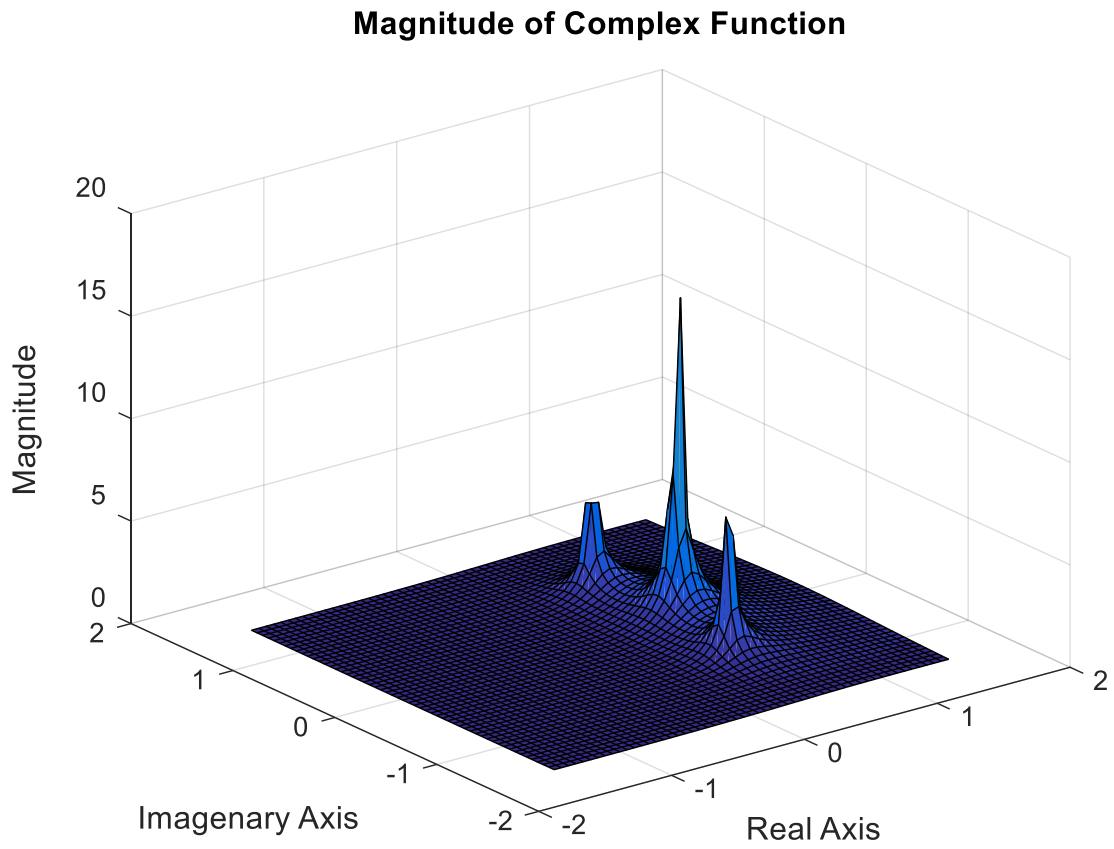




5.2

Draw a 3D plot that shows the magnitude of the complex function described by H(z), the transfer function You have determined when solving the previous problem. Consider a square fragment of the complex plane, so as to encompass the unit circle, eg. $|\mathrm{Re}\{z\}| \leq 1.5$ and $|\mathrm{Im}\{z\}| \leq 1.5$. A plot looks better if it comprises several hundreds of points. The following functions should be helpful: meshgrid, ndgrid, surf, polyval.

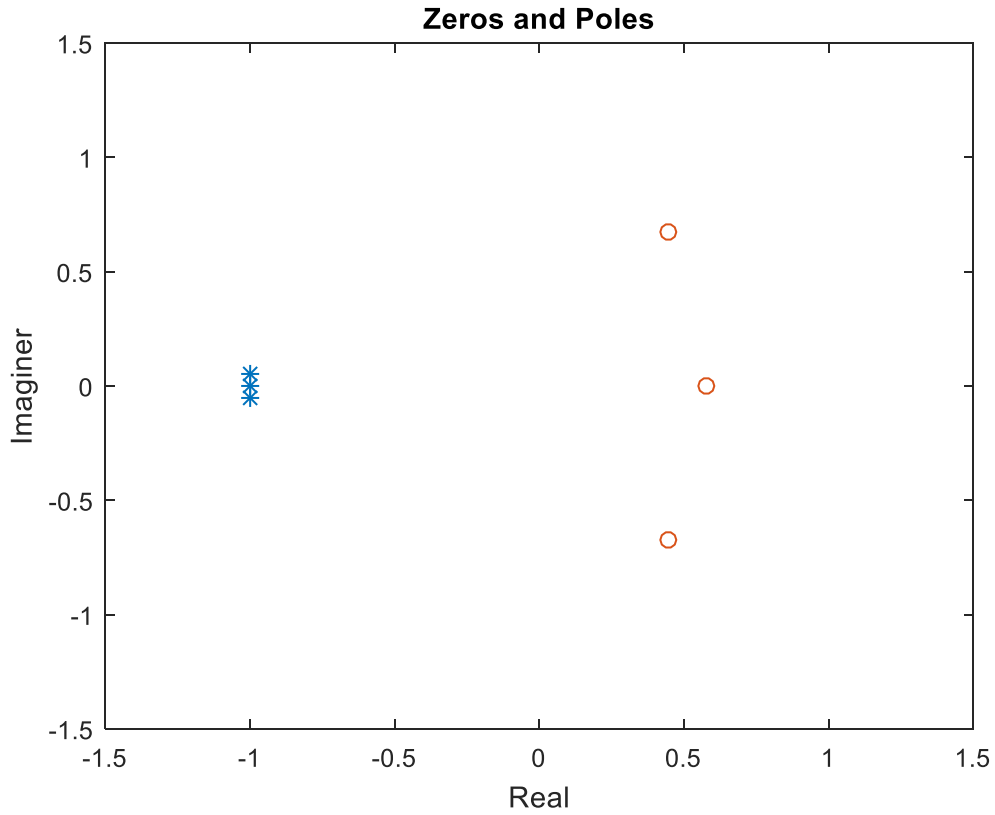
```
clear all

[zRe, zIm] = meshgrid(-1.5 : 0.055 : 1.5, -1.5 : 0.055 : 1.5);

z = zRe + 1i .* zIm;

H  = 0.0403 + 0.1208 .* (z .^ -1) + 0.1208  .* (z .^ -2)  + 0.0403 .* (z .^ -3);
H = H ./ (1 - 1.4726 .* (z .^ -1) + 1.1715  .* (z .^ -2) - 0.3767  .* (z .^ -
3));

surf(zRe,zIm,abs(H));
xlabel('Real Axis')
ylabel('Imagenary Axis')
zlabel('Magnitude')
title('Magnitude of Complex Function')
```



5.3

Determine the poles and zeros of H(z), and use them to write the equation for transfer function. Illustrate how are the poles and zeros located in the complex plane. Use the following functions: roots, poly.

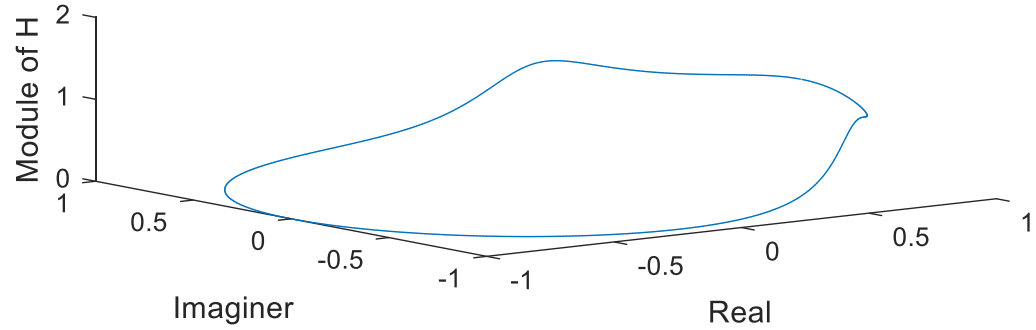
```
h_Num=[0.0403 0.1208 0.1208 0.0403];
h_Den=[1 -1.4726 1.1715 -0.3767];
H= filt (h_Num,h_Den)

bode(H)

Nr=roots(h_Num)
plot(Nr, '*')
Dr=roots(h_Den)
hold on
plot(Dr , 'o')
axis([-1.5 1.5 -1.5 1.5])
xlabel('Real')
ylabel('Imaginer')

den=poly(Dr);
num=poly(Nr);
Hz=num./den;
```

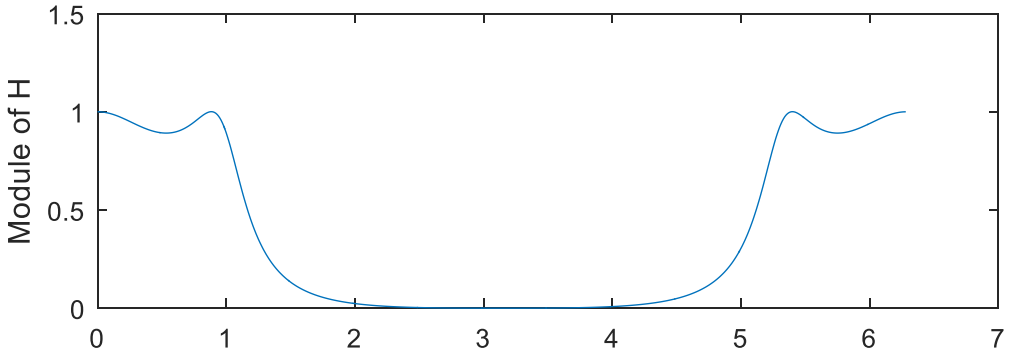


5.4

Compute values of H(z) for several dozens – hundreds of points on the unit circle, z = ejω. Plot modules and arguments of the values You have computed. They should be showed both in 3 dimensions, as s function of z, and in 2 dimensions, as a function of ω.

```
clear all

w=0:0.01:2*pi;

z = exp(1i.*w);

H  = 0.0403 + 0.1208 .* (z .^ -1) + 0.1208  .* (z .^ -2)  + 0.0403 .* (z .^ -3);
H = H ./ (1 - 1.4726 .* (z .^ -1) + 1.1715  .* (z .^ -2) - 0.3767  .* (z .^ -3));

figure
subplot(2, 1, 1)
plot3(real(z),imag(z),abs(H))
xlabel('Real')
ylabel('Imaginer')
zlabel('Module of H')

subplot(2, 1, 2)
plot(w, abs(H))
ylabel('Module of H')
```

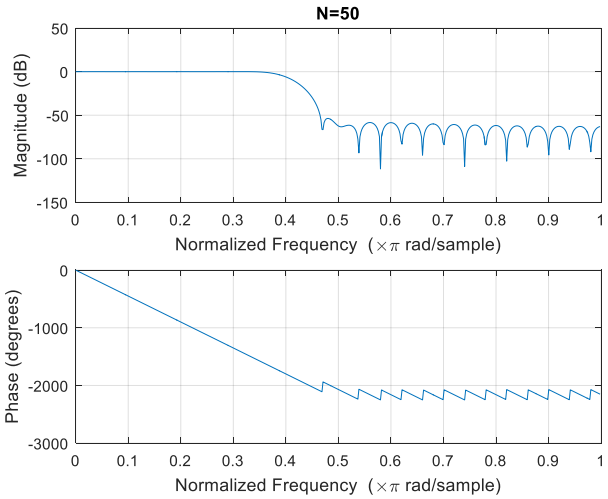


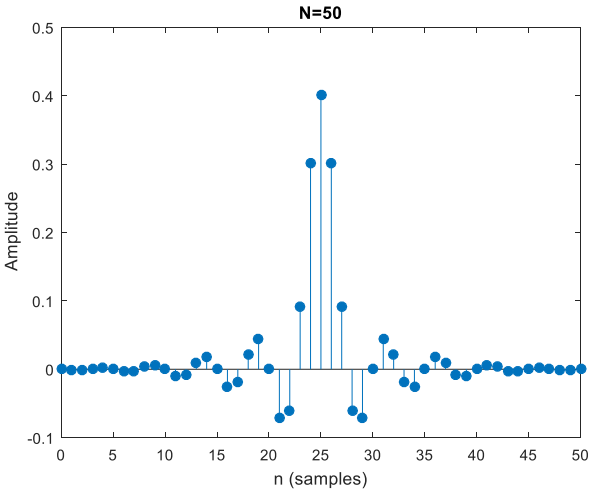


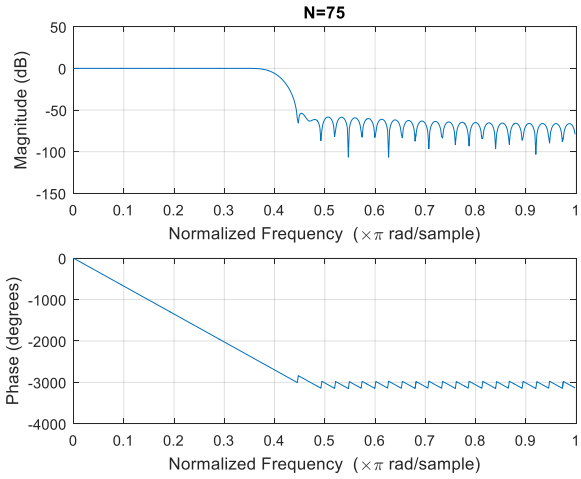
5.5

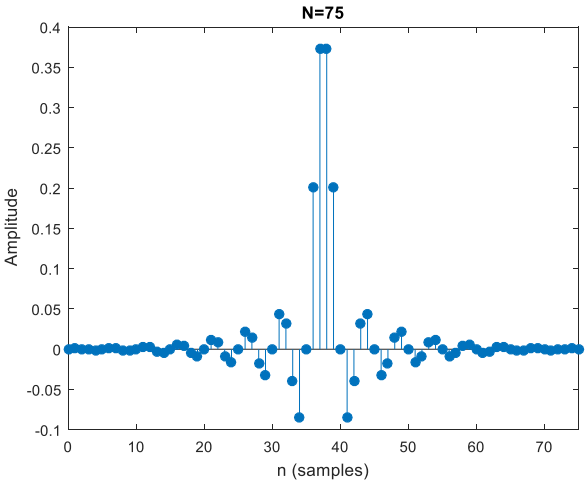
Design a low-pass FIR filter whose cut-off frequency is 1000 Hz for the sampling frequency of 5000 Hz. Use the windowing method, or the fir1 function. Plot the impulse response, magnitude response, and phase response. Compare filters obtained using 3 window functions as well as filters with the orders 50, 75, and 100, for a single window function. What is determined by the window function, and what is determined by the order of a filter.

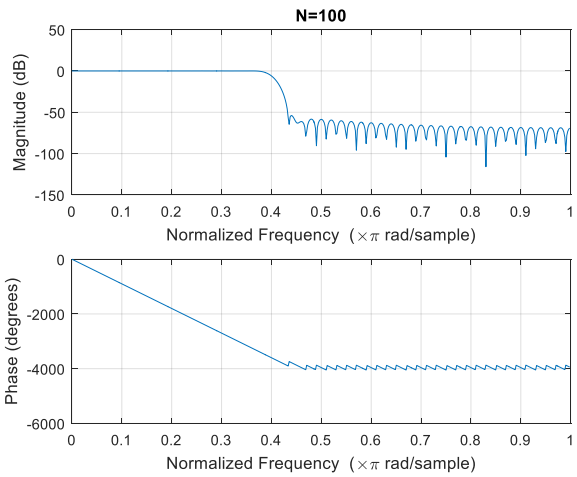
```
clear all
Fs=5000;
Ts=1/Fs;
Ns=512;
t=[0:Ts:Ts*(Ns-1)];
f1=1000;
x=sin(2*pi*f1*t);
grid on;
N=50;
N1=75;
N2=100;
W=[0.4];
wndFunction = @hamming;
B=fir1(N,W, wndFunction(N + 1));
C=fir1(N1,W, wndFunction(N1 + 1));
D=fir1(N2,W, wndFunction(N2 + 1));
A=1;
freqz(B,A);
title('N=50')
figure
freqz(C,A);
title('N=75')
figure
freqz(D,A);
title('N=100')
figure
impz(B, A);
title('N=50')
figure
impz(C, A);
title('N=75')
figure
```

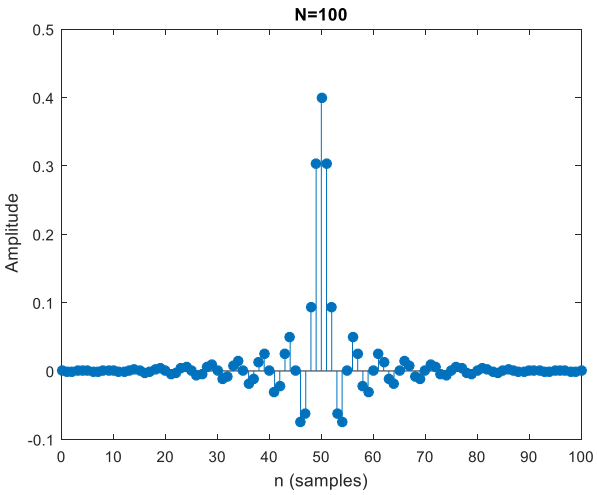

N=50
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)


N=50
Amplitude
n (samples)


N=75
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)


N=75
Amplitude
n (samples)


N=100
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)


N=100
Amplitude
n (samples)