ELEC 342 Lab 4: Functions in MATLAB and the Sampling Theorem

Name :	JUNPENG GAI
ID number:	40009896
Course number :	ELEC342-X Lab 2224
Date performed:	Monday, 20 March 2023
Due Date:	Monday, 3 April 2023
Lab Instructor Name:	Salameh, Ahmed

"I certify that this submission is my original work and meets the Faculty's Expectations of Originality"

盖俊鹏

Contents

1.Objectives

The objectives of this lab are to get familiar with MATLAB functions and how to use function. In the second part of this lab we will use the function and 2 audio files to practice how to use filters to recover the signal.

2.Theory

1. Functions: In the MATLAB we can use function to simplify our scripts so we can reduce the repeated part. Also, there are 2 ways to pass the parameters: pass by reference and pass by value.
2. Global variables: in this lab we will need to use global variables. If a variable is visible to all the functions then use the keyword global is a good choice.
3. Sampling theorems: We will use different step interval and frequency to sample the signal in the lab section. We can see the different results when we applied different sample frequencies.

3.Tasks/Results/Discussion

3.1 Functions

We created this function to take a variable and add one to it.

```
%name:junpeng gai
%sid:40009896
function inc_x = inc(x)
% inc(x) increments the value of x by 1
x = x + 1 ; % since we are modifying the argument
 % within the function, it will be passed
 % by value and only a copy of the argument
 % will be modified
disp("The value of parameter x from within inc is : ");
disp(x);
% return the incremented value
inc_x = x ;
```

So, when we call this function in another script we can see that the result will be add by 1;

```
%name:junpeng gai
%sid:40009896

inc(4);
```

```
The value of parameter x from within inc is :

    5
```

3.2 Pass by value (no changing the original data)

```
%name:junpeng gai
%sid:40009896
function array_squared = array_square(x)
% array_square(x) - returns an array containing the square of
% the elements
% in the passed array x
array_squared = x .^ 2 ;
```

We wrote the function array_square();

Then we give the input and compare the x before and after calling the function.

```
%name:junpeng gai
%sid:40009896
x=3;
array_square(x)
x
```

```
ans =

    9

x =

    3
```

We can see that the value doesn't change.

Pass by reference (Changing the original data)

We wrote the following function:

```
function array = clear_odd2(x)
% clear_odd(x) sets the odd elements of the array x to 0
for index = 1 : 2 : length(x)
 x(index) = 0; % since we are modifying value of argument

end % it will be a pass-by-value argument and
 % we are working only with the argument
array = x; % return the modified copy
```

We run the following code

```
%name:junpeng gai
%sid:40009896
clear
y = [ 1 : 5 ]
y = clear_odd2(y)
```

```
y =

    1    2    3    4    5

y =

    0    2    0    4    0
```

We can see that the original data has been change so this method is pass by reference.

Use for loop to invoke inc function 5 times

If we define the function as following:

```
function out = count_me_up
% adds 1 to a local variable initialized to 0 and
% returns this value
count = 0 ;
count = count + 1 ;
out = count;
```

We invoke it 5 times:

```
%name:junpeng gai
%sid:40009896
for i=(1:5)
count_me_up
end
```

```
ans =

     1

ans =

     1

ans =

     1

ans =

     1

ans =

     1
```

We can see that in stead of 1,2,3,4,5 we saw MATLAB print same output 5 times. This is because the count is not consistent, we modified our code to the following pattern.

```
function out = count_me_up
% adds 1 to a persistent local variable initialized to 0 and
% returns this value
persistent count ;
% check if count is the empty array and set it to 1 if yes
 if isempty(count)
 count = 0; % note this will be performed only 1 time
 end
count = count + 1 ;
out = count;
```

Then we run code:

```
%name:junpeng gai
%sid:40009896

for i=(1:5)
count_me_up
end
```

```
ans =

     1

ans =

     2

ans =

     3

ans =

     4

ans =

     5
```

Now the outputs are the same as we expected.

3.3 Global variables

We defined 2 function count_up and count_down

```
function [] = count_up
% add 1 to the global variable counter
global counter; % declare counter as a global variable
counter = counter + 1;
```

```
function [] = count_down
% subtracts 1 to the global variable counter
global counter; % declare counter as a global variable
counter = counter - 1;
```

We run the following code:

```
clear;


global counter ; % declare counter to be global
counter = 0 ;
disp('Value of counter = ');
disp(counter);
```

```
count_up;
count_up;
disp('Value of counter = ');
disp(counter);
count_down;
disp('Value of counter = ');
disp(counter);
```

```
Value of counter =

    0

Value of counter =

    2

Value of counter =

    1
```

We can see that our code will call 2 functions and print the value, as counter is a global variable, so MATLAB will use the same variable for both functions and our output is:

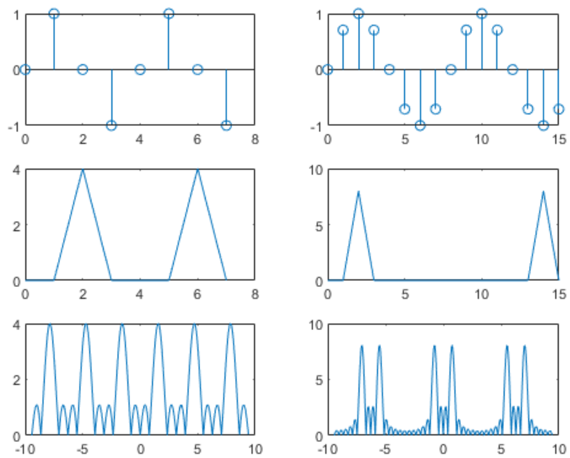
0->1(count up) ->2(count up) ->1(count down)

3.4 Sampling theorems

We will use different sampling frequency to sample the signal : x(n) = sin((2*pi)/N * n)

```
%JUNPENG GAI
%40009896
w1=[ - (3 ) * pi : StepSize : (3 )* pi];
w2=[ - (3 ) * pi : StepSize : (3 )* pi];
N = 4 ;
n1 = [ 0 : 2*N-1 ]
x1 = sin((2*pi)/N * n1);
X1=abs(fft(x1))
XX1=ft(x1,w1,n1);
subplot(3,2,1)
stem(n1,x1)
subplot(3,2,2)
N = 8 ;
n2 = [ 0 : 2*N-1 ]
x2 = sin((2*pi)/N * n2);
X2=abs(fft(x2));
XX2=ft(x2,w2,n2);
stem(n2,x2)

subplot(3,2,3)
plot(n1,X1)
subplot(3,2,4)
plot(n2,X2)
subplot(3,2,5)
plot(w1,abs(XX1));
```

```
subplot(3,2,6)
plot(w2,abs(XX2));
```



Published with MATLAB® R2020a

We get 2 signals, one with 4 sample/ period (Ts=T/4) and the other one is 8 samples per period (Ts=T/8);

Also, we can lot the fft and DTFT graph in the same figure.

4.Question

4.1 Question 1

4.1.1 Question 1 a

The program is to ask the user to input the value of the sampling rate (in terms of the number of times of the Nyquist rate, i.e. 2 times, 3.6 times, etc). The program then plots in one figure window the sampled signal (using stem) and the Fourier transform of the sampled signal. This is to be repeated 5 times.

```
%Name :JUNPENG GAI
%SID:40009896

promote1="input the number of periods \n";
promote2="input the step size of the frequency interval\n";
NumberOfPeriod=input(promote1);
StepSize=input(promote2);
global w;
w=[ - (NumberOfPeriod ) * pi : StepSize : (NumberOfPeriod ) * pi ];
for loop = 1 : 5
 promote3="input the sampling rate(in terms of the number of times of the Nyquist rate, i.e. 2 times, 3.6 times, etc)\n";
 SamplingRate=input(promote3);
 N= floor(2*SamplingRate);
 n= [0:2*N-1];
 x = sin ( 2*pi/N * n);
 figure
 subplot(2,1,1)
 hold on
   title(['Sampled sine wave at sampling rate = ', num2str(SamplingRate),' times the Nyquist rate'])
   xlabel('n')
   ylabel('x[n]')
   stem(n,x);
   hold off
 sum=ft(x);
subplot(2,1,2)

 hold on
   title(['FT with step size', num2str(StepSize)])
   xlabel('w/frequency')
   ylabel('X(w)')
   sum=abs(sum);
   plot(w,sum)
   hold off
 %compute the fourier transform of the signal (by passing x as an argument to
 %your function);
 %plot the transform of the signal;
end
```

We use 2 times,4 times,8 times,16 times,32 times as inputs:

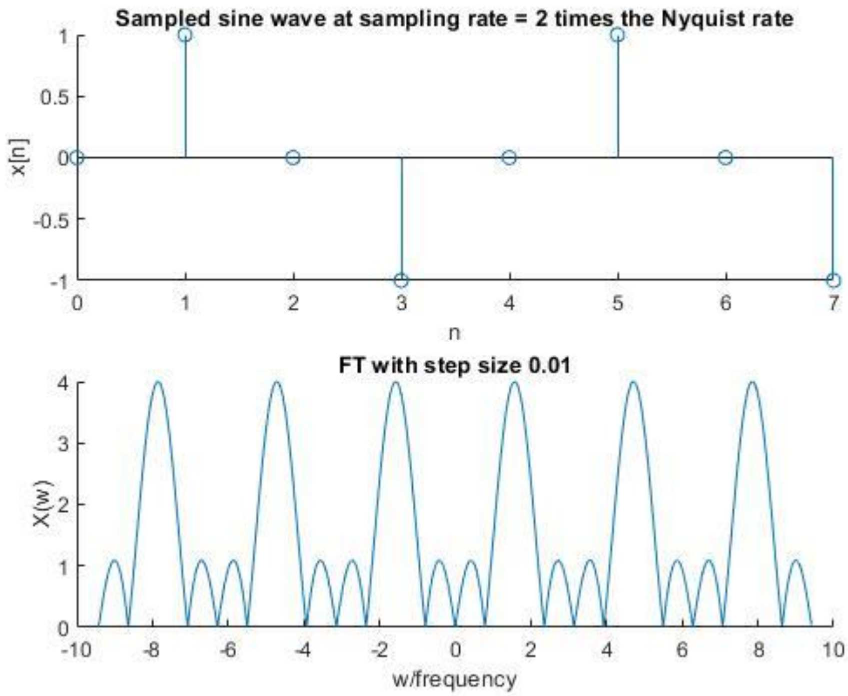


Figure 1 2times Nyquist rate

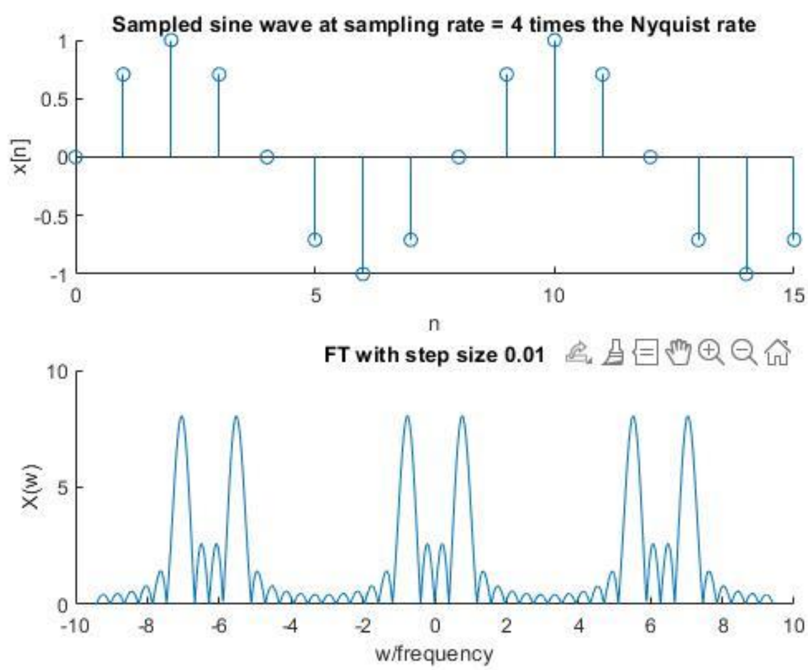


Figure 2 4times Nyquist rate

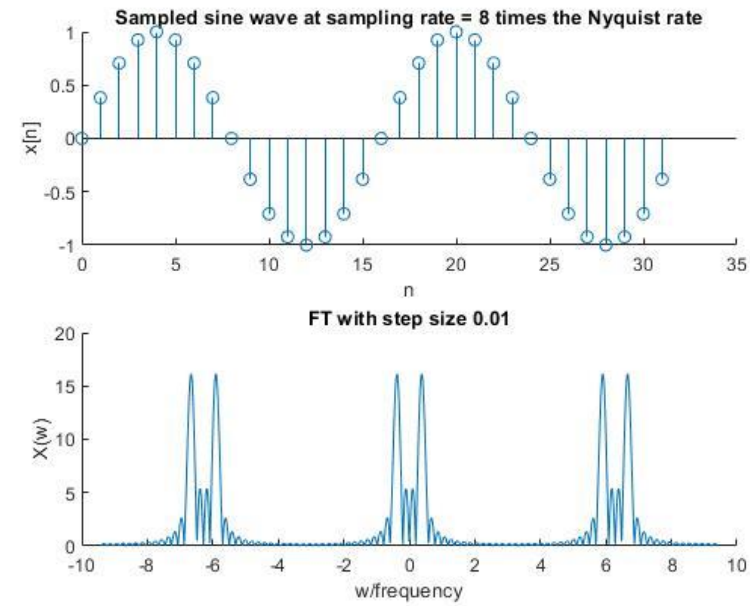


Figure 3 8times Nyquist rate

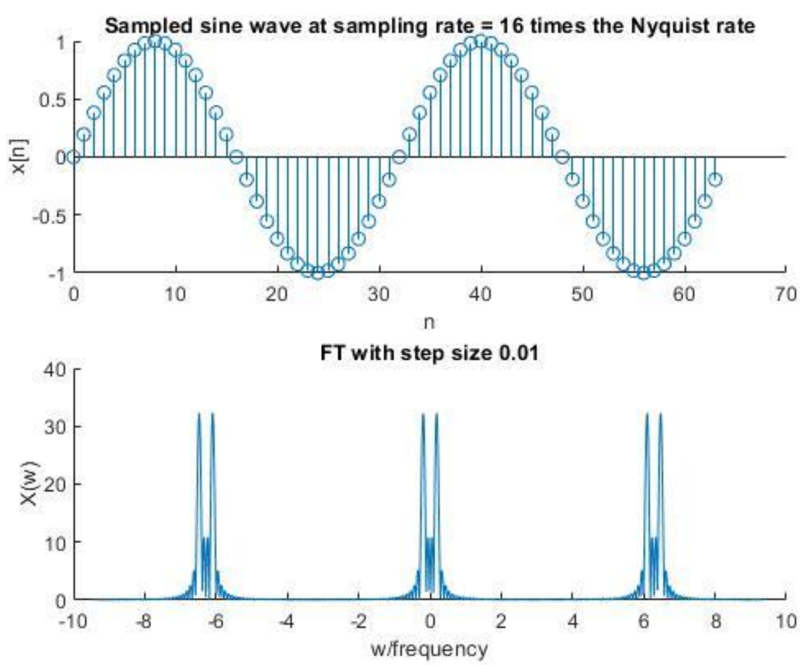


Figure 4 16times Nyquist rate

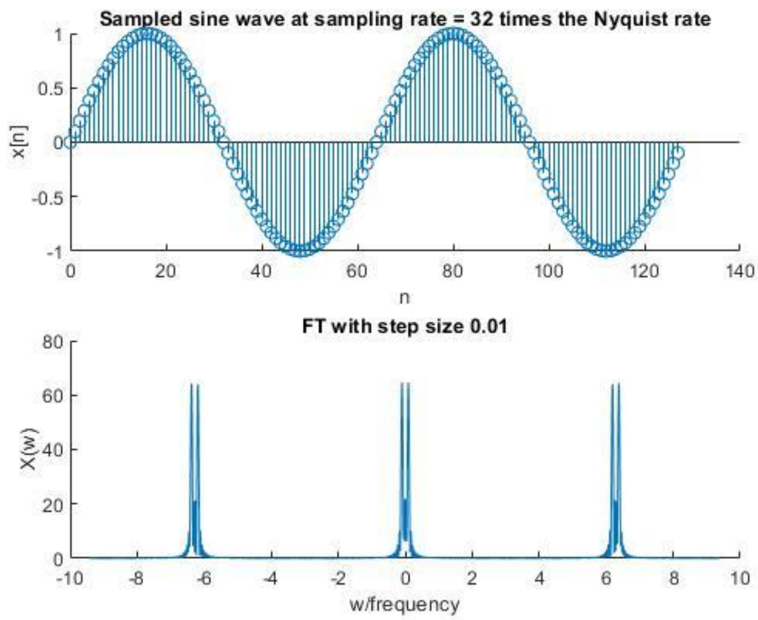


Figure 5 32times Nyquist rate

We can see from the graph with the sampling rate increasing from 2times (64 samples) to 32times (64 samples), the Fourier transform become more and more close to the theoretical graph, which are 2 impulse at 0 and 2pi.

4.1.2 b

This sampling rate will be kept constant and the program is to ask the user to enter the window size (in terms of the number periods of the signal). The sampled signal (over the total number of periods as defined by the value of the window 12 size) and its Fourier transform is to be plotted in one figure window. This is to be repeated 5 times with a different value for the window size.

```
%Name :JUNPENG GAI
%SID:40009896
 promote3="input the sampling rate(in terms of the number of times of the Nyquist rate, i.e. 2 times, 3.6 times, etc)\n";
 SamplingRate=input(promote3);
 N= floor(2*SamplingRate);
global  w

for loop = 1 : 5
promote1="input the number of periods \n";
```

```
NumberOfPeriod=input(promote1);
 n= [0:N*NumberOfPeriod-1];
w=[ - (NumberOfPeriod ) * pi : 0.05 : (NumberOfPeriod ) * pi ];
 x = sin ( 2*pi/N * n);
 figure
 subplot(2,1,1)
 hold on
    title(['over window size',num2str(NumberOfPeriod)])
    xlabel('n')
    ylabel('x[n]')
    stem(n,x);
    hold off
 sum=ft(x);
subplot(2,1,2)

 hold on
    title(['DTFT'])
    xlabel('w/frequency')
    ylabel('X(w)')
    sum=abs(sum);
    plot(w,sum)
    hold off
 %compute the fourier transform of the signal (by passing x as an argument to
 %your function);
 %plot the transform of the signal;
end
```

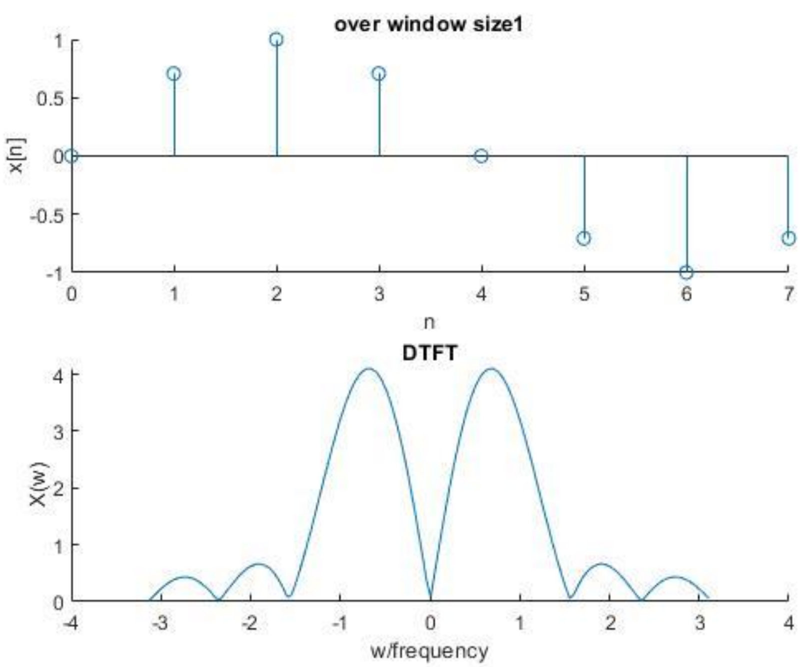


Figure 6 Window size of 1

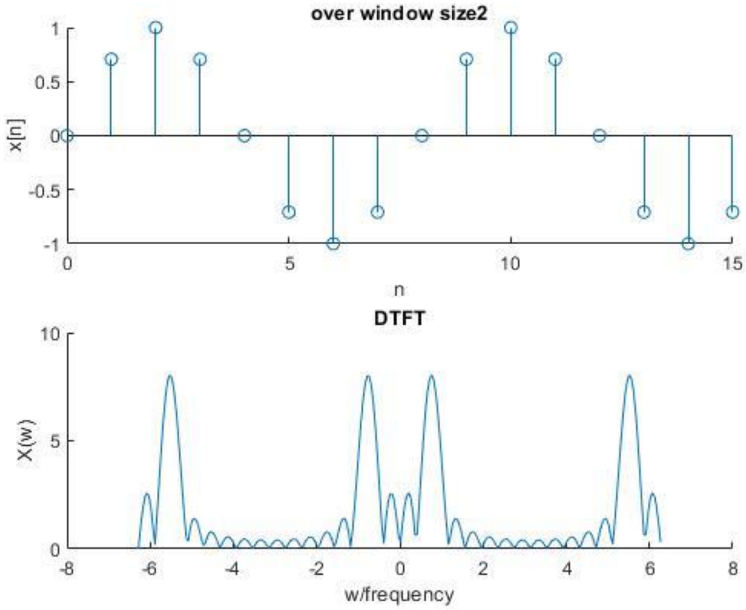


Figure 7 Window size of 2

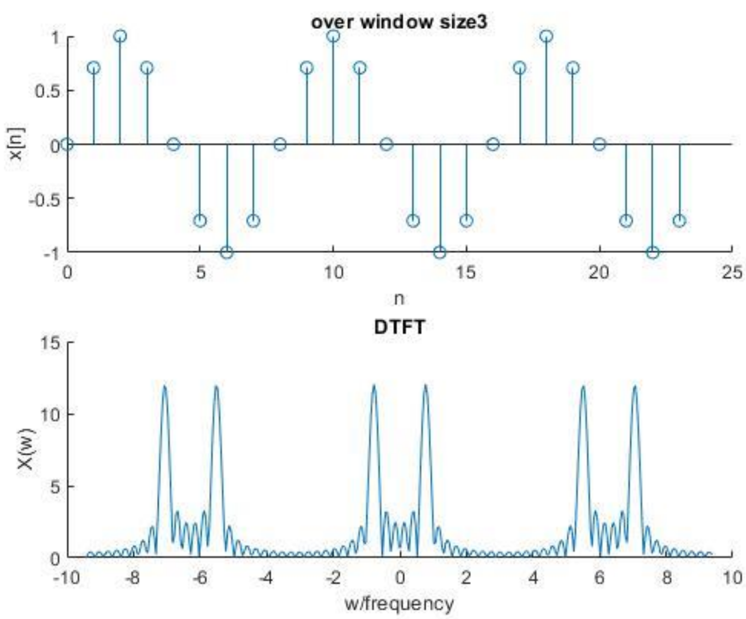


Figure 8 Window size of 3

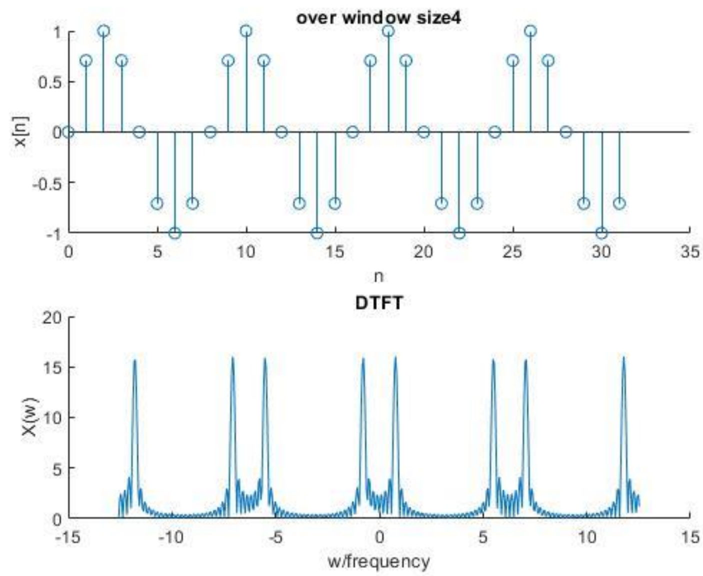


Figure 9 Window size of 4

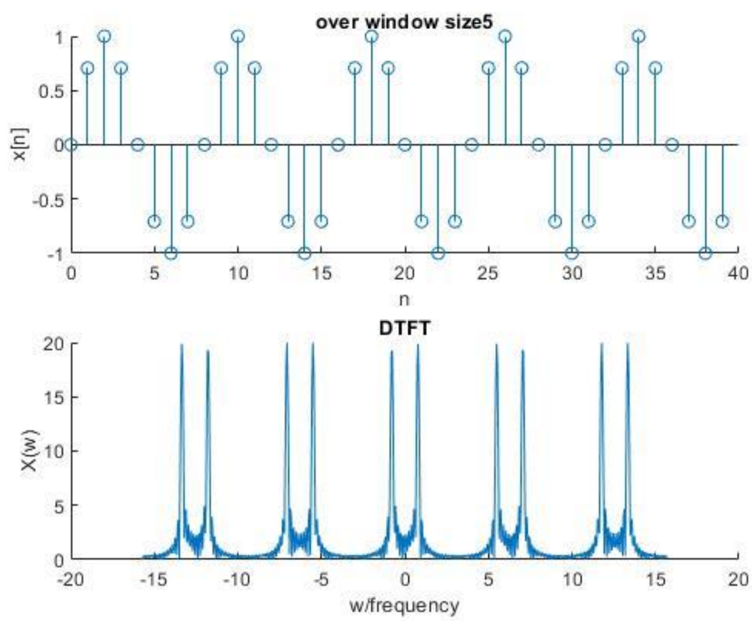


Figure 10 Window size of 5

So, we can see that here are 5 different window sizes, with the same sampling rates, the more widow size we have the more accurate the result is.

4.2Question 2

Polar plots are useful for plotting signals which are periodic. For example, consider a "full-wave rectified'" sine wave (one in which the negative portions have been "inverted"). Repeat Question 1(a) using polar plots for the transform instead of rectangular plots. Use the following for the input signal: x[n] = 0.5*sin(2*pi/N * n) + 0.33* sin(4*pi/N*n);

```
%Name :JUNPENG GAI
%SID:40009896

promote1="input the number of periods \n";
promote2="input the step size of the frequency interval\n";
NumberOfPeriod=input(promote1);
StepSize=input(promote2);
global w;
w=[ - (NumberOfPeriod ) * pi : StepSize : (NumberOfPeriod ) * pi ];
for loop = 1 : 5
 promote3="input the sampling rate(in terms of the number of times of the Nyquist rate, i.e. 2 times, 3.6 times, etc)\n";
 SamplingRate=input(promote3);
 N= floor(2*SamplingRate);
 n= [0:2*N-1];
 x = 0.5*sin(2*pi/N * n) + 0.33* sin(4*pi/N*n);
 figure
 subplot(2,1,1)
 hold on
   title(['Sampled sine wave at sampling rate = ', num2str(N),' times the Nyquist rate'])
   xlabel('n')
   ylabel('x[n]')
   stem(n,x);
   hold off
 sum=ft(x,w,n);
subplot(2,1,2)

 hold on

   sum1=abs(sum);
   polar(angle(sum),sum1);
   hold off
 %compute the fourier transform of the signal (by passing x as an argument to
 %your function);
 %plot the transform of the signal;
End
```

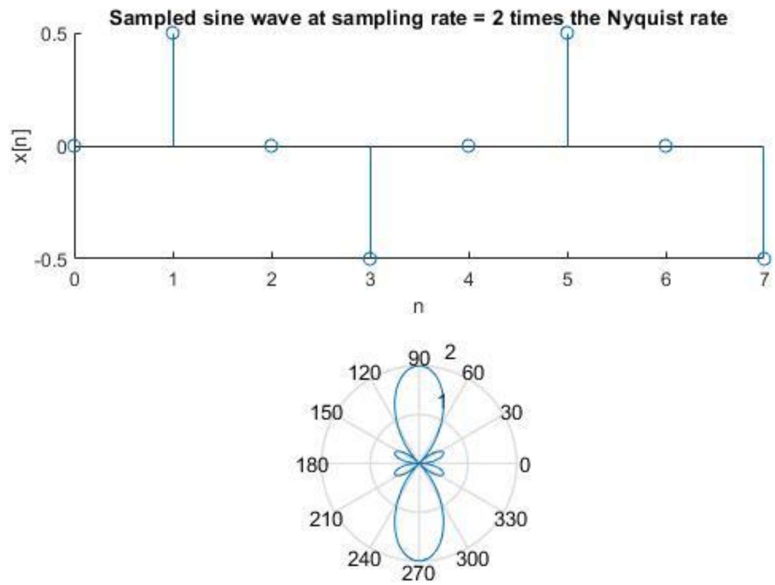


Figure 11 2times Nyquist rate

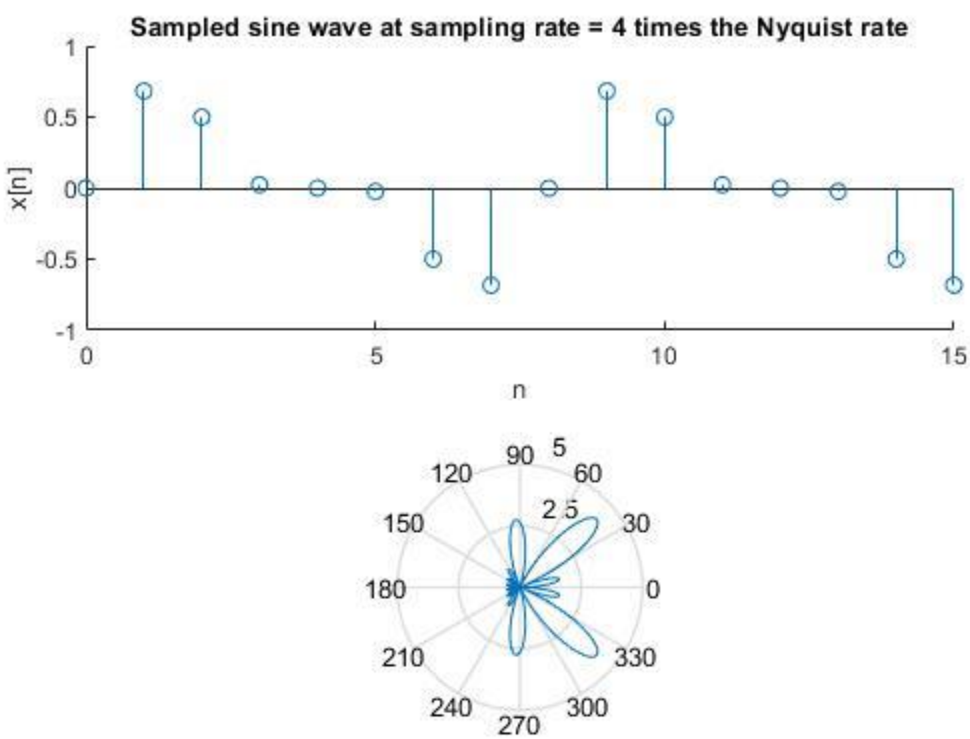


Figure 12 4times Nyquist rate

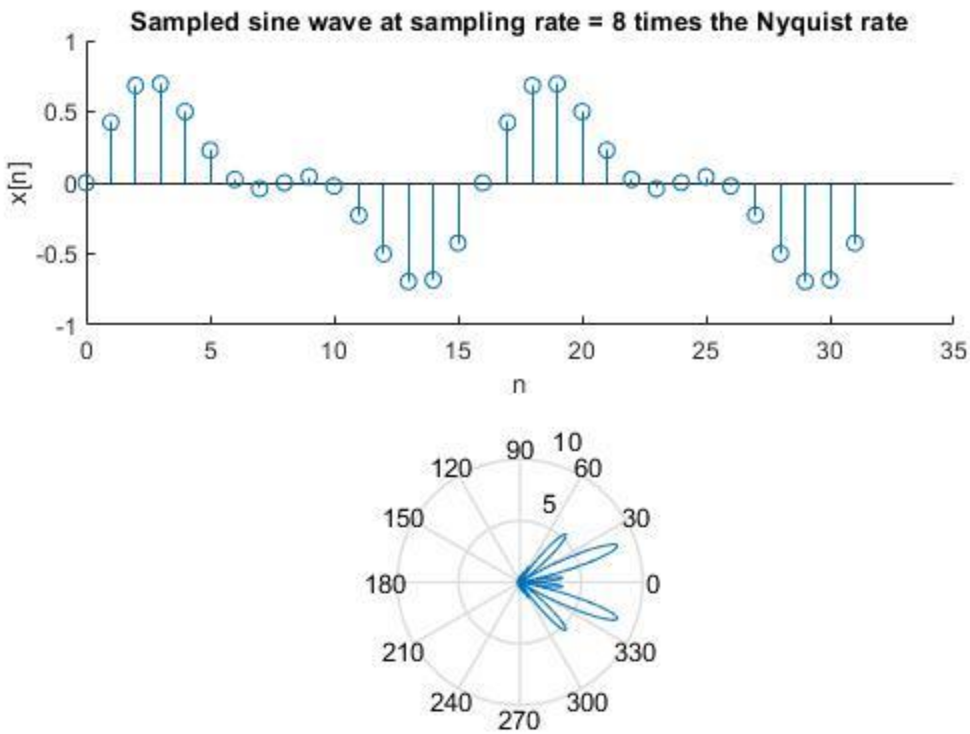


Figure 13 8times Nyquist rate

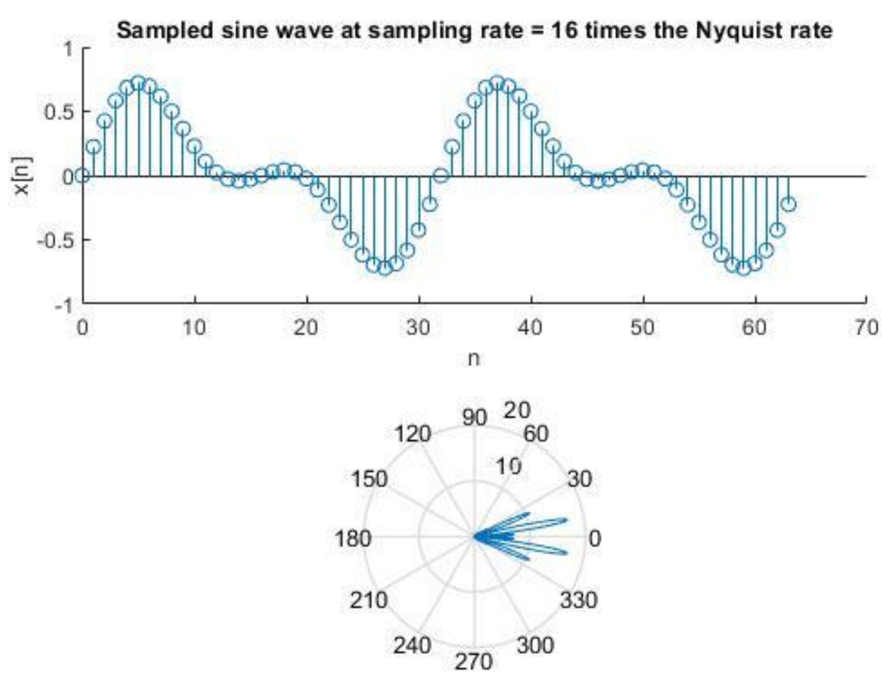
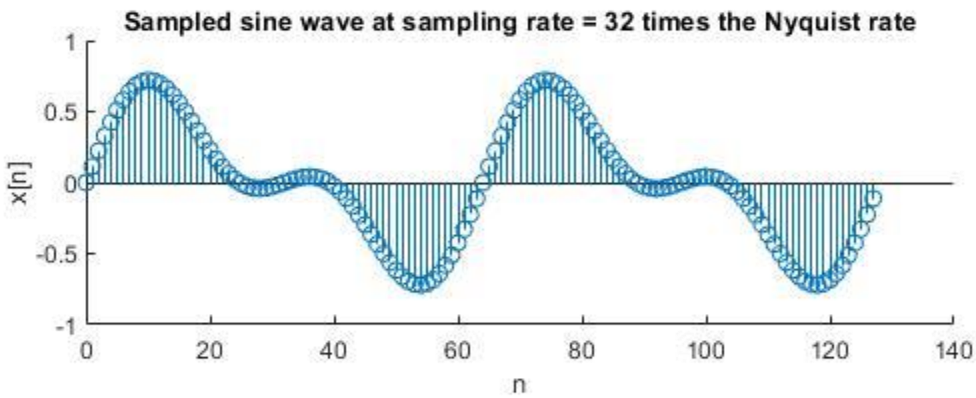


Figure 14 16times Nyquist rate



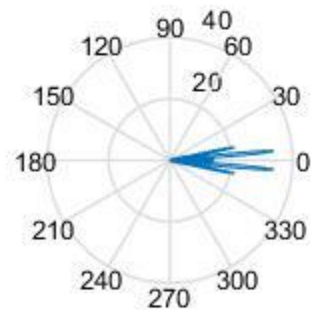


Figure 15 3times Nyquist rate

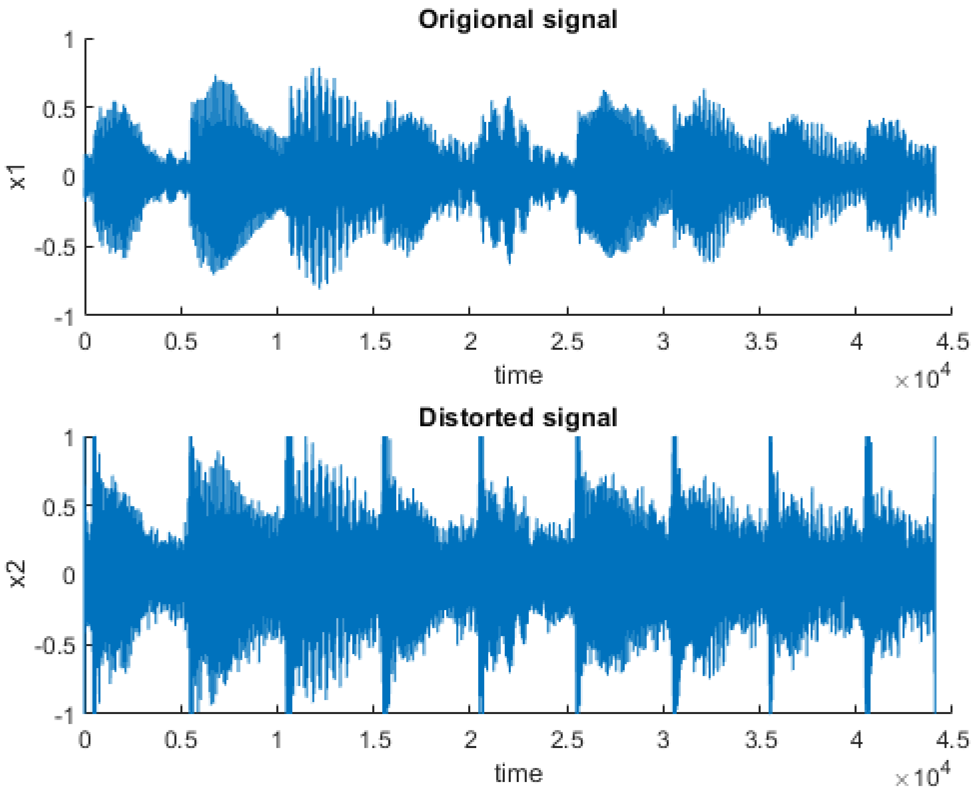
Basically, this is another form, theoretically the graph should be a single line connect to the (0,1) because here I am taking magnitude so no phase.

4.3 Question3

4.3.1 Load two files Original.wav and Distorted.wav to your program and plot them in time domain.

```
%junpeng gai
%40009896
n=[1,44100];
[x1, Fs1] = audioread('Original.wav',  n);
[x2, Fs2] = audioread('Distorted.wav', n);
subplot(2,1,1)
hold on
title('Origional signal');
xlabel('time');
ylabel('x1');
plot((0:44100-1),x1);
hold off
subplot(2,1,2)
hold on
title('Distorted signal');
xlabel('time');
ylabel('x2');
plot((0:44100-1),x2);
```

```
hold off
```



Published with MATLAB® R2020a

Figure 16 2 signals

4.3.2 Use MSE to compare the original signal and the distorted.

```
%junpeng gai
%40009896
n=[1,44100];
[x1, Fs1] = audioread('Original.wav',  n);
[x2, Fs2] = audioread('Distorted.wav', n);
MSE=0;
for i =n
   temp=0;
   for j=(1:i)
   temp=temp+(x1(j)-x2(j)).^2;
   end
   MSE=temp+MSE;
end
MSE=MSE./44100;
disp(['MSE is: ' num2str(MSE)])
```

```
MSE is: 0.025014
```

3.3.3 Design a system that recovers the original signal from the distorted signal

And save it to a file called Recovered.wav. Which domain did you choose for the design: frequency domain or time domain? Why?

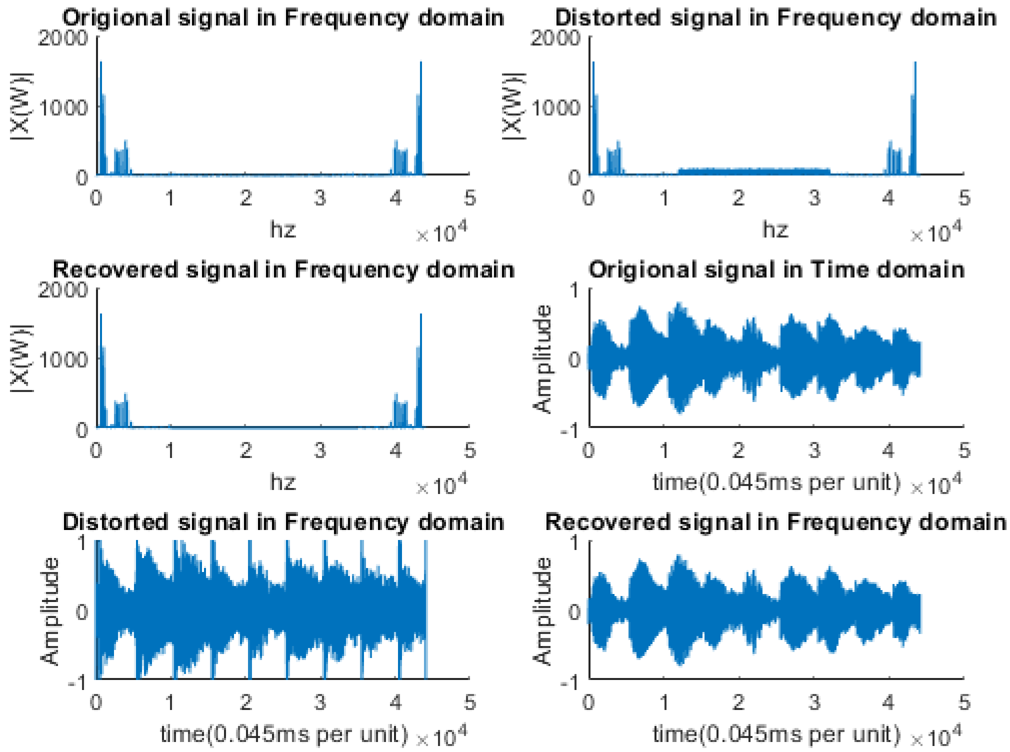
I will use a low pass filter in the frequency domain to show the signal. Because after doing the FFT, I find out that the noise is in the high frequency area, so by using a low pass filter we can get rider of the noise.Here is my code and graph.

```
%junpeng gai
%40009896
n=[1,44100];
[x1, Fs1] = audioread('Original.wav',  n);
[x2, Fs2] = audioread('Distorted.wav', n);
MSE=0;

for i =n
    temp=0;
    for j=(1:i)
    temp=temp+(x1(j)-x2(j)).^2;
    end
    MSE=temp+MSE;
end
MSE=MSE./44100;
disp(['MSE is: ' num2str(MSE)])

X1=fft(x1);
X2=fft(x2);
X3=X2;
for i=(10000:35000)
X3(i)=0+0i;
end


x3=ifft(X3);
filename = 'Recovered.wav';
audiowrite(filename,x4,22050)
subplot(3,2,1)
hold on
title('Origional signal in Frequency domain')
xlabel('hz')
ylabel('|X(W)|')
plot((0:length(x1)-1), abs(X1));

hold off
subplot(3,2,2)
hold on
title('Distorted signal in Frequency domain')
xlabel('hz')
```

```
ylabel('|X(W)|')
plot((0:length(x2)-1), abs(X2));

hold off

subplot(3,2,3)
hold on
title('Recovered signal in Frequency domain')
xlabel('hz')
ylabel('|X(W)|')
plot((0:length(x2)-1), abs(X3));

hold off

subplot(3,2,4)
hold on
title('Origional signal in Time domain')
xlabel('time(0.045ms per unit)')
ylabel('Amplitude')
plot((0:44100-1),x1);

hold off

subplot(3,2,5)
hold on
title('Distorted signal in Frequency domain')
xlabel('time(0.045ms per unit)')
ylabel('Amplitude')
plot((0:44100-1),x2);

hold off

subplot(3,2,6)
hold on
title('Recovered signal in Frequency domain')
xlabel('time(0.045ms per unit)')
ylabel('Amplitude')
plot((0:44100-1),x3);

hold off
```

MSE is: 0.025014

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



[Published with MATLAB® R2020a](#)

After I applied the filter I manually change the magnitude of the high frequency to be zero

```
for i=(10000:35000)
X3(i)=0+0i;
end
```

Or this section can be replaced by following function:

```
x4=lowpass(x2,10000/44100*22050,22050);
```

Because the noise frequency is above around 10000 and below -10000. Because it's a periodic signal in frequency domain, so the cutoff frequency is around 10000.

4.3.4 Compute the MSE between the recovered signal and the original signal. Does your system improve the MSE?

MSE is: 0.025014

MSE is: 0.00032927-5.3338e-07i

We can see the MSE has been significantly improved after applying the filter.

4.3.5 By playing and listening to recovered signal, does your system improve the quality of the sound?

```
filename = 'Recovered.wav';
audiowrite(filename,x4,22050)
```

After I use this code to save the recovered signal, I listen to it. Yes, the quality of the signal is better

5.Conclusions

In this lab section I have learnt a lot, especially about sampling and filter. I get to know how to use the function to pass the parameter and by define a global variable we can reuse the same variable in different functions. Also, about the signal sampling, I have a more comprehensive understanding about effects of the sampling frequency, window size and how to use polar form to plot the phase and magnitude.

6. Appendix

6.1 Fourier Transform script

```
%Name :JUNPENG GAI
%SID:40009896
function ft=ft(x)
global w;
length_w=length(w);
length_n=length(x);
n=[0:1:length_n-1];
sum=zeros([1 length_w]);
for frequency = 1:length_w
 for index = 1:length_n
 sum(frequency) = sum(frequency) + (x(index) * exp((-j*w(frequency)*n(index ))));
 end
end

ft=sum;
```

6.2 Question 1.a

```
%Name :JUNPENG GAI
%SID:40009896

promote1="input the number of periods \n";
promote2="input the step size of the frequency interval\n";
NumberOfPeriod=input(promote1);
StepSize=input(promote2);
global w;
w=[ - (NumberOfPeriod ) * pi : StepSize : (NumberOfPeriod ) * pi ];
for loop = 1 : 5
 promote3="input the sampling rate(in terms of the number of times of the Nyquist rate, i.e. 2 times, 3.6 times, etc)\n";
 SamplingRate=input(promote3);
 N= floor(2*SamplingRate);
 n= [0:2*N-1];
 x = sin ( 2*pi/N * n);
 figure
 subplot(2,1,1)
 hold on
   title(['Sampled sine wave at sampling rate = ', num2str(SamplingRate),' times the Nyquist rate'])
   xlabel('n')
   ylabel('x[n]')
   stem(n,x);
   hold off
 sum=ft(x);
subplot(2,1,2)

 hold on
   title(['FT with step size', num2str(StepSize)])
   xlabel('w/frequency')
   ylabel('X(w)')
   sum=abs(sum);
   plot(w,sum)
   hold off
 %compute the fourier transform of the signal (by passing x as an argument to
 %your function);
 %plot the transform of the signal;
end
```

6.3 Question1.b

```
%Name :JUNPENG GAI
%SID:40009896
 promote3="input the sampling rate(in terms of the number of times of the Nyquist rate, i.e. 2 times, 3.6 times, etc)\n";
 SamplingRate=input(promote3);
 N= floor(2*SamplingRate);
global w

for loop = 1 : 5
promote1="input the number of periods \n";
NumberOfPeriod=input(promote1);
 n= [0:N*NumberOfPeriod-1];
w=[ - (NumberOfPeriod ) * pi : 0.05 : (NumberOfPeriod ) * pi ];
 x = sin ( 2*pi /N * n);
 figure
 subplot(2,1,1)
 hold on
   title(['over window size',num2str(NumberOfPeriod)])
   xlabel('n')
   ylabel('x[n]')
   stem(n,x);
   hold off
 sum=ft(x);
subplot(2,1,2)

 hold on
   title(['DTFT'])
   xlabel('w/frequency')
   ylabel('X(w)')
   sum=abs(sum);
   plot(w,sum)
   hold off
 %compute the fourier transform of the signal (by passing x as an argument to
 %your function);
 %plot the transform of the signal;
End
```

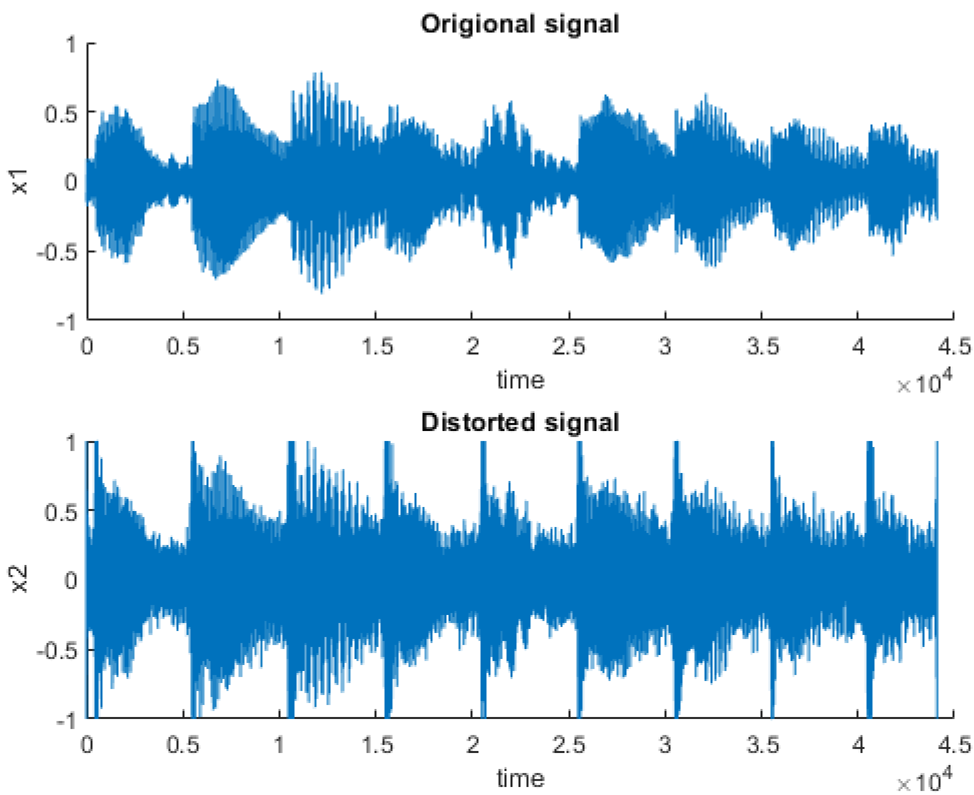
6.4 Question 2

```
%Name :JUNPENG GAI
%SID:40009896

promote1="input the number of periods \n";
promote2="input the step size of the frequency interval\n";
NumberOfPeriod=input(promote1);
StepSize=input(promote2);
global w;
w=[ - (NumberOfPeriod ) * pi : StepSize : (NumberOfPeriod ) * pi ];
for loop = 1 : 5
 promote3="input the sampling rate(in terms of the number of times of the Nyquist rate, i.e. 2 times, 3.6 times, etc)\n";
 SamplingRate=input(promote3);
 N= floor(2*SamplingRate);
 n= [0:2*N-1];
x = 0.5*sin(2*pi/N * n) + 0.33* sin(4*pi/N*n);
 figure
 subplot(2,1,1)
 hold on
   title(['Sampled sine wave at sampling rate = ', num2str(SamplingRate),' times the Nyquist rate'])
```

```
    xlabel('n')
    ylabel('x[n]')
    stem(n,x);
    hold off
 sum=ft(x);
subplot(2,1,2)


    sum=abs(sum);
    polar((w),sum);
 %compute the fourier transform of the signal (by passing x as an argument to
 %your function);
 %plot the transform of the signal;
end
```

6.5 Question 3. Load signal

```
%junpeng gai
%40009896
n=[1,44100];
[x1, Fs1] = audioread('Original.wav', n);
[x2, Fs2] = audioread('Distorted.wav', n);
subplot(2,1,1)
hold on
title('Origional signal');
xlabel('time');
ylabel('x1');
plot((0:44100-1),x1);
hold off
subplot(2,1,2)
hold on
title('Distorted signal');
xlabel('time');
ylabel('x2');
plot((0:44100-1),x2);
hold off
```



Published with MATLAB® R2020a

6.6 Question 3. Compare MSE between original and distorted signal

```
%junpeng gai
%40009896
n=[1,44100];
[x1, Fs1] = audioread('Original.wav', n);
[x2, Fs2] = audioread('Distorted.wav', n);
MSE=0;
for i =n
    temp=0;
    for j=(1:i)
    temp=temp+(x1(j)-x2(j)).^2;
    end
    MSE=temp+MSE;
end
MSE=MSE./44100;
disp(['MSE is: ' num2str(MSE)])
```

```
MSE is: 0.025014
```

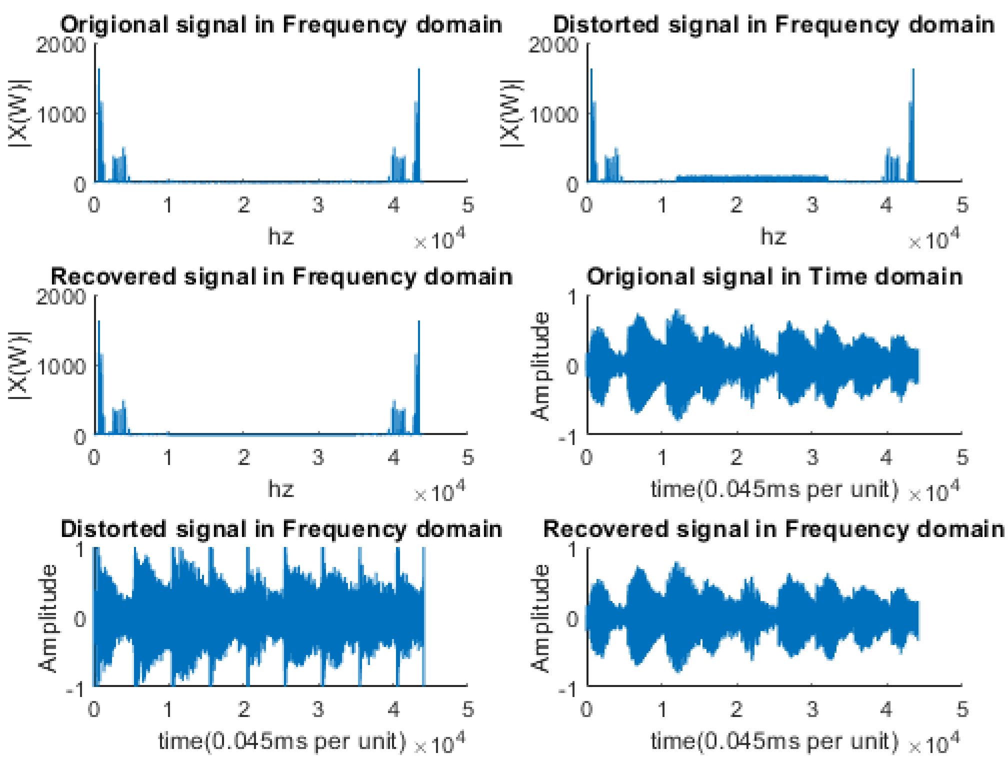
Published with MATLAB® R2020a

6.7 Question 3. Recover

```
%junpeng gai
%40009896
n=[1,44100];
[x1, Fs1] = audioread('Original.wav',  n);
[x2, Fs2] = audioread('Distorted.wav', n);
MSE=0;

for i =n
    temp=0;
    for j=(1:i)
    temp=temp+(x1(j)-x2(j)).^2;
    end
    MSE=temp+MSE;
end
MSE=MSE./44100;
disp(['MSE is: ' num2str(MSE)])

X1=fft(x1);
X2=fft(x2);
X3=X2;
for i=(10000:35000)
X3(i)=0+0i;
end


x3=ifft(X3);
filename = 'Recovered.wav';
audiowrite(filename,x4,22050)
subplot(3,2,1)
hold on
title('Origional signal in Frequency domain')
xlabel('hz')
ylabel('|X(W)|')
plot((0:length(x1)-1), abs(X1));

hold off
subplot(3,2,2)
hold on
title('Distorted signal in Frequency domain')
xlabel('hz')
ylabel('|X(W)|')
plot((0:length(x2)-1), abs(X2));

hold off

subplot(3,2,3)
hold on
title('Recovered signal in Frequency domain')
xlabel('hz')
ylabel('|X(W)|')
plot((0:length(x2)-1), abs(X3));

hold off

subplot(3,2,4)
hold on
title('Origional signal in Time domain')
xlabel('time(0.045ms per unit)')
ylabel('Amplitude')
plot((0:44100-1),x1);
```

```
hold off

subplot(3,2,5)
hold on
title('Distorted signal in Frequency domain')
xlabel('time(0.045ms per unit)')
ylabel('Amplitude')
plot((0:44100-1),x2);

hold off

subplot(3,2,6)
hold on
title('Recovered signal in Frequency domain')
xlabel('time(0.045ms per unit)')
ylabel('Amplitude')
plot((0:44100-1),x3);

hold off
```

```
MSE is: 0.025014

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



Published with MATLAB® R2020a

6.8 Question 3. Compare recovered and distorted and audiowrite

```
%junpeng gai
%40009896
n=[1,44100];
[x1, Fs1] = audioread('Original.wav',  n);
[x2, Fs2] = audioread('Distorted.wav', n);
MSE=0;
for i =n
    temp=0;
    for j=(1:i)
    temp=temp+(x1(j)-x2(j)).^2;
    end
    MSE=temp+MSE;
end
MSE=MSE./44100;
disp(['MSE is: ' num2str(MSE)])

X1=fft(x1);
X2=fft(x2);
x4=lowpass(x2,10000/44100*22050,22050);
X4=fft(x4);

for i =n
    temp=0;
    for j=(1:i)
    temp=temp+(x1(j)-x3(j)).^2;
    end
    MSE=temp+MSE;
end
filename = 'Recovered.wav';
audiowrite(filename,x4,44100)
MSE=MSE./44100;
disp(['MSE is: ' num2str(MSE)])
```

```
MSE is: 0.025014

MSE is: 0.00032927-5.3338e-07i
```