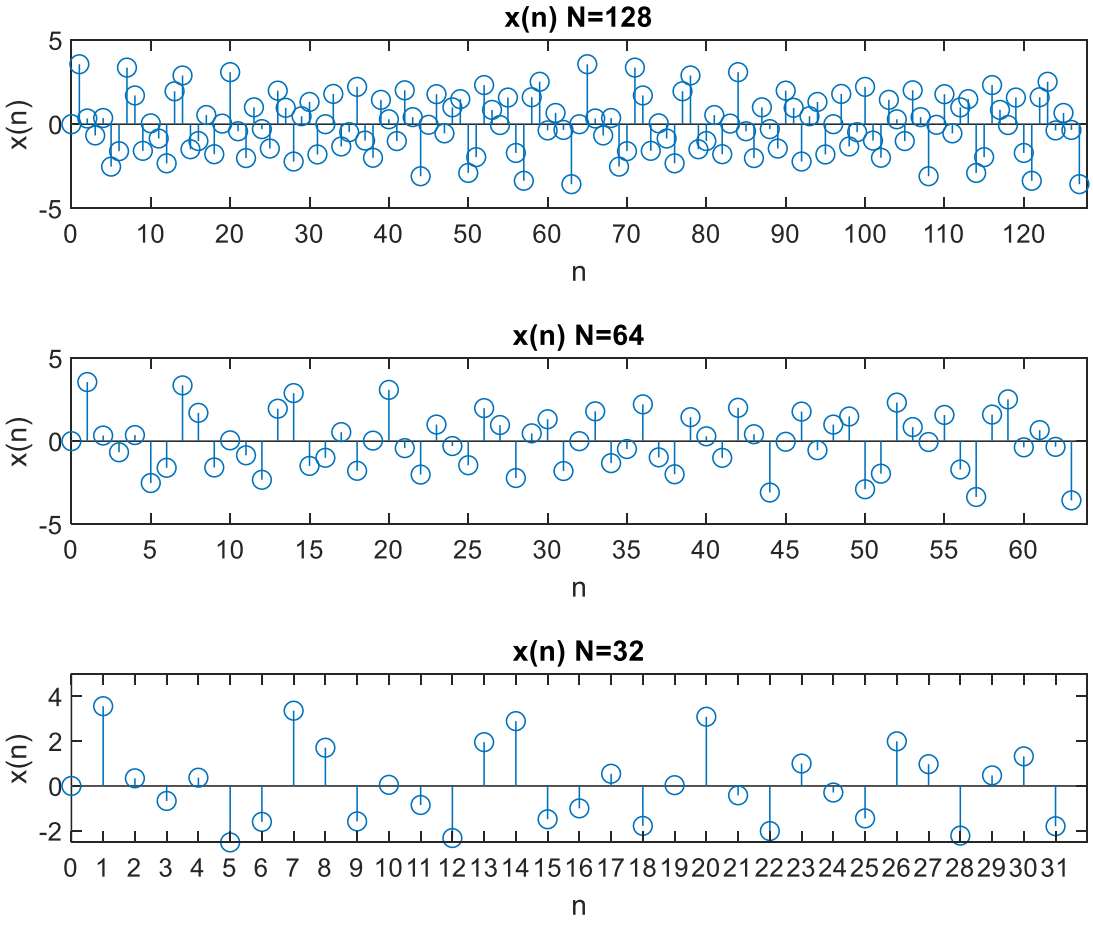
Matlab Assignment:
General Policy
• Though the students are allowed to discuss with each other, each student should work individually and independently on Matlab implementation and report writing.
• A brief report is required, with free form. Contents to be included in the report: results (e.g. figures) and brief discussions of results, and also the codes at the end.
• If a student has a very special situation requiring an unavoidable late submission, she/he should inform the instructor before the due day as early as possible.

Problem 1: DFT leakage, noisy signal, spectrum

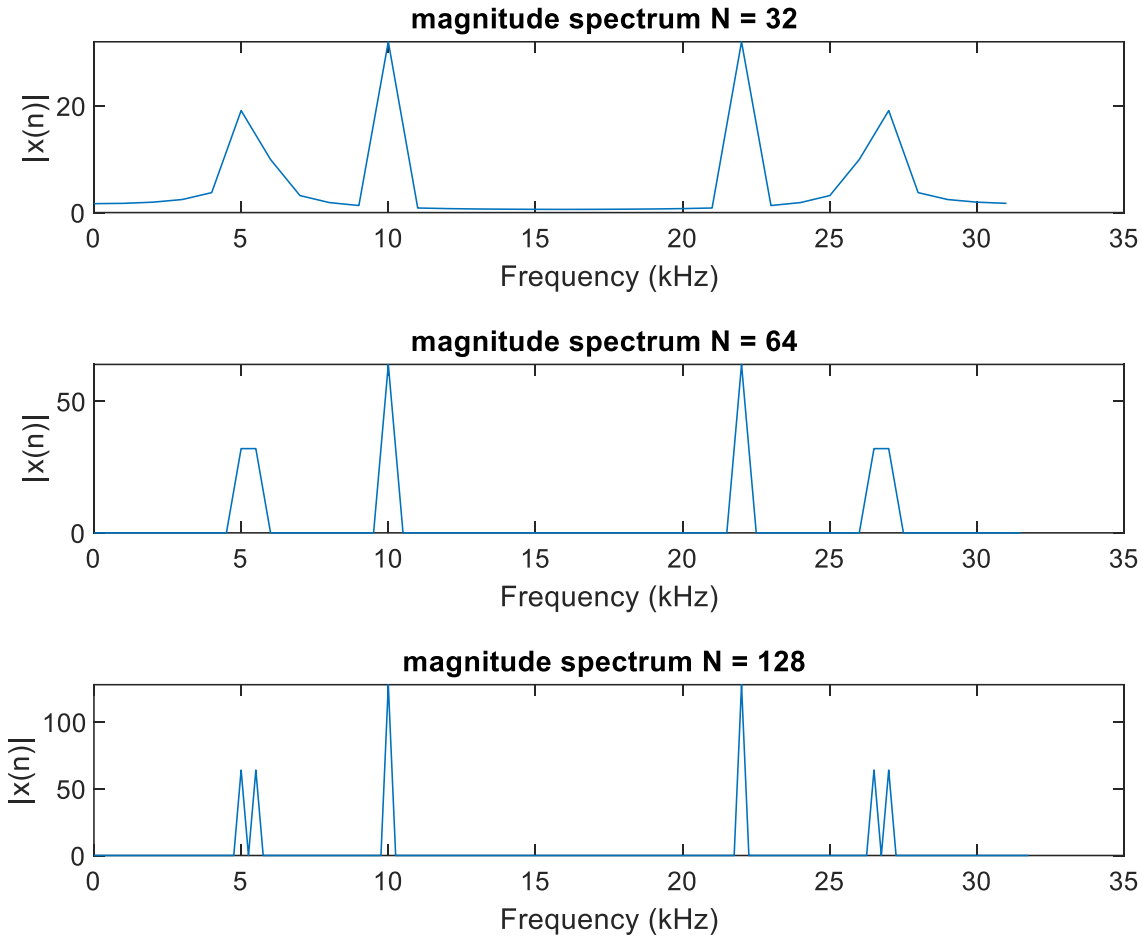
Note: You might want to study the reading material on Spectrum Analysing using DFT. Suppose the signal of interest x(t) consists of 3 sinusoid components, i.e. x(t) = sin(2pf1t) + sin(2pf2t) + 2 sin(2pf3t) with the three frequencies f1=5 kHz, f2=5.5 kHz, and f3=10 kHz. Assume the sampling frequency is fs=32 kHz.

(a) Write down x(n), and plot x(n) for n=0, 1, …, N-1, with N =32, 64, 128 respectively.



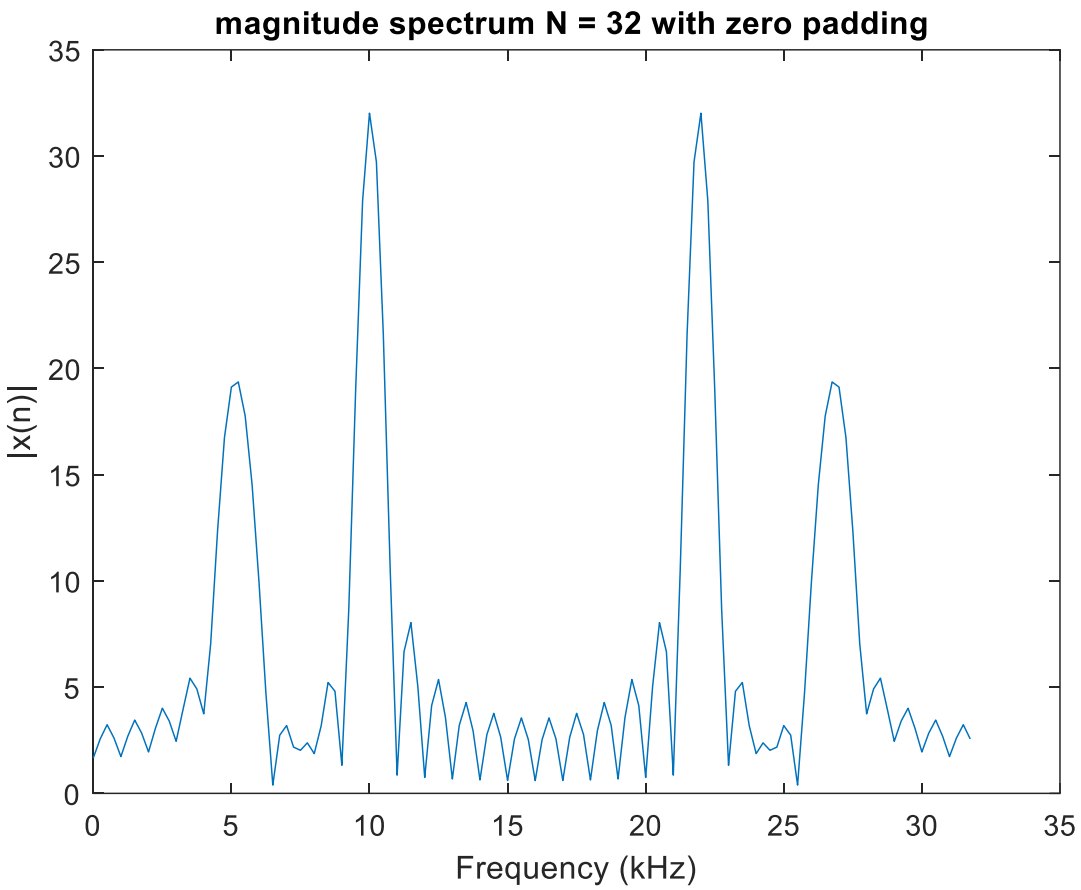
(b) What is the corresponding frequency resolution? For N=32, 64, and 128, plot the corresponding magnitude spectrums of {x(n)}. Comment on the results.

The frequency resolution is the sampling frequency divided by the length N of the FFT therefore the frequency resolutions are 1000Hz, 500Hz and 250Hz respectively, the magnitude spectrums are the FFT of the function.



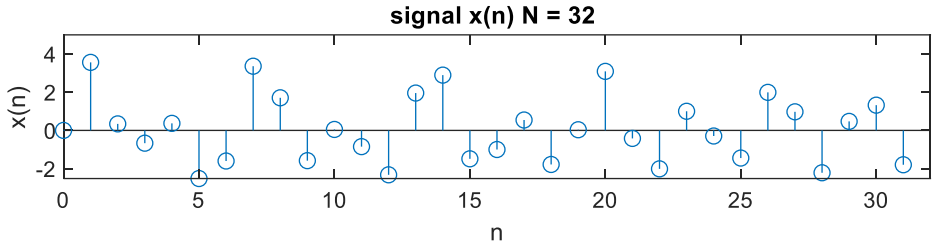
From these results we see that as the frequency resolution drops (as N increases) we can better see the points of interest of our magnitude spectrum. In the first plot the peaks corresponding to 5khz and 5.5khz seem to form 1 larger point making it difficult to distinguish that its 2 separate points. In the N= 64 plot we get a better feel that there are 2 points, but they take the form of a plateau. In the N=128 we finally have enough resolution to see that they are 2 distinct peaks.

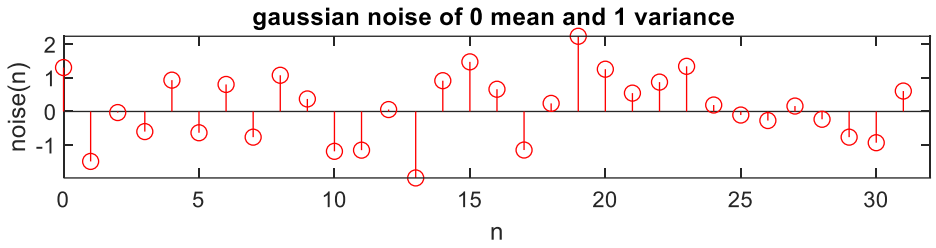
(c) For N=32, zero-padding {x(n)} to be with length 128 first, then plot the corresponding magnitude spectrum. Comment on the results when compared with N=128 in (b).

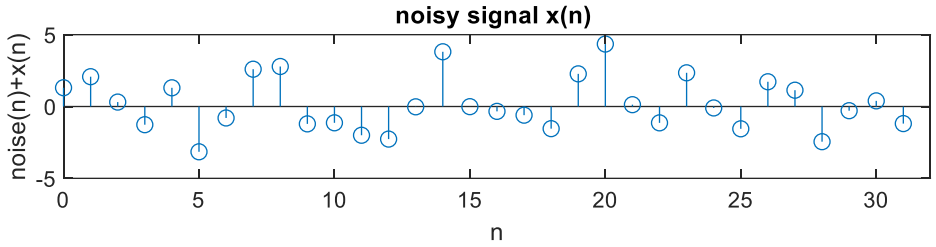


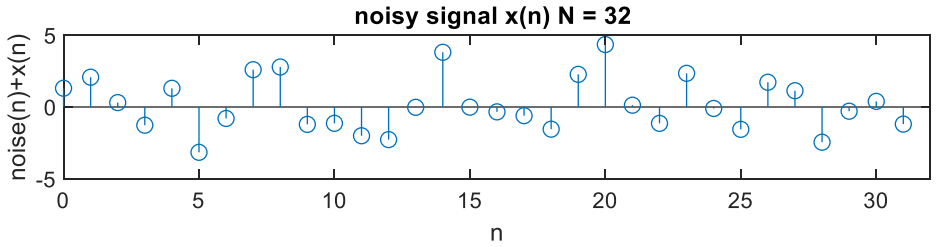
Compared to the previous Magnitude spectrum, the main peaks are better defined, we have a stronger sense of what frequencies contribute to the signal, but we don't have any additional information. We still see the 5kHz and 5.5kHz parts of the signal as a single peak instead of 2 distinct peaks. This is because we are essentially sampling the same information between the 2 graphs, the 0 padded graph just interpolates the same results at more closely spaced frequencies.

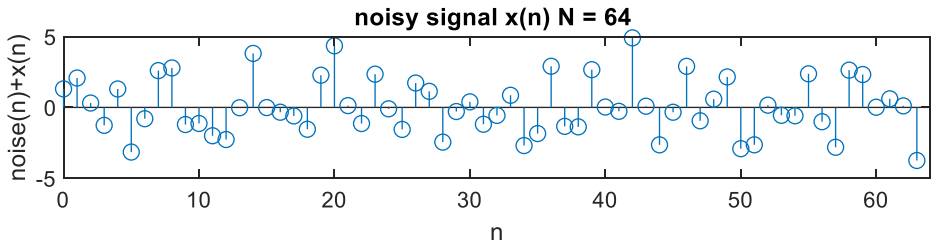
(d) Study the effects of Gaussian white noise (WGN). Generate a zero-mean white Gaussian noise sequence (use 'randn' in Matlab) with variance 1. For N=32, plot the noise sequence, the signal sequence of (a), and the result of adding the two signals. Repeat (a) and (b) by using the above noisy signal.

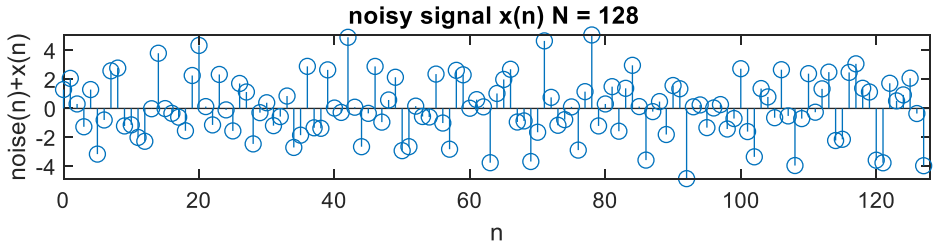

signal x(n) N = 32
x(n)
n

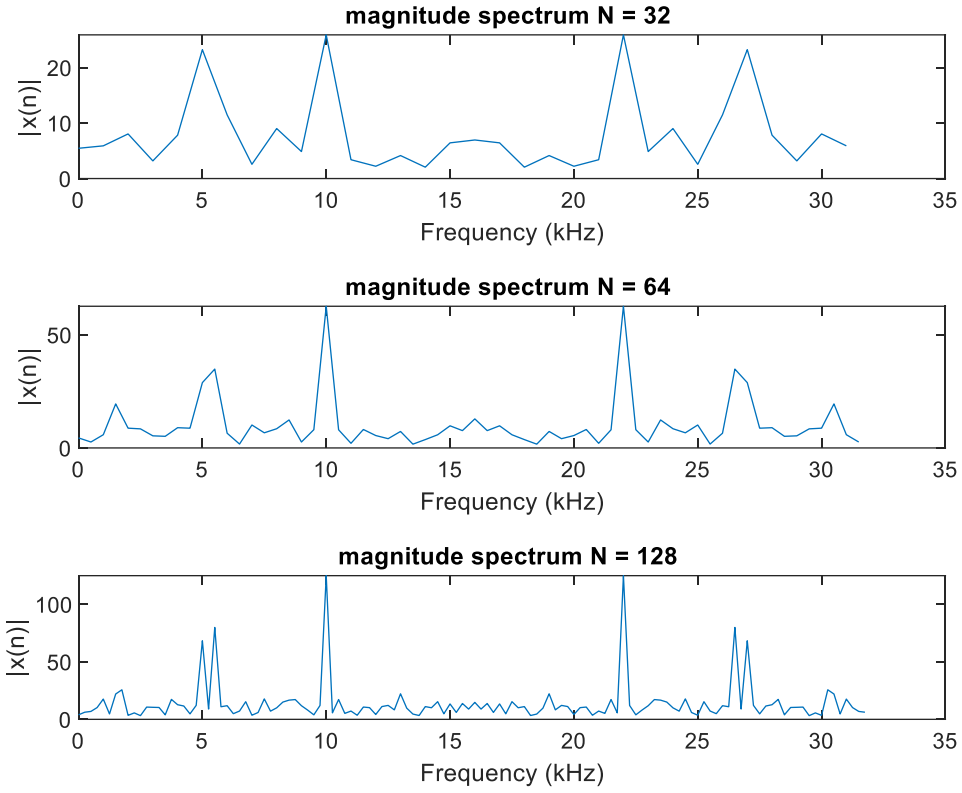

gaussian noise of 0 mean and 1 variance
noise(n)
n


noisy signal x(n)
noise(n)+x(n)
n


noisy signal x(n) N = 32
noise(n)+x(n)
n

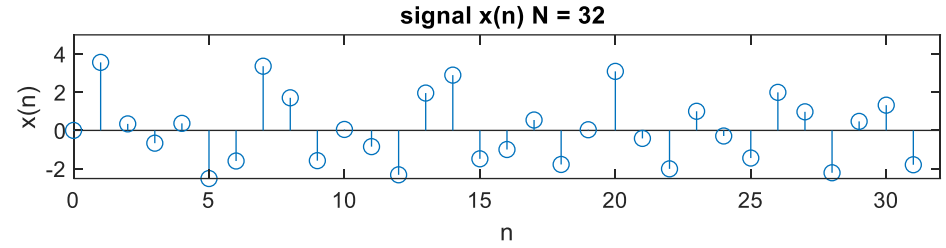

noisy signal x(n) N = 64
noise(n)+x(n)
n

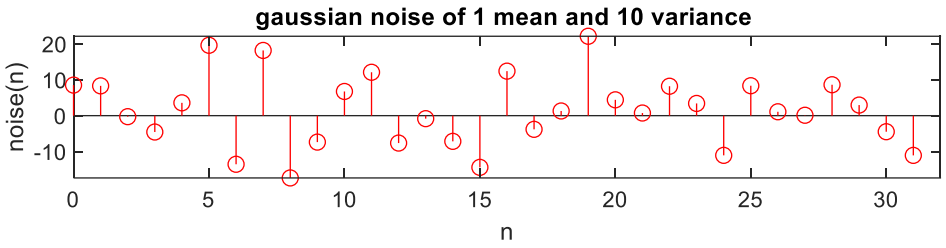

noisy signal x(n) N = 128
noise(n)+x(n)
n

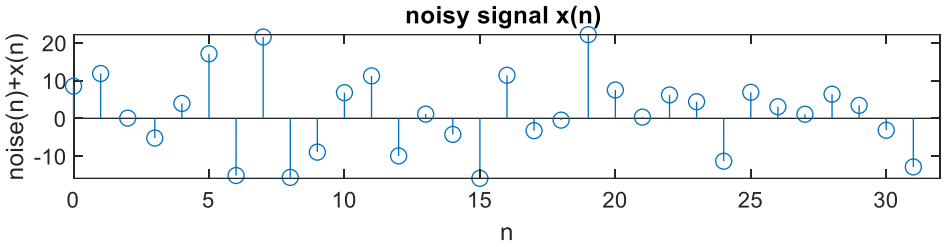


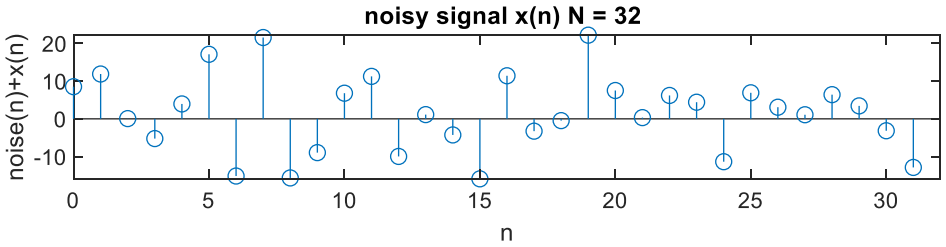
In the magnitude spectrum of the noisy signals, we see that in the lower sampled signals (N=32,64) the noise presents a significant problem. In the N=32 magnitude spectrum we can only clearly make out the 10kHz signal component, the rest of the signal's magnitude is indistinguishable from the magnitude component created by the noise. In the N=64 magnitude spectrum we can now distinguish the noise from the peaks of the actual signal however, there is still enough noise in the magnitude spectrum to hide some lower power components of the signal had there been any. In the N=128 magnitude spectrum we can clearly distinguish the noise from the signal and we have enough points sampled to clearly distinguish that there are 2 distinct peaks as 5kHz and 5.5kHz.

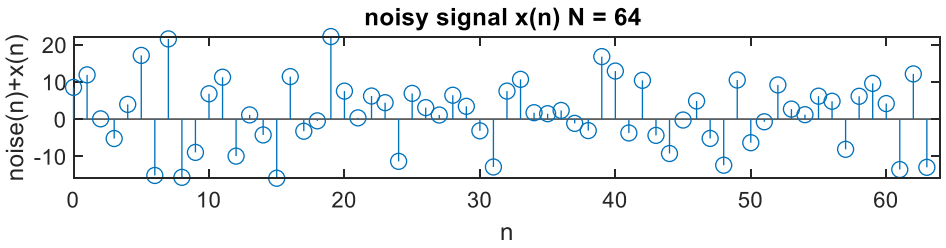
(e) Repeat the steps in (d) but for a WGN signal of mean 1 and variance of 10. Comments on your results.

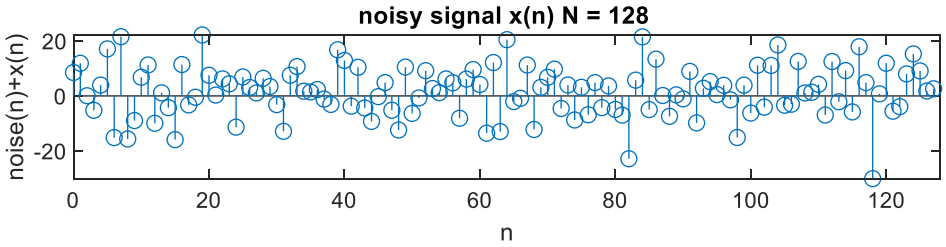

signal x(n) N = 32
x(n)
n

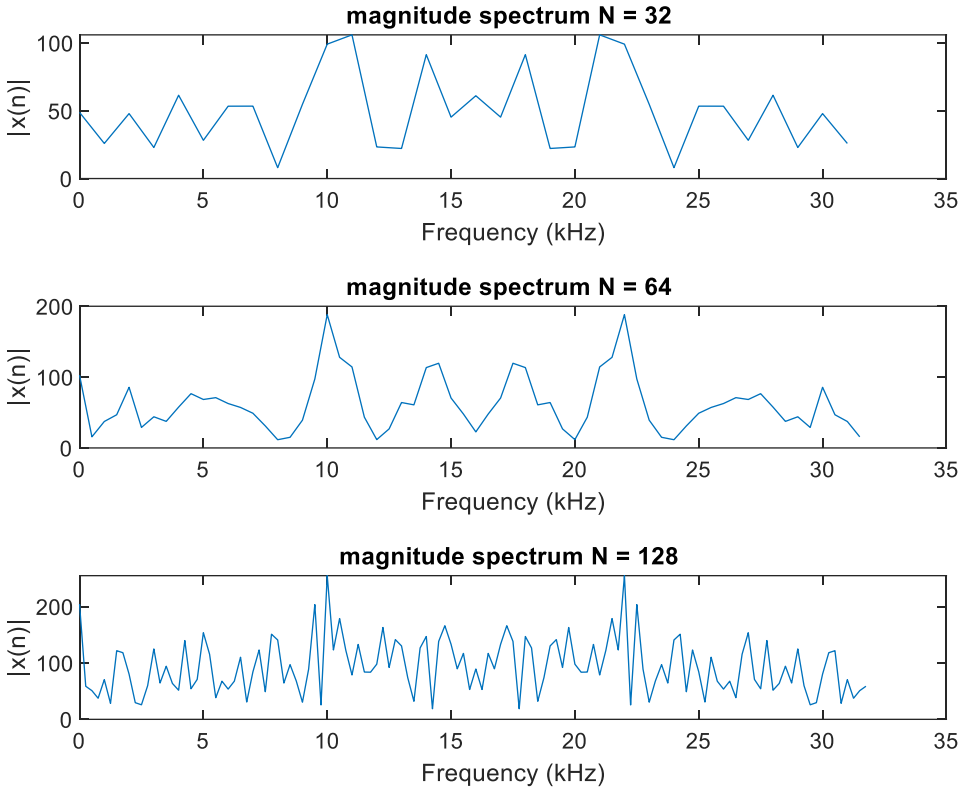

gaussian noise of 1 mean and 10 variance
noise(n)
n


noisy signal x(n)
noise(n)+x(n)
n


noisy signal x(n) N = 32
noise(n)+x(n)
n


noisy signal x(n) N = 64
noise(n)+x(n)
n

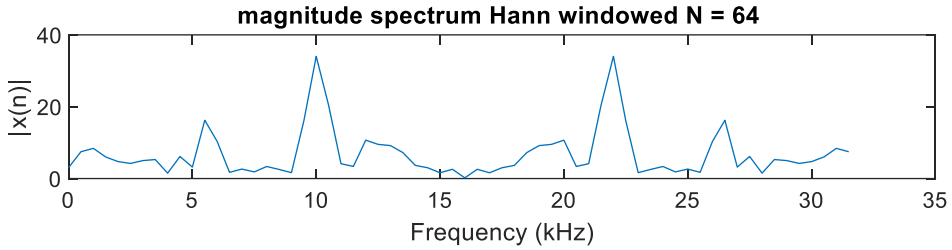

noisy signal x(n) N = 128
noise(n)+x(n)
n

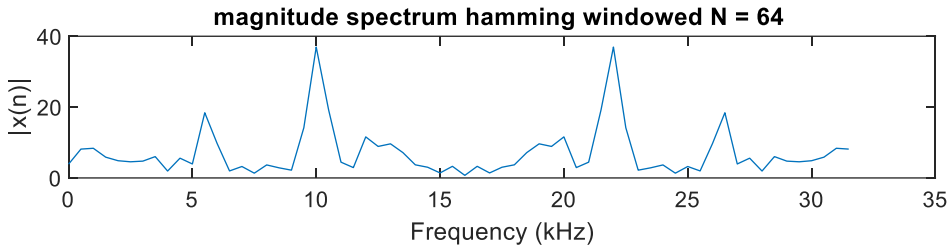


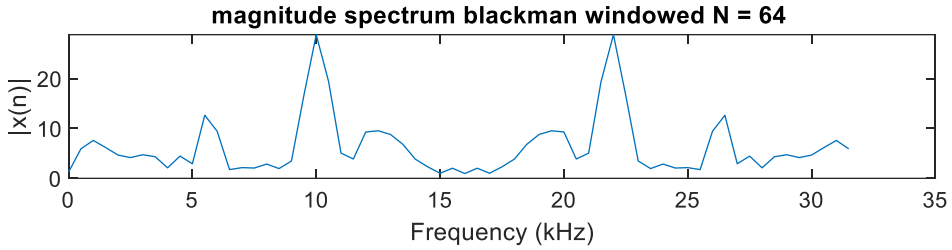
In this set the signal noise poses a much larger problem for our analysis. The noise manages to overshadow our actual signal in all three analysis'. We still manage to make out the largest of the peaks associated with the 10kHz frequency, but as we sample more data points it becomes more difficult to differentiate the peaks due to high frequency noise from the peaks caused by our signal.

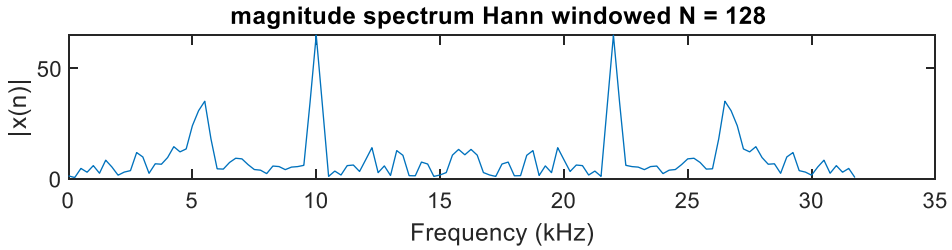
(f) For the noisy signal as in (d) with N= 64 and 128, study the effects of different windows (e.g., Hamming, Hann and Blackman). Comments on your results.

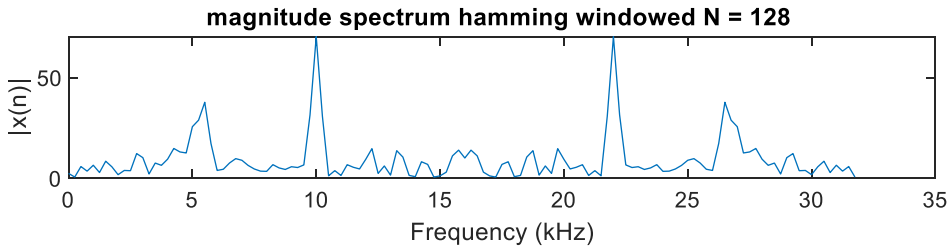
In the following figures we compare the effects of Hamming, Hann, and Blackman windows on our analysis. The 3 windows chosen all have very similar shapes, differing mostly at the values towards the edge of the window. Of the 3 the Hann window has the largest magnitude at the edges of the window. Observably we can see that the Hann window seems to have smooth transitions between peaks. The Blackman window has the smallest sidelobes with the smoothest peaks. And the hamming window has the sharpest peaks.

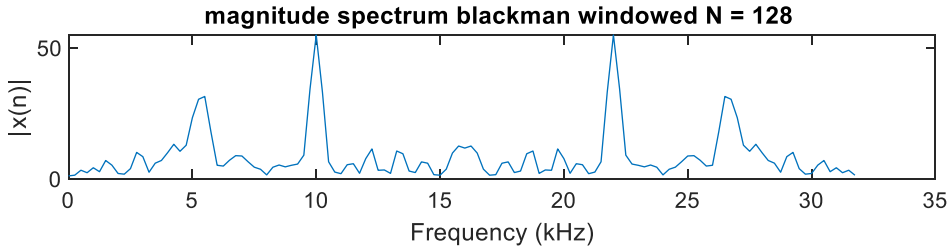

magnitude spectrum Hann windowed N = 64
|x(n)|
Frequency (kHz)


magnitude spectrum hamming windowed N = 64
|x(n)|
Frequency (kHz)


magnitude spectrum blackman windowed N = 64
|x(n)|
Frequency (kHz)


magnitude spectrum Hann windowed N = 128
|x(n)|
Frequency (kHz)


magnitude spectrum hamming windowed N = 128
|x(n)|
Frequency (kHz)

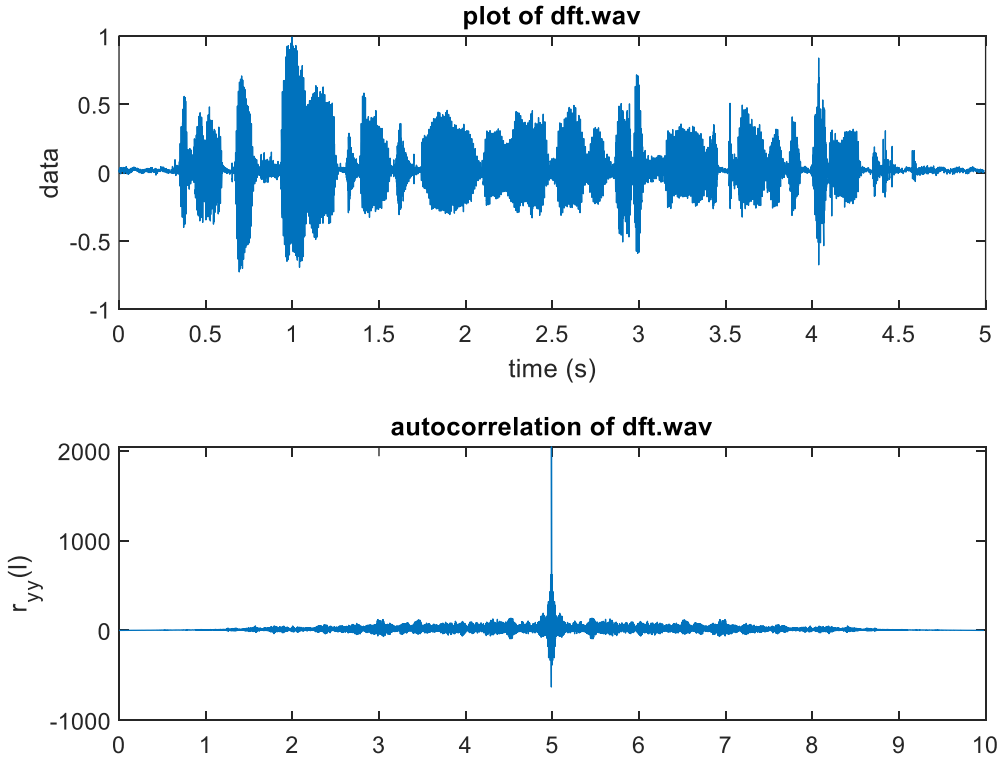

magnitude spectrum blackman windowed N = 128
|x(n)|
Frequency (kHz)

Problem 2: STFT for speech signal spectral analysis

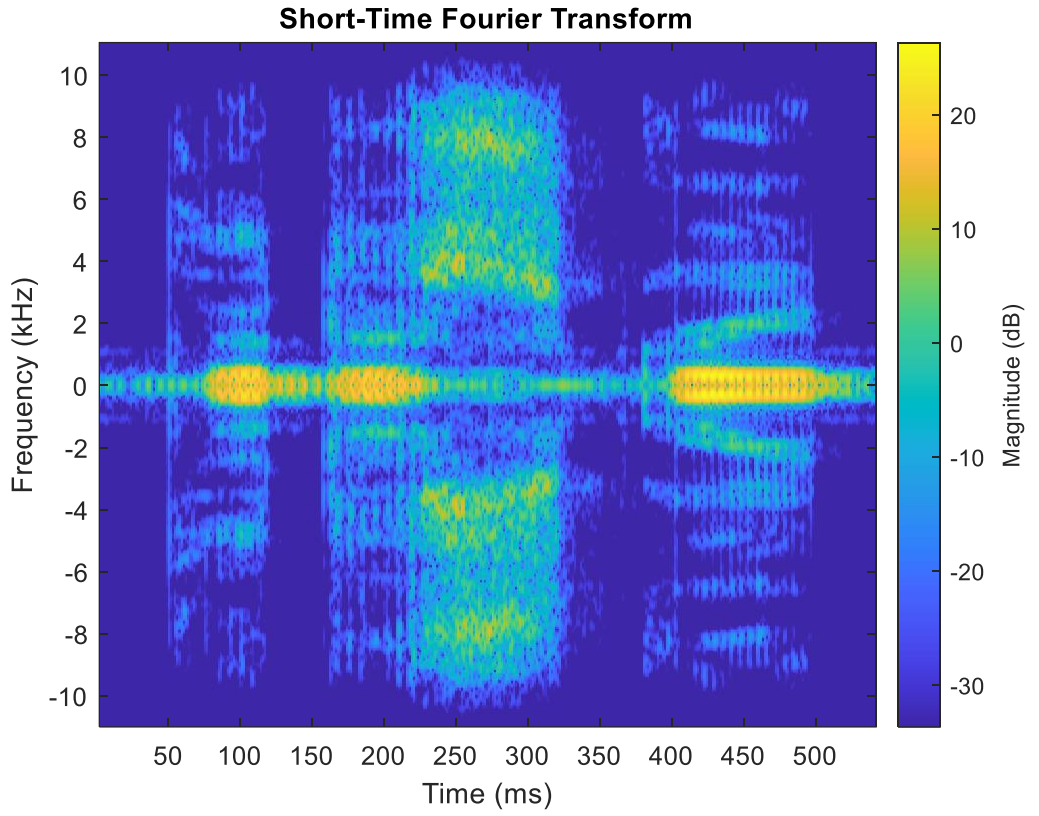
Download the “dft.wav” file from the course website and figure out some information of the .wav file by using ‘wavread’ command in Matlab. E.g., find out the sampling rate and the length of the recording (in second); hear the speech by using either ‘sound’ or ‘wavplay’ command in Matlab.

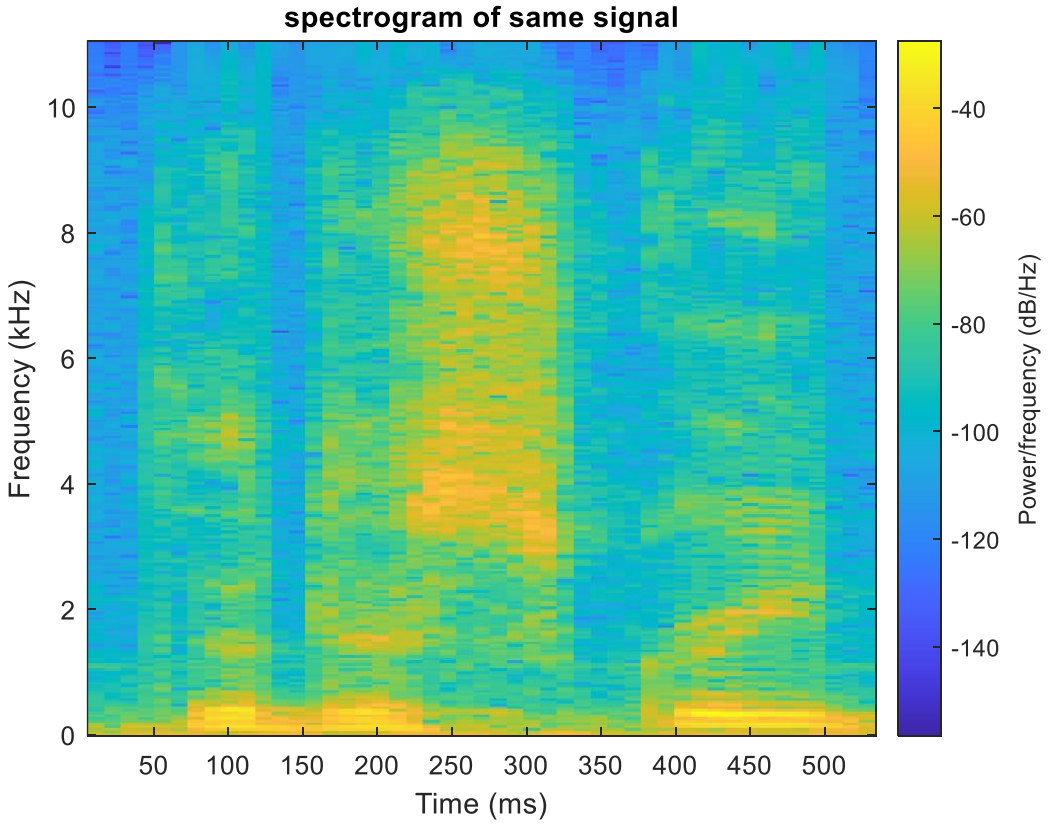
Processing in Matlab the wave is sampled at a rate of 22050Hz, plays for 4.9902 seconds, and says “The discrete Fourier transform of a real valued signal is conjugate symmetric”

(a) Demonstrate the time and frequency domain representations of the speech signals. Download the data file ‘wordSample.mat’, which is part of the above speech data with the same sampling rate, and load it into Matlab (e.g., load wordSample.mat). Plot the data, plot the autocorrelation sequence.



(b) Plot the STFT-based time-varying spectrum of the speech signal, using the data in ‘wordSample.mat’. Make the window length (i.e., the length of each segment) to be 22.5 milliseconds (i.e., the window length N=Fs*22.5/1000).


Short-Time Fourier Transform
Frequency (kHz)
10
8
6
4
2
0
-2
-4
-6
-8
-10
50
100
150
200
250
300
350
400
450
500
Time (ms)
20
10
0
-10
-20
-30
Magnitude (dB)


spectrogram of same signal
Frequency (kHz)
10
8
6
4
2
0
50
100
150
200
250
300
350
400
450
500
Time (ms)
-40
-60
-80
-100
-120
-140
Power/frequency (dB/Hz)

Matlab code

```
clear; close all;
% Assignment 2 q1
% Andrew Munro-West 18363572
%
% Problem 1: DFT leakage, noisy signal, spectrum
% Note: You might want to study the reading material on Spectrum Analysing
using DFT.
% Suppose the signal of interest x(t) consists of 3 sinusoid components, i.e.
% x(t) = sin(2pf1t) + sin(2pf2t) + 2 sin(2pf3t) with the three frequencies
f1=5 kHz, f2=5.5
% kHz, and f3=10 kHz. Assume the sampling frequency is fs=32 kHz.
% (a) Write down x(n), and plot x(n) for n=0, 1, …, N-1, with N =32, 64, 128
respectively.
f1 = 5000; f2 = 5500; f3 = 10000;
Fs = 32000;
n = 0:1/Fs:127/Fs;
n_x = 0:1:127;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);

figure
tiledlayout(3,1)

ax1 = nexttile;
stem(ax1,n_x,x_n)
xticks(0:10:127)
%xticklabels(0:1:127)
title(ax1,'x(n) N=128')
xlim(ax1,[0,128])
ylabel(ax1,'x(n)')
xlabel(ax1,'n')

ax2 = nexttile;
stem(ax2,n_x(1:64),x_n(1:64))
xticks(0:5:63)
title(ax2,'x(n) N=64')
xlim(ax2,[0,64])
ylabel(ax2,'x(n)')
xlabel(ax2,'n')

ax3 = nexttile;
stem(ax3,n_x(1:32),x_n(1:32))
xticks(0:1:31)
title(ax3,'x(n) N=32')
xlim(ax3,[0,32])
ylabel(ax3,'x(n)')
xlabel(ax3,'n')


% (b) What is the corresponding frequency resolution? For N=32, 64, and 128,
plot the
% corresponding magnitude spectrums of {x(n)}. Comment on the results.

%The frequency resolution is the sampling frequency divided by the length N
```

```
%of the FFT therefore the frequency resolutions are 1000Hz, 500Hz and 250Hz
%respectively, the magnitude spectrums are the FFT of the function.
%N_fft1 = 32; N_fft2 = 64; N_fft3 = 128;
%
% n = 0:1/Fs:(N_fft1-1)/Fs;
% x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
% y_n = abs(fft(x_n)); % magnitude spectrum
% f = (0:N_fft1-1)*Fs/N_fft1;
figure
tiledlayout(3,1)

ax1 = nexttile;

N=32;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n)); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax1,f,y_n)
xticklabels(0:5:35)
title(ax1,'magnitude spectrum N = 32 ')
ylabel(ax1,'|{x(n)}|')
xlabel(ax1,'Frequency (kHz)')

ax2 = nexttile;

N=64;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n)); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax2,f,y_n)
xticklabels(0:5:35)
title(ax2,'magnitude spectrum N = 64 ')
ylabel(ax2,'|{x(n)}|')
xlabel(ax2,'Frequency (kHz)')

ax3 = nexttile;

N = 128;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n)); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax3,f,y_n)
xticklabels(0:5:35)
title(ax3,'magnitude spectrum N = 128 ')
ylabel(ax3,'|{x(n)}|')
xlabel(ax3,'Frequency (kHz)')
```

```
% (c) For N=32, zero-padding {x(n)} to be with length 128 first, then plot
the
% corresponding magnitude spectrum. Comment on the results when compared with
% N=128 in (b).

figure
tiledlayout(1,1)

ax1 = nexttile;

N=32;
nfft = 4*N;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft([x_n,zeros(1,3*N)])); % magnitude spectrum
f = (0:nfft-1)*Fs/nfft;

plot(ax1,f,y_n)
xticklabels(0:5:35)
title(ax1,'magnitude spectrum N = 32 with zero padding ')
ylabel(ax1,'|{x(n)}|')
xlabel(ax1,'Frequency (kHz)')


% (d) Study the effects of Gaussian white noise (WGN). Generate a zero-mean
white
% Gaussian noise sequence (use 'randn' in Matlab) with variance 1. For N=32,
plot the
% noise sequence, the signal sequence of (a), and the result of adding the
two signals.
% Repeat (a) and (b) by using the above noisy signal.

figure
tiledlayout(3,1)

ax1 = nexttile;

N=32;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
f = (0:N-1);
stem(ax1,f,x_n)
xticklabels(0:5:35)
title(ax1,'signal x(n) N = 32 ')
ylabel(ax1,'x(n)')
xlabel(ax1,'n')
xlim([0,32])

ax2 = nexttile;

noise = randn(1,4*N);
stem(ax2,f,noise(1:32),'r')
title(ax2,'gaussian noise of 0 mean and 1 variance ')
ylabel(ax2,'noise(n)')
xlabel(ax2,'n')
```

```
xlim([0,32])

ax3 = nexttile;
stem(ax3,f,x_n+noise(1:32))
title(ax3,'noisy signal x(n) ')
ylabel(ax3,'noise(n)+x(n)')
xlabel(ax3,'n')
xlim([0,32])

figure
tiledlayout(3,1)

ax1 = nexttile;

N=32;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
f = (0:N-1);

stem(ax1,f,x_n+noise(1:32))
title(ax1,'noisy signal x(n) N = 32 ')
ylabel(ax1,'noise(n)+x(n)')
xlabel(ax1,'n')
xlim([0,32])

ax2 = nexttile;

N=64;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
f = (0:N-1);

stem(ax2,f,x_n+noise(1:64))
title(ax2,'noisy signal x(n) N = 64 ')
ylabel(ax2,'noise(n)+x(n)')
xlabel(ax2,'n')
xlim([0,64])

ax3 = nexttile;

N=128;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
f = (0:N-1);

stem(ax3,f,x_n+noise(1:128))
title(ax3,'noisy signal x(n) N = 128 ')
ylabel(ax3,'noise(n)+x(n)')
xlabel(ax3,'n')
xlim([0,128])

figure
tiledlayout(3,1)
```

```
ax1 = nexttile;

N=32;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n+noise(1:N))); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax1,f,y_n)
xticklabels(0:5:35)
title(ax1,'magnitude spectrum N = 32 ')
ylabel(ax1,'|{x(n)}|')
xlabel(ax1,'Frequency (kHz)')

ax2 = nexttile;

N=64;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n+noise(1:N))); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax2,f,y_n)
xticklabels(0:5:35)
title(ax2,'magnitude spectrum N = 64 ')
ylabel(ax2,'|{x(n)}|')
xlabel(ax2,'Frequency (kHz)')

ax3 = nexttile;

N = 128;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n+noise(1:N))); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax3,f,y_n)
xticklabels(0:5:35)
title(ax3,'magnitude spectrum N = 128 ')
ylabel(ax3,'|{x(n)}|')
xlabel(ax3,'Frequency (kHz)')


% (e) Repeat the steps in (d) but for a WGN signal of mean 1 and variance of
10.
% Comments on your results.

figure
tiledlayout(3,1)

ax1 = nexttile;

N=32;
n = 0:1/Fs:(N-1)/Fs;
```

```
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
f = (0:N-1);
stem(ax1,f,x_n)
xticklabels(0:5:35)
title(ax1,'signal x(n) N = 32 ')
ylabel(ax1,'x(n)')
xlabel(ax1,'n')
xlim([0,32])

ax2 = nexttile;

noise1 = 10*randn(1,4*N)+1;
stem(ax2,f,noise1(1:32),'r')
title(ax2,'gaussian noise of 1 mean and 10 variance ')
ylabel(ax2,'noise(n)')
xlabel(ax2,'n')
xlim([0,32])

ax3 = nexttile;
stem(ax3,f,x_n+noise1(1:32))
title(ax3,'noisy signal x(n) ')
ylabel(ax3,'noise(n)+x(n)')
xlabel(ax3,'n')
xlim([0,32])

figure
tiledlayout(3,1)

ax1 = nexttile;

N=32;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
f = (0:N-1);

stem(ax1,f,x_n+noise1(1:32))
title(ax1,'noisy signal x(n) N = 32 ')
ylabel(ax1,'noise(n)+x(n)')
xlabel(ax1,'n')
xlim([0,32])

ax2 = nexttile;

N=64;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
f = (0:N-1);

stem(ax2,f,x_n+noise1(1:64))
title(ax2,'noisy signal x(n) N = 64 ')
ylabel(ax2,'noise(n)+x(n)')
xlabel(ax2,'n')
xlim([0,64])

ax3 = nexttile;
```

```
N=128;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
f = (0:N-1);

stem(ax3,f,x_n+noise1(1:128))
title(ax3,'noisy signal x(n) N = 128 ')
ylabel(ax3,'noise(n)+x(n)')
xlabel(ax3,'n')
xlim([0,128])

figure
tiledlayout(3,1)

ax1 = nexttile;

N=32;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n+noise1(1:N))); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax1,f,y_n)
xticklabels(0:5:35)
title(ax1,'magnitude spectrum N = 32 ')
ylabel(ax1,'|{x(n)}|')
xlabel(ax1,'Frequency (kHz)')

ax2 = nexttile;

N=64;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n+noise1(1:N))); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax2,f,y_n)
xticklabels(0:5:35)
title(ax2,'magnitude spectrum N = 64 ')
ylabel(ax2,'|{x(n)}|')
xlabel(ax2,'Frequency (kHz)')

ax3 = nexttile;

N = 128;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
y_n = abs(fft(x_n+noise1(1:N))); % magnitude spectrum
f = (0:N-1)*Fs/N;

plot(ax3,f,y_n)
xticklabels(0:5:35)
title(ax3,'magnitude spectrum N = 128 ')
```

```
ylabel(ax3,'|{x(n)}|')
xlabel(ax3,'Frequency (kHz)')

% (f) For the noisy signal as in (d) with N= 64 and 128, study the effects of
different
% windows (e.g., Hamming, Hann and Blackman). Comments on your results.

figure
tiledlayout(3,1)

ax1 = nexttile;

N=64;
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
signal = (x_n+noise(1:N)).*transpose(hann(N));
y_n = abs(fft(signal)); % magnitude spectrum

f = (0:N-1)*Fs/N;

plot(ax1,f,y_n)
xticklabels(0:5:35)
title(ax1,'magnitude spectrum Hann windowed N = 64 ')
ylabel(ax1,'|{x(n)}|')
xlabel(ax1,'Frequency (kHz)')

ax2 = nexttile;

signal = (x_n+noise(1:N)).*transpose(hamming(N));
y_n = abs(fft(signal)); % magnitude spectrum

plot(ax2,f,y_n)
xticklabels(0:5:35)
title(ax2,'magnitude spectrum hamming windowed N = 64 ')
ylabel(ax2,'|{x(n)}|')
xlabel(ax2,'Frequency (kHz)')

ax3 = nexttile;

signal = (x_n+noise(1:N)).*transpose(blackman(N));
y_n = abs(fft(signal)); % magnitude spectrum

plot(ax3,f,y_n)
xticklabels(0:5:35)
title(ax3,'magnitude spectrum blackman windowed N = 64 ')
ylabel(ax3,'|{x(n)}|')
xlabel(ax3,'Frequency (kHz)')

figure
tiledlayout(3,1)

ax1 = nexttile;

N=128;
```

```
n = 0:1/Fs:(N-1)/Fs;
x_n = sin(2*pi*f1*n) + sin(2*pi*f2*n) + 2*sin(2*pi*f3*n);
signal = (x_n+noise(1:N)).*transpose(hann(N));
y_n = abs(fft(signal)); % magnitude spectrum

f = (0:N-1)*Fs/N;

plot(ax1,f,y_n)
xticklabels(0:5:35)
title(ax1,'magnitude spectrum Hann windowed N = 128 ')
ylabel(ax1,'|{x(n)}|')
xlabel(ax1,'Frequency (kHz)')

ax2 = nexttile;

signal = (x_n+noise(1:N)).*transpose(hamming(N));
y_n = abs(fft(signal)); % magnitude spectrum

plot(ax2,f,y_n)
xticklabels(0:5:35)
title(ax2,'magnitude spectrum hamming windowed N = 128 ')
ylabel(ax2,'|{x(n)}|')
xlabel(ax2,'Frequency (kHz)')

ax3 = nexttile;

signal = (x_n+noise(1:N)).*transpose(blackman(N));
y_n = abs(fft(signal)); % magnitude spectrum

plot(ax3,f,y_n)
xticklabels(0:5:35)
title(ax3,'magnitude spectrum blackman windowed N = 128 ')
ylabel(ax3,'|{x(n)}|')
xlabel(ax3,'Frequency (kHz)')
```

```
clear; close all;
% Assignment 2 q1
% Andrew Munro-West 18363572
%
% Problem 2: STFT for speech signal spectral analysis Download the “dft.wav”
% file from the course website, and figure out some information of the .wav
% file by using ‘wavread’ command in Matlab. E.g. find out the sampling
% rate and the length of the recording (in second); hear the speech by
% using either ‘sound’ or ‘wavplay’ command in Matlab.

A = load('wordsample.mat');
[y,Fs] = audioread('dft.wav');
%sound(y,Fs);

% (a) Demonstrate the time and frequency domain representations of the speech
signals.
```

```
% Download the data file 'wordSample.mat', which is part of the above speech
data
% with the same sampling rate, and load it into Matlab (e.g. load
wordSample.mat).
% Plot the data, plot the autocorrelation sequence.
x = 0:1/Fs:((size(y)-1)/Fs);
x2 = 0:1/Fs:((size(y)-1)*2/Fs);
figure
tiledlayout(2,1)

ax1 = nexttile;
plot(ax1,x,y)
xlabel(ax1,'time (s)')
ylabel(ax1,'data')
title(ax1,'plot of dft.wav')


ax2 = nexttile;
plot(ax2,x2,xcorr(y))
ylabel('r_y_y(l)')
title('autocorrelation of dft.wav')

% (b) Plot the STFT-based time-varying spectrum of the speech signal, using
the data in
% 'wordSample.mat'. Make the window length (i.e. the length of each segment)
to be
% 22.5 milliseconds (i.e. the window length N=Fs*22.5/1000).
%
N = Fs*22.5/1000
s = spectrogram(A.yy,N);

figure
spectrogram(A.yy,N,[],[],Fs,'yaxis')
title('spectrogram of same signal')

figure
stft(A.yy,Fs,'FFTLength',round(N));
```