Assignment 5: Discrete Fourier Transform (DFT) and Fast Fourier Transform

Colt Thomas

June 20, 2022

Book Questions

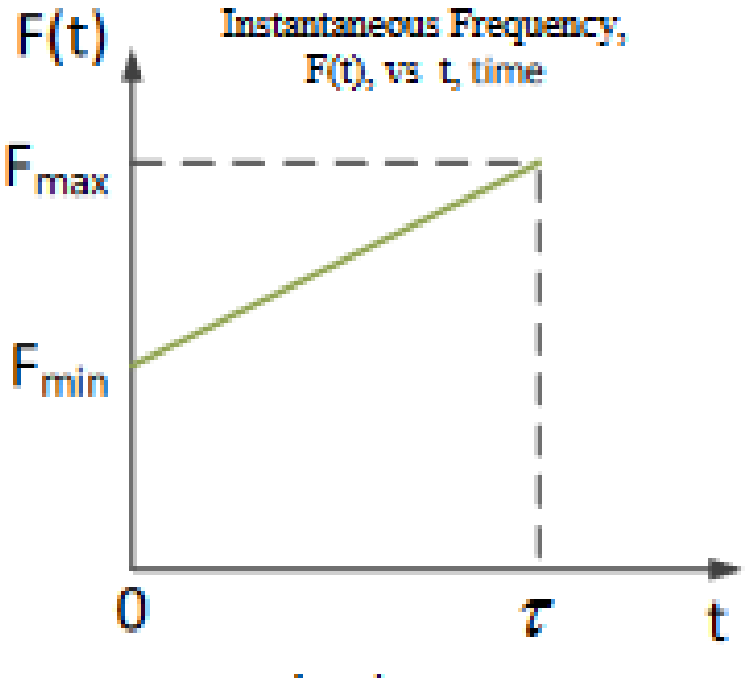
Problem 5.1

For this problem you are to create a linear frequency modulated (LFM) cos signal that has a duration of $\tau = 1$ *msec. The linear frequency starts at 10 KHz and end with a frequency of 35 KHz. This signal can be generated two ways. First is to use the built in MATLAB chirp function which is documented in the subsequent code listing. The second approach is to create the signal using a cosine function. The argument of the cosine is determined as follows:*

$$y = cos(2\pi\phi) \text{where the angle } \phi \text{ has units of cycles}$$

The instantaneous frequency in cycles per sec or Hz is given by $F(t) = \frac{d\phi}{dt}$. *For something familiar consider a cosine signal with a constant frequency* f_o. *In that case,* $\frac{d\phi}{dt} = F_o$ *and* $\phi = F_o t$, *and the resulting signals is given by the usual formula* $y = cos(2\pi F_o t)$. *A cosine signal with a linear instantaneous frequency is shown below.*

Therefore, $\frac{d\phi}{dt} = F(t) = 10 + 25t \leftarrow \phi = 10t + \frac{25}{2}t^2$ *and the LFM cosine signal is* $y = cos\big[2\pi\big(10t + \frac{25}{2}t^2\big)\big]$


F(t)
Instantaneous Frequency,
F(t), vs t, time
Fmax
Fmin
0
τ
t

Part A

Four plots are created vertically using the MATLAB "subplot" format. The first two subplotsdisplay the analog representation of the two signals as a function of time created by the two Circle the parts of the code that produce the first two subplots with a narrative explaining the process that produced them. What is the sample time used for representing the analog signals and how is it related to Fa in the MATLAB code?

The sample time used in this code is found on lines 14-15 below in Figure 1. We have two sample frequencies being used: Fa and Fs. Fa is set to 2MHz and is used to create the time vector t. Fa is also responsible for the resolution of the analog representation of the chirp function, since everything done in MATLAB is technically digital. Fa has to be significantly higher than the frequencies being represented to simulate an actual sampling of an analog signal. Fs is the sample frequency that is being used for the DSP itself, and is set to 75KHz. This gives us a Nyquist frequency of 37.5KHz, which is above Fmax. With Fs we can reconstruct the chirp signal without loss of information.

In Figures 1 and 2 I captured snippets of the subplot creations and circled the chirp function declarations.

```
12 -    tau = 1; %Pulse Width (Pulse Duration)in msec
13 -    Fmin =10; Fmax=35;  %KHz
14 -    Fa = 2000;  %Sample frequency (KHz) to represent CT (analog) in plot
15 -    Fs=75;Ts=1/Fs;%Sample Frequency (KHz) for DSP
16
17      %Using chirp Fcn to simulate analog signal
18 -    t = 0:1/Fa:tau;
19 -    x = chirp(t,Fmin,t(end),Fmax);
20 -    figure(1)
21 -    set(gcf,'Position',[1256 300  560 680])
22
23 -    subplot(4,1,1)
24 -    plot(t,x,'-','markersize',2,'Linewidth',2)
25 -    set(gca,'Ylim',1.5*[-1 1])
26 -    title(['Signal=chirp function with PulseWidth = ',num2str(tau),' msec'])
27 -    ylabel('x(t)'), xlabel('t (msec)')
```

Figure 1: Creation of a chirp using the MATLAB 'chirp' command.

```
29      %Using cos(phi) to compare with analog chirp
30 -    subplot(4,1,2)
31 -    xx=cos(2*pi*(10*t+(25/2)*t.^2));
32 -    plot(t,xx,'-','markersize',2,'Linewidth',2)
33 -    set(gca,'Ylim',1.5*[-1 1])
34 -    title(['Signal=cos(\phi) function with PulseWidth = ',num2str(tau),' msec'])
35 -    ylabel('x(t)'), xlabel('t (msec)')
```

Figure 2: Manual creation of a chirp.

Part B

Circle the code that creates the third subplot including a narrative explaining the process. What is the value of Fs and how does it figure in for the third subplot?

On line 39 is the creation of vector tt, which is used to simulate the sampling at 75KHz. This results in a sampling period of 13μs Sample points are represented as red dots on subplot 3 (see line 41).

```
37      %Sample points and Analog Signal
38 -    subplot(4,1,3)
39 -    tt = 0:1/Fs:tau;
40 -    xn=cos(2*pi*(10*tt+(25/2)*tt.^2));
41 -    plot(tt,xn,'ro','markersize',3,'Linewidth',2,'LineStyle','none')
42 -    hold on
43 -    plot(t,xx,'-k','Linewidth',1)
44 -    set(gca,'Ylim',1.5*[-1 1])
45 -    title(['Sampled cos(\phi) PulseWidth = ',num2str(tau),...
46          ' msec ','Fs = ',num2str(Fs),' KHz'])
47 -    ylabel('x(n)'), xlabel('t=n*Ts')
```

Figure 3: Subplot 3 MATLAB code. Circled is the representation of the sampling process in MATLAB.

Part C

Finally for the fourth subplot the DFT (caution–not the DTFT or the FFT) is computed for the discrete representation of the LFM signal. The magnitude of the DFT is displayed as a two sidedspectrum for physical frequency range of -Fs/2 to +Fs/2. Circle and explain the part of the code producing the spectrum explaining how the directly calculated spectral magnitude is adjusted for a two sided spectral display. Where and how is the discrete LFM signal saved for further use?

Circled in Figure 4 is the part of the MATLAB code that calculates and rearranges the magnitude of the spectrum X. Notice that P is essentially the DFT of X, but split in two and rearranged. Remember that the DFT is a sampled DTFT. Line 61 then saves the LFM signal with command "save 'sonar1' xn", where xn is the LFM signal itself.

```
1 %chirp_signals_script_1.m
2 %Initialize
3 close all, clear, clc
4 addpath('./MATLAB_INCLUDE/');
5 % Change default axes fonts and default text fonts.
6 set(0,'DefaultAxesFontName', 'Times New Roman')
```

```
%DFT
subplot(4,1,4)
N=length(tt);
n=[1:N];
nn=[-N/2:N/2-1];  %[[N/2+1:N],[1:N/2]]
X=dft(xn,N); magX=abs(X);
P=[magX(N/2+1:N),magX(1:N/2)]; % rearrange magX for two sided spectrum
plot(nn*Fs/N,P,'-o','MarkerSize',4,'MarkerEdgeColor','k')
title(['DFT[(x(n)] PulseWidth = ',num2str(tau),...
    ' msec  ','Fs = ',num2str(Fs),'KHz'])
ylabel('|X(F)|'),xlabel('F(KHz)')
```

Figure 4: Subplot 4 MATLAB code.

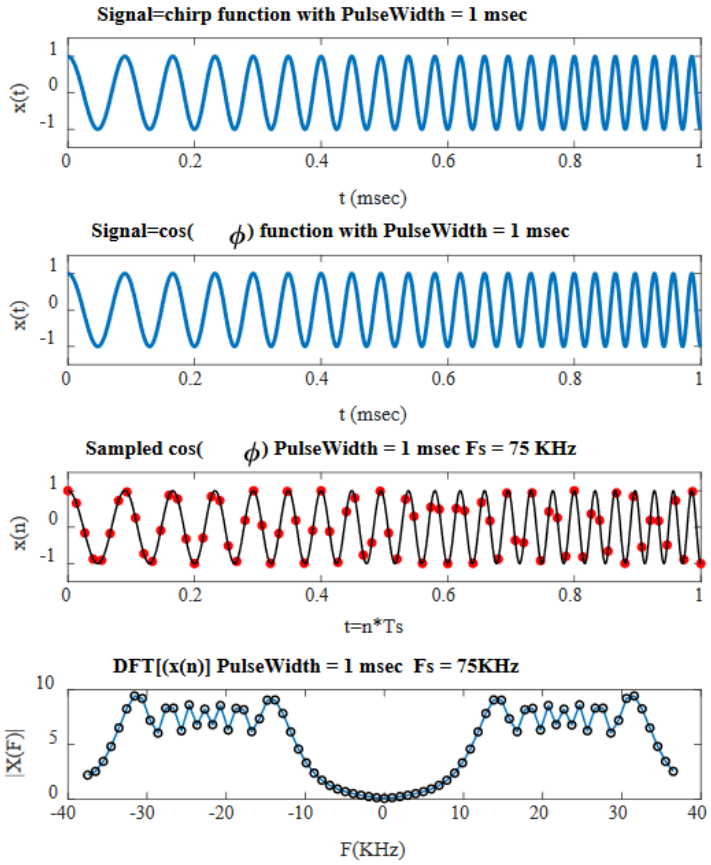


Figure 5: Subplots of a chirp signal from 10KHz to 35KHz for the duration of $1\mu s$, with the DFT in the final subplot.

```
7  set(0,'DefaultAxesFontSize', 15)
8  set(0,'DefaultTextFontname', 'Times New Roman')
9  set(0,'DefaultTextFontSize', 15)
10 set(0, 'DefaultLineLineWidth', 1);
11
12 tau = 1; %Pulse Width (Pulse Duration)in msec
13 Fmin =10; Fmax=35;  %KHz
14 Fa = 2000;  %Sample frequency (KHz) to represent CT (
       analog) in plot
15 Fs=75;Ts=1/Fs;%Sample Frequency (KHz) for DSP
16
17 %Using chirp Fcn to simulate analog signal
18 t = 0:1/Fa:tau;
19 x = chirp(t,Fmin,t(end),Fmax);
20 figure(1)
21 set(gcf,'Position',[1256 300  560 680])
22
23 subplot(4,1,1)
24 plot(t,x,'-','markersize',2,'Linewidth',2)
25 set(gca,'Ylim',1.5*[-1 1])
26 title(['Signal=chirp function with PulseWidth = ',num2str
       (tau),' msec'])
27 ylabel('x(t)'), xlabel('t (msec)')
28
29 %Using cos(phi) to compare with analog chirp
30 subplot(4,1,2)
31 xx=cos(2*pi*(10*t+(25/2)*t.^2));
32 plot(t,xx,'-','markersize',2,'Linewidth',2)
33 set(gca,'Ylim',1.5*[-1 1])
34 title(['Signal=cos(\phi) function with PulseWidth = ',
       num2str(tau),' msec'])
35 ylabel('x(t)'), xlabel('t (msec)')
36
37 %Sample points and Analog Signal
38 subplot(4,1,3)
39 tt = 0:1/Fs:tau;
40 xn=cos(2*pi*(10*tt+(25/2)*tt.^2));
41 plot(tt,xn,'ro','markersize',3,'Linewidth',2,'LineStyle',
       'none')
42 hold on
43 plot(t,xx,'-k','Linewidth',1)
44 set(gca,'Ylim',1.5*[-1 1])
45 title(['Sampled cos(\phi) PulseWidth = ',num2str(tau),...
46      ' msec ','Fs = ',num2str(Fs),' KHz'])
47 ylabel('x(n)'), xlabel('t=n*Ts')
48
```

```
%DFT
subplot(4,1,4)
N=length(tt);
n=[1:N];
nn=[-N/2:N/2-1];   %[[N/2+1:N],[1:N/2]]
X=dft(xn,N); magX=abs(X);
P=[magX(N/2+1:N),magX(1:N/2)]; % rearrange magX for two
    sided spectrum
plot(nn*Fs/N,P,'-o','MarkerSize',4,'MarkerEdgeColor','k')
title(['DFT[(x(n)] PulseWidth = ',num2str(tau),...
     ' msec  ','Fs = ',num2str(Fs),'KHz'])
ylabel('|X(F)|'),xlabel('F(KHz)')

save 'sonar1' xn
```

Problem 5.2

In this problem you are to simulate the dsp processing of a received signal associated with a sonar system. The system consists of an acoustic transmitter (called a transponder) that starts with a discrete signal representation and is converted via D2A converter to a burst (finite time) of an analog signal that is radiated into the water column. Upon hitting a target the transmitted analog signal is reflected back to the sonar system where it is received and digitized with a A2D converter. The discrete received signal is then correlated with the discrete representation of the transmitted signal. The index where the correlation is high equates to the two way delay to and from the target. If the signal is chirped then the correlation process produces a higher precision measurement of the delay. The system is an example of a LTI system and the correlation receiver is more commonly called a matched filter.

Part A

You are to examine the received (incoming discrete) signal, designated as SonarRcv(n). It consists of a discrete replica of the transmitted burst, designated SonarXmit(m) which is provide as MATLAB data with the file name sonar1.mat. It is also used as a correlation template to compare with the incoming received signal.. The received signal SonarRcv(n), is created by concatenating zeros before and after SonarXmit which now represents the incoming signal from the target. A zn array of zeros appended to the front of the SonarXmit signal represents the delay for the echo. For completeness we need to append zeros after the SonarXmit. Therefore, SonarRcv(n)=[zeros(1,K), SonarXmit(m),zeros(1,L)]. For simulation purposes let K=L=30. The discrete received signal array should have a length of K+M+L where M=length(SonarXmit). Plot the discrete SonarRcv signal using MATLAB

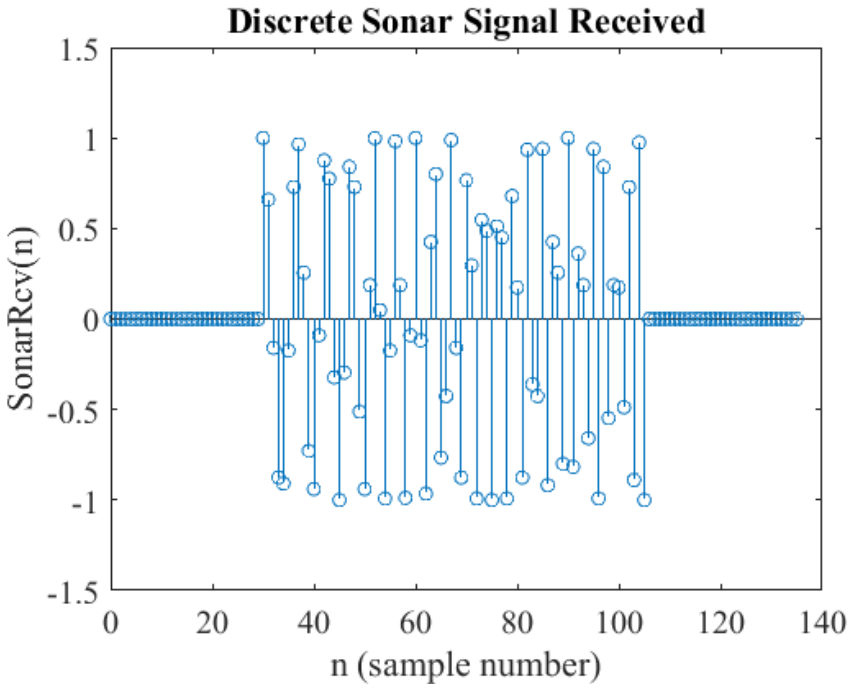


Figure 6: Discrete received sonar signal representation.

Part B

The output of a correlation receiver can be computed using the convolution operation described in earlier modules. For example if T is the discrete transmitted signal and R is the discrete received signal (including zeros sections for range delays) then the correlation output, y equals

$$y = conv[T(-n), R(n)] = conv[T(n), R(-n)]$$

Since the convolution output for an LTI represents the direct response of a filter, then we can simulate the matched filter operation using the MATLAB function y=filter(B,A,x) where the A array is just 1 that the filter output is a zero state operation. B is just the discrete transmitted signal, T, and x is the received signal R. The –n argument is accounted for by flipping the affected array using the MATLAB command "fliplr." The matched filter output, y, would be given by

$$y = filter(fliplr(T)1, R)$$

The transmitted sonar signal is saved as sonar1.mat and included with the module 5 MATLAB files. Write and execute a MATLAB mfile with plots of the sonar transmission signals, the received signal and the matched filter output. Plot the three graphs using the subplot in a vertical format.

Below in Figure 7 is the plot of the original transmitted signal SonarXmit, the received SonarRcv, and the correlation y between the two signals. Noticed that the peak indicates where the transmitted and received signal line up in the correlation process. This can be used to detect the original signal, and the time between transmission and reception can be used to calculate the distance between the source and the target. See the appendix for the code used.

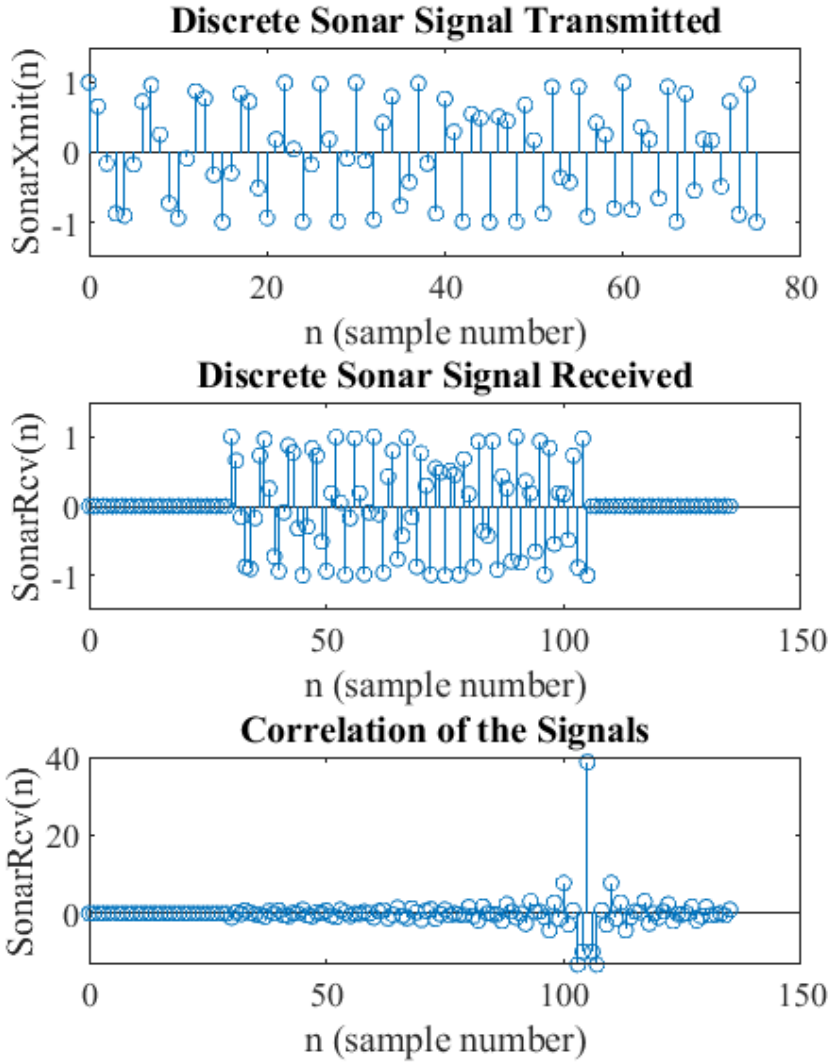


Figure 7: Discrete received sonar signal representation.

Problem 5.3

a.)Use MATLAB to do problem 7.8 (text p. 503) and graphically display results

See the code in the appendix under under the Problem 5.3 header. Below in Figure 8 there is a visual that lines up the flipped and shifted $x2(m-n)$ with $x1(n)$. By multiplying and summing all the elements you obtain $x3(n) = [17, 19, 22, 19]$.

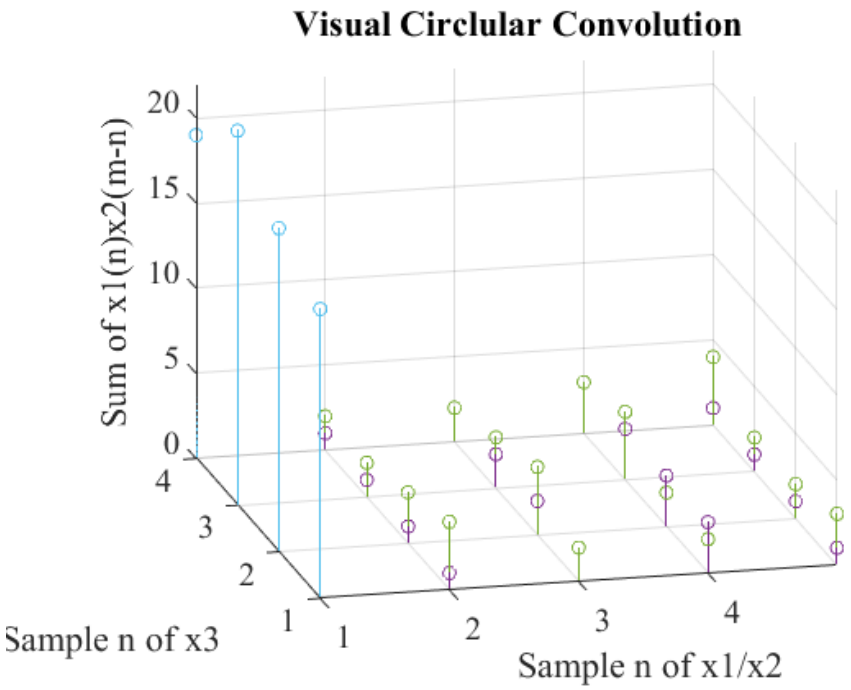


Figure 8: Discrete received sonar signal representation.

b.)Use MATLAB to do problem 7.9 (text p. 503) and graphically display results Comment on similarities or differences of parts a and b

Another way to find the circular convolution of two sequences is to perform the DFT and multiply the results together. This is done in the part b of section 5.3 in the MATLAB code in the appendix. The only difference between the two results is that when we do the IDFT on X3, we get complex numbers back... none of which have any imaginary components.

Problem 5.4

Part A

Assume that a complex multiply takes 1μs and that the amount of time to compute a DFT or FFT is determined by the amount of time it takes to perform all of the multiplications.

a) How much time does it take to compute a 1024-point DFT directly?

The DFT can be computed as:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, 0 <= k <= N-1$$

Where $W_n^{kn} = e^{-j2\pi/N}$. For each value of k, there are N complex multiplications (or 4N real multiplies). And since there are N values of k we are solving for, we have a big-O time of N^2. So for a 1024 point DTF it would take 1.048576 seconds to compute.

b) How much time is required if an FFT is used?

This will depend on what type of FFT we do, but let's look at the Radix-2 FFT algorithm. Note that N must be a factor of 2, which is good since we are dealing with N=1024. We split x(n) up into an $f_1(n) = x(2n)$ and $f_2(n) = x(2n+1)$ and then compute a DFT:

$$X(k) = \sum_{n \text{ even}} x(n) W_N^{kn} + \sum_{n \text{ odd}} x(n) W_N^{kn}$$

You can see pg 519 of the text book for the full breakdown of the algorithm, but it can be simplified to:

$$X(k) = F_1(k) + W_N^k F_2(k)$$

If we do this single breakdown we are left with a complexity of $\left(\frac{N}{2}\right)^2$ But we can go further and break down $F_1(k)$ and $F_2(k)$ until they can't be divided by two anymore. Completing the Radix-2 FFT will result in $(N/2)log_2 N$ complex multiplies, which comes out to a computation time of 5.120ms.

c) Repeat parts A and B for a 4096-point DFT

DFT comes out to 4096^2 complex multiplies, or 16.777216 seconds. FFT comes out to 24.576ms.

Part B

Sampling a continuous-time signal, xa(t), for 1s generates a sequence of 4096 samples

a) What is the highest frequency in xa(t) if it was sampled without aliasing?

We sample at a rate of 4096 samples per second. This means that the Nyquist rate is half that, at 2048 samples per second, or 2048 Hz.

b) If the 4096 point FFT of the sampled signal is computed, what is the frequency spacing, in Hertz, between the output points?

For a 4096 point FFT we have 2048 bins. Our highest represented frequency will be 2048hz. Our frequency resolution can be represented as:

$$\frac{F_{nyquist}}{N/2 \text{ bins}} = \frac{2048}{2048} = 1Hz$$

Problem 5.5

In MATLAB construct a signal consisting of four parts: three sinusoid components and one noise component. The signal is given by:

$$x(n) = A_1 cos(2\pi F_1 T_s) + A_2 sin\left(2\pi F_2 T_s n + \frac{\pi}{4}\right) + A_3 sin(2\pi F_3 T_s n) + Noise(n)$$

You are to calculate and display the magnitude (in dB) of the spectrum as a function frequency for two window functions. The first is to be a rectangular window and the second is to be a Bartlett (Triangular) window. In both cases comment on your observations of the displayed results. The noise is Gaussian or Normal with zero mean.

$F_1 = 46.00KHz, \quad A_1 = 1.0 \quad Strong\ Signal$
$F_2 = 46.38KHz, \quad A_2 = 0.4 \quad Signal\ nearby\ in\ frequency$
$F_3 = 54.00KHz, \quad A_3 = .002\ Weak\ Signal\ atDistant\ frequency$
$NoiseMean = 0$
$NoiseVariance = 10^{-5}$
$Fs = sample\ frequency = 200KHz$
$N = FFT\ Lemgth = 1024$

In Figures 9 and 10 we can see the difference between the two windows. The rectangular window shows a higher magnitude of the DFT and is able to

pick out F_3 at 54KHz. the only downside is the higher noise floor, or spectral leakage. The Bartlett window has a better noise floor but also diminishes the signals.

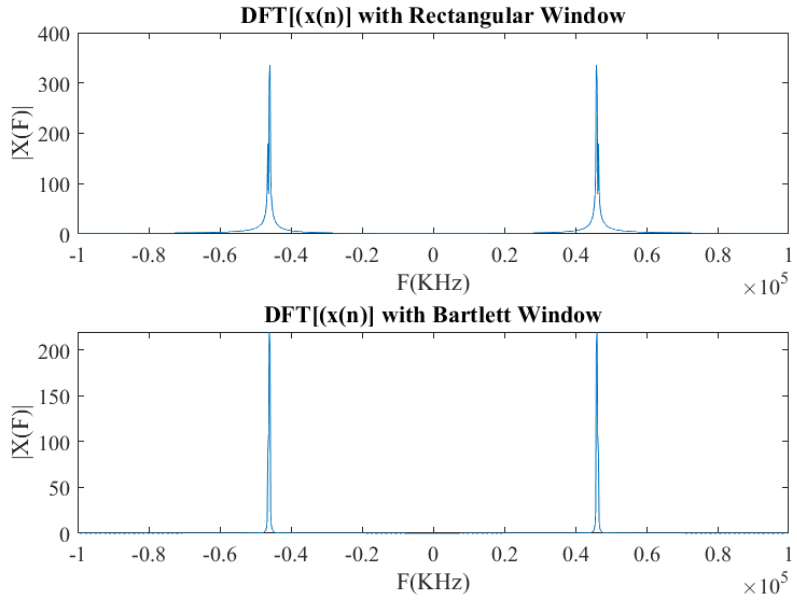


Figure 9: Windowed $DFT[x(n)]$ plots as a function of frequency. Notice the difference of spectral leakage and magnitude.

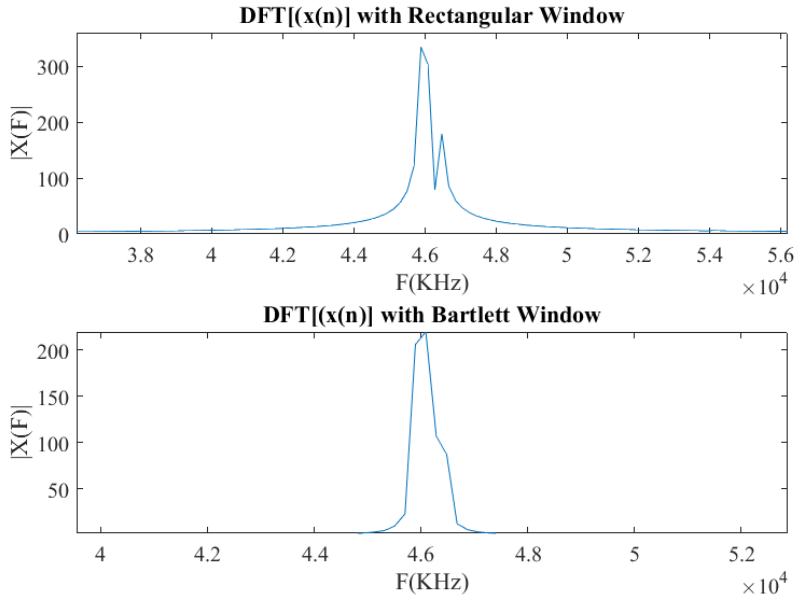


Figure 10: Windowed $DFT[x(n)]$ plots as a function of frequency. Notice that the rectangular window is able to pick out frequencies that are close to each other.

Matlab Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Problem 5.2
%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
load sonar1.mat
% ------------
% - Part A -
% ------------

K = 30;
L = K;
SonarXmit = xn;
SonarRcv =[zeros(1,K), SonarXmit,zeros(1,L)];
M = length(SonarXmit);
N = length(SonarRcv);

n = (0:N-1); % samples
% subplot(4,1,1)
figure(1)
stem(n,SonarRcv)
set(gca,'Ylim',1.5*[-1 1])
title('Discrete Sonar Signal Received')
ylabel('SonarRcv(n)'), xlabel('n (sample number)')

% ------------
% - Part B -
% ------------

% autocorrelation output
% y = conv(flip(SonarRcv),SonarXmit);
y = filter(flip(SonarXmit),1,SonarRcv);

figure(2)
n = (0:M-1); % samples
subplot(3,1,1)
stem(n,SonarXmit)
set(gca,'Ylim',1.5*[-1 1])
title('Discrete Sonar Signal Transmitted')
ylabel('SonarXmit(n)'), xlabel('n (sample number)')

subplot(3,1,2)
n = (0:N-1); % samples
stem(n,SonarRcv)
```

```
44 set(gca,'Ylim',1.5*[-1 1])
45 title('Discrete Sonar Signal Received')
46 ylabel('SonarRcv(n)'), xlabel('n (sample number)')
47
48 subplot(3,1,3)
49 n = (0:length(y)-1); % samples
50 stem(n,y)
51 title('Correlation of the Signals')
52 ylabel('SonarRcv(n)'), xlabel('n (sample number)')
53
54 %%%%%%%%%%%%%%%%%%%%%%%%%
55 %% Problem 5.3
56 %%%%%%%%%%%%%%%%%%%%%%%%%
57
58 % ------------------------
59 % - Book problem 7.8 -
60 % ------------------------
61
62 % Calculating the circular convolution of x1,x2 with eq
       7.2.39
63 x1 = [1,2,3,1];
64 x2 = [4,3,2,2];
65 N = 4;
66 x3 = zeros(1,N);
67
68 % manual method for visibility:
69 %x2 = [x2(1),x2(4),x2(3),x2(2)]; % x2(-n)
70 x2 = [4,2,2,3]; % x2(-n) notice repeaded x2(n) =
       [...4,(3,2,2,{4},3,2,2},4...]
71 x3(1) = sum(x1.*x2);
72
73 x2 = [3,4,2,2]; % x2(1-n)
74 x3(2) = sum(x1.*x2);
75
76 x2 = [2,3,4,2]; % x2(2-n)
77 x3(3) = sum(x1.*x2);
78
79 x2 = [2,2,3,4]; % x2(3-n)
80 x3(4) = sum(x1.*x2);
81
82 % algorithmic method
83 x1 = [1,2,3,1];
84 x2 = [4,3,2,2];
85 x2= [x2(1),fliplr(x2(2:4))]; % x2(-n)
86 x3 = zeros(1,N);
87 for m=1:N
```

```
88      for n=1:N
89  %       x3(m) = x3(m) + (x1(n)*x2(N-n+1));
90          x3(m) = x3(m) + (x1(n)*x2(n));
91      end
92      x2 = [x2(2:4),x2(1)]; % shift
93  end
94  figure(3)
95  n=(1:4)
96  hold on;
97  stem(n,x1)
98  stem(n,x2)
99  stem(n,x3)
100 set(gca,'Ylim',1.5*[0 max(x3)])
101 set(gca,'Xlim',1.5*[0 N])
102 title('Discrete Sonar Signal Transmitted')
103 ylabel('SonarXmit(n)'), xlabel('n (sample number)')
104
105 figure(4)
106 X1 = [[NaN,x1];[NaN,x1];[NaN,x1];[NaN,x1]]
107 stem3(X1)
108 hold on;
109 X2 = [[NaN,[4,2,2,3]];[NaN,[3,4,2,2]];[NaN,[2,3,4,2]];[
        NaN,[2,2,3,4]]]
110 stem3(X2)
111 Z = [x3;NaN(1,4);NaN(1,4);NaN(1,4);NaN(1,4);NaN(1,4)]';
112 stem3(Z)
113 ylabel('Sample n of x3')
114 xlabel('Sample n of x1/x2')
115 zlabel('Sum of x1(n)x2(m-n)')
116 title('Visual Circlular Convolution')
117 % ------------------------
118 % - Book problem 7.9 -
119 % ------------------------
120
121 % Use the 4-point DFT and IDFT to determine the circular
        convolution
122 % of x1 and x2.
123 x1 = [1,2,3,1];
124 x2 = [4,3,2,2];
125 N=4;
126 X1 = dft(x1,N);
127 X2 = dft(x2,N);
128 X3 = X1.*X2;
129 x3 = abs(IDFT(X3,N));
130
131 %%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Problem 5.5
%%%%%%%%%%%%%%%%%%%%%%%%
A1=1.0;
A2=0.4;
A3=0.002;
F1=46.0e3;
F2=46.38e3;
F3=54.0e3;
Fs=200.0e3;
Ts=1/Fs;
N=1024;
NoiseMean=0;
NoiseVar=0;
Noise = NoiseVar.*randn(1,N) + NoiseMean;
n = (0:N-1);
x=A1*cos(2*pi*F1*Ts.*n)+A2*cos(2*pi*F2*Ts.*n+pi/4)+A3*cos
    (2*pi*F3*Ts.*n)+Noise;
figure(5)
plot(n,x)

% Windowing
w = ones(1,N); % rectangle window
xr = x.*w;
% xb = x.*bartlett(N)';
w = w*1.2;
xb = x.*w;
% Take the DFT
% X = dft(x,N);


subplot(2,1,1)
n=[1:N];
nn=[-N/2:N/2-1];  %[[N/2+1:N],[1:N/2]]
X=dft(xr,N); magX=abs(X);
P=[magX(N/2+1:N),magX(1:N/2)]; % rearrange magX for two
    sided spectrum
plot(nn*Fs/N,P)
title(['DFT[x(n)] with Rectangular Window'])
ylabel('|X(F)|'),xlabel('F(KHz)')

subplot(2,1,2)
n=[1:N];
nn=[-N/2:N/2-1];  %[[N/2+1:N],[1:N/2]]
X=dft(xb,N); magX=abs(X);
P=[magX(N/2+1:N),magX(1:N/2)]; % rearrange magX for two
    sided spectrum
```

```
175 plot(nn*Fs/N,P)
176 title(['DFT[(x(n)] with Bartlett Window'])
177 ylabel('|X(F)|'),xlabel('F(KHz)')
```