ECE320-001
SIGNALS AND SYSTEMS II

MATLAB PROJECT I – UPSAMPLING AND DOWNSAMPLING

AUTHOR: Ahad Al Hassan

Introduction

The purpose of this project is to work with upsampling and downsampling discrete-time signals. Our objective is to demonstrate our understanding of upsampling and downsampling using analytical calculations and then verifying our work with MATLAB. From there, we will take an audio signal and upsample/downsample it accordingy.

Results

2. Analytical Calculations

$$x[n] = \left\{ \frac{\sin(A\pi(n-D))}{A\pi(n-D)} \right\}^2$$

zero crossings

$\sin(A\pi(n-D)) = 0$

$A\pi(n-D) = k\pi \; ; k \in \mathbb{Z}$

$n = \frac{k}{A} + D$

zero crossings MATLAB

$n = \frac{2\alpha - 1}{A} + D \; , \alpha \in \mathbb{Z}$

amplitude at zero

$$\left\{ \frac{\sin(A\pi(n-D))}{A\pi(n-D)} \right\}^2$$

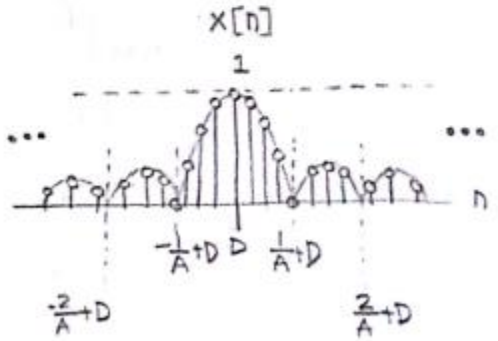
$$\frac{\sin(A\pi(n-D))}{A\pi(n-D)}\Big|_{n=D} \rightarrow \frac{0}{0}$$ L'Hospital's Rule

$$\frac{\frac{d}{dn}\{\sin(A\pi(n-D))\}}{\frac{d}{dn}\{A\pi(n-D)\}} \rightarrow \frac{\cos(A\pi(n-D))A\pi}{A\pi}\Big|_{n=D}$$

$\cos(A\pi(D-D))$

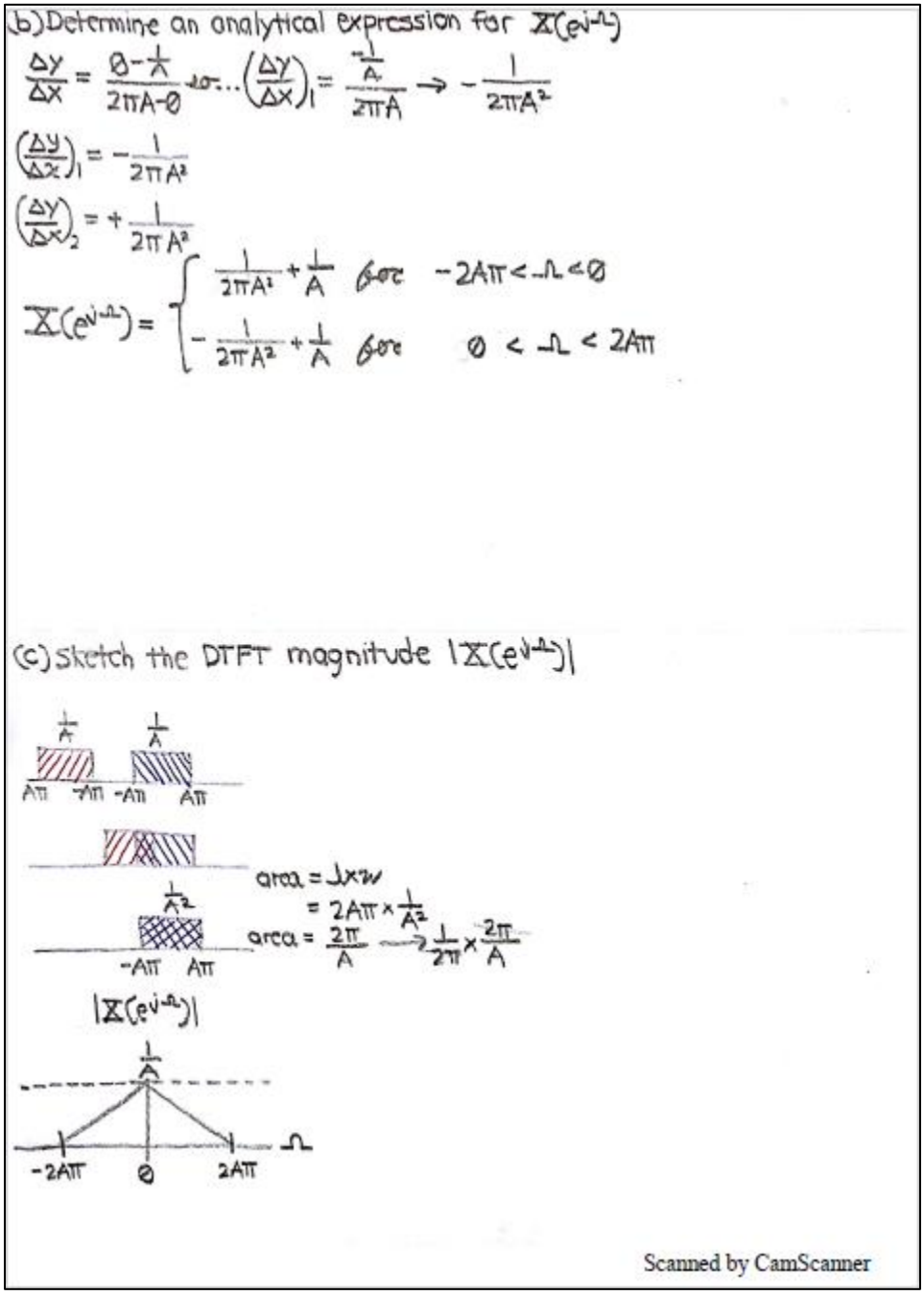
$\cos(0) = 1$

$(1)^2 = 1$



Scanned by CamScanner

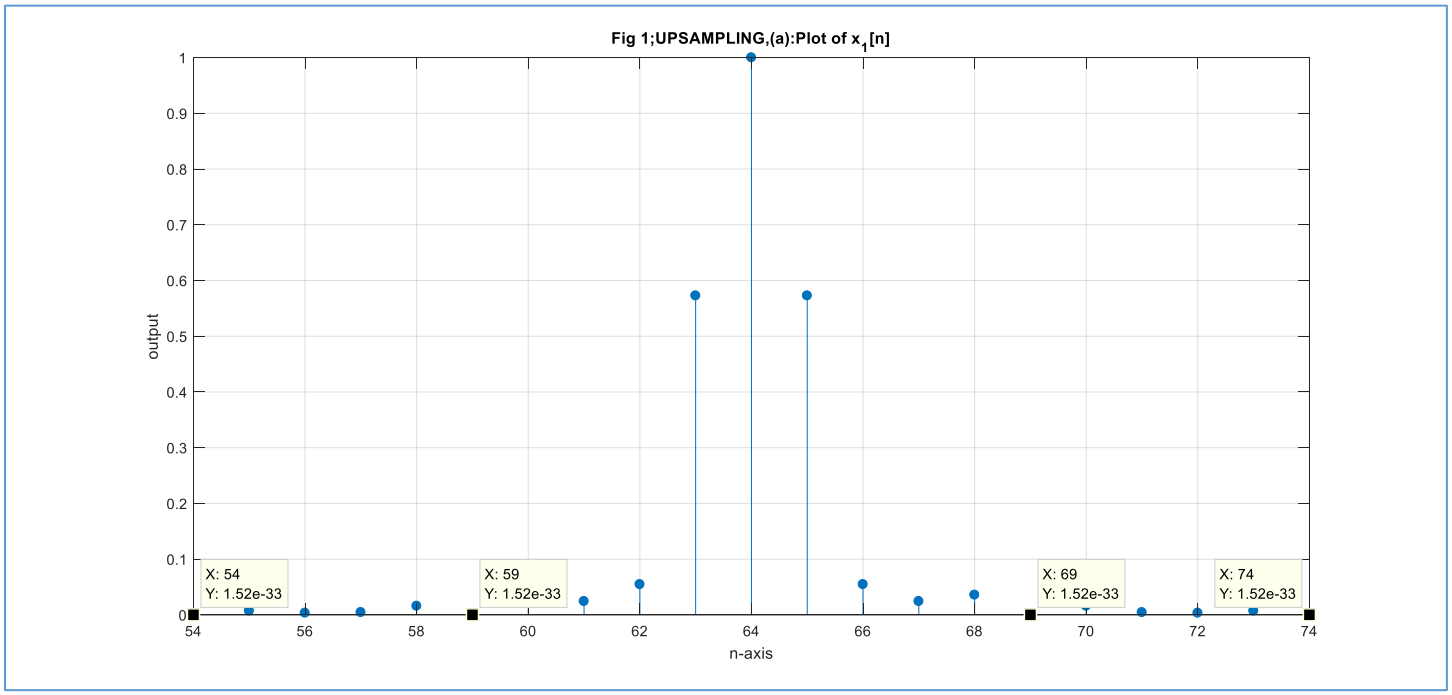
2. Analytical Calculations



NOTE TO THE READER: Since the derived equation for part (b) is an equation for the DFT, the DFT is periodic with 2π. Thus, the piecewise function expressed for part (b) is periodic with 2π

3. Upsampling

3a)



As we can see from this zoomed in plot of *figure 1*, the signal $x_1[n]$ has zero crossings for

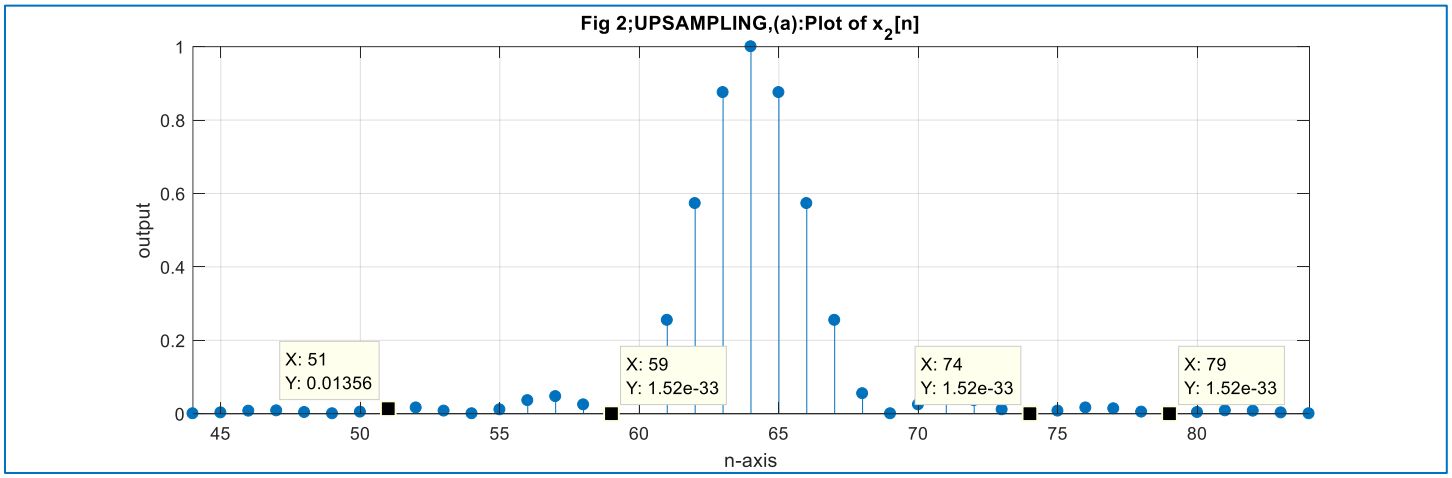
$$zero - crossings_{analytical} = \frac{\alpha}{A} + D, \alpha \epsilon \text{ Z (set of all real integers)}$$

Where A and D when $\alpha = 1$, we can find the first zero crossing at 66.5. Since discrete systems only have integer values for the $n - axis$, we cannot represent this number; therefore, we must calculate the next zero crossing for when $\alpha = 2$. This gives us a zero crossing for the values of $n = 59$ and $n = 69$. We can verify this from the plot above. When $\alpha = 3$, we obtain 71.5. This number isn't valid so we move on to $\alpha = 4$ and obtain 74 and 54, respectively. We can only express odd values of alpha. So, we can modify our equation above for discrete systems.

$$zero - crossings_{MATLAB} = \frac{(2\alpha - 1)}{A} + D$$

3. Upsampling

3a)

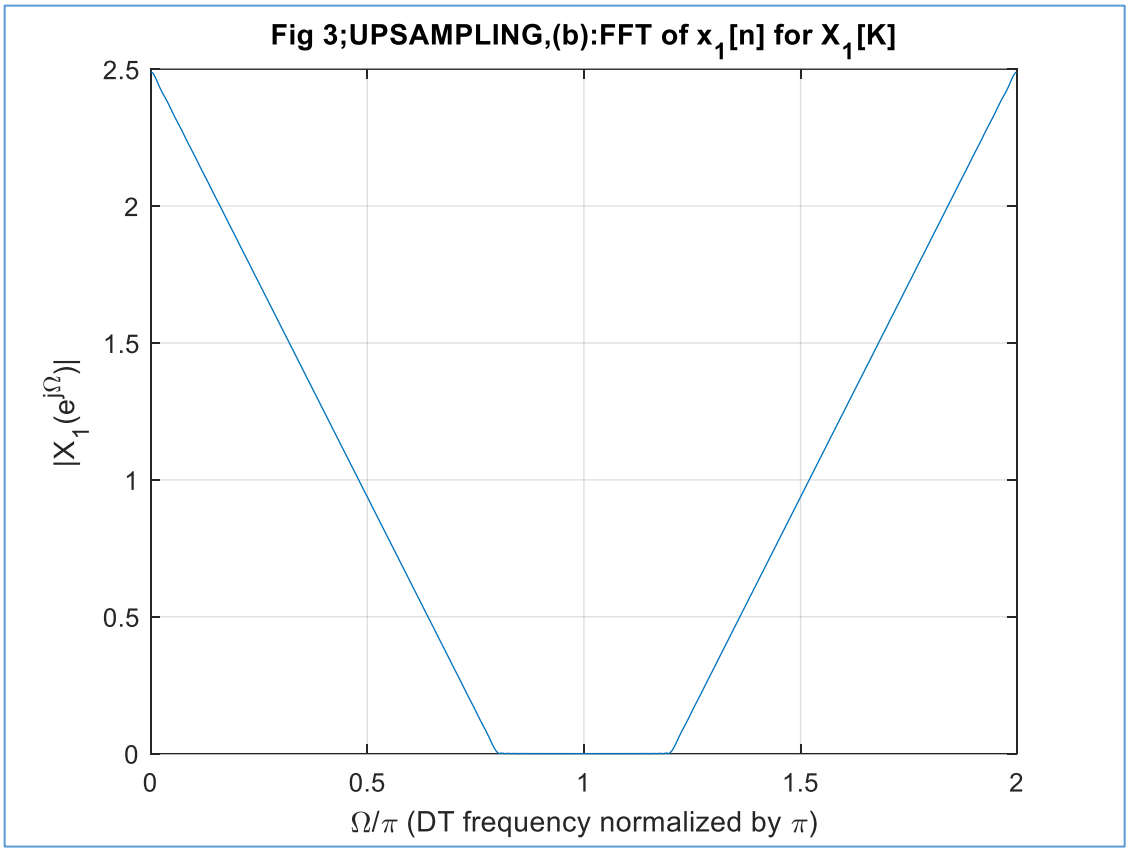


As shown above in *figure 2,* when $A = 0.2$, we can express all the zero crossings as integer values. So, the zero crossings for *figure 2* is the same from what we derived in our analytical calculations.

$$zero - crossings_{analytical} = \frac{\alpha}{A} + D, \alpha \epsilon \text{ Z (set of all real integers)}$$

3. Upsampling

3b)



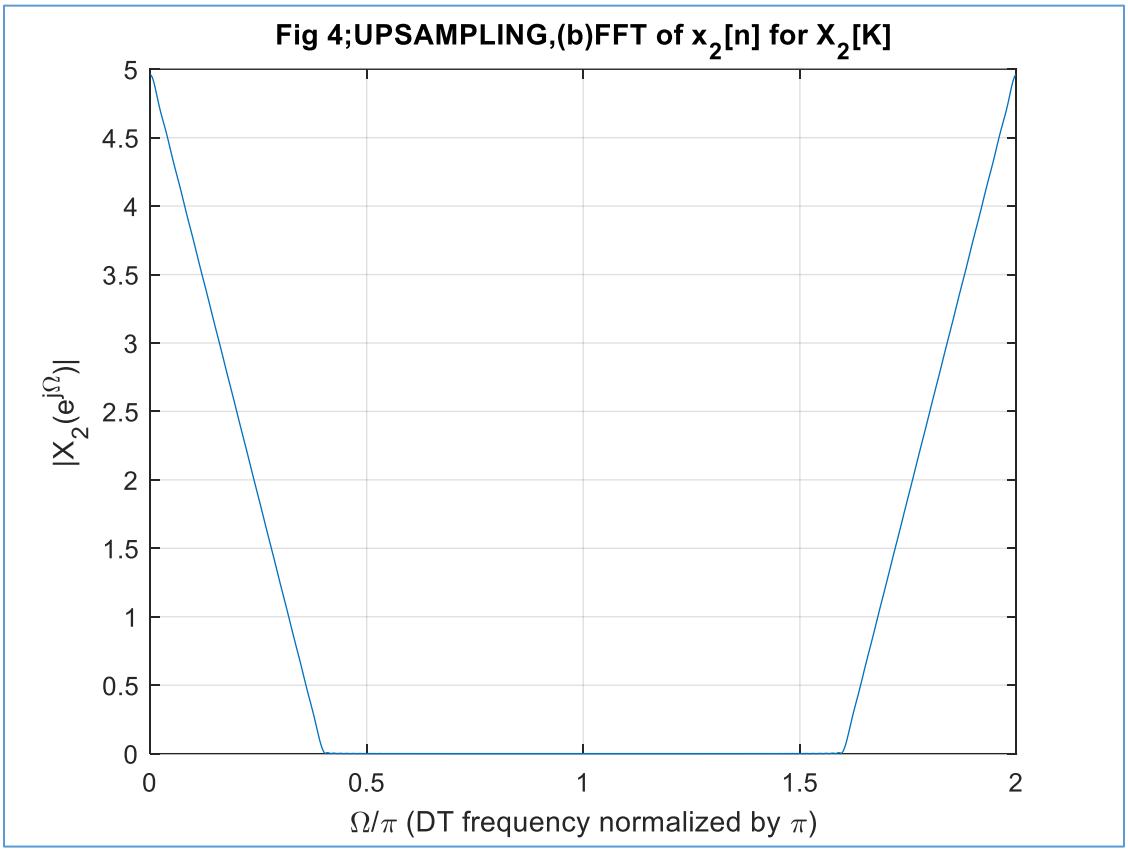
As we can see from *figure 3,* the height of the DFT of $x_1[n]$ is 2.50. This is to be expected, as we demonstrated in our *Ananlytical Calculations* the height of the Fourier transform should be

$$height(\omega = 0) = \frac{1}{A}$$

where A is a given parameter of signals $x_1[n]$ and x_2[n]. For $x_1[n]$ $A = 0.4$, so the height is expected to be 2.5, which is what we got in MATLAB.

3. Upsampling

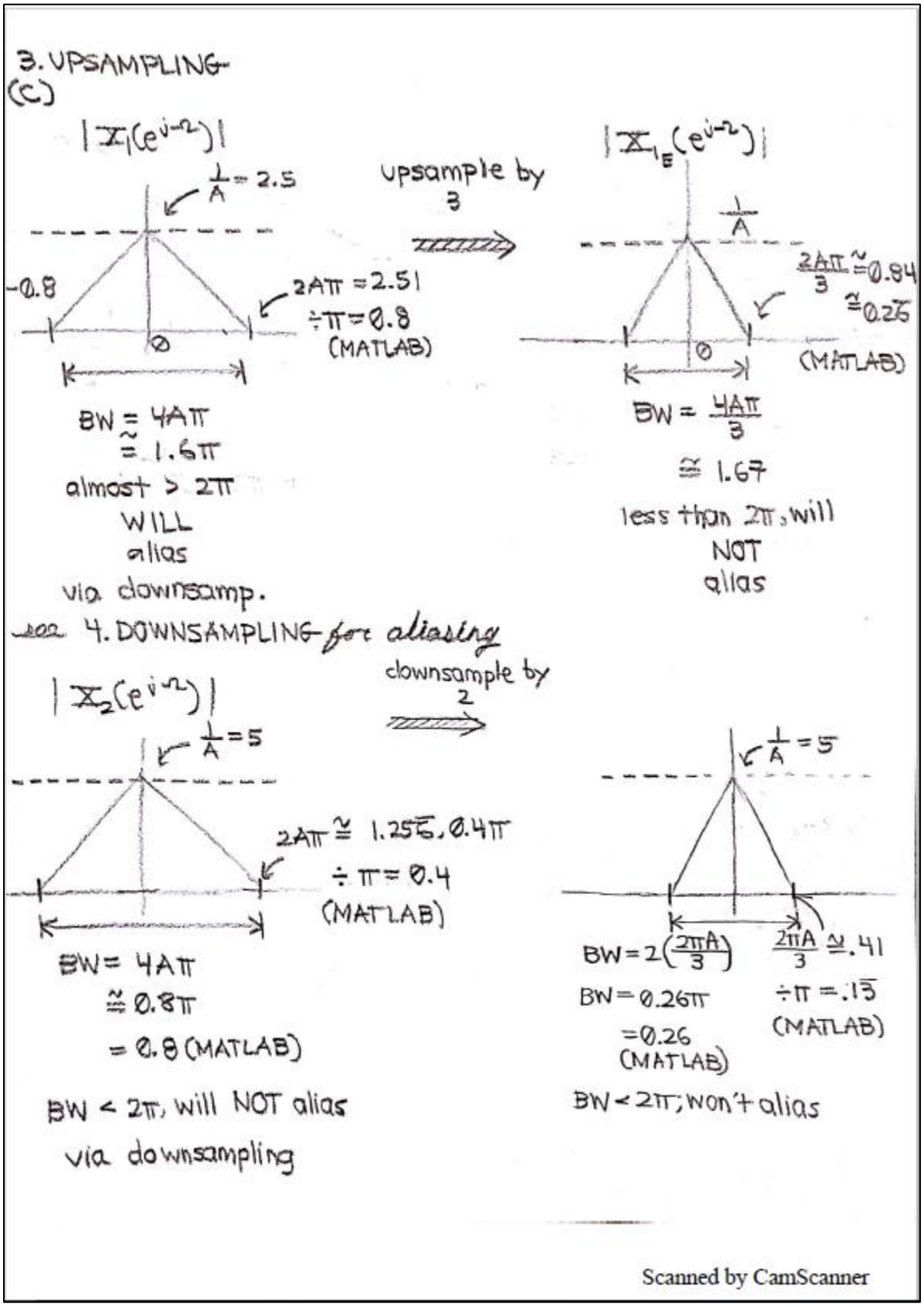
3b)



From our analysis of *figure 3,* we know that the height of the Fourier transform should have a value of $\frac{1}{A}$, where the value A comes from the signal. For the signal $x_2[n]$,we know that A =0.2. As a result, the height should be 5, which is what we get in MATLAB.

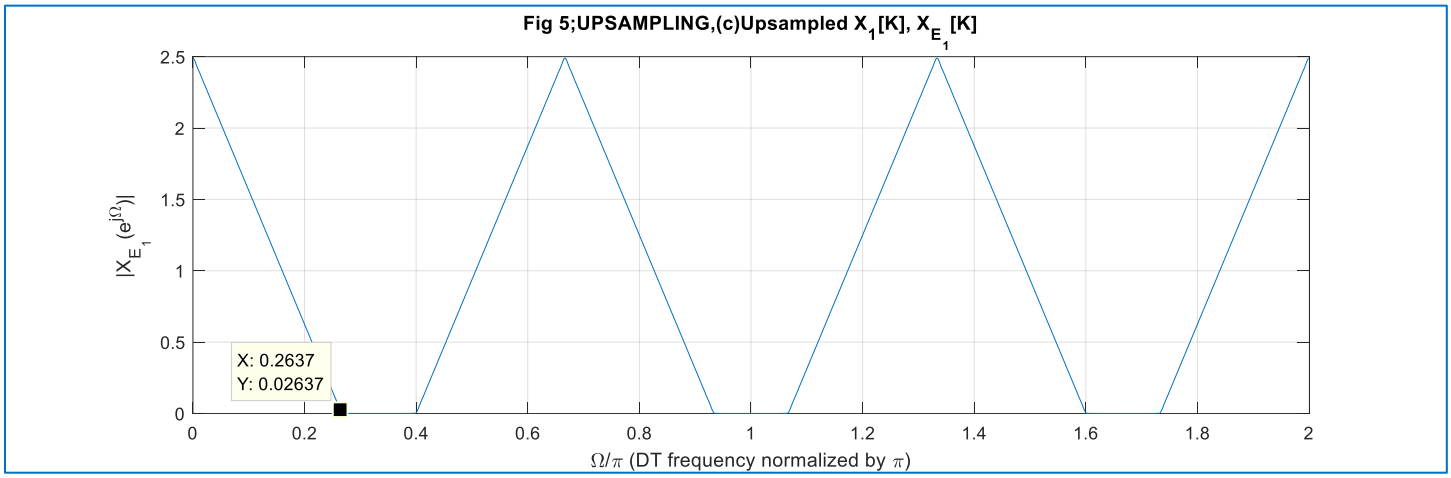
3. Upsampling

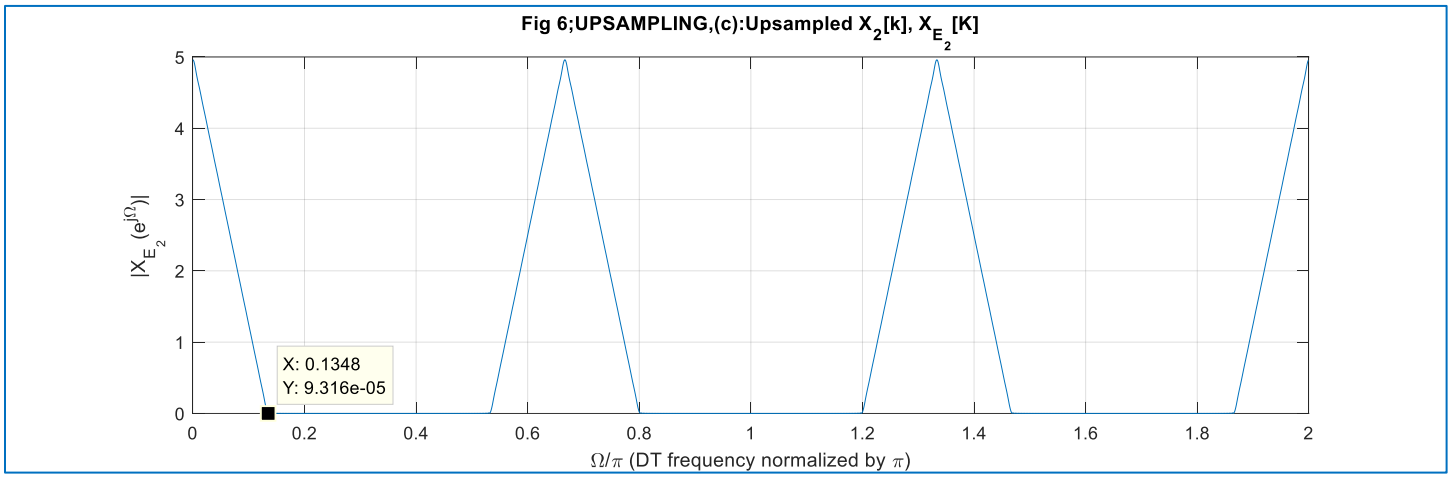
3c) Sketch of the magnitude of 2048-pt FFT upsampled signals $x_1[n]$ and $x_2[n]$ should look like



3. Upsampling

3c)

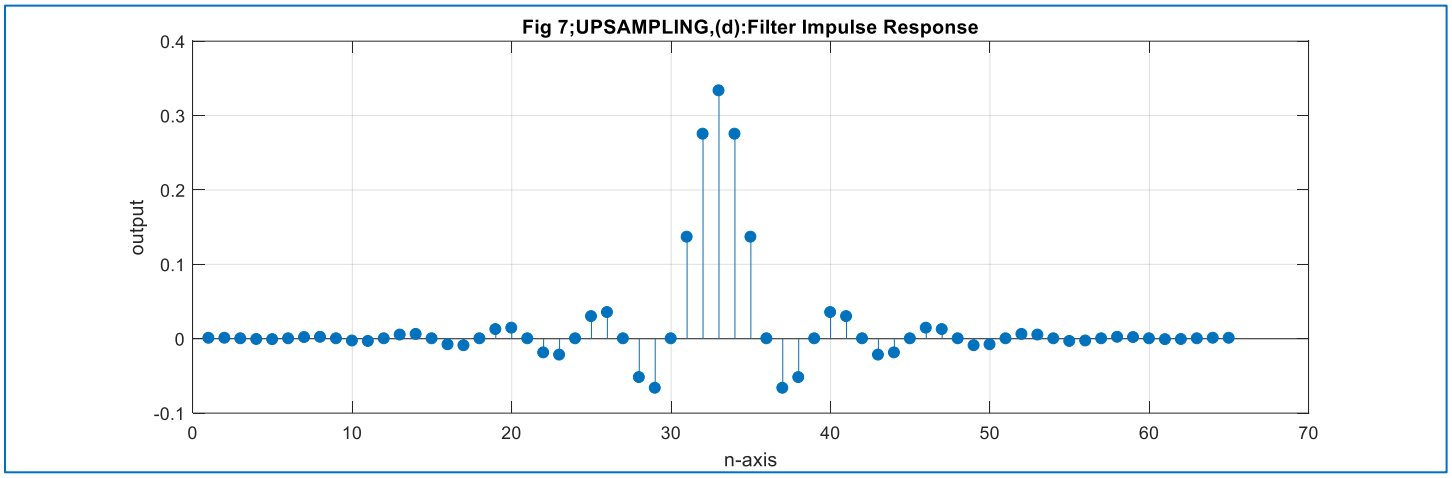


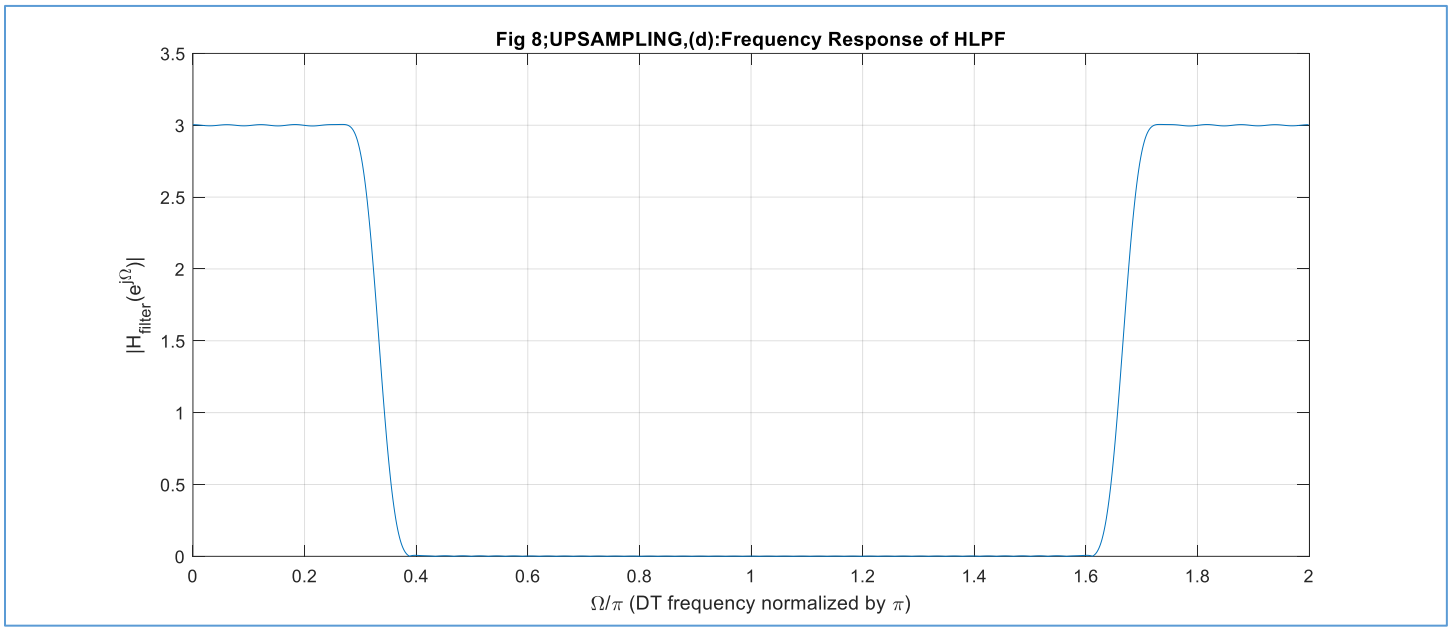


As shown above in our sketch, we expect the Fourier transforms of the signals to have a bandwidth of 0.26 for $x_1[n]$ and 0.13 for signal $x_2[n]$. This is confirmed from our MATLAB plots in *figure 5* and *figure 6.*

3. Upsampling

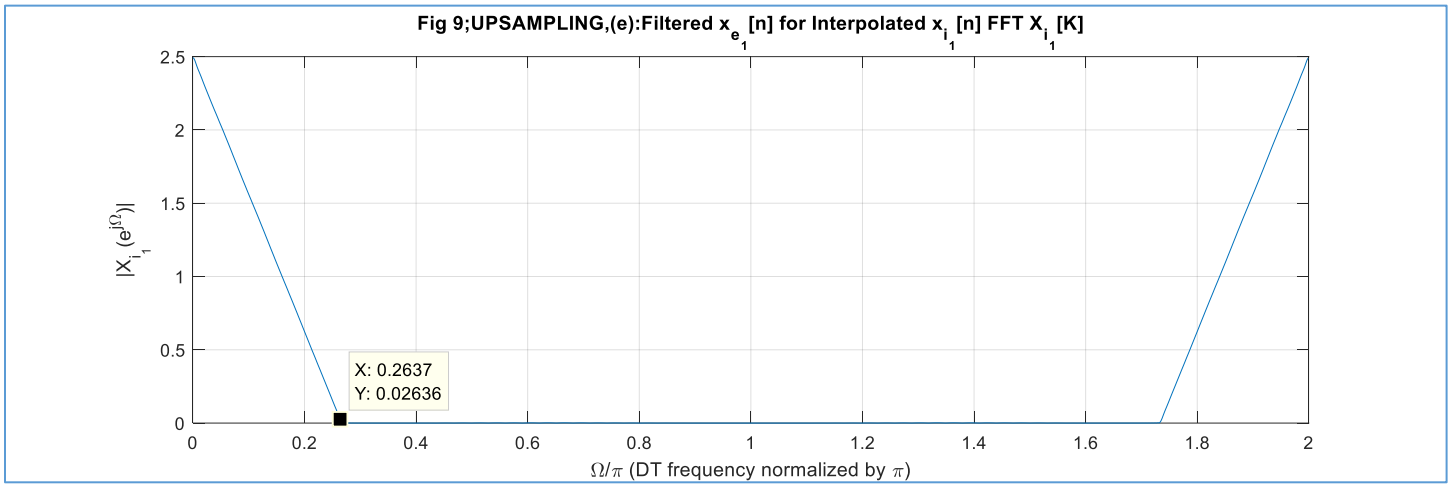
3d)

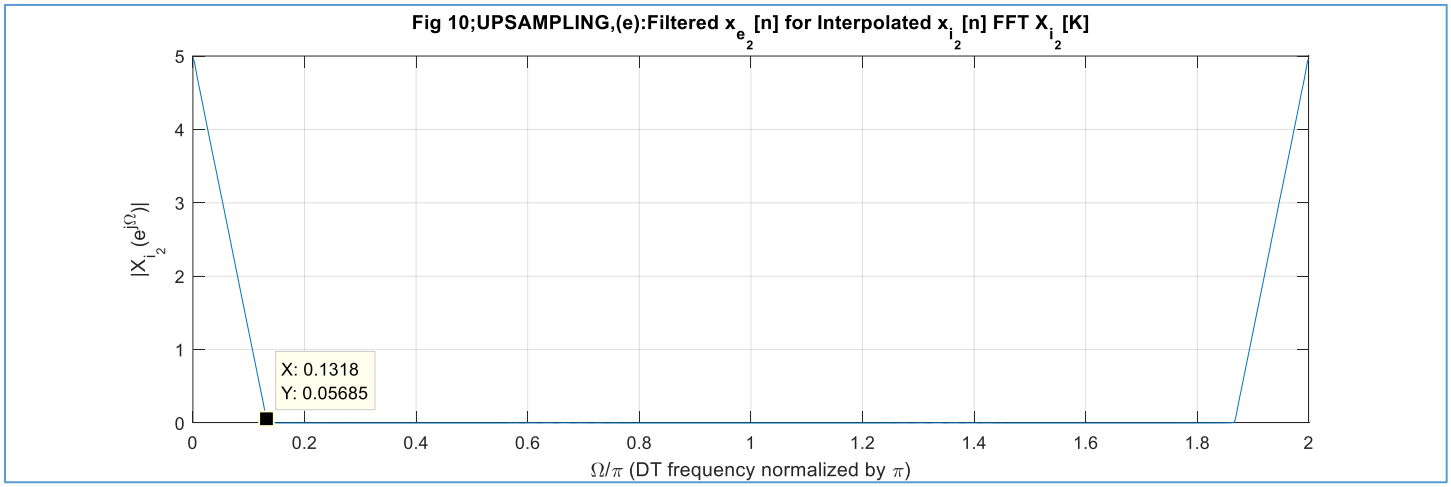




3. Upsampling

3e)



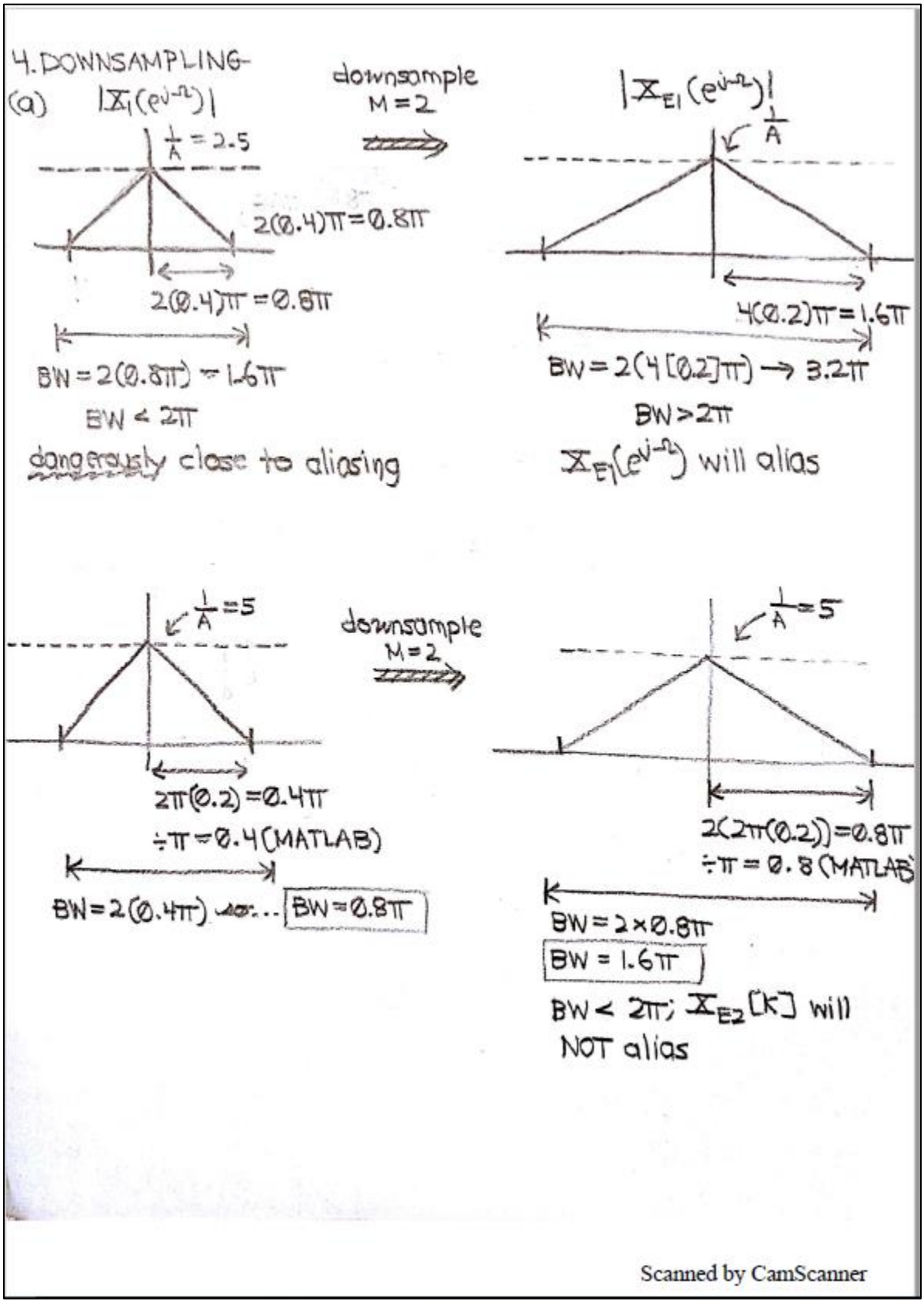


As we can see from the above figures, *figure 9* has bandwidth from 0 to 0.2637 compared to the value of 0 to 0.8 for *figure 3.* If we divide 0.8 by 3, we obtain 0.2666 repeating. This confirms that *figure 9* was sampled 3 times faster than the original plot in *figure 3.*

It's the same case for *figure 10.* From the previous plots, we know that the FFT of $x_2[n]$ in *figure 4* has a bandwidth from 0 to 0.4. When we take 0.4 and divide it by 3, we obtain 0.1333 repeating, which is what we obtained in *figure 10.* Thus, both the interpolated signals of $x_1[n]$ and $x_2[n]$ were plotted successfully.

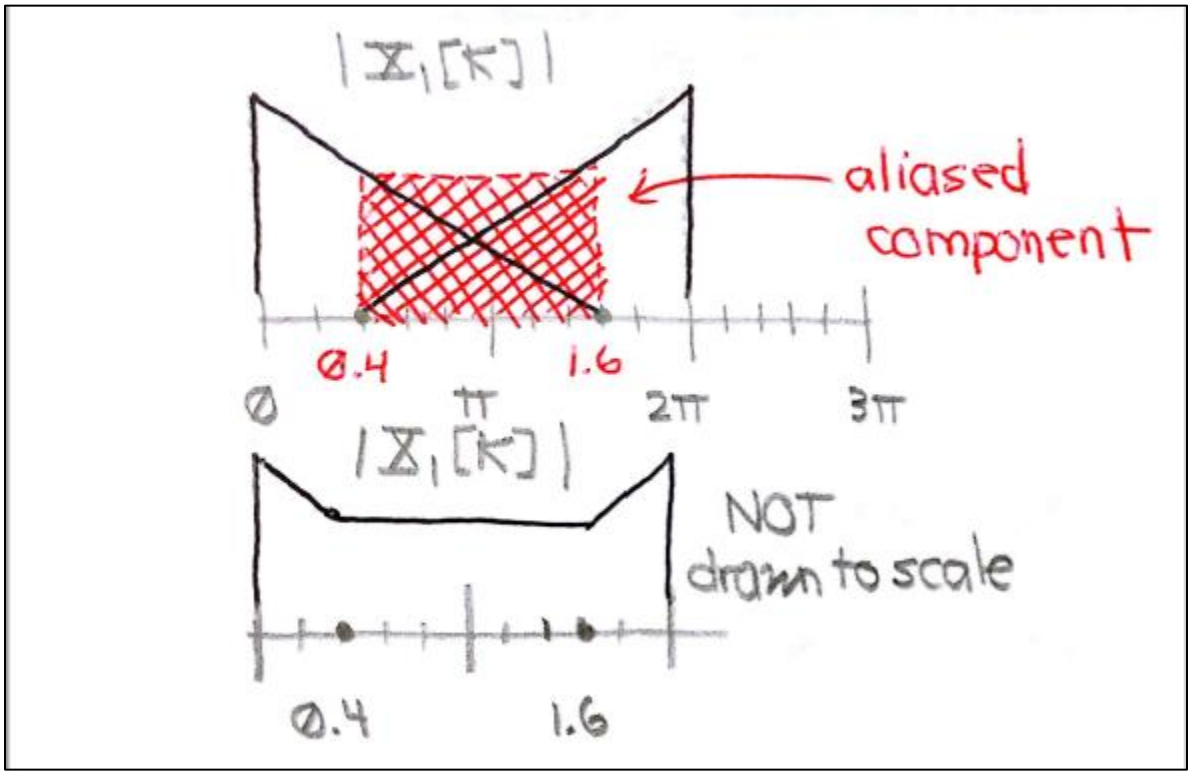
4. Downsampling

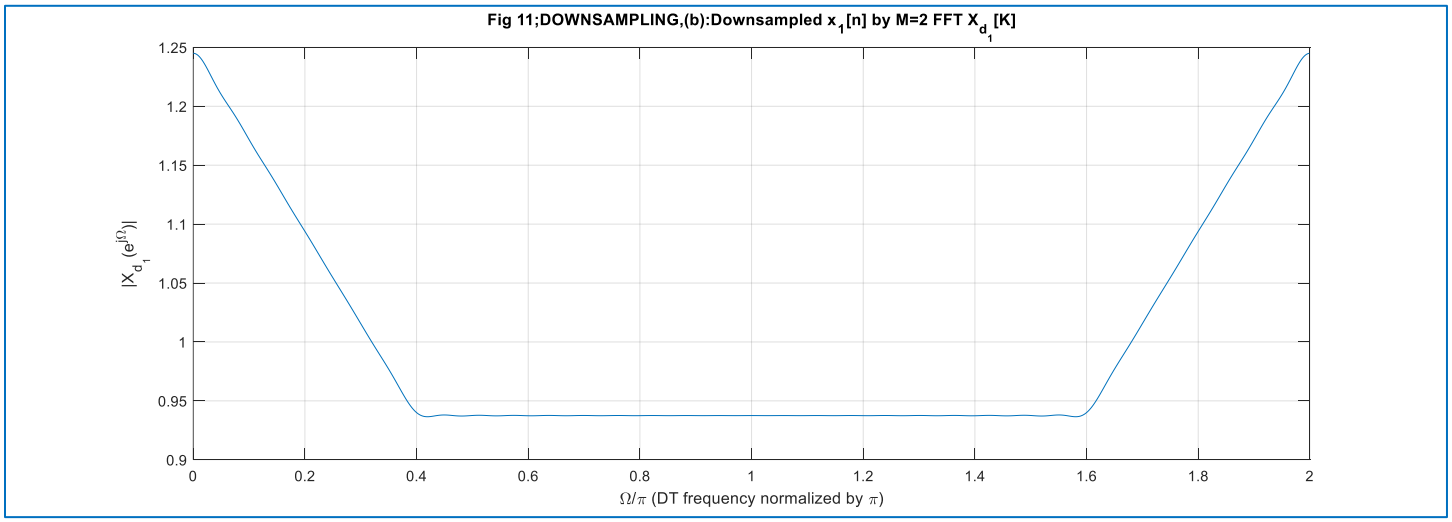
4a)



Downsampling

4b)

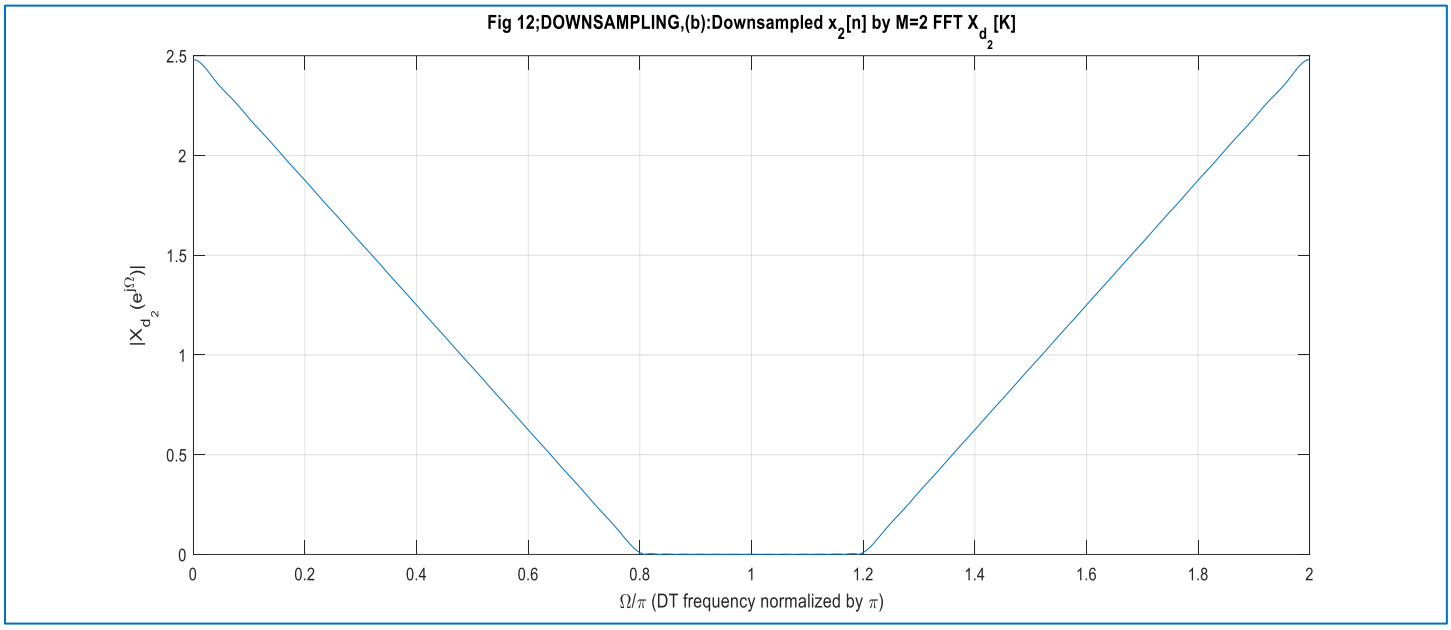




As shown in the analytical calculations above *figure 11,* we know that we will alias $X_1[K]$ between 0.4 and 0.6 since the bandwidth of the DTFT $X_1(e^{j\Omega})$ was shown to be 3.2π. Since the DFT of the DTFT is periodic with 2π, we will alias our signal since $X_1[K]$ is wider than 2π, so it will overlap with itself.

Downsampling

4b)



The DTFT in the analytical calculations had a bandwidth of 1.6π. This is smaller than 2π so the signal will not overlap with itself and thus will not alias, which is confirmed in our MATLAB plot in *figure 12.*

4c)

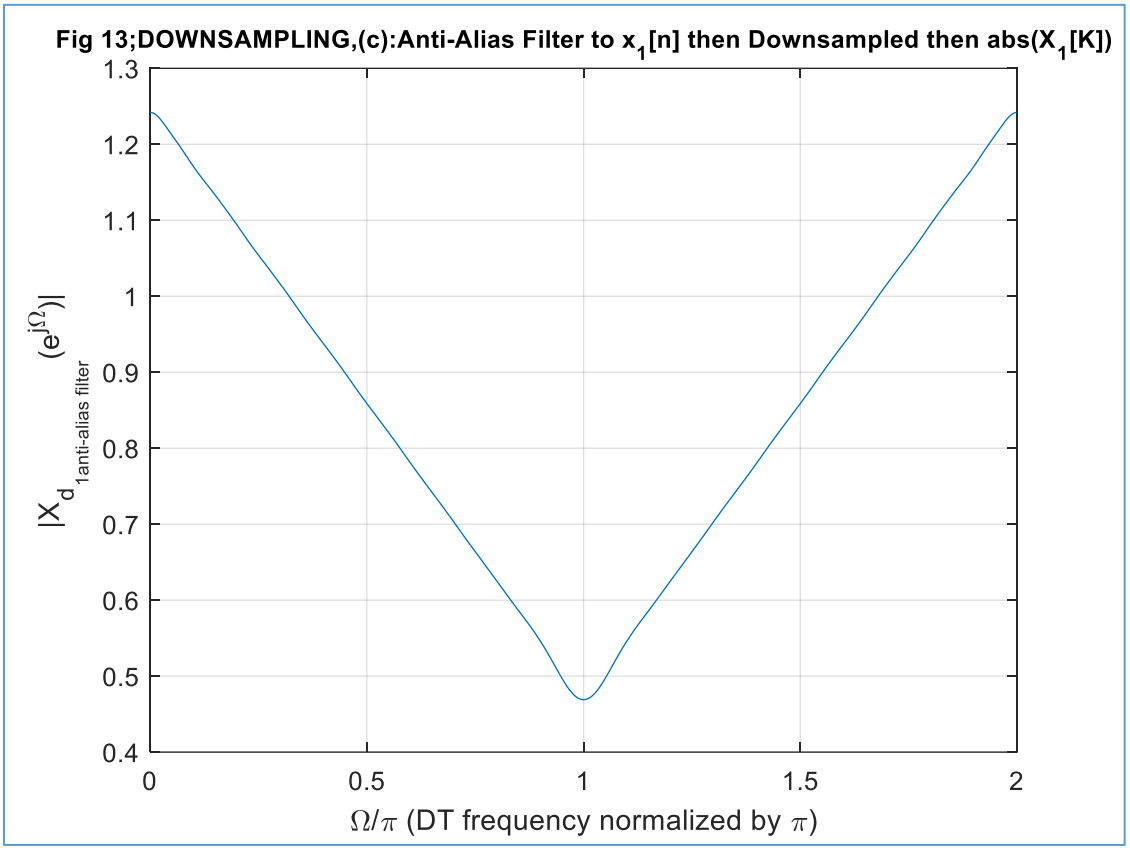


Fig 13;DOWNSAMPLING,(c):Anti-Alias Filter to x1[n] then Downsampled then abs(X1[K])
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0.4
0
0.5
1
1.5
2
|Xd1anti-alias filter(ejΩ)|
Ω/π (DT frequency normalized by π)

5. Changing the sample rate of an audio signal

5a)

If we upsample 11025 by a factor of 8, we end up with 88200. Then, if we downsample by a factor of 10, we end up with 8820. So, the ratio of upsampling to downsampling becomes

$$\frac{upsampling}{downsampling} = \frac{8}{10}$$

reducing this ratio down further can yield the following

$$\frac{upsampling}{downsampling} = \frac{4}{5}$$

which would yield the same results as upsampling 11025 by 8, then downsampling it by a factor of 10 to get to 8820. Both ratios are appropriate from getting from 11025 to 8820.

5b)

Yes, it matters whether you upsample or downsample first. This is because when you upsample, you're adding zeros between the points of signal $x_1[n]$ and increasing the sampling rate conversion. When you increase the sample rate, you compress your signal in the frequency domain. When you upsample, you don't run the risk of losing any part of your signal. All you're doing is expanding your signal in the time domain, and compressing it in the frequency domain. When you downsample, you decrease the sampling frequency and lose pieces of your signal $x[n]$. After decimation, there's no way to get back those truncated values from your original signal. Therefore, you must upsample first, then downsample. You upsample your signal by zero padding it do you don't lose any part of the original signal. Then, you downsample accordingly.

5c)

Since this is a combination of upsampling and downsampling, we are going to need two filters. The first is the upsampling filter, applied *after* we upsample. The second filter is the anit-aliasing filter, applied *before* we downsample. However, since the upsampling filter and the downsampling filter are used back to back, we know that the upsampling filter and the downsampling filter multiply together in time. This is because they are both low pass filters multiplying in frequency. When you multiply two pulses in the frequency domain, the bandwidth of the smaller pulse dominates. The result is a filter with the bandwidth from the smaller low pass filter with a gain that comes from multiplying the height of the first pulse from the second pulse.

5d)

Based on our discussion from 5c, we know that we need 2 filters for downsampling Homer's voice, since we are implementing upsampling and downsampling together to get from 11025 to 8820. However, we also discussed that this could be done with one filter only, since the upsampling filter and the downsampling filter multiply together regardless and have a filter with a cutoff frequency from the downsampling filter with a gain from both the upsampling filter and the downsampling filters multiplied together.

For this lab, I have chosen to use two filters. While this is not the most efficient design, I chose to do this because the idea of using two filters is my own idea. I was able to successfully implement the

downsampling of Homer's voice using two filters on my own. The idea of using one filter was discovered through the collaboration of multiple students during professor Wage's office hours. While I cannot take credit for discovering that you could downsample Homer's voice with one filter, I will acknowledge that it is worth discussing in the report.

My x8k signal sounds identical to Homer's voice. When I play both the audio signals simultaneously, I discover that x11k plays slightly faster than x8k. This is to be expected, since we are playing x11k at a rate of 11025Hz, and x8k at a rate of 8820Hz.

Conclusion

Our goal for this project was to understand the concepts of upsampling and downsampling. We accomplished this throughout this lab report – from the analytical calculations, to the MATLAB deliverables verifying our intuition from our analytical calculations, the questions the project proposed throughout the lab, and finally downsampling Homer's voice from x11k played at 11025Hz to x8k played at 8820Hz.

When we increase the sampling frequency f_s, we expand the signal in the time domain and compress the signal in the frequency domain. When we decrease the sampling rate, we compress in the time domain and expand in the frequency domain. The time domain and the frequency domain have an inverse relationship with one another. One way to think about the relationship between the sampling frequency and the frequency domain is looking at the DFT with the magnifying glass. If we increase the sampling rate, we're zooming out in the frequency domain; thus, we get more copies of our spectrum in the frequency domain. When we decrease the sampling frequency, we zoom in to the frequency domain and expand our signal.

For Homer's voice, we are going from a sampling rate of 11025 to 8820. This requires a combination of upsampling and downsampling as we cannot multiply 11025 by any integer to get to 8820. We need to multiply by a ratio of integers to get from 11025 to 8820. This requires to upsample 11025 by 4, and then downsample by 5. The ratio $\frac{4}{5}$ can be modified to any integer value, if they retain this rate. For the MATLAB portion of the code, I used a ratio of $\frac{8}{10}$ simply because this was the first ratio I discovered when I realized what I needed to do. I have not gone back to change the ratio.

For Homer's voice, x11k has certain pops and cracks to it. When the downsampled signal x8k was played, these pops and cracks disappeared. However, the overall quality of Homer's voice remains reasonably the same to the point where it's difficult to distinguish the two. When the two signals are played together, x11k is slightly faster than x8k. This is to be expected since we are playing x8k at a slower rate of 8820Hz versus 11025Hz for x11k.

If you were to take x11k and play it with soundsc(x11k,12000), you're playing Homer's voice a rate faster than it was sampled, so his voice will speed up. If you play x11k below 11025, you'll slow his voice down. Our objective this entire project was to downsample the original x11k sound file, downsample it and play it at a slower rate while keeping the quality of Homer's voice the same. As expressed in the research portion of the report, downsampling is extensively used to reduce the size of digital files as much as possible so they occupy less space. For a database of fingerprints for a biometric security system, we want to downsample retinal scans and finger prints so they occupy less memory. When biometric data occupies less memory, it can also be read faster since it's a smaller file from its regularly sized file. The same principle does not exactly apply here for Homer's voice, but thanks to our knowledge of upsampling and downsampling, Homer's voice now occupies less space than before while retaining the same quality as it did before for x11k.

```
%--------------------------------------------------------------------------
%//////////////////////////////////////////////////////////////////////////
%_____________________________PART 3 - UPSAMPLING_____________________________
%//////////////////////////////////////////////////////////////////////////
%--------------------------------------------------------------------------
%PART (a)
n = 0:127;
x_1 = (sinc(0.4.*(n-64))).^2;

figure(1);
stem(n,x_1,'filled');
grid;
title('Fig 1;UPSAMPLING,(a):Plot of x_{1}[n]');
xlabel('n-axis');
ylabel('output');
xlim([54 74]);

x_2 = (sinc(0.2*(n-64))).^2;
figure(2);
stem(n,x_2,'filled');
grid;
title('Fig 2;UPSAMPLING,(a):Plot of x_{2}[n]');
xlabel('n-axis');
ylabel('output');
xlim([44 84]);

%--------------------------------------------------------------------------
%//////////////////////////////////////////////////////////////////////////
%--------------------------------------------------------------------------
%PART (b)
k = 0:2047;
wk = 2*pi*k/2048;

X_1_K = fft(x_1,2048);
figure(3);
plot(wk/pi,abs(X_1_K));
grid;
title('Fig 3;UPSAMPLING,(b):FFT of x_{1}[n] for X_{1}[K]');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{1}(e^{j\Omega})|');

X_2_K = fft(x_2,2048);
figure(4);
plot(wk/pi,abs(X_2_K));
title('Fig 4;UPSAMPLING,(b)FFT of x_{2}[n] for X_{2}[K]');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{2}(e^{j\Omega})|');
grid;

%--------------------------------------------------------------------------
%//////////////////////////////////////////////////////////////////////////
%--------------------------------------------------------------------------
%PART (c)
L=3;
xe_1=zeros(1,L*length(x_1));
xe_1(1:L:length(xe_1))=x_1;

XE_1_K=fft(xe_1,2048);
figure(5);
plot(wk/pi,abs(XE_1_K));
grid;
title('Fig 5;UPSAMPLING,(c)Upsampled X_{1}[K], X_{E_{1}}[K]');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{E_1}(e^{j\Omega})|');
```

```
xe_2=zeros(1,L*length(x_2));
xe_2(1:L:length(xe_2))=x_2;

XE_2_K=fft(xe_2,2048);
figure(6);
plot(wk/pi,abs(XE_2_K));
grid;
title('Fig 6;UPSAMPLING,(c):Upsampled X_{2}[k], X_{E_{2}}[K]');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{E_2}(e^{j\Omega})|');

%------------------------------------------------------------------------
%////////////////////////////////////////////////////////////////////////
%------------------------------------------------------------------------
%PART (d)
%FILTER CREATION FOR UPSAMPLING
alpha=(1/3);
P=65;
hlpf=3*alpha*sinc(alpha*(-(P-1)/2:(P-1)/2)).*(hamming(P)');


%filter impulse response for upsampling
hlpf_impulse = impz(hlpf,1);
figure(7);
stem(hlpf_impulse,'filled','o');
grid;
title('Fig 7;UPSAMPLING,(d):Filter Impulse Response');
xlabel('n-axis');
ylabel('output');

%filter frequency response for upsampling
figure(8);
HLPF_IMPULSE_K=fft(hlpf_impulse,2048);
plot(wk/pi,abs(HLPF_IMPULSE_K));
title('Fig 8;UPSAMPLING,(d):Frequency Response of HLPF');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|H_{filter}(e^{j\Omega})|');
grid;

%------------------------------------------------------------------------
%////////////////////////////////////////////////////////////////////////
%------------------------------------------------------------------------
%PART (e)
%upsample, then apply the upsampling filter for anti-aliasing
xi_1 = filter(hlpf,1,xe_1);
XI_1_K = fft(xi_1,2048);
figure(9);
plot(wk/pi,abs(XI_1_K));
grid;
title('Fig 9;UPSAMPLING,(e):Filtered x_{e_{1}}[n] for Interpolated x_{i_{1}}[n] FFT X_{i_1}[K]');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{i_1}(e^{j\Omega})|');

%don't need to do this part, but it's ok
xi_2 = filter(hlpf,1,xe_2);
XI_2_K=fft(xi_2,2048);
figure(10);
plot(wk/pi,abs(XI_2_K));
grid;
title('Fig 10;UPSAMPLING,(e):Filtered x_{e_{2}}[n] for Interpolated x_{i_{2}}[n] FFT X_{i_2}[K]');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{i_2}(e^{j\Omega})|');
%------------------------------------------------------------------------
%////////////////////////////////////////////////////////////////////////
%------------------------------------------------------------------------

%------------------------------------------------------------------------
```

```
%////////////////////////////////////////////////////////////////////////
%_____________________________PART 4 - DOWNSAMPLING_____________________________
%////////////////////////////////////////////////////////////////////////
%------------------------------------------------------------------------
%PART(a)

%refer to analytical calculations

%------------------------------------------------------------------------
%////////////////////////////////////////////////////////////////////////
%------------------------------------------------------------------------
%PART(b)
M=2;
xd_1=x_1(1:M:length(x_1));
xd_2=x_2(1:M:length(x_2));

XD_1_K=fft(xd_1,2048);
figure(11);
plot(wk/pi,abs(XD_1_K));
grid;
title('Fig 11;DOWNSAMPLING,(b):Downsampled x_{1}[n] by M=2 FFT X_{d_1}[K]');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{d_1}(e^{j\Omega})|');

XD_2_K=fft(xd_2,2048);
figure(12);
plot(wk/pi,abs(XD_2_K));
grid;
title('Fig 12;DOWNSAMPLING,(b):Downsampled x_{2}[n] by M=2 FFT X_{d_2}[K]');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{d_2}(e^{j\Omega})|');
%------------------------------------------------------------------------
%////////////////////////////////////////////////////////////////////////
%------------------------------------------------------------------------
%PART(c)
alpha_2=(1/2);
P=65;
hlpf_2=alpha_2*sinc(alpha_2*(-(P-1)/2:(P-1)/2)).*(hamming(P)');

fx_1=filter(hlpf_2,1,x_1);
dfx_1=fx_1(1:M:length(fx_1));
DFX_1_K=fft(dfx_1,2048);
figure(13);
plot(wk/pi,abs(DFX_1_K));
grid;
title('Fig 13;DOWNSAMPLING,(c):Anti-Alias Filter to x_1[n] then Downsampled then
abs(X_1[K])','FontSize',9.5);
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|X_{d_1_{anti-alias filter}}(e^{j\Omega})|');

%------------------------------------------------------------------------
%////////////////////////////////////////////////////////////////////////
%------------------------------------------------------------------------

%------------------------------------------------------------------------
%////////////////////////////////////////////////////////////////////////
%_________PART 5 - CHANGING THE SAMPLE RATE OF AN AUDIO SIGNAL_____________
%////////////////////////////////////////////////////////////////////////
%------------------------------------------------------------------------
%PART(a)

%take 11025 and multiply it by upsample it by 8
%then take that, and downsample by 10
%the final result will take you from 11025 to 8820

%------------------------------------------------------------------------
%////////////////////////////////////////////////////////////////////////
```

```
%-------------------------------------------------------------------------
%PART(b)

%{
        No, the order doesn't matter. It doesn't matter if you upsample or
    downsample first. Taking 11025 and upsampling by 8, then downsampling
    by 10 produces the same result as downsampling by 10, then upsampling
    by 8.
 %}

%-------------------------------------------------------------------------
%/////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------------------------
%PART(c)

%-------------------------------------------------------------------------
%/////////////////////////////////////////////////////////////////////////
%-------------------------------------------------------------------------
%PART(d)
%UPSAMPLINLG FILTER; APPLIED AFTER UPSAMPLING
alpha_up=(1/8);
P_up=65;
hlpfu=alpha_up*sinc(alpha_up*(-(P-1)/2:(P-1)/2)).*(hamming(P)');

%IMPULSE RESPONSE
hlpfu_impulse=impz(hlpfu,1);
figure(14);
stem(hlpfu_impulse,'filled','o');
title('Fig 14;AUDIO,(d):Impulse Response of HLPF_{upsampling}');
xlabel('n-axis');
ylabel('output');
grid;

%FREQUENCY RESPONSE
HLPFU_K=fft(hlpfu_impulse,2048);
figure(15);
plot(wk/pi,abs(HLPFU_K));
title('Fig 15;AUDIO,(d):Frequency Response of HLPF_{upsample}');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|H_{upsampling}(e^{j\Omega})|');
grid;

%DOWNSAMPLING FILTER; APPLIED BEFORE DOWNSAMPLING
alpha_down=(1/10);
P_down=65;
hlpfd=alpha_down*sinc(alpha_down*(-(P-1)/2:(P-1)/2)).*(hamming(P)');

%IMPULSE RESPONSE
hlpfd_impulse=impz(hlpfd,1);
figure(16);
stem(hlpf_impulse,'filled','o');
grid;
title('Fig 16;AUDIO,(d):Impulse Response of HLPF_{downsampling}');
xlabel('n-axis');
ylabel('output');

%FREQUENCY RESPONSE
HLPFD_K=fft(hlpfd_impulse,2048);
figure(17);
plot(wk/pi,abs(HLPFD_K));
title('AUDIO,(d):Frequency Response of HLPF_{downsampling}');
xlabel('\Omega/\pi (DT frequency normalized by \pi)');
ylabel('|H_{downsampling}(e^{j\Omega})|');
grid;

%UPSAMPLING & DOWNSAMPLING HOMER'S VOICE
%UPSAMPLING: UPSAMPLE, THEN FILTER
```

```
L_homer=8;
M_homer=10;

x11ke=zeros(1,L_homer*length(x11k));
x11ke(1:L_homer:length(x11ke))=x11k;

x11ki=filter(hlpfu,1,x11ke);

%DOWNSAMPLING: FILTER, THEN DOWNSAMPLE
x11k_filter_before_downsample=filter(hlpfd,1,x11ki);

x11kd=x11k_filter_before_downsample(1:M_homer:length(x11k_filter_before_downsample));
x8k=x11kd;

%--------------------------------------------------------------------------
```

```
%UPSAMPLINLG FILTER; APPLIED AFTER UPSAMPLING
alpha_up=(1/8);
P=65;
hlpfu=3*alpha_up*sinc(alpha_up*(-(P-1)/2:(P-1)/2)).*(hamming(P)');

%DOWNSAMPLING FILTER; APPLIED BEFORE DOWNSAMPLING
alpha_down=(1/10);
P=65;
hlpfd=alpha_down*sinc(alpha_down*(-(P-1)/2:(P-1)/2)).*(hamming(P)');

%UPSAMPLING & DOWNSAMPLING HOMER'S VOICE
%UPSAMPLING: UPSAMPLE, THEN FILTER
L_homer=8;
M_homer=10;

x11ke=zeros(1,L_homer*length(x11k));
x11ke(1:L_homer:length(x11ke))=x11k;

%UPSAMPLE FILTER AFTER UPSAMPLING
x11ki=filter(hlpfu,1,x11ke);
%x11ki=x11ke;

%DOWNSAMPLING: FILTER, THEN DOWNSAMPLE
x11k_filter_before_downsample=filter(hlpfd,1,x11ki);

x11kd=x11k_filter_before_downsample(1:M_homer:length(x11k_filter_before_downsample));
x8k=x11kd;
```