EEE 314: Digital Signal Processing I Lab

Experiment – 1

Study of Sampling & Quantization

Theory:

Most signals of practical interest, such as speech, biological signals, communication signals etc. are analog. These signals must be processed for different purposes. Digital signal processing of an analog signal is preferable to processing the signals directly in the analog domain because of its flexibility in reconfiguration, better accuracy in control, better storing capability and cost effectiveness. That's why the analog signals are to be converted into corresponding digital domain for the purpose of processing. The analog signals are converted into digital signals through sampling, quantization and encoding.

Lab Work:

Part A

Plotting an Analog Signal

```
F=10;
t=0:0.001:0.4;
x=cos(2*pi*F*t);
plot(t,x),title('Analog');
```

Output:

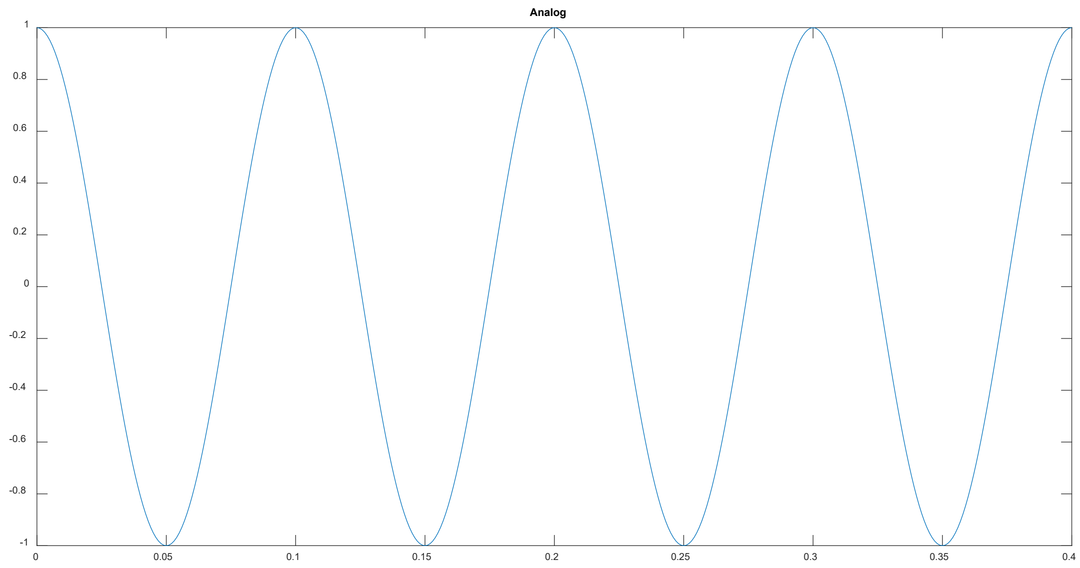


Figure 1 for Part A

Part B

Sampling an Analog Signal

```
F=10;
t=0:0.001:0.4;
x=cos(2*pi*F*t);
subplot(2,1,1),plot(t,x),title('Analog');
Fs=100;
n=0:1/Fs:0.4;
xs=cos(2*pi*F*n);
subplot(2,1,2),stem(n,xs),title('Sampled');
```

Output:

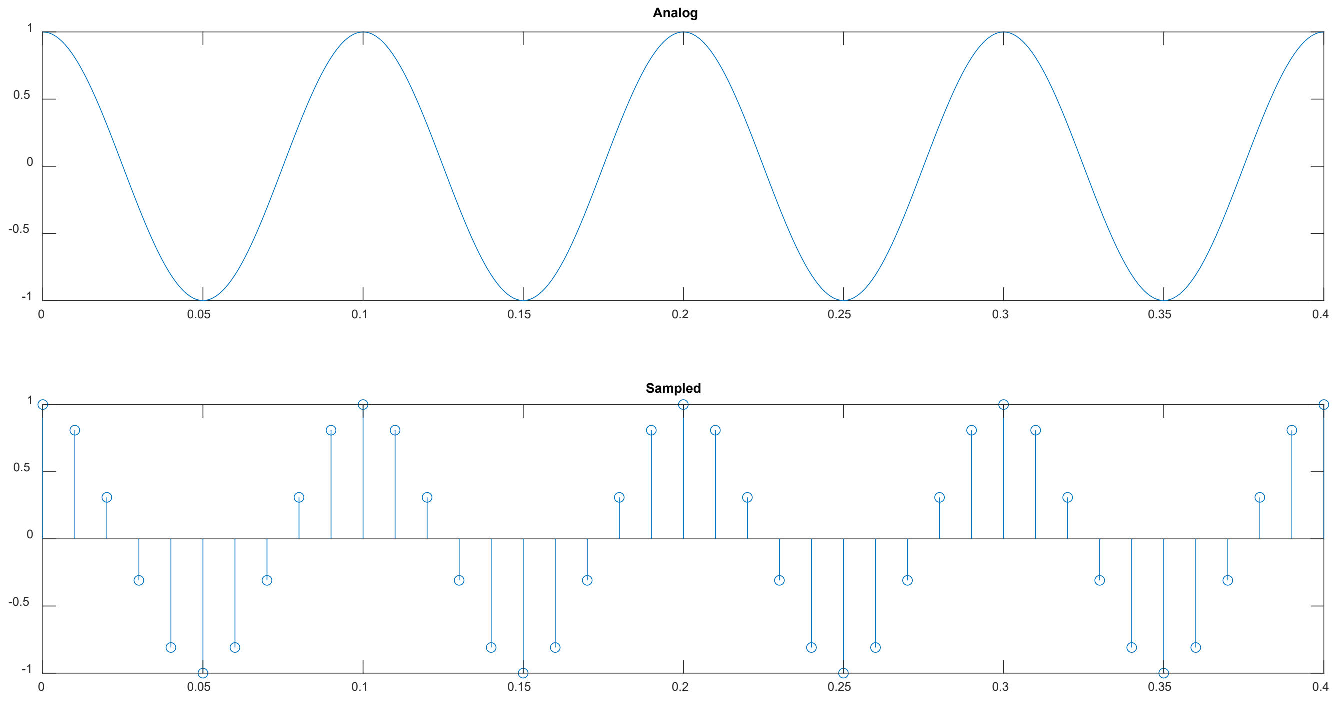


Figure 2 for Part B

Part C

Reconstructing the Signal from Samples

```
F=10;
t=0:0.001:0.4;
x=cos(2*pi*F*t);
subplot(3,1,1),plot(t,x),title('Analog');
Fs=60;
n=0:1/Fs:0.4;
xs=cos(2*pi*F*n);
subplot(3,1,2),stem(n,xs),title('Sampled');
subplot(3,1,3),plot(n,xs),title('Reconstructed from Samples');
```

Output:

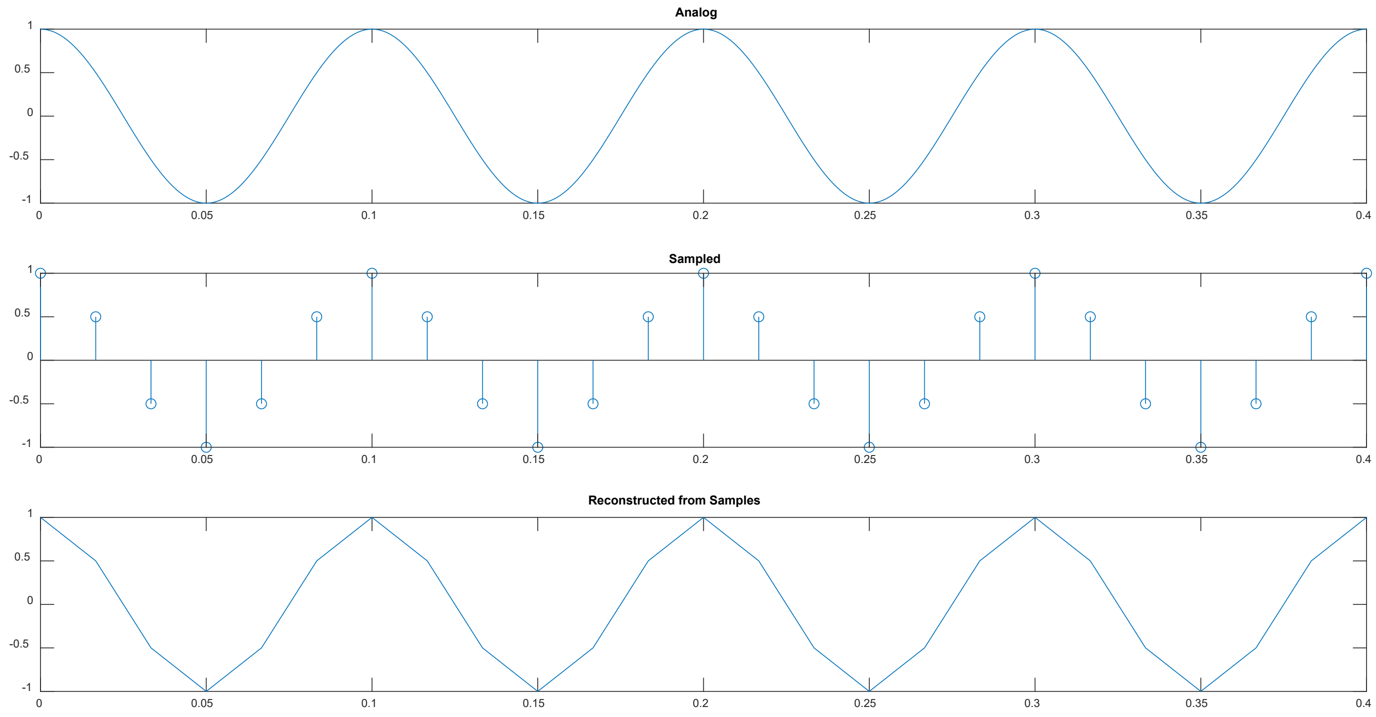


Figure 3 for Part C

Part D

Uniform Quantization of Sampled Signal

```
F1=10;
F2=20;
t=0:0.001:0.4;
x=cos(2*pi*F1*t)+cos(2*pi*F2*t);
subplot(5,1,1),plot(t,x),title('Analog');
Fs=180;
n=0:1/Fs:0.4;
xs=cos(2*pi*F1*n)+cos(2*pi*F2*n);
subplot(5,1,2),stem(n,xs),title('Sampled');
subplot(5,1,3),plot(n,xs),title('Reconstructed from Samples');
b=5;
L=2^b;
delta=(max(xs)-min(xs))/(L-1);
level=min(xs):delta:max(xs);
xq=xs;
for i=1:length(xs)
    for j=1:L
        if (xs(i)>level(j) && xs(i)<level(j+1))
            gap1=xs(i)-level(j);
            gap2=level(j+1)-xs(i);
            if (gap1>gap2)
                xq(i)=level(j+1);
            else
                xq(i)=level(j);
            end
        end
    end
end
subplot(5,1,4),stairs(n,xq),title('Quantized');
subplot(5,1,5),plot(n,xq),title('Reconstructed from Quant Samples');
```

[*output on next page*]

Output:

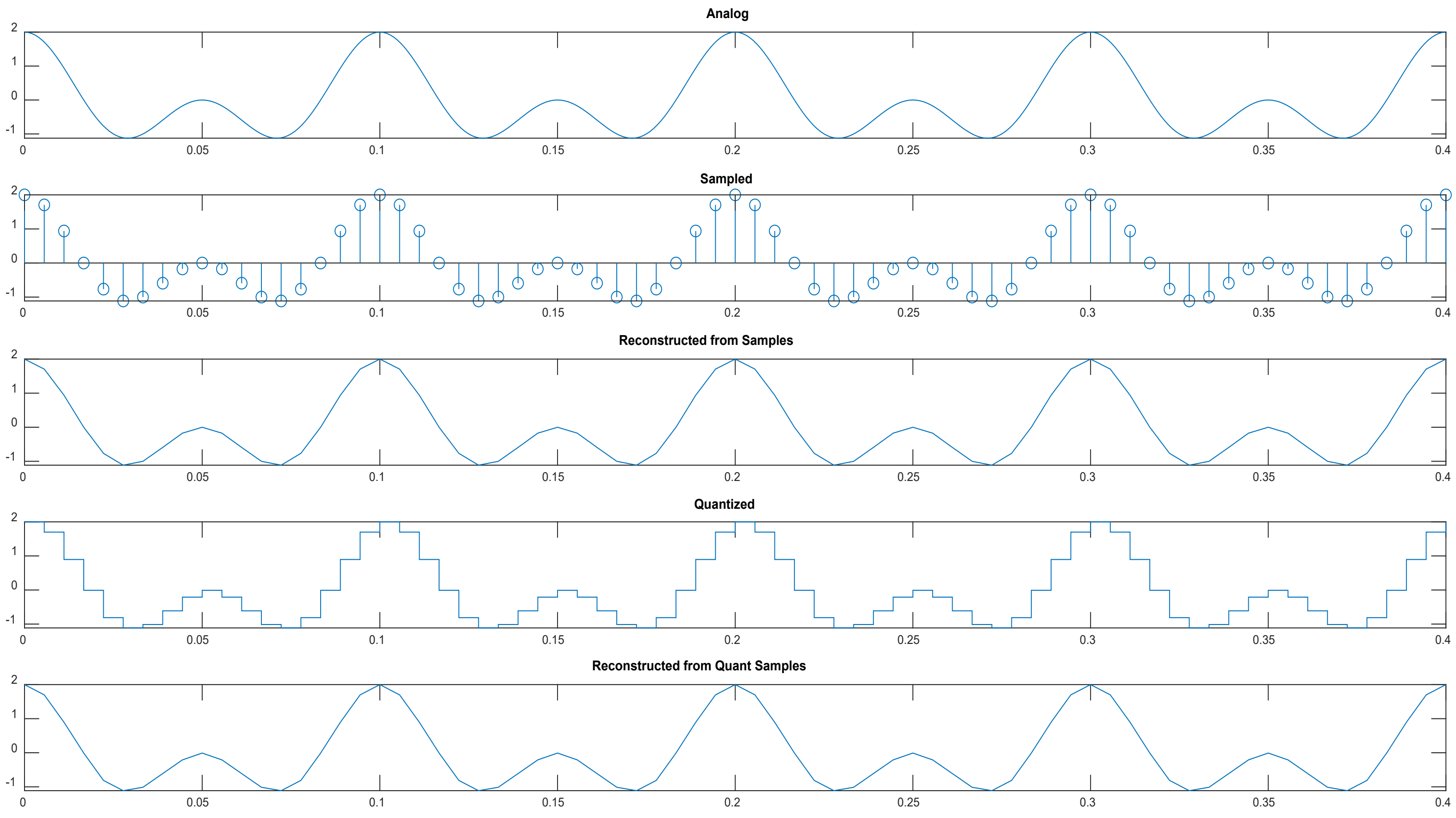


Figure 4 for Part D

Part E

Measuring SQNR (dB) for the Quantization

```
F1=10;
F2=20;
t=0:0.001:0.4;
x=cos(2*pi*F1*t)+cos(2*pi*F2*t);
Fs=180;
n=0:1/Fs:0.4;
xs=cos(2*pi*F1*n)+cos(2*pi*F2*n);
b=6;
L=2^b;
delta=(max(xs)-min(xs))/(L-1);
level=min(xs):delta:max(xs);
xq=xs;
for i=1:length(xs)
    for j=1:L
        if (xs(i)>level(j) && xs(i)<level(j+1))
            gap1=xs(i)-level(j);
            gap2=level(j+1)-xs(i);
            if (gap1>gap2)
                xq(i)=level(j+1);
            else
                xq(i)=level(j);
            end
        end
    end
end
e=xq-xs;
Pe=sum(e.^2)/length(e);
Px=sum(x.^2)/length(x);
SQNR=Px/Pe;
SQNRdB=10*log10(SQNR)
```

Output from command window:

```
SQNRdB = 36.5146
```

Part F

Miscellaneous (1)

```
x=[1.2548, 2.9542, -0.7785, -1.4538, 4.5592, 0.5466];
round(x)
round(x,1)
round(x,2)
round(x,3)
fix(x)
```

Output from command window:

```
1         3         -1        -1        5         1

1.3000    3.0000    -0.8000   -1.5000   4.6000    0.5000

1.2500    2.9500    -0.7800   -1.4500   4.5600    0.5500

1.2550    2.9540    -0.7790   -1.4540   4.5590    0.5470

1         2         0         -1        4         0
```

Miscellaneous (2)

```
x=[1.2, -3.7, 2.8, -0.5, 1.5, -1.1];
ceil(x)
floor(x)
round(x)
fix(x)
```

Output from command window:

```
2         -3        3         0         2         -1

1         -4        2         -1        1         -2

1         -4        3         -1        2         -1

1         -3        2         0         1         -1
```

Discussion:

You can search any function on MATLAB website: https://www.mathworks.com/help/

Documentation link to some important functions are given below:

interp1()	https://www.mathworks.com/help/matlab/ref/interp1.html
stem()	https://www.mathworks.com/help/matlab/ref/stem.html
length()	https://www.mathworks.com/help/matlab/ref/length.html
max()	https://www.mathworks.com/help/matlab/ref/max.html
min()	https://www.mathworks.com/help/matlab/ref/min.html
round()	https://www.mathworks.com/help/matlab/ref/round.html
fix()	https://www.mathworks.com/help/matlab/ref/fix.html
ceil()	https://www.mathworks.com/help/matlab/ref/ceil.html
floor()	https://www.mathworks.com/help/matlab/ref/floor.html

- In theory section, define Nyquist Sampling Theorem & Quantization (with appropriate examples).
- Comment after each output. Discuss why the output is the way it is (in each part). Also discuss how the output can be improved. Lastly, write one sentence for each of the nine functions mentioned above.

EEE 314: Digital Signal Processing I Lab

Experiment – 2

Time Domain Analysis of DT Signals & Systems

Theory:

Discrete-time (DT) signals are represented mathematically as sequences of numbers. DT signals are defined at discrete times, and thus, the independent variable has discrete values. A DT system is defined mathematically as a transformation or operator that maps an input sequence with values x[n] into an output sequence with values y[n]. In this experiment, we will perform different DT sequence generation, their delaying & advancing. We will also perform addition of two DT sequences, up-sampling & down-sampling, even & odd part generation, and convolution. Finally, we will perform correlation of DT sequences.

In discrete time, the unit impulse is simply a sequence that is zero except at n = 0, where it is unity. In discrete time, the unit step is a well-defined sequence, being zero for n < 0, and unity for all other values of n. The unit ramp is a sequence that is zero for n < 0, and n for all other values of n.

The process of converting the sampling rate of a digital signal from one rate to another is sampling rate conversion. Increasing the rate of already sampled signal is up-sampling, whereas decreasing the rate is called down-sampling. To up-sample by an integer factor N, we simply insert N–1 zeros between x[n] and x[n+1] for all n. To down-sample, we select every Nth sample and discard the rest.

Convolution is a mathematical operation, just as multiplication, addition or integration. Addition takes two numbers and produces a third number, while convolution takes two signals and produces a third signal. In linear systems, convolution is used to describe the relationship between three signals of interest: input signal, impulse response, and output signal. If we know a system's impulse response, then we can calculate what the output will be for any possible input signal. This means we know everything about the system.

Correlation is a mathematical operation that is very similar to convolution. Just as with convolution, correlation uses two signals to produce a third signal. This third signal is called the cross-correlation of the two input signals. If a signal is correlated with itself, the resulting signal is instead called the autocorrelation.

Lab Work:

Part A

A.1: Unit Impulse

```
n=-10:10;
impulse=((n-0)==0);
subplot(3,1,1),stem(n,impulse),title('Unit Impulse');
impulse=((n+2)==0);
subplot(3,1,2),stem(n,impulse),title('Advanced Unit Impulse');
impulse=((n-3)==0);
subplot(3,1,3),stem(n,impulse),title('Delayed Unit Impulse');
```

Output:

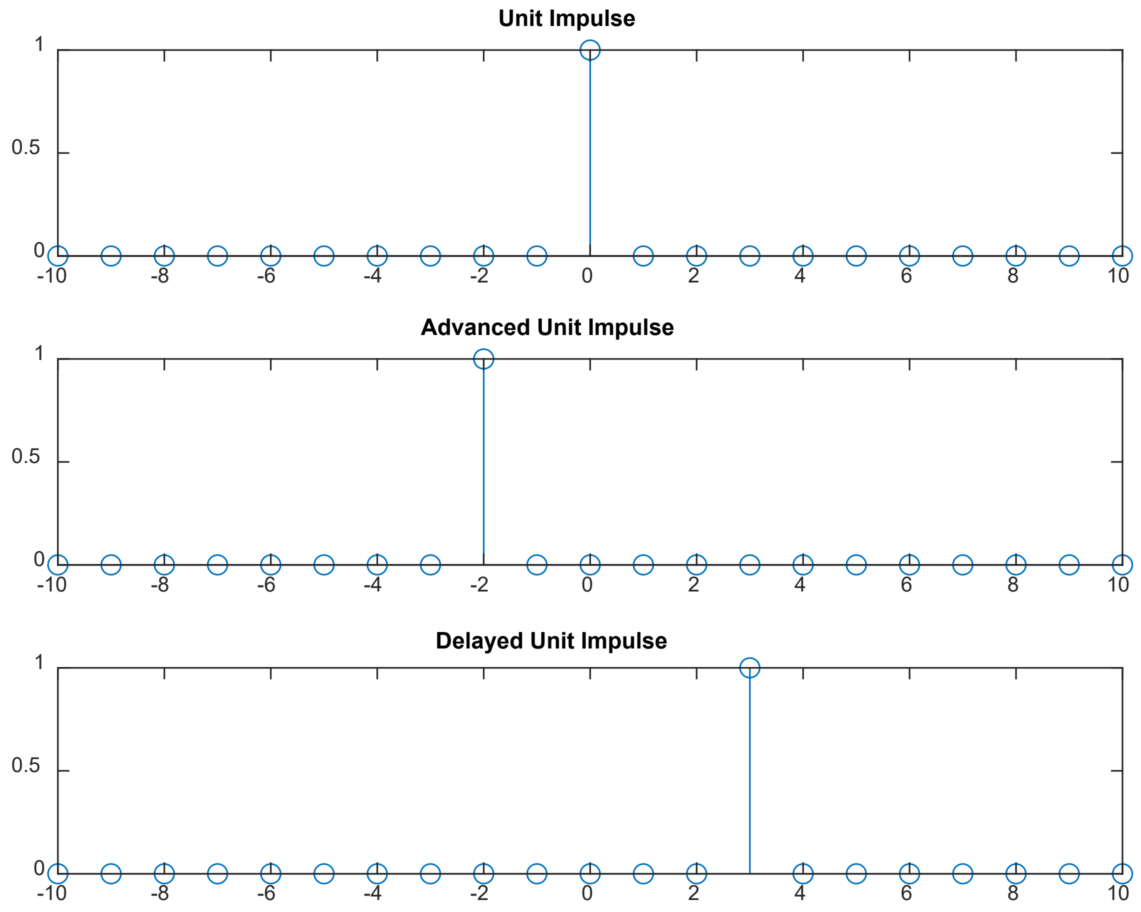


Figure 1 for Part A.1

A.2: Unit Step

```
n=-10:10;
step=((n-0)>=0);
subplot(3,1,1),stem(n,step),title('Unit Step');
step=((n+3)>=0);
subplot(3,1,2),stem(n,step),title('Advanced Unit Step');
step=((n-2)>=0);
subplot(3,1,3),stem(n,step),title('Delayed Unit Step');
```

Output:

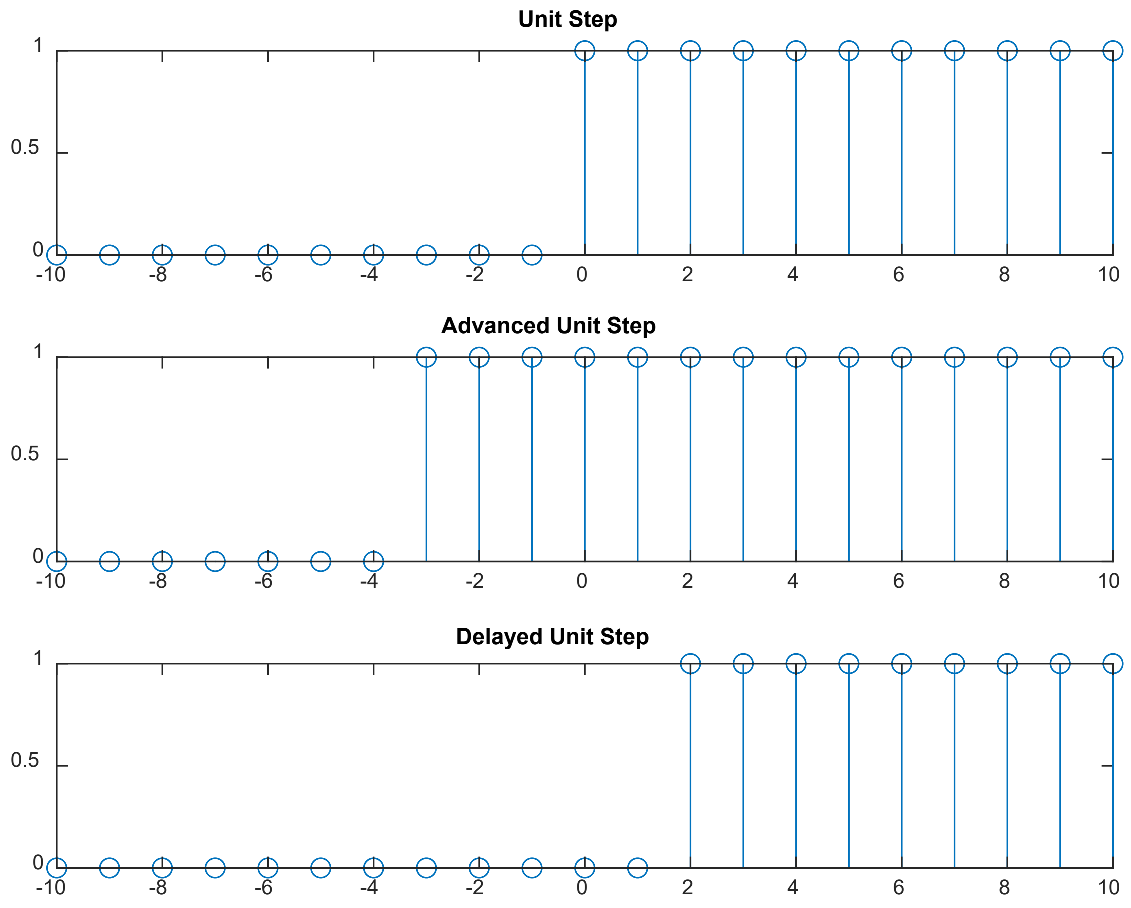


Figure 2 for Part A.2

A.3: Unit Ramp

```
n=-10:10;
step=((n-0)>=0);
ramp=(n-0).*step;
subplot(3,1,1),stem(n,ramp),title('Ramp');
step=((n+4)>=0);
ramp=(n+4).*step;
subplot(3,1,2),stem(n,ramp),title('Advanced Ramp');
step=((n-3)>=0);
ramp=(n-3).*step;
subplot(3,1,3),stem(n,ramp),title('Delayed Ramp');
```

Output:

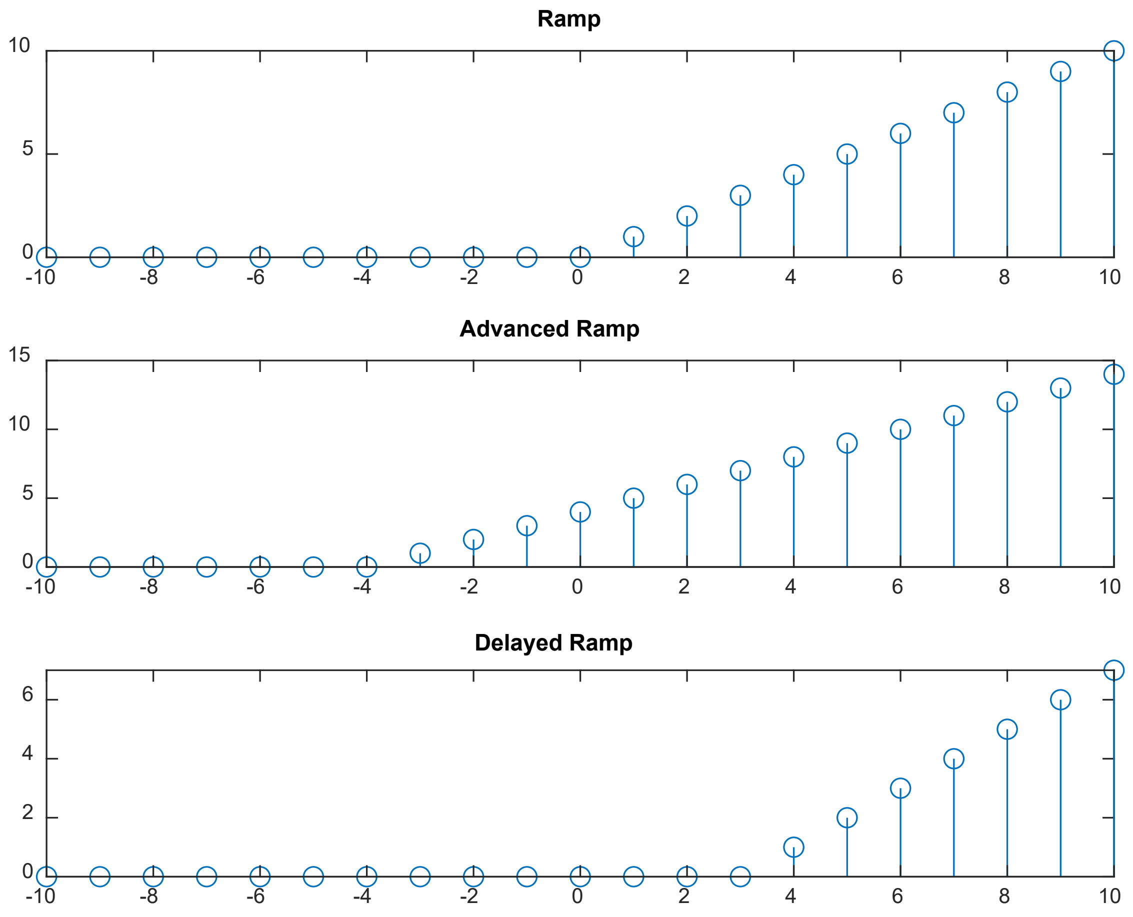


Figure 3 for Part A.3

Part B

Addition of Two Sequence

```
x1=[2 4 1 6 4 9];
x2=[3 5 6 9];
n1=0:5;
n2=-1:2;
n=min(n1(1),n2(1)):max(n1(end),n2(end));
y1=zeros(1,length(n));
y2=y1;
y1((n>=n1(1))&(n<=n1(end)))=x1;
y2((n>=n2(1))&(n<=n2(end)))=x2;
y=y1+y2;
subplot(3,1,1),stem(n,y1),title('First Sequence');
subplot(3,1,2),stem(n,y2),title('Second Sequence');
subplot(3,1,3),stem(n,y),title('Addition');
```

Output:

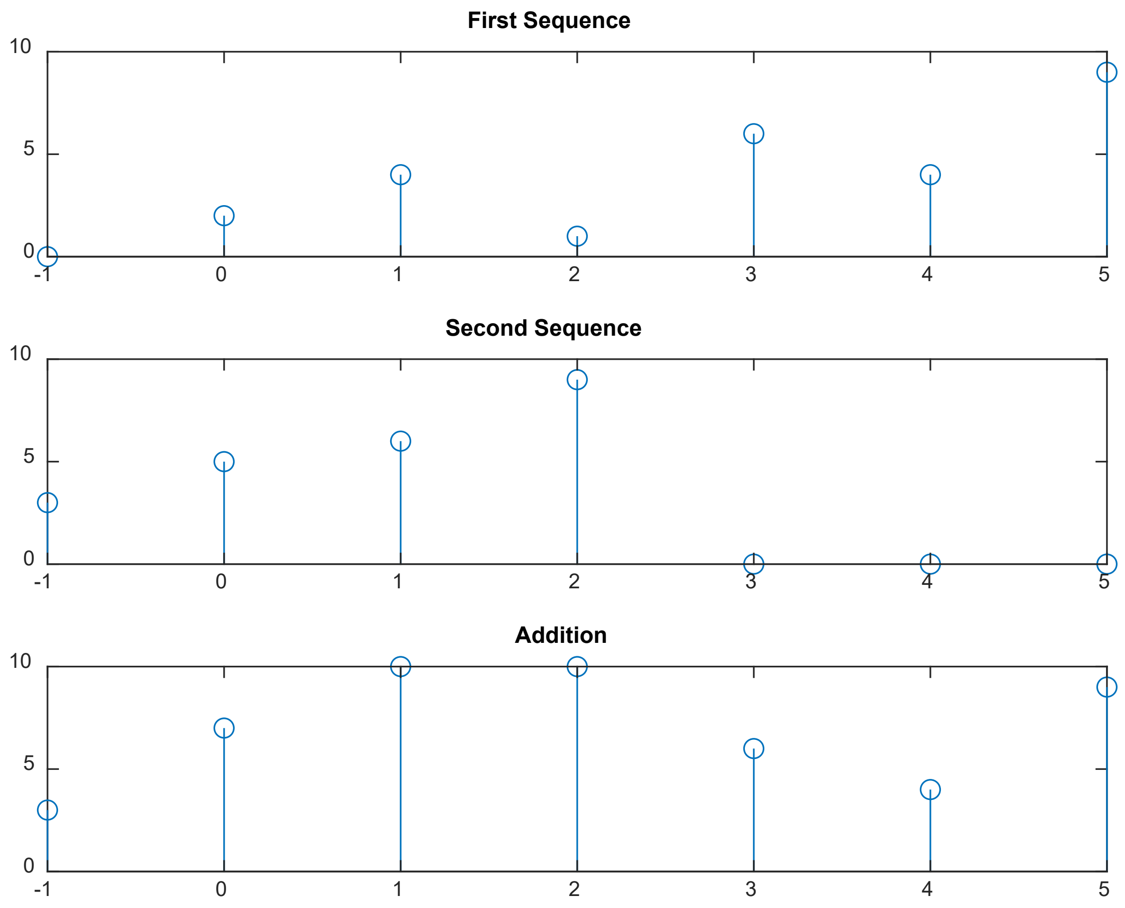


Figure 4 for Part B

Part C

Up-sampling & Down-sampling

```
n=1:60;
xs=sin(.3*n);
subplot(3,1,1),stem(xs),title('Normally Sampled');
y=upsample(xs,2);
subplot(3,1,2),stem(y),title('Upsampled');
y=downsample(xs,2);
subplot(3,1,3),stem(y),title('Downsampled');
```

Output:

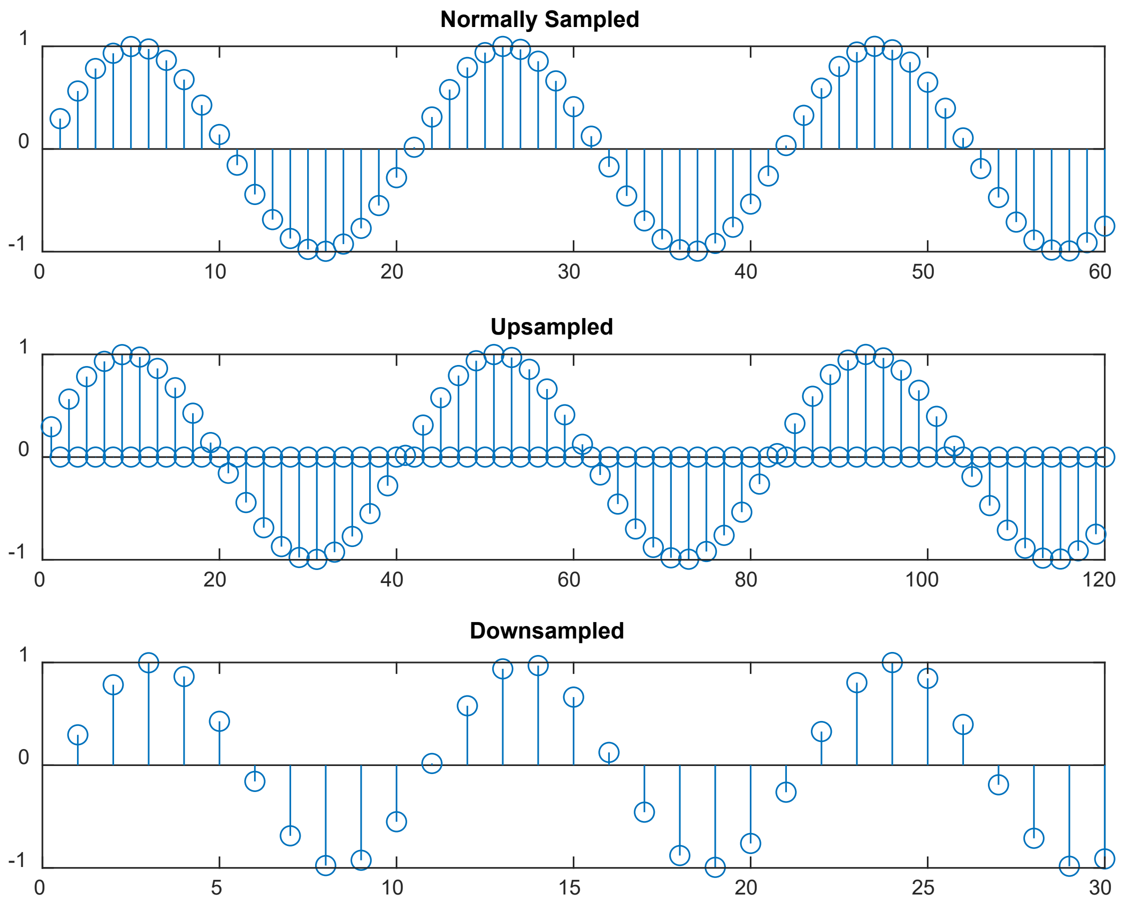


Figure 5 for Part C

Part D

Even Part & Odd Part

```
n=-15:15;
step=((n+1)>=0);
x=(n+1).*step;
xf=fliplr(x);
even=(x+xf)/2;
odd=(x-xf)/2;
subplot(3,1,1),stem(n,x),title('Sequence');
subplot(3,1,2),stem(n,even),title('Even Part');
subplot(3,1,3),stem(n,odd),title('Odd Part');
```

Output:

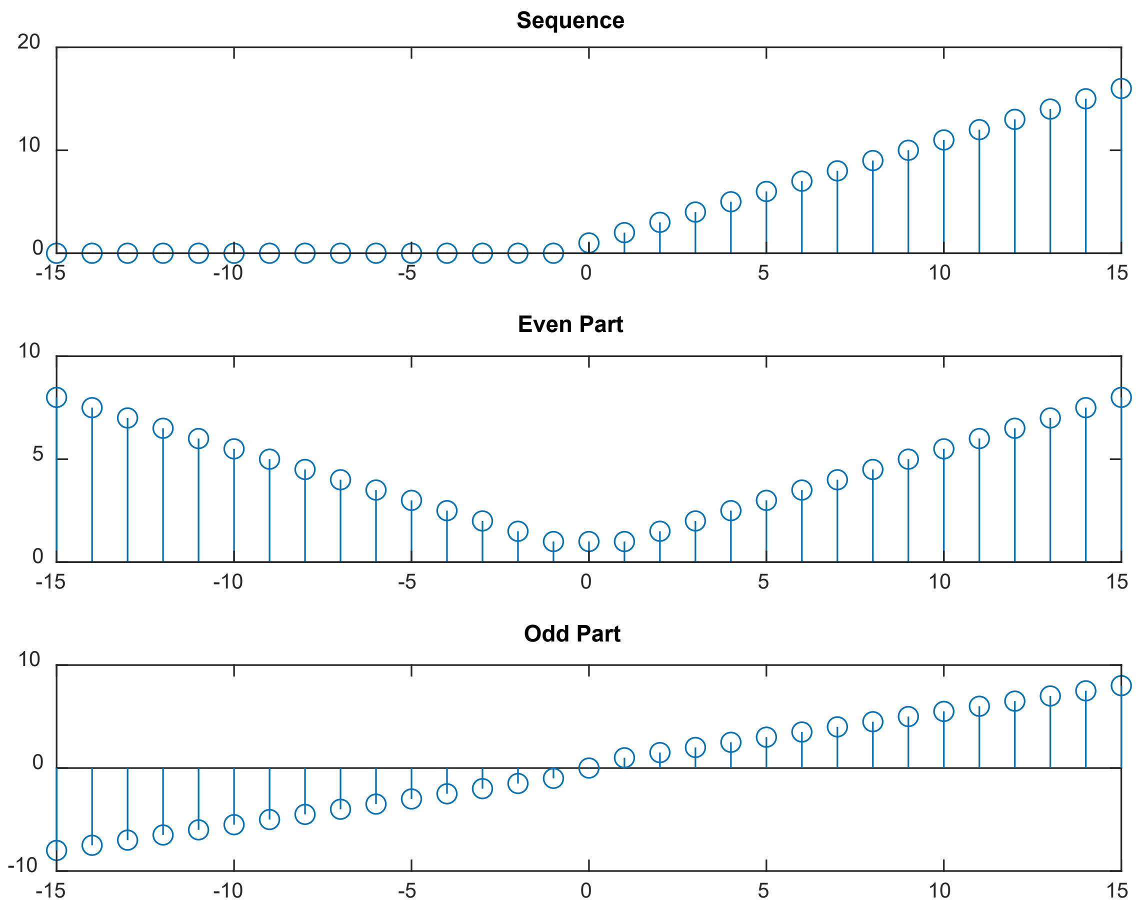


Figure 6 for Part D

Part E

Convolution

```
x1=[4 2 6 3 8 1 5];
x2=[3 8 6 9 6 7];
n1=-2:4;
n2=-4:1;
n=(n1(1)+n2(1)):(n1(end)+n2(end));
y=conv(x1,x2);
subplot(3,1,1),stem(n1,x1),title('1st Sequence');
subplot(3,1,2),stem(n2,x2),title('2nd Sequence');
subplot(3,1,3),stem(n,y),title('Convolution');
```

Output:

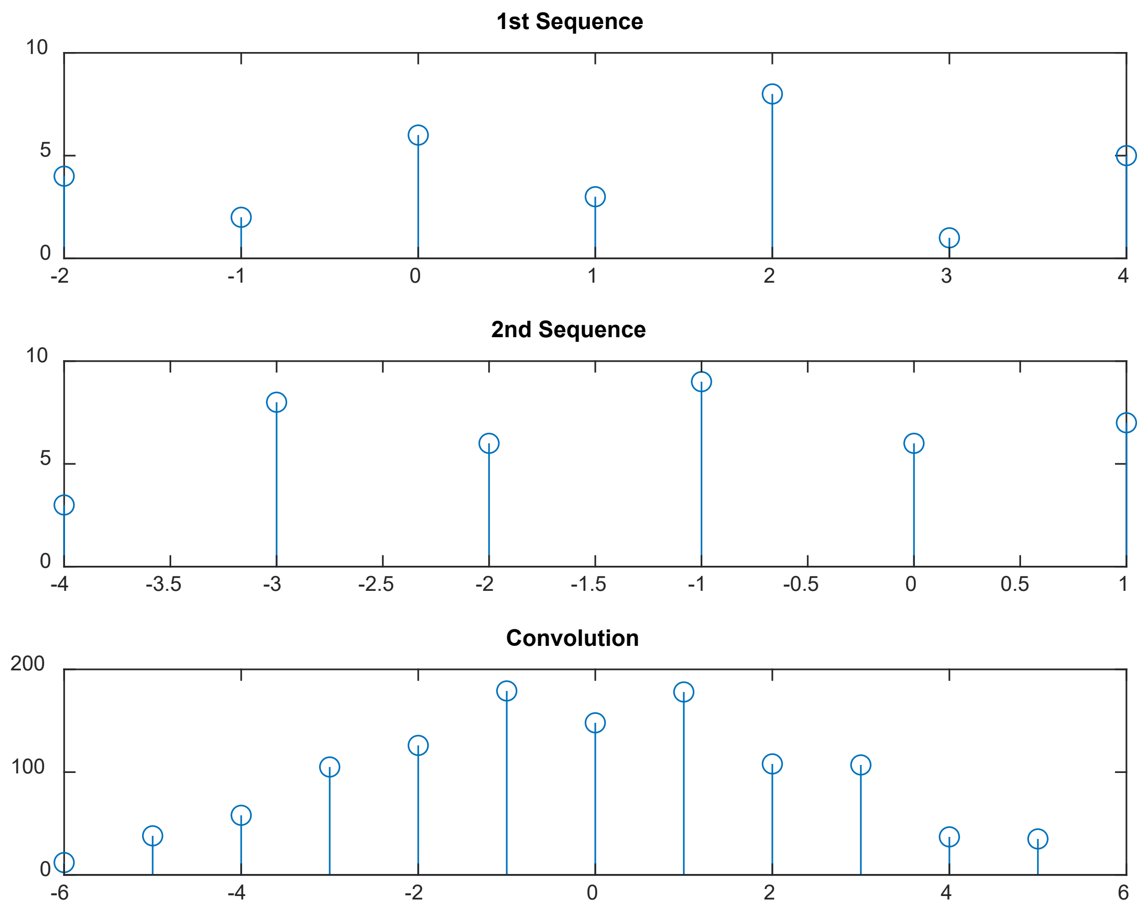


Figure 7 for Part E

Part F

F.1: Auto-Correlation

```
x=[4 7 5 8 7 9 1 4];
nx=-2:5;
subplot(2,1,1),stem(nx,x),title('Sequence');
[r,n]=xcorr(x);
subplot(2,1,2),stem(n,r),title('AutoCorrelation');
```

Output:

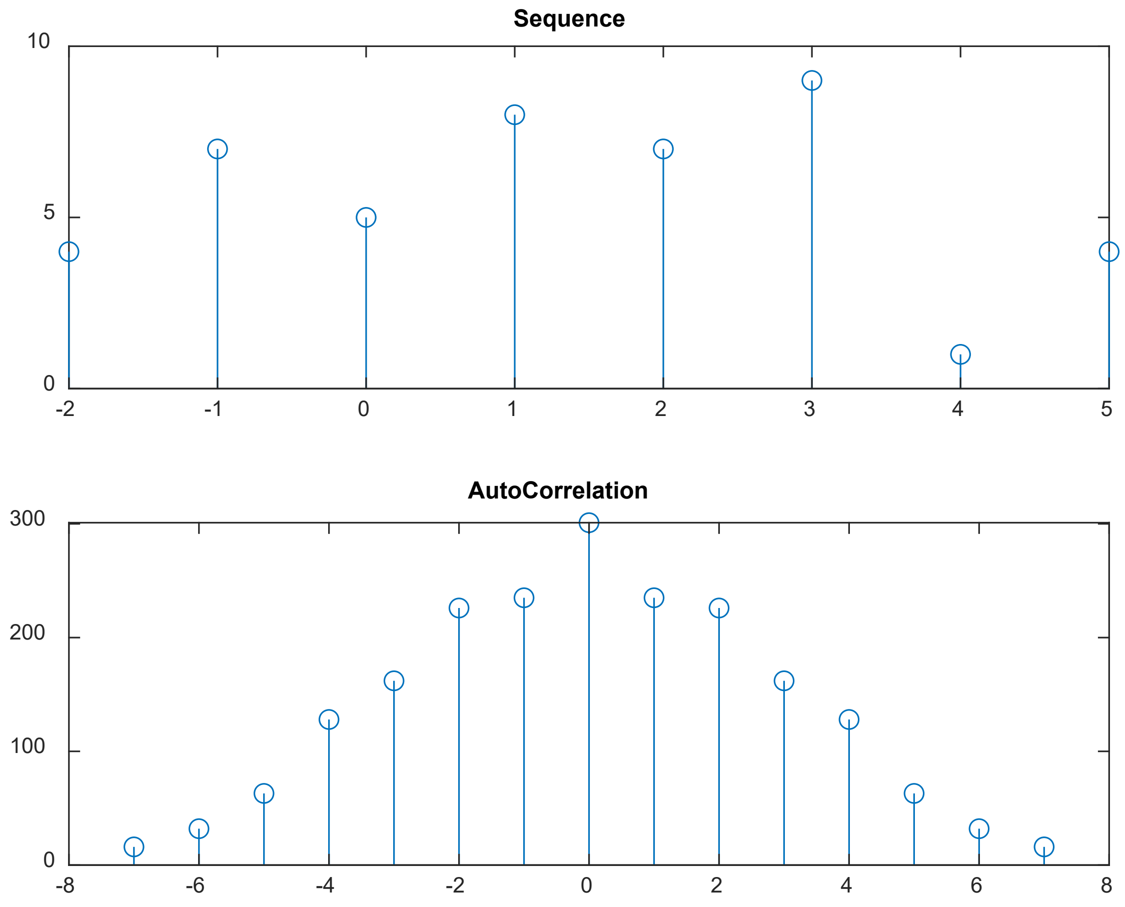


Figure 8 for Part F.1

F.2: Cross-Correlation

```
x1=[2 1 0 3 4 3 2 2];
x2=[3 4 6 5 3 7];
n1=-1:6;
n2=-4:1;
n=(n1(1)+n2(1)):(n1(end)+n2(end));
r=conv(x1,fliplr(x2));
subplot(3,1,1),stem(n1,x1),title('1st Sequence');
subplot(3,1,2),stem(n2,x2),title('2nd Sequence');
subplot(3,1,3),stem(n,r),title('CrossCorrelation');
```

Output:

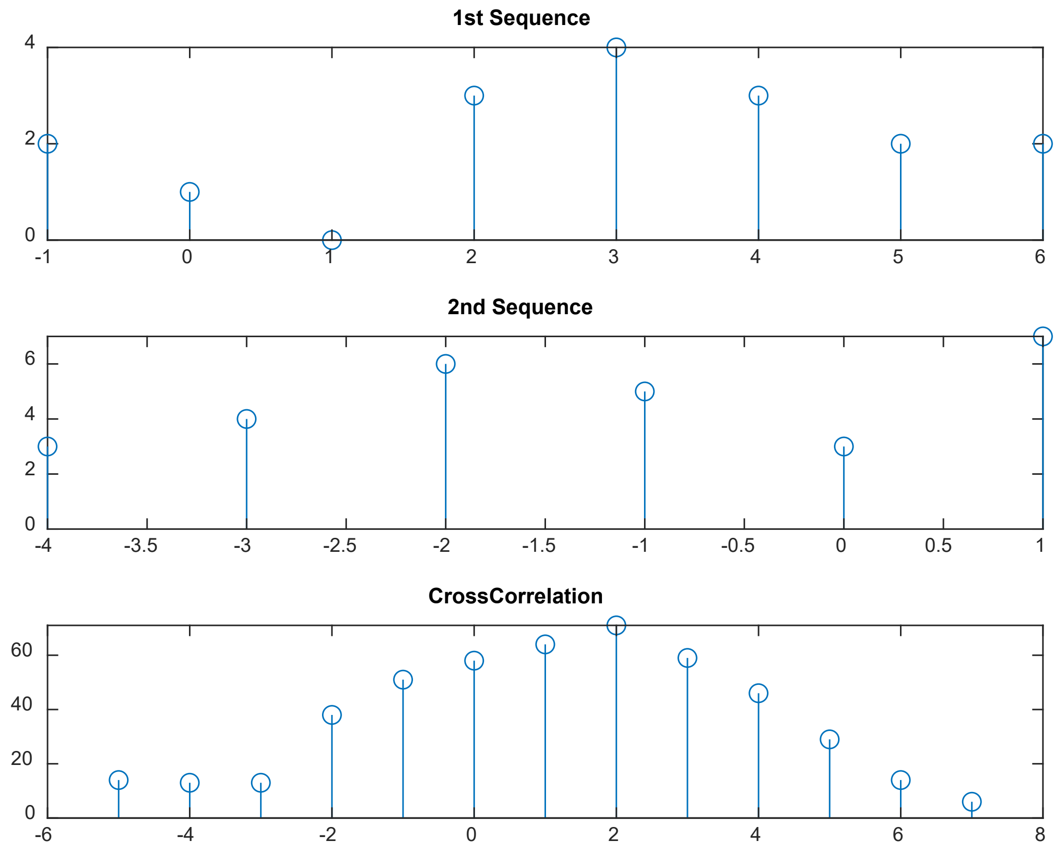


Figure 9 for Part F.2

Discussion:

You can search any function on MATLAB website: https://www.mathworks.com/help/

Documentation link to some important functions are given below:

• zeros()	https://www.mathworks.com/help/matlab/ref/zeros.html
• ones()	https://www.mathworks.com/help/matlab/ref/ones.html
• upsample()	https://www.mathworks.com/help/signal/ref/upsample.html
• downsample()	https://www.mathworks.com/help/signal/ref/downsample.html
• fliplr()	https://www.mathworks.com/help/matlab/ref/fliplr.html
• conv()	https://www.mathworks.com/help/matlab/ref/conv.html
• xcorr()	https://www.mathworks.com/help/matlab/ref/xcorr.html

❖ Comment after each output. Discuss why the output is the way it is (in each part). Lastly, write one sentence for each of the seven functions mentioned above.

EEE 314: Digital Signal Processing I Lab

Experiment – 3

Study of z-Transform

Theory:

The z-transform plays the same role in the analysis of discrete time signals and LTI systems as the Laplace transform does in the analysis of continuous-time signals and LTI systems. The use of z-transform techniques permits simple algebraic manipulations. The z-transform has become an important tool in the analysis and design of digital filters. In this experiment, different features of z-transform have been discussed through various examples. At the end of this experiment, one will be able to perform z-transform, inverse z-transform, system analysis through stability and causality.

The z-transform of a discrete-time signal x[n] is defined as the power series

$$X(z) = \sum_{n=-\infty}^{n=\infty} x[n]z^{-n}$$

where z is a complex variable. Since the z-transform is an infinite power series, it exists only for those values of z for which this series converges. The region of convergence (ROC) of X(z) is the set of all values of z for which X(z) attains a finite value.

The zeros of a z-transform X(z) are the values of z for which $X(z) = 0$. The poles of a z-transform are the values of z for which $X(z) = \infty$. By definition, the ROC of a z-transform should not contain any poles. The time behavior of a signal depends strongly on the location of its poles relative to the unit circle. Zeros also affect the behavior of a signal but not as strongly as poles.

Lab Work:

Part A

A.1: z-Transform using Symbolic Variable

We want to find the z-transform of $x[n] = a^n u[n]$

```
syms a n;
x=a^n;
disp(ztrans(x));
```

Output:

```
    -z/(a - z)
```

The output should be $\frac{1}{1-az^{-1}}$, which is basically same as $\frac{-z}{a-z} = \frac{z}{z-a}$

Now we want to find the z-transform of $x[n] = \left({}^{1}\!/_{3}\right)^{n} u[n]$

```
syms n;
x=(1/3)^n;
disp(ztrans(x));
```

Output:

```
      z/(z - 1/3)
```

The output should be $\frac{1}{1-\frac{1}{3}z^{-1}}$, which is basically same as $\frac{z}{z-\frac{1}{3}}$

Now we want to find the z-transform of $x[n] = 2^{n}u[n]$

```
syms n;
x=2^n;
disp(ztrans(x));
```

Output:

```
      z/(z - 2)
```

The output should be $\frac{1}{1-2z^{-1}}$, which is basically same as $\frac{z}{z-2}$

A.2: z-Transform of Finite-Duration Sequence

We want to find the z-transform of $x[n] = [5\ \ 2\ \ \underset{\uparrow}{4}\ \ 6\ \ 7]$

```
syms z;
x=[5 2 4 6 7];
i=1;
ZT=0;
for n=-2:2
    ZT=ZT+x(i)*z^-n;
    i=i+1;
end
disp(simplify(ZT));
```

Output:

```
      (5*z^4 + 2*z^3 + 4*z^2 + 6*z + 7)/z^2
```

The output is same as $5z^2 + 2z + 4 + 6z^{-1} + 7z^{-2}$, as expected. We have to remember that, using simplify() is not mandatory.

A.3: Convolution Property of z-Transform

We want to check the convolution property of z-transform.

$$\text{If } x_1[n] \overset{z}{\leftrightarrow} X_1(z) \text{ and } x_2[n] \overset{z}{\leftrightarrow} X_2(z)$$

$$\text{Then } x[n] = x_1[n] * x_2[n] \overset{z}{\leftrightarrow} X(z) = X_1(z)X_2(z)$$

```
syms z;
x1=[2 3 4];
i=1;
ZT1=0;
for n=0:2
    ZT1=ZT1+x1(i)*z^-n;
    i=i+1;
end
x2=[3 4 5 6];
i=1;
ZT2=0;
for n=0:3
    ZT2=ZT2+x2(i)*z^-n;
    i=i+1;
end
ZT=ZT1*ZT2;
disp(simplify(expand(ZT)));
disp(conv(x1,x2));
```

Output:

```
     (6*z^5 + 17*z^4 + 34*z^3 + 43*z^2 + 38*z + 24)/z^5
     6    17    34    43    38    24
```

We can see that, the convolution of two time-domain signals is equivalent to multiplication of their corresponding z-transforms.

Part B

B.1: Partial-Fraction Expansion

We want to decompose the following function by partial-fraction expansion:

$$X(z) = \frac{z}{3z^2 - 4z + 1} = \frac{z^{-1}}{3 - 4z^{-1} + z^{-2}}$$

```
b=[0 1];
a=[3 -4 1];
[r,p,k]=residuez(b,a);
disp(rats([r,p]));
disp(rats(k));
```

Output:

```
      1/2               1
      -1/2              1/3
```

1st column has residues, 2nd column has poles. ‘k’ (direct term) is empty here, since `disp(rats(k))` produces no line in this example. So,

$$X(z) = \frac{{}^{1}\!/_{2}}{1 - z^{-1}} + \frac{-{}^{1}\!/_{2}}{1 - \frac{1}{3}z^{-1}}$$

B.2: Inverse z-Transform

If we want to find the inverse z-transform of the function done in Part B.1, we can simply use iztrans() function. We have to remember that iztrans() will consider the causal sequence case.

```
syms z;
X=(z^-1)/(3-4*z^-1+z^-2);
disp(iztrans(X));
```

Output:

```
     1/2 - (1/3)^n/2
```

We can see here, the output is $x[n] = \frac{1}{2}u[n] - \frac{1}{2}\left(\frac{1}{3}\right)^n u[n]$, which also corresponds to the inverse z-transform of partial-fraction expansion done in Part B.1 using residuez().

Now we want to do the inverse z-transform on 2nd example of Part A.1, that is

$$X(z) = \frac{1}{1 - \frac{1}{3}z^{-1}}$$

```
syms z;
X=1/(1-(1/3)*z^-1);
disp(iztrans(X));
```

Output:

```
     (1/3)^n
```

We can see here, the output is $x[n] = \left(\frac{1}{3}\right)^n u[n]$, as expected (considering causal sequence case).

It should be noted that, there are three methods that are often used for the evaluation of the inverse z-transform in practice:

- Direct evaluation by contour integration
- Power series expansion
- Partial-fraction expansion and table lookup

Part C

Impulse Response Plot

We want to plot the impulse response of a system describer by the following transfer function:

$$H(z) = \frac{1 + 2z^{-1}}{1 + 0.4z^{-1} - 0.12z^{-2}}$$

```
n=-10:10;
b=[1 2];
a=[1 0.4 -0.12];
impulse_response=impz(b,a,n);
stem(n,impulse_response),title('Impulse Response using impz()');
```

Output:

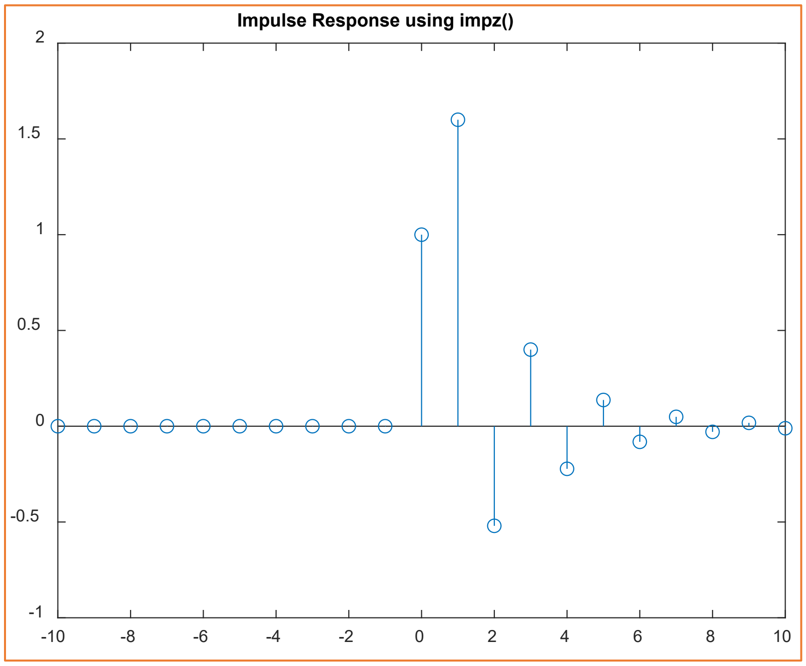


Figure 1 for Part C

We can also find impulse response using filter(), but for that, we have to define impulse function first. We can find any response (step, sinusoid etc.) using filter() too. The code would be:

```
n=-10:10;
b=[1 2];
a=[1 0.4 -0.12];
resp=impz(b,a,n);
subplot(2,1,1),stem(n,resp),title('Impulse Response using impz()');
impulse=(n==0);
resp=filter(b,a,impulse);
subplot(2,1,2),stem(n,resp),title('Impulse Response using filter()');
```

Output:

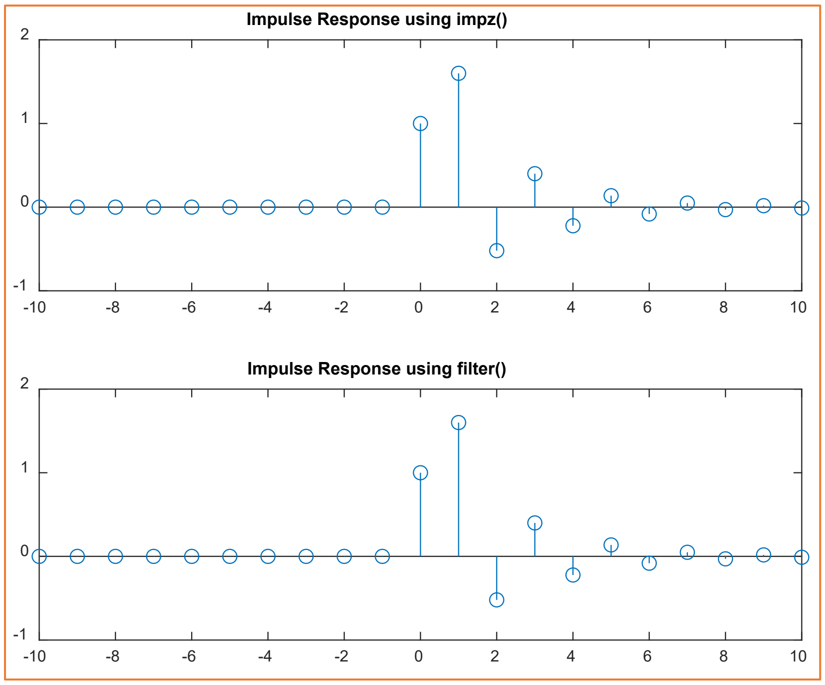


Figure 2 for Part C

Part D

Pole-Zero Plot

A pole-zero plot shows the location in the complex plane of the poles and zeros of the transfer function of a dynamic system. The ROC for a given DT transfer function is a disk or annulus which contains no poles. In general, the ROC is not unique, and the particular ROC in any given case depends on whether the system is causal or anti-causal.

We want to draw the pole-zero plot of an LTI system described by the following LCCDE:

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

Using z-transform,

$$Y(z) = z^{-1}Y(z) + X(z)$$
$$\Rightarrow Y(z)(1 - z^{-1}) = X(z)$$
$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - z^{-1}}$$

```
b=1;
a=[1 -1];
zplane(b,a),title('y[n] - y[n-1] = x[n]');
```

Output:

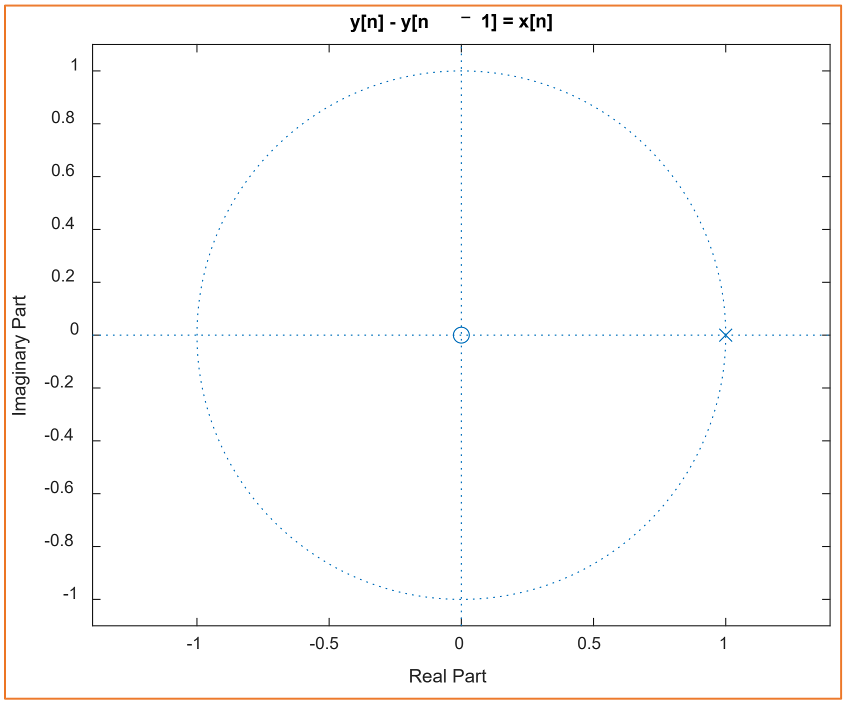


Figure 3 for Part D

From the pole-zero plot, we clearly see that it has one pole at 1, marked by cross (×); one zero at origin, marked by circle (○).

- We know that, an LTI system is causal if and only if the ROC of the system function is the exterior of a circle of radius $r < \infty$, including the point $z = \infty$.
- We also know that, an LTI system is BIBO stable if and only if the ROC of the system function includes the unit circle.

Thus, a causal LTI system is BIBO stable if and only if all the poles of H(z) are inside the unit circle.

If this given system (in this example) is said to be causal, then the ROC: $|z| > 1$. But this ROC doesn't include unit circle. So, the system here is unstable.

Now, we want to draw the pole-zero plot of another LTI system described by the following LCCDE:

$$y[n] = \frac{1}{2}y[n-1] + x[n] + \frac{1}{3}x[n-1]$$

Using z-transform,

$$Y(z) = \frac{1}{2}z^{-1}Y(z) + X(z) + \frac{1}{3}z^{-1}X(z)$$

$$\Rightarrow Y(z)\left(1 - \frac{1}{2}z^{-1}\right) = X(z)\left(1 + \frac{1}{3}z^{-1}\right)$$

$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{1 + \frac{1}{3}z^{-1}}{1 - \frac{1}{2}z^{-1}}$$

```
b=[1 1/3];
a=[1 -1/2];
zplane(b,a);title('y[n] - (1/2)y[n-1] = x[n] + (1/3)x[n-1]');
```

Output:

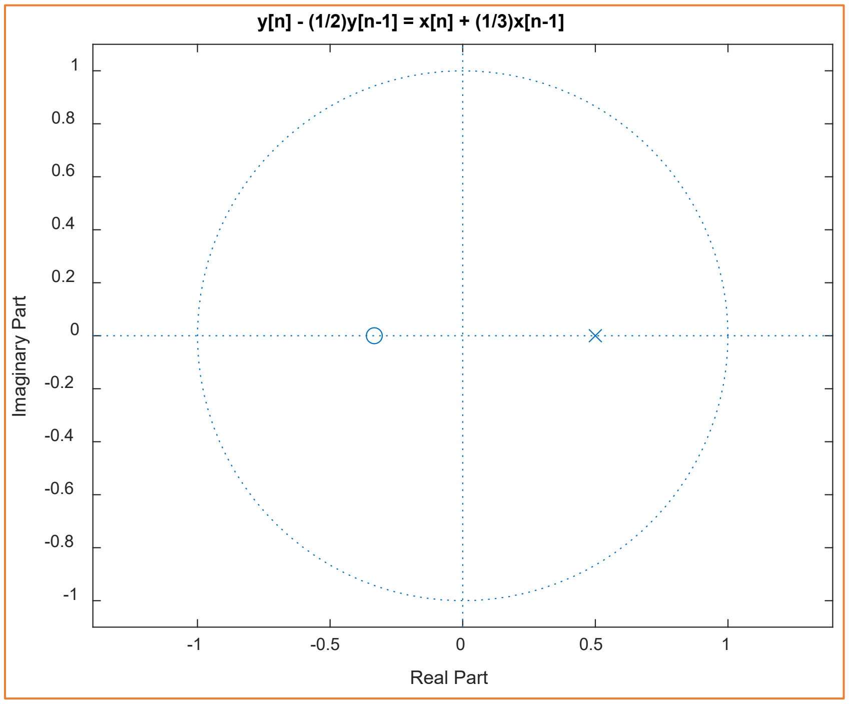


Figure 4 for Part D

The pole is at $^1/_2$, the zero is at $-^1/_3$.

If the system causal, then ROC: $|z| > {}^1/_2$. This system is stable, because unit circle in included in ROC.

If the system anti-causal, then ROC: $|z| < {}^1/_2$. This system is unstable, because unit circle isn't included in ROC here.

Now, we want to draw the pole-zero plot of an LTI system described by the following transfer function:

$$H(z) = \frac{3 - 4z^{-1}}{1 - 3.5z^{-1} + 1.5z^{-2}}$$

```
b=[3 -4];
a=[1 -3.5 1.5];
[r,p,k]=residuez(b,a);
disp(rats([r,p]));
disp(rats(k));
zplane(b,a);
```

Output:

```
     2          3
     1          1/2
```

From this partial-fraction expansion, we can rewrite the transfer function as:

$$H(z) = \frac{2}{1-3z^{-1}} + \frac{1}{1-\frac{1}{2}z^{-1}}$$

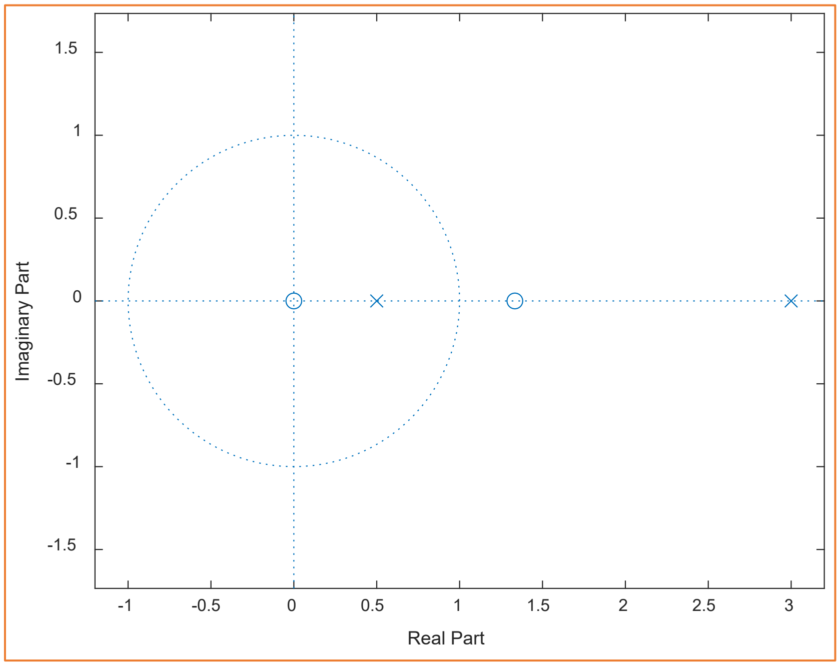


Figure 5 for Part D

From the pole-zero plot, we can see that this system has two poles, at ${}^{1}/{}_{2}$ and 3.

If the system causal, then ROC: $|z| > 3$. This system is unstable.
If the system anti-causal, then ROC: $|z| < {}^{1}/{}_{2}$. This system is unstable.
If the system noncausal, then ROC: ${}^{1}/{}_{2} < |z| < 3$. This system is stable.

Part E

Pole-Zero Cancellation

We want to cancel poles of an LTI system described by the following LCCDE:

$$y[n] = \frac{5}{6}y[n-1] - \frac{1}{6}y[n-2] + x[n]$$

Using z-transform,

$$Y(z) = \frac{5}{6}z^{-1}Y(z) - \frac{1}{6}z^{-2}Y(z) + X(z)$$
$$\Rightarrow Y(z)\left(1 - \frac{5}{6}z^{-1} + \frac{1}{6}z^{-2}\right) = X(z)$$
$$\Rightarrow H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - \frac{5}{6}z^{-1} + \frac{1}{6}z^{-2}}$$

```
b=1;
a=[1 -5/6 1/6];
[r,p,k]=residuez(b,a);
disp(rats([r,p]));
disp(rats(k));
zplane(b,a),title('Pole-Zero Plot');
```

Output:

```
     3           1/2
    -2           1/3
```

From this partial-fraction expansion, we can rewrite the transfer function as:

$$H(z) = \frac{3}{1-\frac{1}{2}z^{-1}} + \frac{-2}{1-\frac{1}{3}z^{-1}}$$

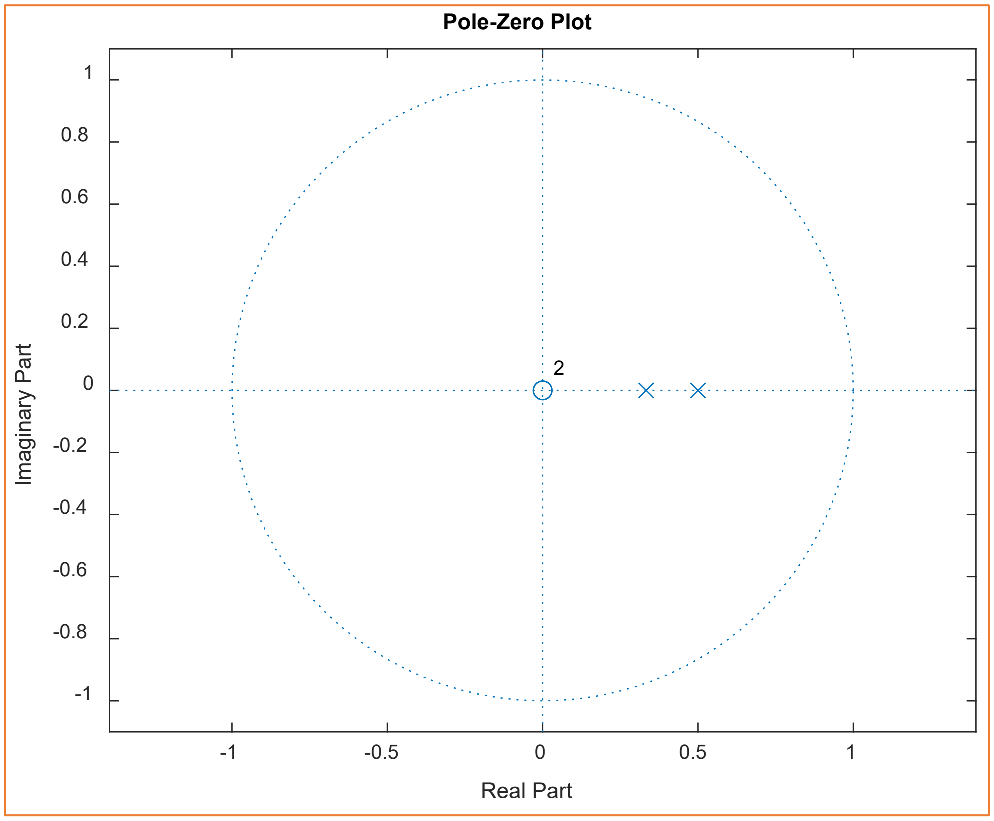


Figure 6 for Part E

From the pole-zero plot, we can see that this system has two poles, at $^1/_3$ and $^1/_2$.

We want to cancel the pole at $^1/_3$, so we simply use input $x[n] = \delta[n] - \frac{1}{3}\delta[n-1]$.

The input has z-transform as:

$$X(z) = 1 - \frac{1}{3}z^{-1}$$

```
b=1;
a=[1 -5/6 1/6];
x1=[1 -1/3];
b1=conv(b,x1);
zplane(b1,a),title('p = 1/3 is cancelled');
[r,p,k]=residuez(b1,a);
disp(rats([r,p]));
disp(rats(k));
```

Output:

```
     1          1/2
     0          1/3
```

From this partial-fraction expansion, we can write output z-transform as:

$$Y(z) = X(z)H(z) = \frac{1}{1-\frac{1}{2}z^{-1}}$$

The mode $\left({}^{1}\!/_{3}\right)^{n}$ is suppressed as a result of pole-zero cancellation.

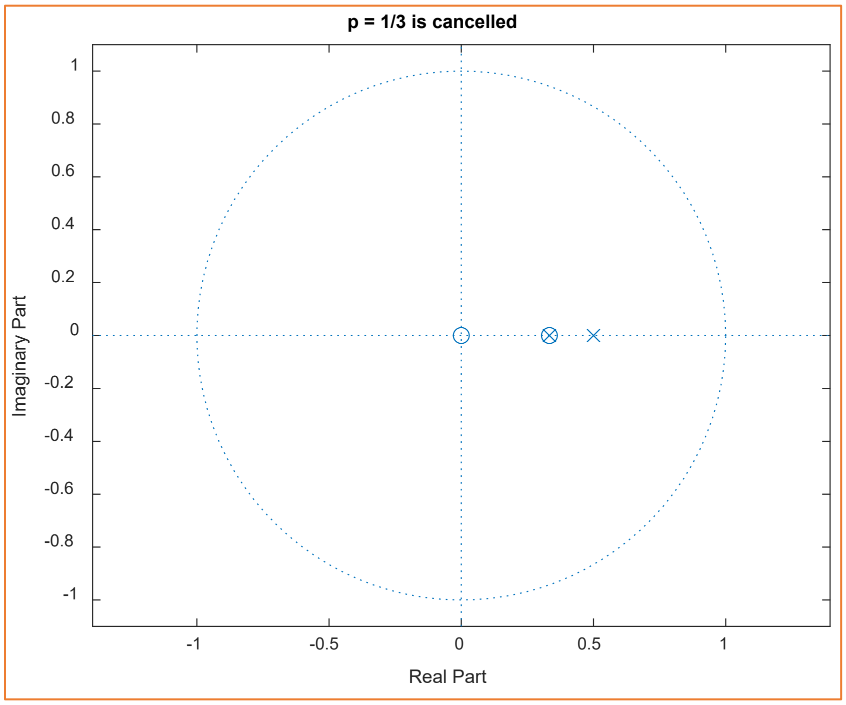


Figure 7 for Part E

From this pole-zero plot, we can see the input zero at ${}^{1}\!/_{3}$ is cancelling the system pole at ${}^{1}\!/_{3}$.

Now, we only want to cancel the pole at ${}^{1}\!/_{2}$, so we simply use input $x[n] = \delta[n] - \frac{1}{2}\delta[n-1]$.

This input has z-transform as:

$$X(z) = 1 - \frac{1}{2}z^{-1}$$

```
b=1;
a=[1 -5/6 1/6];
x2=[1 -1/2];
b2=conv(b,x2);
zplane(b2,a),title('p = 1/2 is cancelled');
[r,p,k]=residuez(b2,a);
disp(rats([r,p]));
disp(rats(k));
```

Output:

```
     0          1/2
     1          1/3
```

From this partial-fraction expansion, we can write output z-transform as:

$$Y(z) = H(z)H(z) = \frac{1}{1 - \frac{1}{3}z^{-1}}$$

The mode $\left({}^{1}/_{2}\right)^{n}$ is suppressed as a result of pole-zero cancellation.

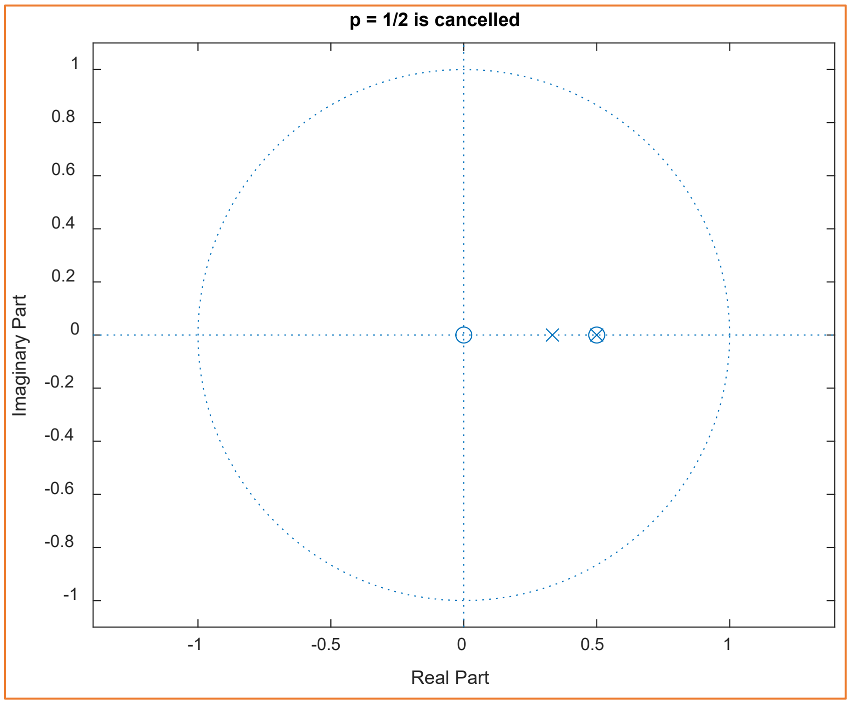


Figure 8 for Part E

From this pole-zero plot, we can see the input zero at ${}^{1}/_{2}$ is cancelling the system pole at ${}^{1}/_{2}$.

Discussion:

You can search any function on MATLAB website: https://www.mathworks.com/help/

Documentation link to some important functions are given below:

• syms	https://www.mathworks.com/help/symbolic/syms.html
• ztrans()	https://www.mathworks.com/help/symbolic/ztrans.html
• simplify()	https://www.mathworks.com/help/symbolic/simplify.html
• expand()	https://www.mathworks.com/help/symbolic/expand.html
• residuez()	https://www.mathworks.com/help/signal/ref/residuez.html
• rats()	https://www.mathworks.com/help/matlab/ref/rats.html
• iztrans()	https://www.mathworks.com/help/symbolic/iztrans.html
• impz()	https://www.mathworks.com/help/signal/ref/impz.html
• zplane()	https://www.mathworks.com/help/signal/ref/zplane.html

❖ Comment after each output. Discuss why the output is the way it is (in each part). Lastly, write one sentence for each of the nine functions mentioned above.

EEE 314: Digital Signal Processing I Lab

Experiment – 4

Frequency Domain Analysis of DT Signals and Systems

Theory:

Frequency analysis of signal involves the resolution of the signal into its frequency (sinusoidal) components. For the class of periodic signals, such decomposition is called a Fourier series. For the class of finite energy (aperiodic) signals, the decomposition is called the Fourier transform. These decompositions are extremely important in the analysis of LTI system because the response of an LTI system to a sinusoidal input is a sinusoid of same frequency but of different amplitude and phase.

The Discrete-Time Fourier Transform (DTFT) of a discrete-time signal x[n] is defined as

$$\mathrm{X}(\omega) = \sum_{n=-\infty}^{n=\infty} x[n]e^{-j\omega n}$$

$\mathrm{X}(\omega)$ is a continuous function of ω, where $\omega = \Omega T_S$. Also, $\mathrm{X}(\omega)$ is periodic with period of 2π.

$$\mathrm{X}(\omega) = \mathrm{X}(\omega + 2\pi k)$$

where 'k' is any integer.

To convert $\mathrm{X}(\omega)$ to $x[n]$, inverse DTFT (IDTFT) is applied:

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{X}(\omega)\, e^{j\omega n} d\omega$$

A finite-duration signal $x[n]$ of length L has Discrete Fourier Transform (DFT) of N-point (where $\mathrm{N} \geq \mathrm{L}$) as

$$\mathrm{X}(k) = \sum_{n=0}^{n=N-1} x[n]e^{-j2\pi kn/N}$$

where k = 0, 1, 2, …, N–1

The N-point inverse DFT (IDFT) is:

$$x[n] = \frac{1}{N}\sum_{k=0}^{k=N-1} \mathrm{X}(k)e^{j2\pi kn/N}$$

where n = 0, 1, 2, …, N–1

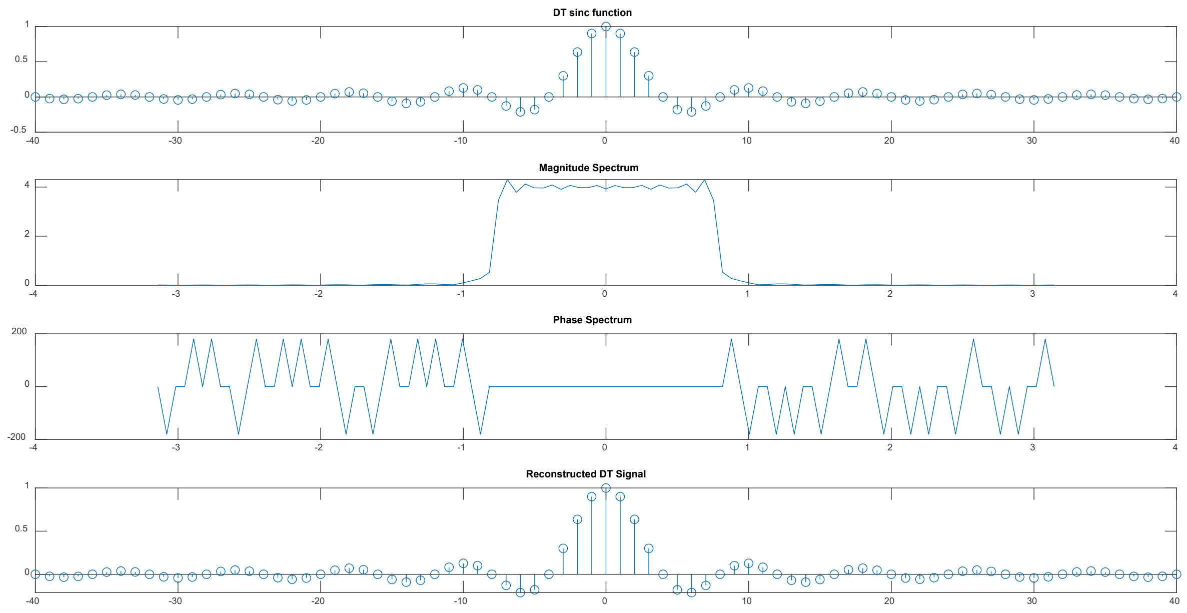
Lab Work:

Part A

DTFT & IDTFT

```
n=-40:40;
x=sinc(n/4);
subplot(4,1,1),stem(n,x),title('DT sinc function');
M=101;
dw=2*pi/(M-1);
w=-pi:dw:pi;
X=zeros(1,M);
for i=1:M
    for j=1:length(x)
        X(i)=X(i)+x(j)*exp(-1i*w(i)*n(j));
    end
end
subplot(4,1,2),plot(w,abs(X)),title('Magnitude Spectrum');
subplot(4,1,3),plot(w,angle(X)*180/pi),title('Phase Spectrum');
n1=-40:40;
xr=zeros(1,length(n1));
for i=1:M
    for j=1:length(xr)
        xr(j)=xr(j)+1/(2*pi)*X(i)*exp(1i*w(i)*n1(j))*dw;
    end
end
subplot(4,1,4),stem(n1,real(xr)),title('Reconstructed DT Signal');
```

Output:

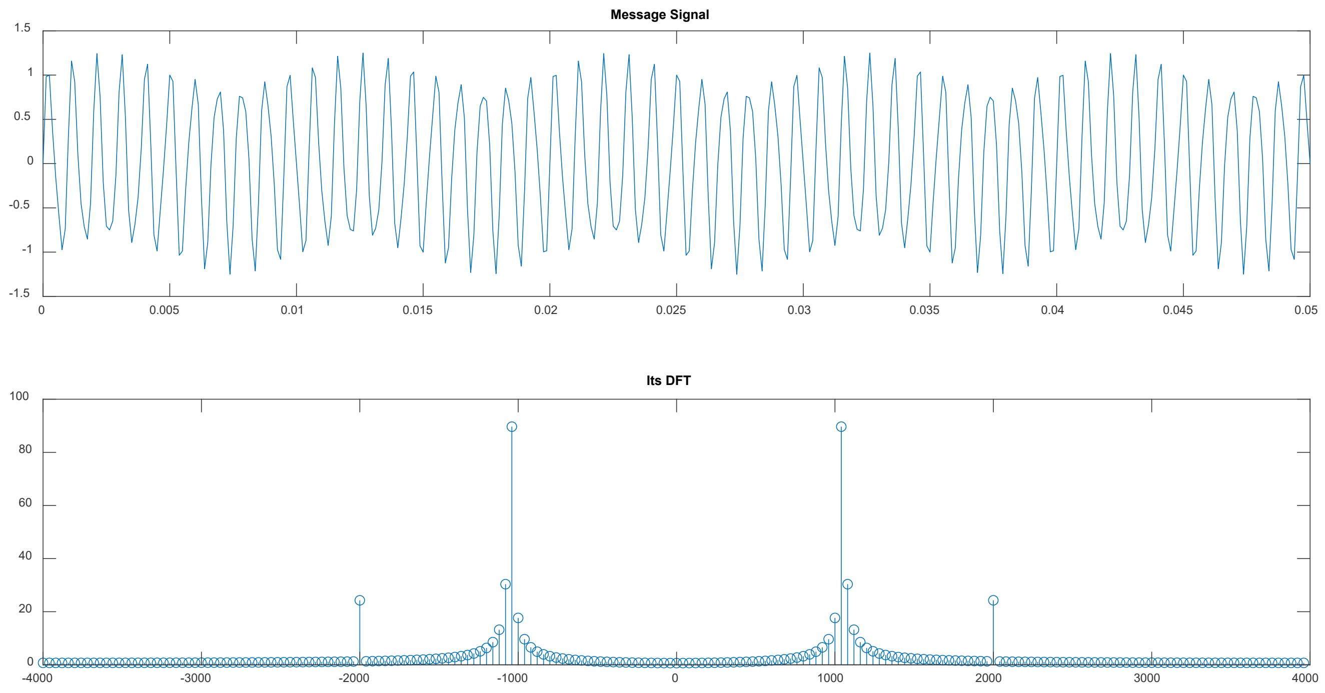


Part B

B.1: DFT for a Simple Signal

```
N=200;
Fs=8000;
F1=1050;
F2=2000;
t=0:1/Fs:0.05;
x=sin(2*pi*F1*t)+0.25*sin(2*pi*F2*t);
f=(0:1/N:1-1/N)*Fs-Fs/2;
subplot(2,1,1),plot(t,x),title('Message Signal');
subplot(2,1,2),stem(f,abs(fftshift(fft(x,N)))),title('Its DFT');
```

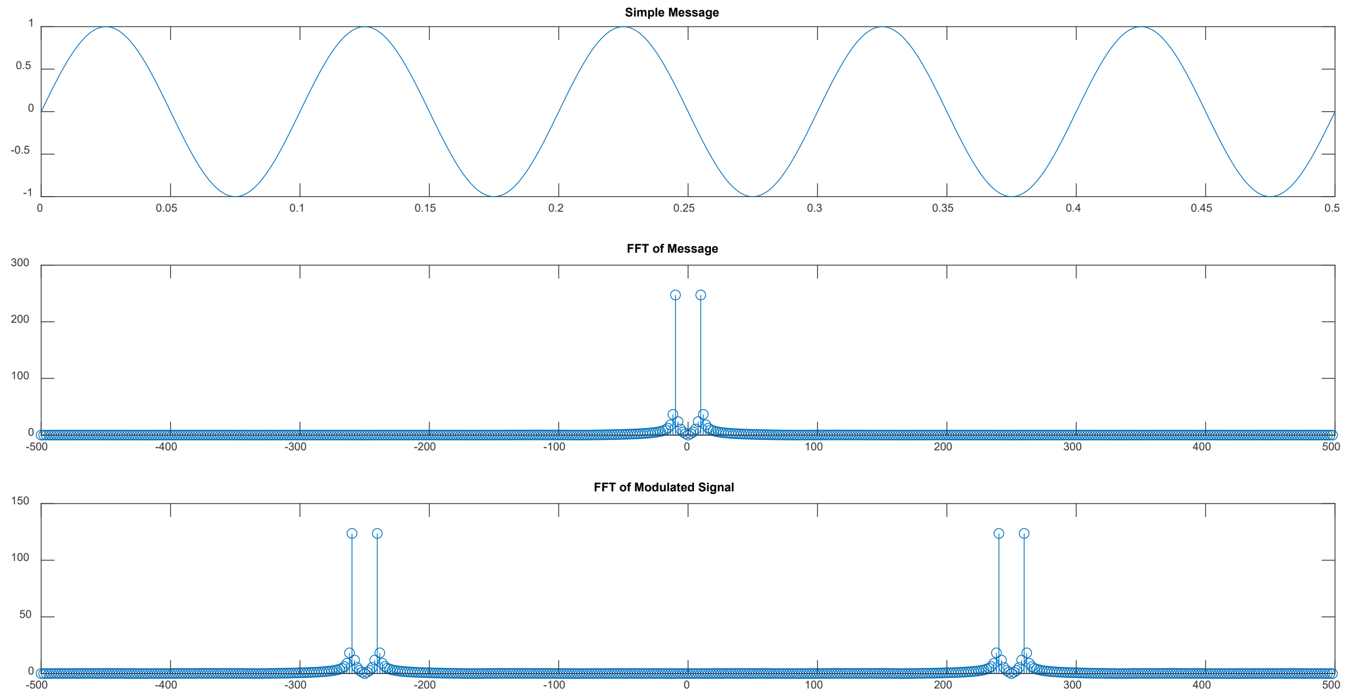
Output:



B.2: DFT for a Modulated Signal

```
N=512;
Fc=250;
Fs=1000;
t=0:1/Fs:0.5;
message=sin(2*pi*10*t);
subplot(3,1,1),plot(t,message),title('Simple Message');
carrier=cos(2*pi*Fc*t);
transmit=message.*carrier;
f=(0:1/N:1-1/N)*Fs-Fs/2;
subplot(3,1,2),stem(f,abs(fftshift(fft(message,N))));
title('FFT of Message');
subplot(3,1,3),stem(f,abs(fftshift(fft(transmit,N))));
title('FFT of Modulated Signal');
```

Output:

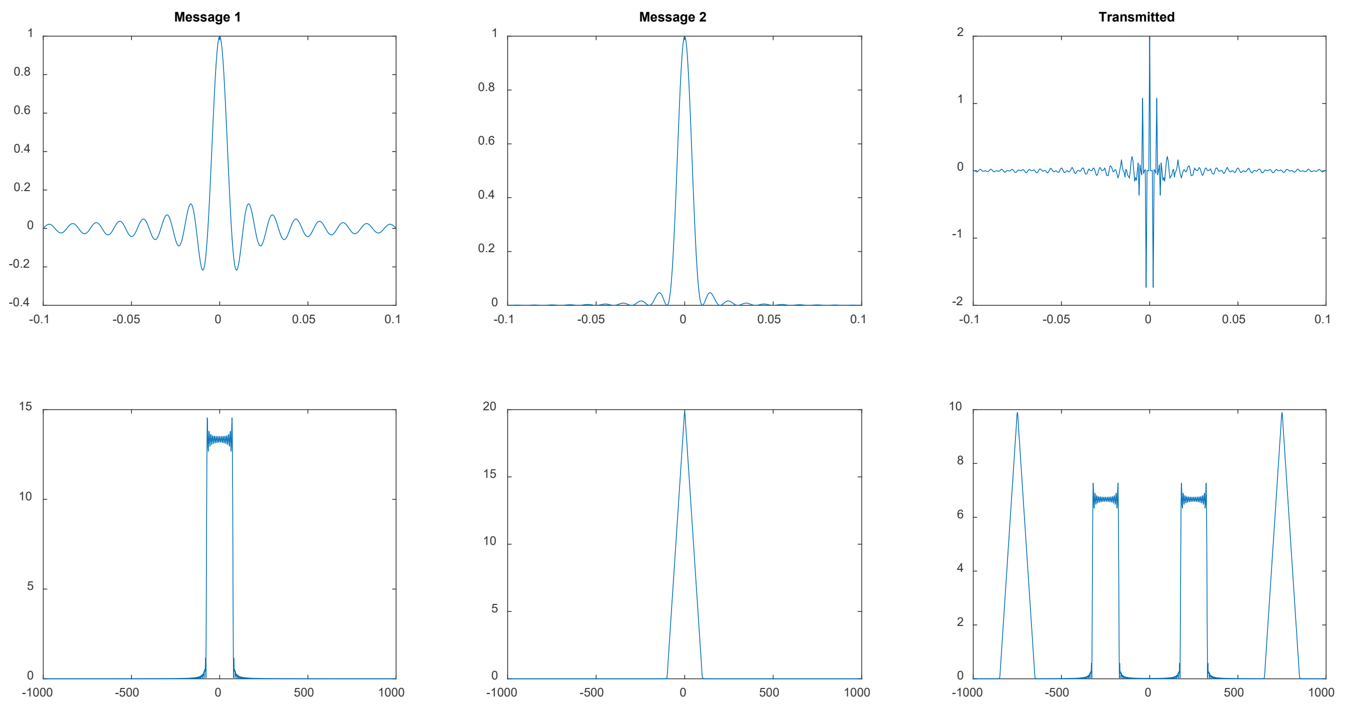


Part C

Simple FDM System

```
Fc1=250;
Fc2=750;
Fs=2000;
t=-0.1:1/Fs:0.1;
message1=sinc(150*t);
message2=sinc(100*t).^2;
carrier1=cos(2*pi*Fc1*t);
carrier2=cos(2*pi*Fc2*t);
transmit=message1.*carrier1+message2.*carrier2;
subplot(2,3,1),plot(t,message1),title('Message 1');
subplot(2,3,2),plot(t,message2),title('Message 2');
subplot(2,3,3),plot(t,transmit),title('Transmitted');
N=1024;
f=(0:1/N:1-1/N)*Fs-Fs/2;
subplot(2,3,4),plot(f,abs(fftshift(fft(message1,N))));
subplot(2,3,5),plot(f,abs(fftshift(fft(message2,N))));
subplot(2,3,6),plot(f,abs(fftshift(fft(transmit,N))));
```

Output:



Discussion:

You can search any function on MATLAB website: https://www.mathworks.com/help/

Documentation links to some important functions are given below:

sinc()	https://www.mathworks.com/help/signal/ref/sinc.html
abs()	https://www.mathworks.com/help/matlab/ref/abs.html
angle()	https://www.mathworks.com/help/matlab/ref/angle.html
exp()	https://www.mathworks.com/help/matlab/ref/exp.html
real()	https://www.mathworks.com/help/matlab/ref/real.html
fft()	https://www.mathworks.com/help/matlab/ref/fft.html
fftshift()	https://www.mathworks.com/help/matlab/ref/fftshift.html

❖ Comment after each output. Discuss why the output is the way it is (in each part). Lastly, write one sentence for each of the seven functions mentioned above.