Bangladesh University of Engineering and Technology

Course Number: EEE 312

Course Title: Digital Signal Processing Laboratory

Experiment Number: 3

Name of the Experiment(s):

Z-transform and its Application

Date of Performance: 18/01/2023

Date of Submission: 29/01/2023

Prepared by:

Tasnimun Razin

Student ID: 1906044

Partner: Tasmin Khan

Student ID: 1906055

Section: A2

Level:3 Term:1

Department: EEE

Problem 1

Question:

1. Consider the systems on the left and answer the questions on the right:

a. $H_1(z) = \frac{4-3z^{-1}}{4-3z^{-1}-z^{-2}}$	(i) Write the time domain <u>causal</u> impulse responses of the given systems using `residuez()`.
b. $H_2(z) = \frac{4+3z^{-1}}{4-3z^{-1}+z^{-2}}$	(ii) Use ***fvtool*** to show the magnitude and phase responses of the systems.
c. $H_3(z) = \frac{1-z^{-1}}{1-\frac{13}{12}z^{-1}+\frac{3}{8}z^{-2}-\frac{1}{24}z^{-3}}$	(iii) Identify the unstable system/s and show a method to stabilize its/their response.
d. $H_4(z) = \frac{1-z^{-1}}{1-\frac{13}{6}z^{-1}+\frac{7}{8}z^{-2}-\frac{1}{24}z^{-3}}$	

Here, the main code is same for all the four systems. Only the values of the coefficient matrices need to be changed. We, hence, will be showing the code for only system a.

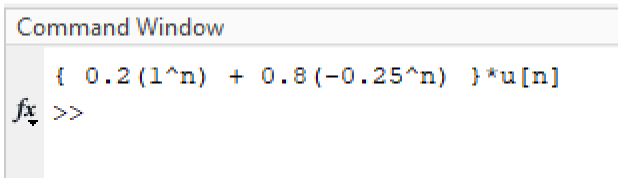
Main Code:

```
clc;
clear all;
close all;

%% problem i
%Generating Transfer Function
b = [4 -3];        %for zeros
a = [4 -3 -1];     %for poles
[r,p,c] = residuez(b,a);

%time domain expression
pstr = string(p);
rstr = string(r);
m = '';
for i=1:length(pstr)
        m = m + rstr(i) + '('+pstr(i)+'^n'+')' + ' + ';
end
m=char(m);
m(find(m=='+',1,'last'):end)=[];
m = '{ ' + string(m) + '}*u[n]';
disp(m)

%Pole-Zero Plot
figure(1)
zplane(b,a);
```

```
%Plotting Causal Impulse Response
n=-10:20;
figure(2)
impz(b,a,n);
xlabel('Sample Number')
ylabel('Amplitude')
title('Causal Impulse Response')

%% Problem ii
%Magnitude & Phase Response
fvtool(b,a);

%% Problem iii
%Stabilizing the Unstable System
bnew=b;
for j=1:length(p)
    if (abs(p(j))) >= 1
        bnew = conv(bnew,[1,-p(j)]);
    end
end
[rnew,pnew,c] = residuez(bnew,a);

figure(3)
zplane(bnew,a)
figure(4)
impz(bnew,a)
```

Problem a

Command window output:

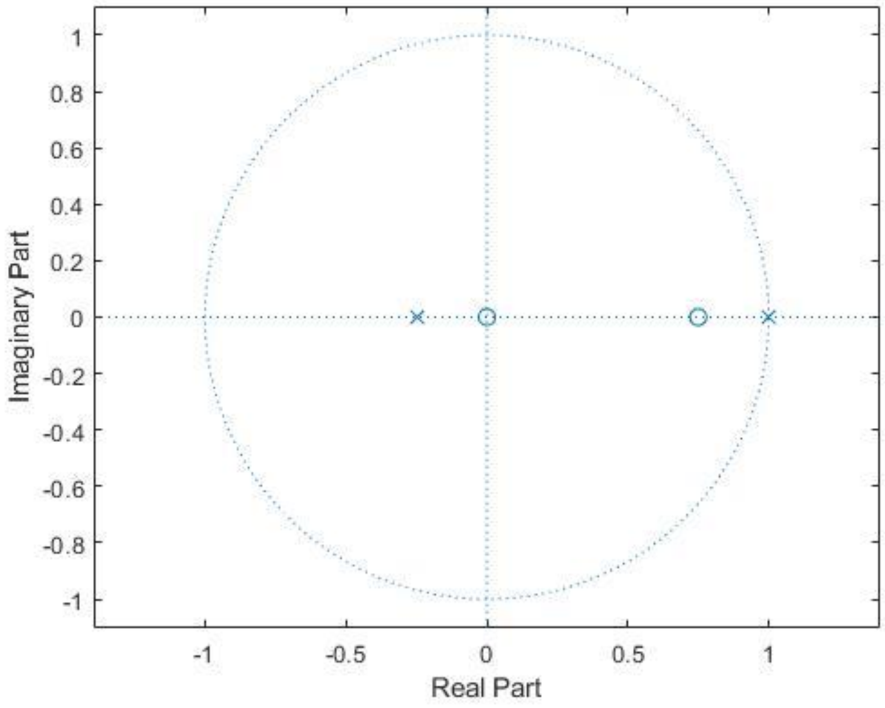


```
{ 0.2(1^n) + 0.8(-0.25^n) }*u[n]
>>
```

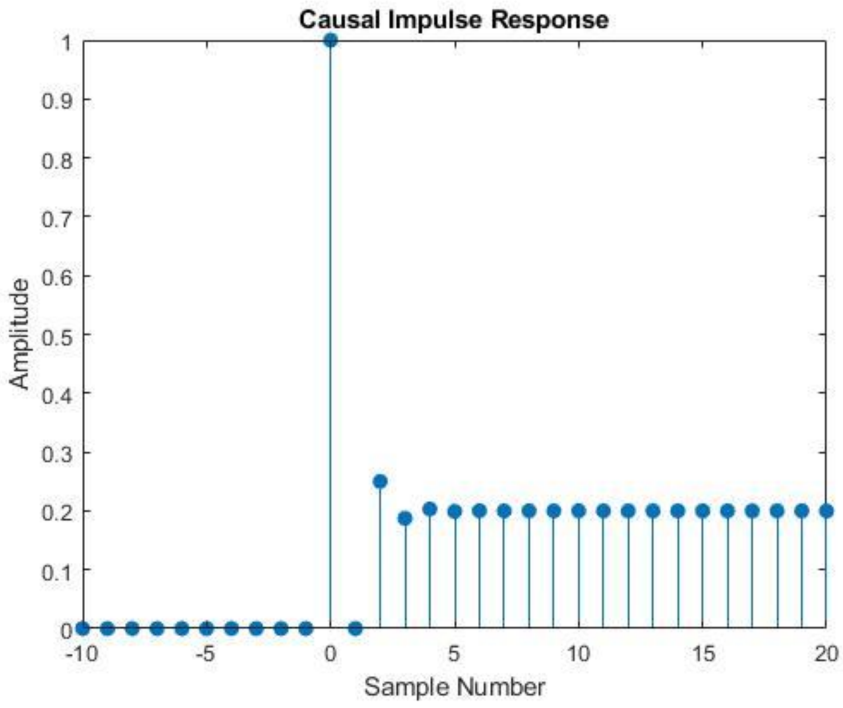
We can see the time (n) domain expression here in the command window.

To create the expression we used characters and strings.

Plot:



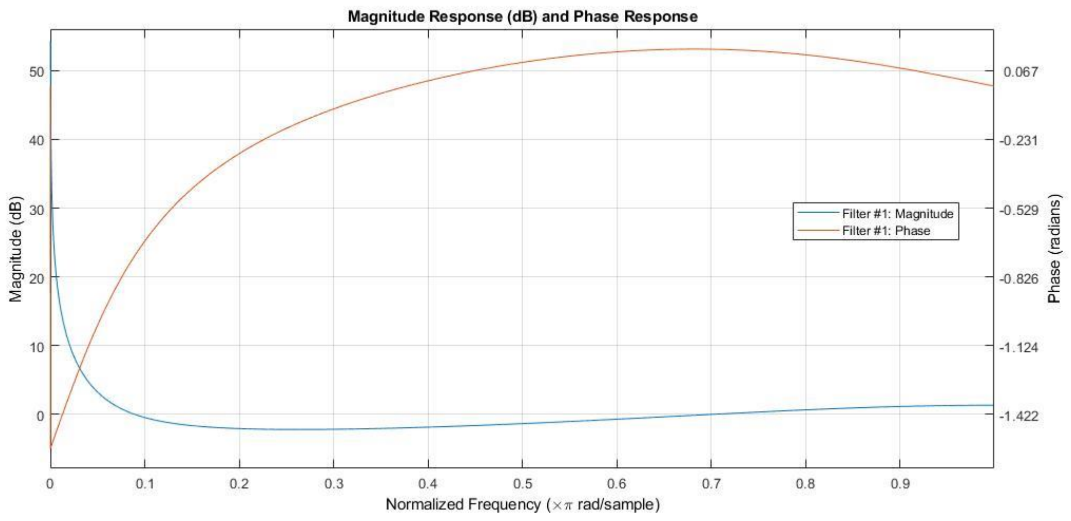
Imaginary Part
Real Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1



Causal Impulse Response
Amplitude
Sample Number
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-10
-5
0
5
10
15
20

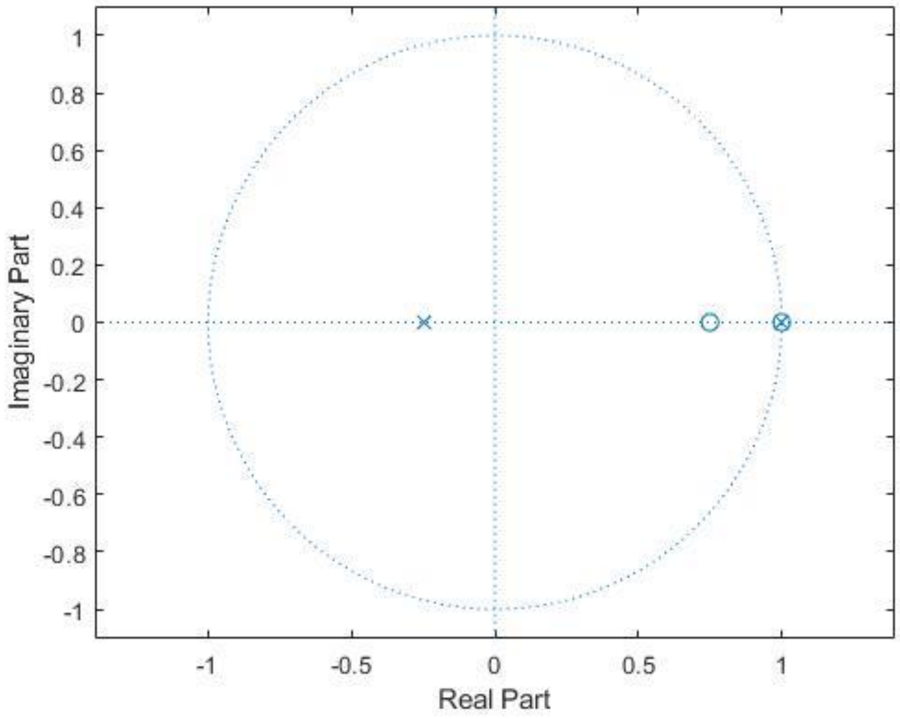
We can see that one of the poles is on the unit circle. Hence, the causal function will be marginally stable. Our impulse response shows the marginal stability as it hold on to a constant value.

The Phase and magnitude responses are as follows:

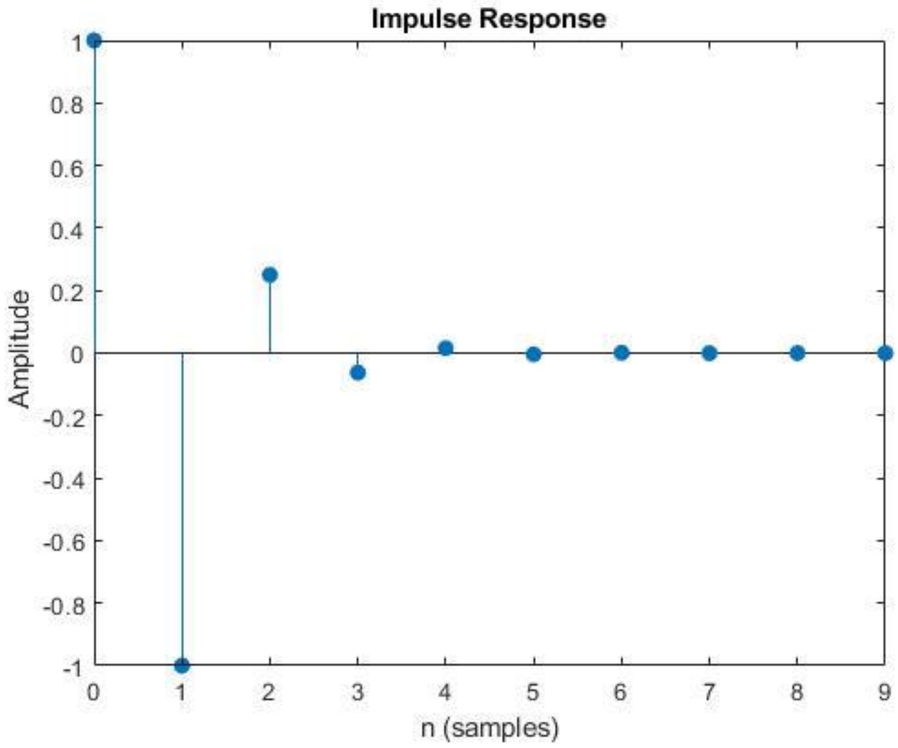


We need to remove any pole outside or on the unit circle to make the system function stable. For this, we multiplied the given system function with the unwanted pole factor to achieve new set of nominator coefficients. Now, multiplying two polynomials is equivalent to convoluting the coefficient sets. We used this property to gain out new set bnew.

Then as expected, we can see the unwanted pole is cancelled on the pole-zero plot by another zero. And the impulse response dies down making the system stable.



Imaginary Part
Real Part



Impulse Response
Amplitude
n (samples)

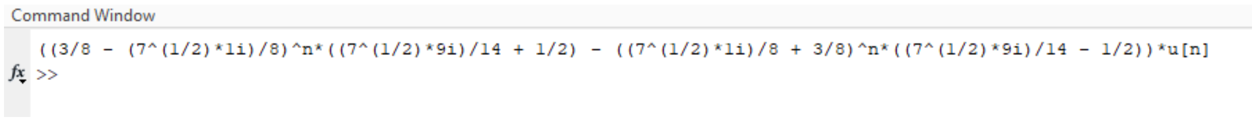
Problem b

We needed to make a slight modification her to get the time domain expression with complex poles. The rest was unchanged.

Alternative code for expression:

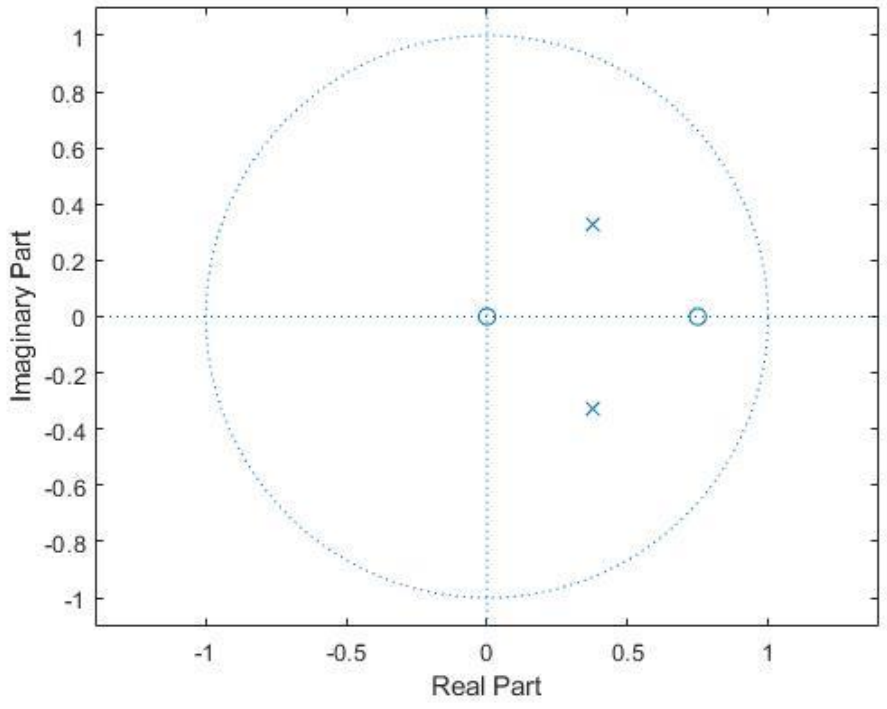
```
syms x y j k m n
x = p(1);
y = p(2);
j = R(1);
k = R(2);
h = (j*x^n + k*y^n);
m ='('+ string(h) + ')*u[n]';
disp(m)
```

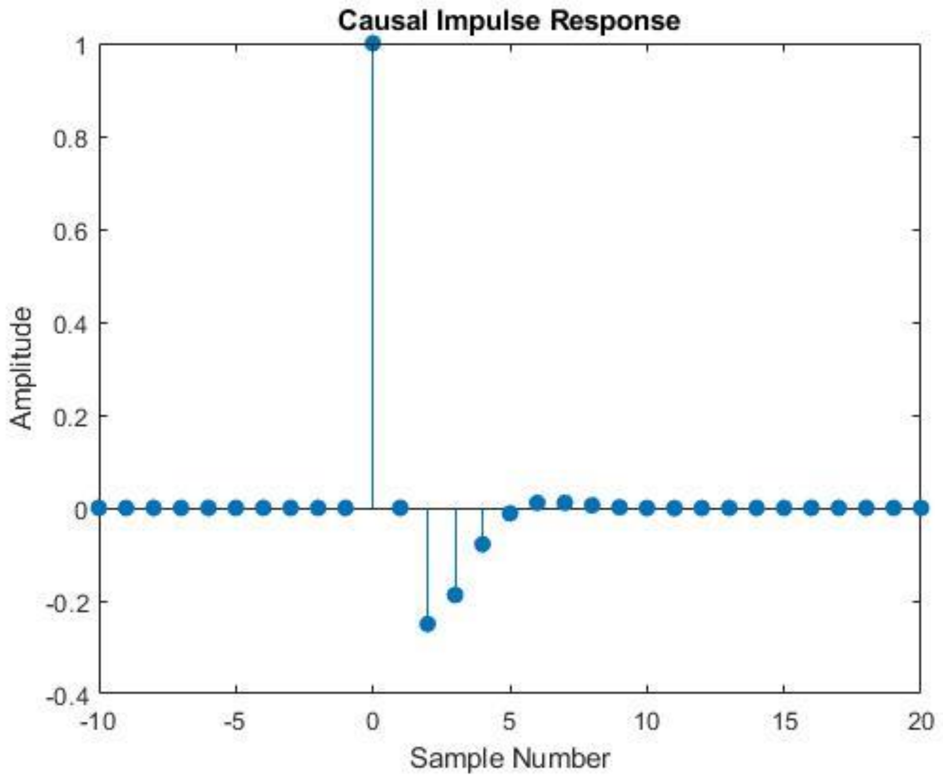
Command window output:



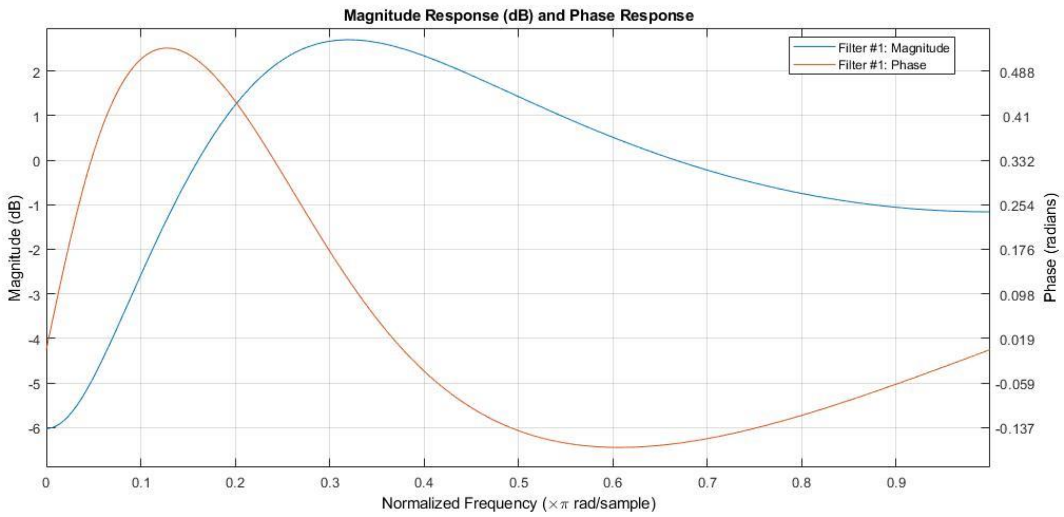
```
((3/8 - (7^(1/2)*1i)/8)^n*((7^(1/2)*9i)/14 + 1/2) - ((7^(1/2)*1i)/8 + 3/8)^n*((7^(1/2)*9i)/14 - 1/2))*u[n]
>>
```

Plot:



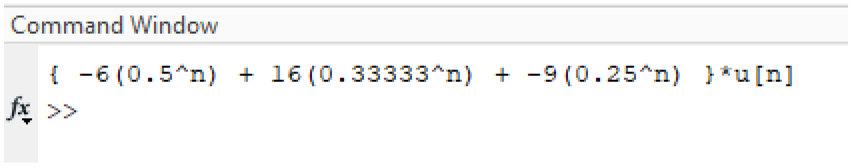


We can see that this system is already stable since the poles are all inside the unit circle. The impulse response dies down too The Phase and magnitude responses are as follows:



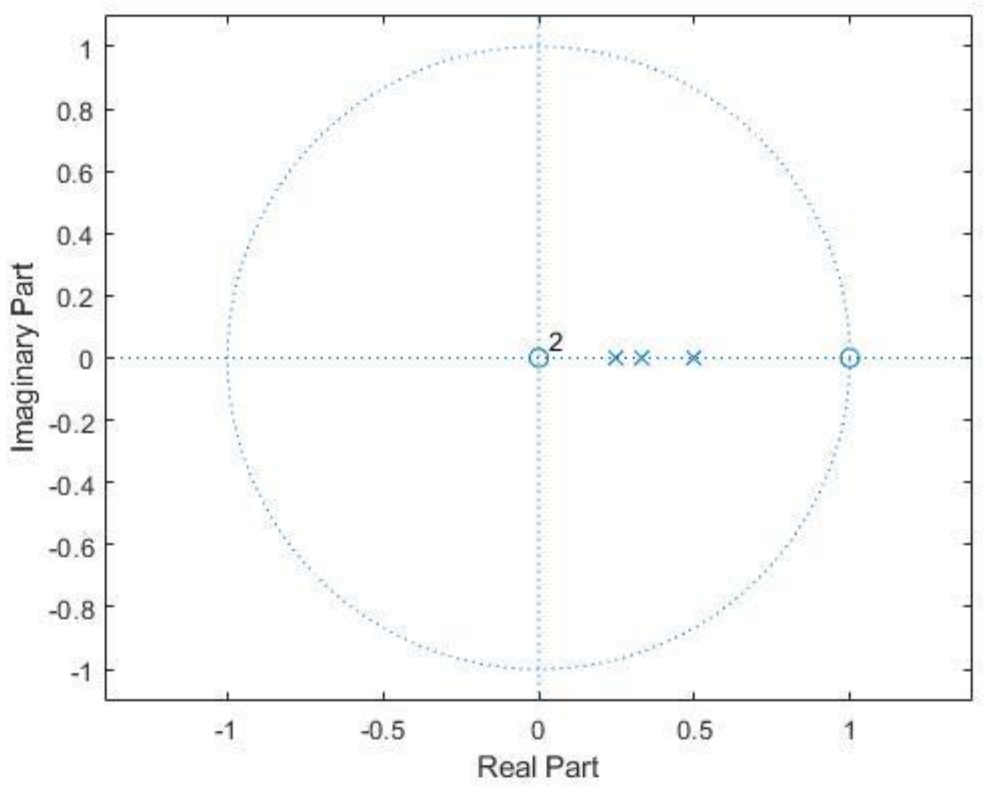
Problem c

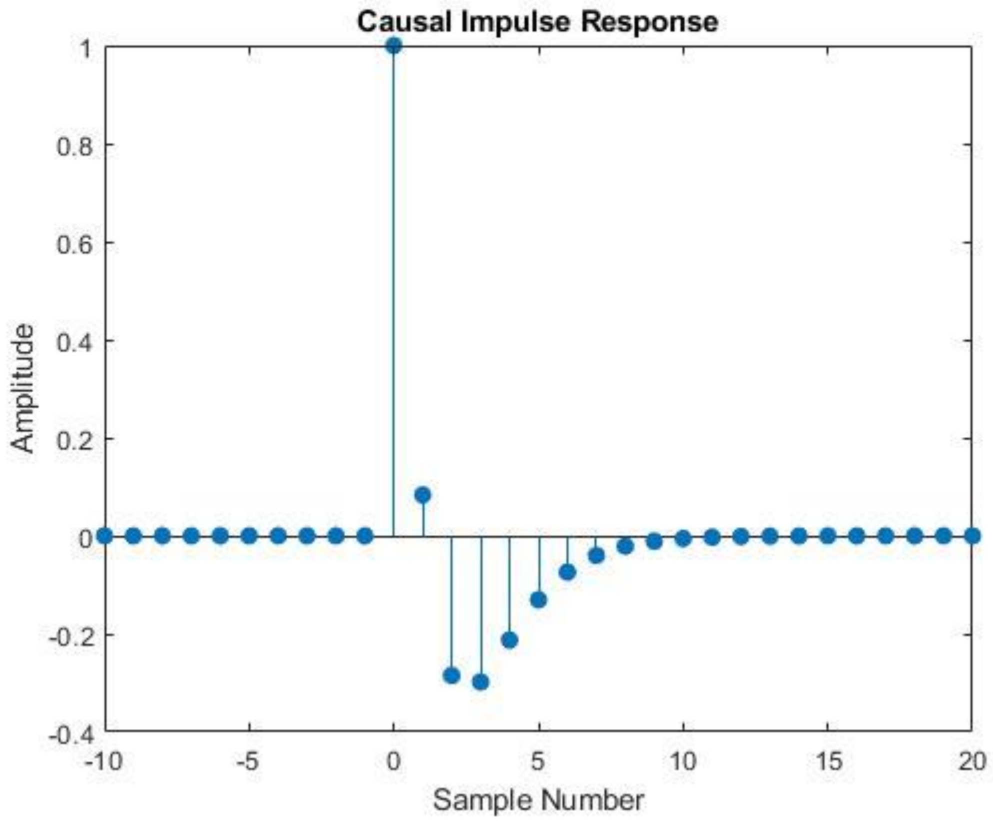
Command window output:



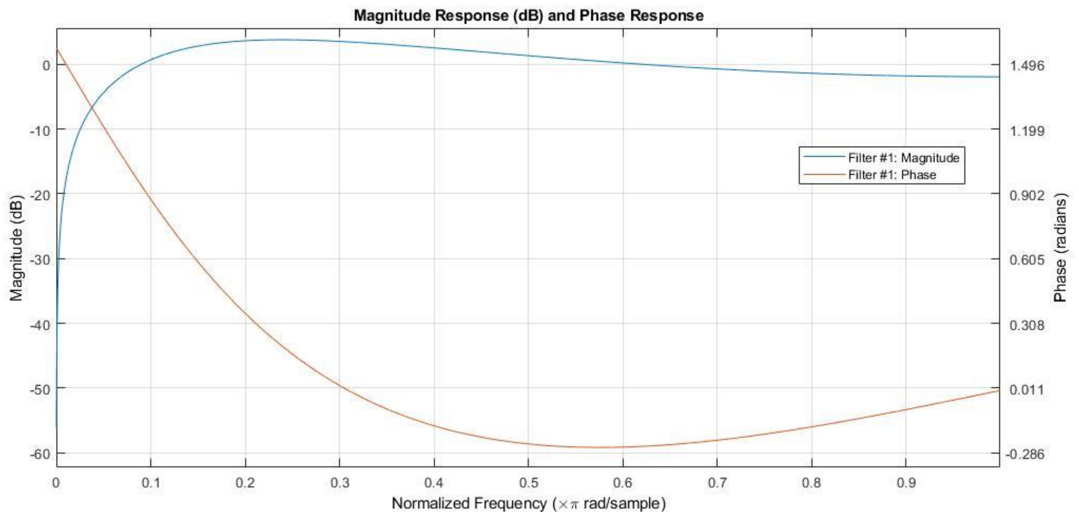
```
Command Window
 { -6(0.5^n) + 16(0.33333^n) + -9(0.25^n) }*u[n]
fx >>
```

Plot:



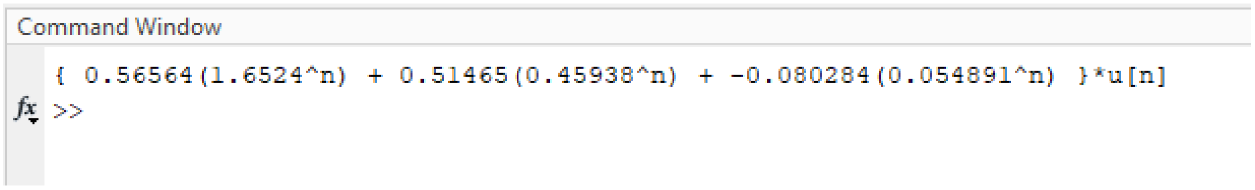


We can see that this system is already stable since the poles are all inside the unit circle. The impulse response dies down too. The Phase and magnitude responses are as follows:



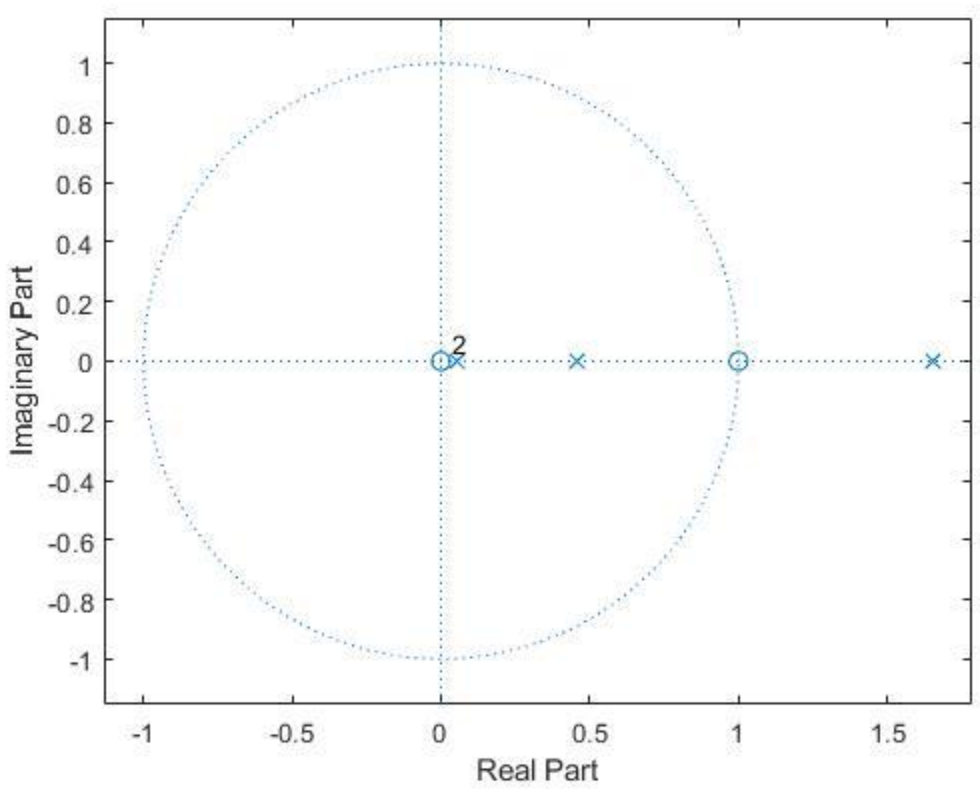
Problem d

Command window output:

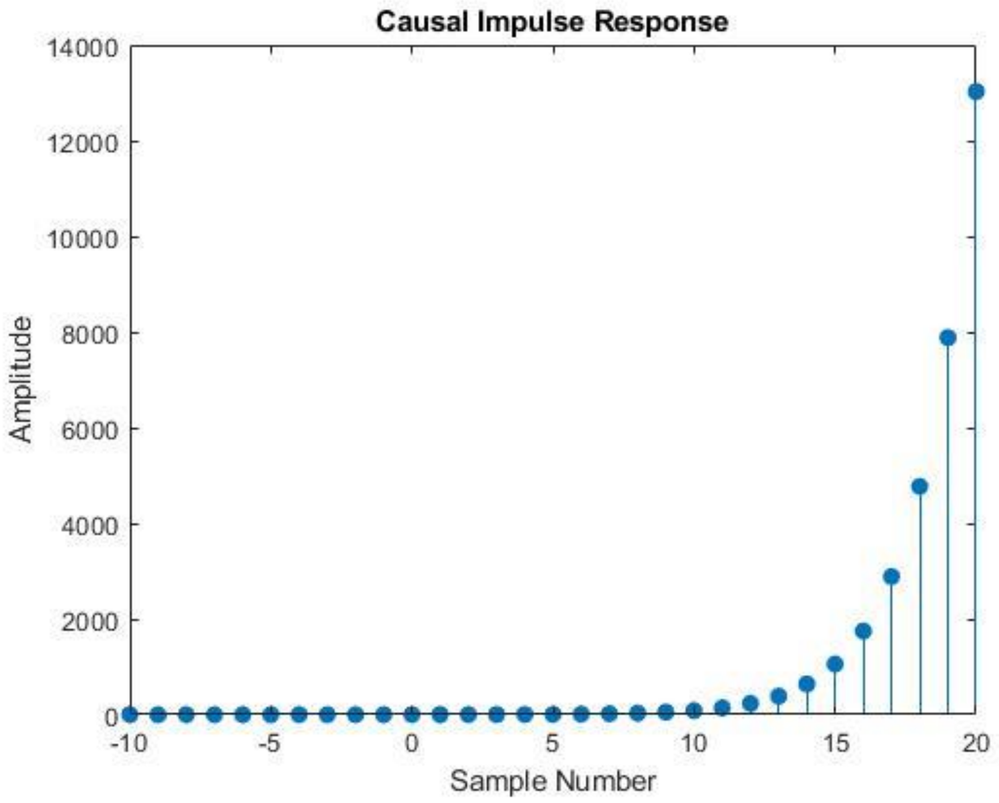


```
Command Window
 { 0.56564(1.6524^n) + 0.51465(0.45938^n) + -0.080284(0.054891^n) }*u[n]
fx >>
```

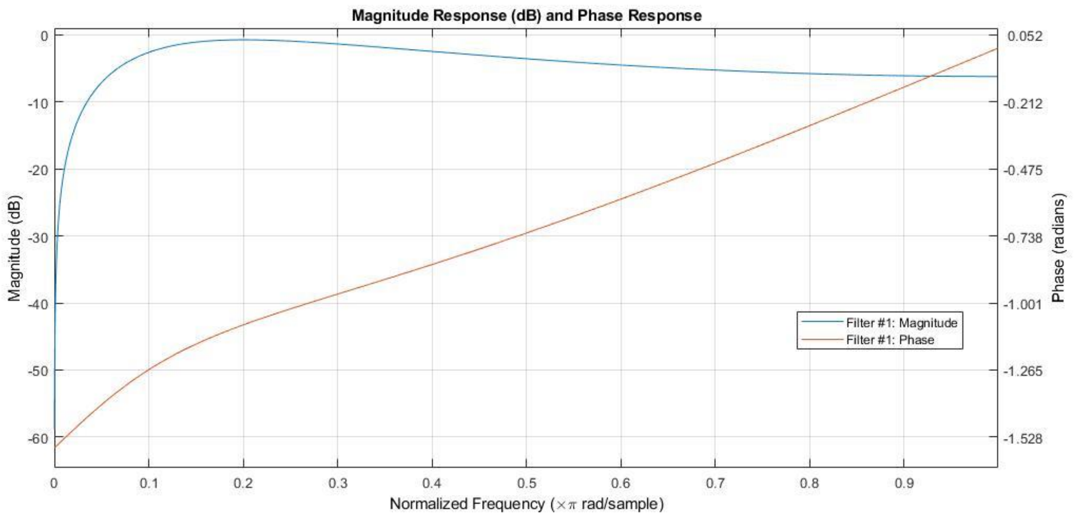
Plot:

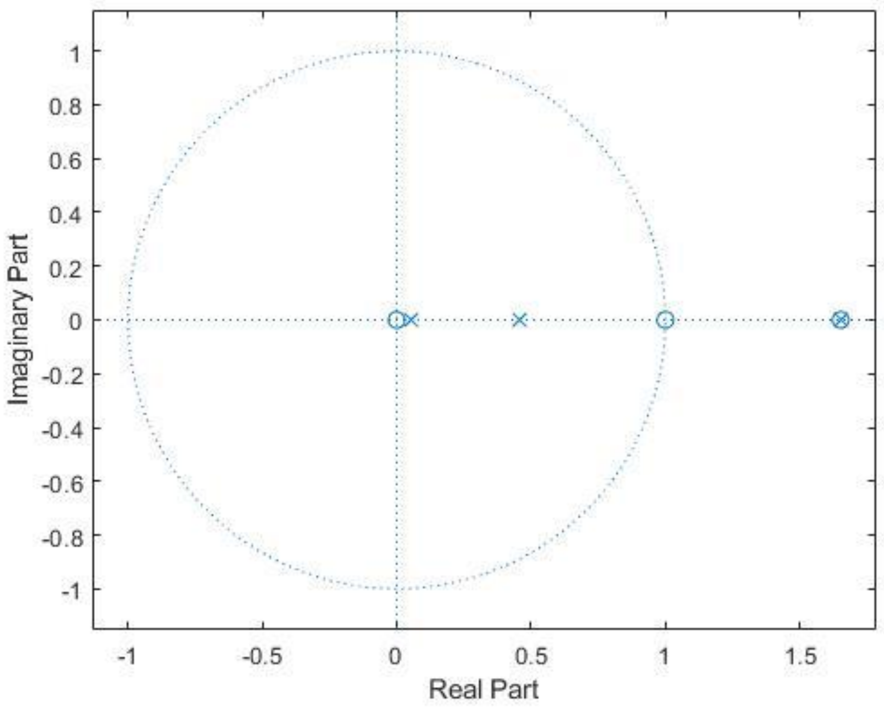


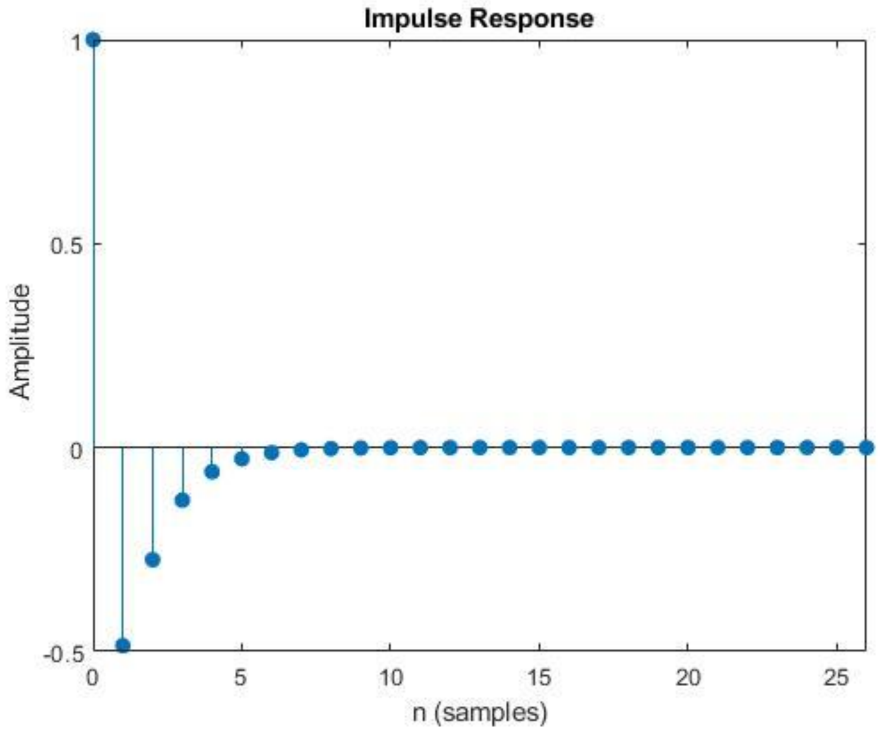
We can see that this system is not stable since there is a pole outside the unit circle. The impulse response rises too indicating an unstable system. Hence, we will need to make the system stable like earlier.



The Phase and magnitude responses are as follows:







as expected, we can see the unwanted pole is cancelled on the pole-zero plot by another zero. And the impulse response dies down making the system stable.

Problem 2

Question:

2. Determine and write the expression for each of the $H_{eq}(z)$ below,

(i) $Y(z) = H_{eq}(z).X(z) = [H_1(z).H_3(z)].X(z)$

(ii) $Y(z) = H_{eq}(z).X(z) = [H_2(z).H_4(z)].X(z)$

(iii) $Y(z) = H_{eq}(z).X(z) = [H_1(z) + H_2(z)].X(z)$

(iv) $Y(z) = H_{eq}(z).X(z) = [H_1(z) + z^{-2}H_2(z)].X(z)$

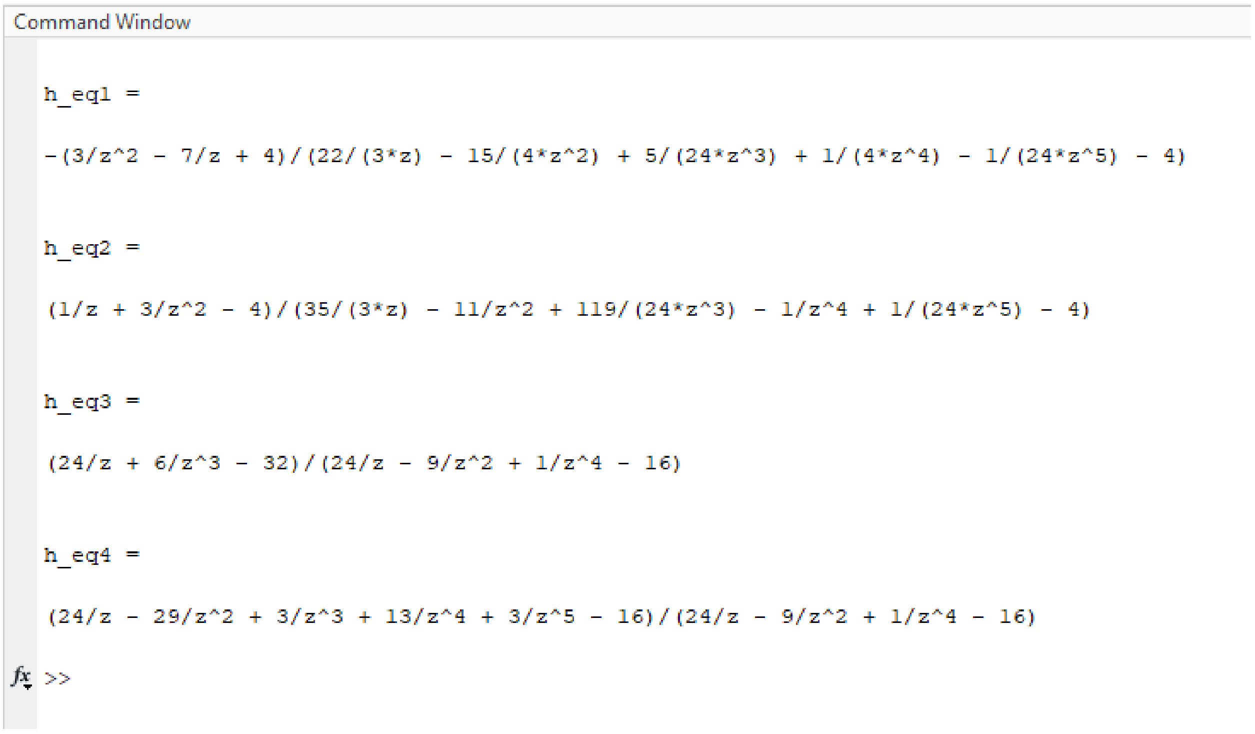
Here, the systems denoted by $H_k(z), k \in \{1,2,3,4\}$ are defined in the table of **Q-1**.

Code:

```
clc;
clear all;
close all;

%Generating Transfer Function
b1=[4 -3];
a1=[4 -3 -1];

b2=[4 3];
a2=[4 -3 1];

b3=[1 -1];
a3=[1 -13/12 3/8 -1/24];

b4=[1 -1];
a4=[1 -13/6 7/8 -1/24];

syms z h_eq1 h_eq2 h_eq3 h_eq4

%% problem i
b_1 = conv(b1,b3); %length(b_1) = {length(b1)+length(b3)}-1 = 3
a_1 = conv(a1,a3); %length(a_1) = {length(a1)+length(a3)}-1 = 6

h_eq1=(b_1(1)+b_1(2)*z^-1+b_1(3)*z^-2)/(a_1(1)+a_1(2)*z^-1+a_1(3)*z^-2+a_1(4)*z^-
3+a_1(5)*z^-4+a_1(6)*z^-5)

%% problem ii
b_2 = conv(b2,b4); %length(b_2) = {length(b2)+length(b4)}-1 = 3
a_2 = conv(a2,a4); %length(a_2) = {length(a2)+length(a4)}-1 = 6

h_eq2=(b_2(1)+b_2(2)*z^-1+b_2(3)*z^-2)/(a_2(1)+a_2(2)*z^-1+a_2(3)*z^-2+a_2(4)*z^-
3+a_2(5)*z^-4+a_2(6)*z^-5)
```

```
%% problem iii
b_3_1 = conv(b1,a2); %length(b_3_1) = {length(b1)+length(a2)}-1 = 4
b_3_2 = conv(b2,a1); %length(b_3_2) = {length(b2)+length(a1)}-1 = 4
b_3 = b_3_1 + b_3_2; %length(b_3) = length(b_3_1)=length(b_3_2) = 4
a_3 = conv(a1,a2);   %length(a_3) = {length(a1)+length(a2)}-1 = 5

h_eq3=(b_3(1)+b_3(2)*z^-1+b_3(3)*z^-2+b_3(4)*z^-3)/(a_3(1)+a_3(2)*z^-1+a_3(3)*z^-
2+a_3(4)*z^-3+a_3(5)*z^-4)

%% problem iv
b_4_zinv2 = [0 0 4 3];
b_4_1 = conv(b1,a2);          %length(b_4_1) = {length(b1)+length(a2)}-1 = 4
b_4_2 = conv(b_4_zinv2,a1); %length(b_4_2) = {length(b_4-zinv2)+length(a1)}-1 = 6
b_4_1 = [b_4_1 0 0];          %to equalize length(b_4_2)
b_4 = b_4_1 + b_4_2;          %length(b_4) = length(b_4_1)=length(b_4_2) = 6
a_4 = conv(a1,a2);            %length(a_3) = {length(a1)+length(a2)}-1 = 5

h_eq4=(b_4(1)+b_4(2)*z^-1+b_4(3)*z^-2+b_4(4)*z^-3+b_4(5)*z^-4+b_4(6)*z^-
5)/(a_4(1)+a_4(2)*z^-1+a_4(3)*z^-2+a_4(4)*z^-3+a_4(5)*z^-4)
```

Command window:


Command Window

```
h_eq1 =

-(3/z^2 - 7/z + 4)/(22/(3*z) - 15/(4*z^2) + 5/(24*z^3) + 1/(4*z^4) - 1/(24*z^5) - 4)


h_eq2 =

(1/z + 3/z^2 - 4)/(35/(3*z) - 11/z^2 + 119/(24*z^3) - 1/z^4 + 1/(24*z^5) - 4)


h_eq3 =

(24/z + 6/z^3 - 32)/(24/z - 9/z^2 + 1/z^4 - 16)


h_eq4 =

(24/z - 29/z^2 + 3/z^3 + 13/z^4 + 3/z^5 - 16)/(24/z - 9/z^2 + 1/z^4 - 16)

>>
```

Discussion:

As we can see the system expressions are given in the command window. We used convolution to gain the multiplication of polynomials. We also did necessary padding to equalize the dimensions of matrices.

Problem 3

Question:

3. An input signal $x(n)$ is provided to a system and the obtained output is $y(n)$:

$$x(n) = \left(\frac{1}{2}\right)^n u(n) - \left(\frac{3}{4}\right)^{n-1} u(n-1)$$

$$y(n) = 3\left(\frac{1}{2}\right)^n u(n)$$

For which type of causality is the system stable? (i) Write the analytic expression of that stable system, (ii) plot the time domain response of it.

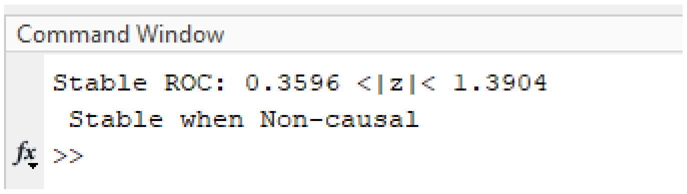
Code:

```
clc;
clear all;

%Generating Input =  X(z)
b_x1 = [1];
a_x1 = [1 -1/2];
b_x2 = [0 1];
a_x2 = [1 -3/4];

b_1 = conv(b_x1,a_x2); %length(b_1) = {length(b_x1)+length(a_x2)}-1 = 2
b_2 = conv(b_x2,a_x1); %length(b_2) = {length(b_x2)+length(a_x1)}-1 = 3
b_1 = [b_1 0];         %to equalize length(b_2)
b_x = b_1 - b_2;       %length(b_x) = {length(b_1)=length(b_2)}-1 = 3
a_x = conv(a_x1,a_x2); %length(a_x) = {length(a_x1)+length(a_x2)}-1 = 3

%Generating Output = Y(z)
b_y = [3];
a_y = [1 -1/2];

%Generating Transfer Function = H(Z) = Y(z)/X(z)
b_h = conv(b_y,a_x); %length(b_h) = {length(b_y)+length(a_x)}-1 = 3
a_h = conv(a_y,b_x); %length(b_1) = {length(b_x2)+length(a_x2)}-1 = 4
```

```
%Pole-Zero Plots
figure(1)
zplane(b_x,a_x);
title('Pole Zero Plot of Input Function')
figure(2)
zplane(b_y,a_y);
title('Pole Zero Plot of Output Function')
figure(3)
zplane(b_h,a_h);
title('Pole Zero Plot of Transfer Function')

%i
%Finding Stable ROC
 [r,p,c] = residuez(b_h,a_h);

%defining u(n) and u(n-1)
n = -20:20;
u=1.*[n>=0];
un1=1.*[-n-1>=0];


%if r->0, eliminating the pole
for i=1:length(r)
   if r(i)<1e-10
       p(i)=0;
   end
end

%if all poles are inside unit circle |z|=1
if (max(abs(p))<1)
    c = 1;
    fprintf('Stable ROC: |z|> %.4f \n',max(p));

%if all poles are outside unit circle |z|=1
elseif (min(abs(p)>1))
    fprintf('Stable ROC: |z|< %.4f \n',min(p));
    c = 2;

%if poles are both inside & outside unit circle |z|=1
else
    left = min(abs(p));
    right = max(abs(p));

    for i=1:length(p)
        if(abs(p(i))<1 && abs(p(i))>left)
            left = abs(p(i));
        elseif(abs(p(i))>1 && abs(p(i))<right )
            right = abs(p(i));
        end
    end
    fprintf('Stable ROC: %.4f <|z|< %.4f \n ',left,right);
    c = 3;
end
```

```
%ii
%Time Domain Output
if c==1
    fprintf('Stable when Causal\n'); %RS Sequence
elseif c==2
    fprintf('Stable when anti-causal\n'); %LS Sequence
else
    fprintf('Stable when Non-causal\n'); %TS Sequence
end

figure(4)
    h_n = zeros(1,length(n));
    for k=1:length(p)
        h_n = h_n + (r(k))*(p(k)).^n.*u; %RS Sequence
    end
subplot(311)
stem(n,h_n);
title('Time Domain Response for RS sequence');
    h_n = zeros(1,length(n));
    for k=1:length(p)
        h_n = h_n - (r(k))*(p(k)).^n.*un1; %LS Sequence
    end
subplot(312)
stem(n,h_n);
title('Time Domain Response for LS sequence');
    h_n = zeros(1,length(n));
    for k=1:length(p)
        if p(k)==0
            continue;
        elseif abs(p(k))<1
            h_n = h_n+(r(k))*(p(k)).^n.*u; %RS Sequence
        else
            h_n = h_n - (r(k))*(p(k)).^n.*un1; %LS Sequence
        end
    end
subplot(313)
stem(n,h_n);
title('Time Domain Response for TS sequence');
```

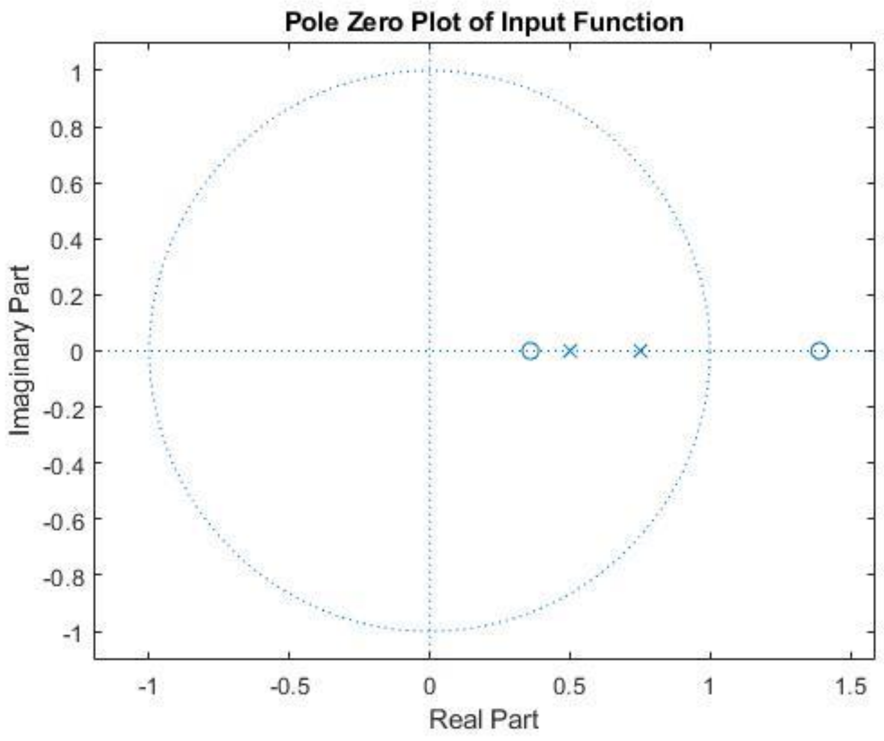
Command window:

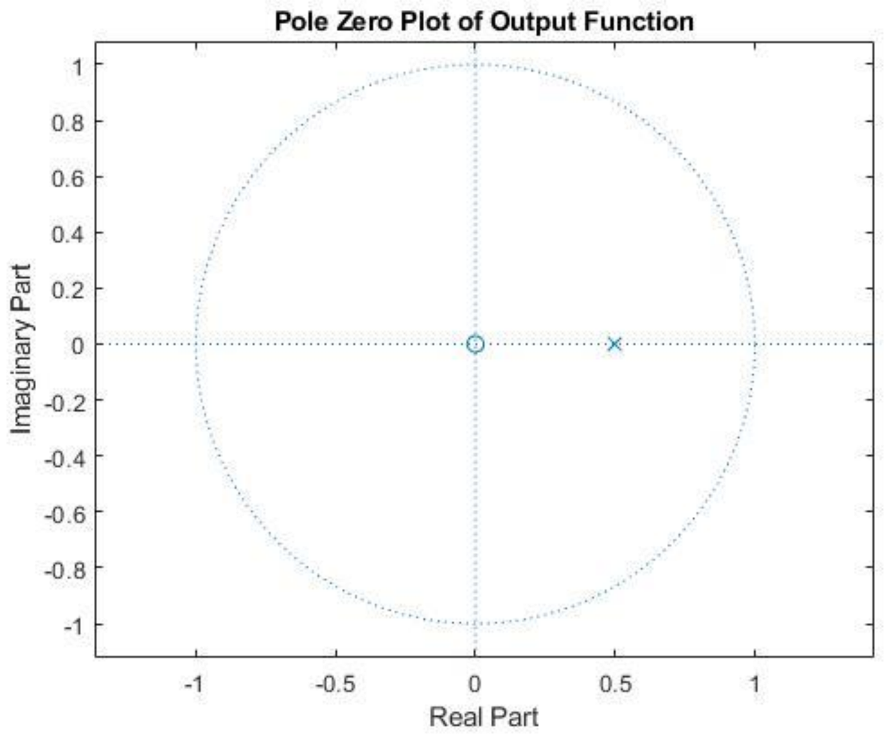


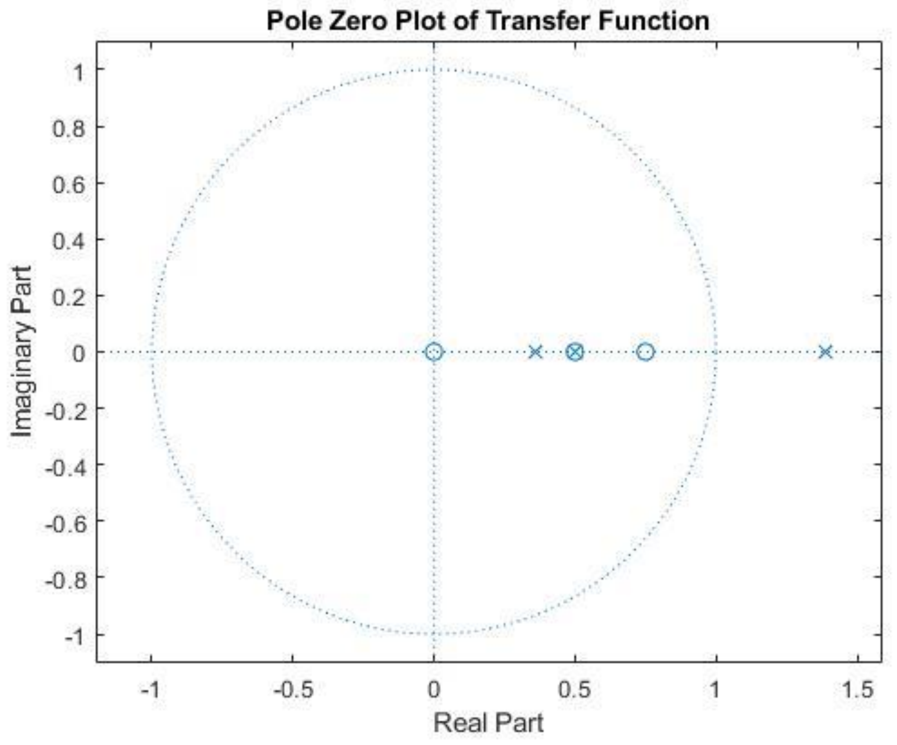
Command Window

```
Stable ROC: 0.3596 <|z|< 1.3904
 Stable when Non-causal
>>
```

Plot:


Pole Zero Plot of Input Function
Imaginary Part
Real Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1
1.5


Pole Zero Plot of Output Function
Imaginary Part
Real Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1



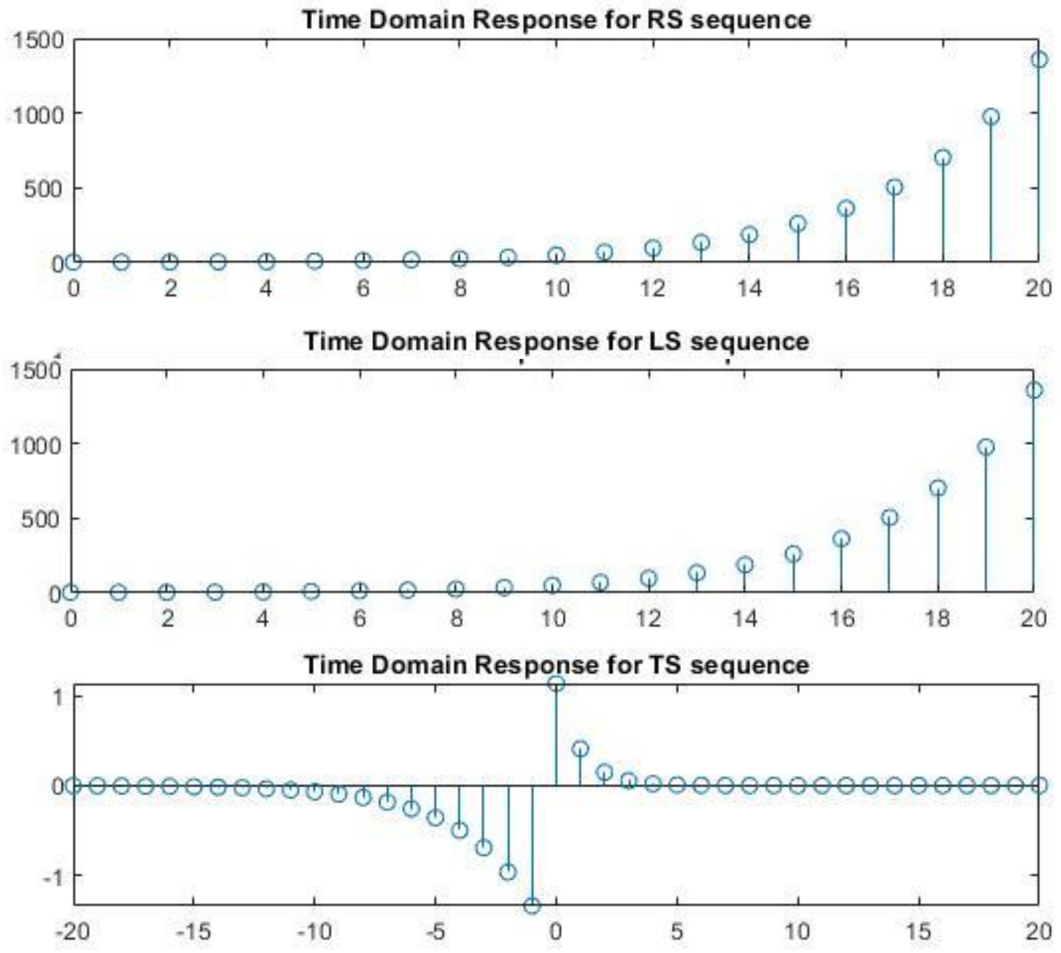
As we can see there is are two valid poles and one cancelled pole. For the two poles we can get three regions as ROC.

When we select the right sided region, our system becomes causal. Since it does not contain the unit circle, we get an unstable system.

When we select the left sided region, our system becomes anti causal and for similar reasons, we get an unstable system.

When we select the two sided region, our system becomes non causal. This ROC includes the unit circle and hence we get a stable system.

We can see the stability being obvious in the impulse response plots on the next page.


Time Domain Response for RS sequence
Time Domain Response for LS sequence
Time Domain Response for TS sequence

Problem 4

Question:

4. The system function of a 4-th order Butterworth low pass filter (LPF) is provided below:

$$H_{butter}(z) = \frac{0.0102 + 0.0408z^{-1} + 0.0613z^{-2} + 0.0408z^{-3} + 0.0102z^{-4}}{1 - 1.9684z^{-1} + 1.7359z^{-2} - 0.7245z^{-3} + 0.1204z^{-4}}$$

Using only `impz(b,a,n=50)`, find the outputs $Y(z)$ of the following inputs:

$\cos(0.2\pi n)u(n)$	$\cos(0.3\pi n)u(n)$	$\cos(0.4\pi n)u(n)$
$\cos(0.5\pi n)u(n)$	$\cos(0.6\pi n)u(n)$	$\cos(0.7\pi n)u(n)$

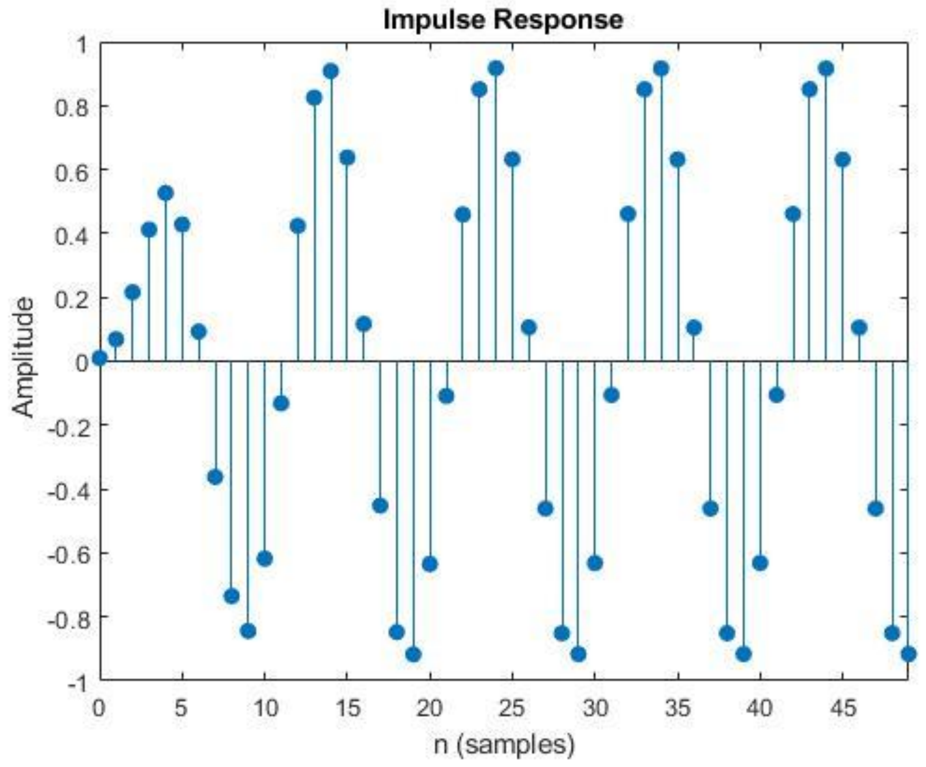
Comment on which inputs were not effectively passed by the LPF. (Also, in `zplane`, observe each pole-zero of $Y(z)$ as your input frequency increases. Only some poles change positions. Can you explain the reason behind it?)

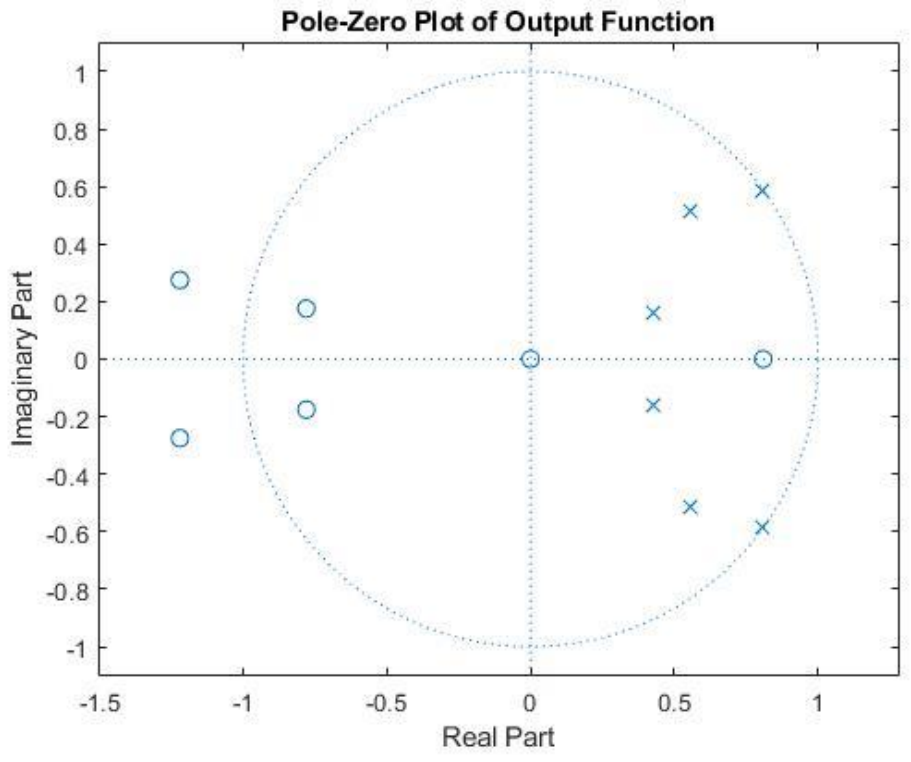
Code:

```
clc;
clear all;
close all;

%Generating Transfer Function => H(z)
b_h = [0.0102 0.0408 0.0613 0.0408 0.0102];
a_h = [1 -1.9684 1.7359 -0.7245 0.1204];

%generating Input => X(z)
w = input('Please Enter the value of w: '); %Variable Phase
b_x = [1 -cos(w)];
a_x = [1 -2*cos(w) 1];

%Generating Output => Y(z)=X(z)*H(z)
b_y = conv(b_x,b_h);
a_y = conv(a_x,a_h);

%Plotting
n = 50;
figure(1)
impz(b_y,a_y,n);

figure(2)
zplane(b_y,a_y);
title('Pole-Zero Plot of Output Function')
```

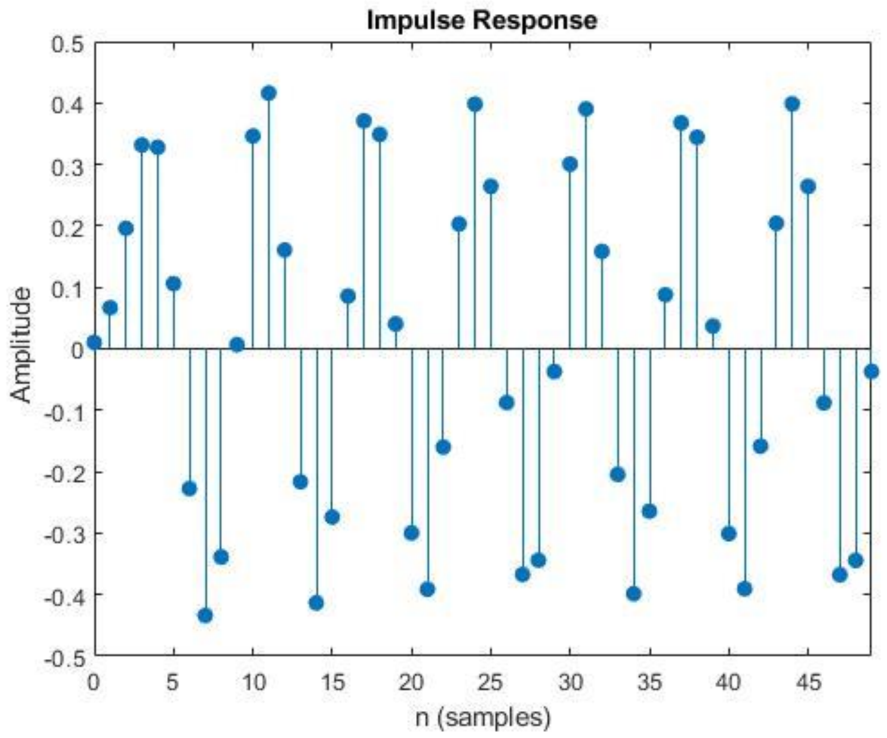
Plot:

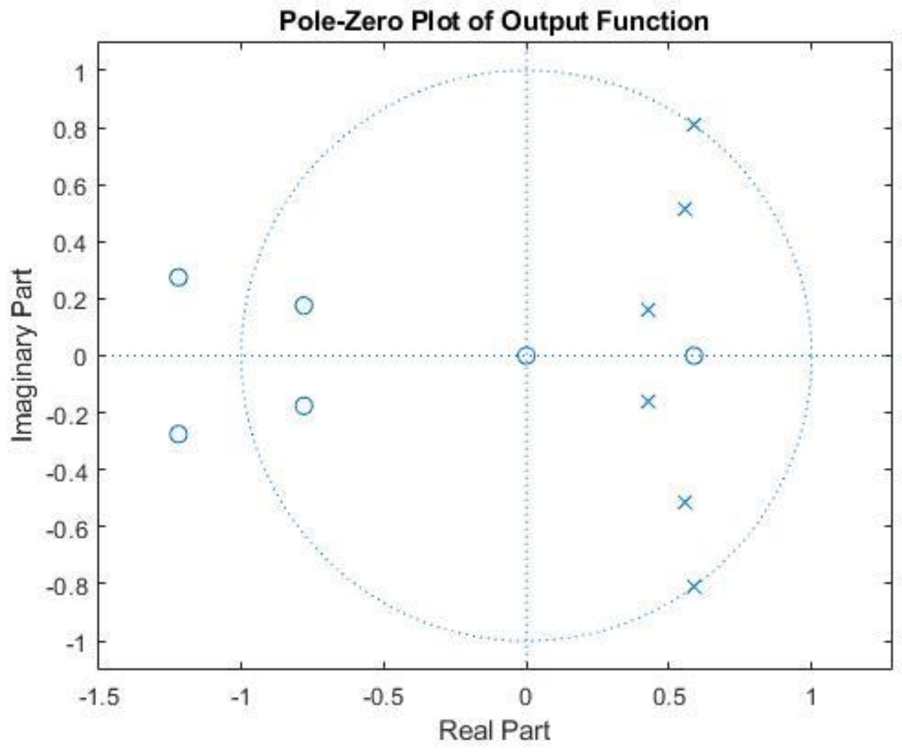
For $\omega = 0.2\pi$



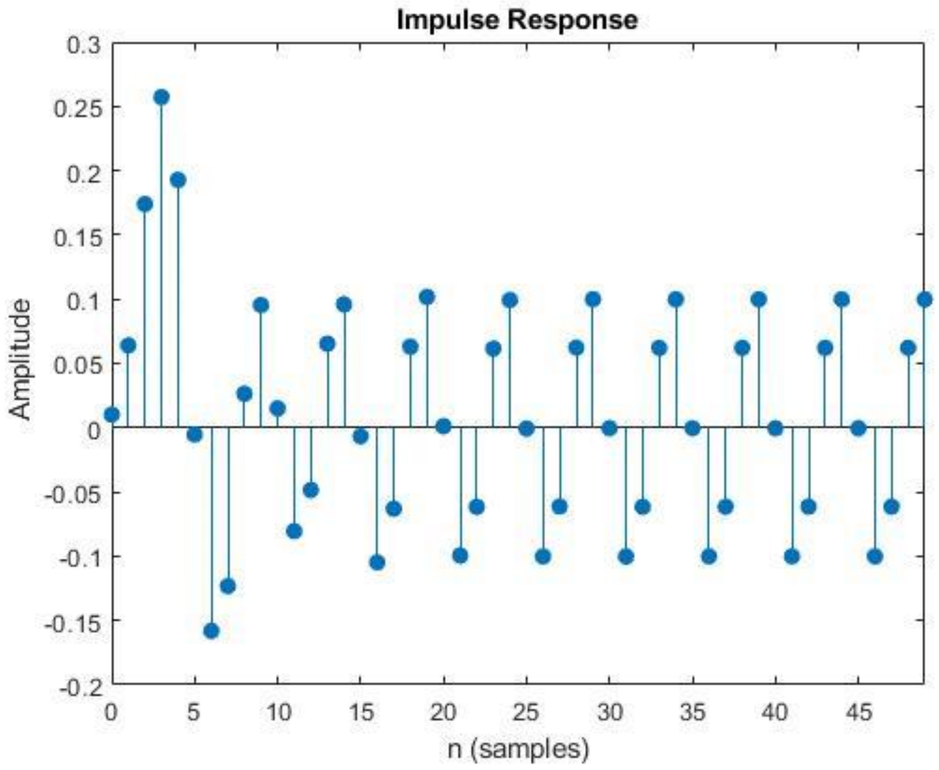


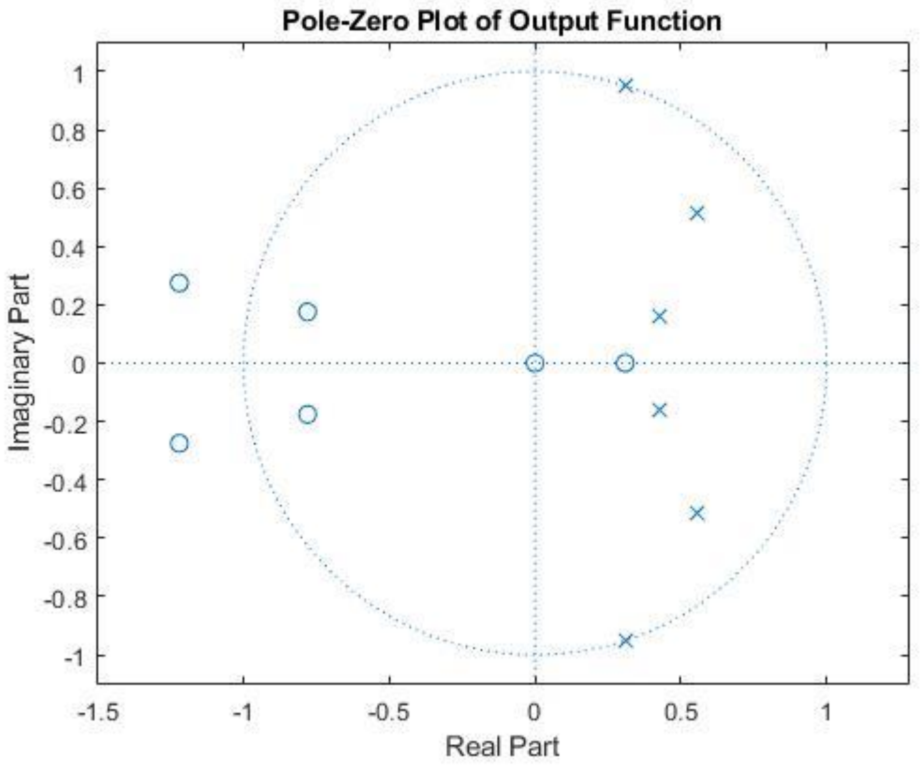
For ω = 0.3π


Impulse Response
Amplitude
n (samples)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
0
5
10
15
20
25
30
35
40
45

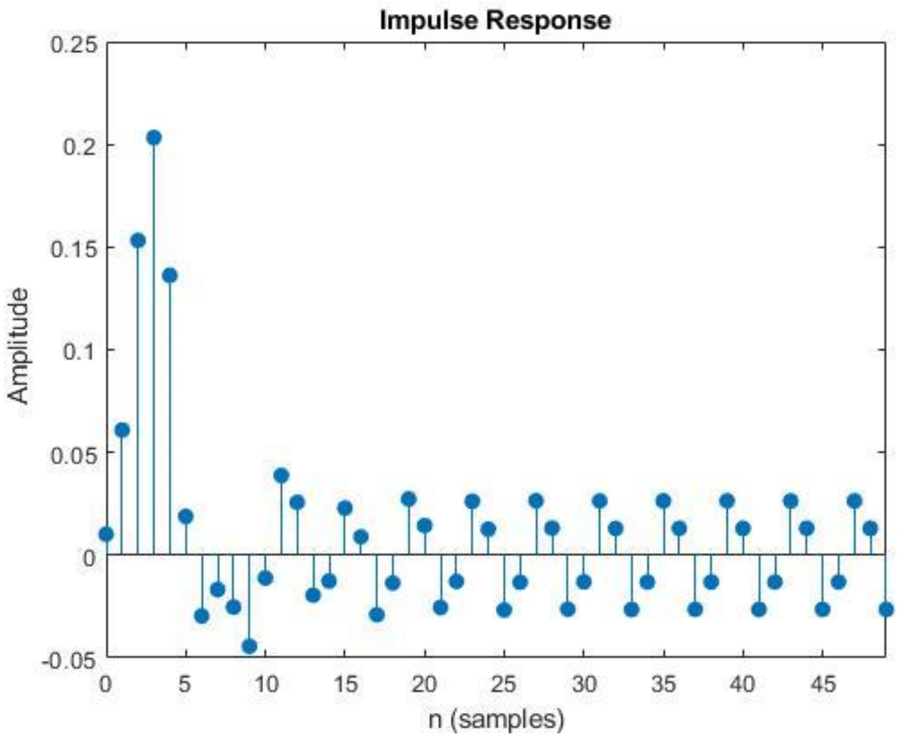

Pole-Zero Plot of Output Function
Imaginary Part
Real Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.5
-1
-0.5
0
0.5
1

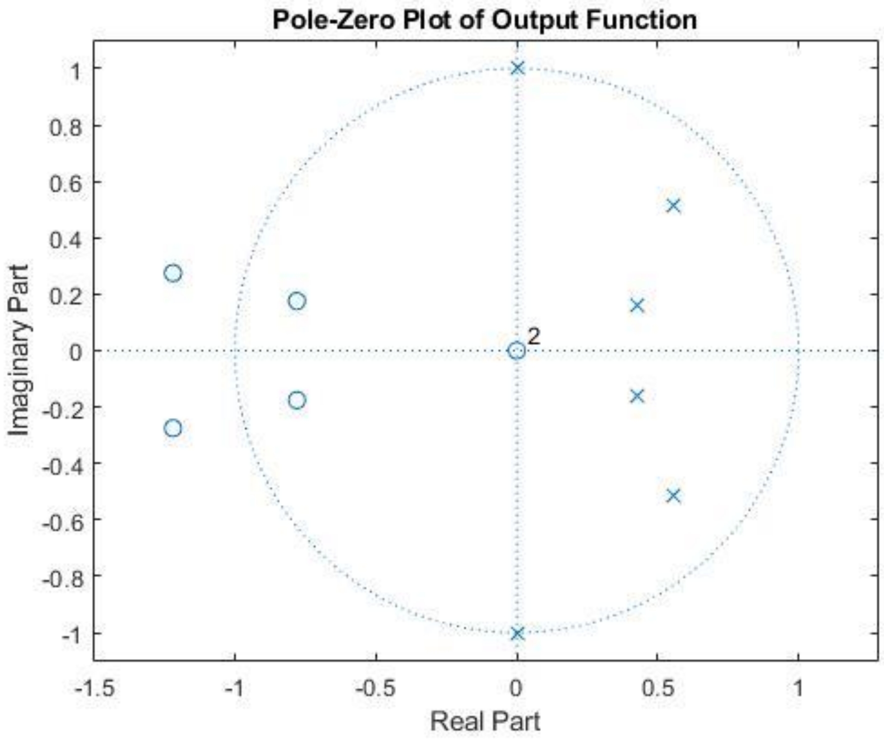
For $\omega = 0.4\pi$



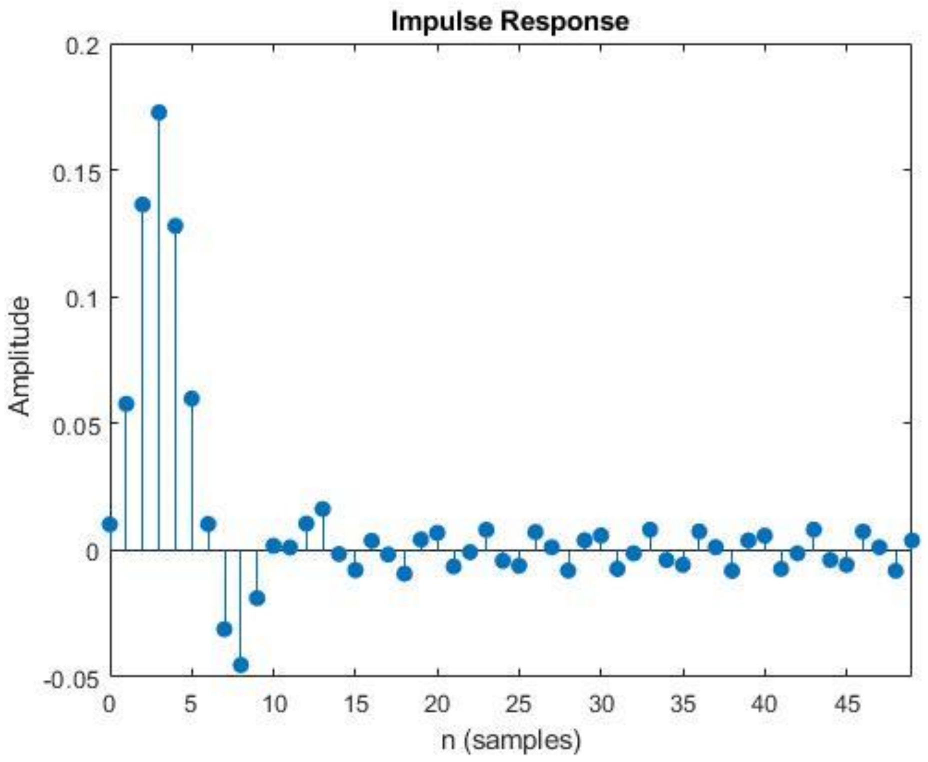


For ω = 0.5π

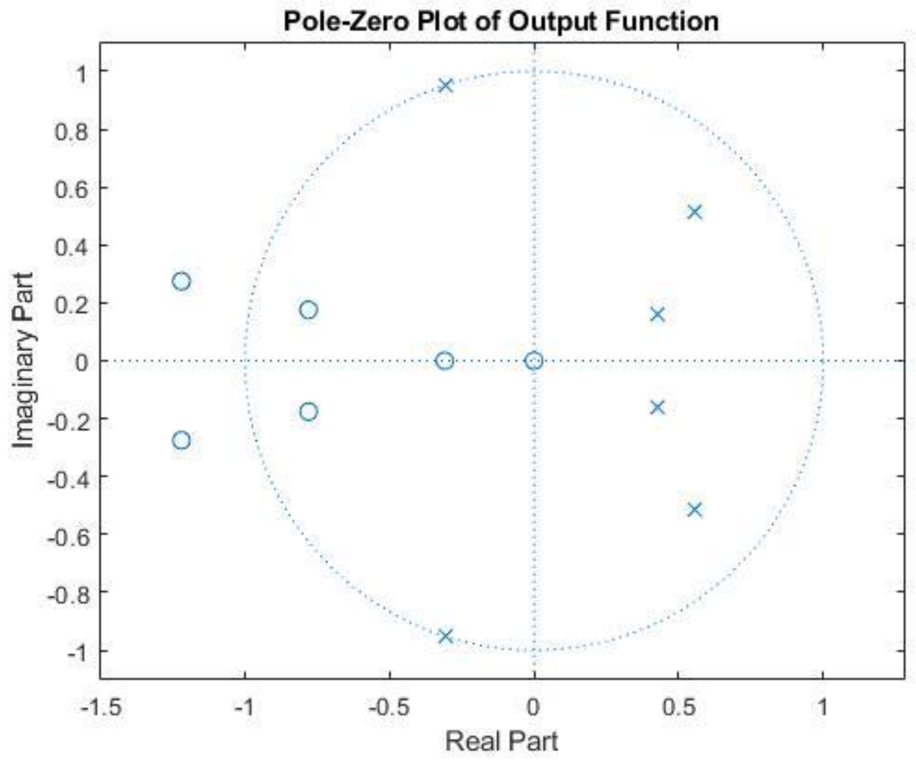

Impulse Response
Amplitude
n (samples)
0.25
0.2
0.15
0.1
0.05
0
-0.05
0
5
10
15
20
25
30
35
40
45


Pole-Zero Plot of Output Function
Imaginary Part
Real Part
2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.5
-1
-0.5
0
0.5
1

For ω = 0.6π

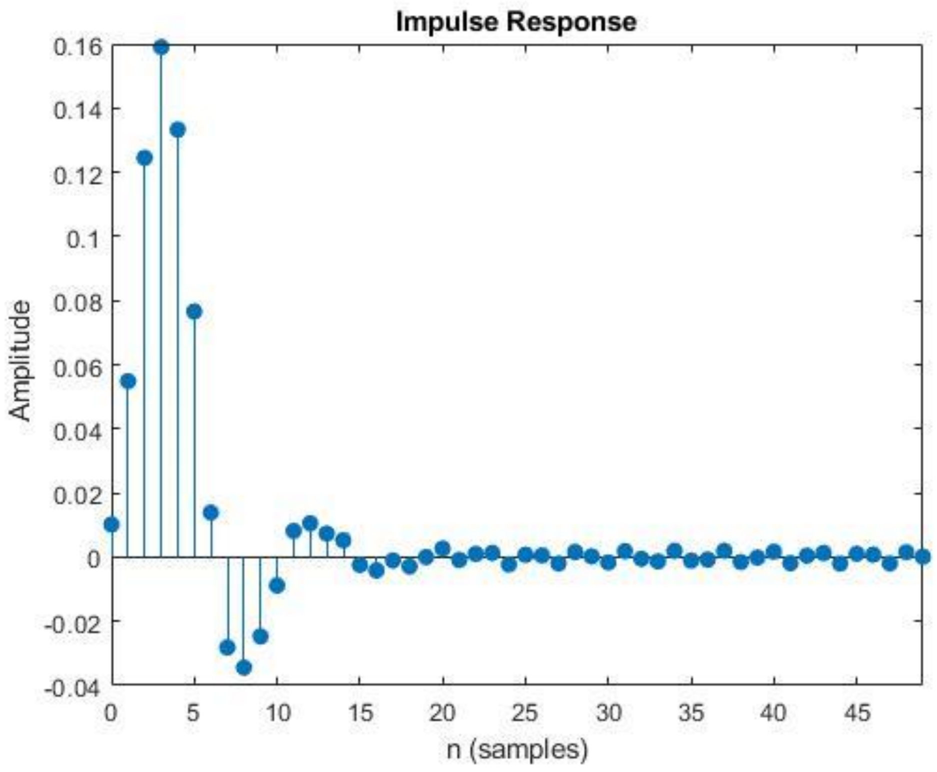


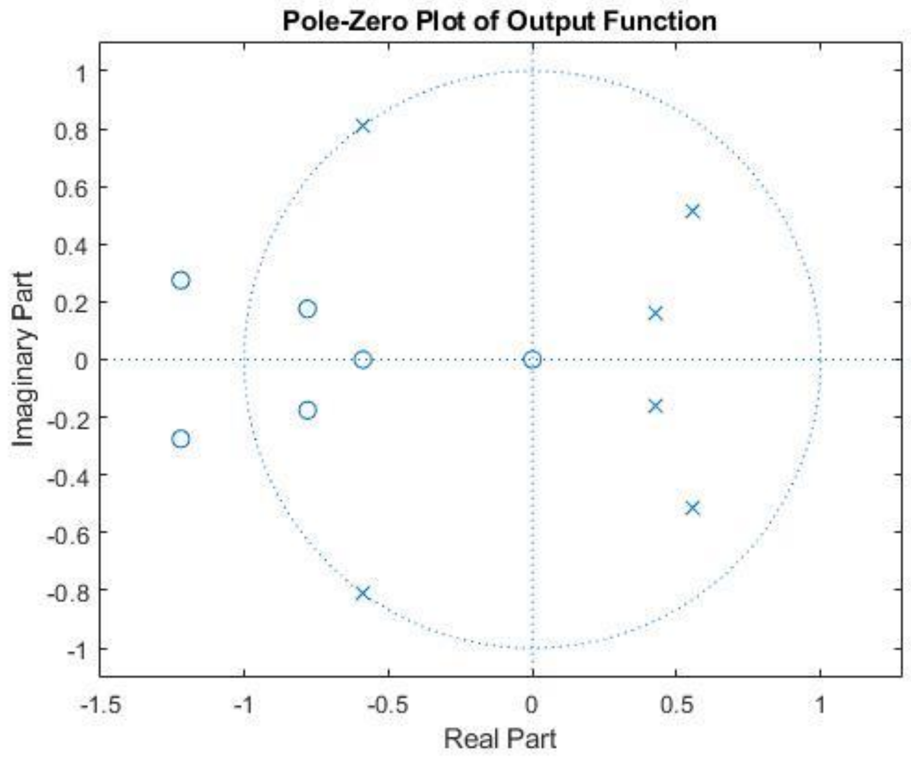
Impulse Response
Amplitude
n (samples)
0.2
0.15
0.1
0.05
0
-0.05
0
5
10
15
20
25
30
35
40
45



Pole-Zero Plot of Output Function
Imaginary Part
Real Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.5
-1
-0.5
0
0.5
1

For $\omega = 0.7\pi$





Discussion:

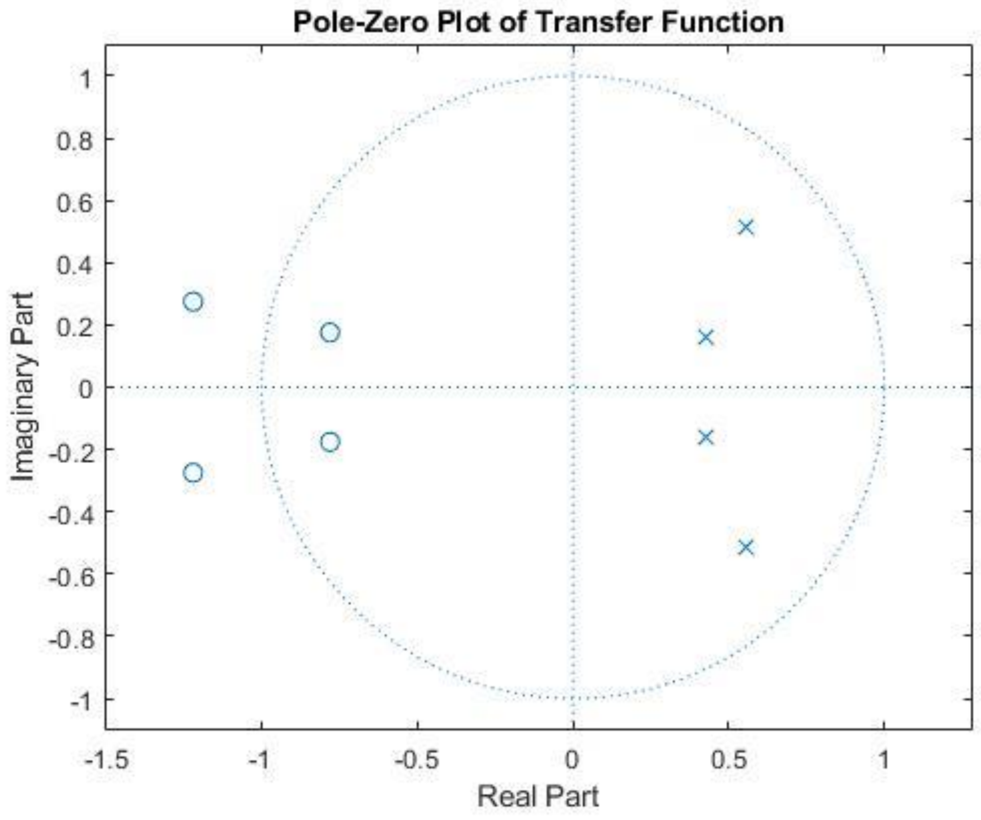
To calculate and plot, we have used the coefficient matrices of the Z function. We did this part manually to get the Z function from the time domain function where for any cos function the transform is

$$X(z) = \frac{1 - \cos\omega_0 z^{-1}}{1 - 2\cos\omega_0 z^{-1} + z^{-2}}.$$

Now Butterworth is a low pass filter that removes the high frequency components of the input function. As we increase the frequency of the input function, the signal faces significant attenuation, eventually resulting in distortion. The filter is able to pass signals for angular frequency lower than 0.5*pi without much distortion.

When we plot the poles and zeros of the output function, there are poles from both the filter system function and the input signa;. Changing the input signal frequency changes the poles of input and thus some poles seen to be changing position with frequency.

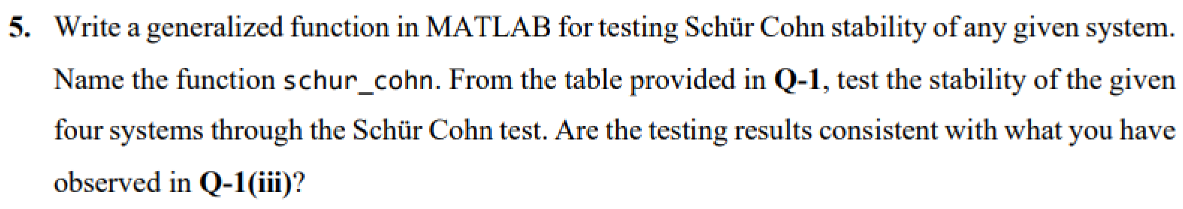
But the transfer function poles remain the same at constant position as seen in the plot below.



A fourth order filter has four poles and four zeros as we can see.

Problem 5

Question:



5. Write a generalized function in MATLAB for testing Schür Cohn stability of any given system. Name the function `schur_cohn`. From the table provided in **Q-1**, test the stability of the given four systems through the Schür Cohn test. Are the testing results consistent with what you have observed in **Q-1(iii)**?

Function declaration:

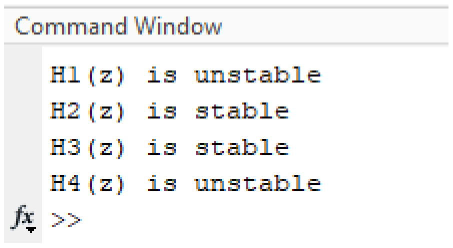
```
function s=schur_cohn(Am)
    Am = Am / Am(1); %normalizing
    stable = 1;
    for i=1:length(Am)-1
        Bm = flip(Am);
        Km = Am(end);

        if abs(Km)>=1 %unstable
            stable=0;
            break;

        else
            Am = (Am-Km*Bm) / (1-Km^2);
            Am = Am(1:end-1);
        end
    end

    if stable==0
        s='unstable';
    else
        s='stable';
    end
end
```

We defined the testing function here that takes the denominator as input and gives a string output stating whether the system function is stable or not.

Code:

```
clc;
clear all;
close all;

a1 = [4 -3 -1];
a2 = [4 -3 1];
a3 = [1 -13/12 3/8 -1/24];
a4 = [1 -13/6 7/8 -1/24];

s1 = schur_cohn(a1);
s2 = schur_cohn(a2);
s3 = schur_cohn(a3);
s4 = schur_cohn(a4);

fprintf('H1(z) is %s\n',s1);
fprintf('H2(z) is %s\n',s2);
fprintf('H3(z) is %s\n',s3);
fprintf('H4(z) is %s\n',s4);
```

Command window output:



```
H1(z) is unstable
H2(z) is stable
H3(z) is stable
H4(z) is unstable
>>
```

Conclusion:

We have used various functions to observe different properties of Z-transformation. We learnt to identify system stability and take necessary steps accordingly. We also learnt to identify the stable ROC for any given function. Thus, this experiment has been successful to gain an insight on Z-transform as a whole.