**Name: KVNG Vikram**

**SC code: SC15B148**

**Roll no: 8**

**Labsheet 6**

1. For the tasks set out in this labsheet, we will use rational z-transforms. Note down your answers to the following questions.
   * Write down the definition of rational z-transforms.
   * We note that rational z-transforms can be represented using their poles and zeros. Think about and note down how you will represent rational z-transforms on a computer.
   * Can rational z-transforms be represented using matlab vectors? Do you need to represent their multiplicities separately? Does there exist a case where separate representation of multiplicities is useful?
   * Does the ordering of poles and zeros (e.g., a magnitude-wise sorted vector) in your representation matter? Does there exist a case where an ordered representation is useful?

clear;clc;

%Give the zeros and poles  
zero = [1+1j 2 3 4 3];  
pole = [-1-2j 0 1 1 1 2 ];  
  
% defining a symbolic variable z  
syms z  
syms f(z) % symbolic function  
f = z\*0+1; % initially assigned zero  
  
% multiplying the terms of zeros in nominator  
for i = 1:numel(zero)  
 f = f\*(z-zero(i)) ;  
end  
% dividing terms of poles in denominator  
for i = 1:numel(pole)  
 f = f/(z-pole(i)) ;  
end  
  
f = simplify(f) % simplifying the expression

f =  
   
((z - 3)^2\*(z - 4)\*(z - 1 - 1i))/(z\*(z - 1)^3\*(z + 1 + 2i))

%{  
Here the zeros and poles of z transforms are given in the form of matlab  
vectors. So it can be said that rational z transforms can be represented  
using vectors. Here for there is no need for seperate representation of  
multiplicities, this can be done by giving corresponding number of similar  
values in the vectors itself.  
%}

%{  
Here the ordering of zeros and poles does not matter because the expression is being simplified.  
%}

[*Published with MATLAB® R2017a*](http://www.mathworks.com/products/matlab)

1. In this task, you will visualize the magnitude |*X*(*z*)| of the z-transform *X*(*z*) of a sequence *x*[*n*]. Assume that *X*(*z*) is a rational z-transform which is represented using the representation that you have obtained above in (Task 2). Obtain a 3D plot of |*X*(*z*)| as a function of the real and imaginary part of *z* for three choices of rational *X*(*z*). You are free to make these three choices. Comment on how |*X*(*z*)| behaves near poles and zeros. How does the behaviour near poles and zeros change with the multiplicity of poles and zeros? Comment on the region of convergence of the rational z-transforms you have chosen.

clear;clc;

%Give the zeros and poles  
zero = [1 -1 1j -1j];  
pole = [0 1+1j 1-1j -1+1j -1-1j ];  
  
R = 3; % range of |X(z)| plot from -R to R  
res = .05; % resolution of |X(z)| plot  
  
syms z % defining the symbolic variable z  
syms f(z) % symbolic function  
f = z\*0+1; % initially zero  
  
% multiplying terms of zeros  
for i = 1:numel(zero)  
 f = f\*(z-zero(i)) ;  
end  
% dividing terms of poles  
for i = 1:numel(pole)  
 f = f/(z-pole(i)) ;  
end  
f = simplify(f) % simplifying

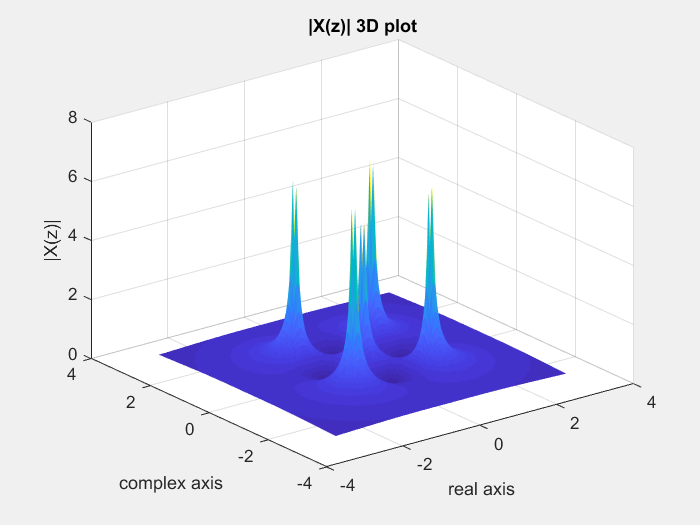
f =  
   
(z^4 - 1)/(z\*(z^4 + 4))

real = -R:res:R; % real axis  
complex = 1j\*real; % complex axis  
c = real+complex'; % complex plane  
  
% defining the function which will calculate values.  
myfunc = matlabFunction(f,'vars',z)  
  
% p is the matrix that stores the magnitude values  
% caluclating the value of p for each point in the plane  
for i = 1:numel(real)  
 for j = 1:numel(complex)  
 p(i,j) = myfunc(real(i)+complex(j));  
 % value of p will be left empty if there is a pole  
 end  
end

myfunc =  
  
 function\_handle with value:  
  
 @(z)(z.^4-1.0)./(z.\*(z.^4+4.0))

surface plotting

surf(real,complex/1j,abs(p),'LineStyle','none')  
title('|X(z)| 3D plot')  
xlabel('real axis')  
ylabel('complex axis')  
zlabel('|X(z)|')



%{  
Inference: subs() function takes a lot of time but in the above procedure  
the same process can be done with lot lesser time of execution.  
  
Near zeros the |X(z)| plot will have the value zero and at poles the  
magnitude value is infinite. Here in the code it takes the just before  
value which is also high. As the multiplicity of zero at a point increases  
the magnitude decreases to zero more rapidly and as the multiplicity of  
pole at a point increases the magnitude increases to infinity more rapidly.  
  
For the given poles and zeros the ROC |z|>2^(0.5)  
  
For  
2)  
zeros = [2 3 4 ];  
poles = [0];  
ROC is |z|>0  
3)  
zeros = [0 1 2];  
poles = [3];  
ROC is |z|>3 or |z|<3  
%}

[*Published with MATLAB® R2017a*](http://www.mathworks.com/products/matlab)

1. In this task, you will obtain the DTFT of a sequence *x*[*n*] from its z-transform *X*(*z*). Assume that the sequence you are looking at has a rational z-transform *X*(*z*) represented using the representation in (Task 2).
   1. Obtain the magnitude plot of the DTFT from *X*(*z*) as discussed in class (hint: use the distance from *ejω* method). Plot the magnitude plot from 0 to 4*π* radians/sample and comment on the periodicity of the DTFT (use an appropriately sampled *ω* so that the magnitude plot is smooth).
   2. Obtain the phase plot of the DTFT from *X*(*z*) as discussed in class (hint: use the angle from *ejω* method). Plot the phase plot from 0 to 4*π* radians/sample and comment on the periodicity of the DTFT (use an appropriately sampled *ω* so that the phase plot is smooth).

Obtain the DTFT of two rational z-transforms. You are free to choose these two z-transforms (we had learned how to plot the approximate DTFT from the poles and zeros, try to choose two z-transforms which will give you a high pass and a low pass filter response respectively).

clear;clc;

Give signal x[n] and n from start point

x = [5 5 5 4 4 4 3 2 1 ]; % signal x[n]  
n = [-1 0 1 2 ]; % values of n corresponding to x[n]  
  
% matching the array sizes of x[n] and n  
if numel(x)~=numel(n)  
 if numel(x)<numel(n)  
 x = [x zeros(1,(numel(n)-numel(x)))];  
 else  
 n = [n [n(numel(n))+1:n(numel(n))+(numel(x)-numel(n))]];  
 end  
end  
  
syms z  
syms f(z)  
f = z\*0;  
  
for i = 1:numel(x)  
 f = f + x(i)\*z^(-n(i));  
end  
f = simplify(f)

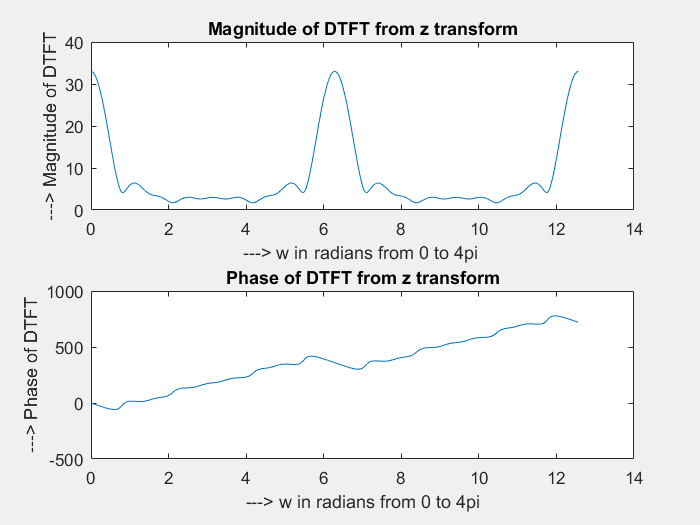
f =  
   
(5\*z^8 + 5\*z^7 + 5\*z^6 + 4\*z^5 + 4\*z^4 + 4\*z^3 + 3\*z^2 + 2\*z + 1)/z^7

%DTFT x axis  
wresol = 0.01;  
w = 0:wresol:4\*pi;  
c = exp(1j.\*w);  
  
myfunc = matlabFunction(f,'vars',z)  
  
for i = 1:numel(c)  
  
 p(i) = myfunc(c(i));  
end

myfunc =  
  
 function\_handle with value:  
  
 @(z)1.0./z.^7.\*(z.\*2.0+z.^2.\*3.0+z.^3.\*4.0+z.^4.\*4.0+z.^5.\*4.0+z.^6.\*5.0+z.^7.\*5.0+z.^8.\*5.0+1.0)

plotting the magnitude and phase

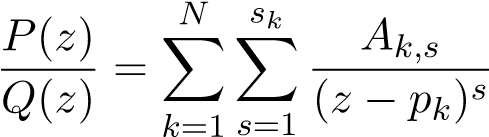
subplot(2,1,1)  
plot(w,abs(p))  
title('Magnitude of DTFT from z transform')  
xlabel('---> w in radians from 0 to 4pi ')  
ylabel('---> Magnitude of DTFT')  
subplot(2,1,2)  
plot(w,phase(p)\*180/pi)  
title('Phase of DTFT from z transform')  
xlabel('---> w in radians from 0 to 4pi')  
ylabel('---> Phase of DTFT')



%{  
The obtained DTFT is having a periodicity of 2pi  
%}

[*Published with MATLAB® R2017a*](http://www.mathworks.com/products/matlab)

1. In this task, you will write a function that will “expand out” a rational z-transform as a partial fractions expansion.
   1. Write a matlab function “partialFractionExpander” that has as input the rational ztransform represented as in Task (2).
   2. The function should first check if the number of zeros is strictly less than the number of poles.
   3. If it is so, then the function should use the method that we had discussed in class in order to obtain the coefficients of the different terms that arise in the partial fraction expansion. For example, the function should expand rational *X*(*z*) as

*,*

where *P*(*z*) and *Q*(*z*) are polynomials in *z*, *N* is the number of poles, *pk* represents the *kth* pole, and *sk* is the multiplicity of the *kth* pole.

* 1. The function should put out an output which represents the RHS above. What representation would you choose? Would a list of 3-tuples, with the tuple containing *pk,sk*, and *Ak,s* be sufficient?
  2. Implement this function under the assumption that *sk* ≤ 2 for all *k*.
  3. Test your function with five different *X*(*z*). You are free to choose *X*(*z*) (choose *X*(*z*) with different number of poles and zeros and multiplicities).
  4. In each case, obtain the partial fractions expansion manually. Compare the expansion that you obtain with the function with the expansion that you obtain manually.

function partialFractionExpander(zer, mzeros, poles, mpoles)  
zer=[1 2];  
mzeros=[1 1];  
poles=[3];  
mpoles=[4];  
syms z H N D  
K=zeros(length(mpoles),max(mpoles));  
for i=1:length(poles)  
for j=1:mpoles(i)  
%Computing residue  
N=1\*(z-zer).^mzeros;  
D=(z-poles).^mpoles;  
H=prod(N)/prod(D);  
d=diff(H\*(z-poles(i))^mpoles(i),j-1);  
a=subs(d, z, poles(i));  
  
K(i,j)=1/factorial(mpoles(i)-j)\*a;  
end;  
end;  
syms z X  
Y=0;  
for i=1:length(mpoles)  
for j=1:mpoles(i)  
Y=Y+K(i,j)/(z-poles(i))^j;  
end;  
end;  
Y  
end

Y =  
   
1/(3\*(z - 3)) + 3/(2\*(z - 3)^2) + 2/(z - 3)^3