To Calculate Continuous and Discrete time standard signal in octave

```
% Time vectors
t = 0:0.01:2;
n = 0:0.1:2;

% Signals
continuous_signal = sin(pi*t);
discrete_signal = sin(pi*n);
continuous_unit_step = t >= 0;
discrete_unit_step = n >= 0;

% Plot continuous and discrete signals
figure;
subplot(2,2,1);
plot(t, continuous_signal);
title('Continuous Signal');
xlabel('Time (t)');
ylabel('Amplitude');
grid on;

subplot(2,2,2);
stem(n, discrete_signal, 'filled');
title('Discrete Signal');
xlabel('Sample (n)');
ylabel('Amplitude');
grid on;

subplot(2,2,3);
plot(t, continuous_unit_step);
title('Continuous Unit Step');
xlabel('Time (t)');
```

```
ylabel('Amplitude');
grid on;

subplot(2,2,4);
stem(n, discrete_unit_step, 'filled');
title('Discrete Unit Step');
xlabel('Sample (n)');
ylabel('Amplitude');
ylim([0 1.2]);
grid on;

% Continuous and discrete unit ramp signals
t = 0:0.01:1;
r_cont = t;

figure;
subplot(2,2,1);
plot(t, r_cont);
title('Continuous Unit Ramp Signal');
xlabel('t');
ylabel('r(t)');
grid on;

n = 0:0.1:1;
r_disc = n;
subplot(2,2,2);
stem(n, r_disc, 'filled');
title('Discrete Unit Ramp Signal');
xlabel('n');
ylabel('r[n]');
grid on;
```

```
% Continuous and discrete impulse signals
t = -1:0.01:1;
impulse = t == 0;
subplot(2,2,3);
plot(t, impulse);
title('Continuous Impulse Signal');
xlabel('t');
ylabel('\delta(t)');
grid on;

n = 0:0.1:1;
delta_disc = (n == 0);
subplot(2,2,4);
stem(n, delta_disc, 'filled');
title('Discrete Impulse Signal');
xlabel('n');
ylabel('\delta(n)');
grid on;

% Continuous and discrete exponential signals
alpha = 1;
t_const = 0:0.01:1;
n_disc = 0:0.1:1;
x_cont = exp(alpha * t_const);
x_disc = exp(alpha * n_disc);

figure;
subplot(1,2,1);
plot(t_const, x_cont, 'g');
title('Continuous Exponential Signal');
```

```
xlabel('t');
ylabel('x(t)');
grid on;

subplot(1,2,2);
stem(n_disc, x_disc, 'g');
title('Discrete Exponential Signal');
xlabel('n');
ylabel('x[n]');
grid on;
```

Aim : - To Plot frequency response by using z – transform or dtft

```
omega = linspace(0,2*pi,50);

z=exp(j*omega);

%H(z)=1+2z^-1+3z^-2+4z^-4
h_z = 1+2 * (1./z) + 3*(1./z.^2) + 4*(1./z.^3);

magnitude  = abs(h_z);
phase = angle(h_z);

figure;
subplot(2,2,1);
plot(omega,magnitude);
title('Magnitude Response');
xlabel('Frequency (Omega)');
ylabel('|h(e^(j\omega))|');
grid on;
```

```
subplot(2,2,2);
plot(omega,phase);
title('Phase Response');
xlabel('Frequency(omega)');
ylabel('angle H(e^(j\omega))');
grid on;
```

Aim : To plot frequency response DT-LTI system using DFT

```
% Define the signal
n = 0:3; % Time index
x = [1 1 1 1]; % Example signal
X = fft(x);
magnitudeX = abs(X);
phaseX = angle(X);
figure;
subplot(3,1,1);
stem(n, x, 'filled');
title('Original Signal');
xlabel('n');
ylabel('x[n]');
subplot(3,1,2);
stem(n, magnitudeX, 'filled');
title('Magnitude Spectrum');
xlabel('Frequency Index');
ylabel('|X[k]|');
subplot(3,1,3);
stem(n, phaseX, 'filled');
title('Phase Spectrum');
xlabel('Frequency Index');
ylabel('Phase of X[k] (radians)');
```

Aim : Calculate Circular Concolution using dft & idft

```
clc;
x1 = [0,2,3,4];
x2 = [2,4,3,5];
X1 = fft(x1);
X2 = fft(x2);
Xk = X1 .* conj(X2);
Convolution = ifft(Xk);
disp('x1:');
disp(x1);
disp('x2:');
disp(x2);
disp('DFT of x1:');
disp(X1);
disp('DFT of x2:');
disp(X2);
disp('DFT of x1 and x2 (Xk):');
disp(Xk);
disp('IDFT of circular Convolution:');
disp(Convolution);
figure;
stem(real(Convolution), 'filled');
title('Circular Convolution (IDFT)');
xlabel('n');
ylabel('y[n]');
grid on;
```

Aim : To Calculate & plot linear Convolution

```
clc;
x1 = [1,3,4,6,4];
x2 = [1,2,3,1,3];
linear = conv(x1, x2);
disp('x1:');
disp(x1);
disp('x2:');
disp(x2);
disp('Linear Convolution:');
disp(linear);
figure;
stem(linear, 'filled');
title('Linear Convolution');
xlabel('n');
ylabel('y[n]');
grid on;
```

Aim : -To calculate circular cross convolution & auto correlation using dft & Idft & plot the output

```
% Define the sequences
x1 = [4,2,1,2];
x2 = [0,1,-2,3];
% Ensure both sequences have the same length
N = max(length(x1), length(x2));
x1 = [x1, zeros(1, N - length(x1))]; % Zero-padding if necessary
x2 = [x2, zeros(1, N - length(x2))];
% Compute the DFT of both sequences
X1 = fft(x1);
X2 = fft(x2);
% Circular Cross-Correlation using DFT and IDFT
cross_corr_dft = ifft(X1 .* conj(X2));
```

```
% Circular Auto-Correlation using DFT and IDFT

auto_corr_dft = ifft(X1 .* conj(X1));

% Plotting the results

figure;

subplot(2, 1, 1);

stem(0:N-1, auto_corr_dft, 'filled');

title('Circular Auto-Correlation x1 = [n] ');

xlabel('Lag');

ylabel('Correlation');

grid on;

subplot(2, 1, 2);

stem(0:N-1, cross_corr_dft ,'filled');

title('Circular Cross-Correlation x1 [n] x2 = [n] ');

xlabel('Lag');

ylabel('Correlation');

grid on;
```

Aim : - To plot frequency response for low pass and high pass filter of fir with order x = 48 and cutoff frequency fc=0.65 also plot band pass filter and band rejection filter for fc(1) = 0.2 and fc(2)=0.6

```
pkg load signal

%low pass filter

N=48;

fc=0.65;

figure(1)

freqz(a1)

%high pass filter

N=48;

fc=0.65;

ah=firl(N,fc,'high');

figure(2)

freqz(ah)
```

```
%band pass filter
N=48;
abp=firl(N,[0.2,0.6],'pass')
figure(3)
freqz(abp)
%band restriction filter
N=48;
abr=firl(N,[0.2,0.6],'stop')
figure(4)
freqz(abr)
```

Aim : To design IIR filter using Impulse Invarient Method

```
pkg loadsignal
clc;
clear all;
fs=1
num1=9;
den1=conv([1,2],[1,3])
[b1,a1]=impinvar(num1,den1,fs)

fs=1
num2=1*[1 ,3];
den2=conv([1,2],[1,5])
[b1,a1]=impinvar(num2,den2,fs)
```