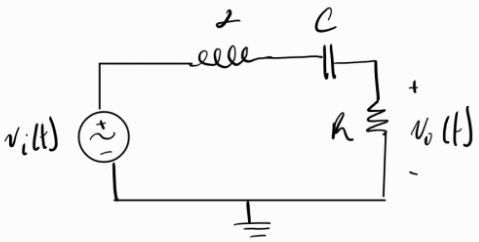
ELEC 3600 – Signals and Systems
Summer 2024: Project 1
Analysis of an LTI-CT System: Analog filter

Sophia Sistachs

1.) Input-output differential equation



$$1)\ v(t) = v_L(t) + v_C(t) + v_R(t)$$

$$v(t) = L\frac{d}{dt}\frac{v(t)}{R} + \frac{1}{C}\int\left(\frac{v(t)}{R}\right)dt + v(t)$$

$$\frac{dv(t)}{dt} = \frac{L}{R}\frac{d^2v(t)}{dt^2} + \frac{1}{RC}v(t) + \frac{dv(t)}{dt}$$

$$R\cdot D\,y(t) = (D^2L + DR + \tfrac{1}{C})\,x(t)$$

2.) Linearity and Time Check

$$2)\ K_1\left[(D^2L + DR + \tfrac{1}{C})\,x(t)\right] + K_2\left[(D^2L + DR + \tfrac{1}{C})\,x(t)\right] = K_1(RD\,y(t)) + K_2(RD\,y(t))$$

System is linear

$$R\cdot D\,x(t) = (D^2L + DR + \tfrac{1}{C})\,y(t-T)$$

$$R\cdot D\,x(t) = (D^2L + DR + \tfrac{1}{C})\,y(t-T)$$

System is Time-invariant

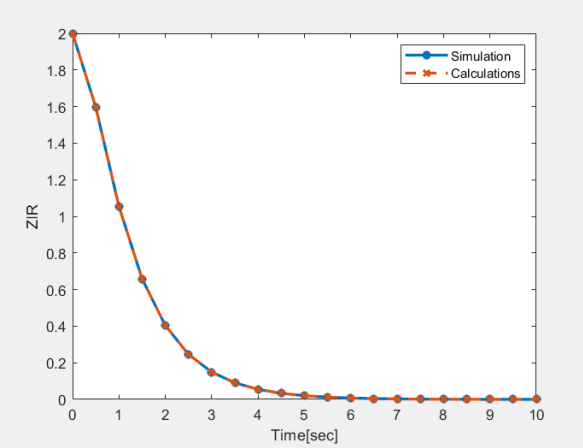
3.) Zero-Input-Response:

The blue line showcases the graph generated by the function calculated by MATLAB and the orange line showcases the graph generated by using the calculated response. As we can see, the graphs match perfectly, which confirms the results.

The zero-input response is based on the value of the initial conditions with its input being zero. The zero-state response is based on its input with the initial conditions being zero. The total state response is the sum of the two. Below are the hand calculations for the Zero Input Response, and Zero State Response.

Zero Input Response

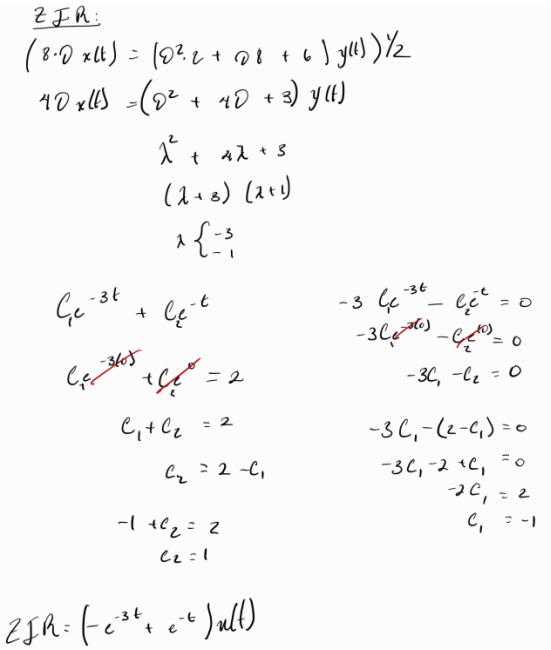
Graph:



Code:

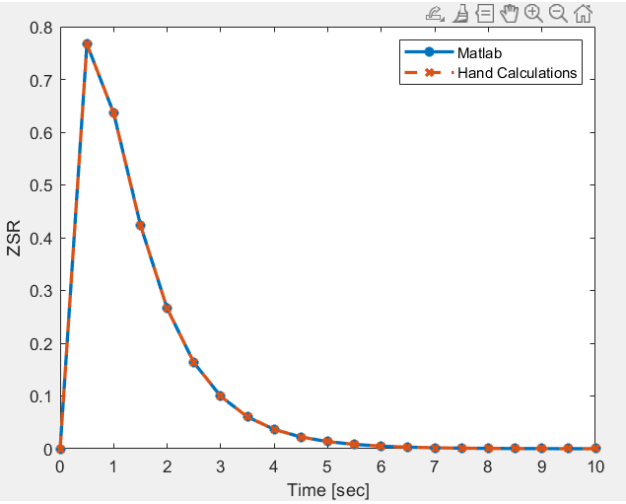
```
%% Signals and Systems Project 1
%% Zero Input Response
syms y(t) t tau xt
% ZIR
%time vector
tt = [0:0.5:10];
Dy = diff(y,1);
D2y = diff(y,2);

% Initial conditions
cond1_zir = y(0) == 2;
cond2_zir = Dy(0) == 0;
conds_zir = [cond1_zir,cond2_zir]; % vector containing both the initial conditions

% differential equation
sys = D2y + 4*Dy + 3*y == 0;
y_zir = dsolve(sys, conds_zir); % solution of the differential equation
y_zir_d = double(subs(y_zir,t,tt)); % ZIR evaluated at the time vector
y_zir2 = y_zir*heaviside(t); % ZIR mutiplied by the unit step fuction
y_zir_d_calculated = (-1*exp(-3.*tt)) + (3*exp(-1.*tt)); %ZIR calculated by hands
```

Calculations:



Zero-State-Response:

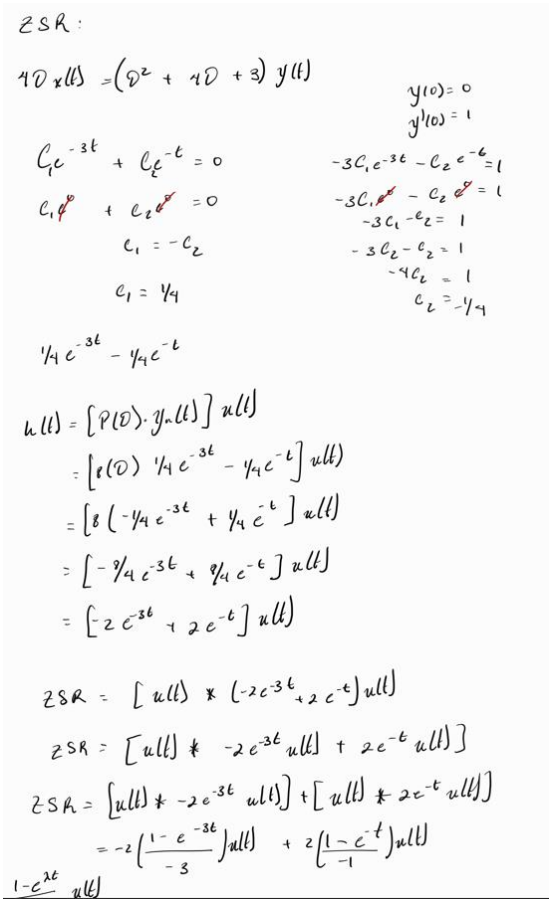
Graph:



Code:

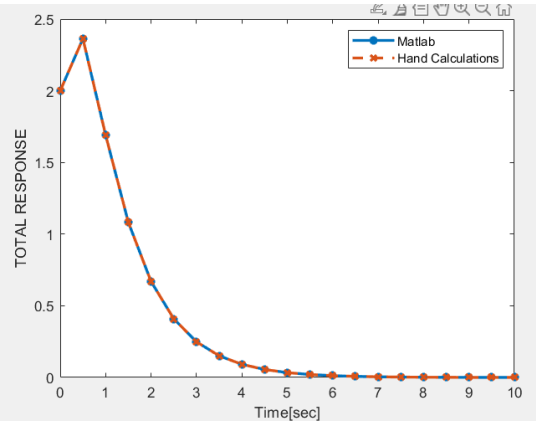
```
%% Zero State Response
% Input Signal
xt = heaviside(t);
% Define the system's differential equation
Dy = diff(y, 1);
% Initial Conditions
cond1_zsr = y(0) == 0;
cond2_zsr = Dy(0) == 1;
conds_zsr = [cond1_zsr,cond2_zsr];
sys = diff(y,2) + 4*diff(y,1) + 3*y(t) == 0;
% differential equation for ZSR
y_n = dsolve(sys,conds_zsr);
% Impulse response
ht = 4*diff(y_n);
%Convolution
y_zsr = int(subs(xt, tau) * subs(ht, t - tau), tau, 0, t);% Convolution Integral
ysc_zsr = y_zsr * heaviside(t); % ZSR Multiplied by unit step function
ysp_zsr = subs(ysc_zsr, t, tt); %ZSR Evaluated at time vector
y_zsr_calculated = (-2*exp(-3*tt) + 2*exp(-1*tt)); %ZSR Hand Calculations

figure
plot(tt, ysp_zsr, '-*', 'LineWidth',2)
hold on
plot(tt, y_zsr_calculated, '--X', 'LineWidth', 2)
legend('Matlab', 'Hand Calculations')
xlabel('Time [sec]')
ylabel('ZSR')
```

Calculations:



Total – State-Response:

Graph:

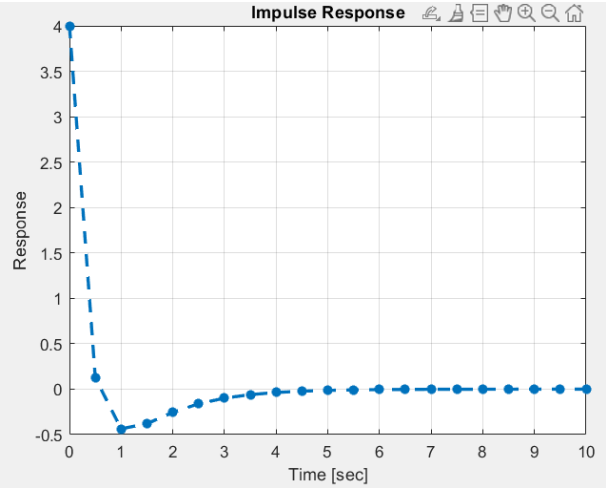


Code:

```
%% Toatal State Response

y_tc_calculated  = y_zir_d_calculated + y_zsr_calculated; %total response calculated
y_tc = ysp_zsr + y_zir_d; % total response with simulations

figure
plot(tt, y_tc, '-*','LineWidth', 2)
hold on
plot(tt, y_tc_calculated, '--x', 'LineWidth', 2)
legend ('Matlab', 'Hand Calculations')
xlabel('Time[sec]')
ylabel('TOTAL RESPONSE')
```

Impulse Response:

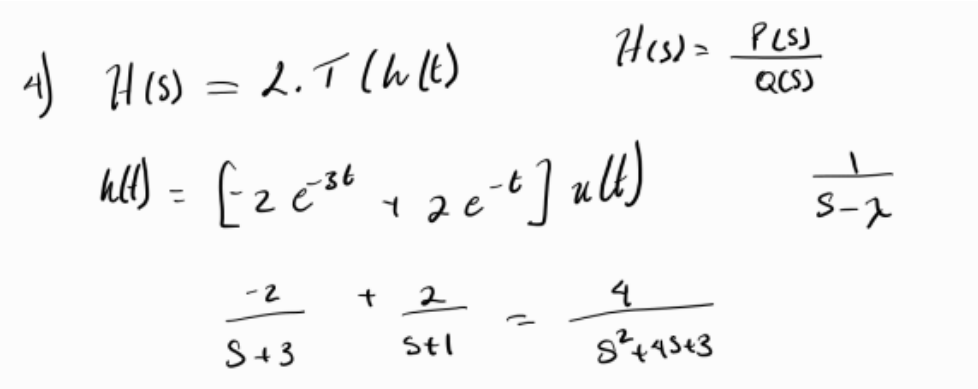
Graph:



Code:

```
%% Impulse Response
F = 4*s / (s^2 + 4*s + 3);
f = ilaplace(F, s, t);
f1 = double(subs(f, t, tt));

% Plot the impulse response
figure;
plot(tt, f1, '--*', 'LineWidth', 2)
title('Impulse Response')
xlabel('Time [sec]')
ylabel('Response')
grid on
```

4.) System Transfer Functions



5.) Step & Impulse Response

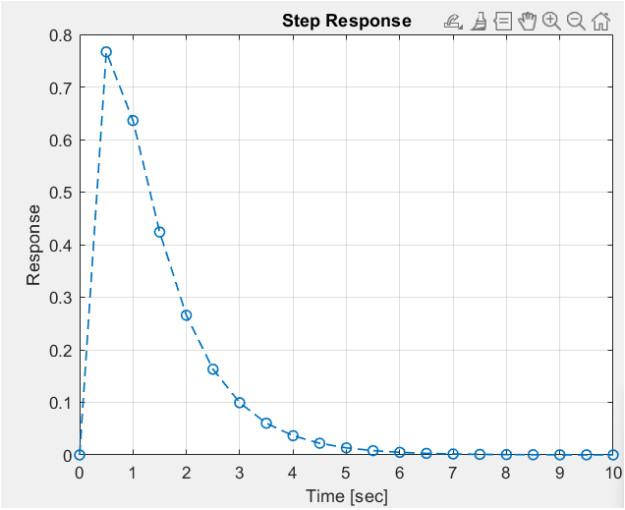
The following graphs showcase the step and impulse response which were graphed using the step() and impulse() functions in MATLAB.

The results obtained in this section of the poroject match those obtained in part 3. With the step response matching the graph of the zero-state response and both versions of the impulse response matching up as well.

In this case the input signal is a unit step function which means the ZSR becomes the step response. Similarly the LaPlace transform of an impulse function, yields the systems transfer function.

Step Response

Graph:



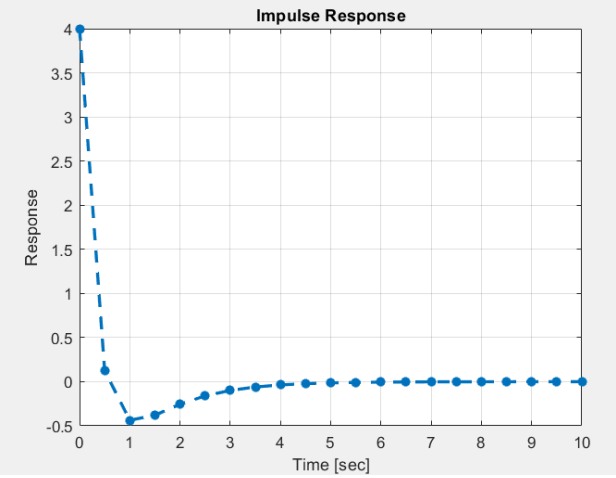
Code:

```
%% Step Response
% Calculating the function using step()
[st, t_step] = step(sys, tt);

% Plotting the step response
figure
plot(t_step, st, '--o', 'LineWidth', 1)
title('Step Response')
xlabel('Time [sec]')
ylabel('Response')
grid on
```

Impulse Response

Graph:

Code:

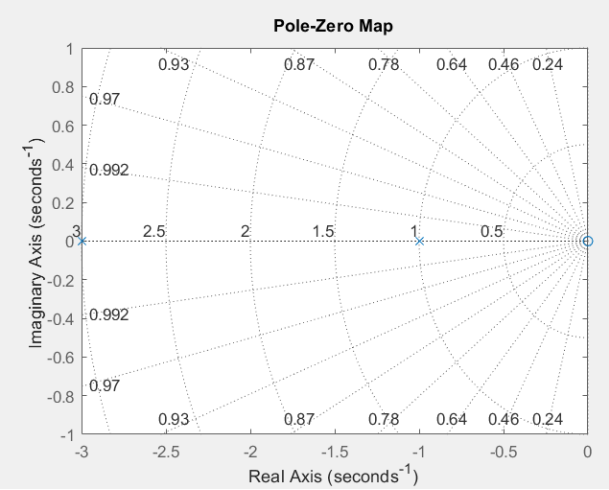



```
%% Impulse Response

% Calculating the function using impulse()
[ht, t_impulse] = impulse(sys, tt);

% Plotting the impulse response
figure
plot(t_impulse, ht, '--*', 'LineWidth', 2)
title('Impulse Response')
xlabel('Time [sec]')
ylabel('Response')
grid on
```

6.) Poles and Zeros Maps

The system is stable since both poles are located in the left-hand plane.

Graph: Code:




```
%% Zeros and Poles
% Plotting the poles and zeros using pzmap()
figure;
pzmap(sys)
title('Pole-Zero Map')
grid on
```

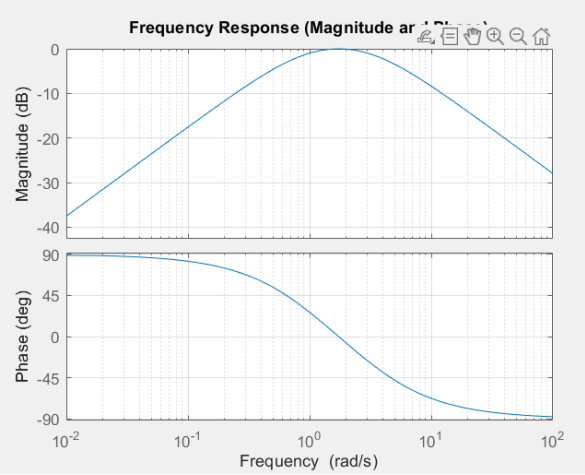
7.) Bode Plot

This Frequency Response represents a band pass filter. It passes frequencies around the cutoff frequency while attenuating both low and high frequencies.

From looking at the peak of magnitude plot, we can approximate the central frequency to be around 1.73.

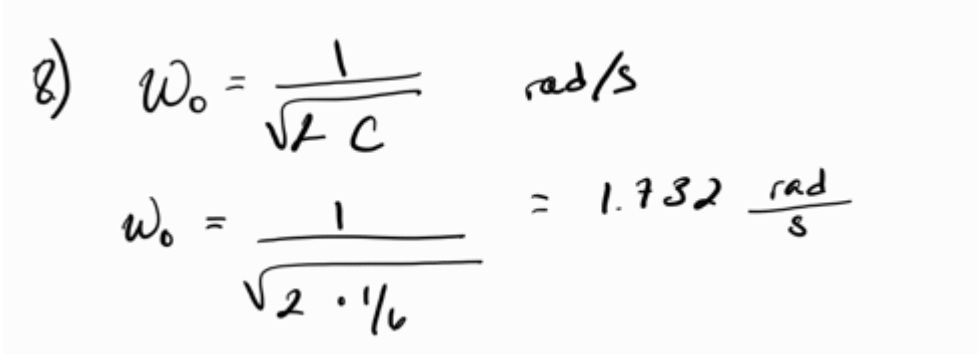
From looking at the bode plot we can estimate the lower cutoff frequency to be around 1 rad/s and the upper cutoff frequency to be around 3 rad/s

Graph: Code:




```
%% Bode Plot
% Bode plot to inspect magnitude and phase response
figure;
bode(sys)
title('Frequency Response (Magnitude and Phase)')
grid on
```

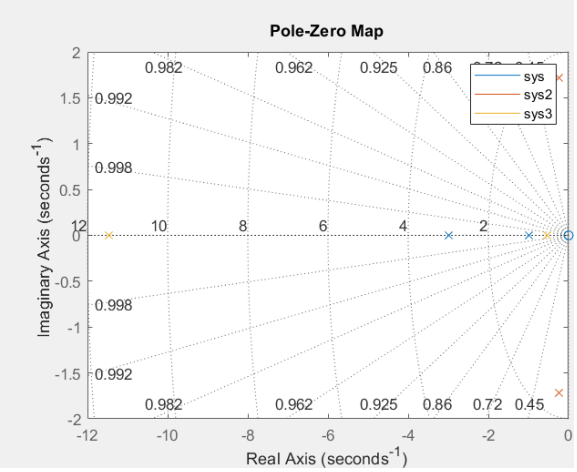
8.) Below is the calculations for the cutoff frequency, which matches with the estimation gathered from the bode plot.



9.) <u>Zeros and Poles with New Resistance Values:</u>

Increasing the resistance moves the poles closer to the real (x) axis, while decreasing it moves them closer to the imaginary (y) axis.

<u>Graph</u>: <u>Code</u>:

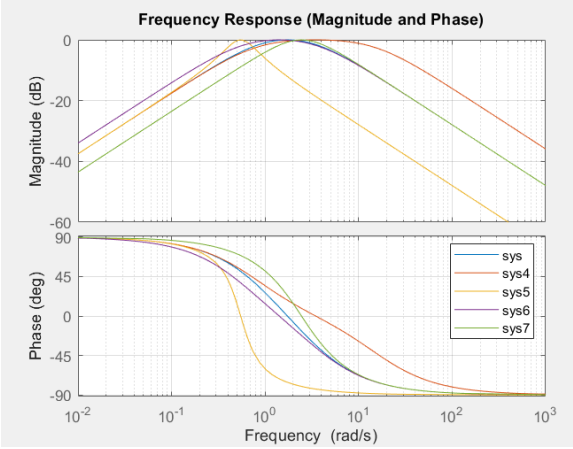



```
%% Zeros and Poles-New Restance Values
num2 = [0 1];           % Numerator for sys2
den2 = [2 1 6];         % Denominator for sys2: 2s^2 + s + 6

num3 = [0 1];           % Numerator for sys3
den3 = [1 12 6];        % Denominator for sys3: s^2 + 12s + 6

% Define the transfer functions
sys2 = tf(num2, den2);
sys3 = tf(num3, den3);

% Plotting the poles and zeros using pzmap()
figure;
pzmap(sys)
hold on
pzmap(sys2)
pzmap(sys3)
title('Pole-Zero Map')
legend('sys', 'sys2', 'sys3')
grid on
```

10.) Bode Plot with New Values

In this case, the inductor and capacitor are changing values. Increasing the values of L or C narrows the bandwidth which is shown by the sharper peaks in magnitude. Decreasing the values of L or C broadens the bandwidth, which is shown by the lower peak in the magnitude response.

Graph: Code:




```
%% Bode Plot- New Values
% L = 1/2
num4 = [8 0];
den4 = [1/2 8 6];
% L = 20
num5 = [4 0];
den5 = [10 4 3];
% C = 1/2
num6 = [4 0];
den6 = [1 4 2];
% C = 1/12
num7 = [4 0];
den7 = [1 4 6];


% Define the transfer functions
sys4 = tf(num4, den4);
sys5 = tf(num5, den5);
sys6 = tf(num6, den6);
sys7 = tf(num7, den7);

% Bode plot to inspect magnitude and phase response
figure;
bode(sys)
hold on
bode(sys4)
bode(sys5)
bode(sys6)
bode(sys7)
title('Frequency Response (Magnitude and Phase)')
legend('sys', 'sys4', 'sys5', 'sys6', 'sys7')
grid on
```

Appendix

Matlab Code:

```
%% Signals and Systems Project 1
%% Zero Input Response
syms y(t) t tau xt s
%time vector
tt = [0:0.5:10];
```

```
Dy = diff(y,1);
D2y = diff(y,2);
% ZIR
% Initial conditions
cond1_zir = y(0) == 2;
cond2_zir = Dy(0) == 0;
conds_zir = [cond1_zir,cond2_zir]; % vector containing both the initial conditions
% differential equation
sys = D2y + 4*Dy + 3*y == 0;
y_zir = dsolve(sys, conds_zir); % solution of the differential equation
y_zir_d = double(subs(y_zir,t,tt)); % ZIR evaluated at the time vector
y_zir2 = y_zir*heaviside(t); % ZIR mutiplied by the unit step fuction
y_zir_d_calculated = (-1*exp(-3.*tt)) + (3*exp(-1.*tt)); %ZIR calculated by hands
%Graph
figure
plot(tt,y_zir_d,'-*','LineWidth', 2)
hold on
plot (tt,y_zir_d_calculated,'--x','LineWidth', 2)
legend ('Simulation','Calculations')
xlabel('Time[sec]')
ylabel('ZIR')
%% Zero State Response
% Input Signal
xt = heaviside(t);
% Define the system's differential equation
Dy = diff(y, 1);
% Initial Conditions
cond1_zsr = y(0) == 0;
cond2_zsr = Dy(0) == 1;
conds_zsr = [cond1_zsr,cond2_zsr];
sys = diff(y,2) + 4*diff(y,1) + 3*y(t) == 0;
% differential equation for ZSR
y_n = dsolve(sys,conds_zsr);
% Impulse response
ht = 4*diff(y_n);
%Convolution
y_zsr = int(subs(xt, tau) * subs(ht, t - tau), tau, 0, t);% Convolution Integral
ysc_zsr = y_zsr * heaviside(t); % ZSR Multiplied by unit step function
ysp_zsr = subs(ysc_zsr, t, tt); %ZSR Evaluated at time vector
y_zsr_calculated = (-2*exp(-3*tt) + 2*exp(-1*tt)); %ZSR Hand Calculations
figure
plot(tt, ysp_zsr, '-*', 'LineWidth',2)
hold on
plot(tt, y_zsr_calculated, '--X', 'LineWidth', 2)
legend('Matlab', 'Hand Calculations')
xlabel('Time [sec]')
ylabel('ZSR')
```

```
%% Toatal State Response
y_tc_calculated = y_zir_d_calculated + y_zsr_calculated; %total response calculated
y_tc = ysp_zsr + y_zir_d; % total response with simulations
figure
plot(tt, y_tc, '-*','LineWidth', 2)
hold on
plot(tt, y_tc_calculated, '--x', 'LineWidth', 2)
legend ('Matlab', 'Hand Calculations')
xlabel('Time[sec]')
ylabel('TOTAL RESPONSE')
%System Transfer Function
num = [4 0]; % Numerator coefficients for s
den = [1 4 3]; % Denominator coefficients for s^2 + 4s + 3
sys = tf(num, den);
%% Step Response
% Calculating the function using step()
[st, t_step] = step(sys, tt);
% Plotting the step response
figure
plot(t_step, st, '--o', 'LineWidth', 1)
title('Step Response')
xlabel('Time [sec]')
ylabel('Response')
grid on
%% Impulse Response
% Calculating the function using impulse()
[ht, t_impulse] = impulse(sys, tt);
% Plotting the impulse response
figure
plot(t_impulse, ht, '--*', 'LineWidth', 2)
title('Impulse Response')
xlabel('Time [sec]')
ylabel('Response')
grid on
%% Zeros and Poles
% Plotting the poles and zeros using pzmap()
figure;
pzmap(sys)
title('Pole-Zero Map')
grid on
%% Bode Plot
% Bode plot to inspect magnitude and phase response
figure;
bode(sys)
title('Frequency Response (Magnitude and Phase)')
grid on
%% Zeros and Poles-New Restance Values
```

```
num2 = [0 1]; % Numerator for sys2
den2 = [2 1 6]; % Denominator for sys2: 2s^2 + s + 6
num3 = [0 1]; % Numerator for sys3
den3 = [1 12 6]; % Denominator for sys3: s^2 + 12s + 6
% Define the transfer functions
sys2 = tf(num2, den2);
sys3 = tf(num3, den3);
% Plotting the poles and zeros using pzmap()
figure;
pzmap(sys)
hold on
pzmap(sys2)
pzmap(sys3)
title('Pole-Zero Map')
legend('sys', 'sys2', 'sys3')
grid on
%% Bode Plot- New Values
% L = 1/2
num4 = [8 0];
den4 = [1/2 8 6];
% L = 20
num5 = [4 0];
den5 = [10 4 3];
% C = 1/2
num6 = [4 0];
den6 = [1 4 2];
% C = 1/12
num7 = [4 0];
den7 = [1 4 6];
% Define the transfer functions
sys4 = tf(num4, den4);
sys5 = tf(num5, den5);
sys6 = tf(num6, den6);
sys7 = tf(num7, den7);
% Bode plot to inspect magnitude and phase response
figure;
bode(sys)
hold on
bode(sys4)
bode(sys5)
bode(sys6)
bode(sys7)
title('Frequency Response (Magnitude and Phase)')
legend('sys', 'sys4', 'sys5', 'sys6', 'sys7')
grid on
%% Impulse Response
F = 4*s / (s^2 + 4*s + 3);
```

```
f = ilaplace(F, s, t);
f1 = double(subs(f, t, tt));
% Plot the impulse response
figure;
plot(tt, f1, '--*', 'LineWidth', 2)
title('Impulse Response')
xlabel('Time [sec]')
ylabel('Response')
grid on
```