Assignment 1

SAURAV KUMAR

29 January, 2024

1 Question 1

1.1 part 1

Plotted the required graph using normal and the method suggested i.e. using specifying delays.

```
    s = tf('s');                                  % creating transfer function var

%normal method:
%G_original = ((0.1*exp(-2*s))+s)/ (s^2+4*s+0.1);
%[out1, time1] = step(G_original, 10);
%bode(G_original)
%step(G_original)


% method suggested:
num = 0.1;
den = [1 4 0.1];
P = tf(num,den,'InputDelay',2)
num1 = [1 0];
P1 = tf(num1,den,'InputDelay',0)
P2= P+P1;
step(P2)
bode(P2)
hold off
```

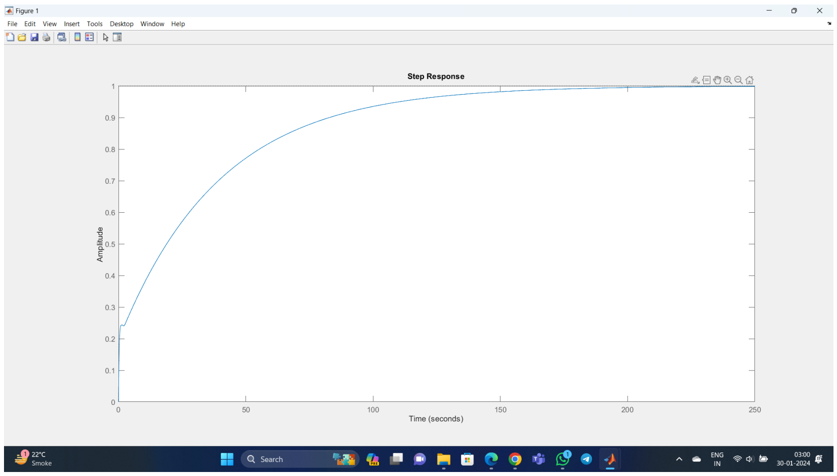

Figure 1: step response

1.2 part 2

```
    % Define the plant transfer function
numerator = [0.1 1];
denominator = [1 4 0.1];
G = tf(numerator, denominator, 'InputDelay', 2);

% Define the actuator saturation limits
saturation_limits = [0, 10];

% Bump test input
t_bump = 0:0.01:10;
u_bump = 0.5*sin(2*pi*1*t_bump) + 0.2*sin(2*pi*5*t_bump);

% Simulate the response with actuator saturation
[y_bump, t_bump_actual, x_bump] = lsim(G, u_bump, t_bump, 'saturation', saturatio

% Plot the bump test input and output
figure;
subplot(2,1,1);
plot(t_bump_actual, u_bump);
xlabel('Time');
ylabel('Input');
```

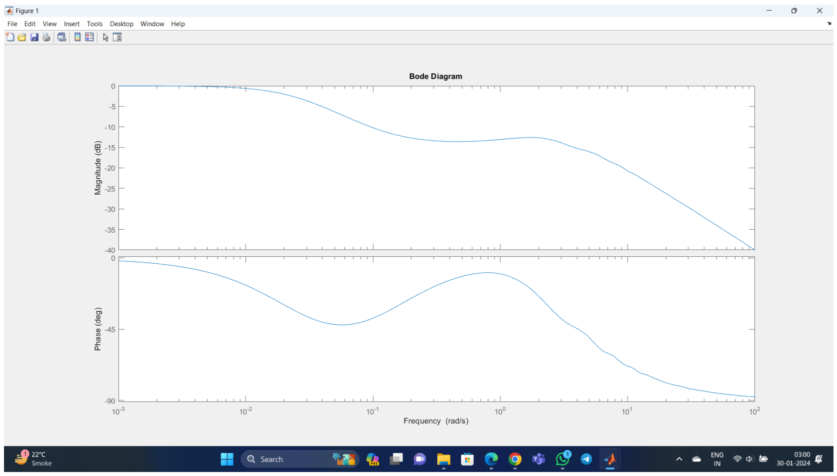

Figure 2: bode plot

```
title('Bump Test Input Signal');
grid on;

subplot(2,1,2);
plot(t_bump_actual, y_bump);
xlabel('Time');
ylabel('Output');
title('Plant Response to Bump Test');
grid on;
```

1.3 part3

Models are generally good having a high no.of pole and high zeroes. Delay is present in the transfer function itself. and it is a 3 pole and 2 zero transfer function. justification lies in the polynomial expansion of the exponential part of the transfer function.

1.4 part4

attached as a figure. 98 percent matched in the case of 3 pole 2 zero.

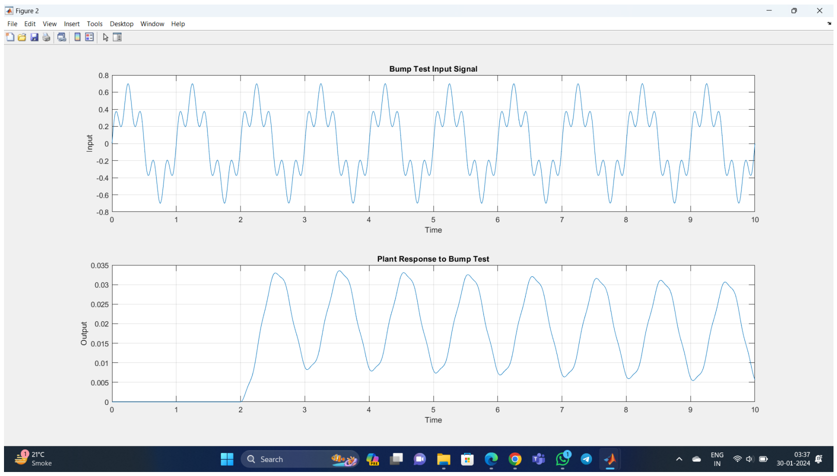

Figure 3: bump test

1.5 part4

attached as a figure. 98 percent matched in the case of 3 pole 2 zero.

2 Question 2

steps for calculation involved have been described with codes.

2.1 part 1

values I got for models:

System model 1 parameters – K = 1.000000,Tar = 4.124999
system Model 2 parameters – a = -0.372842, L = 1.669956

```
%{
Two parameter approximate models can be used to approximate this transfer function
as taught in the lecture:
1) Gain average resident time :
   G1 = K / (1+s*Tar)           --parameters: K, Tar
2) Integrator with time delay :
   G2 = -a * exp(-sL) / s*L     --parameters: a, L
%}
```

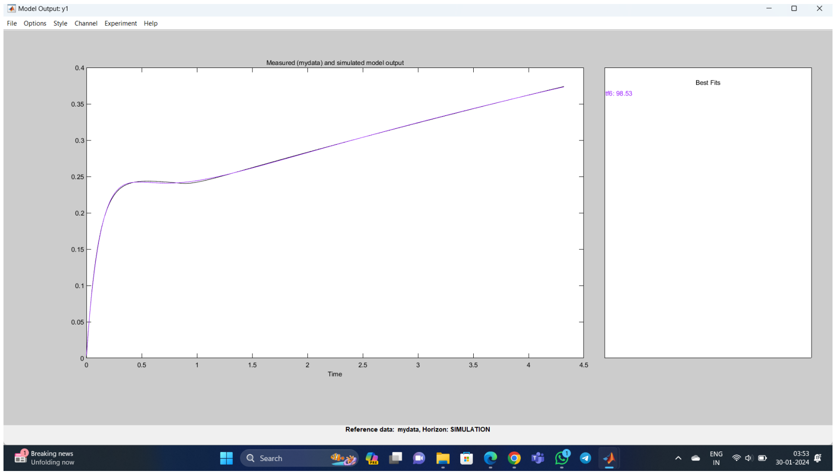

Figure 4: system id for q1 part 4

```
% system model 1:

s = tf('s');                                    % creating trasfer function variable
G_original = 1 / (s+1)^4;                       % defining actual trasfer function
[out1, time1] = step(G_original, 25);           % step response of original trasfer
unitstep1 = ones(size(out1));                   % creating unit step of only ones ar
delta_t = time1(2) - time1(1);                  % taking time difference to spply fu
K = out1(end);                                  % final value of output must be equa
A_0 = sum((out1(end) - out1) * delta_t);        % getting the integral like specifie
Tar = A_0 / K;                                  % getting average resident time by t
G_new_1 = K / (1+ s*Tar);                       % new trasfer function
[out2, time2] = step(G_new_1, 25);              % step response of new TF
X = sprintf('K = %f | Tar = %f',K, Tar);        %creating string X with parameters o
disp(['System model 1 parameters --> ', X])     %displaying in command window




% System model 2:
```

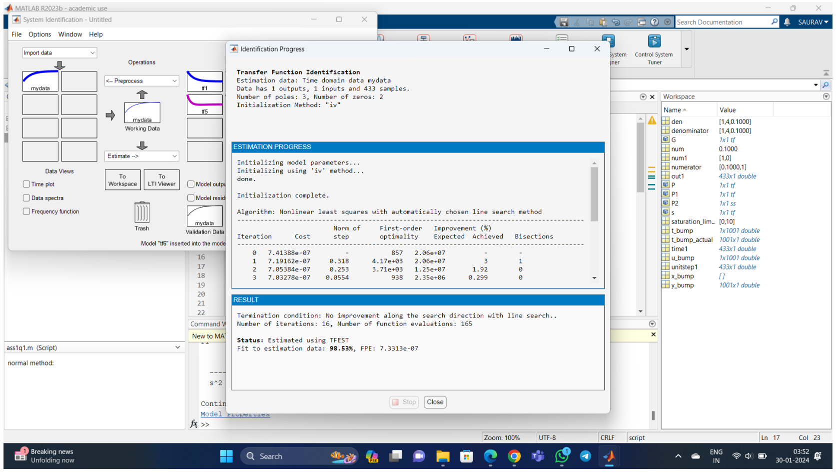

Figure 5: system id for q1 part 4

```
grad_out1 = gradient(out1);                       % gradient of step response of or
[max_changeinout1, index] = max(grad_out1);       % finding maximum value of gradie
maximum_slope = max_changeinout1 / delta_t;       % finding maximum slope as divinc
max_slope_time_stamp = index * delta_t;           % getting time stamp of maximum s
max_slope_out1_coordinate = out1(index);          % We now have the line in the for
a = max_slope_y_coordinate - maximum_slope * max_slope_time_stamp; % getting the v
L = -a / maximum_slope;                            % parameter L ( USING NEgaITVE V
G_new_2=(-a*exp(-L*s))/(L*s);                      % new transfer function
[out3,time3]= step(G_new_2, 15);                   %step response of second model
X = sprintf('a = %f | L = %f',a, L);               % printing as string
disp(['system Model 2 parameters --> ', X])        % display

% plot(time1, out1)
% hold on
% plot(time2, out2)
% hold on
% plot(time3, out3)
% hold off

% bode plots of all models
bode(G_original)
hold on
```

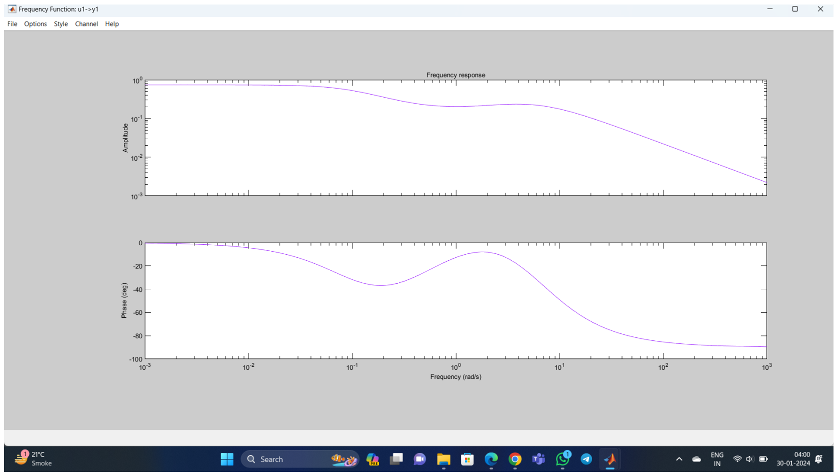

Figure 6: bode plot of model for q1 part 4

```
bode(G_new_1)
hold on
bode(G_new_2)
hold off

title('Step Response in one plot');
legend('original tf', 'system model1', 'system model2');
saveas(gcf, 'comparisonplot.png');               % Saving comparison plot
```

2.2 part2

system identification: plots are attached and system identification done using matlab tool. obtained values attached as photos.

2.3 part3

it can be in comparison plot and step response plot. It is very clear after running the code.

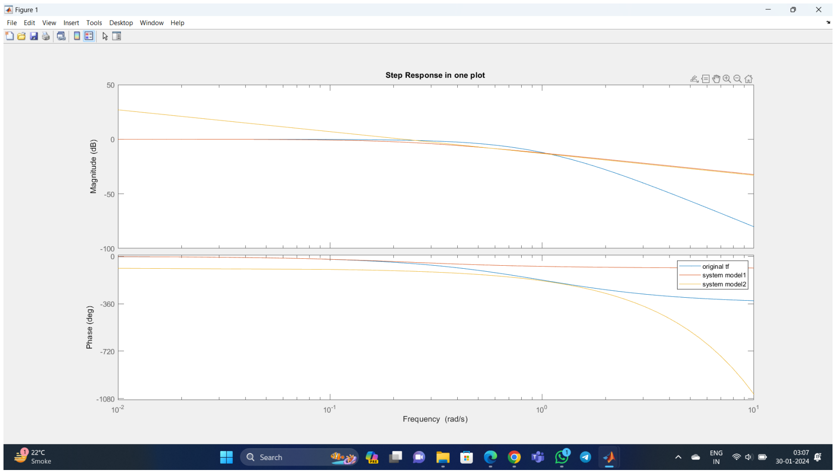

Figure 7: comparison plot

2.4 part4

steps involved have been described with codes.

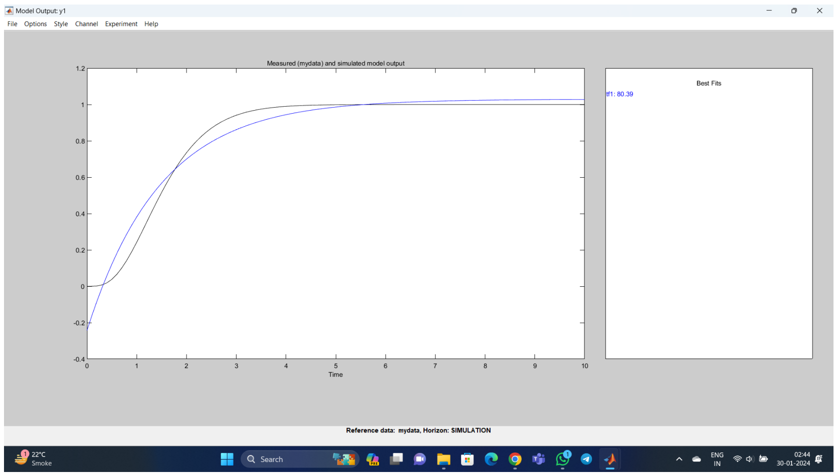

Figure 8: system identification step response comparison

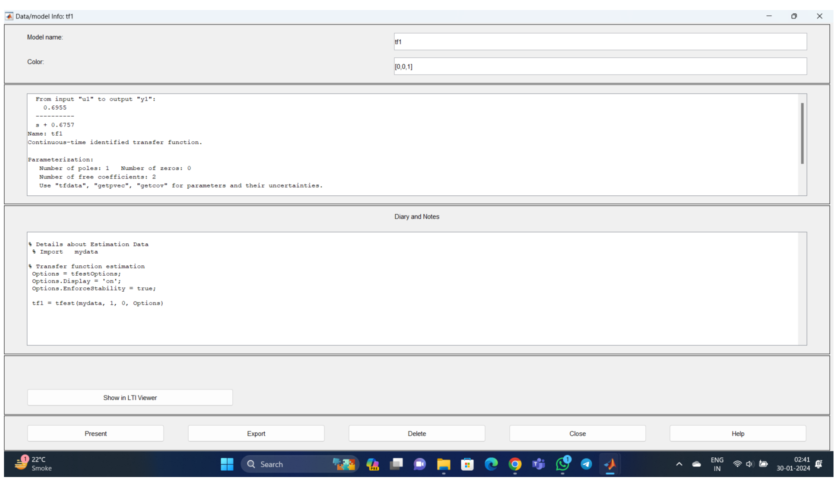

Figure 9: comparison plot

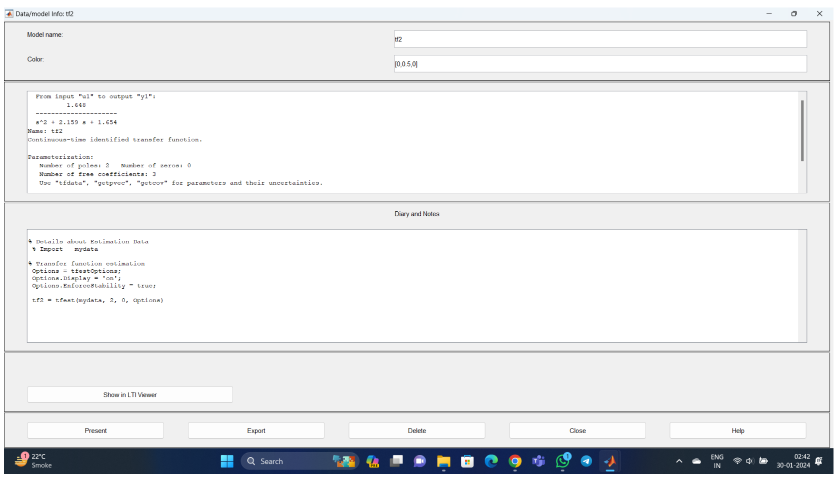

Figure 10: system identification with 2 pole