Lab1 Control

933655698 Kengo Akano

336531876 Eve Haddad

5. In-Lab Process.

Every question must be implemented using appropriate Matlab commands, and plots must be saved and presented at the post-lab report.

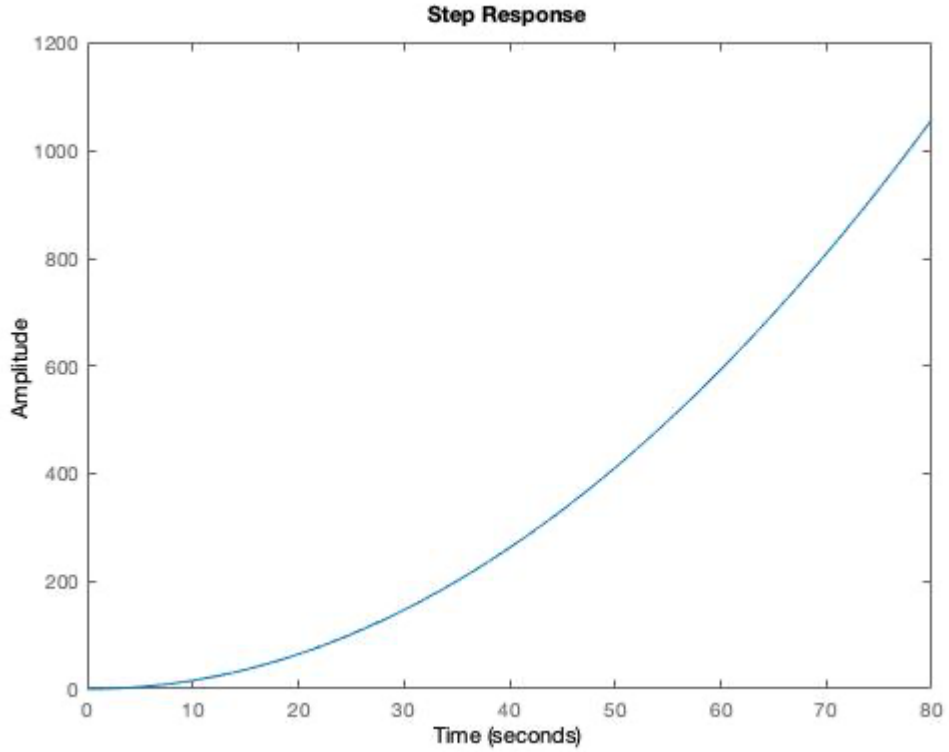
Note – every variable name is case sensitive!!

5.1. Open Loop.

Use the instructions in section 2 to create a new .m file.

Define the following plant: $P(s)$
Plot the plant's open-loop response to a step input. Is the system stable? How can you tell? Explain.

```
s=linspace(-20,20);
p = tf([1],[1,3,0,0]);
step(p);
```



The system is not stable since it has a pole at zero, moreover looking at the graph we observe that the open loop response to a step input explodes at zero.

Convert this transfer function into a state-space representation. Does the outcome match your expectations made in the pre-lab? Explain why.

```
sys1=ss(p);
A=[-3 0 0; 1 0 0; 0 1 0];
B=[1;0;0];
C=[0 0 1];
D=[0];
```

```
sys1 =

  A =
       x1  x2  x3
   x1  -3   0   0
   x2   1   0   0
   x3   0   1   0

  B =
       u1
   x1   1
   x2   0
   x3   0

  C =
       x1  x2  x3
   y1   0   0   1

  D =
       u1
   y1   0

Continuous-time state-space model.
```

Find the eigenvalues of the A matrix. Compare them against the eigenvalues of you pre-lab A matrix.

```
e = eig(A);
```

```
A =

    -3     0     0
     1     0     0
     0     1     0


e =

     0
     0
    -3
```

The eigenvalues of A are 0, 0 and 3 similarly as in the pre lab.

Complete the state-space representation's necessary matrices, and convert the plant back to a transfer-function form. Did you get the same result?

```
[num, den] = ss2tf(A,B,C,D);
G = tf (num, den);
```

```
G =

        1
  -----------
  s^3 + 3 s^2

Continuous-time transfer function.
```

we get the same result as the original plant.

5.2. Closed Loop.

Define the following compensator: $C(s)$

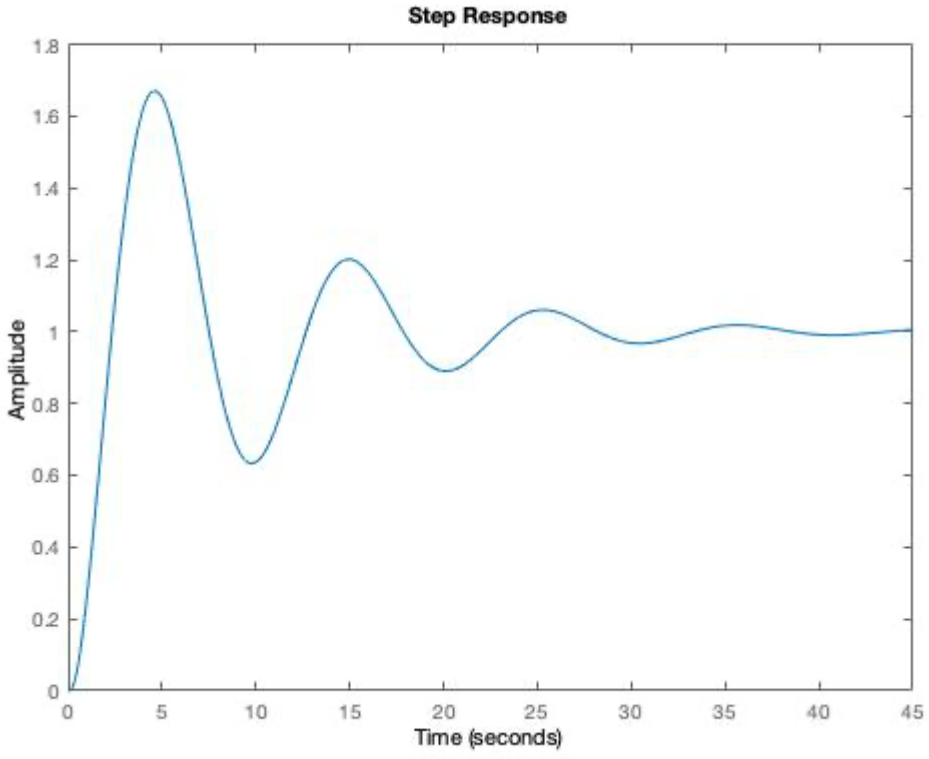
Create a new variable named sys_cl, in which you should store the closed

$y(s)$

loop transfer-function, $r(s)$ (according to the diagram in section 4).

Plot the closed-loop system step response. Is it stable? Explain.

```
c = tf([1.15,1],[0.15,1]);
sys_cl = p*c/(1+p*c);
step(sys_cl)
```

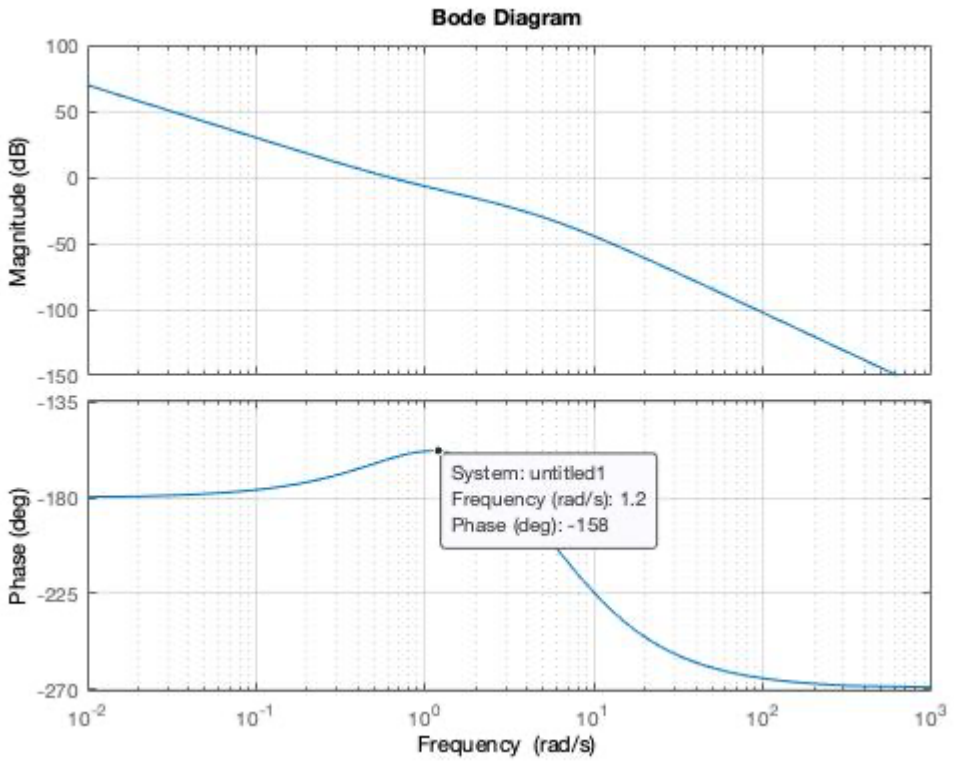


The system is stable since it has negative real values, moreover it stabelizes at approximately 1.

5.3. Plots.

Create a Bode plot of the open-loop system with the compensator, C*P. Mark the minimum stability margins by right-clicking on top of the figure. Add a grid to the plot.

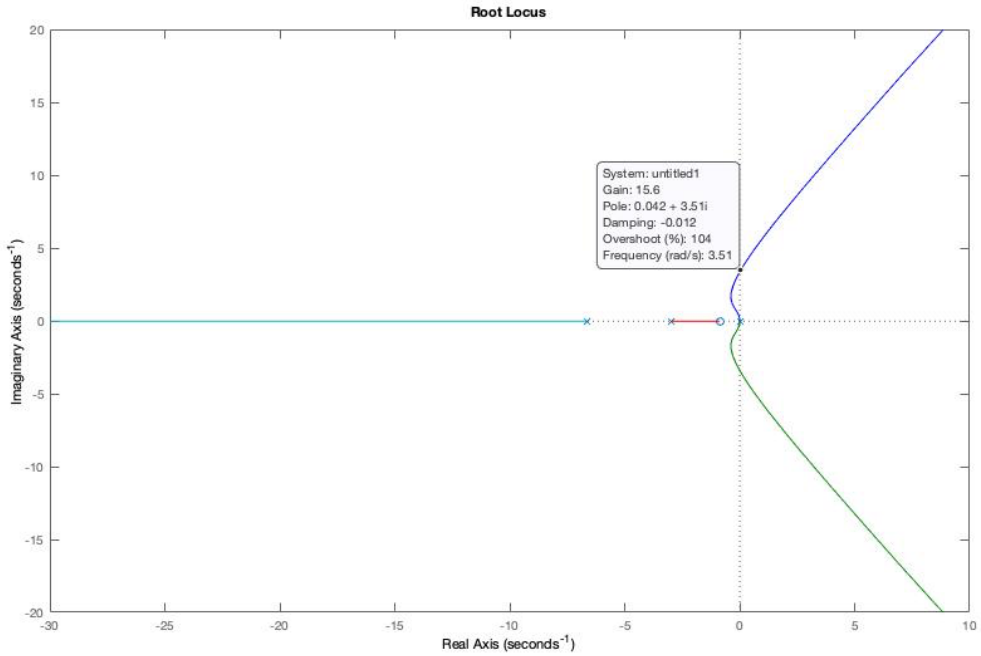
```
bode(c*p);
grid;
```



Create a root-locus plot for the system, together with the compensator.

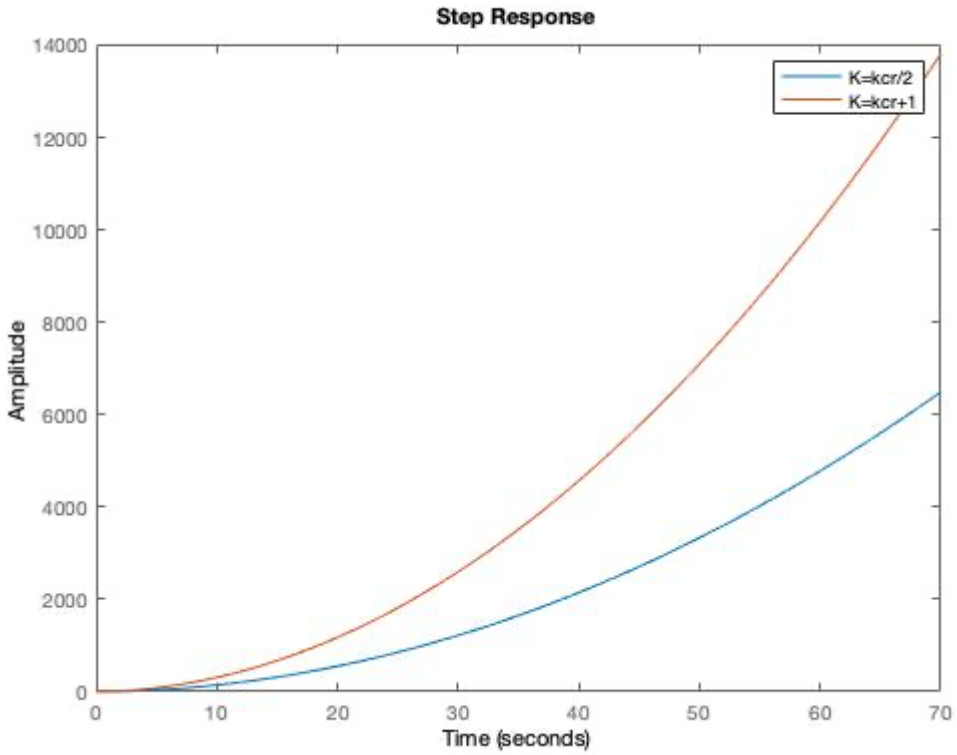
```
rlocus(c*p);
```

Mark a point on the curve and drag it to the point where the critical gain is.



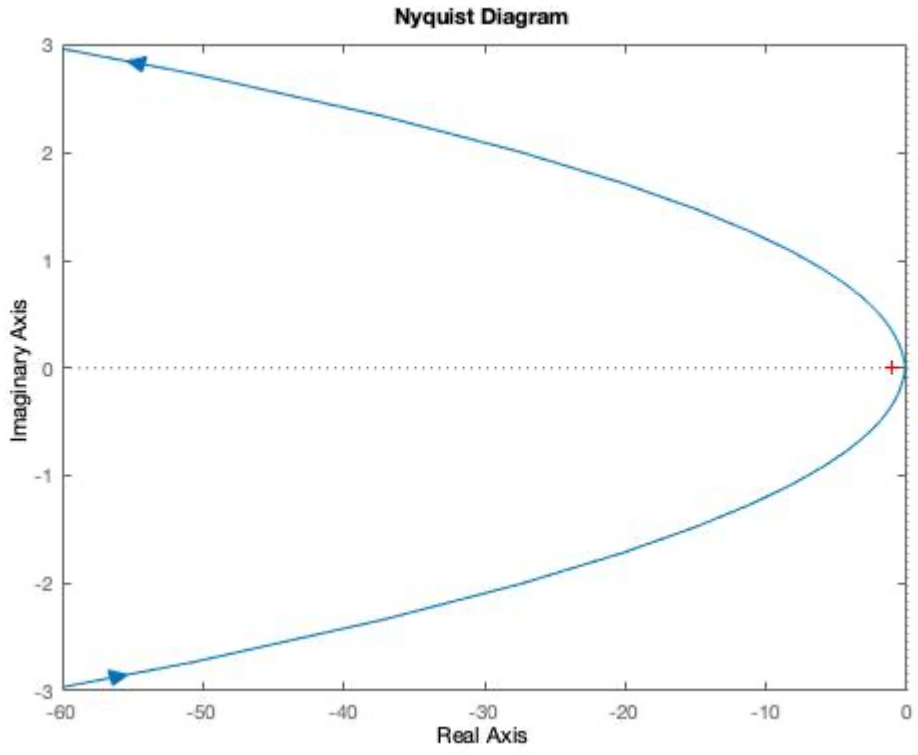
Add a gain K to the system so that the open-loop will be K*C*P. Plot the system response to a step for K=(k_{cr}/2) and for K=(k_{cr}+1) on the same figure, without using the "hold on" command. Explain the results.

```
kcr = 15.6;
K1=kcr./2;
K2=kcr+1;
figure();
step(K1*c*p);
%%legend();
hold on
step(K2*c*p);
legend('K=kcr/2','K=kcr+1');
hold off
```



Create a Nyquist plot of the C*P system. Explain the difficulty of assessing the system stability using the results you got.

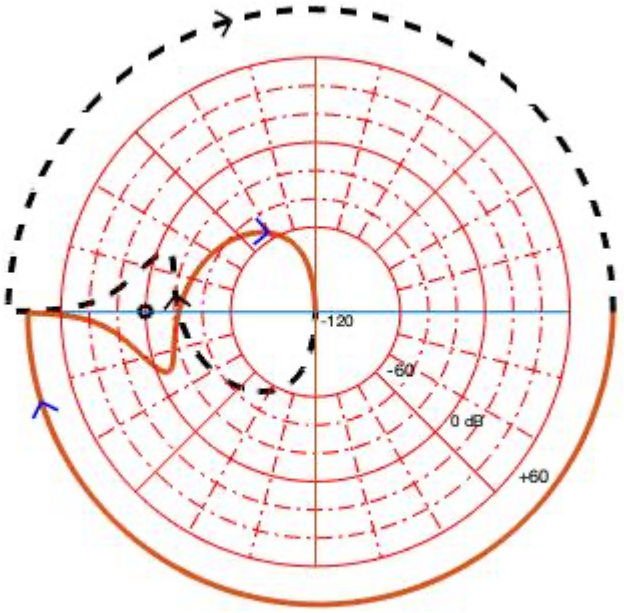
```
nyquist(c*p);
```



Assessing the system stability is difficult here since the values are very high, that's why we prefer to use the log scale

Plot the same system using the command nyqlog() (you must copy the function from the main MATLAB directory to your folder!!).

```
nyqlog(c*p);
```



Explain how to obtain the stability of this system using the Nyquist criterion,

to check the stability of a system we need to check if the -1 point is encircled by the trajectory, we can see in our nyquist plot that it is not the case, so the system is stable.

and do the same again for K*C*P with K=(k_{cr}+1).

In this case, the -1 point is encircled, so the system is not stable as expected since it is plotted for K higher than the critical gain.

5.4. Simulink.

Use the instructions in section 3 to open a new Simulink model, and create the following open-loop diagram of the system, made of:

Step input.
Gain.
Transfer function.
Mux.

Scope.
Clock.
3 "To workspace" blocks.

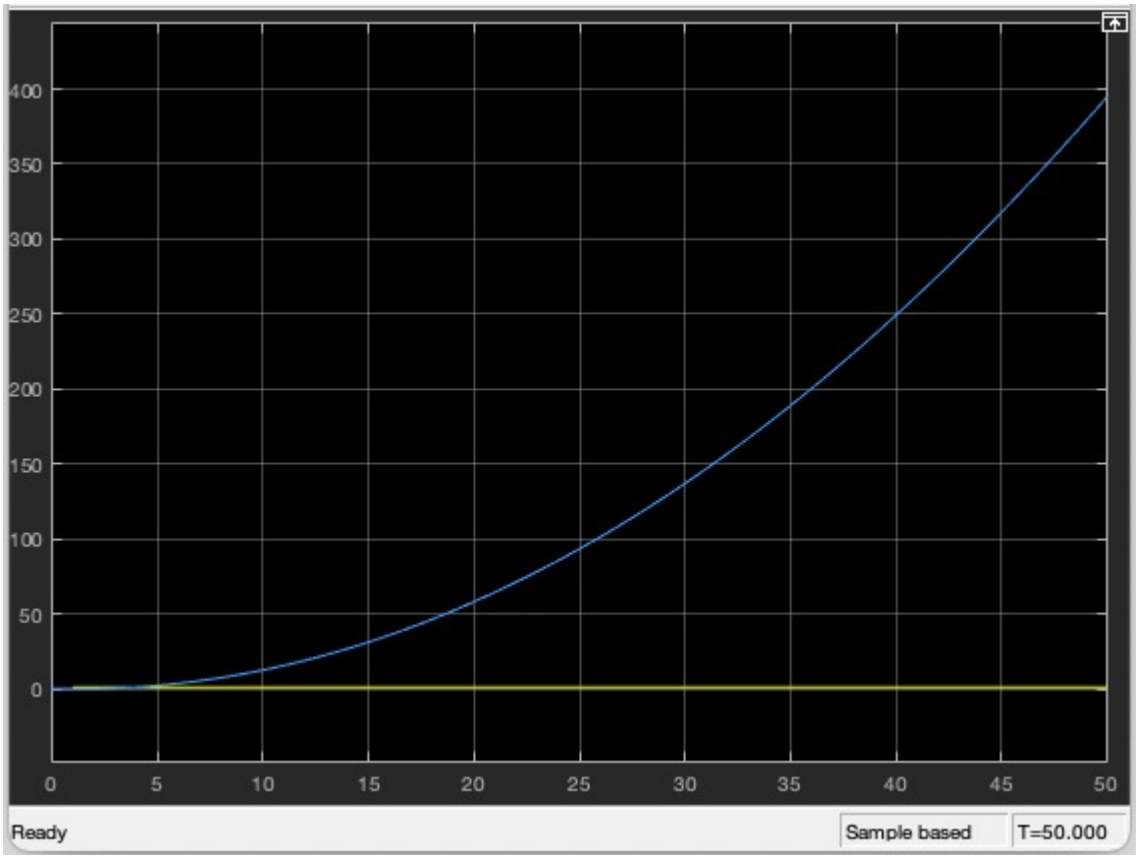
Set the "To workspace" blocks so that the exported variable names match the ones in the figure on the right, with an array-type output.

In addition:

insert the letter K into the gain block and define in the Matlab workspace a variable named K with the value 1. Open the scope and export the outputs to workspace under the name: SysResponse. Turn to section 3.3 for help.

Save the model under the name: "open_loop.slx" and play it with a simulation time of 50 seconds.

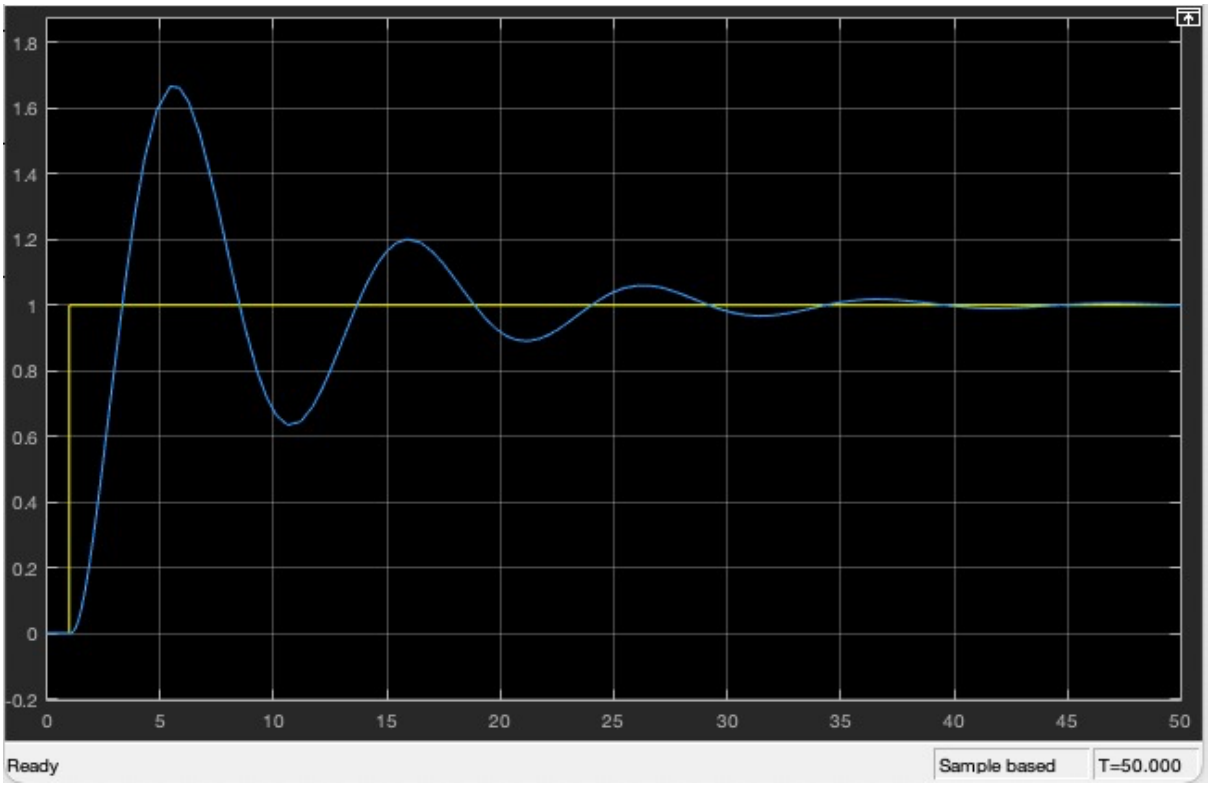
Plot the input and output results vs. time, on the same figure.



Edit your model so that it depicts a closed loop in the following manner:

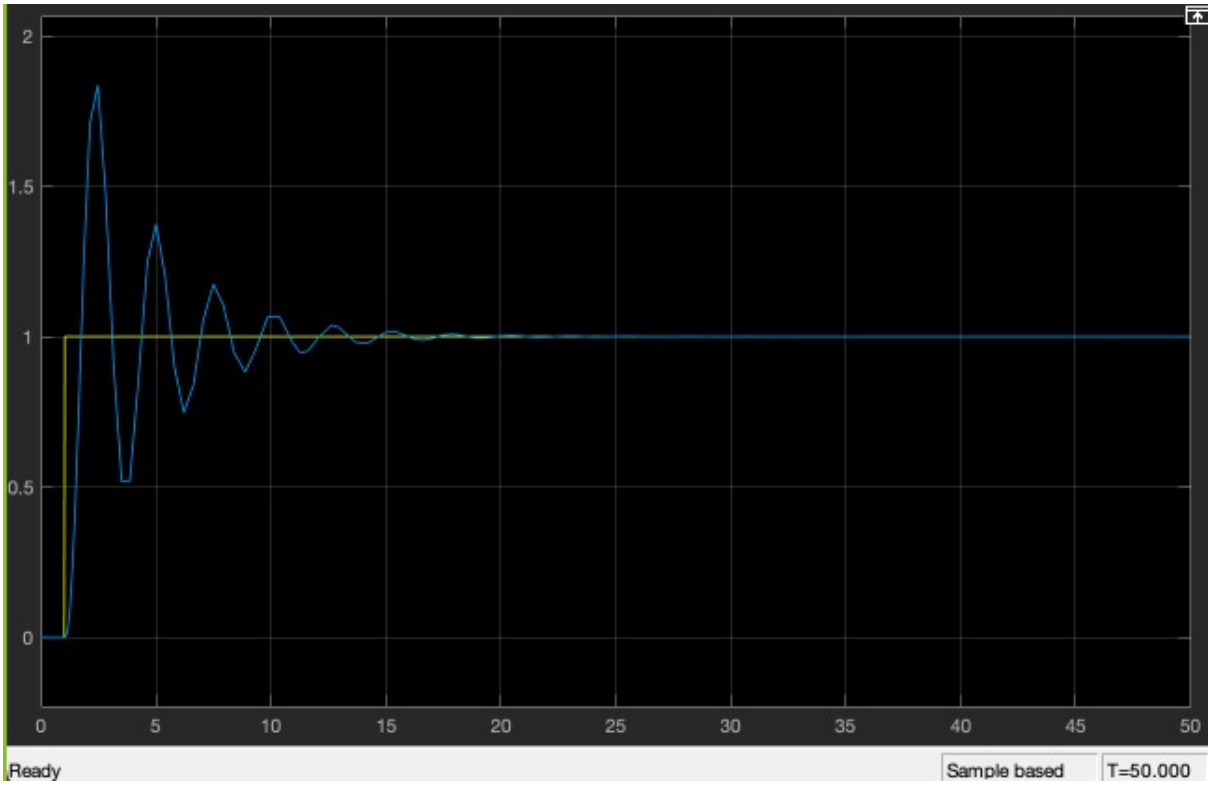
Save the model under the name: "closed_loop.slx" .

Run the model system with K=1 (through the workspace!) and a simulation time
of 50 seconds and plot the results.

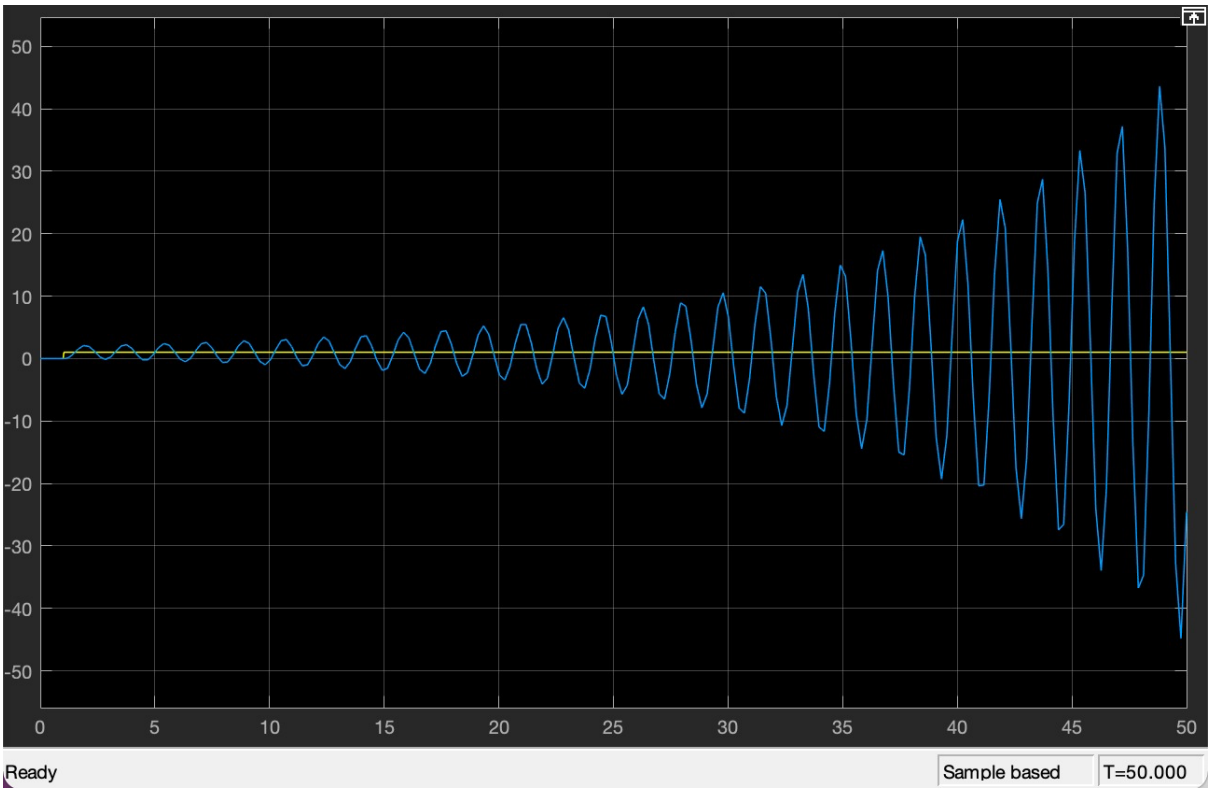


Repeat this with the 2 other K values mentioned in section 5.3.

K = 7.8



K= 16.6



Discuss the results and compare them to the step simulation done inside Matlab, are they identical?

Remove the compensator block, and change the plant so that it matches the following transfer function: $P(s) \quad 1$. Save and exit the model. $2s^3 3s^2 4s$

5.5 Code following

Write the following code on matlab and explain what each line does.
Use the matlab help section if needed

```
x=0:1:4;
num=x(2);
den=[x(3:5), x(1)];
sys=tf(num,den);
for i=1:3,
    figure(i);
    K=i*1.5;
    [y,t]=step(feedback(K*sys,1));
    sim('closed_loop.slx');
    plot(SysResponse(:,1),SysResponse(:,3),'--r');
    hold on
    plot(t,y,':b','LineWidth',1.5);
    grid on
    hold off
end
```

line 1 - create an equally spaced vector 0,1,2,3,4
line 2 - create a variable num1 which is the second element of the array
lien 3 - create a variable den which correspond to the denominator of [2 3 4 0]
line 4 - create a transfer function num/ den
line 5 - for loop with i = 1, 2, 3
line 6 - open a new figure
line 7- create a new gain K by multiplying the index by 1.5
line 8 - create a closed loop system by adding a negative feedback oop step response
line 9 - simulation of he dynamic system closed loop
line 10 - plot the system response from simulink
line 11 - keep holding to plot in the same figure
line 12 - plot the response of matlab
line 13 - display the grid on the plot
line 14 - stop holding
line 15 - End the for loop

- Add a legend to figure 1 by using a Matlab command, and to figure 2 by using the legend icon at the top bar in the figure window. The legend has to correspond to the data it represents.

```
x=0:1:4;
num=x(2);
den=[x(3:5), x(1)];
sys=tf(num,den);
for i=1:3
 figure(i);
 K=i*1.5;
 [y,t]=step(feedback(K*sys,1));
 sim('closed_loop.slx');
 plot(out.SysResponse(:,1),out.SysResponse(:,3),'--r');
 if i==1
     legend('SysResponse1(K=1.5)','SysResponse3(K=1.5)');
 end
 hold on
 plot(t,y,':b','LineWidth',1.5);
 grid on
 hold off
end
```

- Use the "Show plot tools" icon at the top bar in order to change the line width of the red curve to 2.
- Again, what are the differences between the outputs? Try and figure out what is the reason this happens,
  we can see that there are differences between the outputs of simulink and the matlab responses, we observe a certain delay but the principle behaviour is the same.

- Save only figures 1 and 2.
  fig 1

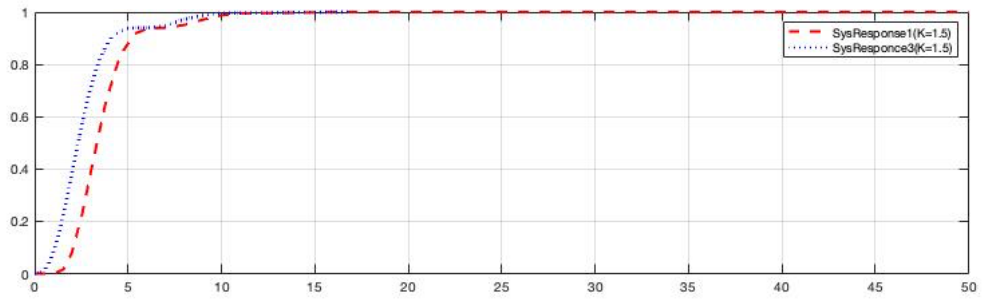
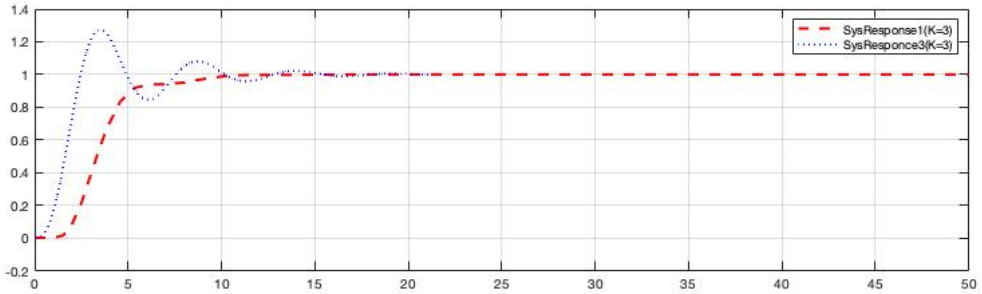


fig 2


SysResponse1(K=3)
SysResponce3(K=3)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
20
25
30
35
40
45
50