Sessions 2, 3 & 4 - Introduction to Control - 2023/2024

Authors:
Lourenço Faria
Gustavo Toste
Adriano Cardoso

Date:
03/12/2023

Contents


```
clc; clear all; close all;
```

Definition of variables

```
G = 9.8; %m s^-2
M = 1; %kg
K_t = 3.575E-5; % N s^2 rad^-2
Z0 = 2; %m
```

Q2.4

We know the initial position of the drone. The remaining variables at equilibrium are the drone's blades angular velocity and the derired anglular velocity.

```
omega0 = sqrt(G*M/K_t); %rad s^-1

disp(omega0); %rad s^-1
```

```
   523.5703
```

U0 is equal to omega0:

```
U0 = omega0; %rad s^-1

disp(U0); %rad s^-1
```

```
   523.5703
```

The angular velocity in revolutions per minute (rpm) will be:

```
omega_0_rpm = omega0/(2*pi)*60; %rot min^-1

disp(omega_0_rpm); %rot min^-1
```

```
    4.9997e+03
```

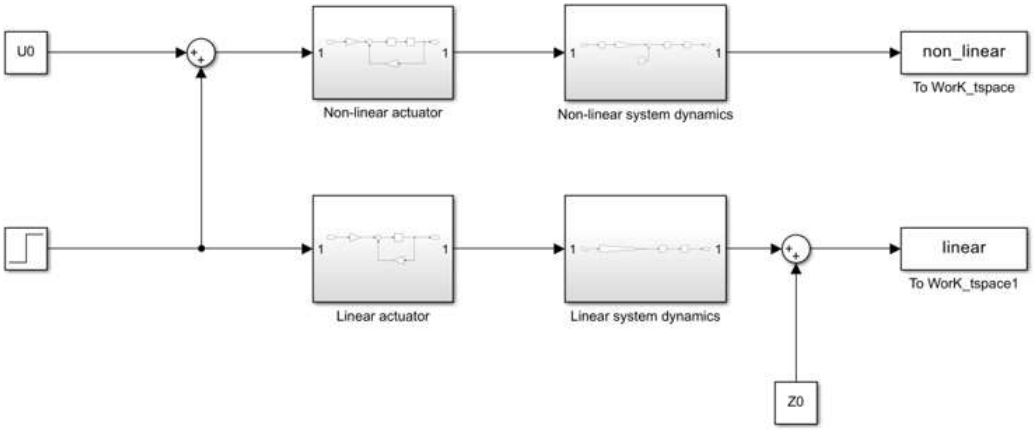
Q2.5

```
% Simulation Step Size
StepSize = 1e-4;

% Simulation Time
SimTime = 20;
```

```
%Rotor array
w_dot_raw_array = [100 1000 5000];
w_dot_array = (w_dot_raw_array./60).*2.*pi;
```

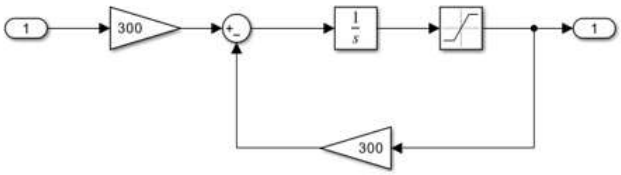
Simulink Model

```
open_system('lab_2.slx');
```



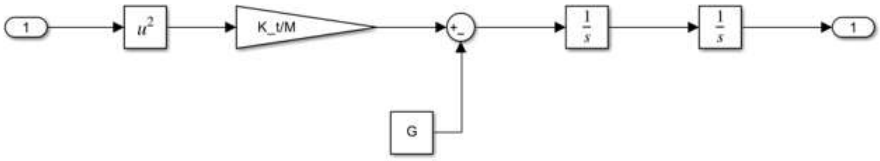
Non-Linear Actuator subsystem

```
open_system('lab_2/Non-linear actuator');
```



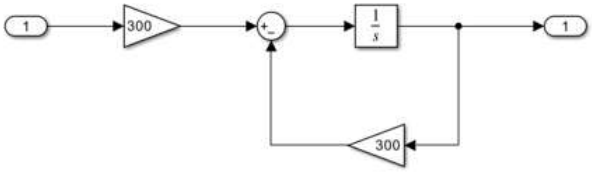
Non-linear System Dynamics subsystem

```
open_system('lab_2/Non-linear system dynamics');
```



Linear Actuator subsystem

```
open_system('lab_2/Linear actuator');
```

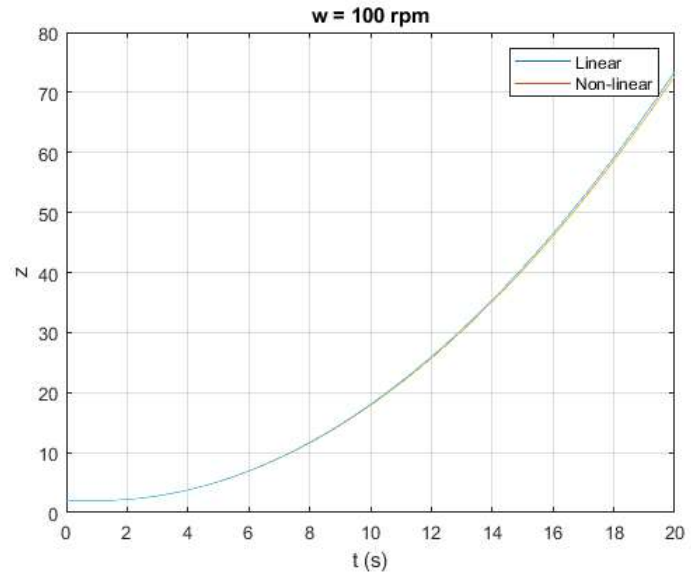


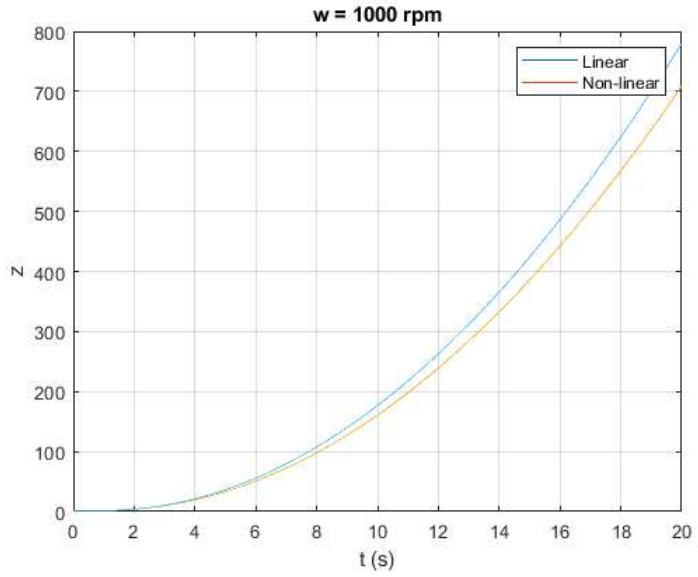
Linear System Dynamics subsystem

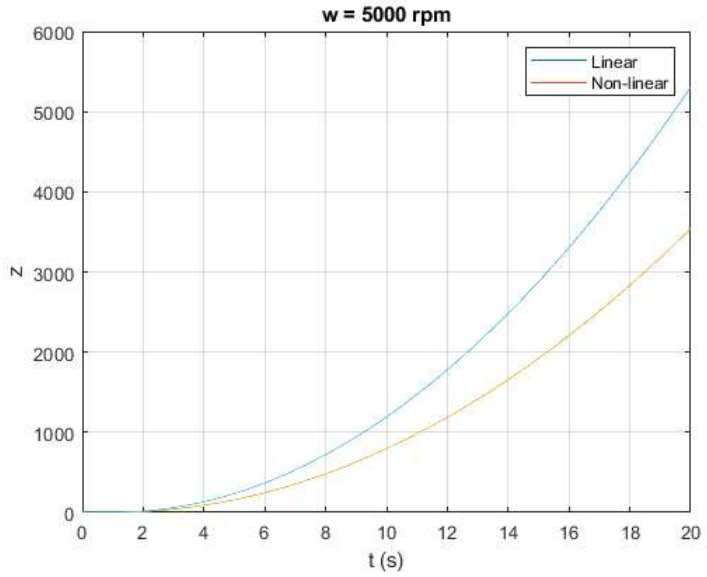
```
open_system('lab_2/Linear system dynamics');
```


```
% Initialization of matrices to store different simulation conditions
time_array = zeros(SimTime/StepSize + 1, 1);
linear_values_array = zeros(SimTime/StepSize + 1, length(w_dot_array));
non_linear_values_array = zeros(SimTime/StepSize + 1, length(w_dot_array));

% Determination of de response of the system for different blade
% angular rotations

for i = 1:length(w_dot_array)
        w_dot = w_dot_array(i);
        sim('lab_2.slx');
        time_array(:, i) = linear.time;
        linear_values_array(:, i) = linear.signals.values;
        non_linear_values_array(:, i) = non_linear.signals.values;
end
```

```
% Plot results

for i=1:length(w_dot_array)
    figure; hold on; title("w = " + num2str(w_dot_raw_array(i)) + " rpm");
    plot(time_array, linear_values_array(:, i));
    plot(time_array, non_linear_values_array(:, i));
    xlabel('t (s)');
    ylabel('z');
    legend('Linear', 'Non-linear');
    grid on;
    box on;
end
```



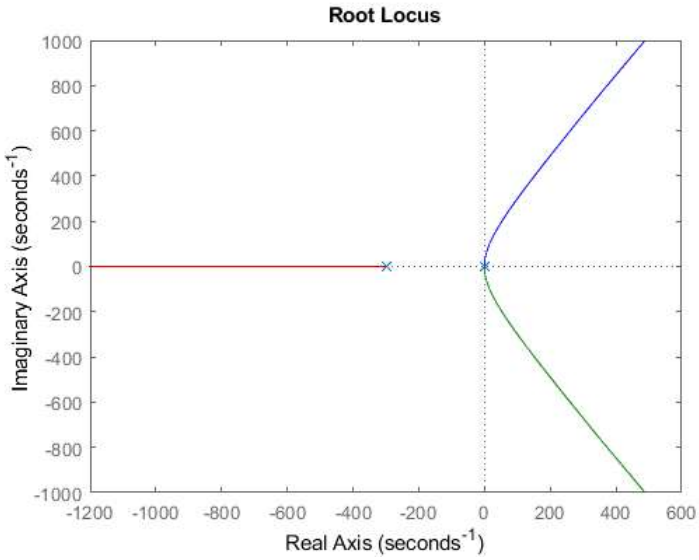




The first plot shows us that the linear system can follow the non linear up to a certain degree if the simulation time is considerably small. It is 'ahead' of the non linear curve the whole time. The other plots, however, let us conclude that for high inputs of u, the linear system no longer makes for a good dynamic system because it deviates from the non linear.

Q3.4

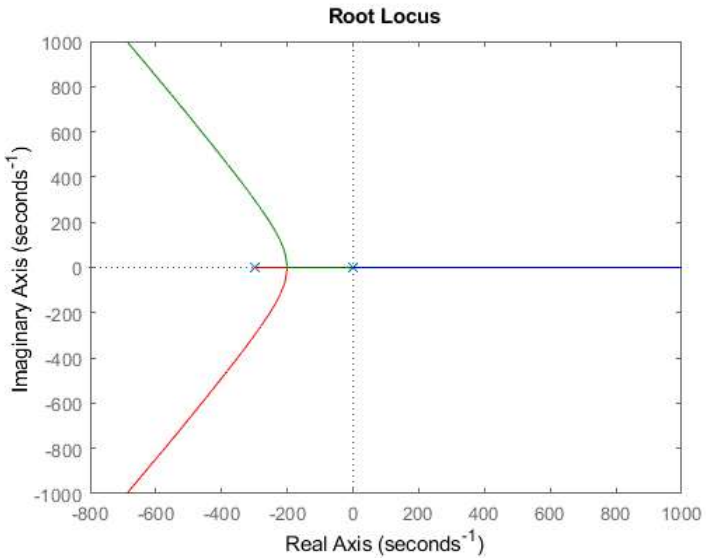
```
hold off;

K_test = 1; %check for a positive gain K

rlocus(tf(K_test, [1 300 0 0])); %1
hold off;
```



We can cleary see that the poles for the closed-loop P system represent a non stable system. There is a root at -300 and another with multiplicity 2 at the origin. The poles related to the latter diverge into the imaginary field for positive gains, meaning that the response of the system will have oscilations. Additionally, these two poles are always equal or greater than 0, meaning that the system is naturally non stable. The other pole is always negative and real.

Q3.5

```
K_test = -1; %check for a positive gain K

rlocus(tf(K_test, [1 300 0 0])); %1
hold off;
```



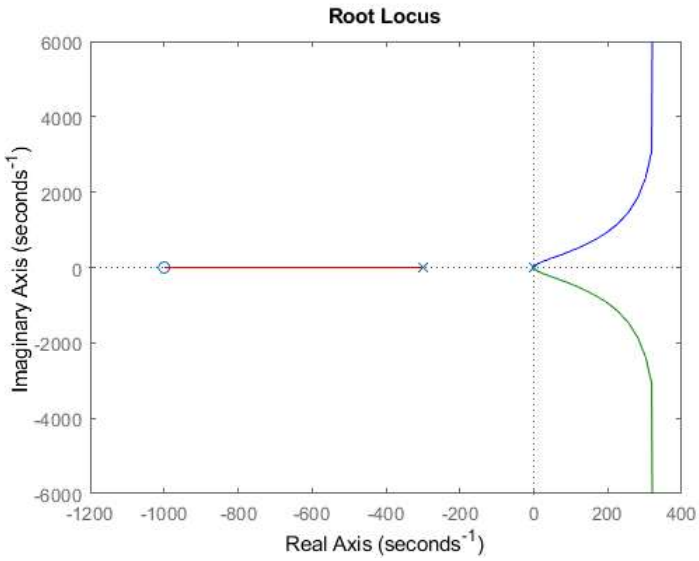
By assuming the gain of the transfer function to be negative, we can see that one of the poles related to the zero with multiplicity 2 stays on the real axis and remains equal to or greater than 0. The other one is negative and real for small gains and diverges to the imaginary numbers for big gains. The remaining one is always negative and also diverges to the imaginary numbers for big gains. Since there is always a pole in the right side of the coordinate referential, the system is non stable.

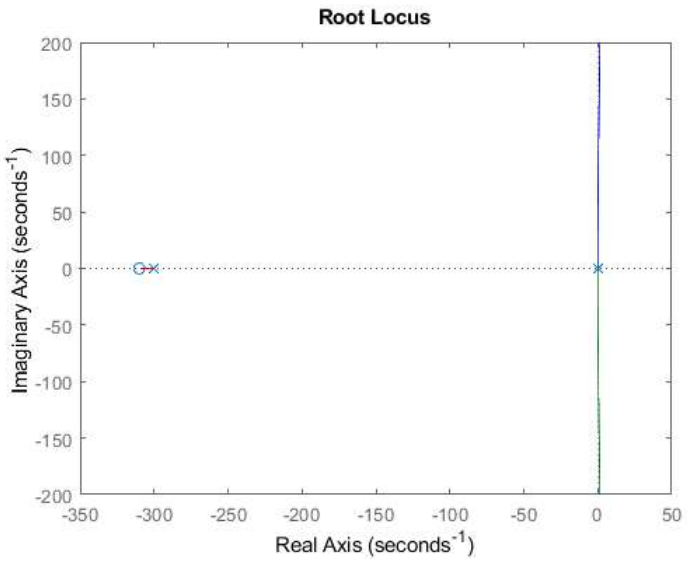
Q3.7

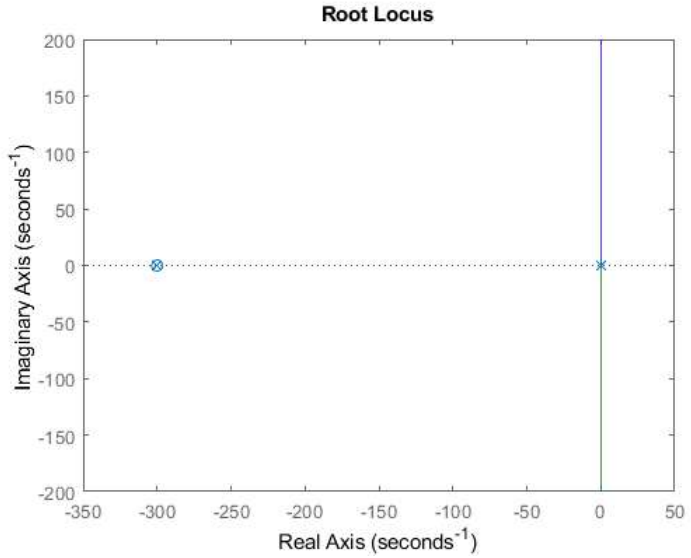
We can use different points to anaylise the root-locus technique. We shall set a zero close to each root, between and far away.

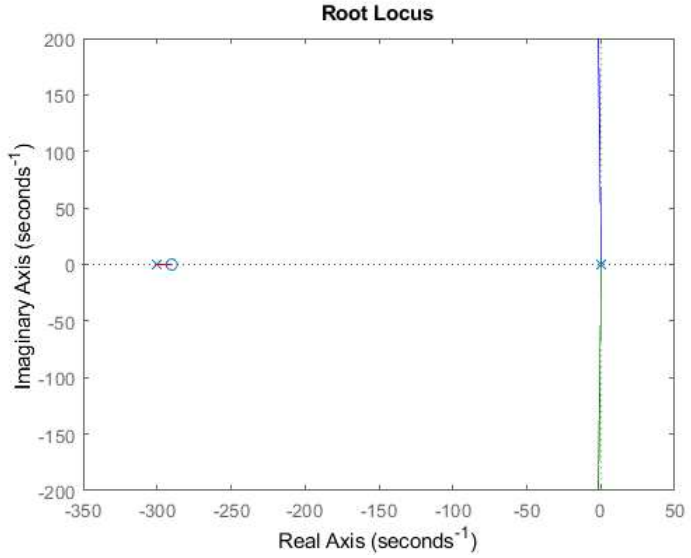
```
z_1 = 1000; %left and far away
z_2 = 310;  %left and near
z_3 = 300;  %root mult. 1
z_4 = 290;  %between and near
z_5 = 150;  %between
z_6 = 100/3;%between
z_7 = 10;   %between and near
z_8 = 0;    %root mult. 2
z_9 = -10;  %right and near
z_10 = -700; %right and far away

z_array = [z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_10];
```

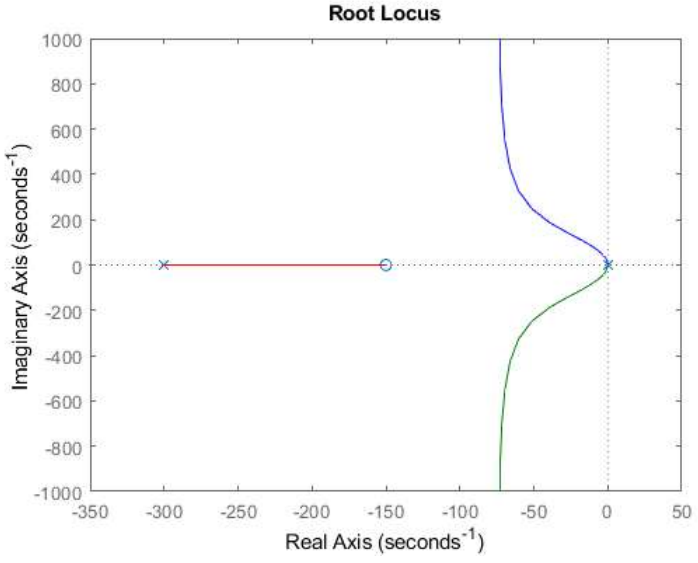
```
% K = 1 for testing with a positive gain

for i=1:length(z_array)
    figure;
    z = z_array(i);
    rlocus(tf([1 z],[1 300 0 0]));
    hold off;
end
```

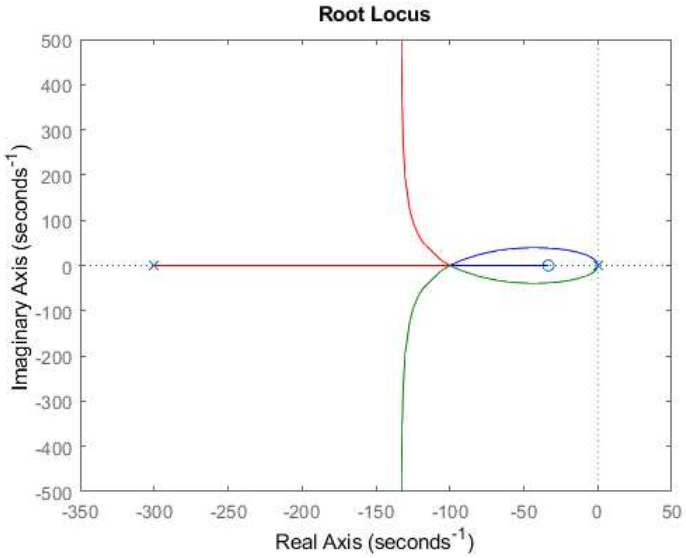


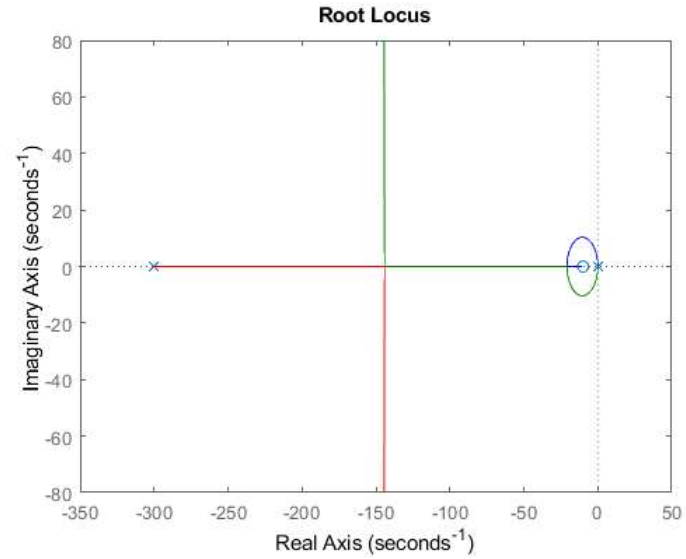


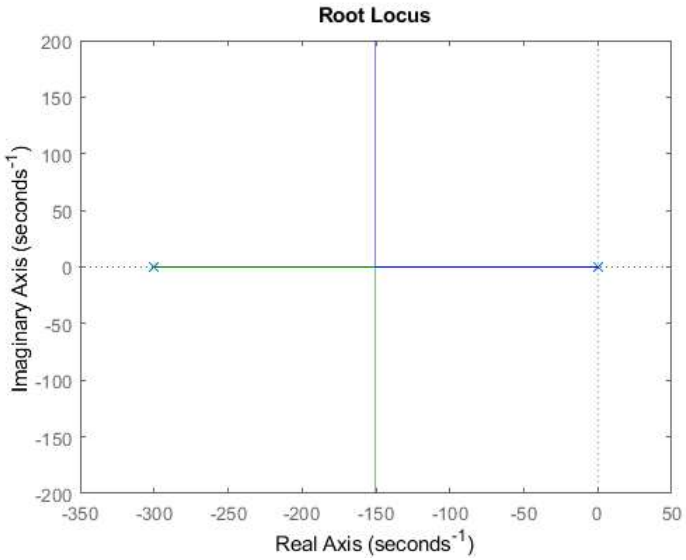

Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
200
150
100
50
0
-50
-100
-150
-200
-350
-300
-250
-200
-150
-100
-50
0
50

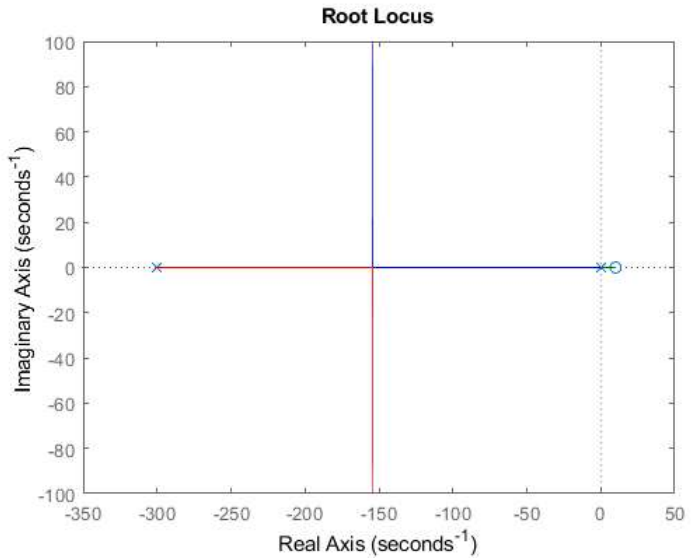

Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
200
150
100
50
0
-50
-100
-150
-200
-350
-300
-250
-200
-150
-100
-50
0
50

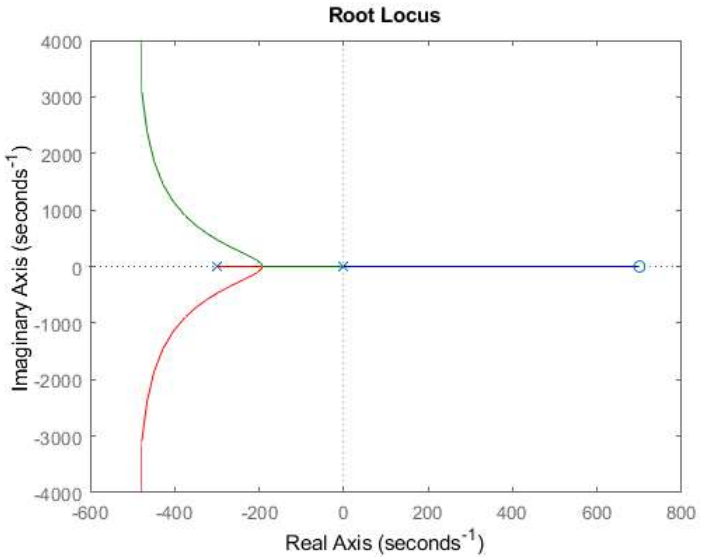

Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
1000
800
600
400
200
0
-200
-400
-600
-800
-1000
-350
-300
-250
-200
-150
-100
-50
0
50


Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
500
400
300
200
100
0
-100
-200
-300
-400
-500
-350
-300
-250
-200
-150
-100
-50
0
50


Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
80
60
40
20
0
-20
-40
-60
-80
-350
-300
-250
-200
-150
-100
-50
0
50


Root Locus
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
200
150
100
50
0
-50
-100
-150
-200
-350
-300
-250
-200
-150
-100
-50
0
50





When the zeros are equal to the poles, we can see that, from a specific gain, the trajectory of the poles assumes a constant real number. If that is not the case, for high gains, the trajectory converges to a real number. The imaginary coordintate may or may not converge.

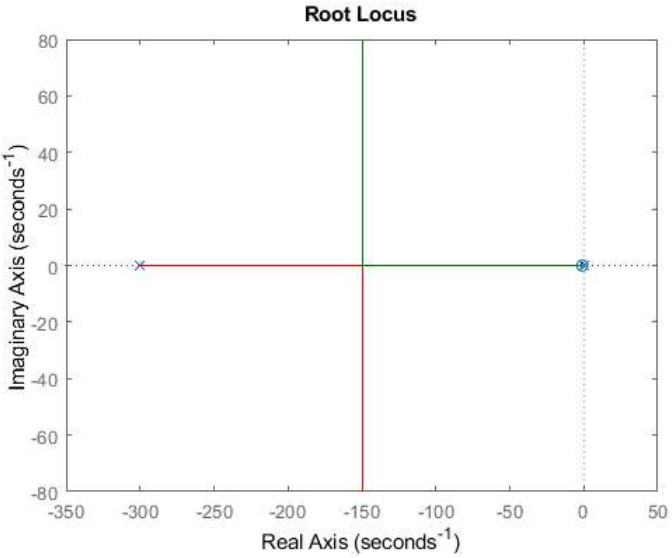
If the zero in analysis is between the roots of the transfer, there is a high chance for this real number to be found between these real values. For zeros far away from the [-300, 0] domain, it no longer converges to a real number included on the interval.

Another thing to be noted is that, for all cases, for a null gain, the pole assumes the root of the transfer function.

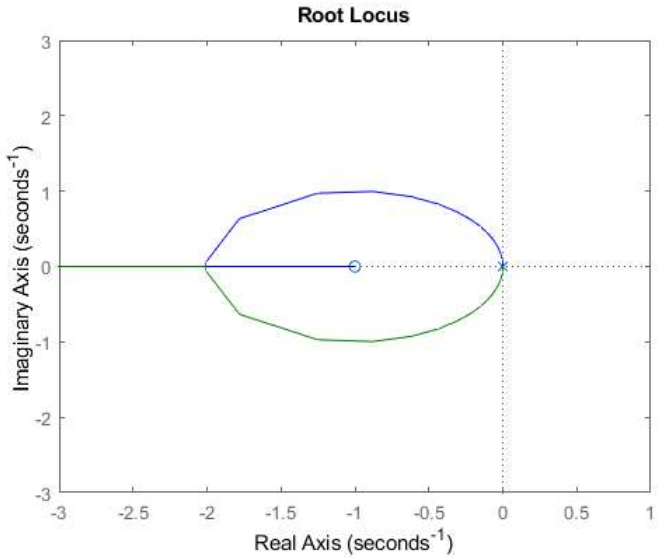
Q3.8

To obtain the gain for the double pole around $s \approx -2.01$, one must go to the root_locus for the PD controller and click on the intersection of the 'path' of the poles that are related to the root found at the origin of the plot.

```
z = 1;
rlocus(tf([1 z],[1 300 0 0]));
hold off;
```




```
z = 1;
rlocus(tf([1 z],[1 300 0 0]));
xlim([-3 1]);
ylim([-3 3]);
hold off;
```

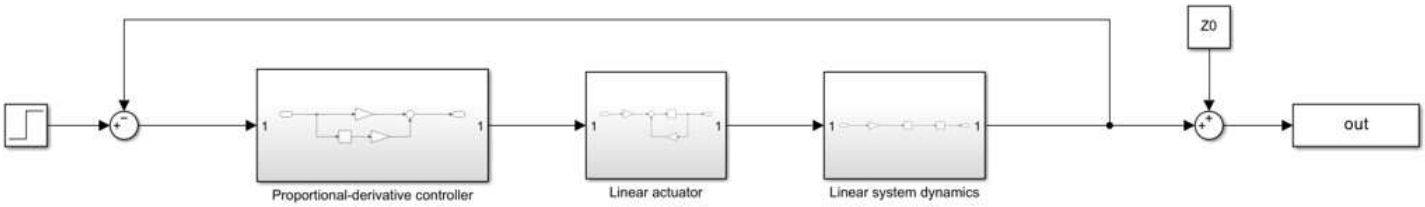



```
%We get that the gain is:

K = 1.19E3;
```

```
%System conditions

K_d = (K*M)/(600*K_t*omega0);
K_p = z * K_d;
delta_zr = 1; % We want to tell the control system how far away we want to be from the initial condition Z0 = 2
              % This value can be changed and used only for testing
              % purposes
```

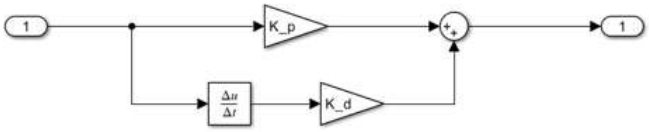
Simulink Model

```
open_system('lab_3.slx');
```



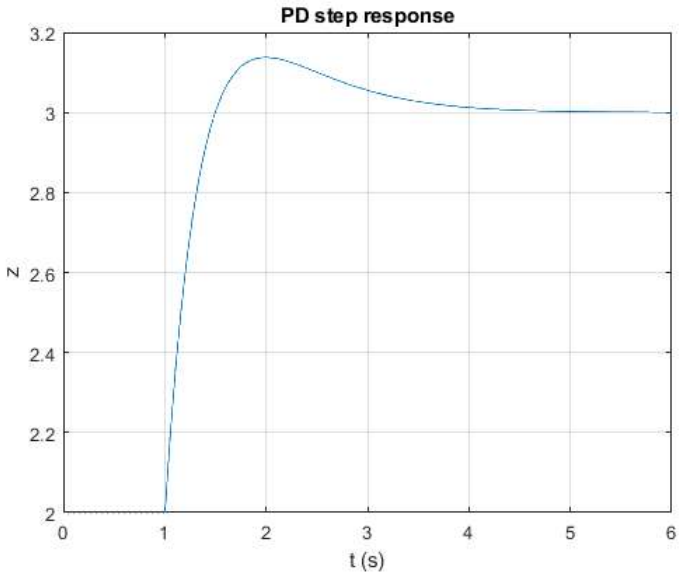
Proportional-derivative controller

```
open_system('lab_3/Proportional-derivative controller');
```




```
SimTime = 6;

% Initialization of matrices to store different simulation conditions
time_array = zeros(SimTime/StepSize + 1, 1);
pd_values_array = zeros(SimTime/StepSize + 1, 1);

% Determination of de response of the controller from the intial conditions

sim('lab_3.slx');
time_array(:, 1) = out.time;
pd_values_array(:, 1) = out.signals.values(:,1);
```

```
% Plot results
figure; hold on; title('PD step response');
plot(time_array(:, 1), pd_values_array(:, 1));
xlabel('t (s)');
ylabel('z');
grid on;
box on;
```

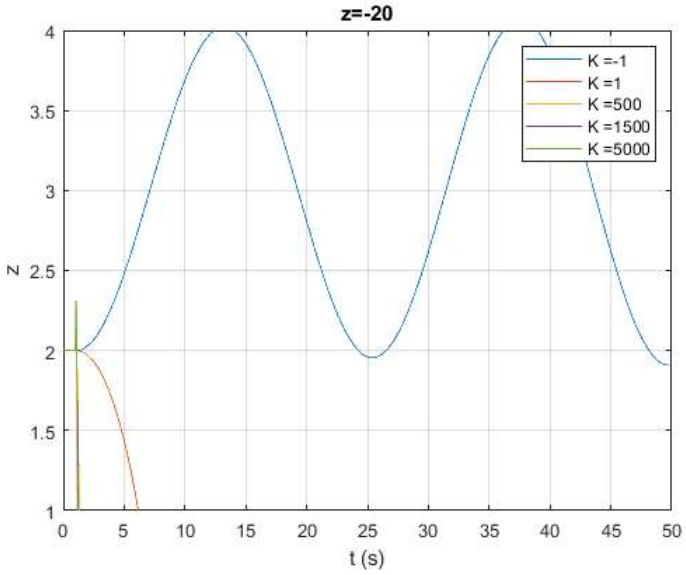


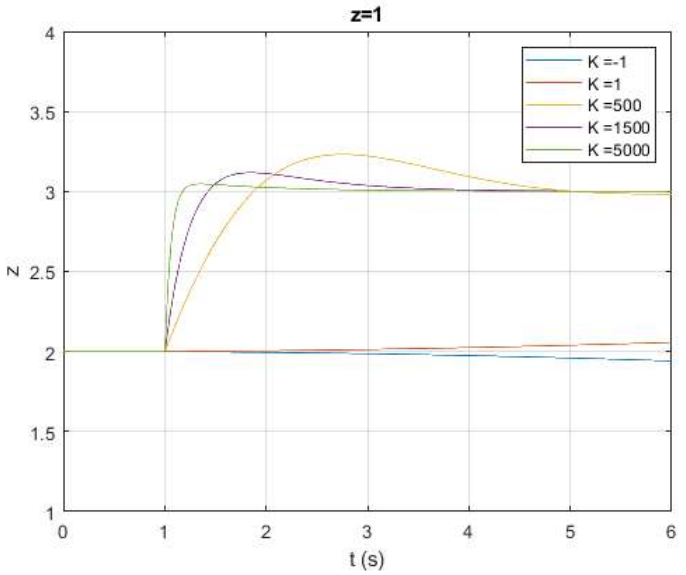
From the graphic of the step response, we can see that the system converges to a meaningful solution after 4 seconds of the application of the altitude error. Additionally, we can observe a slight overshoot before reaching the horizontal assymptote.

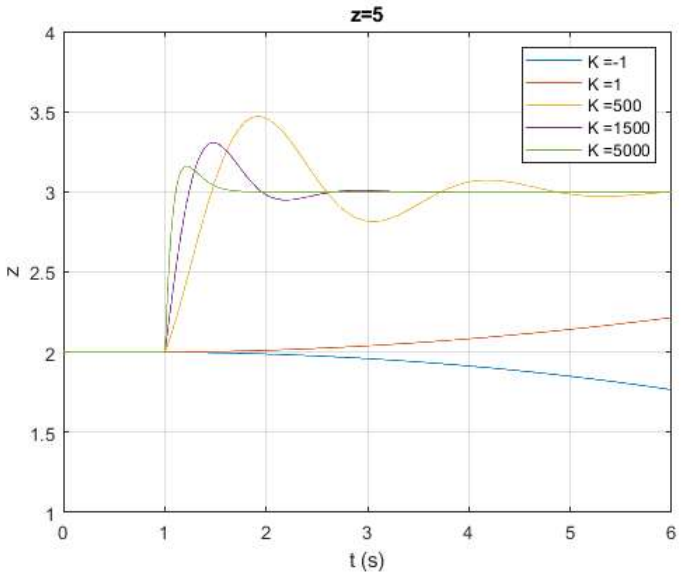
Q3.9

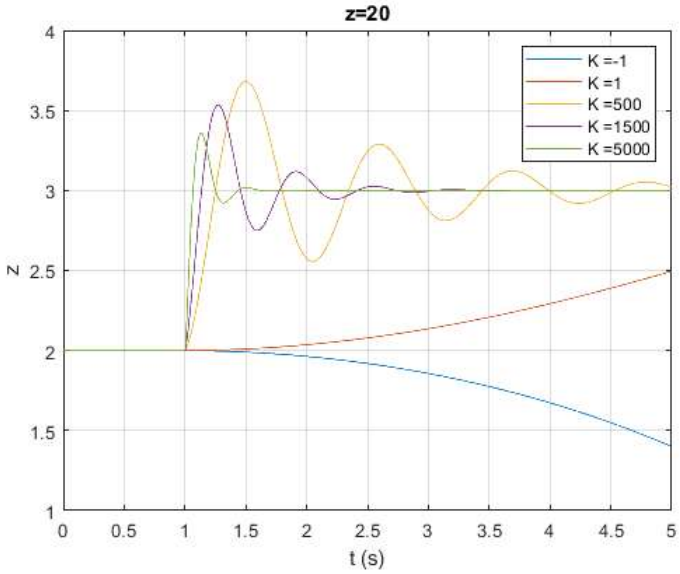
```
K_array = [-1, 1, 500, 1500, 5000];
z_array = [-20, 1, 5, 20, 100, 299, 500];
SimTime_array = [50, 6, 6, 5, 4, 3.5, 3];
K_d = (K*M)/(600*K_t*omega0);
```

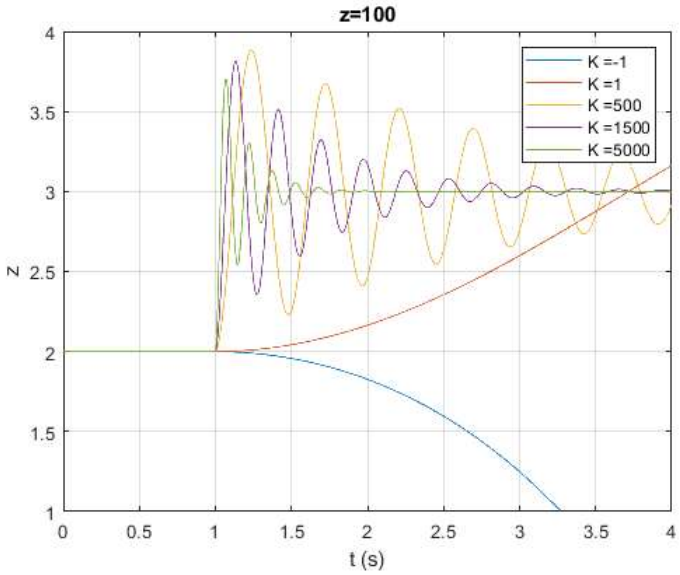
```
K_p = z * K_d;
delta_zr = 1; % We want to tell the control system how far away we want to be from the initial condition Z0 = 2

% Plot results
for i=1:length(z_array)
    figure;
    SimTime = SimTime_array(i);
    for j=1:length(K_array)
        z = z_array(i);
        K = K_array(j);
        K_d = (K*M)/(600*K_t*omega0);
        K_p = z * K_d;
        delta_zr = 1; % We want to tell the control system how far away we want to be from the initial condition Z0 = 2
        % Initialization of matrices to store different simulation conditions
        time_array = zeros(SimTime/StepSize + 1, 1);
        pd_values_array = zeros(SimTime/StepSize + 1, 1);
        % Plots
        sim('lab_3.slx');
        time_array(:, 1) = out.time;
        pd_values_array(:, 1) = out.signals.values(:,1);
        hold on; title("z=" + num2str(z_array(i)));
        ylim([1 4]);
        plot(time_array, pd_values_array(:, 1));
        xlabel('t (s)');
        ylabel('z');
        grid on;
        box on;
    end
    legend("K =" + num2str(K_array(1)), "K =" + num2str(K_array(2)), "K =" + num2str(K_array(3)), "K =" + num2str(K_array(4)), "K =" + num2str(K_array(5)));
end
```

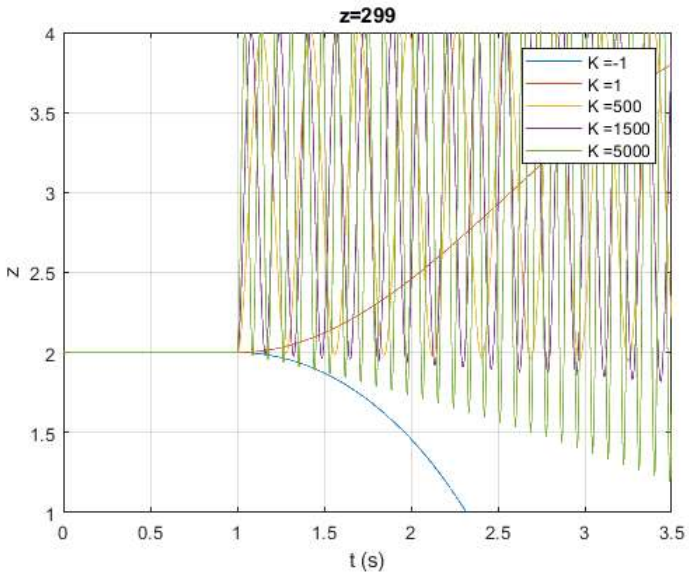

z=-20
K =-1
K =1
K =500
K =1500
K =5000
z
t (s)

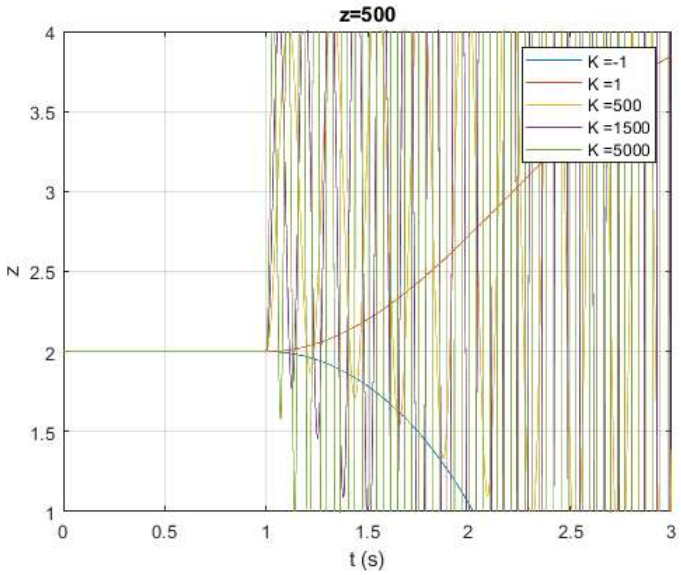

z=1
K =-1
K =1
K =500
K =1500
K =5000
z
t (s)


z=5
K =-1
K =1
K =500
K =1500
K =5000
z
t (s)


z=20
K =-1
K =1
K =500
K =1500
K =5000
z
t (s)


z=100
K =-1
K =1
K =500
K =1500
K =5000
z
t (s)





Looking at the root-locus of question 3.7, we can presume that the system will converge for zeros higher than -300 and smaller than 0. For high gains, we can see that the poles imaginary part becomes more visible and, therefore, high oscilations are expected in the step response of the system.

Now let's look at the graphs obtained in this section. In all of the situations presented above, the system does not converge to any solution for a positive zero of the transfer function of the PD.

For very small negative values of zeros of the transfer function (around -300) and positive gains, either the system oscilates exporadically or does not display any meaningful information.

For a small negative gain, the system does not converge to a solution.

For zeros greater than -300, the system converges to a meaningful solution. The higher the gain, the faster the system reaches the desired altitude. However, there is the onsequence of having a high frequency of oscilations.

Published with MATLAB® R2023a