Linear Control Design II - Group Work Problem Module 2 Solution

Description

In Figure 1 an electro-mechanical system is illustrated. The mechanical part is composed of a flexible structure buit by three masses interconnected by flexible beams. The electrical part is composed of a pair of electromagnets. The lateral movements of the lower mass is coupled to the pair of electromagnets by means of the electromagnetic forces generated in the small gap illustrated in Figure 2. It means there is no mechanical contact between the two parts. The main goal is: a) to build an electro-mechanical model based on simplifying assumptions; b) to build a mathematical model for describing the dynamics of the mechanical and electro-magnetic parts based on first principles postulated by Newton, Ampere, Ohm, Faraday; c) to build a simulik model to describe the coupled dynamics between the lateral movements of the three masses $x_1(t),\ x_2(t),\ x_3(t)$ and the current $i(t)$ induced in the coils depending on the electrical potential $u(t)$; d) investigate the vibrations of the structure when different initial conditions and inputs are present.

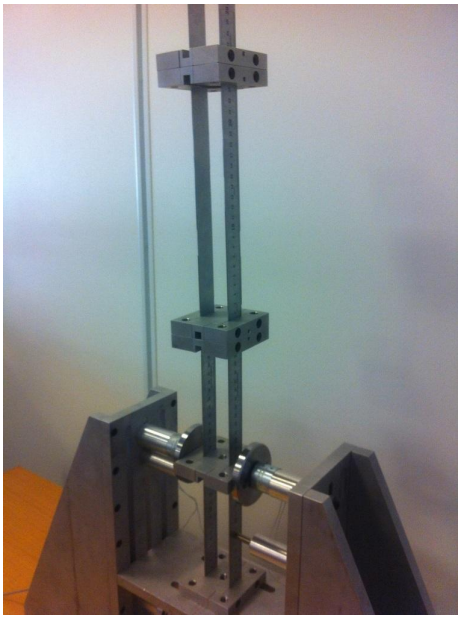
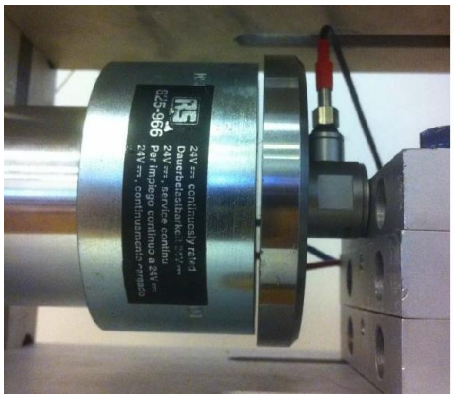
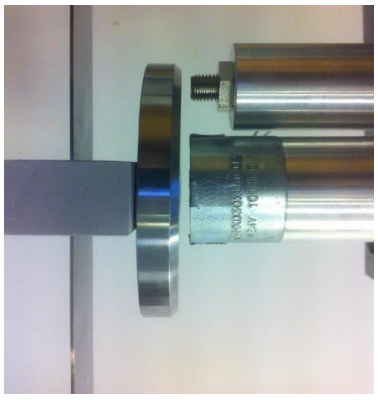


Figure 1 : Electro – mechanical system composed of three masses interconnected by flexible beams (mechanicalparts) and a pair of electromagnets (electricalparts).



(a)



(b)

Figure 2 : The electroparts : (a) one single electromagnet and the gap where the electromagnetic force will be generated by the input voltage $u(t)$; (b) electromagnetic actuator and eddy − current displacement sensor using to control and to measure the lateral movements of the lower mass .

For the simulations the following parameters are given.

```
% Parameters
m1 = 0.712;         % [kg] mass of platform 1
m2 = 0.428;         % [kg] mass of platform 1
m3 = 0.428;         % [kg] mass of platform 1

R = 148.2; % [Ohm]
L = 800e-3; % [H]

b = 0.025;          % [m] beam width
h = 1.0e-3;         % [m] beam thickness
E = 2.0e11;         % [N/m2] elasticity modulus of steel
Iz = (b*h^3)/12;    % [m4] area moment of inertia

L1= 0.145;           % [m] length of beam 1
L2= 0.134;           % [m] length of beam 2
L3= 0.229;           % [m] length of beam 3

% Force constants obtained from the linearisation

k1 = 2*12*E*Iz/L1^3;   % [N/m] stiffness of beam 1
k2 = 2*12*E*Iz/L2^3;   % [N/m] stiffness of beam 2
k3 = 2*12*E*Iz/L3^3;   % [N/m] stiffness of beam 3

Ki5 = 22.4;
Ks5 = -313.6;

Ki3 = 37.33;
Ks3 = -871.11;

Ki2 = 56;
Ks2 = -1.96e3;

Ki1 = 112;
Ks1 = -7.84e3;
```

Where $Ki\#$ indicates the magnitude of the linearized force dependent on current change when the air gap s_0=# mm.

Problem 1

Build an electro-mechanical model based on simplifying assumptions.

Solution:

The beams of the structure can be simplified as simple massless linear springs. When a spring is compressed the reaction forces will point away from the center of the spring while the reaction forces will point towards the center when the spring is extended.

The lateral dynamics of the three masses is described based on Newtons 2nd law. There are three types of forces to be considered: a) the linear spring forces which are proportional to the relative displacement between the masses described by the stiffness coefficients k_1, k_2, and k_3; b) the electromagnetic force $f(t)$ acting on the lowest mass which is depending on the gap variation and current described by the coefficients ks and ki ; and finally c) the inertia forces whcih are depending on the absolute linear acceleration of the masses described by their masses. The mechanical and electrical models are shown in Figure 3. As the actuator is an electromagnet the dynamics of the actuators can be described from Faraday's law.

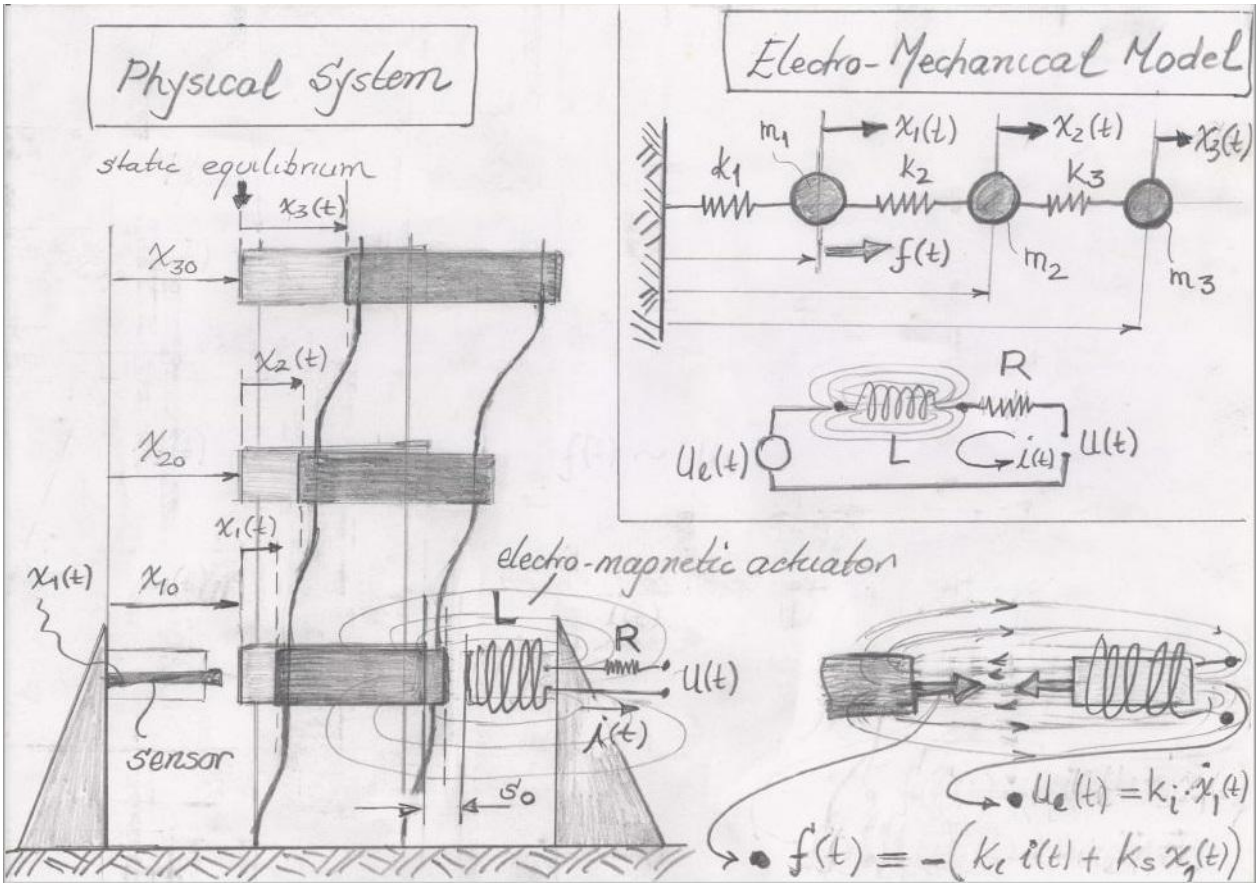


Figure 3 : Mechanical and electrical models

Problem 2

Build the mathematical model for describing the dynamics of the mechanical and electro-magnetic parts based on first principles postulated by Newton, Ampere, Ohm, Faraday, based on the electro-mechanical model and simplifying assumptions done in problem 1.Use as main variables the three masses $x_1(t)$, $x_2(t)$, $x_3(t)$ and the current $i(t)$ induced in the coils depending on the electrical potential $u(t)$.

Solution:

The differential equations in Figure 4 describe the lateral dynamics of the flexible structure, i.e. the accelerations of the three masses $\ddot{x}_1(t)$, $\ddot{x}_2(t)$ and $\ddot{x}_3(t)$, and the dynamic behavior of the current $\dot{i}(t)$ in the electrical circuit composed of resistance R, inductance L, the input voltage $u(t)$ and the induced current due to the linear velocity of the lowest mass (which will also induce the electromagnetic force). The differential equations are derived from the electro-mechanical model.

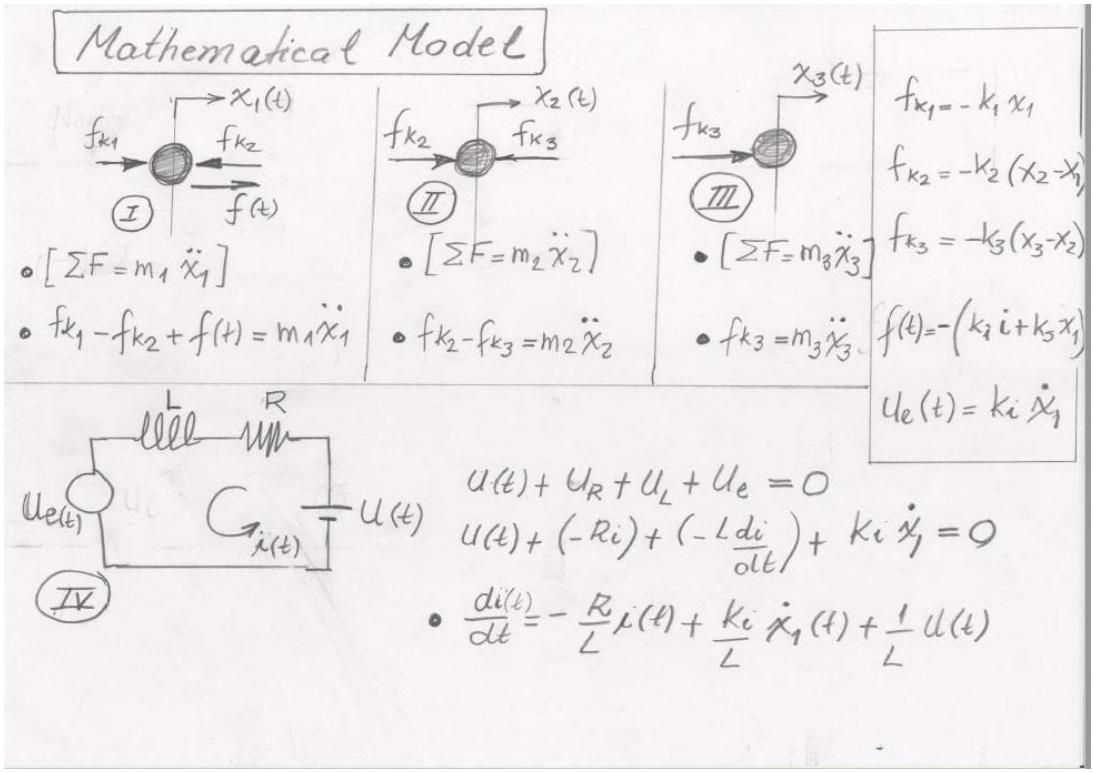


Figure 4 : The global mathematical model

Problem 3

Build a simulik model to describe the coupled dynamics between the lateral movements of the three masses $x_1(t)$, $x_2(t)$, $x_3(t)$ and the current $i(t)$ induced in the coils depending on the electrical potential $u(t)$.

Solution:

The simulink diagram deducted from the mathematical model and used for the simulations is presented in Figure 5.

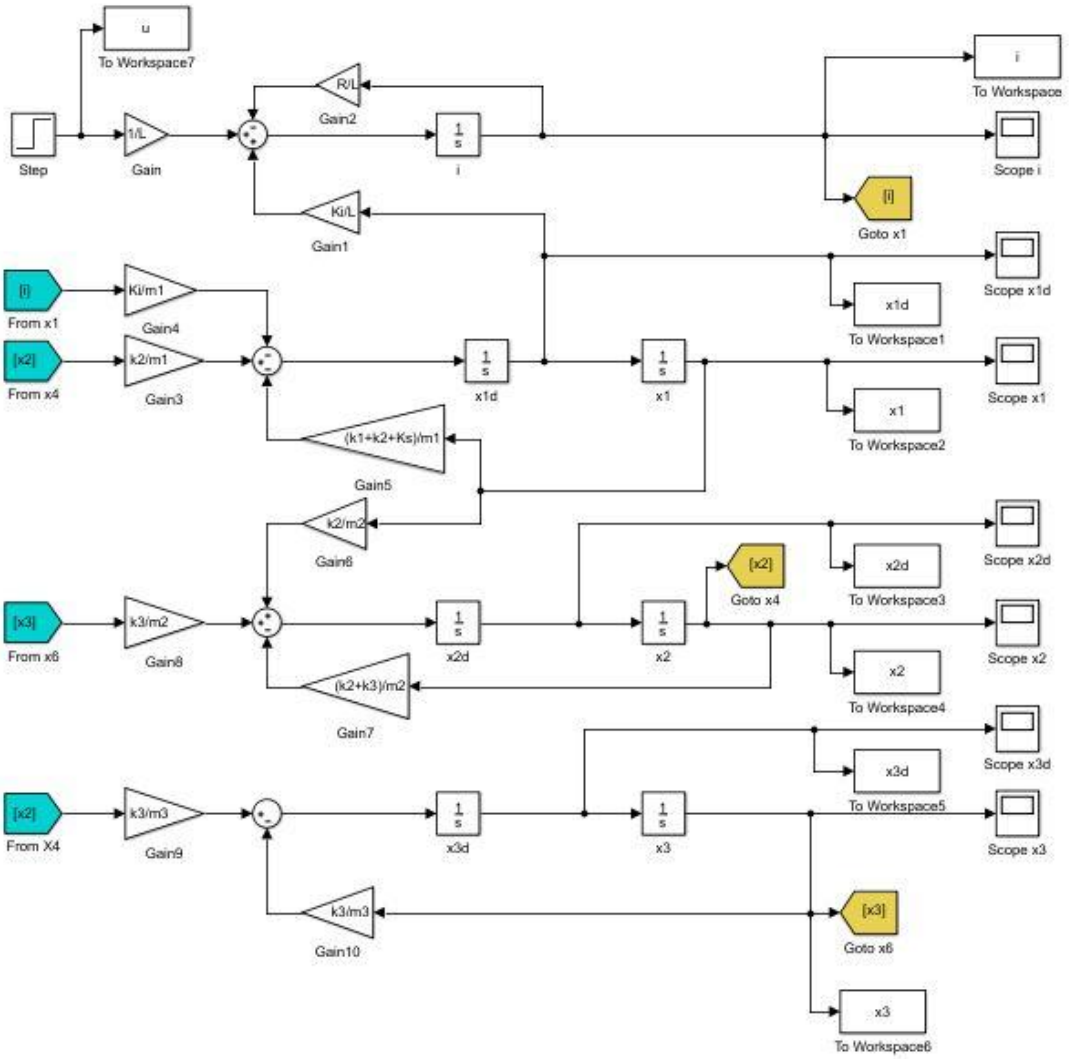


Figure 5 : The simulink diagram used to simulate the coupled dynamics between the lateral movements of the three masses $x_1(t), x_2(t), x_3(t)$ and the current $i(t)$ induced in the coils depending on the electrical potential $u(t)$

Problem 4

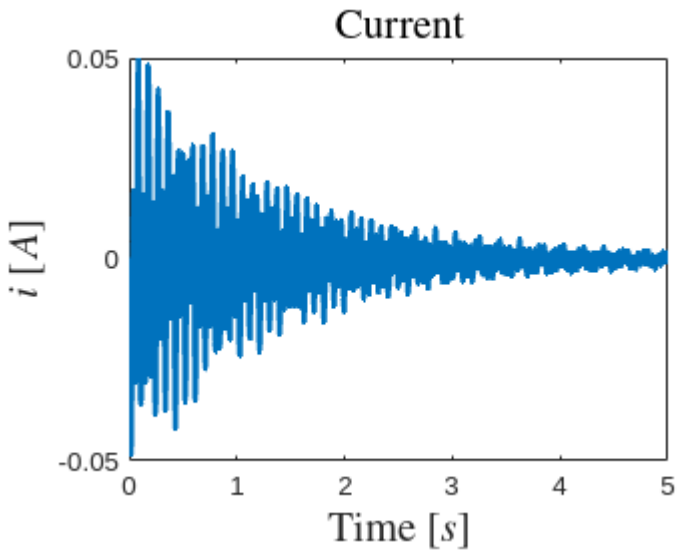
The goal of the problem is to investigate how the structure behaves when s_0 (air gap) is varied from 5 mm to 1 mm. The initial conditions of lateral displacements $x_2(0)$, $x_3(0)$ and velocities $\dot{x}_1(0)$, $\dot{x}_2(0)$, $\dot{x}_3(0)$ are zero and kept for all simulation cases. Only the lateral displacement of the lowest mass $x_1(0)$ is offset by 1 mm from center. Simulate three different values of s_0=5 mm, s_0=3 mm, s_0=1 mm. Explain the simulation results.

Solution:

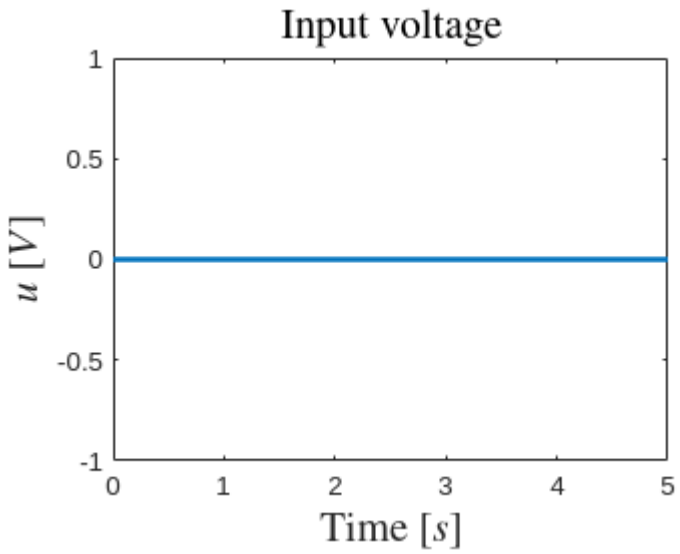
4a) Behavior of system when s_0=5 mm

```
Ki = Ki5;
Ks = Ks5;
```

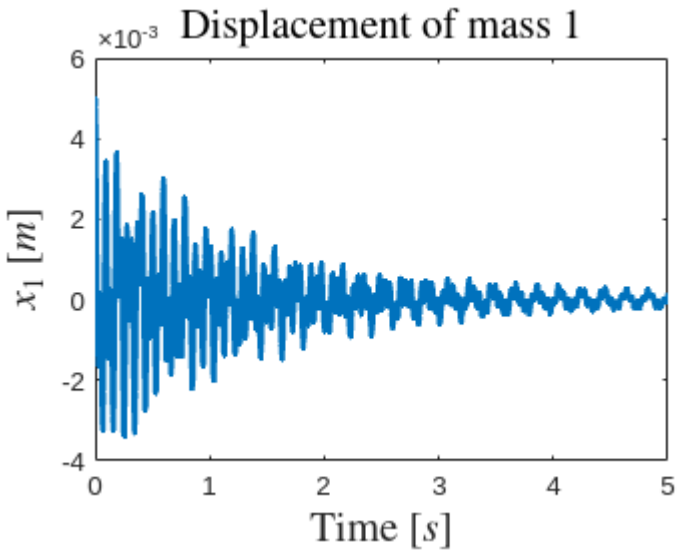
```
% initial conditions
x_0 = [0 0.005     0 0 0 0 0]';

% input
u_0 = 0;

%% Simulations
sim('electroMagnetModel');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```



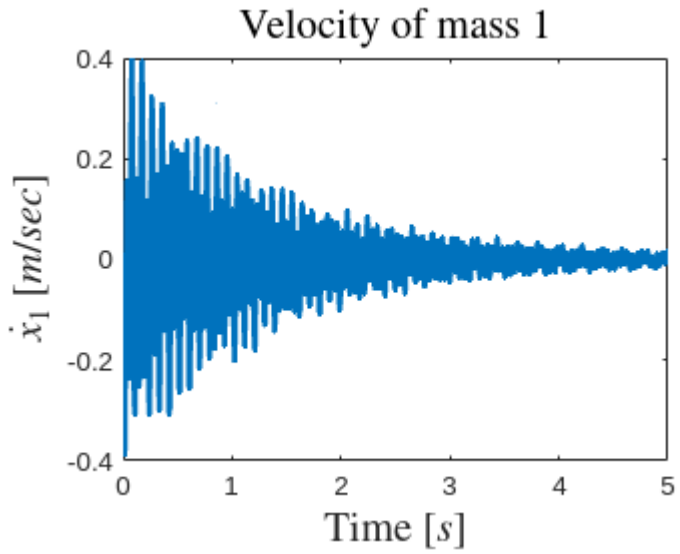

```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



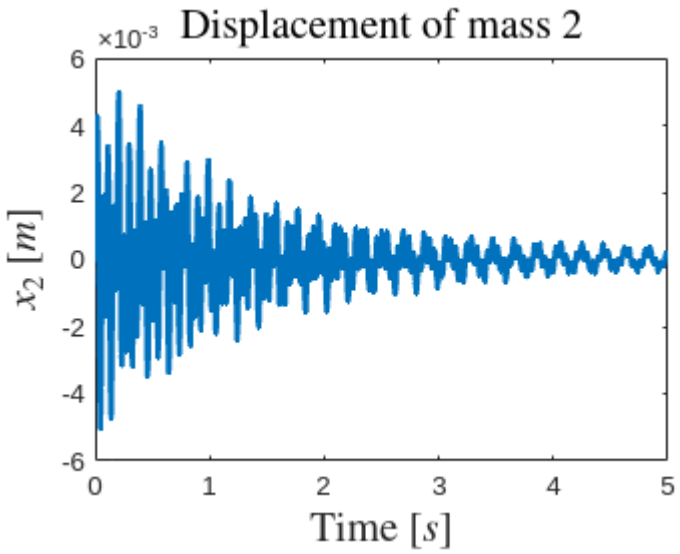

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



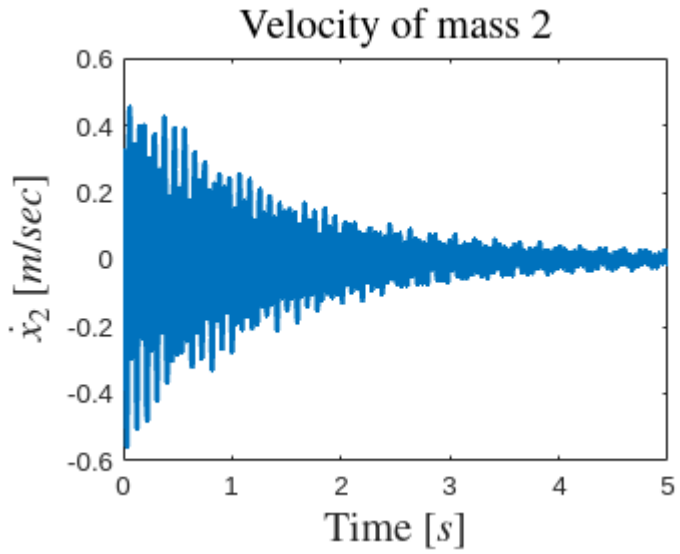

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



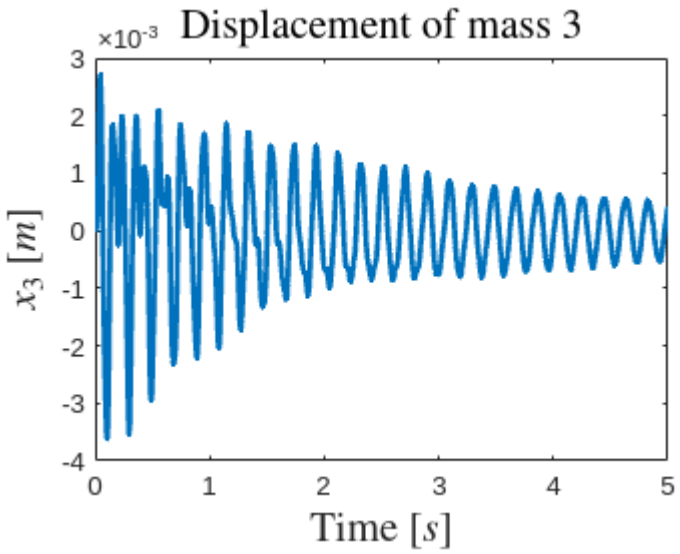

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



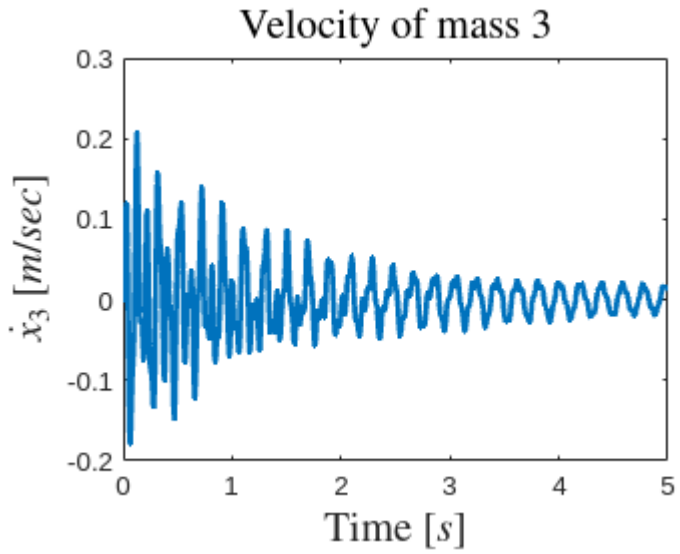

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```



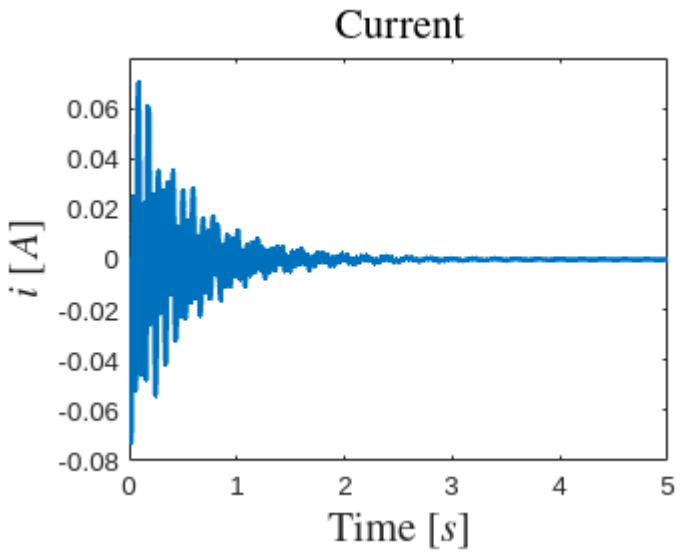

```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```



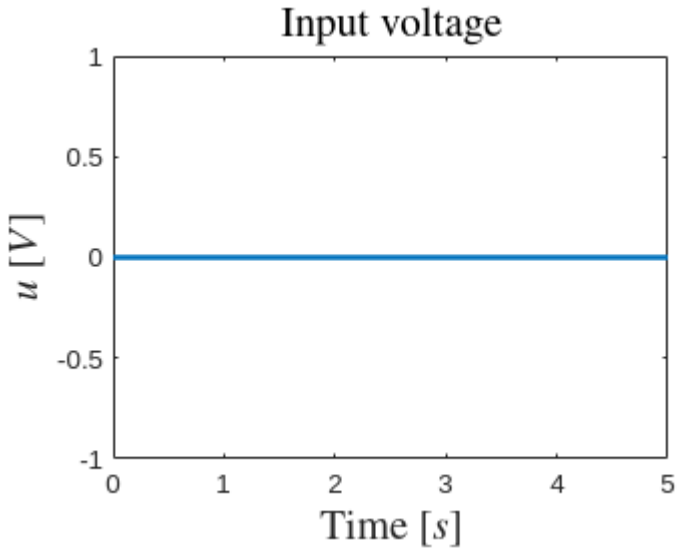
As we can see in the plots, when there is no external input (voltage or forces) acting on the system, the responses decay over time to their equilibrium positions, i.e. zero displacement, zero velocity and zero current. This is expected, since the system is very similar to the mass-spring-damper system. After approximately 10 seconds all the transient behaviors of the states have died out and the state settle to its equilibrium point $\mathbf{x}(t) = [0\ 0\ 0\ 0\ 0\ 0\ 0]^T$.

4b) Behavior of system when s_0=3 mm

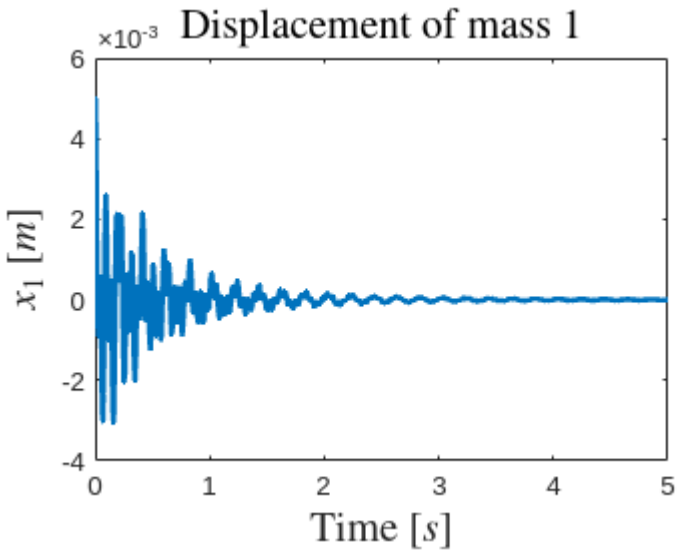
```
Ki = Ki3;
Ks = Ks3;

%% Simulations
sim('electroMagnetModel');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```



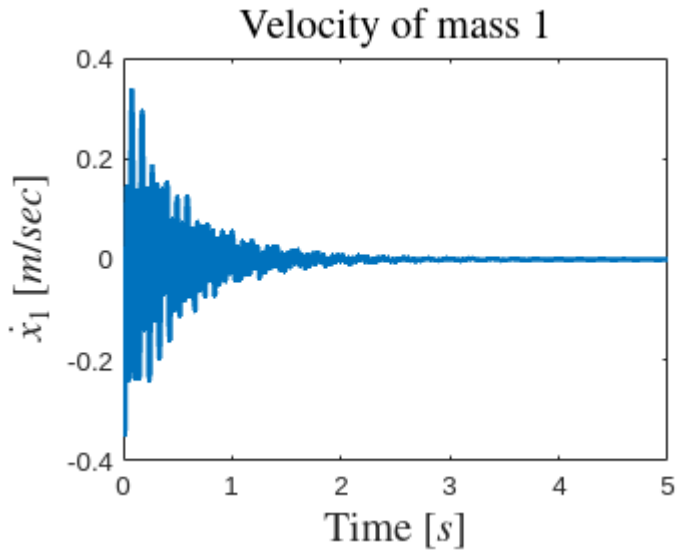

```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



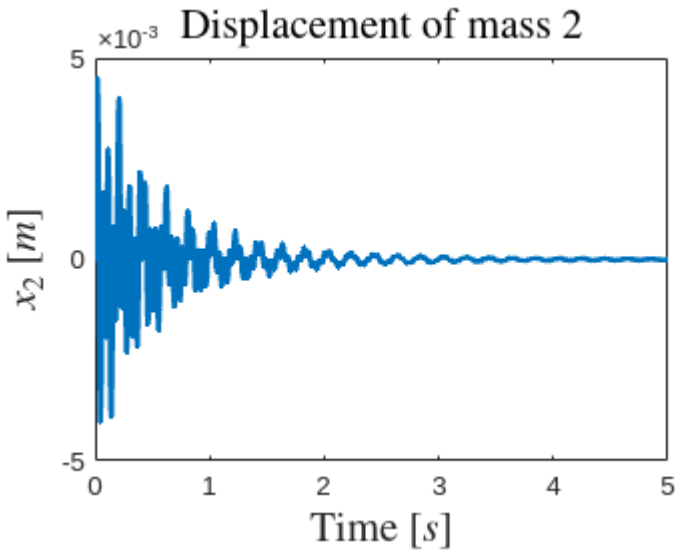

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



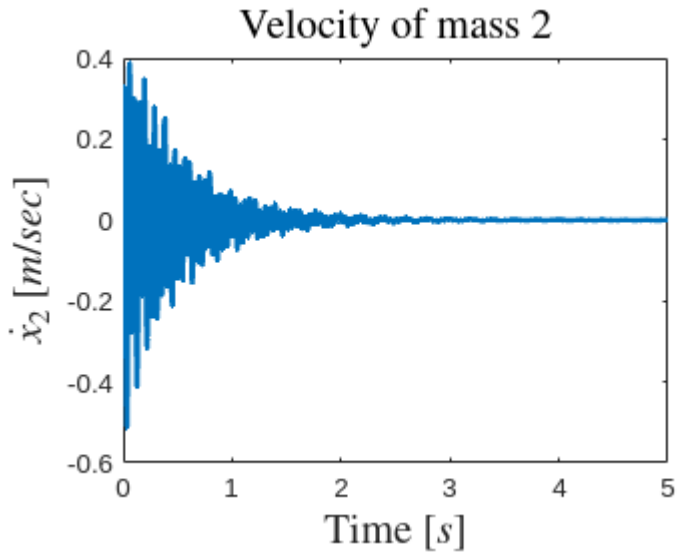

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



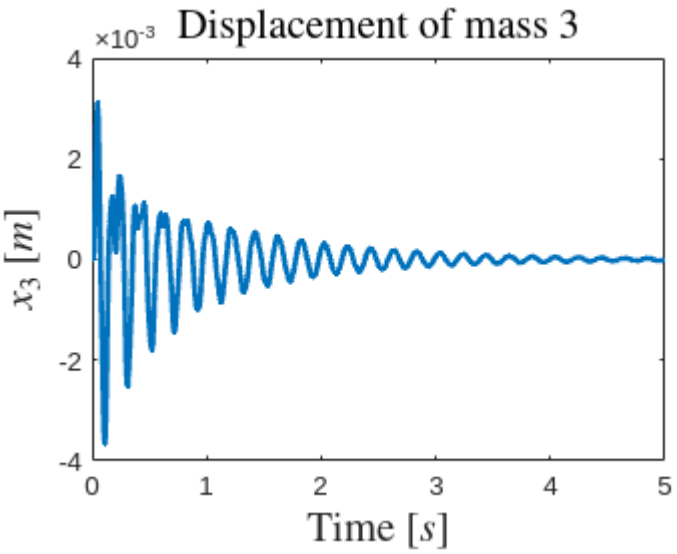

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



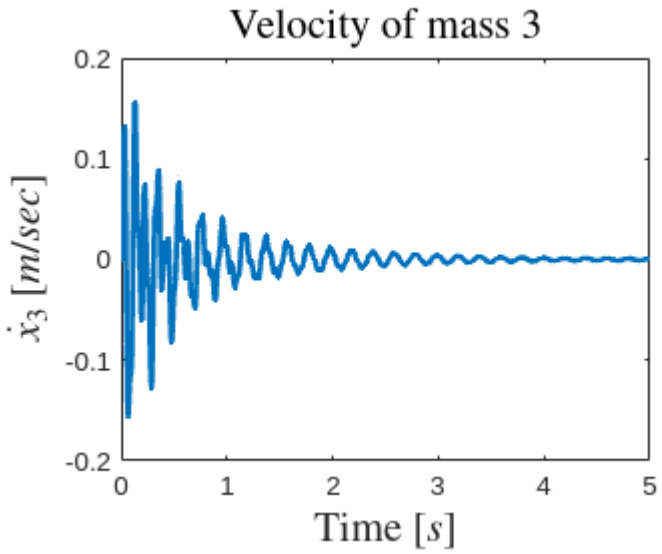

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```

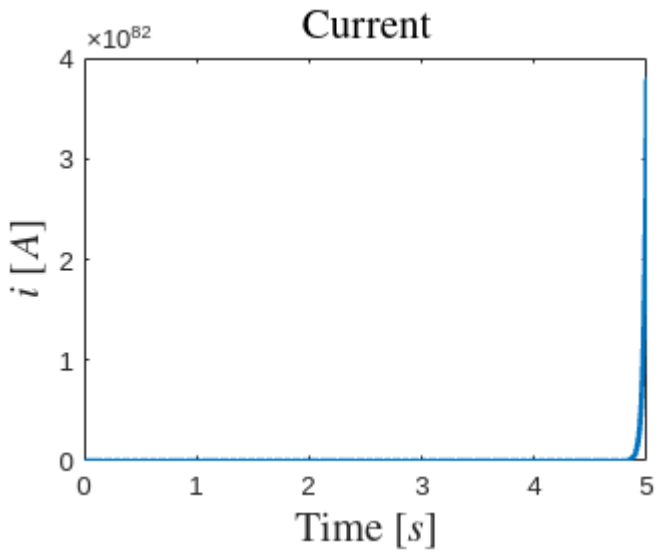


The behavior of the system is similar to the previous case with the difference that the state variables now settle to their equilibrium positions much faster, almost in 3 seconds. This can be explained by the fact that smaller air gaps between the lowest mass and the magnets generate larger magnetic forces, which in turn drive the system dynamics faster. This will be better understood next class when the nonlinar expression for the electro-magnetic forces will be presented depending on the gap and current.

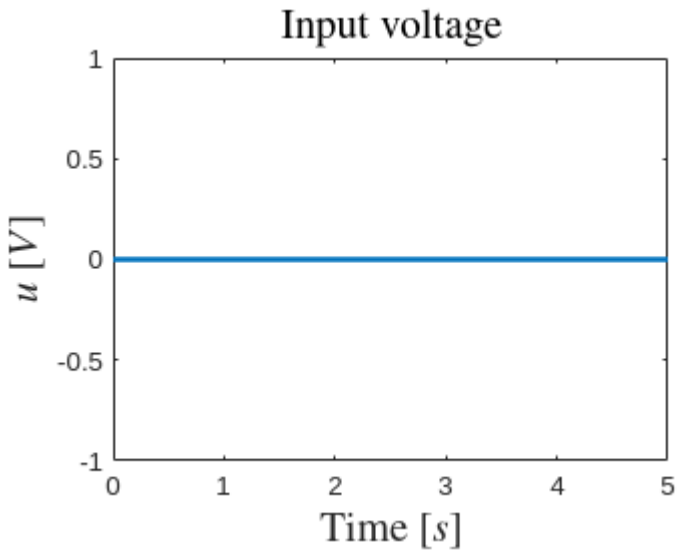
4c) Behavior of system when s_0=1 mm

```
Ki = Ki1;
Ks = Ks1;
```

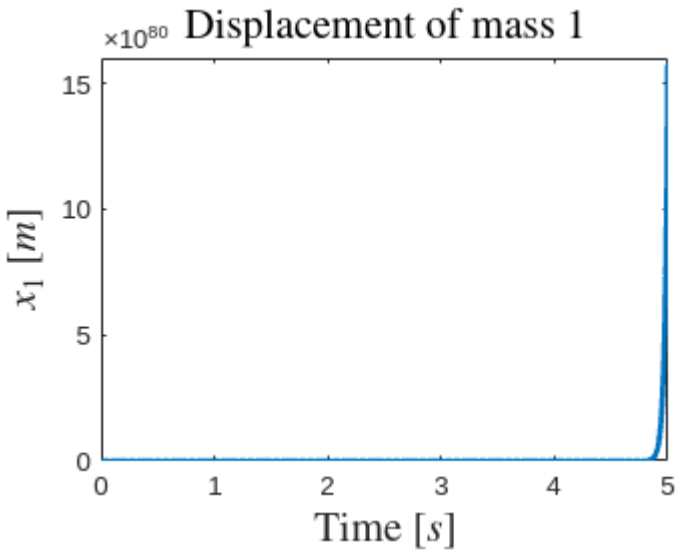
```
%% Simulations
sim('electroMagnetModel');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```



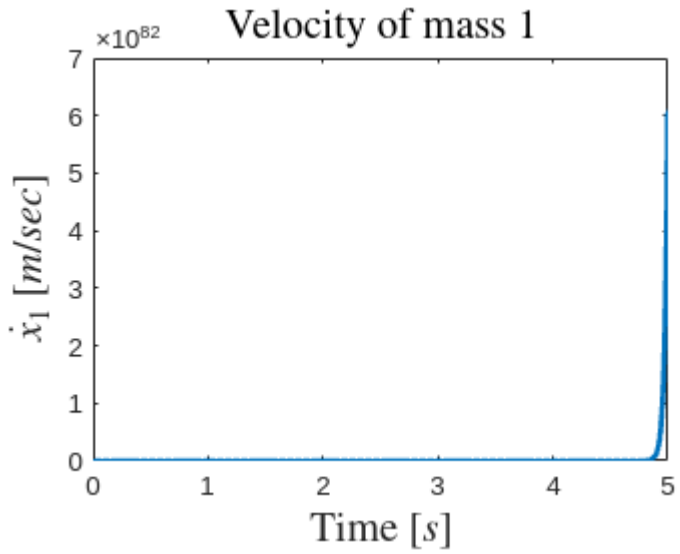

```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



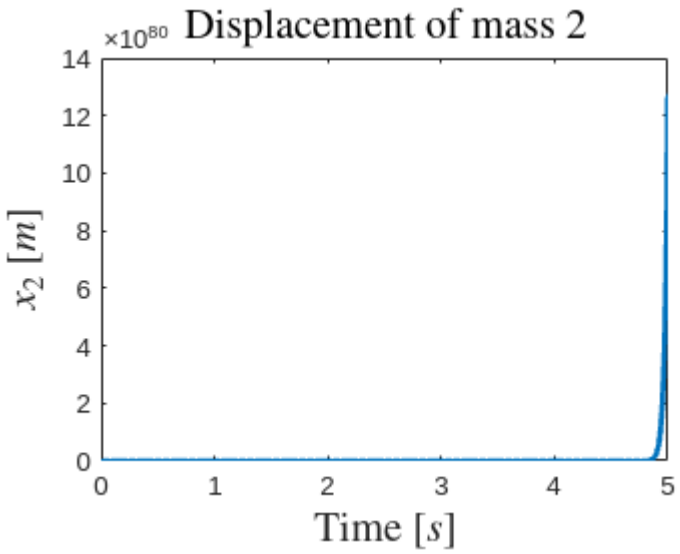

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



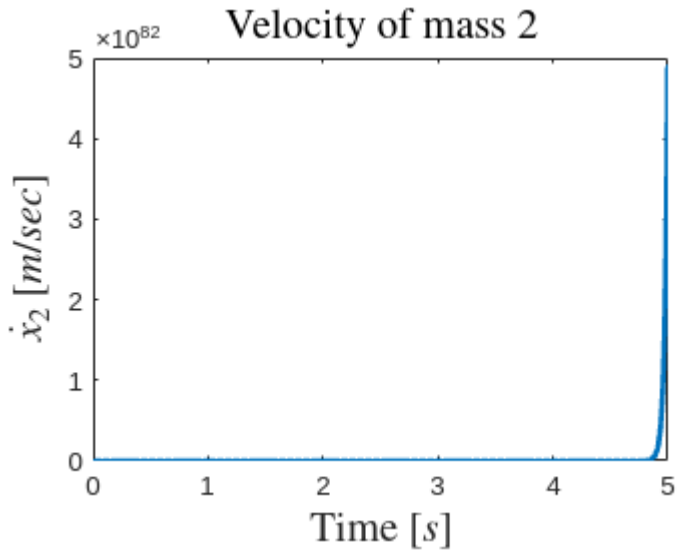

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



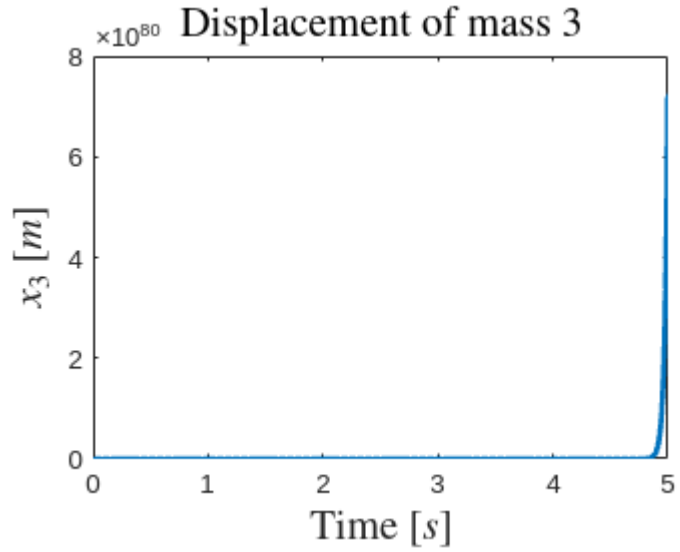

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



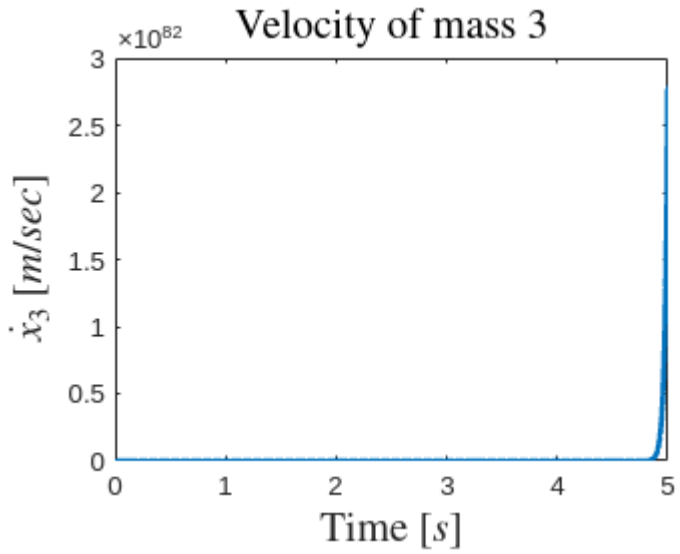

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```



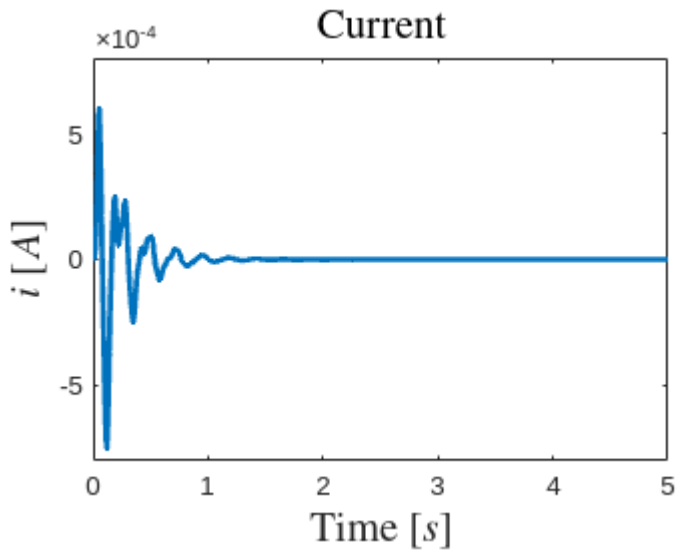
Last, the system is simulated with s_0=1 mm. As it can be seen in the plots the system becomes unstable, i.e. the state variables do not settle to any equilibrium point but they keep growing unbounded. This can be explained by the fact that in very small distances the magnetic force is too large to allow the system to settle anywhere. Of course if the plots were real sensor data, we should expect to see the states reaching a saturation level since the system is physically constraint.

Problem 5

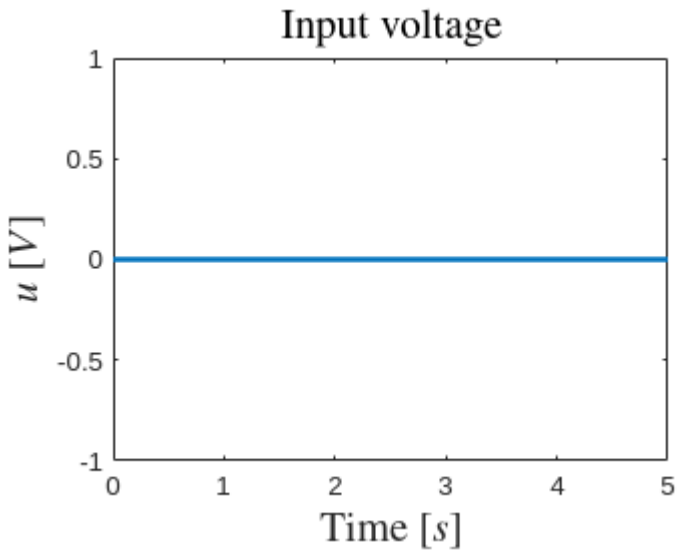
What is the maximum lateral displacement experienced by the three masses when an initial condition of velocity of 5 mm/s acts on mass 3, assuming s_0=2 mm?

Solution:

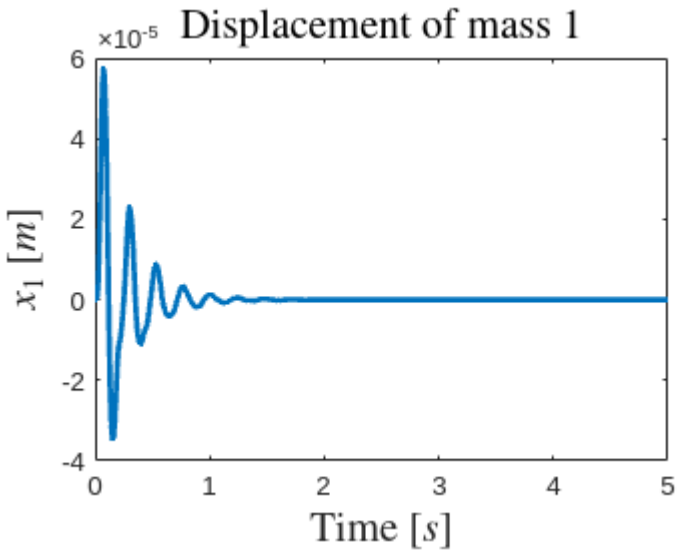
```
% initial conditions
x_0 = [0 0 0 0 0 0 0.005]';

Ki = Ki2;
Ks = Ks2;

%% Simulations
sim('electroMagnetModel');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```



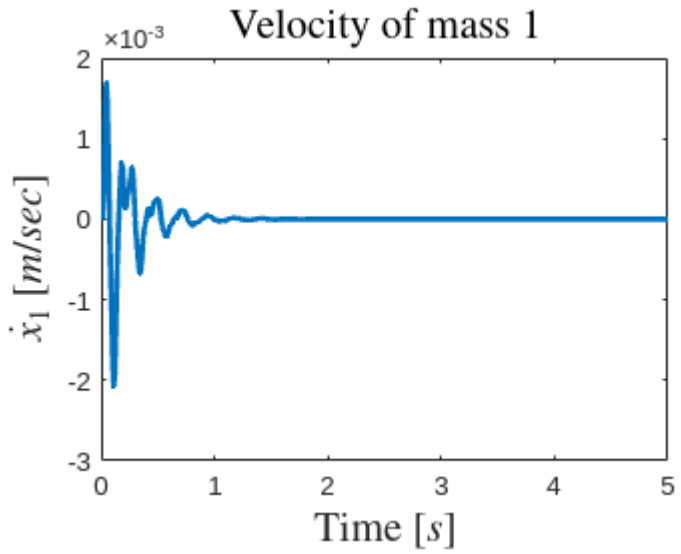

```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



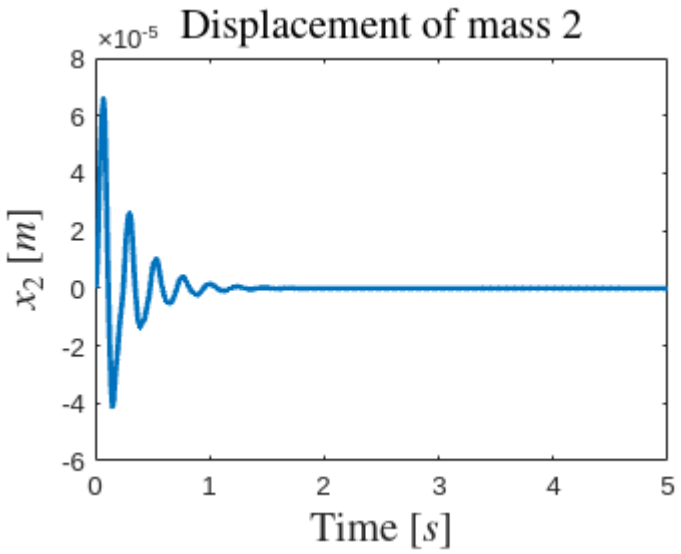

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



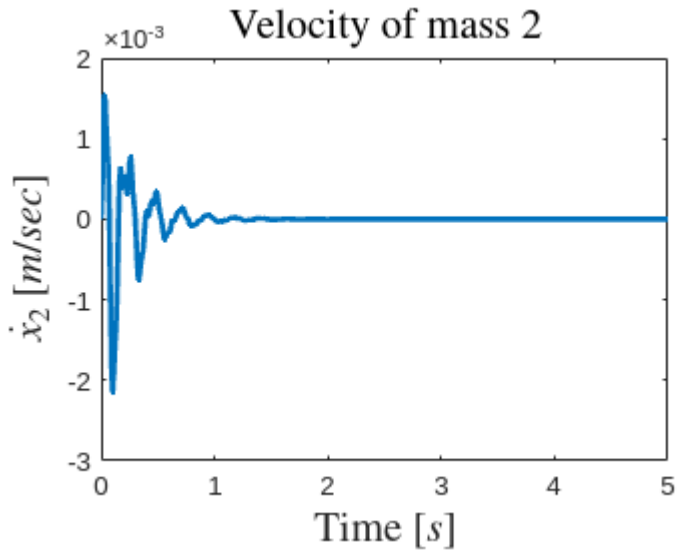

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



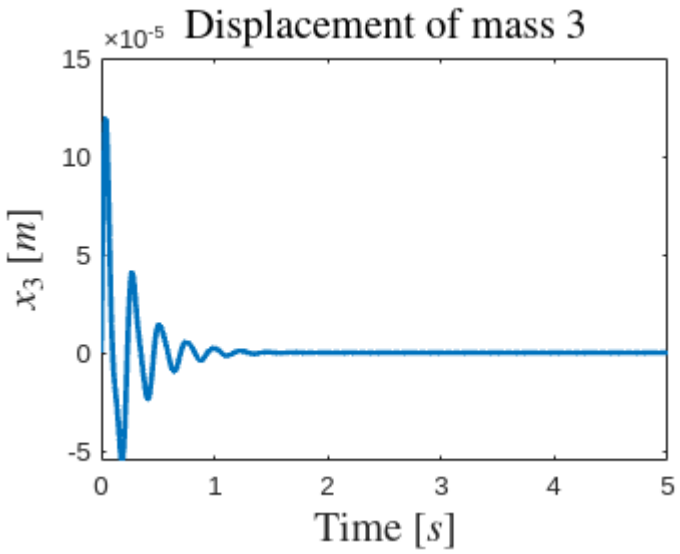

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



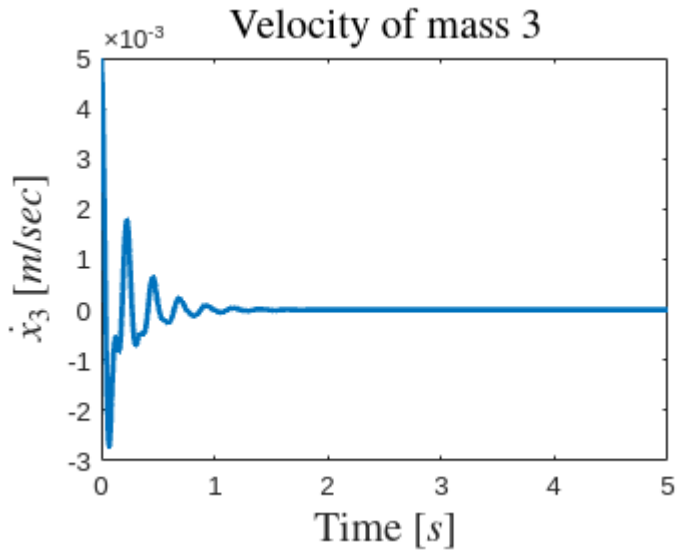

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```



From the time response of the state variables corresponding to the displacement of the masses from the simulation (the three plots on the left hand side) we can measure the maximum displacement for each mass as

$$x_{1,max} = 5.74 \cdot 10^{-5} m$$

$$x_{2,max} = 6.57 \cdot 10^{-5} m$$

$$x_{3,max} = 1.18 \cdot 10^{-5} m$$

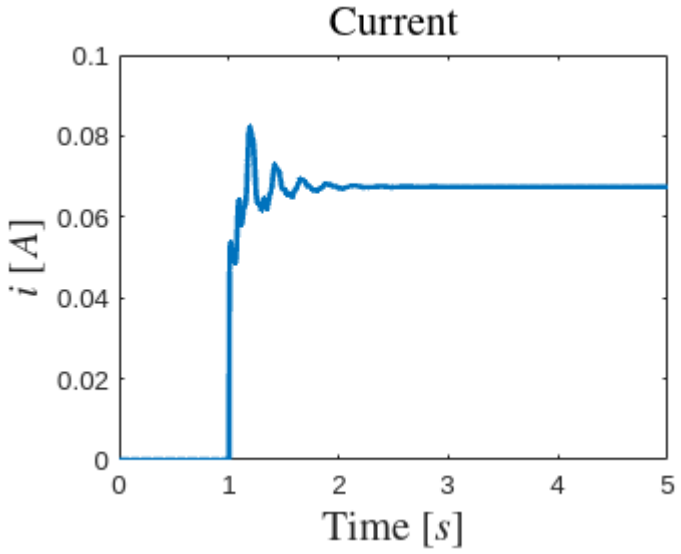
The displacement of the third (upper) mass is larger, almost double, in comparison to the other displacements. This is expected since the third mass on the top of the structure is globally more flexible, accumulating the deflection of the two lower masses.

Problem 6

A step function $u(t)$=10 V is send to the electromagnets, assuming all initial conditions of lateral displacement, velocity and current nill. How will the three masses vibrate and how will the current change, i.e. amplitudes and frequencies.

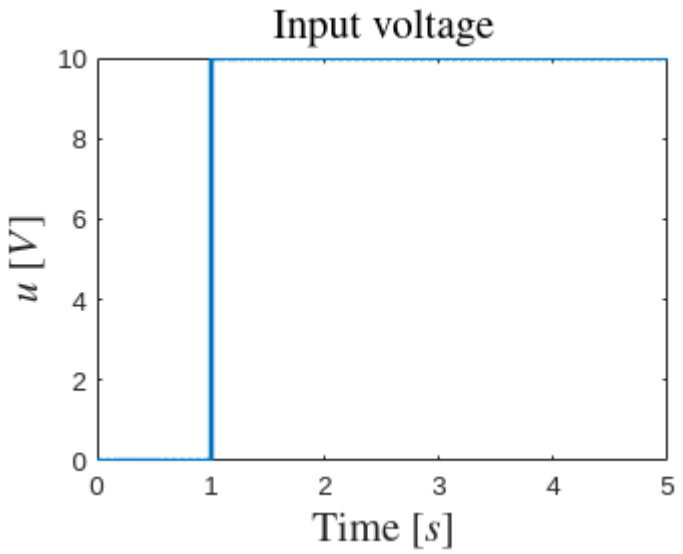
Solution:

```
% initial conditions
x_0 = [0 0 0 0 0 0 0]';

% input
u_0 = 10;

%% Simulations
sim('electroMagnetModel');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```

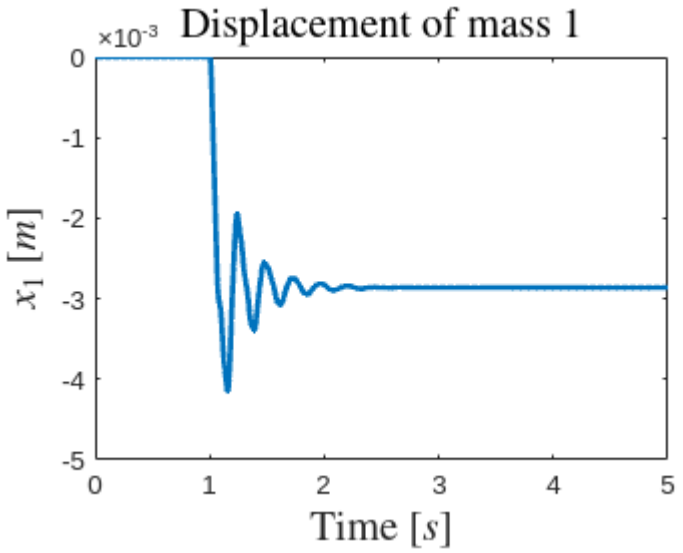



```
figure
plot(time,u.signals.values, 'LineWidth', 2);
```

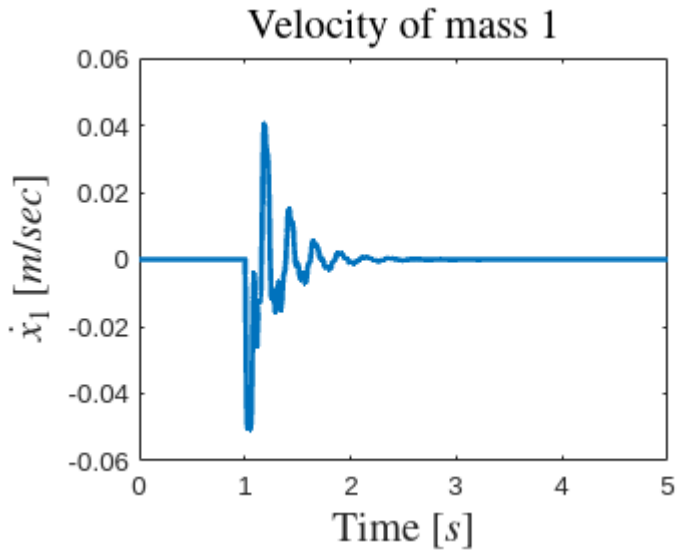
```
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



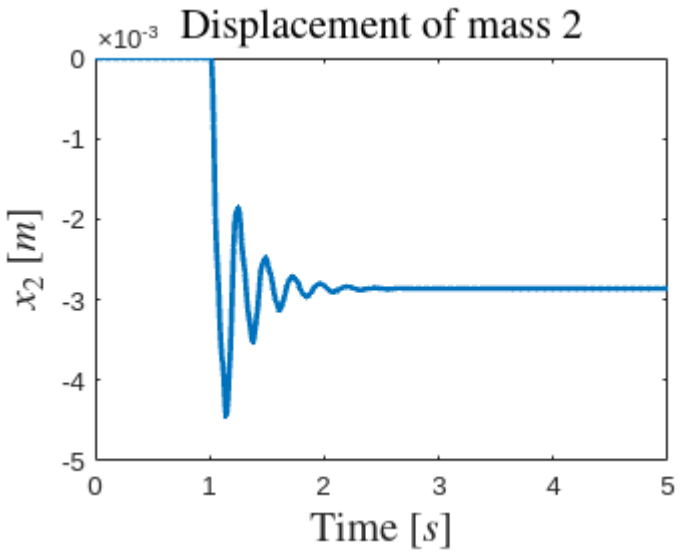

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



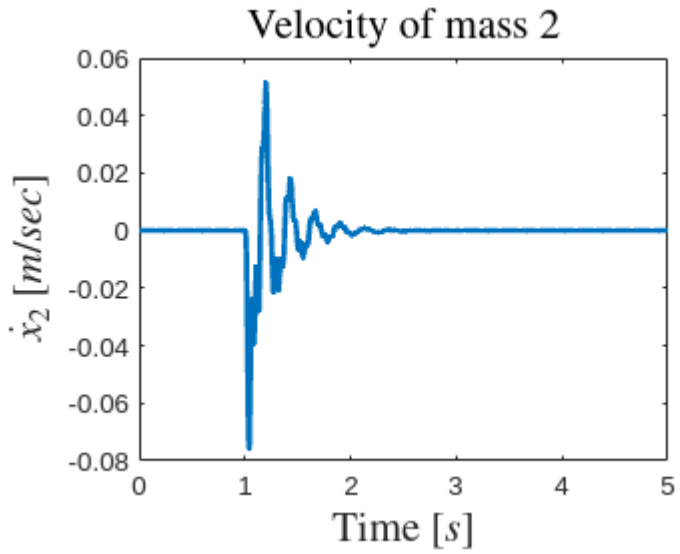

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



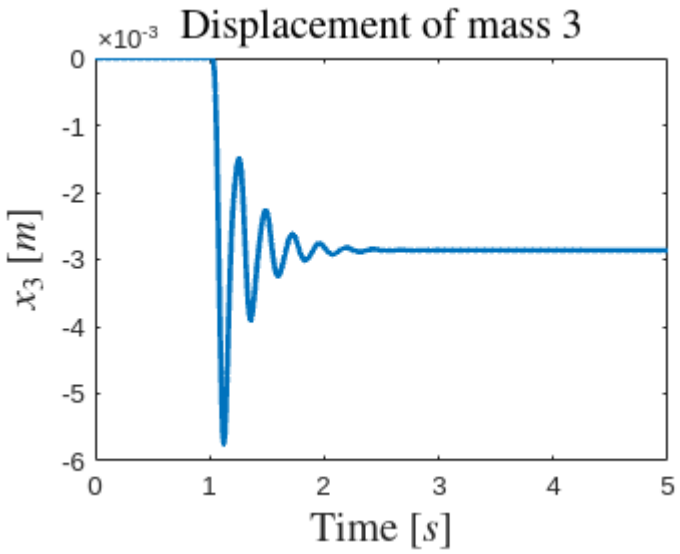

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



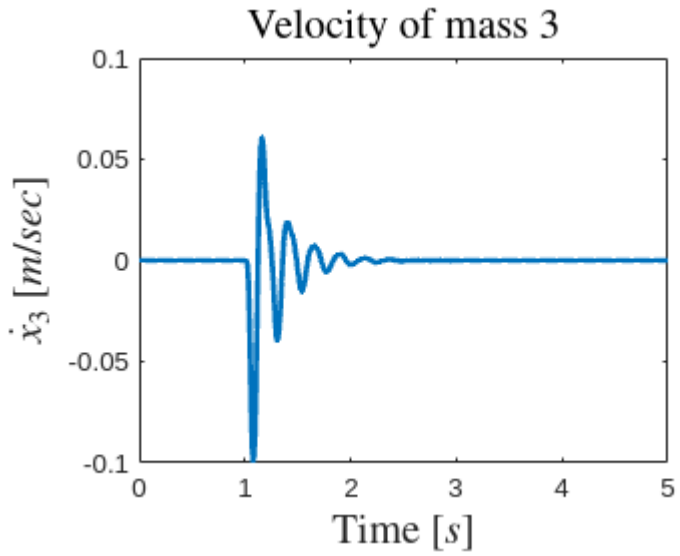

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```



As it is shown by the plots, after the transient behavior of the states has died out, the latter settle at a constant steady-state value. The velocities, as expected settle to zero, while the positions assume a nonzero value since there is a constant voltage that keeps exciting the system. The transient behavior can be described as a fading oscillatory response, which is expected, given that there is damping in the system. From the plots, by measuring the period (approximately) of the oscillations and the amplitude of the responses we can approximate the magnitude and the frequency of the signals. In the next lectures, when the modal analysis will be introduced, the relation between the oscillatory behavior of the responses and the eigenvalues of the system matrix $\mathbf{A}$ will provide an accurate method of calculating the frequencies and amplitudes of the state responses.

Linear Control Design II - Group Work Problem Module 3 Solution

Description

This exercise is based on the work carried out in the Group Work Problem Module 2. For the electro-mechanical system illustrated in Figure 1 and described in Group Work Problem Module 2 the goals for this assignment is: a) linearize the electromagnetic force resulted from the change in current and change in displacement. I.e. Ki and Ks; b) derive the SISO state space model ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ & $\mathbf{D}$) for the electro-mechanical system using the state parameters

$$\mathbf{x} = [q_1, q_2, q_3, q_4, q_5, q_6, q_7]^T = [i, x_1, \dot{x}_1, x_2, \dot{x}_2, x_3, \dot{x}_3]^T$$

where

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

c) investigate the vibrations of the structure when different initial conditions and inputs are present and compare with the results from Group Work Module 2;

d) compare the obtained results from Group Work Module 2 with the results from this Group Work Module.

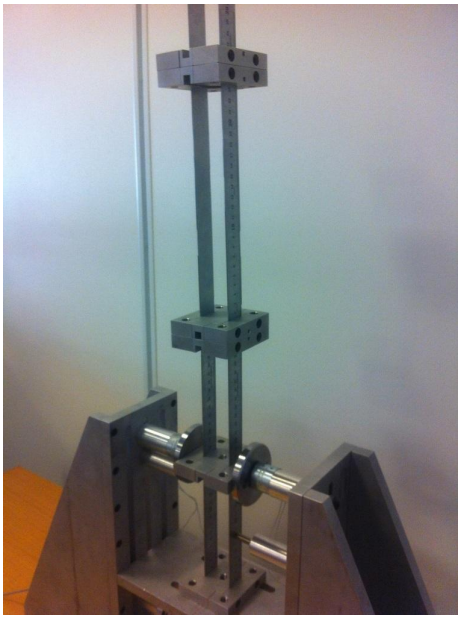


Figure 1 : Electro − mechanical system composed of three masses interconnected by flexible beams (mechanicalparts) and a pair of electromagnets (electricalparts).

Problem 1

Linearize the electromagnetic force resulted from the change in current and change in displacement.

Define Ki and Ks

Solution:

The electromagnetic force derived by $i(t)$ yields Ki.

The electromagnetic force derived by $x(t)$ yields Ks.

Ki and Ks from slides given below.

```
syms mu N A ib s0
Ki = mu*N^2*A*ib / s0^2
```

```
Ki =
```

$$\frac{A\,N^2\,\mathrm{ib}\,\mu}{s_0{}^2}$$

```
Ks = -mu*N^2*A*(ib^2)/s0^3
```

```
Ks =
```

$$-\frac{A\,N^2\,\mathrm{ib}^2\,\mu}{s_0{}^3}$$

Problem 2

Derive the SISO state space model ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ & $\mathbf{D}$) for the electro-mechanical system using the state parameters

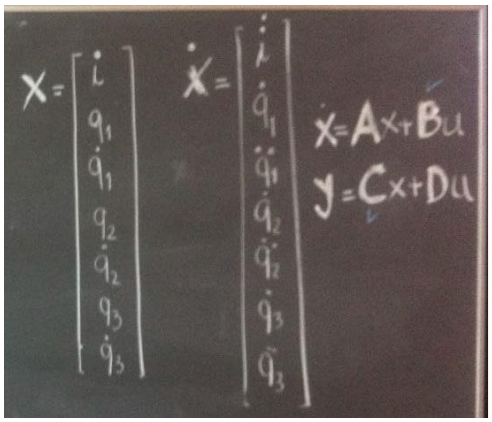
$$\mathbf{x} = [q_1, q_2, q_3, q_4, q_5, q_6, q_7]^T = [i, x_1, \dot{x}_1, x_2, \dot{x}_2, x_3, \dot{x}_3]^T$$

where

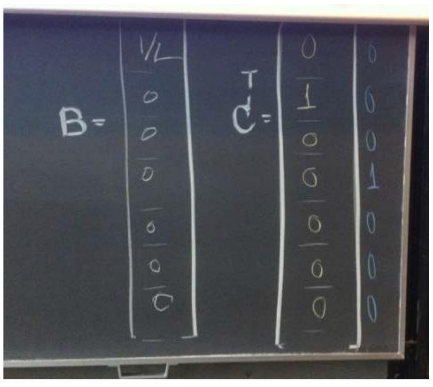
$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

Solution:

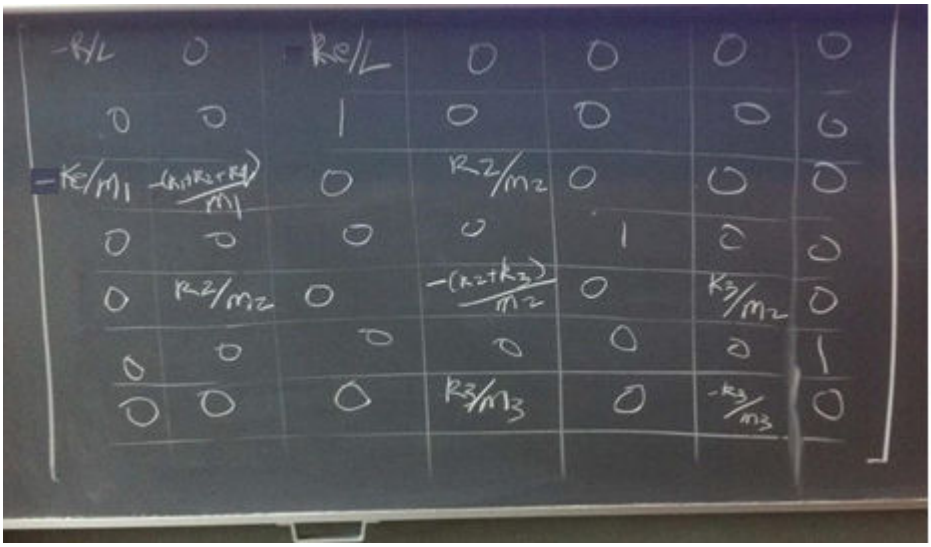
The time dependent variables are arranged into the defined state vector and the electro-mechanical model is arranged into the state space matrices ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ & $\mathbf{D}$). The resulting matrices are illustrated in Figure 2.



a)



b)



c)

Figure 2 : a) The arrangement of the states and the form of the state space model are shown ; b) the $\mathbf{B}-$ *and* $\mathbf{C}-$ *matrices is presented ; c) the* $\mathbf{A}-$ *matrix is illustrated .*

```
syms R L K_i K_s m1 m2 m3 k1 k2 k3

Ax = [
        -R/L         0          K_i/L       0          0    0    0
          0          0              1       0          0    0    0
       -K_i/m1 -(k1+k2+K_s)/m1    0      k2/m1        0    0    0
          0          0              0       0          1    0    0
          0       k2/m2             0  -(k2 + k3)/m2  0  k3/m2 0
          0          0              0       0          0    0    1
          0          0              0      k3/m3       0 -k3/m3 0 ]
```

Ax =

$$\begin{pmatrix} -\frac{R}{L} & 0 & \frac{K_i}{L} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{K_i}{m_1} & -\frac{K_s+k_1+k_2}{m_1} & 0 & \frac{k_2}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{k_2}{m_2} & 0 & -\frac{k_2+k_3}{m_2} & 0 & \frac{k_3}{m_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{k_3}{m_3} & 0 & -\frac{k_3}{m_3} & 0 \end{pmatrix}$$

```
Bx = [   1/L  0   0   0   0   0   0 ]'
```

Bx =

$$\begin{pmatrix} \frac{1}{L} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

```
Cx = [   0    1   0   0   0   0   0 ]
```

```
Cx = 1×7
     0     1     0     0     0     0     0
```

```
Dx = 0;
```

Here we have chosen as output the position of the first mass. We could have chosen another state variable by placing the 1 in $\mathbf{C}$ matrix in an other position or even change the dimensions of the matrix, so that we can measure more than one state variables like in the following selection of the output matrix, where we can obtain both the position and the velocity of the first mass.

```
C1 = [   0    1   0   0   0   0   0
         0    0   1   0   0   0   0 ]
```

```
C1 = 2×7
     0     1     0     0     0     0     0
     0     0     1     0     0     0     0
```

Problem 3

The goal of the problem is to investigate how the structure behaves when s_0 (air gap) is varied from 5 mm to 1 mm. The initial conditions of lateral displacements $x_2(0)$, $x_3(0)$ and velocities $\dot{x}_1(0)$, $\dot{x}_2(0)$, $\dot{x}_3(0)$ are zero and kept for all simulation cases. Only the lateral displacement of the lowest mass $x_1(0)$ is offset by 1 mm from center. Simulate three different values of s_0=5 mm, s_0=3 mm, s_0=1 mm. Explain the simulation results.

Solution:

The simulations can be carried out in simulink using matrix products as in the simulink diagram presented in Fiugre 3.

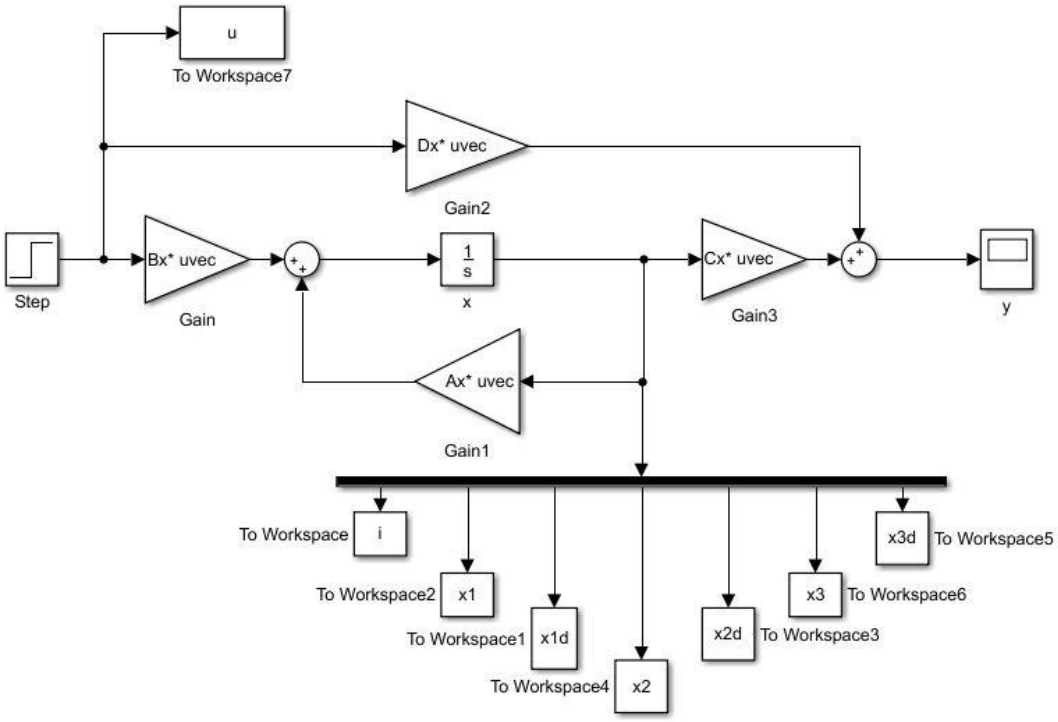
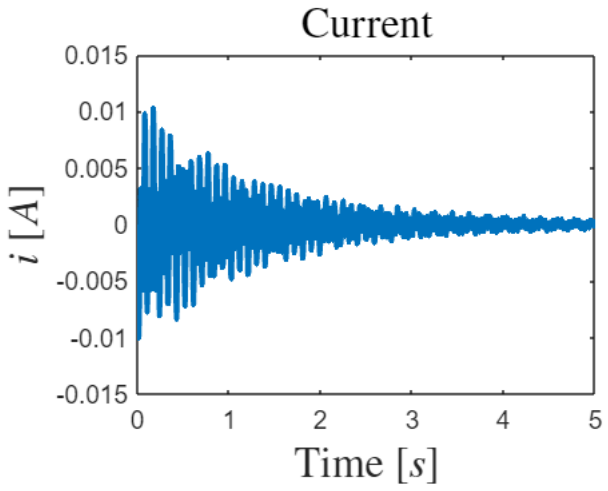


Figure 3 : Simulink block diagram of the state space model

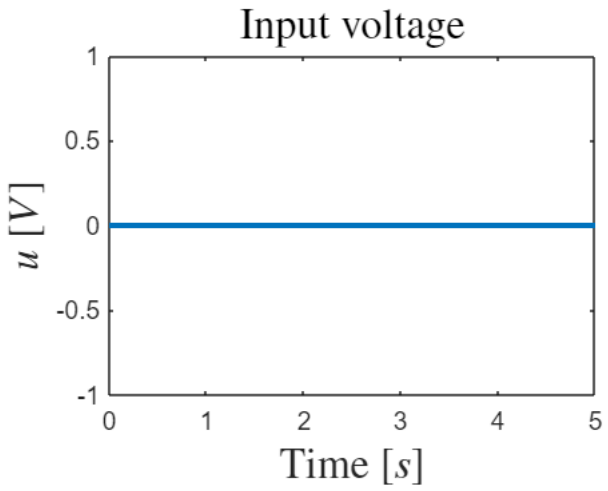
a) Behavior of system when s_0=5 mm.

```
% Parameters
m1 = 0.712;         % [kg] mass of platform 1
m2 = 0.428;         % [kg] mass of platform 1
m3 = 0.428;         % [kg] mass of platform 1

l = 24e-3; % mm, small
r = 14e-3 / 2;
A = pi*r^2;
R = 148.2;
L = 800e-3; % [H]
iMax = 0.14; % measured at 24v

gam = 0.5;          % bias ratio - normally chosen between 0.2-0.5
ib = iMax*gam;      % Bias current
mu = 4*pi*1e-7;     % Permeability of free space (vacuum)
s0 = 5e-3;          % displacement operation point

b = 0.025;          % [m] beam width
h = 1.0e-3;         % [m] beam thickness
N = sqrt(L*2*s0 / (mu*A)); % estimate number of windings based on above
formular
E = 2.0e11;         % [N/m2] elasticity modulus of steel
Iz = (b*h^3)/12;    % [m4] area moment of inertia

L1= 0.145;           % [m] length of beam 1
```

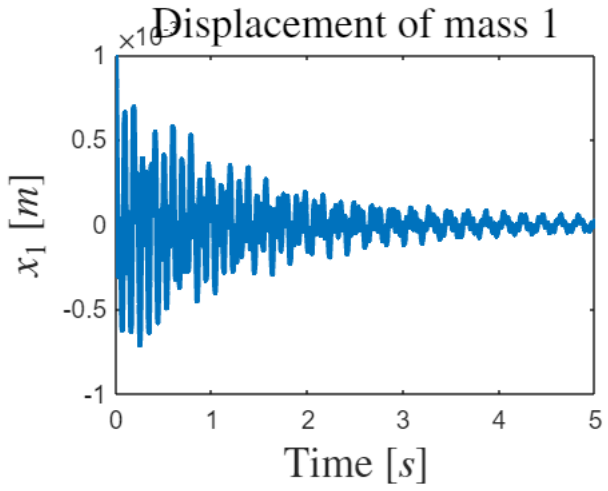
```
L2= 0.134;            % [m] length of beam 2
L3= 0.229;            % [m] length of beam 3

% Force constants obtained from the linearisation

k1 = 2*12*E*Iz/L1^3;   % [N/m] stiffness of beam 1
k2 = 2*12*E*Iz/L2^3;   % [N/m] stiffness of beam 2
k3 = 2*12*E*Iz/L3^3;   % [N/m] stiffness of beam 3

Ki = mu*N^2*A*ib / s0^2;
Ks = -mu*N^2*A*(ib^2)/s0^3;

Ax = [
        -R/L        0          Ki/L       0         0    0   0
          0         0             1       0         0    0   0
       -Ki/m1 -(k1+k2+Ks)/m1      0     k2/m1       0    0   0
          0         0             0       0         1    0   0
          0       k2/m2           0  -(k2 + k3)/m2  0  k3/m2 0
          0         0             0       0         0    0   1
          0         0             0     k3/m3       0 -k3/m3 0 ];

Bx = [   1/L  0   0   0   0   0   0 ]';

Cx = [   0    1   0   0   0   0   0 ];

Dx = 0;

% initial conditions
x_0 = [0 0.001 0 0 0 0 0]';

% input
u_0 = 0;


%% Simulations
sim('electroMagnetModelStateSpace');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```



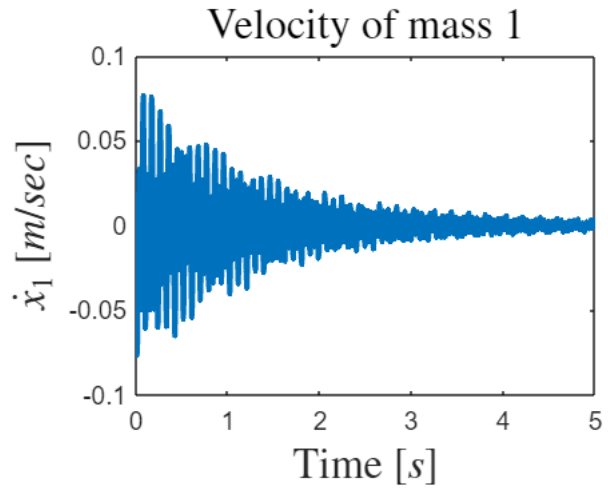

```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



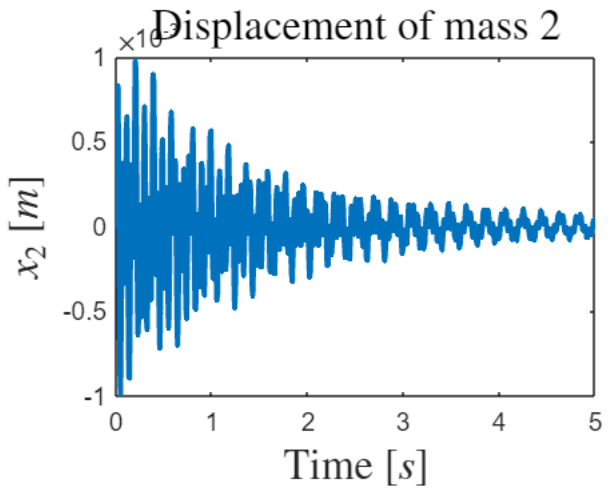

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



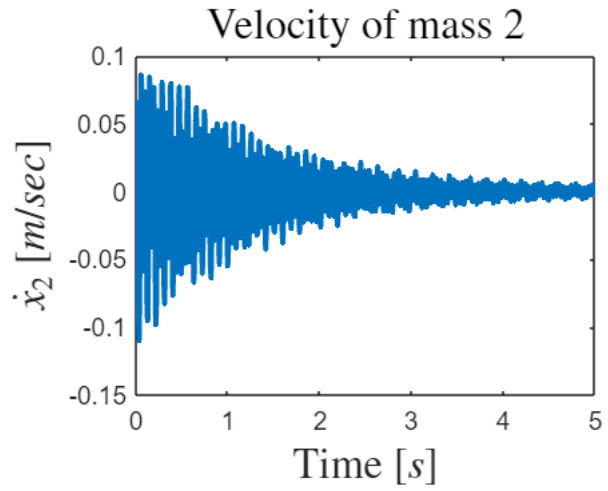

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



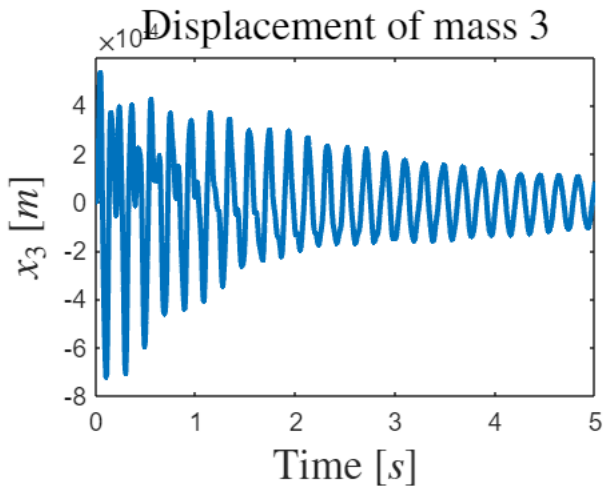

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



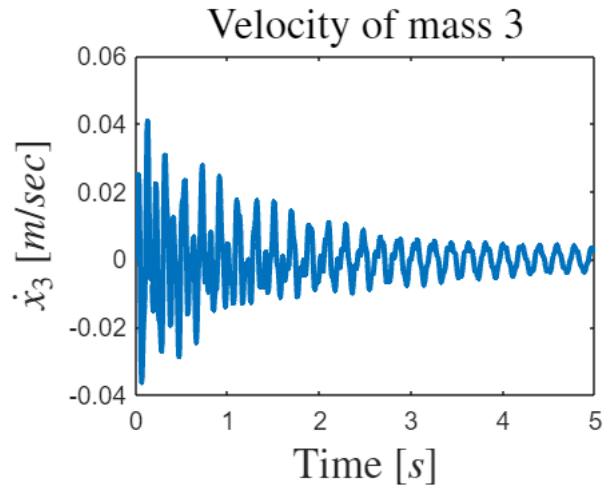

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```

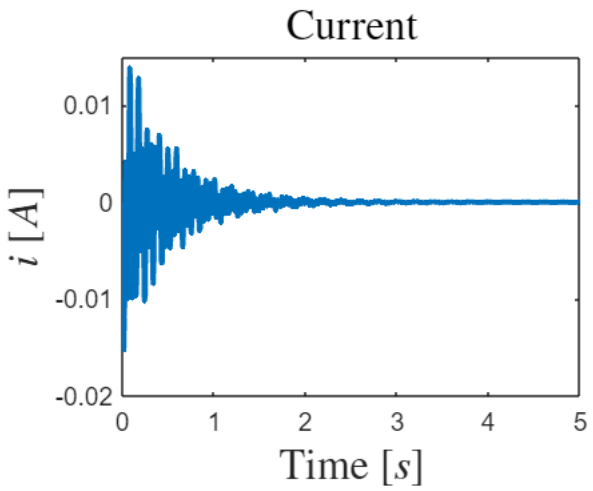


As we can see in the plots, when there is no external input (voltage or forces) acting on the system, the responses decay over time to their equilibrium positions, i.e. zero displacement, zero velocity and zero current. This is expected, since the system is very similar to the mass-spring-damper system. After approximately 10 seconds all the transient behaviors of the states have died out and the state settle to its equilibrium point $\mathbf{x}(t) = [0\ 0\ 0\ 0\ 0\ 0\ 0]^T$.

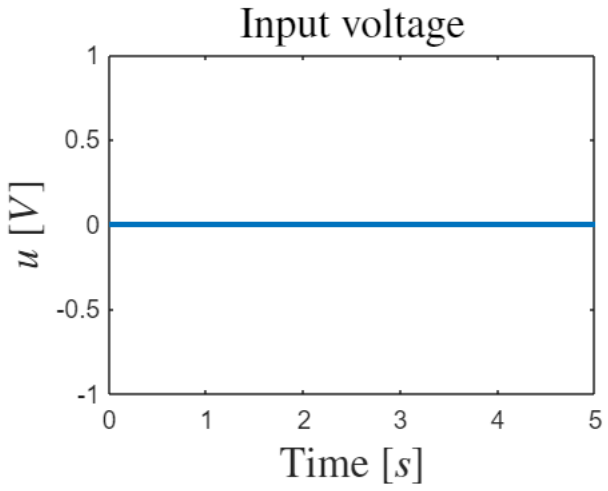
b) Behavior of system when s_0=3 mm.

```
s0 = 3e-3;          % displacement operation point

N = sqrt(L*2*s0 / (mu*A)); % estimate number of windings based on above
formular
```

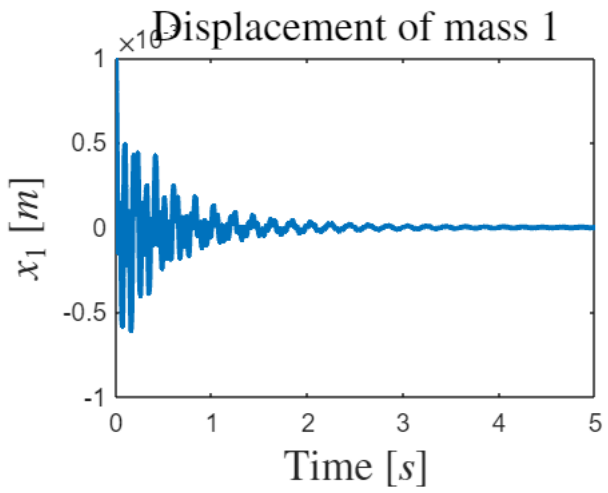
```
Ki = mu*N^2*A*ib / s0^2;
Ks = -mu*N^2*A*(ib^2)/s0^3;

Ax = [
        -R/L        0          Ki/L       0         0    0   0
          0         0             1       0         0    0   0
       -Ki/m1 -(k1 + k2 + Ks)/m1  0     k2/m1       0    0   0
          0         0             0       0         1    0   0
          0       k2/m2           0  -(k2 + k3)/m2  0  k3/m2 0
          0         0             0       0         0    0   1
          0         0             0     k3/m3       0 -k3/m3 0 ];

%% Simulations
sim('electroMagnetModelStateSpace');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```



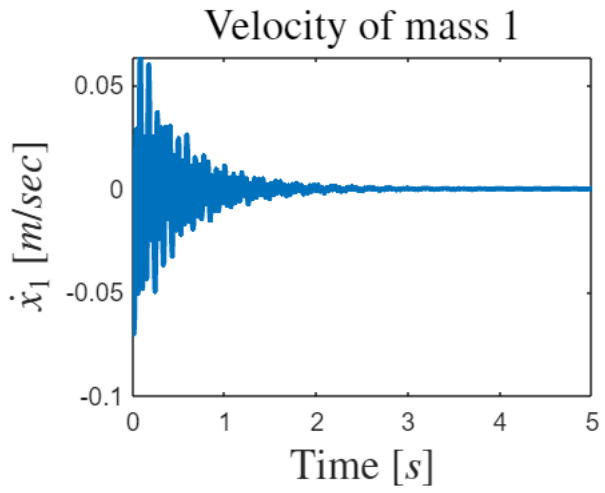

```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



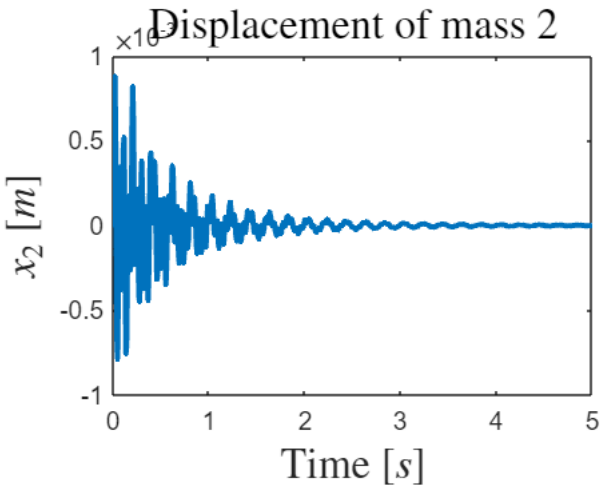

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



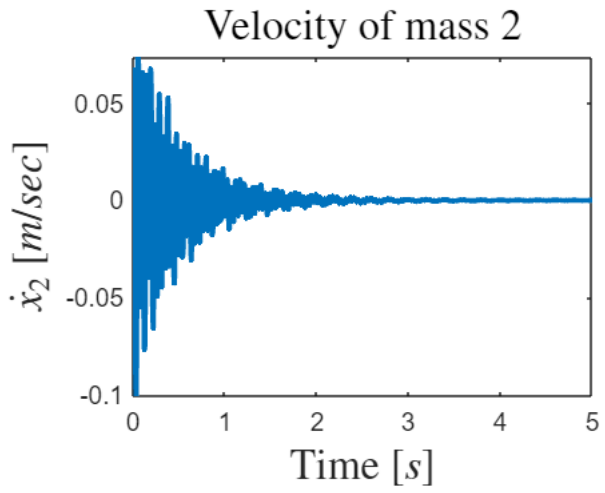

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



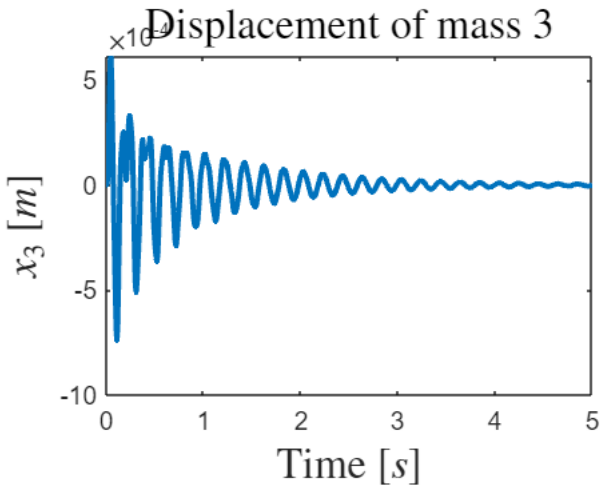

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



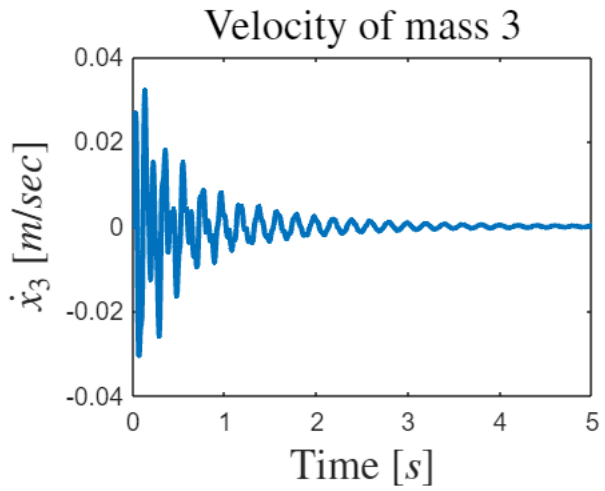

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```



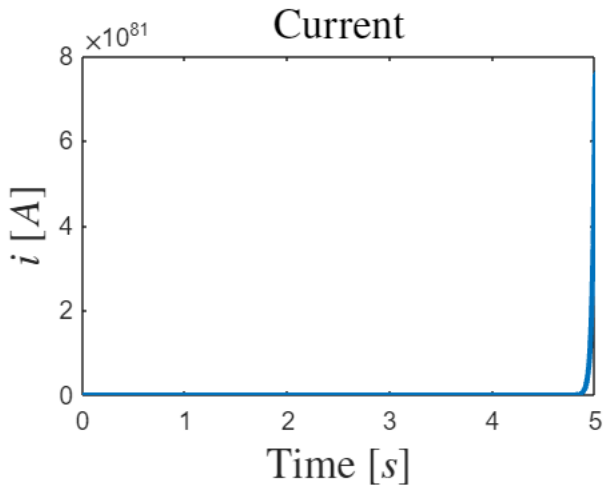
The behavior of the system is similar to the previous case with the difference that the state variables now settle to their equilibrium positions much faster, almost in 3 seconds. This can be explained by the fact that smaller air gaps between the lowest mass and the magnets generate larger magnetic forces, which in turn drive the system dynamics faster. This can be also deducted from the equation describing the magnetic force.

$$f = \frac{\mu_0 n^2 A_0}{4}\left(\frac{(i_0 + i(t))^2}{(s_0 - x_1(t))^2} - \frac{(i_0 + i(t))^2}{(s_0 + x_1(t))^2}\right)$$

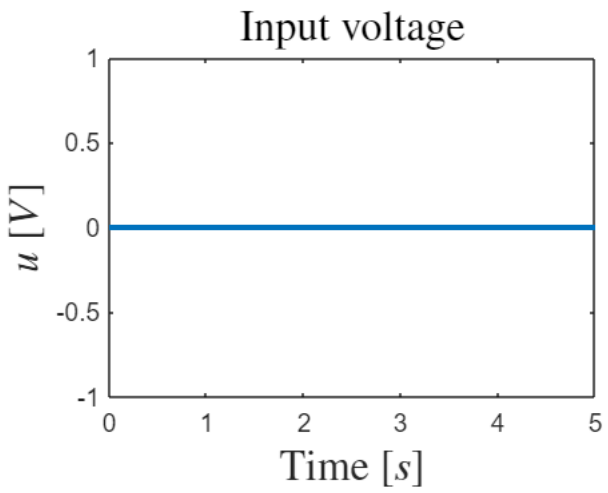
c) Behavior of system when s_0=1 mm.

```
s0 = 1e-3;          % displacement operation point

N = sqrt(L*2*s0 / (mu*A)); % estimate number of windings based on above
formular

Ki = mu*N^2*A*ib / s0^2;
Ks = -mu*N^2*A*(ib^2)/s0^3;

Ax = [
        -R/L         0          Ki/L        0          0    0   0
          0          0             1        0          0    0   0
       -Ki/m1 -(k1 + k2 + Ks)/m1  0      k2/m1        0    0   0
          0          0             0        0          1    0   0
          0        k2/m2           0  -(k2 + k3)/m2   0  k3/m2 0
          0          0             0        0          0    0   1
          0          0             0      k3/m3        0 -k3/m3 0 ];

%% Simulations
sim('electroMagnetModelStateSpace');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
```

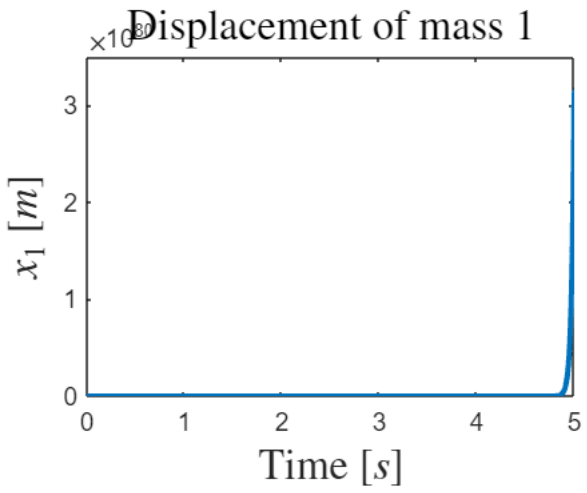
```
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```



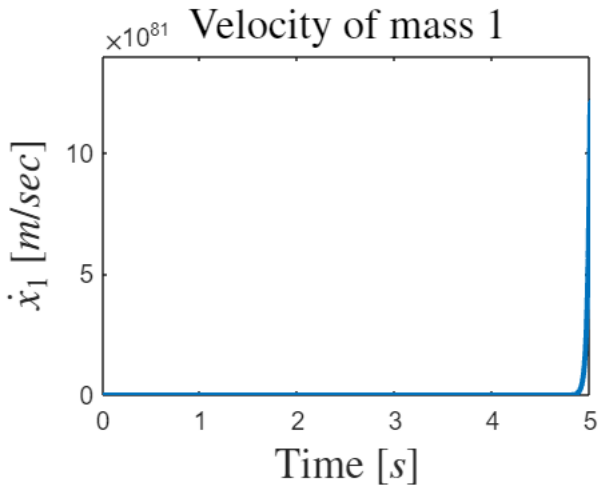

```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



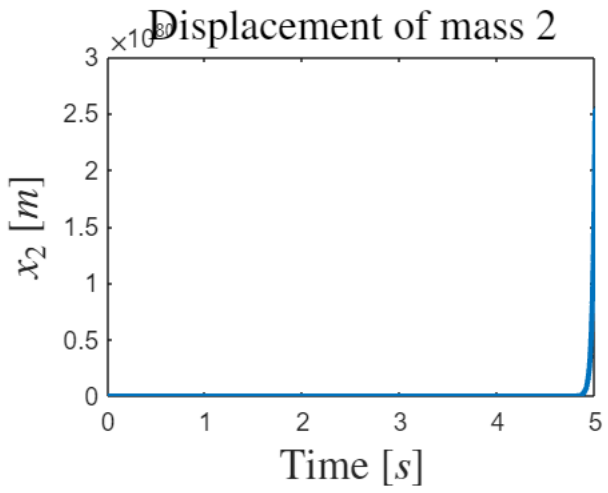

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



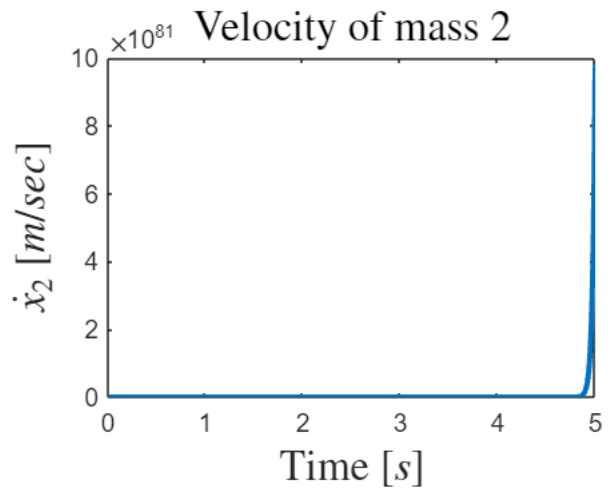

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



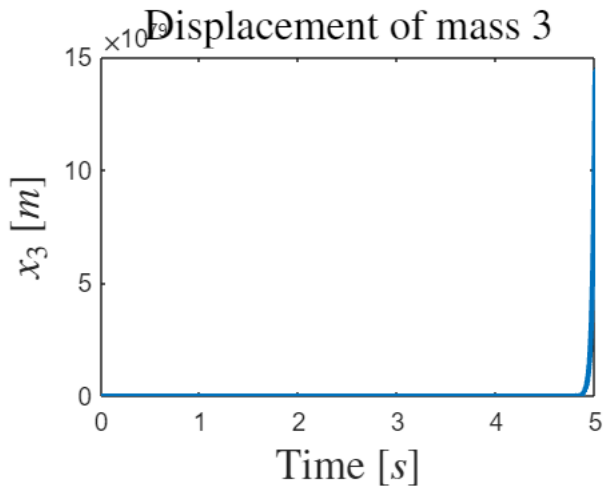

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



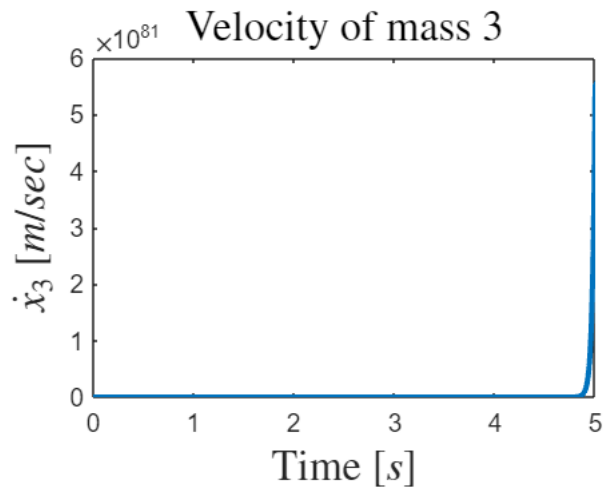

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```



Last, the system is simulated with s_0=1 mm. As it can be seen in the plots the system becomes unstable, i.e. the state variables do not settle to any equilibrium point but they keep growing unbounded. This can be explained by the fact that in very small distances the magnetic force is too large to allow the system to settle anywhere. Of course if the plots were real sensor data, we should expect to see the states reaching a saturation level since the system is physically constraint.

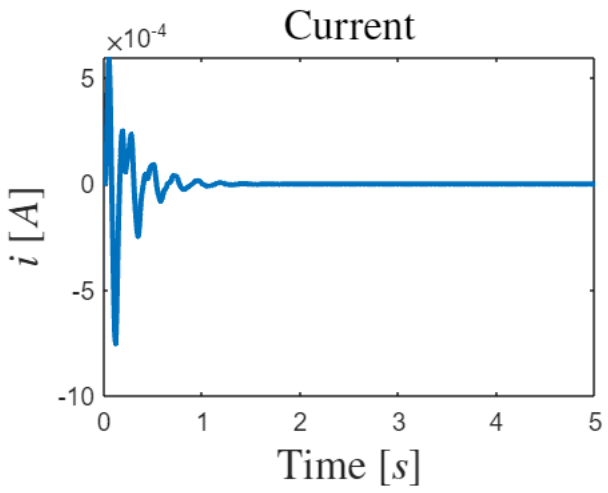
Problem 4

What is the maximum lateral displacement experienced by the three masses when an initial condition of velocity of 5 mm/s acts on mass 3, assuming s_0=2 mm?

Solution:

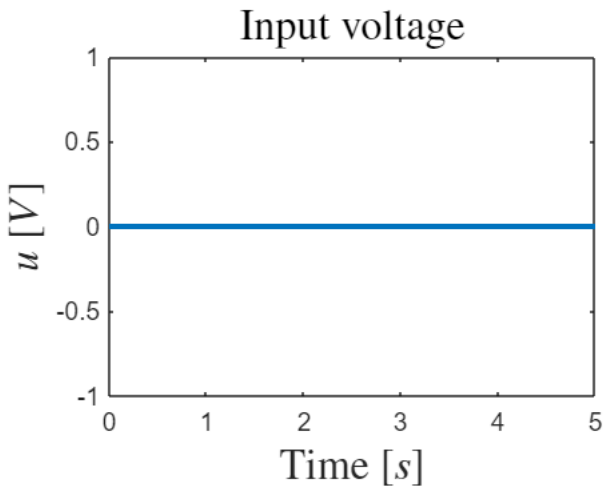
```
% initial conditions
```

```
x_0 = [0 0 0 0 0 0 0.005]';

s0 = 2e-3;          % displacement operation point

N = sqrt(L*2*s0 / (mu*A)); % estimate number of windings based on above
formular

Ki = mu*N^2*A*ib / s0^2;
Ks = -mu*N^2*A*(ib^2)/s0^3;

Ax = [
        -R/L        0          Ki/L       0         0    0   0
          0         0             1       0         0    0   0
       -Ki/m1 -(k1 + k2 + Ks)/m1  0     k2/m1       0    0   0
          0         0             0       0         1    0   0
          0       k2/m2           0  -(k2 + k3)/m2  0  k3/m2 0
          0         0             0       0         0    0   1
          0         0             0     k3/m3       0 -k3/m3 0 ];

%% Simulations
sim('electroMagnetModelStateSpace');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```

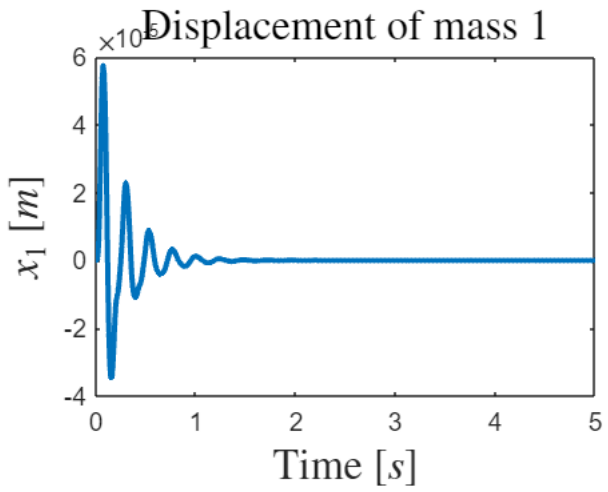



```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
```

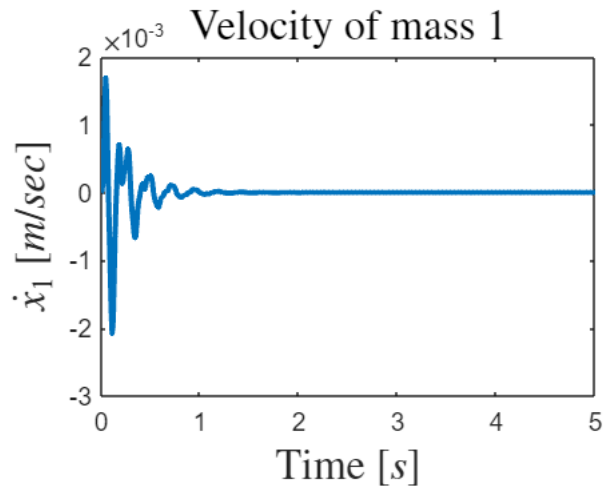
```
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



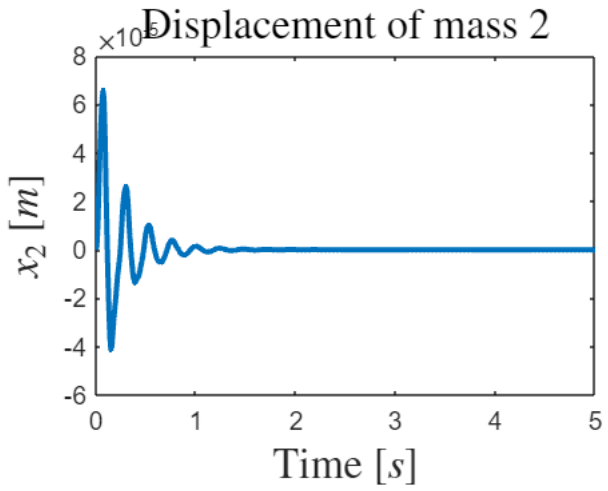

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



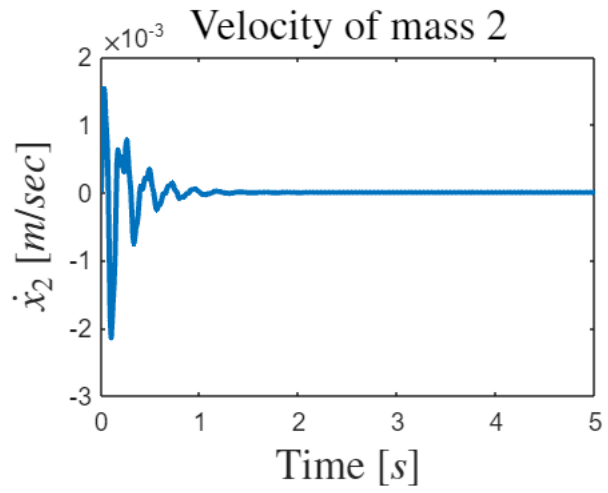

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



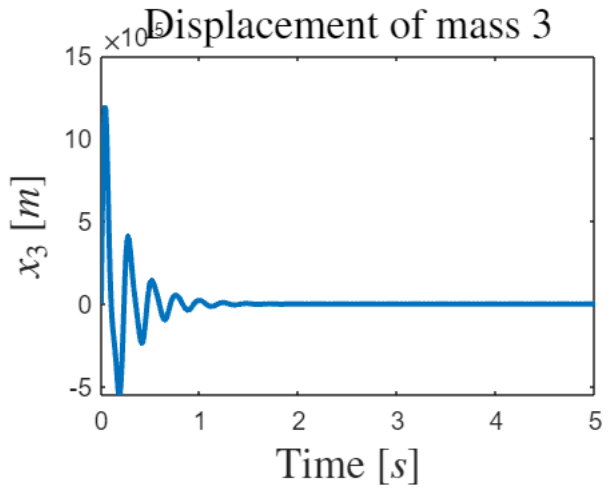

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



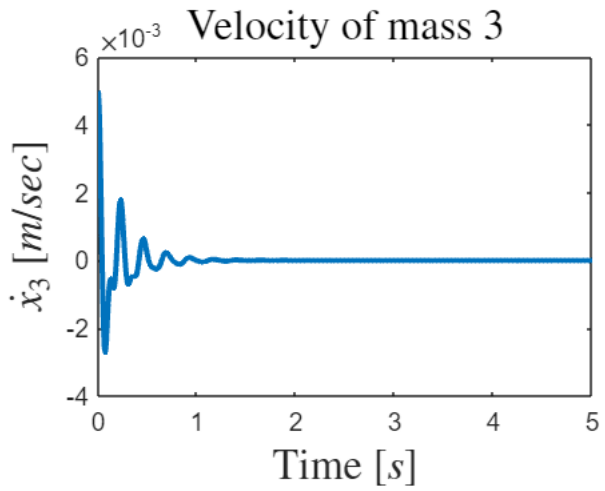

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```



From the time response of the state variables corresponding to the displacement of the masses from the simulation (the three plots on the left hand side) we can measure the maximum displacement for each mass as

$$x_{1,max} = 5.74 \cdot 10^{-5} m$$
$$x_{2,max} = 6.57 \cdot 10^{-5} m$$
$$x_{3,max} = 1.18 \cdot 10^{-5} m$$

The displacement of the third (upper) mass is larger, almost double, in comparison to the other displacements. This is expected since the third mass on the top of the structure is globally more flexible, accumulating the deflection of the two lower masses.

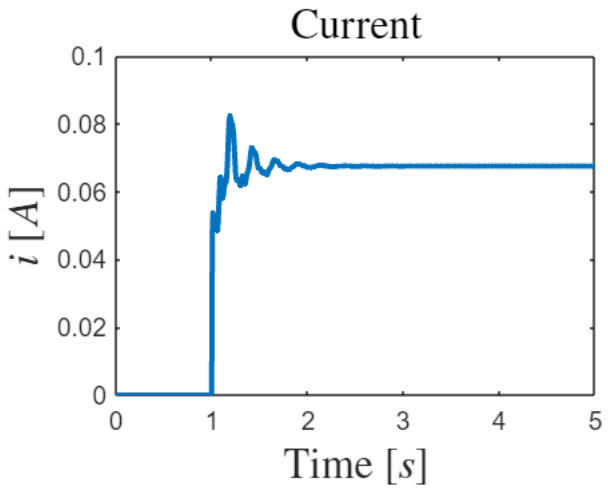
Problem 5

A step function $u(t)$=10 V is send to the electromagnets, assuming all initial conditions of lateral displacement, velocity and current nill. How will the three masses vibrate and how will the current change, i.e. amplitudes and frequencies.

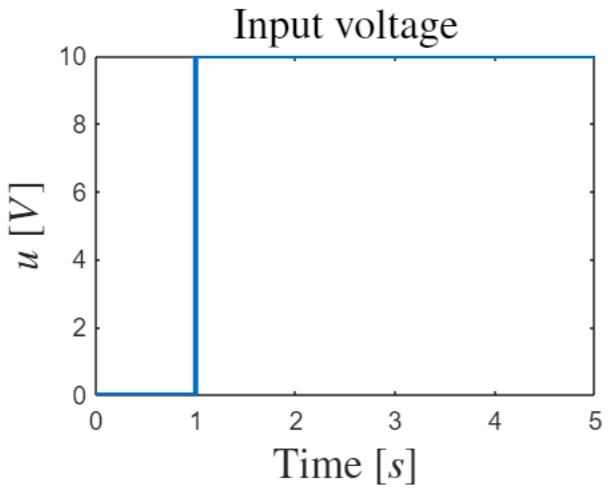
Solution:

```
% initial conditions
x_0 = [0 0 0 0 0 0 0]';

% input
u_0 = 10;

%% Simulations
sim('electroMagnetModelStateSpace');

%% Plots
time = i.time;

figure
plot(time,i.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
```

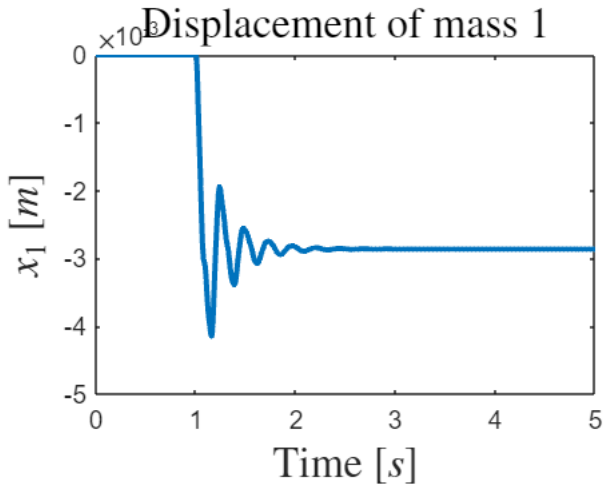
```
ylabel('$i$ $[A]$', 'FontSize',16,'Interpreter','latex');
title('Current', 'FontSize',16,'Interpreter','latex');
```



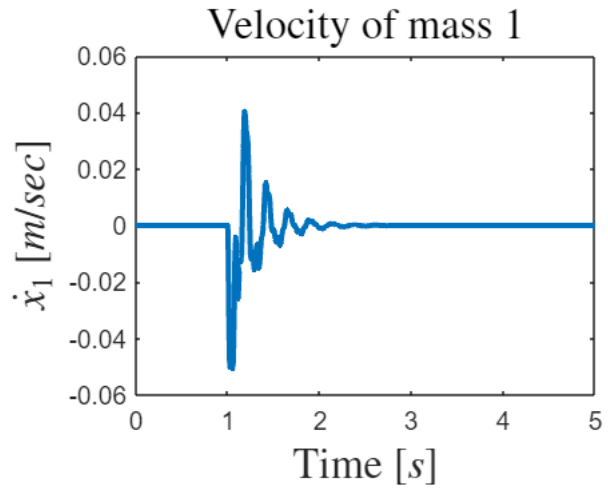

```
figure
plot(time,u.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$u$ $[V]$', 'FontSize',16,'Interpreter','latex');
title('Input voltage', 'FontSize',16,'Interpreter','latex');
```



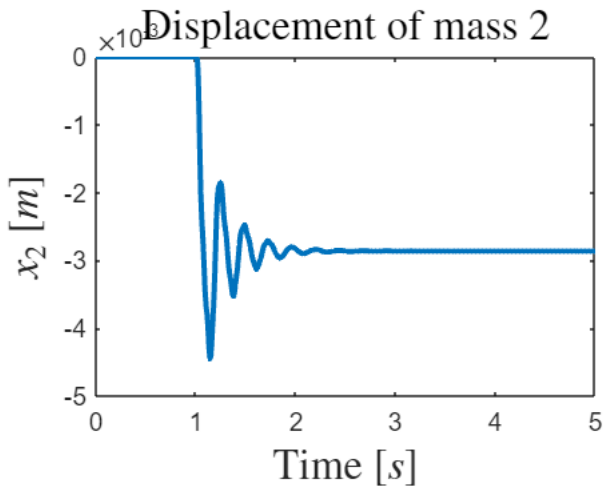

```
figure
plot(time,x1.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_1$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 1', 'FontSize',16,'Interpreter','latex');
```



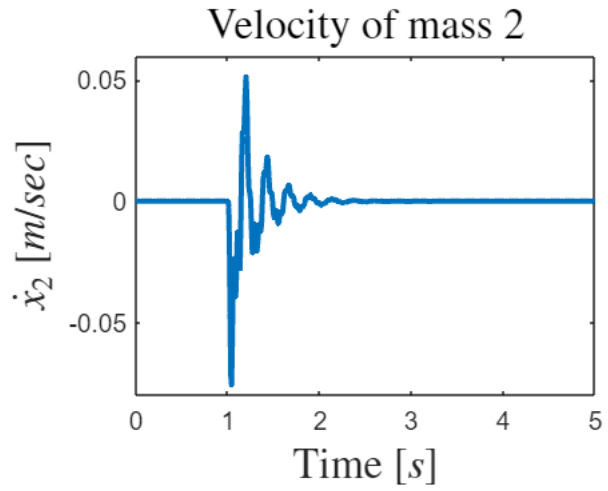

```
figure
plot(time,x1d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_1$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 1', 'FontSize',16,'Interpreter','latex');
```



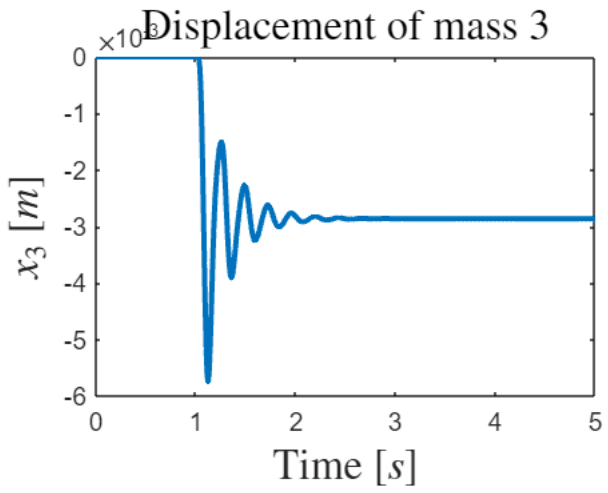

```
figure
plot(time,x2.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_2$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 2', 'FontSize',16,'Interpreter','latex');
```



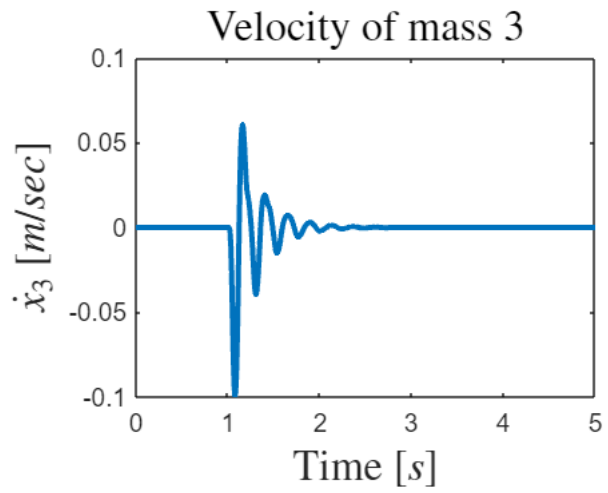

```
figure
plot(time,x2d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_2$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 2', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$x_3$ $[m]$', 'FontSize',16,'Interpreter','latex');
title('Displacement of mass 3', 'FontSize',16,'Interpreter','latex');
```




```
figure
plot(time,x3d.signals.values, 'LineWidth', 2);
xlabel('Time $[s]$', 'FontSize',16,'Interpreter','latex');
ylabel('$\dot{x}_3$ $[m/sec]$', 'FontSize',16,'Interpreter','latex');
title('Velocity of mass 3', 'FontSize',16,'Interpreter','latex');
```



As it is shown by the plots, after the transient behavior of the states has died out, the latter settle at a constant steady-state value. The velocities, as expected settle to zero, while the positions assume a nonzero value since there is a constant voltage that keeps exciting the system. The transient behavior can be described as a fading oscillatory response, which is expected, given that there is damping in the system. From the plots, by measuring the period (approximately) of the oscillations and the amplitude of the responses we can approximate the magnitude and the frequency of the signals. In the next lectures, when the modal analysis will be introduced, the relation between the oscillatory behavior of the responses and the eigenvalues of the system matrix $\mathbf{A}$ will provide an accurate method of calculating the frequencies and amplitudes of the state responses.

Problem 6

Compare the obtained results from Group Work Module 2 with the results from this Group Work Module.

Solution:

It is the exact same results.

Linear Control Design II - Group Work Problem Module 4 Solution

Description

Problem 1 Solve Problem 2.10 from the textbook.

Solution:

From the problem text four heat power equations are given:

$$q_c = k_c(T_s - T_a)$$
$$q_g = k_g(T_s - T_g)$$
$$q_r = k_r(T_s^4 - T_a^4)$$
$$q = ku$$

Remember that all temperatures are absolute temperatures.

1.1 Nonlinear model and state space model

The total heat capacity of the oven air space and the product are denoted by C_s and C_g respectively. The expressions for the time derivatives of the temperatures of the different parts of the system can be written as: the change of heat = heat in - heat out.

For the oven the heat balance can be written as:

$$C_s \frac{dT_s}{dt} = q - q_c - q_g - q_r$$

whereas for the product the heat balance can be written as:

$$C_g \frac{T_g}{dt} = q_g$$

T_s and T_g are chosen as state variables: $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} T_s \\ T_g \end{bmatrix}$. T_a is considered as a disturbance and is called v.

The system input is u. The state space model can now be written as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \frac{1}{C_s}(ku - k_c(x_1 - v) - k_g(x_1 - x_2) - k_r(x_1^4 - v^4)) \\ \frac{1}{C_g}k_g(x_1 - x_2) \end{bmatrix} = \begin{bmatrix} \frac{1}{C_s}((-k_c - k_g)x_1 - k_r x_1^4 + k_g x_2) \\ \frac{1}{C_g}k_g(x_1 - x_2) \end{bmatrix} + \begin{bmatrix} \frac{1}{C_s}k \\ 0 \end{bmatrix} u + \begin{bmatrix} \frac{1}{C_s}(k_c v + k_r v^4) \\ 0 \end{bmatrix}$$

$$y = [0 \quad 1]\mathbf{x}$$

1.2 Stationary state

All time derivatives in $C_s \frac{dT_s}{dt}$ and $C_g \frac{T_g}{dt}$ are set equal to zero and the equations are solved for the state vector $\mathbf{x}$ and the control input u

$$0 = q - q_c - q_r - q_g$$
$$0 = q_g u$$

This leds to:

$$T_{s0} = T_{g0}$$
$$0 = q - q_c - q_r \leftrightarrow$$
$$q = q_c - q_r \leftrightarrow$$
$$u_0 = \frac{1}{k}(k_c(T_{s0} - T_a) + k_r(T_{s0}^4 - T_a^4))$$

1.3 Linearization

Linearization around the stationary point can now be carried out. We define:

$$T_s = x_1 = x_{10} + \Delta x_1$$
$$T_g = x_2 = x_{20} + \Delta x_2$$
$$u = u_0 + \Delta u$$
$$T_a = v = v_0 + \Delta v$$

The components of the vector field $\mathbf{f}(\mathbf{x}, u, v)$ are given by

$$f_1 = \dot{x_1} = \frac{1}{C_s}(ku - (k_c + k_g)x_1 + k_g x_2 + k_c v - k_r x_1^4 + k_r v^4)$$
$$f_2 = \dot{x_2} = \frac{1}{C_g} k_g (x_1 - x_2)$$

The system matrices can now be derived using formula (2.66), (2.68), (2.69) and (2.71) from the textbook.

$$A = \frac{\partial \mathbf{f}(\mathbf{x}_0, u_0, v_0)}{\partial \mathbf{x}} = \begin{bmatrix} \frac{1}{C_s}(-k_g - k_c - 4k_r x_{10}^3) & \frac{k_g}{C_s} \\ \frac{k_g}{C_g} & -\frac{k_g}{C_g} \end{bmatrix}$$

$$B = \frac{\partial \mathbf{f}(\mathbf{x}_0, u_0, v_0)}{\partial u} = \begin{bmatrix} \frac{k}{C_s} \\ 0 \end{bmatrix}$$

$$B_v = \frac{\partial \mathbf{f}(\mathbf{x}_0, u_0, v_0)}{\partial v} = \begin{bmatrix} \frac{1}{C_s}(k_c + 4k_r v_0^3) \\ 0 \end{bmatrix}$$

$$C = [0 \quad 1]$$

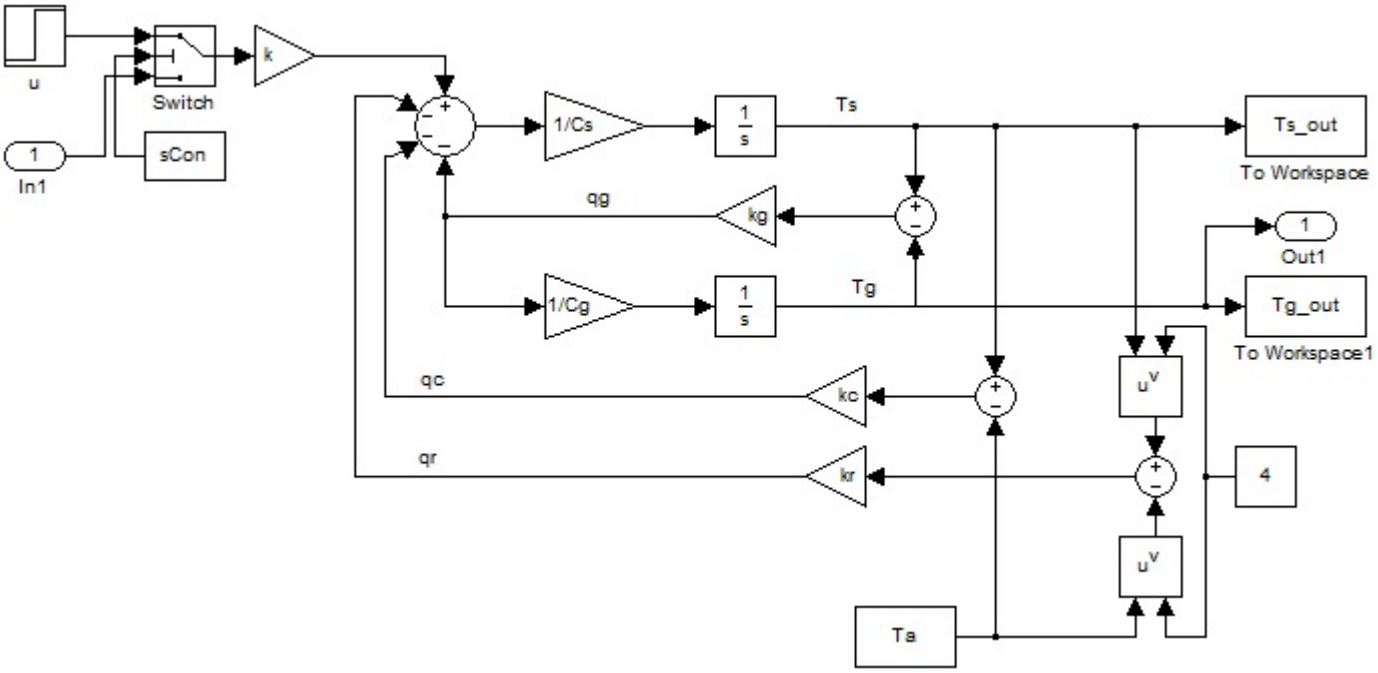


Figure 1: Simulink block diagram of the model

Problem 2 Implement a model of the system in the Simulink environment, and use the following numerical values for the model parameters

$$k_c = 0.85 \ W/^\circ C$$

$$k_g = 1.2 \ W/^\circ C$$

$$k_r = 7.8 \cdot 10^{-10} \ W/K^4$$

$$C_s = 75 \ J/^\circ C$$

$$C_g = 800 \ J/^\circ C$$

$$k = 100 \ W/V$$

The control signal u can assume values in the interval $0 - 10\ V$.

Note that you must use absolute temperatures.

Carry out simulations on the model with the initial values $u_0 = 5\ V$, $T_{si} = 200^{\circ}C$, $T_{gi} = 25^{\circ}C$ and $T_a = 25^{\circ}C$. Determine the stationary state by simulation until all variables have assumed constant values.

Solution

Simulink model

A Simulink model of the system can be seen on figure 1, where the given data has been used. Remember to insert the initial temperature values in the integrators and set the priority of the integrator.

The stationary state is found by simulating until all variables have assumed constant values.

```
%The constants...
kc = 0.85; %Watt/deg.C
kg = 1.2;  %Watt/deg.C
kr = 7.8e-10; %Watt/K^4
Cs = 75;   %Joule/deg.C
Cg = 800;  %Joule/deg.C
k  = 100;  %Watt/volt
Ta = 298.15; %K
u0 = 5; %V

 % u0 is set to the constant value of 5 volt.
 % Ta is set to the constant value of 298.15 K.
 % Tsi is set to 473.15 K in the integrator initial condition
 % Tgi is set to 298.15 K in the integrator initial condition

 % After around 8000 seconds the temperature has stabilised. Tg becomes 688
 % K and Ts becomes 688 K... they are end up having the same value... wauw.

 sCon = 1; %setting sCon to 1 take the stepinput, setting sCon to 0 take the
IN.
 sim('Q2')

 figure
 plot(Ts_out.time,Ts_out.signals.values,'b' ,Ts_out.time,
Tg_out.signals.values,'r')
 xlabel('Time');
 ylabel('Temperature [Kelvin]');
 title('Ts and Tg vs. time');
 legend('Ts', 'Tg');
```

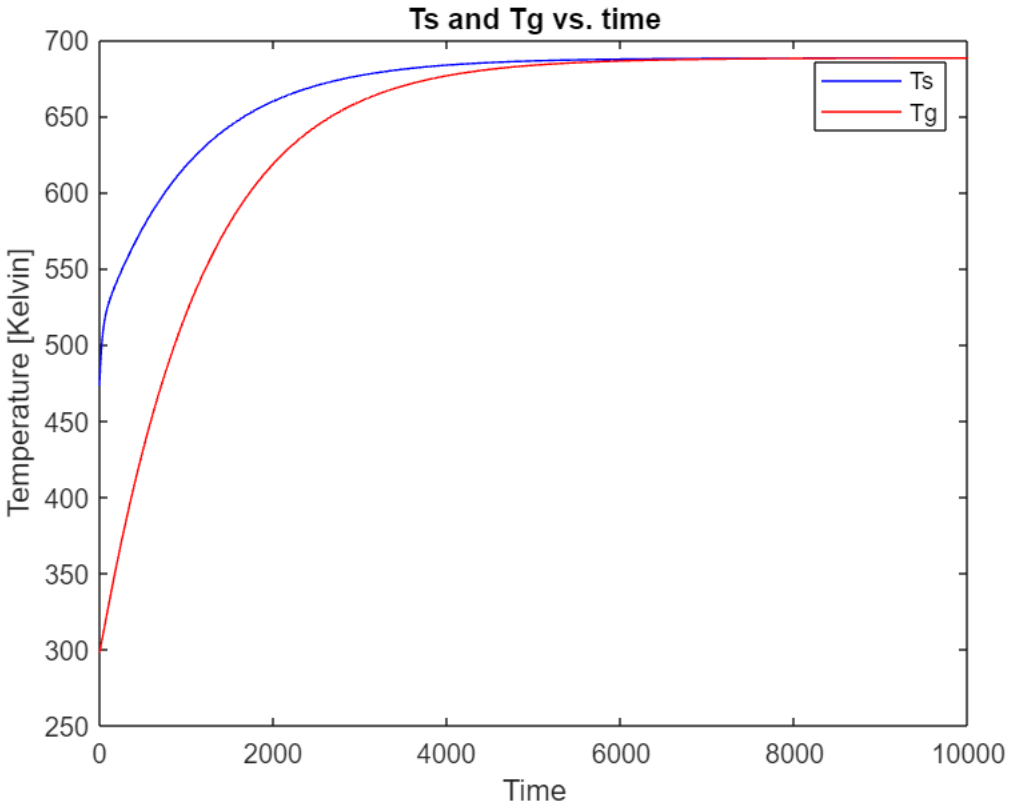


Figure 2: Steady state response of the system

As seen on figure 2 the values stabilizes around 688 Kelvin corresponding to 414.85 degrees Celsius. The settling time is around 5500 time units. If you insert these stationary values along with the initial data in equation (15) then you get an input of $u = 5.06$ which is very close to the given input. Higher accuracy may be gained by simulating over a longer period.

Problem 3 Linearize the nonlinear Simulink model applying the Matlab function `linmod`. Use the function as follows

```
[A,B,C,D] = linmod('modelname',x0,u0)
```

where x_0 and u_0 are the stationary states and inputs.

Compare the result with that from the manual linearization from question "c" in the Problem 2.10.

Find the eigenvalues of the $\mathbf{A}$ matrix by using the Matlab function `eig()`. Repeat the linearization for $u_0 = 2\ V$ and $u_0 = 8\ V$; $T_a = 10^{\circ}C$ and $T_a = 35^{\circ}C$, and note

how the linear model changes with the stationary state.

Remove the radiation effect (set $k_r = 0$) and repeat the linearization for different values u_0.

Solution

Linearization with Matlab

When using the Matlab command **`linmod`** on the Simulink model the following matrices are obtained

$$A = \begin{pmatrix} -0.0317 & 0.0160 \\ 0.0015 & -0.0015 \end{pmatrix}$$
$$B = \begin{pmatrix} 1.333 \\ 0 \end{pmatrix}$$
$$C = (1 \quad 0)$$
$$D = (0)$$

If we insert the values for the constants in the matrices in Section 1.3 we get the same values as in the matrices numerically calculated in Matlab.

```
%First get initial state and then use the linmod command.
 sCon = 0; %setting sCon to 1 take the stepinput, setting sCon to 0 take the
IN.
 x = [Ts_out.signals.values(end); Tg_out.signals.values(end)];
 [A,B,C,D]=linmod('Q2',x,[u0])
```

```
A = 2×2
   -0.0409    0.0160
    0.0015   -0.0015
B = 2×1
    1.3333
         0
C = 1×2
     0     1
D = 0
```

```
 eigenval = eig(A)
```

```
eigenval = 2×1
   -0.0415
   -0.0009
```

The eigenvalues of the dynamic matrix are found using `eig(A)`: $\lambda_1 = -0.0325$ and $\lambda_2 = -0.0007$. Note that both eigenvalues are in the left-half complex plane, hence the system is stable.

At $u_0 = 2$ V the stationary value for T_s and T_g is 488 K and T_s will decline initially but start climbing again after around 120 time units. The settling time of the responses have increased making the system response slower

than with $u_0 = 5$ V. At $u_0 = 8$ V the stationary value for T_s and T_g is 824 K and both responses have shorter settling times than with $u_0 = 5$ V. The system has become faster.

With u_0 back at the original 5 volts, the effect of T_a is tested.

Figure 3 shows the three graphs in the same window, the topmost (blue,red) pair is at $T_a = 308.15$ K, the middle is at $T_a = 298.15$ K and the bottommost is at $T_a = 283.15$ K. It is clear that the higher temperature the higher the stationary value, when the outside temperature is low the internal temperature will drop and vice versa.

```
Ta1 = [308.15 298.15 283.15]; %K
u0 = 5; %V
 figure
  for i=1:3
       Ta=Ta1(i);
 sCon = 1; %setting sCon to 1 take the stepinput, setting sCon to 0 take the
IN.
 sim('Q2')


  plot(Ts_out.time,Ts_out.signals.values,'b', Ts_out.time,
Tg_out.signals.values,'r')
  hold on
 xlabel('Time');
 ylabel('Temperature [Kelvin]');
 title('Ts and Tg vs. time');
 legend('Ts', 'Tg');
 end
```

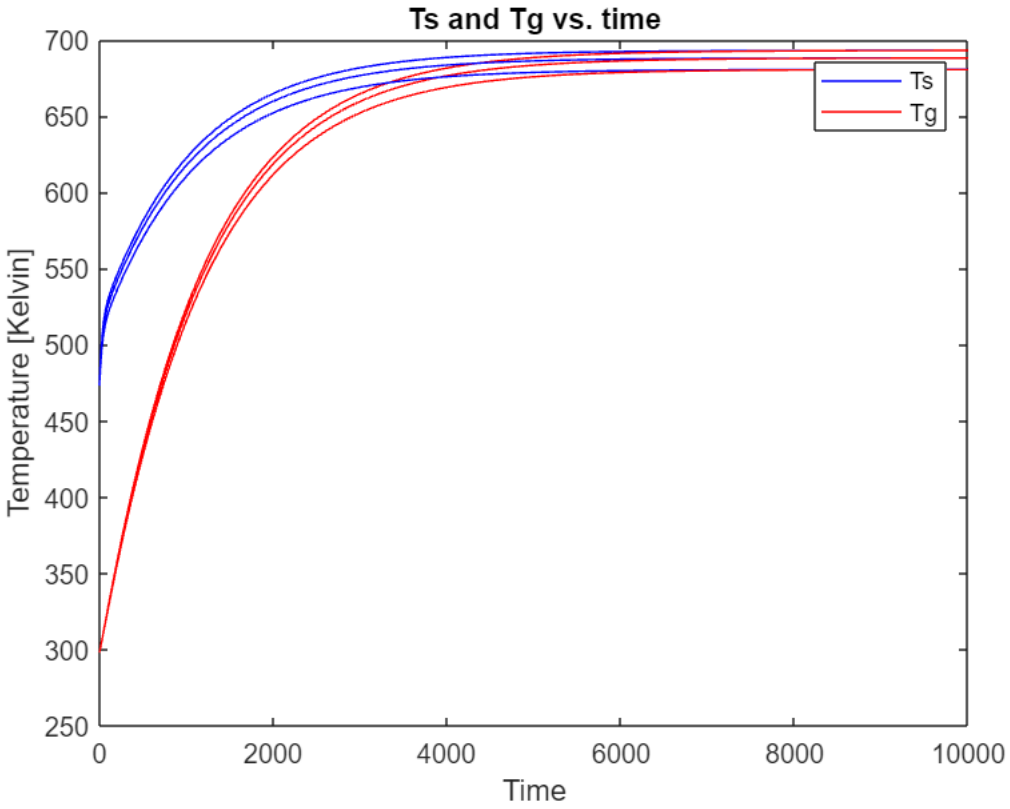


Figure 3: The effect of changing T_a among 308.15 K (top), 298.15 K (middle) and 283.15 K (lowest).

If the radiation effect is neglected $k_r = 0$ then the stationary point will increase to around 880 K with a longer settling time. This is also reflected in the linearized system whose dynamics is characterized by smaller eigenvalues ($\lambda_1 = -0.0282$ and $\lambda_2 = -0.0006$). Fig.4 shows the difference between $k_r = 0$ (top) and $k_r = 7.8 * 10^{-10}$ (bottom).

```
kr1 = [0 7.8e-10]; %K
u0 = 5; %V
 figure
  for i=1:2
       kr=kr1(i);
 sCon = 1; %setting sCon to 1 take the stepinput, setting sCon to 0 take the
IN.
 sim('Q2')


  plot(Ts_out.time,Ts_out.signals.values,'b', Ts_out.time,
Tg_out.signals.values,'r')
  hold on
 xlabel('Time');
 ylabel('Temperature [Kelvin]');
 title('Ts and Tg vs. time');
 legend('Ts', 'Tg');
```

```
end
```

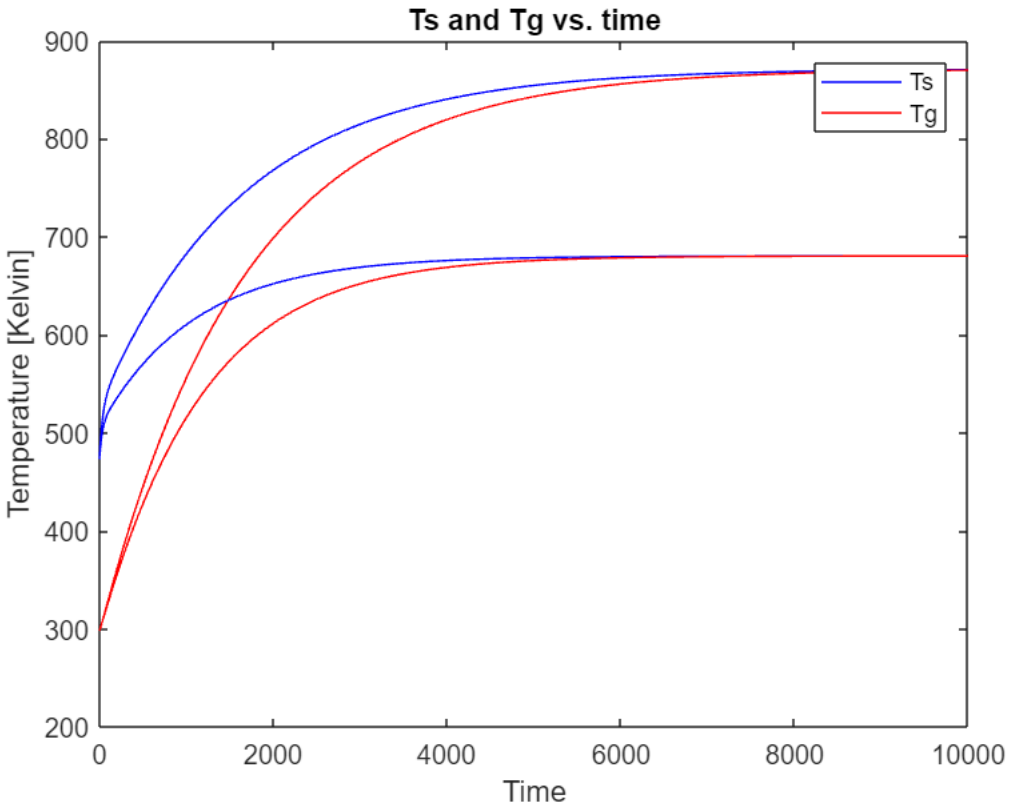


Figure4: The effect of changing k_r between $0 W/K^4$ 0 (top) and $7.8 \cdot 10^{-10}$ W/K^4 W/K^4(bottom)

In absence of radiation the temperature settles in a longer time since the oven cannot dissipate the heat in excess as fast as when radiation is accounted for. The increased stationary temperature is expected because if there is less heat loss there is more heating in the oven.

Linear Control Design II - Group Work Problem Module 5 Solution

Description

A system with three interconnected tanks is shown in Fig.1. The tanks have the cross sectional areas A_1, A_2 and A_3. The input flows are proportional to the input voltages, that is

$$q_a = ku_1$$
$$q_b = ku_2$$

The flows follow the square root law and since the pressures in the bottoms of the tanks are proportional to the level, thus one has the following outflow equations

$$q_1 = c_1\sqrt{x_1 - x_2}$$
$$q_2 = c_2\sqrt{x_3 - x_2}$$
$$q_0 = c_0\sqrt{x_2}$$

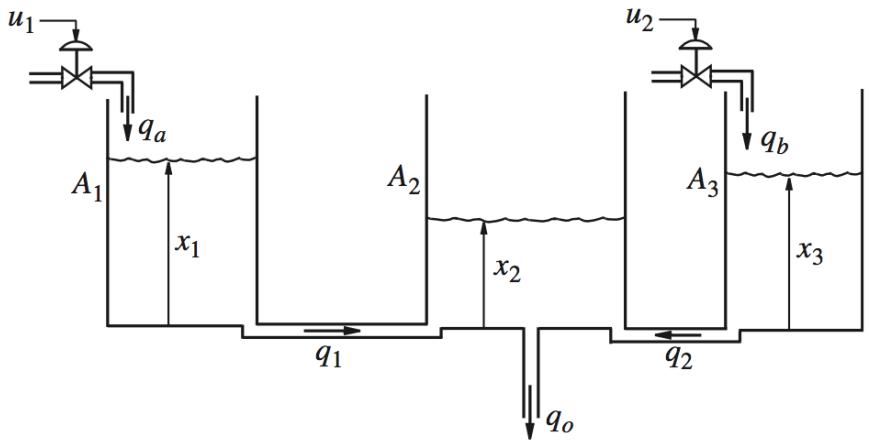


Figure 1: Schematic of the three tank system.

Problem 1 Derive the set of equations describing the system and formulate the nonlinear state model with the two inputs u_1 and u_2, and the output = q_0.

Solution:

Using the mass conversation law (which is the same as volume balance here, since density is assumed constant), three equations describing the volume change in each of the tanks are set up

$$A_1\dot{x}_1 = q_a - q_1$$
$$A_2\dot{x}_2 = q_1 + q_2 - q_0$$
$$A_3\dot{x}_3 = q_b - q_2$$

Inserting the flow equations into the conversation law equations yields the nonlinear state space model :

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \dot{\mathbf{x}} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{A_1}\left(-c_1\sqrt{x_1 - x_2} + ku_1\right) \\ \frac{1}{A_2}\left(c_1\sqrt{x_1 - x_2} + c_2\sqrt{x_3 - x_2} - c_0\sqrt{x_2}\right) \\ \frac{1}{A_3}\left(-c_2\sqrt{x_3 - x_2} + ku_2\right) \end{pmatrix}$$
$$\mathbf{g}(\mathbf{x},\mathbf{u}) = y = q_0 = c_0\sqrt{x_2}$$

Problem 2 Find the stationary point under the assumption that the inputs are $u_1 = u_{10}$ and $u_2 = u_{20}$.

Solution:

The stationary states are obtained by setting the time derivatives to zero, i.e.

$$\dot{x}_1 = 0 \Rightarrow ku_{10} = c_1\sqrt{x_{10} - x_{20}}$$
$$\dot{x}_2 = 0 \Rightarrow c_1\sqrt{x_{10} - x_{20}} + c_2\sqrt{x_{30} - x_{20}} = c_0\sqrt{x_{20}}$$
$$\dot{x}_3 = 0 \Rightarrow ku_{20} = c_2\sqrt{x_{30} - x_{20}}$$

Solving this system of 3 equations with 3 unknowns (x_{10}, x_{20}, x_{30}) can be done by first inserting the equations $(\dot{x}_1 = 0)$ and $(\dot{x}_3 = 0)$ into the equation $(\dot{x}_2 = 0)$, and solve for x_{20}, which yields:

$$ku_{10} + ku_{20} = c_0\sqrt{x_{20}} \Rightarrow x_{20} = \left(\frac{k}{c_0}(u_{10} + u_{20})\right)^2$$

Now that x_{20} is known, the equations $(\dot{x}_1 = 0)$ and $(\dot{x}_3 = 0)$ can be solved for x_{10} and x_{30}, respectively:

$$x_{10} = \left(\frac{k}{c_1}u_{10}\right)^2 + x_{20}$$
$$x_{30} = \left(\frac{k}{c_2}u_{20}\right)^2 + x_{20}$$

Finally, the stationary output is

$$y_0 = c_0\sqrt{x_{20}}$$

Solving this symbolically in MatLab can be done as follows:

```
% Declare the symbols
syms c0 c1 c2 A1 A2 A3 x1 x2 x3 k u1 u2
syms x10 x20 x30 u10 u20

% Define the nonlinear state-space model
disp('The nonlinear state-space model in symbolic form:');
```

```
The nonlinear state-space model in symbolic form:
```

```
f1 = (-c1*sqrt(x1-x2)+k*u1)/A1;
f2 = (c1*sqrt(x1-x2)+c2*sqrt(x3-x2)-c0*sqrt(x2))/A2;
f3 = (-c2*sqrt(x3-x2)+k*u2)/A3;
f = [f1; f2; f3]
```

```
f =
```

$$\begin{pmatrix} \frac{k\,u_1 - c_1\,\sqrt{x_1 - x_2}}{A_1} \\ \frac{c_1\,\sqrt{x_1 - x_2} - c_0\,\sqrt{x_2} + c_2\,\sqrt{x_3 - x_2}}{A_2} \\ \frac{k\,u_2 - c_2\,\sqrt{x_3 - x_2}}{A_3} \end{pmatrix}$$

```
x = [x1; x2; x3]; u = [u1; u2];
g = c0*sqrt(x2)
```

```
g =
```

$$c_0\,\sqrt{x_2}$$

```
% Solve the system of three equations with three unknowns:
disp('The stationary point:');
```

```
The stationary point:
```

```
solVar = solve([f1==0,f2==0,f3==0],[x1,x2,x3]);
```

```
Warning: Possibly spurious solutions.
```

```
disp('x10 = ')
```

```
x10 =
```

```
disp(subs(simplify(solVar.x1),[u1,u2],[u10,u20]))
```

$$\frac{k^2\,\left(c_0^{\,2}\,u_{10}^{\,2} + c_1^{\,2}\,u_{10}^{\,2} + 2\,c_1^{\,2}\,u_{10}\,u_{20} + c_1^{\,2}\,u_{20}^{\,2}\right)}{c_0^{\,2}\,c_1^{\,2}}$$

```
disp('x20 = ')
```

```
x20 =
```

```
disp(subs(simplify(solVar.x2),[u1,u2],[u10,u20]))
```

$$\frac{k^2\ (u_{10}+u_{20})^2}{{c_0}^2}$$

```
disp('x30 = ')
```

```
x30 =
```

```
disp(subs(simplify(solVar.x3),[u1,u2],[u10,u20]))
```

$$\frac{k^2\ \left({c_0}^2\ {u_{20}}^2+{c_2}^2\ {u_{10}}^2+2\ {c_2}^2\ u_{10}\ u_{20}+{c_2}^2\ {u_{20}}^2\right)}{{c_0}^2\ {c_2}^2}$$

```
disp('y0 = ')
```

```
y0 =
```

```
disp(subs(g,[x1,x2,x3,u1,u2],[x10,x20,x30,u10,u20]))
```

$$c_0\ \sqrt{x_{20}}$$

```

```

Problem 3 Linearize the model and derive the linear state space model

Process equation: $\Delta\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u}$

Output equation: $\Delta\mathbf{y} = \mathbf{C}\Delta\mathbf{x}$

Solution:

The incremental variables are chosen to be

$$x_1 = x_{10} + \Delta x_1 \qquad u_1 = u_{10} + \Delta u_1$$
$$x_2 = x_{20} + \Delta x_2 \qquad u_2 = u_{20} + \Delta u_2$$
$$x_3 = x_{30} + \Delta x_3 \qquad y = y_0 + \Delta y$$

The state space matrices can then be found as

$$\mathbf{A} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{0}} = \begin{pmatrix} \frac{-c_1}{2A_1\sqrt{x_{10}-x_{20}}} & \frac{c_1}{2A_1\sqrt{x_{10}-x_{20}}} & 0 \\ \frac{c_1}{2A_2\sqrt{x_{10}-x_{20}}} & \frac{-c_1}{2A_2\sqrt{x_{10}-x_{20}}} - \frac{-c_0}{2A_2\sqrt{x_{20}}} - \frac{c_2}{2A_2\sqrt{x_{30}-x_{20}}} & \frac{c_2}{2A_2\sqrt{x_{30}-x_{20}}} \\ 0 & \frac{c_2}{2A_3\sqrt{x_{30}-x_{20}}} & \frac{-c_2}{2A_3\sqrt{x_{30}-x_{20}}} \end{pmatrix}$$

$$\mathbf{B} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{0}} = \begin{pmatrix} \frac{k}{A_1} & 0 \\ 0 & 0 \\ 0 & \frac{k}{A_3} \end{pmatrix}, \qquad \mathbf{C} = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{x}}\right|_{\mathbf{0}} = \begin{pmatrix} 0 & \frac{c_0}{2\sqrt{x_{20}}} & 0 \end{pmatrix}, \qquad \mathbf{D} = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right|_{\mathbf{0}} = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

This can be performed symbolically in MatLab as follows:

```
% Calculace the jacobians for the A,B,C,D matrices
disp('The linearized state-space model in symbolic form:');
```

```
The linearized state-space model in symbolic form:
```

```
A = subs(jacobian(f,x), [x1 x2 x3 u1 u2], [x10 x20 x30 u10 u20])
```

```
A =
```

$$\begin{pmatrix} -\sigma_2 & \sigma_2 & 0 \\ \frac{c_1}{2\,A_2\,\sqrt{x_{10}-x_{20}}} & -\frac{\frac{c_0}{2\,\sqrt{x_{20}}} + \frac{c_1}{2\,\sqrt{x_{10}-x_{20}}} + \frac{c_2}{2\,\sqrt{x_{30}-x_{20}}}}{A_2} & \frac{c_2}{2\,A_2\,\sqrt{x_{30}-x_{20}}} \\ 0 & \sigma_1 & -\sigma_1 \end{pmatrix}$$

where

$$\sigma_1 = \frac{c_2}{2\,A_3\,\sqrt{x_{30}-x_{20}}}$$

$$\sigma_2 = \frac{c_1}{2\,A_1\,\sqrt{x_{10}-x_{20}}}$$

```
B = subs(jacobian(f,u), [x1 x2 x3 u1 u2], [x10 x20 x30 u10 u20])
```

```
B =
```

$$\begin{pmatrix} \frac{k}{A_1} & 0 \\ 0 & 0 \\ 0 & \frac{k}{A_3} \end{pmatrix}$$

```
C = subs(jacobian(g,x), [x1 x2 x3 u1 u2], [x10 x20 x30 u10 u20])
```

C =

$$\begin{pmatrix} 0 & \frac{c_0}{2\sqrt{x_{20}}} & 0 \end{pmatrix}$$

```
D = subs(jacobian(g,u), [x1 x2 x3 u1 u2], [x10 x20 x30 u10 u20])
```

D = $\begin{pmatrix} 0 & 0 \end{pmatrix}$

Table 1 shows the numerical value of the system's parameters. The working interval of the input voltages u_1 and u_2 is 0-10 volts.

Table 1: System Parameters

Parameter	Value
A_1	0.7 m^2
A_2	1.2 m^2
A_3	0.7 m^2
k	0.002 $m^3/(sec \cdot volt)$
c_1	0.008 $m^{5/2}/sec$
c_2	0.008 $m^{5/2}/sec$
c_0	0.02 $m^{5/2}/sec$

Lets go ahead and define the constant variables in MatLab:

```
% Define System parameters:
A1 = 0.7;
A2 = 1.2;
A3 = 0.7;

k = .002;

c1 = .008;
c2 = .008;
c0 = .02;
```

The next block of code simply evaluates the symbolic calculations performed earlier with the constants plugged in. This allows us to see the numerical point of operation and the numerical system Matrices. Theese can be compared with results obtained later in the assignment.

```
% The stationary point
disp('The stationary point:');
```

```
The stationary point:
```

```
u10 = 5
```

```
u10 = 5
```

```
u20 = 5
```

```
u20 = 5
```

```
x20 = ((k/c0)*(u10+u20))^2
```

```
x20 = 1
```

```
x10 = ((k/c1)*u10)^2+x20
```

```
x10 = 2.5625
```

```
x30 = ((k/c2)*u20)^2+x20
```

```
x30 = 2.5625
```

```

% Show the numerical system matrices
disp('The numerical system matrices:');
```

```
The numerical system matrices:
```

```
disp('A =');
```

```
A =
```

```
disp(eval(subs(A)))
```

```
   -0.0046    0.0046         0
    0.0027   -0.0137    0.0027
         0    0.0046   -0.0046
```

```
disp('B =');
```

```
B =
```

```
disp(eval(subs(B)))
```

```
    0.0029         0
         0         0
         0    0.0029
```

```
disp('C =');
```

```
C =
```

```
disp(eval(subs(C)))
```

```
         0    0.0100         0
```

```
disp('D =');
```

```
D =
```

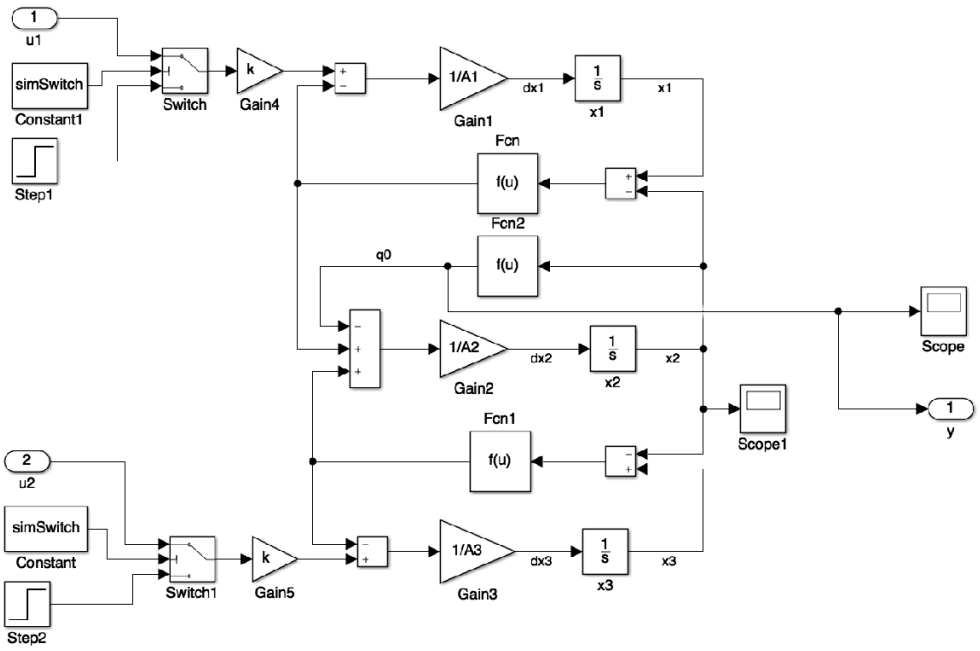
```
disp(eval(subs(D)))
```

```
     0     0
```

Problem 4 Set up a Simulink model of the nonlinear system and write a MatLab script to initialize the model with the needed parameter values. Remember that all input variables must be provided with an in port, and all output variables must be provided with an out port in the model.

Solution:

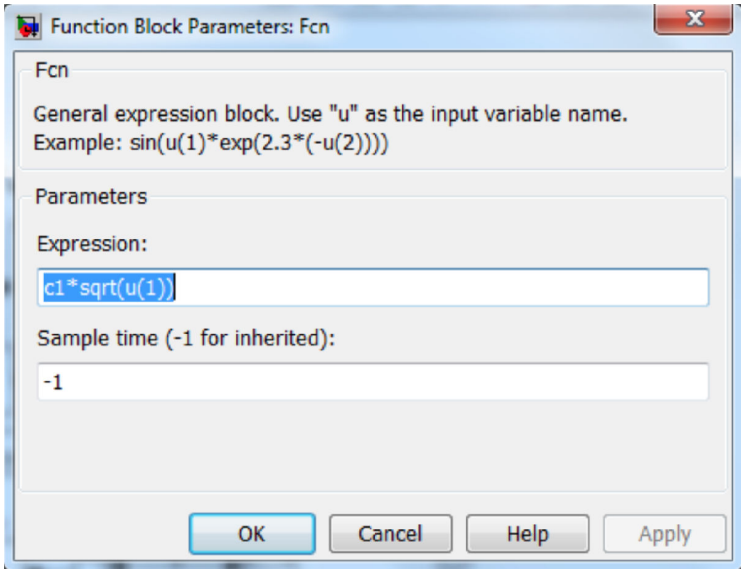
To simulate the nonlinear system a Simulink block diagram has been setup as shown in Figure below:



To open the simulink model run the following line of code:

```
% open('Module6_NonlinearModel.slx')
```

In the Figure below, it is shown how the block properties for the fcn block in Simulink is set to work as a square root function.



Problem 5 Linearize the model using linmod. Set both inputs $u_1 = u_2 = 5$ volts. Simulink by default determine the numbering of the states in the model when linmod is used. It can be a little confusing when comparing Simulinks results with those obtained by manual calculations. If desired, one can determine one's own numbering in the following way:

- Choose an integrator in the model and do a right-button click
- Choose Block Properties. A window opens
- Write the number desired in the Priority-field
- Close the Block Properties window

Solution:

To linearise the system, a simulation of 5000 seconds is run with a chosen linearisation point. The plot is then inspected to see if the system has reached steady state or more time is required.

```
%% System linearization
x0 = zeros(3,1); % initial state

simSwitch = -1; % switching variable for simuling model
u = 5; % input voltage
simopts = simset('FinalStateName','xFinal'); % set the sim function options
to obtain the final state values as xFinal
[t,x,y] = sim('Module5_NonlinearModel',5000,simopts); % simulate for 5000
seconds to optain steady state
xf = xFinal;
disp(' ');
```

```
disp(['xFinal :' num2str(xf)]);
```

```
xFinal :2.5625     0.99999      2.5625
```

```
disp(['yFinal :' num2str(y(end))]);
```

```
yFinal :0.02
```

```
simSwitch = 1;
disp(' ');
```

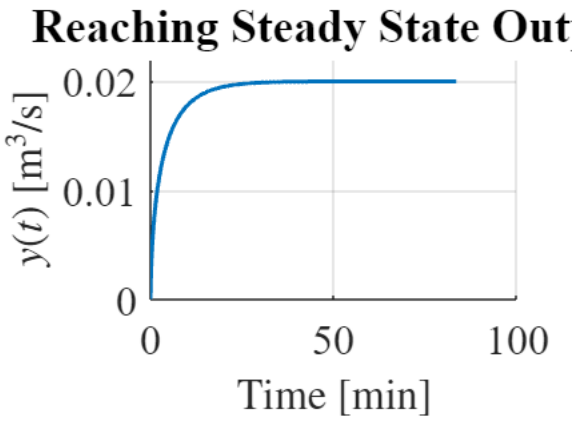
```
disp('The numerical system matrices (linmod):');
```

```
The numerical system matrices (linmod):
```

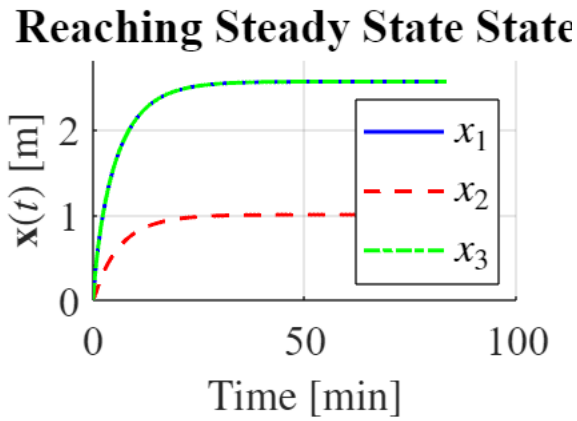
```
[Aa,Bb,Cc,Dd] = linmod('Module5_NonlinearModel',xf,[u u]') % obtain
linearized model using linmod applying steady state values for u = 5V.
```

```
Aa = 3×3
   -0.0046    0.0046         0
    0.0027   -0.0137    0.0027
         0    0.0046   -0.0046
Bb = 3×2
    0.0029         0
         0         0
         0    0.0029
Cc = 1×3
         0    0.0100         0
Dd = 1×2
     0     0
```

```
% Plot output
figure, h1 = axes; set(h1,'FontName','times','FontSize',16)
hold on, grid on
plot(t/60,y,'LineWidth',1.5) % time vector is plotted in minutes
title('Reaching Steady State Output')
ylim([0 max(y)*1.1])
xlabel('Time [min]','FontName','times','FontSize',16,'Interpreter','Latex')
ylabel('$y(t)$ [m$^3$/
s]','FontName','times','FontSize',16,'Interpreter','Latex')
```




```
% Plot states
figure, h2 = axes; set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(t/60,x(:,1),'b',t/60,x(:,2),'--r',t/60,x(:,3),'-.g','LineWidth',1.5)
title('Reaching Steady State States')
ylim([0 max(x(:))*1.1])
xlabel('Time [min]','FontName','times','FontSize',16,'Interpreter','Latex')
ylabel('$\mathbf{x}(t)$
[m]','FontName','times','FontSize',16,'Interpreter','Latex')
l = legend('$x_1$','$x_2$','$x_3$','Location','SouthEast');
set(l,'FontName','times','FontSize',16,'Interpreter','Latex')
```



The plot is then inspected to see if the system has reached steady state or more time is required. The plots shows the output and state time series. The plots show that 5000 seconds are sufficient to reach steady state. The stationary values of the states and the output are $x_1 = 2.56$ m, $x_2 = 1.0$ m, $x_3 = 2.56$ m, and $y = 0.02$ m^3/s respectively.

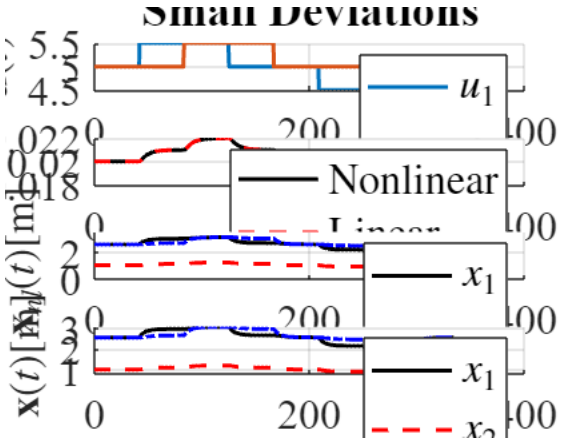
Problem 6 Simulate the nonlinear and linearized model for small and large variations of the system's inputs u_1 and u_2 around their stationary values. Plot the state and output responses and try to determine which is the range of input variations where the linearized model is a good approximation of the nonlinear system.

Solution

To test if the linearised system is performing as the nonlinear system around the steady state, the two systems are simulated by imposing small steps on the input around the linearisation point.

```
% ------------------------------------------------------------------------
% Simulating for small deviations from the stationary input value (+/- 0.5
Volts)
% ------------------------------------------------------------------------
sys = ss(Aa,Bb,Cc,Dd);
sys.u = {'u1','u2'};
sys.y = {'y'};
runl = 20000;
T = 0:0.1:runl;
% input signal
U1 = 0;
Us = U1*ones(length(T),2);
Us(25000:75000,1) = U1+.5;
Us(50000:100000,2) = U1+.5;
Us(125000:175000,1) = U1-.5;
Us(150000:175000,2) = U1-.5;
inp = zeros(length(T),3);

% linear model
[yl,tl,xl] = lsim(sys,Us,T);
yl = yl+0.02;
xl(:,1) = xl(:,1) + xf(1);
xl(:,2) = xl(:,2) + xf(2);
xl(:,3) = xl(:,3) + xf(3);
% nonlinear model
Us = Us+5;
inp(:,1) = T';
inp(:,2:3) = Us;
x0 = xf; %start at linearisation point
[t1,x1,y1] = sim('Module5_NonlinearModel',runl,[],inp);

% ------------------------------------------------------------------------
% Plotting the results
% ------------------------------------------------------------------------
figure, h3 = subplot(4,1,1); set(h3,'FontName','times','FontSize',16)
title('Small Deviations')
hold on, grid on
plot(t1/60,Us,'LineWidth',1.5)
ylabel('$\mathbf{u}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
l = legend('$u_1$','$u_2$','Location','northeast');
```

```
set(l,'FontName','times','FontSize',16,'interpreter','latex')
h4 = subplot(4,1,2); set(h4,'FontName','times','FontSize',16)
hold on, grid on
plot(t1/60,y1,'k',tl/60,yl,'--r','LineWidth',1.5)
ylabel('$y(t)$ [m$^3$/
s]','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('Nonlinear','Linear','Location','northeast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')

h5 = subplot(4,1,3); set(h5,'FontName','times','FontSize',16)
hold on, grid on
plot(t1/60,x1(:,1),'k',t1/60,x1(:,2),'--
r',t1/60,x1(:,3),'-.b','LineWidth',1.5)
ylabel('$\mathbf{x}_{nl}(t)$[m]
','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$x_1$','$x_2$','$x_3$','Location','northeast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')
ylim([0 3.5])
h6 = subplot(4,1,4); set(h6,'FontName','times','FontSize',16)
hold on, grid on
plot(tl/60,xl(:,1),'k',tl/60,xl(:,2),'--r',tl/
60,xl(:,3),'-.b','LineWidth',1.5)
xlabel('Time [min]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$\mathbf{x}(t)$[m]
','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$x_1$','$x_2$','$x_3$','Location','northeast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')
```

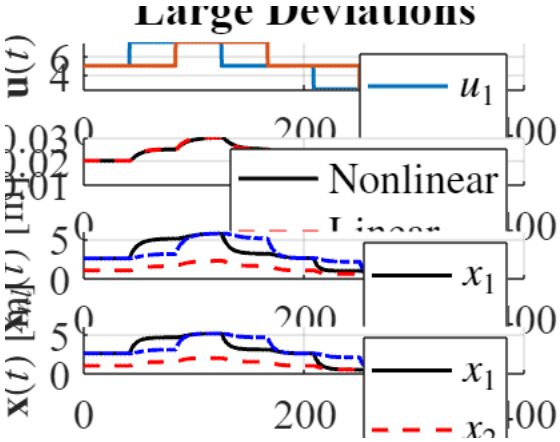


The Figure **"Small Deviations"** shows that as long as the deviations from the stationary input $u_0 = 5V$ are small the linear model suffices. When simulating the linearised system it is important to remember that you are now operating around a new zero so the difference has to be added to the system!

Below, the system is furthermore simulated for larger deviations to assess if the linear model will divert from the nonlinear one.

```
% ------------------------------------------------------------------------
% Simulating for large deviations from the stationary input value (+/- 2.5
Volts)
% ------------------------------------------------------------------------

% input signal
U1 = 0;
Ul = U1*ones(length(T),2);
Ul(25000:75000,1) = U1+2.5;
Ul(50000:100000,2) = U1+2.5;
Ul(125000:175000,1)=U1-2.5;
Ul(150000:175000,2)=U1-2.5;
inp = zeros(length(T),3);

% linear model
[ylb,tlb,xlb] = lsim(sys,Ul,T);
ylb = ylb+0.02;
xlb(:,1) = xlb(:,1)+xf(1);
xlb(:,2) = xlb(:,2)+xf(2);
xlb(:,3) = xlb(:,3)+xf(3);
% nonlinear model
Ul = Ul+5;
inp(:,1) = T';
inp(:,2:3) = Ul;
[t1b,x1b,y1b] = sim('Module5_NonlinearModel',runl,[],inp);

% ------------------------------------------------------------------------
% Plotting the results
% ------------------------------------------------------------------------
figure, h7 = subplot(4,1,1); set(h7,'FontName','times','FontSize',16)
title('Large Deviations')
hold on, grid on
plot(t1/60,Ul,'LineWidth',1.5)
ylabel('$\mathbf{u}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
l = legend('$u_1$','$u_2$','Location','northeast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')
h8 = subplot(4,1,2); set(h8,'FontName','times','FontSize',16)
hold on, grid on
plot(t1b/60,y1b,'k',tlb/60,ylb,'--r','LineWidth',1.5)
ylabel('$y(t)$ [m$^3$/
s]','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('Nonlinear','Linear','Location','northeast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')

h9 = subplot(4,1,3); set(h9,'FontName','times','FontSize',16)
hold on, grid on
plot(t1b/60,x1b(:,1),'k',t1b/60,x1b(:,2),'--r',t1b/
60,x1b(:,3),'-.b','LineWidth',1.5)
```

```
ylabel('$\mathbf{x}_{nl}(t)$
[m]','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$x_1$','$x_2$','$x_3$','Location','northeast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')
ylim([0 6])
h10 = subplot(4,1,4); set(h10,'FontName','times','FontSize',16)
hold on, grid on
plot(tlb/60,xlb(:,1),'k',tlb/60,xlb(:,2),'--r',tlb/
60,xlb(:,3),'-.b','LineWidth',1.5)
xlabel('Time [min]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$\mathbf{x}(t)$
[m]','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$x_1$','$x_2$','$x_3$','Location','northeast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')
ylim([0 6])
```



The Figure **"Large Deviations"** shows that for larger steps the linearised model is no longer valid in describing the dynamics of the original system.

Problem 7 Determine the transfer functions from u_1 to y and from u_2 to y.

Solution

The two (identical) transfer functions are found using the Matlab command tf() on the state-space system declared earlier.

```
%% Transfer Functions
G = tf(sys)

% Lets check the Poles and Zeros of the transfer functions.
% From u1 -> y:
p_yu = pole(G(1,1))
z_yu = zero(G(1,1))
```

```
% No need to check u2 -> y as the two tf's are identical.
```

Note that a zero-pole cancellation takes place reducing the transfer function (use minreal to obtain the zero-pole cancellation in Matlab)

```
G = minreal(tf(sys)) % Minimal realisation
```

Problem 8 Choose a suitable sampling time T_s and discretize the system obtained in Problem 5 using the Matlab `c2d`-function. Simulate the nonlinear model, the linear continuous time model and the linear discrete time model for small step changes of the system inputs, and assess if the chosen sampling time is suitable to capture the system dynamics and how the discretization affects the state and outpute responses of the system.

Solution:

Choosing a sampling time of $T_s = 13s$ makes it so that the sampling time is $\approx$ 10 times smaller that the fastest time constant. We obtain the discretized system:

$$\mathbf{x}(k+1) = \mathbf{F}\mathbf{x}(k) + \mathbf{G}\mathbf{u}(k)$$
$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$$

```
% Parameters
m1 = 0.712;         % [kg] mass of platform 1
m2 = 0.428;         % [kg] mass of platform 1
m3 = 0.428;         % [kg] mass of platform 1

l = 24e-3; % mm, small
r = 14e-3 / 2;
A = pi*r^2;
R = 148.2;
L = 800e-3; % [H]
iMax = 0.14; % measured at 24v

gam = 0.5;          % bias ratio - normally chosen between 0.2-0.5
ib = iMax*gam;      % Bias current
mu = 4*pi*1e-7;     % Permeability of free space (vacuum)
s0 = 5e-3;          % displacement operation point

b = 0.025;          % [m] beam width
h = 1.0e-3;         % [m] beam thickness
N = sqrt(L*2*s0 / (mu*A)); % estimate number of windings based on above
formular
E = 2.0e11;         % [N/m2] elasticity modulus of steel
Iz = (b*h^3)/12;    % [m4] area moment of inertia

L1= 0.145;           % [m] length of beam 1
L2= 0.134;           % [m] length of beam 2
L3= 0.229;           % [m] length of beam 3

% Force constants obtained from the linearisation

k1 = 2*12*E*Iz/L1^3;   % [N/m] stiffness of beam 1
k2 = 2*12*E*Iz/L2^3;   % [N/m] stiffness of beam 2
k3 = 2*12*E*Iz/L3^3;   % [N/m] stiffness of beam 3

Ki = mu*N^2*A*ib / s0^2;
Ks = -mu*N^2*A*(ib^2)/s0^3;
```

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u, \quad \mathbf{x} \in \mathbb{R}^7,\ u \in \mathbb{R}$$
$$y = \mathbf{C}\mathbf{x} + Du \quad y \in \mathbb{R}$$

$$x = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \end{bmatrix}^T$$
$$= \begin{bmatrix} i & q_1 & \dot{q}_1 & q_2 & \dot{q}_2 & q_3 & \dot{q}_3 \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} -\frac{R}{L} & 0 & -\frac{K_i}{L} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{K_i}{m_1} & -\frac{k_1+k_2+K_s}{m_1} & 0 & \frac{k_2}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{k_2}{m_2} & 0 & -\frac{k_2+k_3}{m_2} & 0 & \frac{k_3}{m_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{k_3}{m_3} & 0 & -\frac{k_3}{m_3} & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \frac{1}{L} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{C}^T = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ D = 0$$

$$\mathbf{z} = \mathbf{P}\mathbf{x} \Leftrightarrow \mathbf{x} = \mathbf{P}^{-1}\mathbf{z}$$

$$\dot{\mathbf{z}}(t) = \mathbf{P}\mathbf{A}\mathbf{P}^{-1}\mathbf{z}(t) + \mathbf{P}\mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{P}^{-1}\mathbf{z}(t) + \mathbf{D}\mathbf{u}(t)$$

$$\dot{\mathbf{z}}(t) = \mathbf{\Lambda}\mathbf{z}(t) + \mathbf{B}_t\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}_t\mathbf{z}(t) + \mathbf{D}_t\mathbf{u}(t)$$

```
% dx = Ax + Bu
```

```
% y = Cx + Du
% x = transpose([x1, x2, x3, x4, x5, x6, x7])
% x = transpose([i, q1, dq1, q2, dq2, q3, dq3])
% A = [[-R/L, 0, -Ki/L, 0, 0, 0, 0];[0, 0, 1, 0, 0, 0, 0];[Ki/m1,-(k1+k2-
Ks)/m1, 0, k2/m1, 0, 0, 0];[0, 0, 0, 0, 1, 0, 0];[0, k2/m2, 0, -(k2+k3)/m2,
0, k3/m2, 0];[0, 0, 0, 0, 0, 0, 1];[0, 0, 0, k3/m3, 0, -k3/m3, 0]]
A = [[-0.0185, 0, -0.0014, 0, 0, 0, 0];[0, 0, 0.0001, 0, 0, 0, 0];
[0.0016,-1.0334, 0, 0.5837, 0, 0, 0];[0, 0, 0, 0, 0.0001, 0, 0];[0, 0.9711,
0, -1.1656, 0, 0.1946, 0];[0, 0, 0, 0, 0, 0, 0.0001];[0, 0, 0, 0.1946, 0,
-0.1946, 0]]
```

```
A = 7×7
   -0.0185         0   -0.0014         0         0         0         0
         0         0    0.0001         0         0         0         0
    0.0016   -1.0334         0    0.5837         0         0         0
         0         0         0         0    0.0001         0         0
         0    0.9711         0   -1.1656         0    0.1946         0
         0         0         0         0         0         0    0.0001
         0         0         0    0.1946         0   -0.1946         0
```

```
A = A*10000
```

```
A = 7×7
10^4 ×
   -0.0185         0   -0.0014         0         0         0         0
         0         0    0.0001         0         0         0         0
    0.0016   -1.0334         0    0.5837         0         0         0
         0         0         0         0    0.0001         0         0
         0    0.9711         0   -1.1656         0    0.1946         0
         0         0         0         0         0         0    0.0001
         0         0         0    0.1946         0   -0.1946         0
```

```
B = [1.25,0,0,0,0,0,0]
```

```
B = 1×7
    1.2500         0         0         0         0         0         0
```

```
C = [0,1,0,0,0,0,0]
```

```
C = 1×7
     0     1     0     0     0     0     0
```

1) Calculate **Λ**.
2) Calculate **M**.
3) Calculate **P**, i.e. the inverse of **M**.
4) Calculated **C**t.
5) Calculate the time constant of the electro-mechanical system.
6) Calculate the damped natural frequencies of the electro-mechanical system
7) Calculate the damping ratios associated to the damped natural frequencies of the system.
8) Explain the physical meaning of the natural modes of the electro-mechanical system and their to eigenvectors and eigenvalues (modal superposition).

```
[V,D] = eig(A);
% 1)
Lambda = sort(diag(D))
```

```
Lambda = 7×1 complex
10^2 ×
  -0.0006 - 0.3305i
  -0.0006 + 0.3305i
  -0.0025 - 0.6464i
  -0.0025 + 0.6464i
  -0.0018 - 1.3679i
  -0.0018 + 1.3679i
  -1.8404 + 0.0000i
```

```
% 2)
M = V
```

```
M = 7×7 complex
   0.9976 + 0.0000i   0.0282 - 0.0205i   0.0282 + 0.0205i   0.0390 - 0.0139i ···
   0.0004 + 0.0000i  -0.0000 + 0.0042i  -0.0000 - 0.0042i   0.0001 + 0.0090i
  -0.0684 + 0.0000i  -0.5723 - 0.0041i  -0.5723 + 0.0041i  -0.5795 + 0.0036i
   0.0001 + 0.0000i  -0.0000 - 0.0059i  -0.0000 + 0.0059i   0.0000 + 0.0095i
  -0.0146 + 0.0000i   0.8138 + 0.0000i   0.8138 + 0.0000i  -0.6135 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0007i   0.0000 - 0.0007i  -0.0002 - 0.0083i
  -0.0008 + 0.0000i  -0.0945 + 0.0003i  -0.0945 - 0.0003i   0.5346 - 0.0077i
```

```
% 3)
P = inv(M)
```

```
P = 7×7 complex
   1.0057 - 0.0000i   2.9781 + 0.0000i   0.0603 - 0.0000i  -1.4267 + 0.0000i ···
  -0.0219 + 0.0162i  -0.1759 -53.3929i  -0.3920 - 0.0005i   0.2145 +45.8226i
  -0.0219 - 0.0162i  -0.1759 +53.3929i  -0.3920 + 0.0005i   0.2145 -45.8226i
  -0.0306 + 0.0104i  -0.0961 -25.4389i  -0.3958 - 0.0036i   0.3117 -16.2786i
  -0.0306 - 0.0104i  -0.0961 +25.4389i  -0.3958 + 0.0036i   0.3117 +16.2786i
   0.0165 - 0.0030i   0.2439 + 6.4627i   0.1968 - 0.0007i  -0.0111 + 6.1952i
   0.0165 + 0.0030i   0.2439 - 6.4627i   0.1968 + 0.0007i  -0.0111 - 6.1952i
```

```
% 4)
Ct = C*M
```

```
Ct = 1×7 complex
   0.0004 + 0.0000i  -0.0000 + 0.0042i  -0.0000 - 0.0042i   0.0001 + 0.0090i ···
```

```
Bt = B*M
```

```
Bt = 1×7 complex
   1.2469 + 0.0000i   0.0352 - 0.0257i   0.0352 + 0.0257i   0.0488 - 0.0174i ···
```

```
% 5)
tau = 1 / abs(Lambda) % time constant
```

```
tau = 1×7
         0         0         0         0         0         0    0.0054
```

```
% 6,7)
alpha = real(Lambda);
beta = imag(Lambda);
for s=1:7
```

```
    omega_n(s) = sqrt(alpha(s)^2+beta(s)^2); % dampening freq
    epsilon(s) = -alpha(s)/sqrt(alpha(s)^2+beta(s)^2); % damping ratio
end
omega_n
```

```
omega_n = 1×7
   33.0481   33.0481   64.6452   64.6452  136.7915  136.7915  184.0403
```

```
epsilon
```

```
epsilon = 1×7
    0.0017    0.0017    0.0038    0.0038    0.0013    0.0013    1.0000
```

Linear Control Design II - Group Work Problem Module 8 Solution

Description

Figure 1 shows an aircraft in lateral motion where the movements are yaw (rotation around the vertical axis) and roll (rotation around the longitudinal axis). The lateral motion of an aircraft takes primarily place in the horizontal plane.

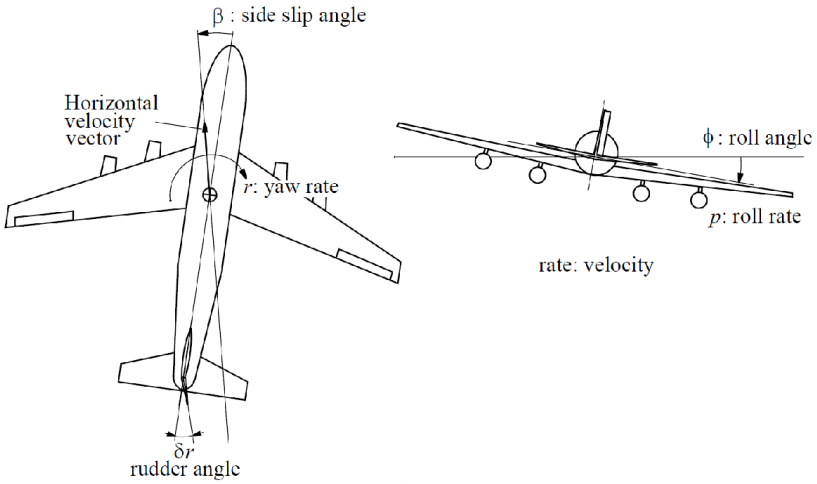


Figure 1: Lateral dynamics of a Boeing 747:

A linearized state space model of the lateral dynamics of a Boeing 747 in altitude 40000 ft and speed 774 ft/sec (850 km/h) is shown below:

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \dot{p} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} -0.0558 & -0.9968 & 0.0802 & 0.0415 \\ 0.598 & -0.115 & -0.0318 & 0 \\ -3.05 & 0.388 & -0.4650 & 0 \\ 0 & 0.0805 & 1 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ p \\ \phi \end{bmatrix} + \begin{bmatrix} 0.00729 \\ -0.475 \\ 0.153 \\ 0 \end{bmatrix} \delta_r$$

$$y = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ p \\ \phi \end{bmatrix}$$

where β is the side-slip angle (rad), r is the yaw rate (rad/s), p is the roll rate (rad/s), φ is the roll angle (rad), and δ_r is the rudder deflection (rad).

```
% Define system matrices:
A = [-0.0558 -0.9968  0.0802  0.0415;
      0.598  -0.115  -0.0318  0;
     -3.050   0.388  -0.4650  0;
      0       0.0805  1       0]
```

```
A =
   -0.0558   -0.9968    0.0802    0.0415
    0.5980   -0.1150   -0.0318         0
   -3.0500    0.3880   -0.4650         0
         0    0.0805    1.0000         0
```

```
B = [0.00729; -0.475; 0.153; 0]
```

```
B =
    0.0073
   -0.4750
    0.1530
         0
```

```
C = [0 1 0 0]
```

```
C =
     0     1     0     0
```

```
D = 0;
```

This assignment guides you through the analysis of the aircraft's lateral dynamics in terms of natural modes and system responses to non-zero initial conditions and external inputs.

Problem 1 Calculate the eigenvalues of the system dynamical matrix A, and based on those identify, if relevant, the time constants, natural frequencies and damping ratios.

Solution:

The eigenvalues are found by solving the characteristic equation:

$$\mathbf{P}_{ch,\mathbf{A}} = \det(\lambda \mathbf{I} - \mathbf{A}) = 0$$

which yields the following eigenvalues:

$$\lambda_1 = -0.0073\,, \lambda_2 = -0.5627\,, \lambda_{3,4} = -0.0329 \pm 0.9467\,i$$

```
% In Matlab, the eigenvalues can be easily found using as:
evals = eig(A)
```

```
evals =
  -0.0329 + 0.9467i
  -0.0329 - 0.9467i
  -0.5627 + 0.0000i
  -0.0073 + 0.0000i
```

Note that all the eigenvalues have negative real part, so they lie in the left half complex plane. This ensures the asymptotic stability of the system.

The time constants for the real eigenvalues can be found using the formula, $t_i = -1/\lambda_i$, (see page 74 of the textbook) which in this case gives the time constants: $t_1 = 137.4010$ and $t_2 = 1.7773$.

For the complex conjugated pair of eigenvalues the system will have a natural frequency and a damping ratio given by $\omega_n = \sqrt{\alpha^2 + \beta^2}$ and $\zeta = -\alpha / \sqrt{\alpha^2 + \beta^2}$ (see page 75 of the textbook). In this case the natural frequency and damping ratios are: $\omega_n = 0.9472$ and $\zeta = 0.0348$.

```
% In Matlab, the time constants, natural frequencies and damping ratios are
% found using the damp-function:
damp(A)
```

```
          Pole              Damping       Frequency       Time Constant
                                       (rad/TimeUnit)     (TimeUnit)

  -3.29e-02 + 9.47e-01i     3.48e-02       9.47e-01          3.04e+01
  -3.29e-02 - 9.47e-01i     3.48e-02       9.47e-01          3.04e+01
  -5.63e-01                 1.00e+00       5.63e-01          1.78e+00
  -7.28e-03                 1.00e+00       7.28e-03          1.37e+02
```

```
% --- or by manual calculation:
% time constants s74-75
tau_i = [-1/evals(3) -1/evals(4)]'
```

```
tau_i =
    1.7773
  137.4010
```

```
% natural frequencies
omega_n = abs(evals(1))
```

```
omega_n = 0.9472
```

```
% damping ratios
zeta = -real(evals(1))/omega_n(1)
```

```
zeta = 0.0348
```

Problem 2 Based on the calculated eigenvalues determine the natural modes (also known as eigenmodes) of the system.

Solution:

The natural modes of the system can be written from the eigenvalues as (see page 69 of the textbook):

$e^{-0.0073\,t}$, $e^{-0.5627\,t}$, and $e^{(-0.0329 \pm 0.947\,i)t}$ or using Euler decomposition $e^{-0.0329\,t}(\cos(0.947\,t) \pm i \sin(0.947\,t))$

```
% Plot the temporal evolution of the eigenmodes
t_t1 = 0:1:1000;
t_t2 = 0:0.01:10;
t_wn = 0:0.1:150;

E1 = exp(evals(4)*t_t1);
E2 = exp(evals(3)*t_t2);
```

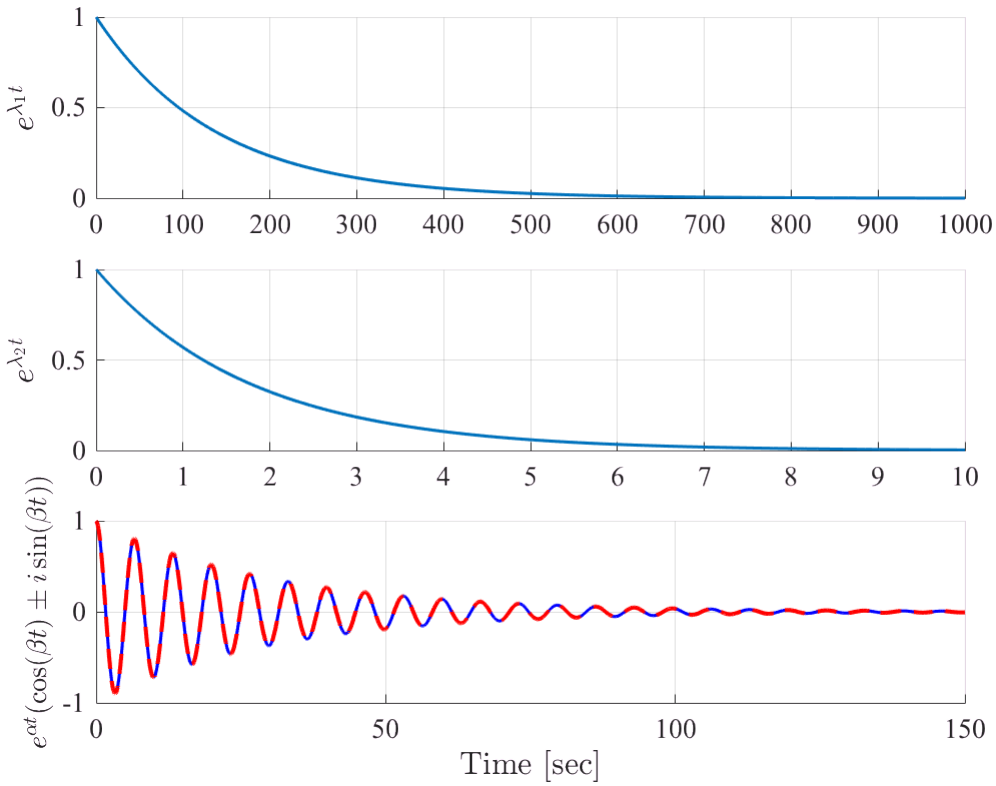
```
%E3 = exp(real(evals(1))*t_wn).*(cos(imag(evals(1))*t_wn) +
1i*sin(imag(evals(1))*t_wn));
%E4 = exp(real(evals(1))*t_wn).*(cos(imag(evals(1))*t_wn) -
1i*sin(imag(evals(1))*t_wn));
E3 = exp(evals(1)*t_wn)
```

```
E3 =
  Columns 1 through 526
   1.0000 + 0.0000i   0.9922 + 0.0942i   0.9757 + 0.1870i   0.9505 + 0.2774i   0.9170 + 0.3648i   0.8755 +
  Columns 527 through 1052
   0.1576 - 0.0803i   0.1639 - 0.0649i   0.1687 - 0.0489i   0.1720 - 0.0326i   0.1738 - 0.0162i   0.1740 +
  Columns 1053 through 1501
   0.0184 - 0.0253i   0.0206 - 0.0234i   0.0227 - 0.0213i   0.0245 - 0.0190i   0.0261 - 0.0165i   0.0274 -
```

```
E4 = exp(evals(2)*t_wn)
```

```
E4 =
  Columns 1 through 526
   1.0000 + 0.0000i   0.9922 - 0.0942i   0.9757 - 0.1870i   0.9505 - 0.2774i   0.9170 - 0.3648i   0.8755 -
  Columns 527 through 1052
   0.1576 + 0.0803i   0.1639 + 0.0649i   0.1687 + 0.0489i   0.1720 + 0.0326i   0.1738 + 0.0162i   0.1740 -
  Columns 1053 through 1501
   0.0184 + 0.0253i   0.0206 + 0.0234i   0.0227 + 0.0213i   0.0245 + 0.0190i   0.0261 + 0.0165i   0.0274 +
```

```
figure, h1 = subplot(3,1,1); set(h1,'FontSize',14,'FontName','times');
hold on, grid on
plot(t_t1,E1,'LineWidth',1.5)
ylabel('$e^{\lambda_1 t}
$','FontSize',16,'FontName','times','Interpreter','latex')
h2 = subplot(3,1,2); set(h2,'FontSize',14,'FontName','times');
hold on, grid on
plot(t_t2,E2,'LineWidth',1.5)
ylabel('$e^{\lambda_2 t}
$','FontSize',16,'FontName','times','Interpreter','latex')
h3 = subplot(3,1,3); set(h3,'FontSize',14,'FontName','times');
hold on, grid on
plot(t_wn,real(E3),'b',t_wn,real(E4),'--r','LineWidth',1.5)
ylabel('$e^{\alpha t}(\cos(\beta t) \pm i \sin(\beta t))
$','FontSize',14,'FontName','times','Interpreter','latex')
xlabel('Time [sec]','FontSize',16,'FontName','times','Interpreter','latex')
```



The eigenstructure of the lateral dynamics is usually represented by two real eigenvalues and one pair of complex conjugate eigenvalues. The three lateral modes are referred to as ***roll subsidence***, ***spiral*** and ***dutch roll***.

- The **roll subsidence mode** is non-oscillatory and is usually decoupled from the spiral mode and from the dutch roll modes. In rolling motion the wing experiences a component of velocity normal to the wing resulting in a small increase in the angle of attack on the down- going wing and a small decrease on the up-going wing thus, generating a lift difference over the wing span. Moreover, the lift difference gives rise to a restoring roll moment represented by the roll subsidence mode.
- The **spiral mode** is usually slow non-oscillatory and involves complex coupled motion in roll, yaw and side-slip angle. The spiral mode is usually excited by a disturbance in side-slip which is typically followed by a disturbance in roll. The side-slip angle generates a force on the tail fin, which in turn generates a yawing moment in the direction of the side-slip angle. The yawing motion generates increased differential lift across the wing span inducing increased roll moment and thereby, aggravates the situation.
- The **dutch roll mode** is a damped oscillation and the motion is a complex interaction between the three lateral degrees of freedom. The dutch roll mode makes the wing tip move in an elliptic pattern seen from the side parallel to the wing.

Problem 3 Based on the found eigenvalues and the aforementioned modes, associate the eigenvalues to the corresponding mode of the lateral dynamics.

Solution:

From the description three lateral modes have been described two non oscillating; **roll subsidence mode** and **spiral mode**, and one with damped oscillation **Dutch roll mode.** This directly leads to the conclusion that the Dutch roll mode is associated with the complex conjugated pair of eigenvalues.

Dutch roll mode: $e^{-0.0329\,t}(\cos(0.947\,t) \pm i\sin(0.947\,t))$

The Spiral mode is described as the slow non-oscillatory thus the eigenmode with the larger of the two time constants (smaller eigenvalue) is associated with the spiral mode.

Spiral mode: $e^{-0.0073\,t}$

The roll subsidence mode is described as non-oscillatory and thus the eigenmode with the smaller of the two time constants (larger eigenvalue) is associated with the roll subsidence mode.

Roll subsidence mode: $e^{-0.5627\,t}$

Problem 4 Compute the modal matrix $\mathbf{M}_{lat}$ of the system and discuss which natural modes contributes most and least to the dynamics of the state variables β, r, p, φ (Make sure that your modal matrix has real elements)

Solution:

The modal matrix is the matrix consisting of the eigenvectors of the system matrix A. These are found by solving the characteristic equation:

$$\mathbf{P}_{ch,\mathbf{A}} = \det(\lambda\mathbf{I} - \mathbf{A}) = 0$$

And then substituting one at a time the $\lambda_i$into the equation:

$$\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i \Rightarrow \mathbf{A}\mathbf{v}_i - \lambda_i\mathbf{v}_i = 0 \Rightarrow (\lambda_i\mathbf{I} - \mathbf{A})\mathbf{v}_i = 0$$

Matlab can do this calculation with:

```
% M is the Modal matrix
% E is a diagonal matrix with the corresponding eigenvalues.
[M,E] = eig(A)
```

```
M =
   0.1994 - 0.1063i   0.1994 + 0.1063i  -0.0172 + 0.0000i   0.0067 + 0.0000i
  -0.0780 - 0.1333i  -0.0780 + 0.1333i  -0.0118 + 0.0000i   0.0404 + 0.0000i
  -0.0165 + 0.6668i  -0.0165 - 0.6668i  -0.4895 + 0.0000i  -0.0105 + 0.0000i
   0.6930 + 0.0000i   0.6930 + 0.0000i   0.8717 + 0.0000i   0.9991 + 0.0000i
E =
  -0.0329 + 0.9467i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.0329 - 0.9467i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.5627 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0073 + 0.0000i
```

Two of the eigenvectors are complex and conjugates. To get the mode shapes, it is known from for example math 2 (Ole Christensen, Differential ligninger og uendelige raekker, chapter 2) that to obtain the real solutions for the system, the cosine with the real part of the complex eigenvector and the sine part with the imaginary

part of the eigenvector is a solution and is independent from all the other solutions consisting of the other eigenvectors.

This means that the following matrix $\mathbf{M}_{real}$ represents the real solutions and real mode shapes of the system.

```
Mreal = [real(M(:,1)) imag(M(:,1)) M(:, 3:end)]
```

```
Mreal =
    0.1994   -0.1063   -0.0172    0.0067
   -0.0780   -0.1333   -0.0118    0.0404
   -0.0165    0.6668   -0.4895   -0.0105
    0.6930         0    0.8717    0.9991
```

Looking at the columns of this matrix describes the 4 different mode shapes of the system. The first 2 is the oscillatory dutch roll motion, it is seen that the approximately evenly distributed elements in all 4 directions, means all the variables are excited during this motion. The third eigenvector corresponds to the roll subsidence mode, which affects the roll rate and the bank angle. Note different signs, meaning that the roll rate and bank angle is opposite, i.e., with bank to the right (positive φ), the roll rate is negative (to the left). The last mode is the spiral divergence mode, which almost only affects the bank angle.

Problem 5 Determine the transfer function from rudder deflection δ_r to yaw rate r, and calculate the Bode's diagram. Which of the aforementioned modes are visible in the magnitude response of the Bode's diagram?

Solution:

In the states-space description of equation, the input is δ_r and the output matrix $\mathbf{C}$ already selects the yaw rate $r$as the output. The transfer function can then be calculated as (see eq. (3.67) page 73):

$$\mathbf{G}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$$

or using the matlab function ss2tf()-function. The transfer function is found as follows:

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

```
G =

     -0.475 s^3 - 0.2479 s^2 - 0.1187 s - 0.05633
  ------------------------------------------------
  s^4 + 0.6358 s^3 + 0.9389 s^2 + 0.5116 s + 0.003674

Continuous-time transfer function.
```

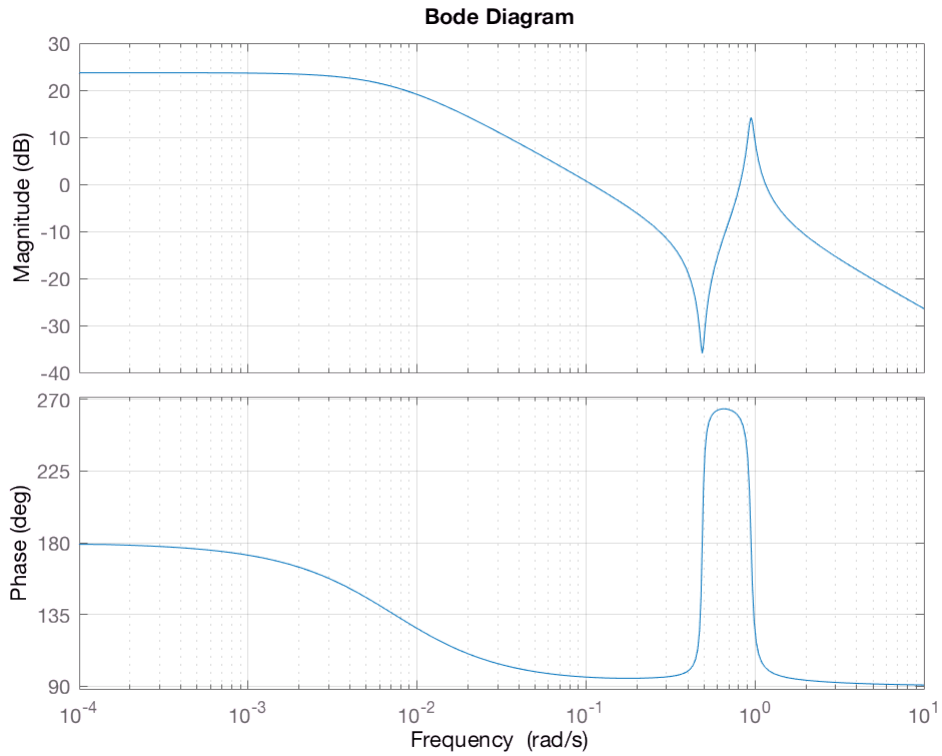
```
% Another way to compute is by first defining the state-space system in
Matlab
sys_ss = ss(A,B,C,D);
% and convert to a transfer function:
G = tf(sys_ss)
```

```
G =

     -0.475 s^3 - 0.2479 s^2 - 0.1187 s - 0.05633
```

```
   ---------------------------------------------------
   s^4 + 0.6358 s^3 + 0.9389 s^2 + 0.5116 s + 0.003674

 Continuous-time transfer function.
```

The Bode Diagram is calculated as follows in MatLab:

```
figure
bode(G)
grid on
```




```
% You can also call bode on the state-space system, i.e.
% bode(sys_ss)
```

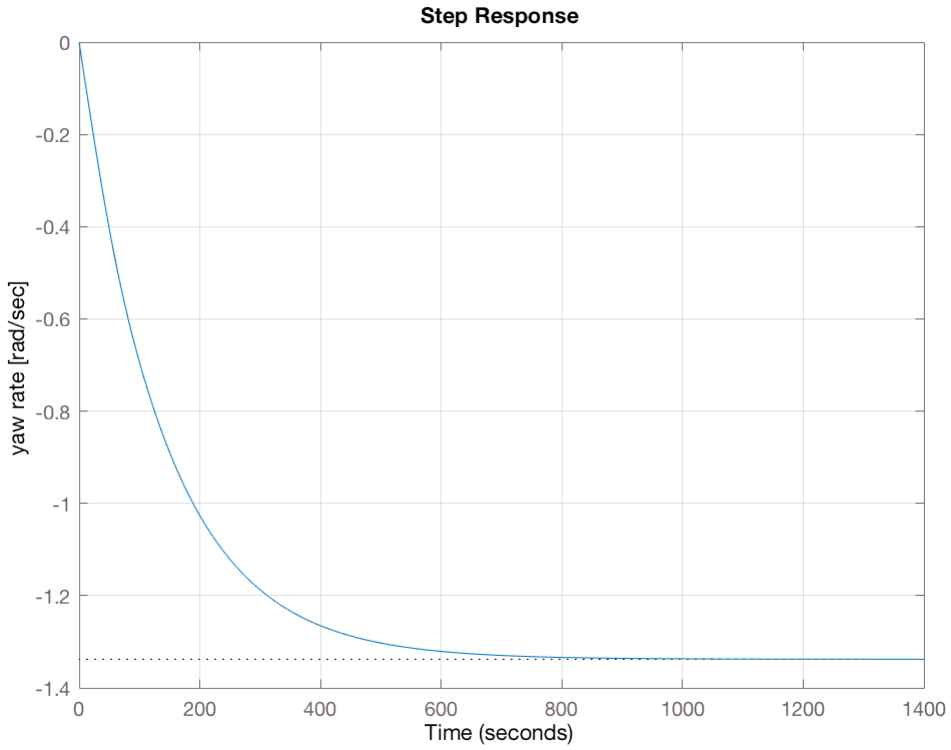
From the Bode plot it is clearly seen that there is a pole around 10^{-2} rad/s and there is a resonance peak (complex pole pair) at around1rad/s. Furthermore there is a anti-resonance (complex zero pair) around 0.5 rad/s. What is not clear from the plot is that there are another pole and zero around 0.5 rad/s, and since they are so close, they almost cancel each other.

Problem 6 Calculate the zero-state response for a step of 5 degrees in the rudder input. Analyse the response in connection with the modal analysis.

Solution:

Now that the transfer function is found it is possible to simulate a step on the rudder angle δ_r by using the step()-function on the transfer function. Making a step on the transfer function is the same as calculating the zero state response as the transfer function has the initial condition of the states as 0.

```
step_magnitude = 5*pi/180;
step(G * step_magnitude), grid
ylabel('yaw rate [rad/sec]');
```



The reason for the yaw rate to become negative is due to the sign of the corresponding constant (-0.475) in the B-matrix. Due to the size of this constant even small inputs result in large effects on the yaw rate. The behaviour of this input is dominated by the natural mode associated with the small real eigenvalue, as the plot shows only a decay with a high time constant.

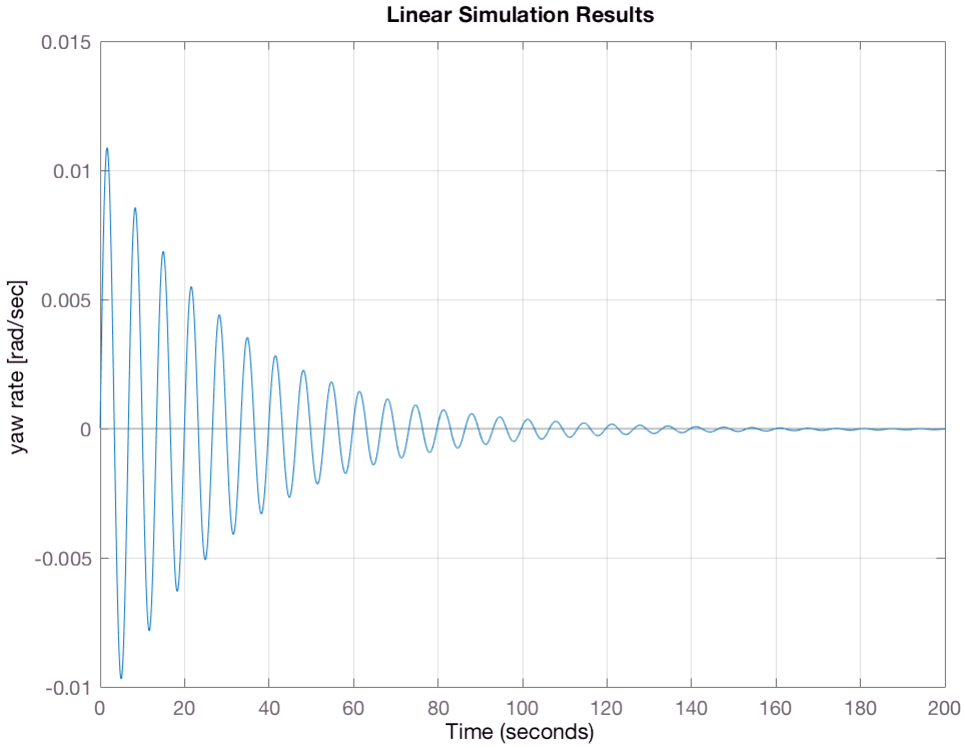
Problem 7 Calculate the zero-input response for a side-slip initial angle $\beta_0 = 1$ degree. Analyse the response in connection with the modal analysis.

Solution:

In this part the plane is oriented a little off the right course and is supposed to get back on track. As the zero input response is the evolution of the system to an initial condition of the states (see page 62 of the textbook)

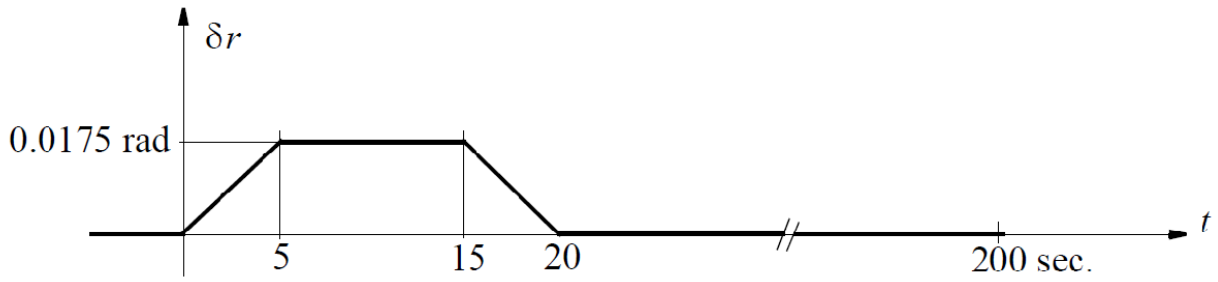
Using the Matlab lsim()-function it is possible to simulate the system with an initial value of the side slip of $\beta_0 = 1\text{deg} = \frac{\pi}{180}\text{rad}$. Remember to set the u-input in lsim() to zeroes.

```
%% Q7 Reponse of a side slip
figure
t = 0:.1:200;
u = zeros(length(t),1);
beta0 = 1*pi/180;
x0 = [beta0 0 0 0]';
lsim(A,B,C,D,u,t,x0),grid
ylabel('yaw rate [rad/sec]');
```



It takes long time for the system to stabilise after the disturbance of the initial side slip. From the oscillatory behaviour it can be seen that the of the side slip predominantly behaves according to the complex natural mode.

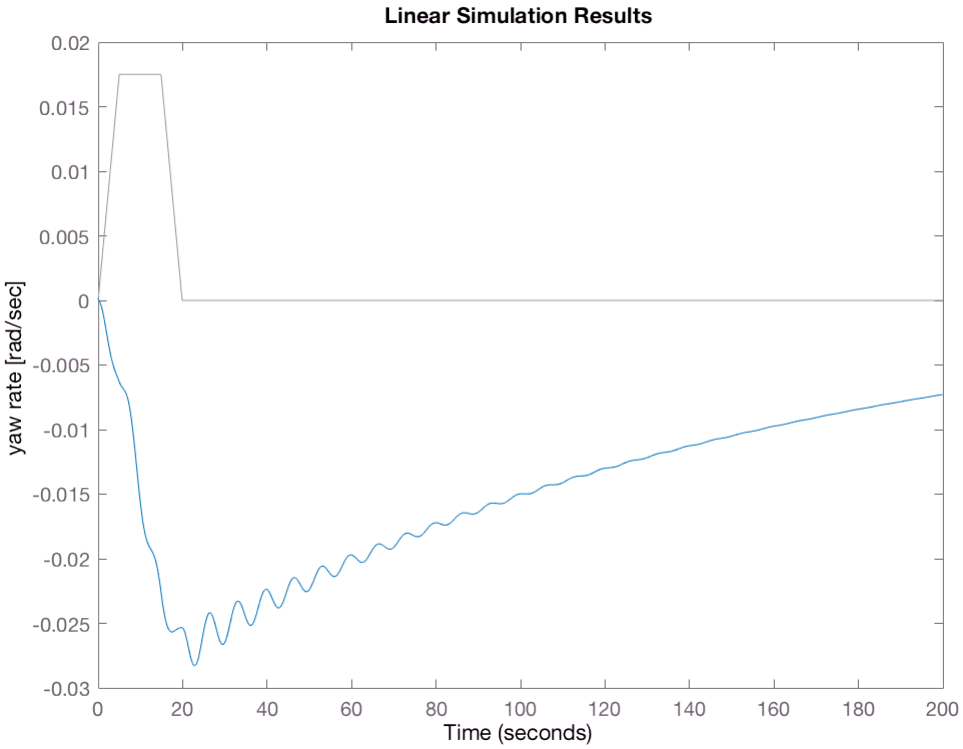
Problem 8 Calculate the zero-state response for the rudder deflection shown in Fig. 2 and discuss the aircraft's dynamical behaviour during these manoeuvres.



Solution:

The rudder time function can be made by combining four signals; the rising ramp, the steady value of 1, the decreasing ramp and the steady value 0. The Matlab code for making these signals and combining them to the rudder time function is shown in Algorithm 1. The rudder time function is input to the lsim()-function that returns a plot of the yaw rate. Notice that the input signal is also depicted in the plot-output from lsim():

```
figure
t = 0:.1:199.9; % Time vector
ut1 = (0:50) * 0.0175/50; % Rising Ramp
ut2 = ones(1,98) * 0.0175; % Steady value of 1
ut3 = (0:50) * (-0.0175/50) + 0.0175; % Decreasing ramp
ut4 = zeros(1,1800); % Steady value of 0
uttot = [ut1 ut2 ut3 ut4]; % Rudder time function
x0 = [0 0 0 0]'; % Initial State
lsim(A,B,C,D,uttot,t,x0)
ylabel('yaw rate [rad/sec]');
```



From a rudder change of only 1 deg, it takes more than four minutes to converge back to the original track, this could be for the sake of the passengers, however the time it takes to settle also says something about the speed at which the airplane is able to navigate, which is rather slow.

Linear Control Design II - Group Work Problem Module 9

Description

The dynamics of a wind turbine operating in the full load region reads

$$J_r\dot{\omega}_r = \frac{1}{2}\rho A R \mathrm{CP}(\beta,\lambda)\frac{v^2}{\lambda} - K_d\theta - B_d\left(\omega_r - \frac{\omega_g}{N_g}\right) \qquad (1)$$

$$J_g\dot{\omega}_g = \frac{K_d}{N_g}\theta + \frac{B_d}{N_g}\left(\omega_r - \frac{\omega_g}{N_g}\right) - T_g \qquad (2)$$

$$\dot{\theta} = \omega_r - \frac{\omega_g}{N_g} \qquad (3)$$

$$y_1 = \omega_r \qquad (4)$$

$$y_2 = P_e = \omega_g T_g \qquad (5)$$

where all quantities of interest are defined Group work Module 1. Based on the provided Simulink model '*OpenLoopWindTurbineModel.slx*' and on the system parameter file 'OpenLoopWindTurbine_Parameters.mat' address the following problems.

Problems

P1 Set the operating wind speed $v_{OP} = 20\ \mathrm{m/s}$ and compute analytically as well as numerically the operating point for the state vector $\mathbf{x}_{OP} = [\omega_{r,OP}, \omega_{g,OP}, \theta_{OP}]^{\mathrm{T}}$ and the input vector $\mathbf{u}_{OP} = [\beta_{OP}, T_{g,OP}]^{\mathrm{T}}$.

P1.Solution

Let us derive the operating point analiticaly. Five unknowns have to be determined including the input and the state vector in order to fully determine the operating point. As known from GroupWork module 1 the wind speed defines the range the Turbine is operating in. $v_{OP} = 20\ \mathrm{m/s}$ implies that the turbine is operating in the full load region fixing both P_e, the generated power in [MW] and ω_r, the rotor speed [rad/s] as

$$P_{e,OP} = 5 \quad mW$$

$$\omega_{r,OP} = 1.2671 \quad rad/s$$

At the operating point $\dot{\mathbf{x}}_{OP} = \left[\dot{\omega}_{r,OP}, \dot{\omega}_{g,OP}, \dot{\theta}_{OP}\right]^{\mathrm{T}} = 0$. From (3) it follows

$$\omega_{g,OP} = \omega_{r,OP} \cdot N_g \qquad (6)$$

From (5)

$$T_{g,OP} = \frac{P_{e,OP}}{\omega_{g,OP}} \qquad (7)$$

Substituting (3) in to (2) and (1) simplifies the expressions and lead to

$$\theta_{OP} = T_{g,OP}\frac{N_g}{K_d} \qquad (8)$$

$$\mathrm{CP}(\beta_{OP}, \lambda_{OP}) = 2\frac{K_d}{\rho A R}\frac{\lambda_{OP}}{v^2}\theta_{OP} \qquad (9)$$

(9) is a non linear equation that can be used to find β_{OP} knowing CP at the operating point. CP is in fact known because λ_{OP} is also known. In fact

$$\lambda_{OP} = \frac{\omega_{r,OP}\cdot R}{v_{OP}} \qquad (10)$$

(9) depends only on β_{OP} then.

```
clear
close all
clc

% Analytical Derivation
load('OpenLoopWindTurbine_Parameters.mat')
```

```
Error using load
OpenLoopWindTurbine_Parameters.mat is not found in the current folder or on the MATLAB path, but
exists in:
    /home/Emil/Documents/DTU_Masters/Linear Control Design 2/Lectures/Lecture09/Module 9

Change the MATLAB current folder or add its folder to the MATLAB path.
```

```
% Display the CP function contour plot and gradient field, just to visualize
it.
[dCpBeta, dCpLambda] = gradient(CP);

figure(1)
contour(Pitch,Lambda,CP)
hold on
quiver(Pitch,Lambda,dCpBeta,dCpLambda)
xlabel('$\beta [deg]$','Interpreter','latex')
ylabel('$\lambda$','Interpreter','latex')
title('CP contour and gradient field','Interpreter','latex')
hold off

% Following the equations above one finds all the unknowns but beta
v_OP = 20;  %[m/s]
omega_r_OP = 1.2671; %[rad/s]
Pe_OP = 5e6; %[W]

omega_g_OP = Ng*omega_r_OP;
Tg_OP = Pe_OP/omega_g_OP;
```

```
theta_Op = Tg_OP*Ng/Kd;

% Analytically only an aproximate beta can be found
LBD_OP = R*omega_r_OP/v_OP;
Diff = Lambda0-LBD_OP;  % Find the closest lambda to the operating point
value;
[~,Index_LBD_OP] = min(abs(Diff));  % Find the corresponding index;
CP_fun_beta = CP0(Index_LBD_OP,:); % Find the corresponding CP values given
lambda OP.

figure(2)
plot(Pitch0,CP_fun_beta)
xlabel('$\beta$','interpreter','latex')
ylabel('$C_p(\beta) [deg]$','interpreter','latex')
title('Level Curve at $\approx \lambda_{OP}$','Interpreter','latex')

% Exclude CP values corresponding to negative beta. There the turbine is
stalled.
i_beta_pos = Pitch0>0; % Logical vector. 1 if Beta >0. 0 elsewhere

figure(3)
plot(Pitch0(i_beta_pos),CP_fun_beta(i_beta_pos))
title('Level Curve at $\approx \lambda_{OP}$ and $\beta >
0$','Interpreter','latex')
xlabel('$\beta [deg]$','interpreter','latex')
ylabel('$C_p(\beta)$','interpreter','latex')

CP_OP_value = 2*omega_r_OP*Ng/(rho_air*A*v_OP^3)*Tg_OP; % CP value at the
operating point. See (9)
Diff1 = CP_fun_beta(i_beta_pos)-CP_OP_value; % Find the closest CP to
operating point.
[~,Index_Beta_OP] = min(abs(Diff1)); % Find the corresponding index
one = find(i_beta_pos == 1); % Find the first positive bete index.
Beta_OP_deg = Pitch0(one(1)+Index_Beta_OP) % [Deg]
Beta_OP_rad = Beta_OP_deg*pi/180 % [rad]
```

So far the operating point is not exactly determined because of β_{OP}. To exactly determine the operating point one has to trim the system. *trim()* inputs are

- Simulink model to trim;
- inital guesses for x_{OP}, u_{OP} and y_{OP};
- priority vector to each of the states, inputs and outputs. Here both the outputs are equaly important hence they are assigned with the same priority. No priority is given to the states and inputs.

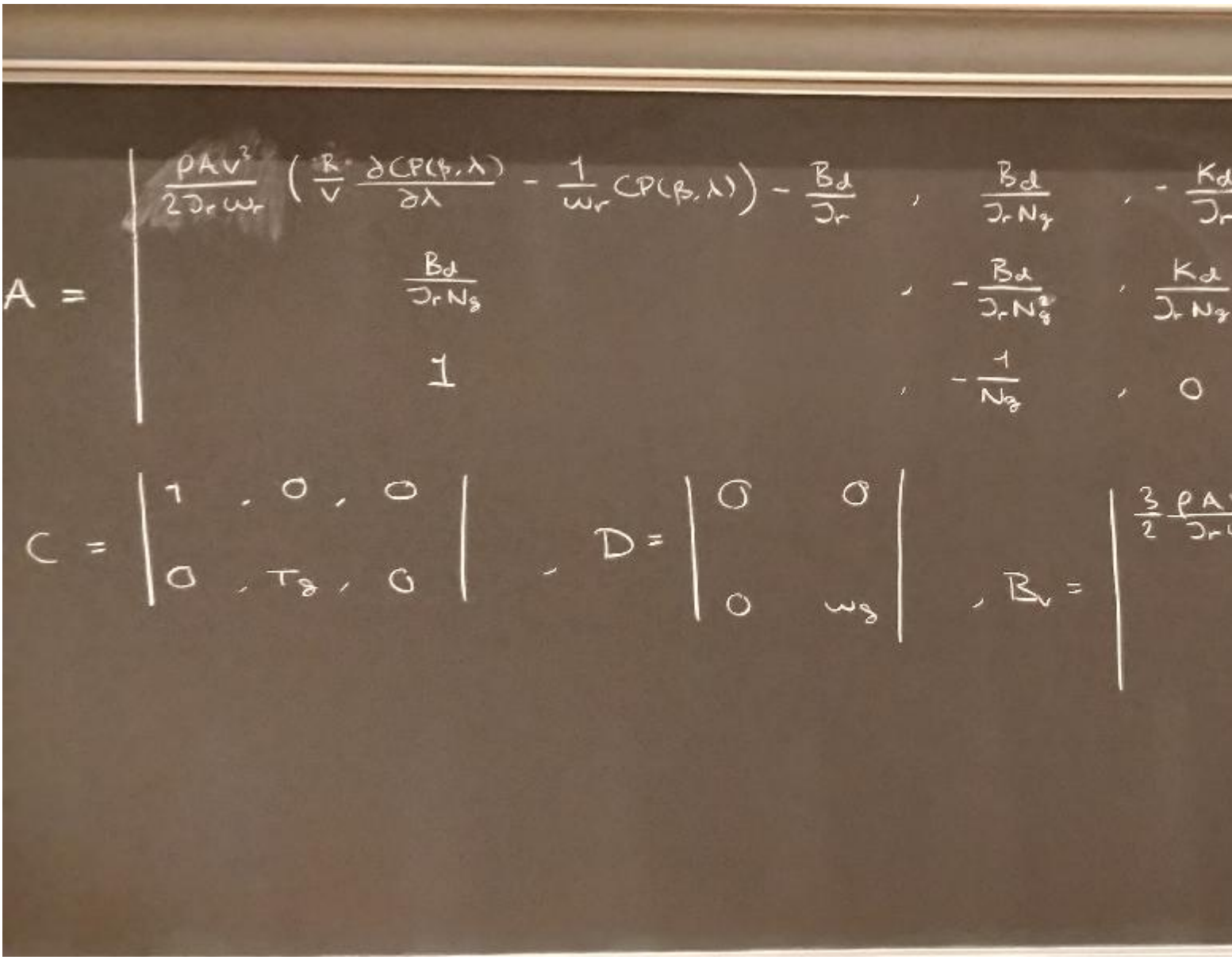
```
% Initial Guess
xOP_0 = [omega_r_OP;omega_g_OP;theta_Op];
uOP_0 = [Beta_OP_rad;Tg_OP];
yOP_0 = [omega_r_OP; 5e6];
```

```
x0 = xOP_0;% Integrators initial conditions
format short g
[xOP,uOP]=trim('OpenLoopWindTurbineModelTrim2015a',xOP_0,uOP_0,yOP_0,[],[],
[1 1])
```

P2 Linearize analytically and numerically the nonlinear wind turbine model around the operaitng point determined in **P1**.

P2.Solution

The analytic solution follows:



When using *linmod()* one has to recall that the order of the input and output ports in the Simulink model is used as reference to sort the state and input vector. Hence the order must be the same as specified in **P1**.

The disturbance v_{OP} acts on the system hence B_v has to be computed too. One can expand u_{OP} to include the disturbance. Finally B_v will be the last column of the output B matrix from *linmod()*.

Moreover second and third rows in the C and D matrices have to be removed since the actual outputs are ω_r and P_e.

```
% For the numerical linearization use the Matlab function linmod
x0 = xOP; % update initial conditions
uOPDist = [uOP;v_OP]; % update the input vector
[Aa,Bb,Cc,Dd] = linmod('OpenLoopWindTurbineModelLinmod2015a',xOP,uOPDist);
% Find the actual B, C and D matrix. Also find Bv matrix.
Aa
```

```
Aa = 3×3
    -0.32972    0.0010831      -14.667
      119.96      -1.2367        16747
           1    -0.010309            0
```

```
Bv = Bb(:,3)
```

```
Bv = 3×1
    0.020904
           0
           0
```

```
Bb = Bb(:,1:2)
```

```
Bb = 3×2
     -1.1066           0
           0   -0.0018723
           0           0
```

```
Cc = Cc([1 4],:)
```

```
Cc = 2×3
           1           0           0
           0    0.040681           0
```

```
Dd = Dd([1 4],1:2)
```

```
Dd = 2×2
           0           0
           0   0.00012291
```

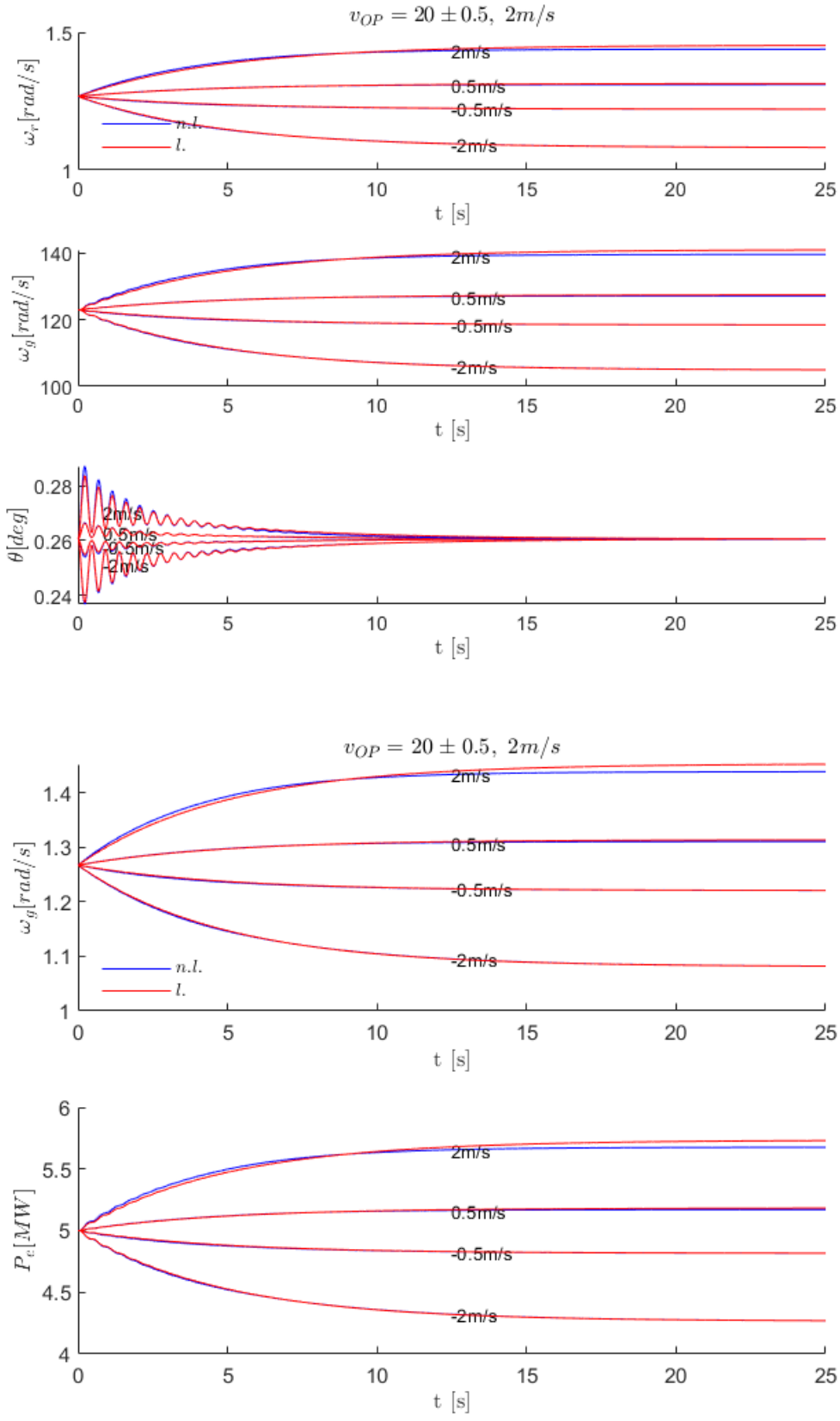
P3 Through simulation compare the state and output responses for step changes in the operating wind speed. Perform simulations for $\Delta v_{OP} = \pm 0.5$ m/s and for $\Delta v_{OP} = \pm 2$ m/s and discuss how well the linear model approximate the nonlinear dynamics.

P3.Solution

```
% Perform simulations around the operating point and plot the state and
output responses
DU = [0;0];
```

```
TIME_SIM = 25; % Simulation Time updated
for Dv_OP = [-0.5 0.5 -2 2]
        figure(4)
        sim('OpenLoopWindTurbineModelLVsNL2015a')

        % non linear states and output
        x1 = yout(:,1);
        x2 = yout(:,2);
        x3 = yout(:,3);
        y1 = yout(:,1);
        y2 = yout(:,4);

        % linear states and output
        x1l = yout(:,5);
        x2l = yout(:,6);
        x3l = yout(:,7)*180/pi;
        y1l = yout(:,8);
        y2l = yout(:,9);

        subplot(3,1,1)
        hold on
        plot(tout,x1,'b')
        text(tout(floor(end/2)),x1(floor(end/2)),[num2str(Dv_OP) 'm/
s'],'FontSize',8)
        plot(tout,x1l+xOP(1),'r')
        leg1 = legend('$n.l.$','$l.$','location','best');
        set(leg1,'Interpreter','latex');
        set(leg1,'FontSize',8);
        legend boxoff
        xlabel('t [s]','Interpreter','latex')
        ylabel('$\omega_r [rad/s]$','Interpreter','latex')
        title('$v_{OP} = 20 \pm 0.5, \ 2  m/s$','Interpreter','latex')
        hold off
        subplot(3,1,2)
        hold on
        plot(tout,x2,'b')
        text(tout(floor(end/2)),x2(floor(end/2)),[num2str(Dv_OP) 'm/
s'],'FontSize',8)
        plot(tout,x2l+xOP(2),'r')
        xlabel('t [s]','Interpreter','latex')
        ylabel('$\omega_g [rad/s]$','Interpreter','latex')
        hold off
        subplot(3,1,3)
        hold on
        plot(tout,x3,'b')
        text(tout(floor(end/30)),x3(floor(end/30)),[num2str(Dv_OP) 'm/
s'],'FontSize',8)
        plot(tout,x3l+xOP(3)*180/pi,'r')
        xlabel('t [s]','Interpreter','latex')
        ylabel('$\theta [deg]$','Interpreter','latex')
```

```
        hold off

        figure(5)
        subplot(2,1,1)
        hold on
        plot(tout,y1,'b')
        text(tout(floor(end/2)),y1(floor(end/2)),[num2str(Dv_OP) 'm/
s'],'FontSize',8)
        plot(tout,y1l+xOP(1),'r')
        leg1 = legend('$n.l.$','$l.$','location','best');
        set(leg1,'Interpreter','latex');
        set(leg1,'FontSize',8);
        legend boxoff
        xlabel('t [s]','Interpreter','latex')
        ylabel('$\omega_g [rad/s]$','Interpreter','latex')
        title('$v_{OP} = 20 \pm 0.5, \ 2  m/
s$','Interpreter','latex')
        hold off
        subplot(2,1,2)
        hold on
        plot(tout,y2,'b')
        text(tout(floor(end/2)),y2(floor(end/2)),[num2str(Dv_OP) 'm/
s'],'FontSize',8)
        plot(tout,y2l+Pe_OP/1e6,'r')
        xlabel('t [s]','Interpreter','latex')
        ylabel('$P_e [MW]$','Interpreter','latex')
        hold off
end
```



The model responds similarly to the variation in the wind speed though the larger the variation the less accurate the linear model compared to the non linear one.

P4 Sweep the wind speed in the range $v_{OP} \in [18, 25]\,\mathrm{m/s}$ and assess how the eigenvalues of the linear model vary in relation to changes in the operating wind speed. What conclusion can be reached about the stability of the origin of the linear model?

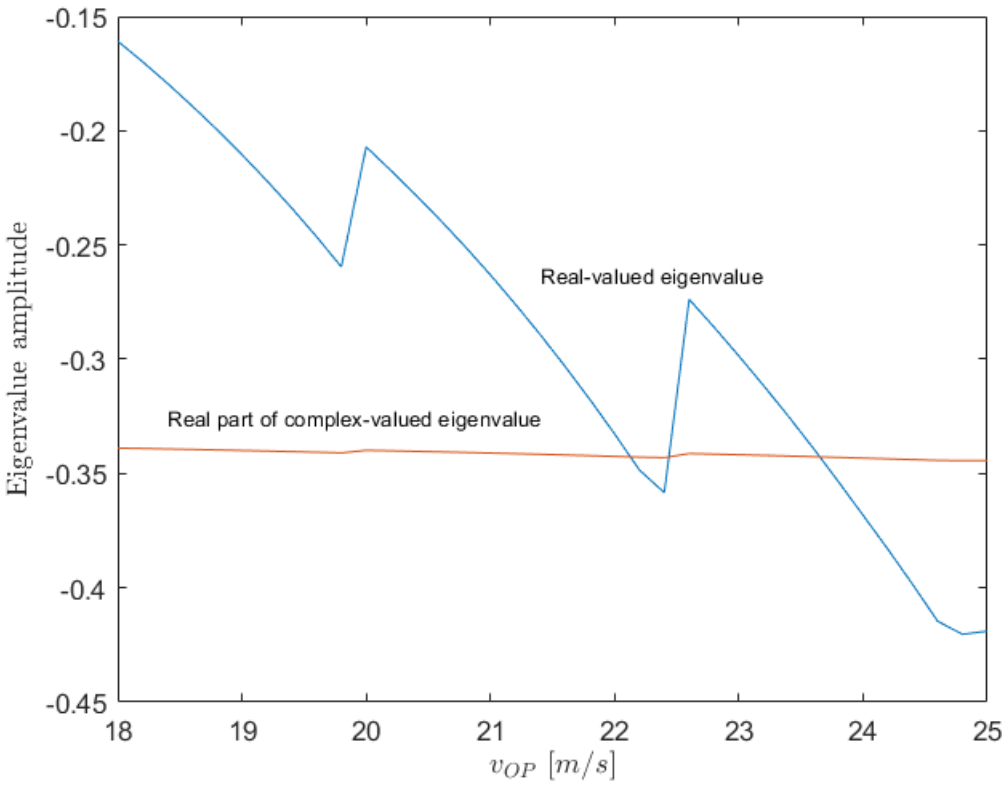
P4.Solution

It is assumed, that the rated power and angular velocity stay the same. The eigenvalues as a function of varying windspeed can be found by repeated use of trim and linmod at the different windspeeds. A possible way of finding it numerically is shown below:

```
% For each operating wind speed in the given range the system dynamical
matrix A needs to be recomputed.
v_OP_step = 0.2;
vVector = 18:v_OP_step:25;
ev = zeros(length(vVector),3);
for i = 1:length(vVector)
    v_OPn = 18+(i-1)*v_OP_step;
    xOP_0 = [omega_r_OP;omega_g_OP;theta_Op];
    uOP_0 = [Beta_OP_rad;Tg_OP;v_OPn];
    yOP_0 = [omega_r_OP; omega_g_OP;theta_Op;5];

    x0 = xOP_0;% Integrators initial conditions
    format short g
    [xOP,uOP]=trim('OpenLoopWindTurbineModelLinmod2015a',xOP_0,uOP_0,yOP_0,
[],[],[1 1]);
    [Aol,~,~,~] = linmod('OpenLoopWindTurbineModelLinmod2015a',xOP,uOP);
    ev(i,:) = eig(Aol);
end
```

Looking at the eigenvalues we have one real and a complex conjugated eigenvalue pair. The imaginary part of the complex eigenvalue does not change with varying windspeed. Hence only the real part is plotted below.

```
figure(6)
plot(vVector,ev(:,1))
hold on
plot(vVector,real(ev(:,2))*0.5) % The real part of the second eigenvalue is
scaled to better visualize it
xlabel('$v_{OP} \ [m/s]$','Interpreter','latex')
ylabel('Eigenvalue amplitude','Interpreter','latex')
text(vVector(floor(end/2)),ev(floor(end/2),1)/1.1,'Real-valued
eigenvalue','FontSize',8)
text(vVector(floor(end/10)),ev(floor(end/2),2)/2.1,'Real part of complex-
valued eigenvalue','FontSize',8)
```



P5 Set the wind speed at $v_{OP} = 20$ m/s and compute the modal matrix. Write the state response as the linear combination of the eignemodes of the system and determine which eigenmode is dominant in each of the state variables.

P5.solution

```
% The modal matrix can be found by using the Matlab function eig
[M,lbd,~] = eig(Aa);
% time constants
lbd = diag(lbd);
tau = -1./lbd(imag(lbd)==0)
```

```
tau =
       4.8291
```

```
% natural frequencies and damping ratios
wn = sqrt(real(lbd(2))^2+imag(lbd(2))^2)
```

```
wn =
       13.686
```

```
z = -real(lbd(2))/wn
```

```
z =
     0.049661
```

```
% Make Modal Matrix real
lbd = diag(lbd);
[M,~] = cdf2rdf(M,lbd)
```

```
M = 3×3
    -0.010311  -0.00087627  -1.4378e-05
     -0.99995           1            0
   1.2383e-05   3.9538e-05   0.00081633
```

ω_r is influenced mainly by the first eigenmode because -0.0103 is two and three orders of magnitude bigger than -8.7e-4 and -1.43e-5 respectively.

ω_g is mainly influenced by the first eigenmode and from the real part of the complex pair. The imaginary part has no effect.

θ is mainly influenced by the imaginary part of the complex pair. Again 8.16e-4 is 3 orders of magnitude bigger than 1.238e-5 and 3.953e-5.

Overall the analysis confirms what is seen in **Problem 3.**

Firstly $\omega_r(t)$ does not exhibit any oscillation. This is due to the fact that only the real eigenvalues contributes mainly to the response. Secondly $\omega_g(t)$ is slightly oscillatory during the transient phase probably because the complex eigenmode is weighted less than the real one in the partial fractions decomposition.

Finally $\theta(t)$ shows more evident oscillations because of the complex eigenmode.

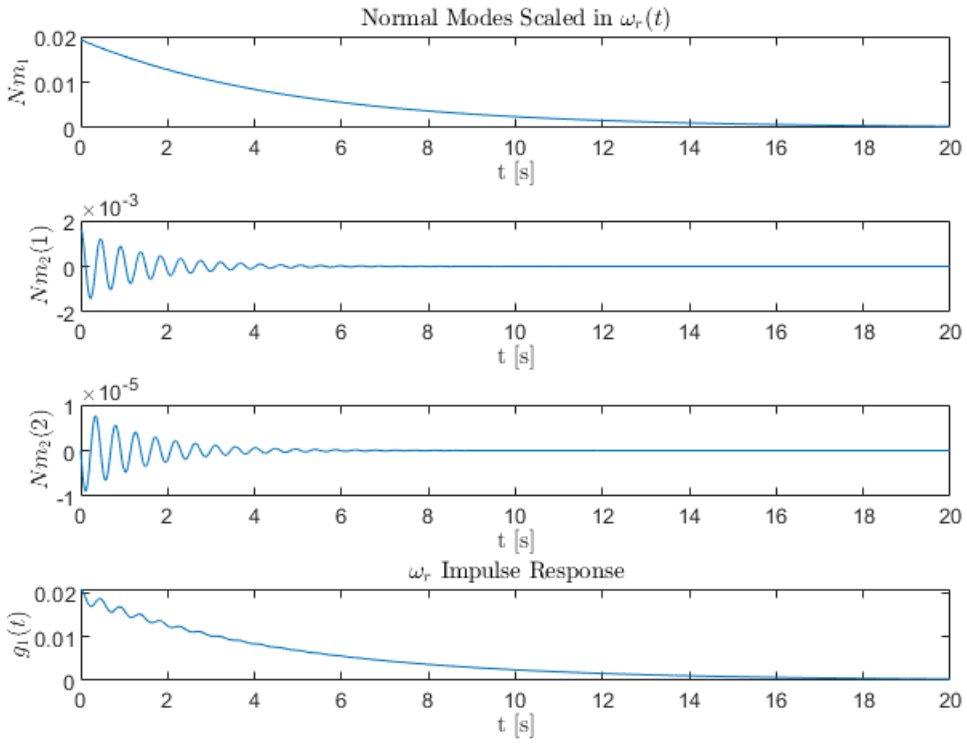
Now let us derive the partial fraction decomposition of the *States* transfer function matrix. For that reason let us assume C and D to be the I^3 and $0^{3,1}$ matrices respectively.

$B = B_v$ since so far we studied the response of the system provided some Δv_{OP}. Each state is then caracterized by a single transfer function.

```
Cn = eye(3);
Dn = zeros(3,1);

% Transfer Function Matrix v (Disturbance) -> X (states)
sysn = ss(Aa,Bv,Cn,Dn);
Gs = minreal(tf(sysn));

% Find Constant residuals matrix
syms ax bx c1 s
lbd = diag(lbd);
Pol = c1*(s^2+2*z*wn*s+wn^2)+(ax*(s-real(lbd(2)))+bx*abs(imag(lbd(2))))*(s-
lbd(1));
Pol = expand(Pol);
Pol = vpa(Pol);
```

```
% Omega_r
```

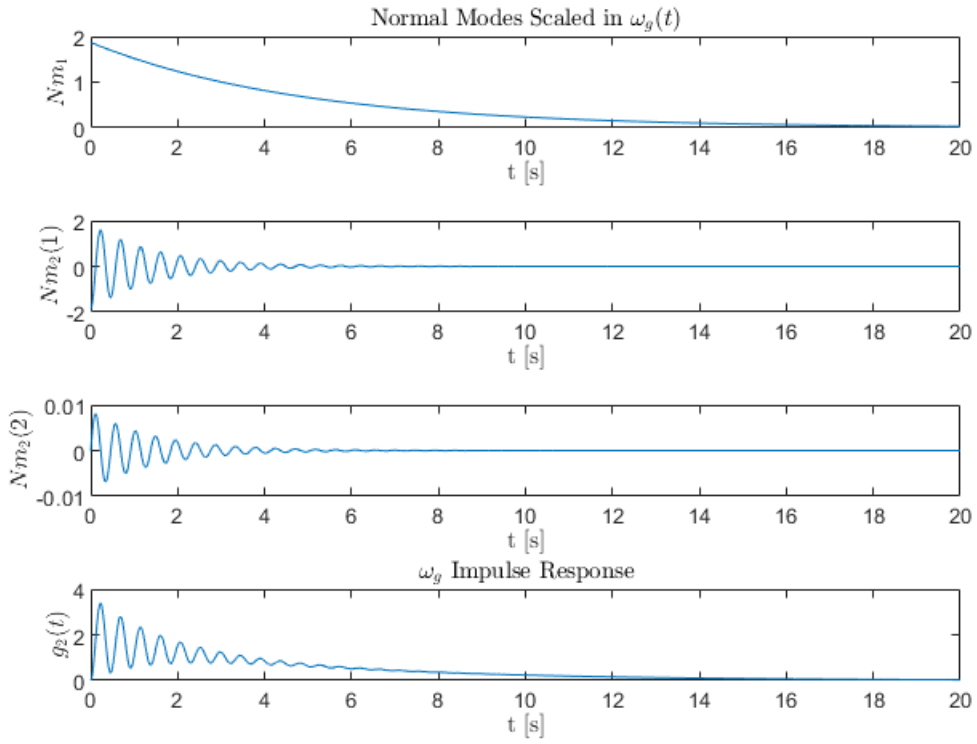
```
CfRes = coeffs(Pol,s);
CfTf = fliplr(Gs.Numerator{1}(2:end));

eqn1 = CfRes(1) - CfTf(1);
eqn2 = CfRes(2) - CfTf(2);
eqn3 = CfRes(3) - CfTf(3);

sol = solve([eqn1 == 0,eqn2 == 0,eqn3 == 0],[c1 ax bx]);

% Plot Natural Modes Weighted by the constant residuals
t= 0:0.01:20;
t2 = 0:0.01:20;

Nm2 = [sol.ax*exp(real(lbd(2))*t2).*cos(abs(imag(lbd(2)))*t2)
      sol.bx/
abs(imag(lbd(2)))*exp(real(lbd(2))*t2).*sin(abs(imag(lbd(2)))*t2)];
Nm = sol.c1*exp(lbd(1)*t);

figure(7)
subplot(4,1,1)
plot(t,Nm)
hold on
title('Normal Modes Scaled in $\omega_r(t)$','Interpreter','latex')
xlabel('t [s]','Interpreter','latex')
ylabel('$ Nm_1$','Interpreter','latex')
hold off
subplot(4,1,2)
plot(t2,Nm2(1,:))
hold on
xlabel('t [s]','Interpreter','latex')
ylabel('$ Nm_2$(1)','Interpreter','latex')
hold off
subplot(4,1,3)
plot(t2,Nm2(2,:))
hold on
xlabel('t [s]','Interpreter','latex')
ylabel('$Nm_2$(2)','Interpreter','latex')
hold off
subplot(4,1,4)
plot(t2,Nm+Nm2(1,:)+Nm2(2,:))
hold on
title('$\omega_r$ Impulse Response','Interpreter','latex')
xlabel('t [s]','Interpreter','latex')
ylabel('$g_1(t)$','Interpreter','latex')
hold off
```



$\omega_r(t)$ impulse response reflects what stated above, i.e that the response is mainly influenced by the real eigenmode. Still the complex conjugate pair exerts some influence. But it is limited due to the magnitude of the corresponding eigenmodes.

```
% Omega_g
CfTf = fliplr(Gs.Numerator{2}(2:end));

eqn1 = CfRes(1) - CfTf(1);
eqn2 = CfRes(2) - CfTf(2);
eqn3 = CfRes(3) - CfTf(3);
sol = solve([eqn1 == 0,eqn2 == 0,eqn3 == 0],[c1 ax bx]);

% Plot Natural Modes Weighted by the constant residuals
t= 0:0.01:20;
t2 = 0:0.01:20;

Nm2 = [sol.ax*exp(real(lbd(2))*t2).*cos(abs(imag(lbd(2)))*t2)
      sol.bx/
abs(imag(lbd(2)))*exp(real(lbd(2))*t2).*sin(abs(imag(lbd(2)))*t2)];
Nm = sol.c1*exp(lbd(1)*t);

figure(8)
subplot(4,1,1)
plot(t,Nm)
```

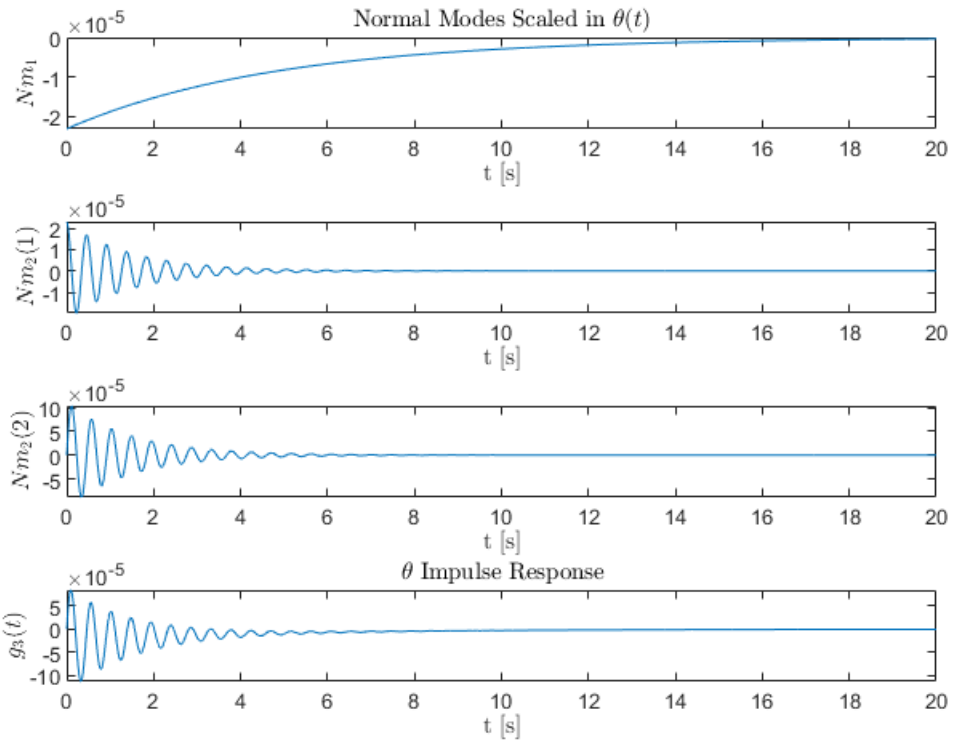
```
hold on
title('Normal Modes Scaled in $\omega_g(t)$','Interpreter','latex')
xlabel('t [s]','Interpreter','latex')
ylabel('$ Nm_1$','Interpreter','latex')
hold off
subplot(4,1,2)
plot(t2,Nm2(1,:))
hold on
xlabel('t [s]','Interpreter','latex')
ylabel('$ Nm_2$(1)','Interpreter','latex')
hold off
subplot(4,1,3)
plot(t2,Nm2(2,:))
hold on
xlabel('t [s]','Interpreter','latex')
ylabel('$Nm_2$(2)','Interpreter','latex')
hold off
subplot(4,1,4)
plot(t2,Nm+Nm2(1,:)+Nm2(2,:))
hold on
title('$\omega_g$ Impulse Response','Interpreter','latex')
xlabel('t [s]','Interpreter','latex')
ylabel('$g_2(t)$','Interpreter','latex')
hold off
```



$\omega_g(t)$ is equally influenced by the real eigenmode and the real part of the complex eigenmode. In fact their magnitude is comparable at the beginning of the time horizon. The contribute of the imaginary part is two order of magnitude smaller.

```
% Theta
CfTf = fliplr(Gs.Numerator{3}(2:end));

eqn1 = CfRes(1) - CfTf(1);
eqn2 = CfRes(2) - CfTf(2);
eqn3 = CfRes(3) - CfTf(3);
sol = solve([eqn1 == 0,eqn2 == 0,eqn3 == 0],[c1 ax bx]);

% Plot Natural Modes Weighted by the constant residuals
t= 0:0.01:20;
t2 = 0:0.01:20;

Nm2 = [sol.ax*exp(real(lbd(2))*t2).*cos(abs(imag(lbd(2)))*t2)
      sol.bx/
abs(imag(lbd(2)))*exp(real(lbd(2))*t2).*sin(abs(imag(lbd(2)))*t2)];
Nm = sol.c1*exp(lbd(1)*t);

figure(9)
subplot(4,1,1)
plot(t,Nm)
hold on
title('Normal Modes Scaled in $\theta(t)$','Interpreter','latex')
xlabel('t [s]','Interpreter','latex')
ylabel('$ Nm_1$','Interpreter','latex')
hold off
subplot(4,1,2)
plot(t2,Nm2(1,:))
hold on
xlabel('t [s]','Interpreter','latex')
ylabel('$ Nm_2$(1)','Interpreter','latex')
hold off
subplot(4,1,3)
plot(t2,Nm2(2,:))
hold on
xlabel('t [s]','Interpreter','latex')
ylabel('$Nm_2$(2)','Interpreter','latex')
hold off
subplot(4,1,4)
plot(t2,Nm+Nm2(1,:)+Nm2(2,:))
hold on
title('$\theta$ Impulse Response','Interpreter','latex')
xlabel('t [s]','Interpreter','latex')
ylabel('$g_3(t)$','Interpreter','latex')
```

```
hold off
```



Finally the impulse response of $\theta(t)$. As already seen in the modal matrix the imaginary part of the complex pair plays the most relevant role.

P6 Determine a suitable sampling time for the linear system and discretize the linear model. Compute the eigenvalues of the discrete time linear model and use Frobenius theorem to show the equivalence with the eigenvalues of the continuous time linear system.

P6.Solution

To choice of a suitable samping time depends on the fastest dynamics that has to be captured.

The time constant refers to the transient phase of the real eigenmode.

The period $T = 2 \cdot \pi/\omega_n$ deals with the time interval such that the imaginary and the real parts of the complex pair repeats.

The sampling time is chosen 10 time smaller than the smaller between T and τ.

```
% To discretize the continuous time linear system use the Matlab function c2d
T = 2*pi/wn;
if T < tau(1)
    Ts = tau(1)/10;
else
    Ts = T/10;
```

```
end
[Ff,Gg] = c2d(Aa,Bb,Ts) %discretize the LTIC
```

```
Ff = 3×3
      0.88616   0.00018968     -0.23805
       21.009      0.69162        276.1
     0.016231  -0.00016997      0.69531
Gg = 3×2
     -0.47011  -6.6998e-07
      -43.858  -9.5855e-05
   -0.0012595   3.2197e-08
```

```
lbd_d = eig(Ff) %find the discrete eigenvalues
```

```
lbd_d = 3×1 complex
      0.90484 +          0i
      0.68413 +    0.22509i
      0.68413 -    0.22509i
```

```
lbd_d_calc = exp(Ts*lbd) %apply Frobenius theorem
```

```
lbd_d_calc = 3×1 complex
      0.90484 +          0i
      0.68413 +    0.22509i
      0.68413 -    0.22509i
```

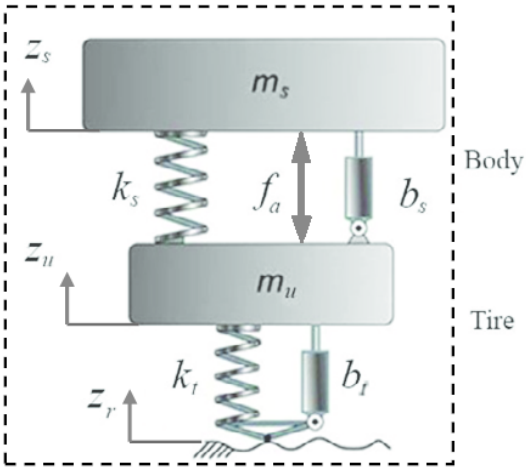
Linear Control Design II - Group Work Problem Module 10

Description

The suspension system is a key component of a vehicle directly affecting the passenger comfort and the ability of the vehicle itself to hold the road. In active suspension system an actuator is positioned between the sprung mass of the vehicle and the unsprung mass (wheel, tyre, brake, etc.) parallel to the suspension elements with the objective of pulling/pushing the vehicle body in order to suppress the vibrations due to road irregularities.

Quarter-car suspension system

A simplified car suspension system is shown on the following figure



where

- m_s, b_s, k_s are the mass, damping and stiffness of the sprung element (quarter-car)
- m_u, b_t, k_t are the mass, damping and stiffness of the unsprung element (wheel-tyre-brake)
- z_s is the displacement of the sprung element
- z_u is the displacement of the unsprung element
- z_r is the displacement of the road
- f_a is the force exerted by an hydraulic actuator placed between the sprung and unsprung elements

The applied force on the tyre from the road

$$f_d = b_t\dot{z}_r + k_t z_r \qquad (1)$$

is considered as an unknown disturbance in the sytem, which should be properly handled by the active suspension.

The numerical values for the model parameters are provided right below.

```
% Model paramters (RG)
% Test the provided active suspension model on deterministic road profile
(RG)
```

```
SIM_TIME = 100;
STEP_SIZE = 0.01; % integration step size of Simulink (can be changed as
needed)
% System parameters
m_s = 243; % sprung mass [kg]
m_u = 40; % unsprung mass [kg]
b_s = 370; % damping coefficient [N*s/m]
b_t = 414; % tyre damping coefficient [N*s/m]
k_s = 14761; % stiffness coefficient [N/m]
k_t = 124660; % tyre stiffness coefficient [N/m]
```

Open-loop system analysis

Problem 1 Looking at the diagram of the quarter-car suspension and applying Newton's second law derive a state space model of the system. Identify state variables, control inputs and possible disturbances. Then compute the stationary state when the control input and the disturbnace are equal to zero.

Solution Applying Newton's second law two second order differential equations are derived that described the coupled dynamics of the sprung and unsprung masses, i.e.

$$\begin{aligned} m_s\ddot{z}_s &= -b_s(\dot{z}_s - \dot{z}_u) - k_s(z_s - z_u) + f_a \\ m_u\ddot{z}_u &= b_s(\dot{z}_s - \dot{z}_u) + k_s(z_s - z_u) - f_a + b_t(\dot{z}_r - \dot{z}_u) + k_t(z_r - z_u) \end{aligned} \tag{2}$$

Using the definition of the unknown disturbance given in Eq. (1), then Eq. (2) can be rewritten as

$$\begin{aligned} m_s\ddot{z}_s &= -b_s(\dot{z}_s - \dot{z}_u) - k_s(z_s - z_u) + f_a \\ m_u\ddot{z}_u &= b_s(\dot{z}_s - \dot{z}_u) - b_t\dot{z}_u + k_s(z_s - z_u) - k_t z_u - f_a + f_d \end{aligned} \tag{3}$$

Given the model (3), the state vector is defined as $\mathbf{x} = [z_s, \dot{z}_s, z_u, \dot{z}_u]^{\mathrm{T}}$, the control input is the force exerted by the hydraulic actuator f_a, and the disturbance is applied force on the tyre f_d.

The state space model is then given by

```
% Your solutions goes here

A = [0 1 0 0; -k_s/m_s -b_s/m_s k_s/m_s b_s/m_s; 0 0 0 1; k_s/m_u b_s/m_u
-(k_s+k_t)/m_u -(b_s+b_t)/m_u]
```

```
A = 4×4
10^3 ×
         0    0.0010         0         0
   -0.0607   -0.0015    0.0607    0.0015
         0         0         0    0.0010
    0.3690    0.0092   -3.4855   -0.0196
```

```
B = [0 1/m_s 0 -1/m_u]';
Bv = [0 0 0 1/m_u]';
```

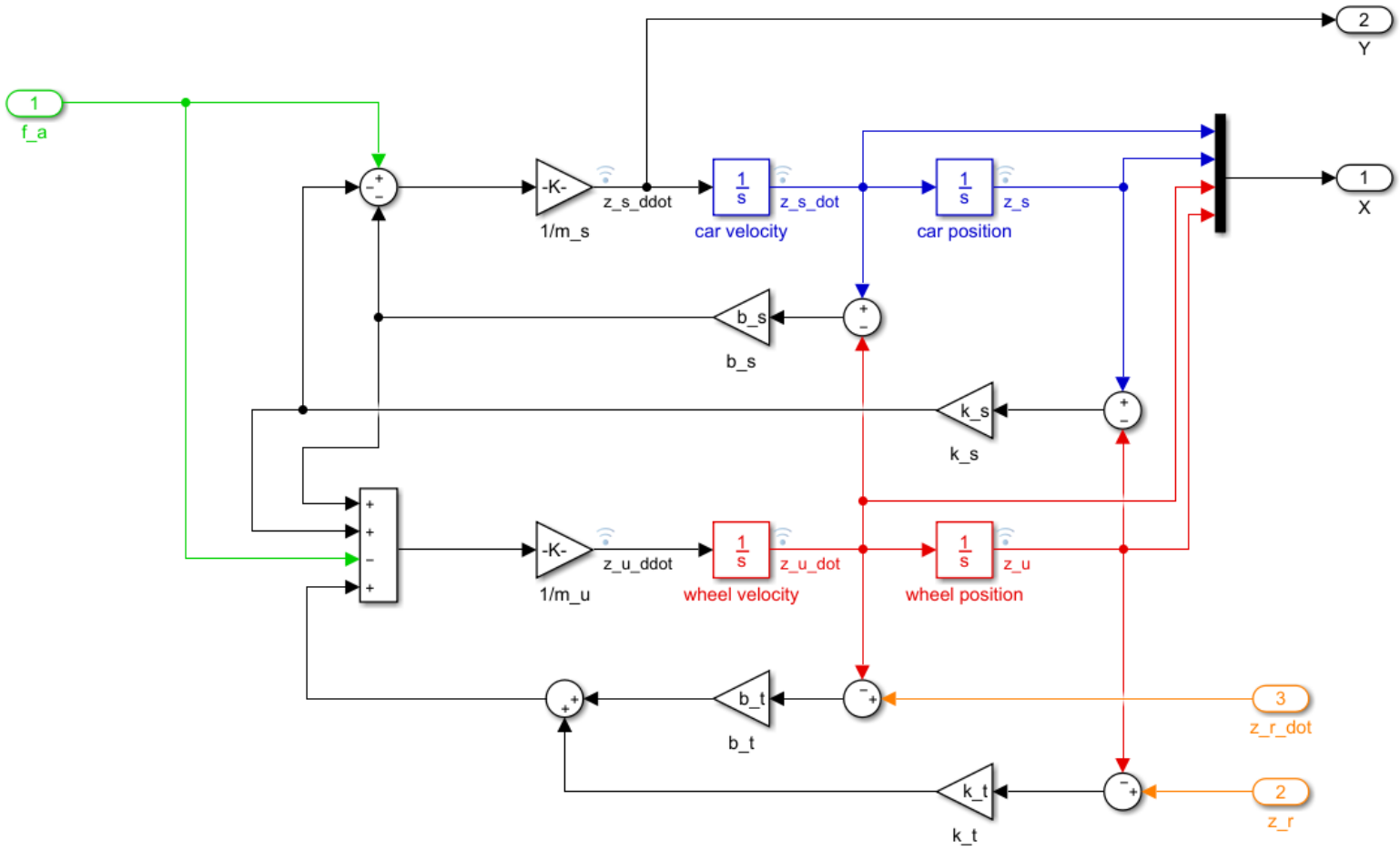
If the control input and the disturbance are both set to zero, then the stationary point of the system is $\mathbf{x}_{ss} = [0, 0, 0, 0]^{T}$. Please notice that the numerical solution is confirming the analysis since the numerical entries of the state, input and output vectors have magnitude 10^-13 or smaller.

```
x0 = [0.01,0,0.01,0]'; % initial condition for finding the stationary state
u0 = [0,0,0]';
[xss,uss,yss] = trim('QuarterCar_ActiveSusp_Simulink2019b',x0,u0,[],[],
[1;2;3],[])
```

```
xss = 4×1
10^-13 ×
   -0.5939
   -0.0393
   -0.5939
   -0.0393
uss = 3×1
10^-32 ×
   -0.0433
    0.0241
    0.9647
yss = 5×1
10^-13 ×
   -0.5939
   -0.0393
   -0.5939
   -0.0393
    0.0015
```

Problem 2 Draw a block diagram of the system and then implement the model in Simulink.

Solution The block diagram (and the Simulink diagram) can be drawn starting from the differential equations in (2) and placing the integrators of the system first. Once the integrators are in place, the next step will be to sketch the feedback loops connecting the output of each integrator to the input of the integrators mapping accelerations into velocities. Last summation points are put in place and inputs and outputs are connected. The block diagram is then implemented into the Simulink model "QuarterCar_ActiveSusp_Simulinnk2019b.slx".




```
% Your solution goes here
```

Problem 3 Set the initial position of the sprung element to $z_s = -0.05\,\mathrm{m}$ and simulate the response of the system when all inputs are at zero and the other state variables retain the stationary value.

```
% Your solutions goes here

% To compute the zero input response we set the initial condition of the
% state vector to the specified value and then we run the simulation
% exploiting the simulink diagram.

SIM_TIME = 10;
x0 = [-0.05,0,0,0]'; % initial condition for the state vector
sim('QuarterCar_ActiveSusp_Simulink2019b',SIM_TIME,[],[]);

time_det = logsout.getElement('z_s').Values.Time;
z_s = logsout.getElement('z_s').Values.Data;
z_s_dot = logsout.getElement('z_s_dot').Values.Data;
z_u = logsout.getElement('z_u').Values.Data;
z_u_dot = logsout.getElement('z_u_dot').Values.Data;
```

```
figure, h1 = subplot(2,2,1); set(h1,'FontName','times','FontSize',16)
```

```
Warning: MATLAB has disabled some advanced graphics rendering features by switching to software
OpenGL. For more information, click here.
```

```
hold on, grid on
plot(time_det,z_s,'k','LineWidth',1.5)
ylabel('$z_s$ [m]','FontName','times','FontSize',16,'Interpreter','latex')
h2 = subplot(2,2,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(time_det,z_s_dot,'k','LineWidth',1.5)
ylabel('$\dot{z}_s$ [m$/
$s]','FontName','times','FontSize',16,'Interpreter','latex')
h3 = subplot(2,2,3); set(h3,'FontName','times','FontSize',16)
hold on, grid on
plot(time_det,z_u,'k','Bv = [0 0 0 1/m_u]';LineWidth',1.5)
ylabel('$z_u$ [m]','FontName','times','FontSize',16,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
h4 = subplot(2,2,4); set(h4,'FontName','times','FontSize',16)
hold on, grid on
plot(time_det,z_u_dot,'k','LineWidth',1.5)
ylabel('$\dot{z}_u$ [m$/
$s]','FontName','times','FontSize',16,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
```

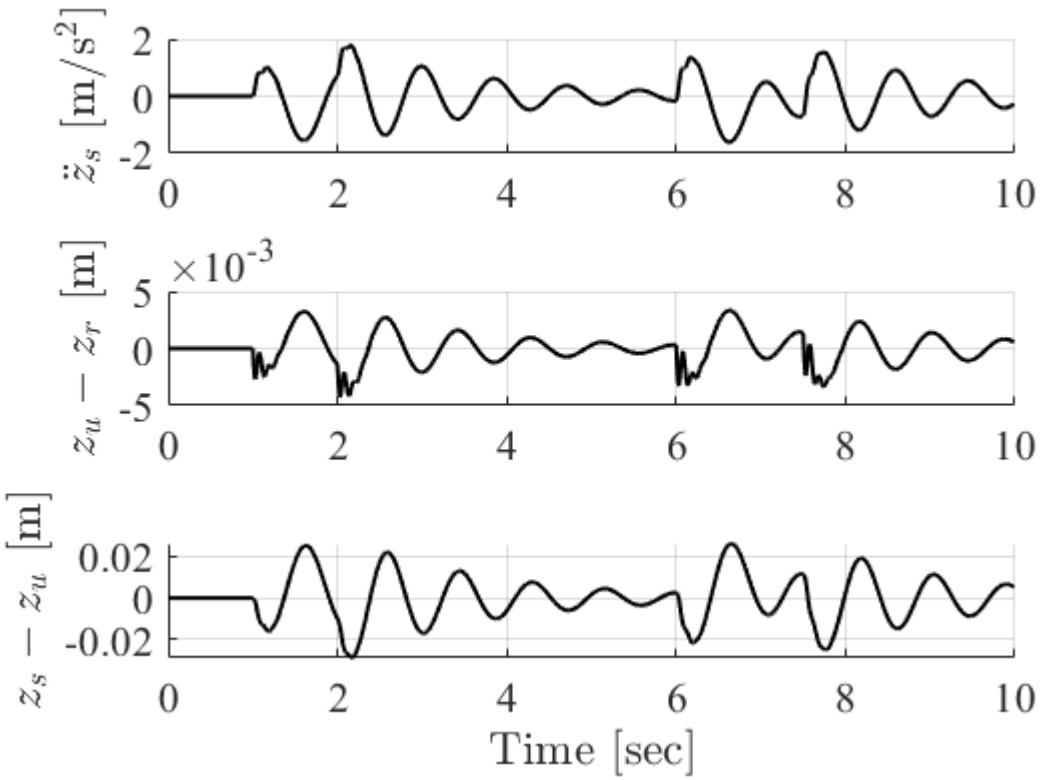
Problem 4 Assess the open-loop characteristics of the

- vertical acceleration $\ddot{z}_s$
- tire deflection $(z_u - z_r)$
- suspension deflection $(z_s - z_u)$

in terms of largest deviation from the stationary state, decay rate and settling time when the system is subject to the deterministic road profile $(z_r(t), \dot{z}_r(t))_{\text{det}}$.

```
% Your solutions goes here

% Road profiles for open simulations (RG)
load road_profile.mat
time_det = time_det'; % column vector
z_r_det = z_r_det'; % column vector
% [t_det,z_r_det,z_r_dot_det] - deterministic road profile where t_det is the
% travelling time, z_r_det is the vertical road displacement and z_r_dot_det
% is the rate of change of the road profile.
% The profile has been generated for a car traveling with a forward
```

```
% speed of 40 km/h

SIM_TIME = time_det(end);
x0 = zeros(4,1); % remember to set the initial conditions to zero, since we
are computing the zero state response
f_a = zeros(length(time_det),1); % control input set to zero since we are
assessing the
% system behaviour in reponse to the road disturance
sim('QuarterCar_ActiveSusp_Simulink2019b',SIM_TIME,[],[time_det f_a z_r_det
z_r_dot_det])

% Plot the temporal behaviour of
%   *   vertical accelaration
%   *   tire deflection
%   *   suspension deflection

z_s = logsout.getElement('z_s').Values.Data;
z_s_dot = logsout.getElement('z_s_dot').Values.Data;
z_s_ddot = logsout.getElement('z_s_ddot').Values.Data;
z_u = logsout.getElement('z_u').Values.Data;
z_u_dot = logsout.getElement('z_u_dot').Values.Data;

tire_defl = z_u - z_r_det;
susp_defl = z_s - z_u;

figure, h6 = subplot(3,1,1); set(h6,'FontName','times','FontSize',16)
hold on, grid on
plot(time_det,z_s_ddot,'k','LineWidth',1.5)
ylabel('$\ddot{z}_s$ [m/
s$^2$]','FontName','times','FontSize',16,'Interpreter','latex')
h7 = subplot(3,1,2); set(h7,'FontName','times','FontSize',16)
hold on, grid on
plot(time_det,tire_defl,'k','LineWidth',1.5)
ylabel('$z_u -z_r$
[m]','FontName','times','FontSize',16,'Interpreter','latex')
h8 = subplot(3,1,3); set(h8,'FontName','times','FontSize',16)
hold on, grid on
plot(time_det,susp_defl,'k','LineWidth',1.5)
ylabel('$z_s-z_u$
[m]','FontName','times','FontSize',16,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
```




```
max_z_s_ddot = max(abs(z_s_ddot));
disp('The maximum peak of the vertical acceleration in open loop is [m/
s^2]:')
```

```
The maximum peak of the vertical acceleration in open loop is [m/s^2]:
```

```
disp(max_z_s_ddot)
```

```
    1.7917
```

```
max_tire_defl = max(abs(tire_defl));
disp('The maximum peak of the tire deflection in open loop is [m]:')
```

```
The maximum peak of the tire deflection in open loop is [m]:
```

```
disp(max_tire_defl)
```

```
    0.0043
```

```
max_susp_defl = max(abs(susp_defl));
disp('The maximum peak of the suspension deflection in open loop is [m]:')
```

```
The maximum peak of the suspension deflection in open loop is [m]:
```

```
disp(max_susp_defl)
```

```
    0.0289
```

An estimate of the decay rate and settling time can be obtained through the analysis of the system eigenvalues

```
[wn,zeta] = damp(A) % the function damp computes the natural frequencies
```

```
wn = 4×1
   59.0183
   59.0183
    7.3723
    7.3723
zeta = 4×1
    0.1685
    0.1685
    0.0836
    0.0836
```

```
% and damping ratios associated with the eigenvalues of the system
% dynamical matrix A
T_osc = 2*pi./[wn(1) wn(3)] % periods of oscialltory modes
```

```
T_osc = 1×2
    0.1065    0.8523
```

Two oscillatory modes characterize the state and output response of the system: one oscillation with period of approximately 0.1s, and a second oscillation with period of approximately 0.85s. These different oscillations are clearly visible from the plots of acceleration, tire and suspension deflection. The decay rate and settling time are determined by the oscillatory mode with the larger oscillation period, which is also associated with the smallest damping ratio ($\zeta = 0.0836$).

```
% Decay rate
alpha = -wn(3)*zeta(3) % this is the real part of the complex eigenvalue,
which is also appearing in the exponent of the oscillatory eigenmode
```

```
alpha = -0.6162
```

```
% Settling time
t_set = 5*1/abs(alpha) % the settling time [s] is estimated as 5 times the
time constant associated with the slow complex eigenvalue
```

```
t_set = 8.1137
```

Problem 5 Compute the eigenvalues of the system and asses the internal stability properties. Calculate the damping rato and natural frequency of each couple of complex poles.

```
% Your solutions goes here
lambda_A_ol = eig(A);
disp(lambda_A_ol)
```

```
  -9.9451 +58.1744i
  -9.9451 -58.1744i
  -0.6162 + 7.3465i
  -0.6162 - 7.3465i
```

```
disp('The real part of all four eigenvalues is negative, hence')
```

```
The real part of all four eigenvalues is negative, hence
```

```
disp('the system is internally asymptotically stable.')
```

```
the system is internally asymptotically stable.
```

```
% The damping ratios and the natural frequencies can be assessed using the
% function damp or throug the formulas:
% wn = sqrt(Re{lambda}^2 + Im{lambda}^2)
% zeta = -Re{lambda}/wn

[wn_A_ol,zeta_A_ol] = damp(A)
```

```
wn_A_ol = 4×1
   59.0183
   59.0183
    7.3723
    7.3723
zeta_A_ol = 4×1
    0.1685
    0.1685
    0.0836
    0.0836
```

```
wn_A_ol2 = sqrt(real(lambda_A_ol).^2+imag(lambda_A_ol).^2)
```

```
wn_A_ol2 = 4×1
   59.0183
   59.0183
    7.3723
    7.3723
```

```
zeta_A_ol2 = -real(lambda_A_ol)./wn_A_ol2
```

```
zeta_A_ol2 = 4×1
    0.1685
    0.1685
    0.0836
    0.0836
```

From this analysis we can see once again that the system has two under damped oscillatory modes.

Problem 6 Calculate the transfer function from the control input f_a to the vertical acceleration of sprung mass $\ddot{z}_s$. Determine the position of the system poles and conclude about extenal stability.

Solution To compute the transfer function from the control input to the vertical acceleration of the sprung mass, we need to set such acceleration as an ouput of the linear system. To do that we can notice that the acceleration of the sprung mass is actually given by the second state equation. Therefore by setting the output matrix to

$$\mathbf{C} = \begin{bmatrix} -\frac{k_s}{m_s} & -\frac{b_s}{m_s} & \frac{k_s}{m_s} & \frac{b_s}{m_s} \end{bmatrix}$$

we can generate the needed system output.

```
% Your solutions goes here
C = [-k_s/m_s -b_s/m_s k_s/m_s b_s/m_s];
[numG,denG] = ss2tf(A,[B Bv],C,zeros(1,2),1); % the first input is the
control input

disp('The transfer function from control input f_a to acceleration of the
sprung mass is')
```

```
The transfer function from control input f_a to acceleration of the sprung mass is
```

```
minreal(tf(numG,denG)) % the minreal function ensures that zero-pole
cancellations are done
```

```
ans =


  -0.04433 s^3 - 1.833 s^2 - 22.12 s

                               - 779.1

  -------------------------------------

  s^4 + 21.12 s^3 + 3562 s^2 + 5374 s

                            + 1.893e05


Continuous-time transfer function.
```

Problem 7 Select a suitable sampling time T_s and discretize the continuous time model obtained in Problem 1.

Solution Since we have no control requirements the selection of the sampling time is solely based on the dynamic characteristics of the system. In particular we should consider the smallest period of oscillation of the system and take the sampling time to be at least 10 times as fast. The validity of the chosen sampling time could be verified e.g. by simulating the continuous time and the discrete time systems subject to the same input or initial conditions and comparing the obtained responses.

```
% Your solutions goes here
% In Problem 4 the periods of oscillation T_osc were already computed. So
% we can take the smallest of the found values and divide that by e.g. 10

min_T_osc = min(T_osc);
Ts = min_T_osc/10;
disp('The sampling time is')
```

```
The sampling time is
```

```
disp(Ts)
```

```
    0.0106
```

```
% The discrete time linear model is then found by applying the c2d function
[F,G] = c2d(A,B,Ts);
[~,Gv] = c2d(A,Bv,Ts);

susp_ss_dt = ss(F,[G Gv],C,zeros(1,2),Ts) % generate the discrete time state
space model
```

```
susp_ss_dt =

  A =
              x1         x2         x3
   x1     0.9967    0.01055   0.002325
   x2    -0.6081     0.9814     0.3312
   x3    0.01879  0.0005399     0.8215
   x4      3.285     0.1011     -31.31

              x4
   x1  8.887e-05
   x2    0.01665
   x3   0.008992
   x4     0.6461

  B =
               u1          u2
   x1    2.24e-07   7.868e-09
   x2    4.12e-05   2.222e-06
   x3  -1.273e-06   1.281e-06
   x4  -0.0002226   0.0002248

  C =
          x1      x2      x3      x4
   y1  -60.74  -1.523   60.74   1.523

  D =
       u1  u2
   y1   0   0

Sample time: 0.010646 seconds
Discrete-time state-space model.
```

Linear Control Design II - Group Work Problem Module 11 Solution

Description

The airplane model is to be extended by adding a second input: The aileron angle δ_a as shown on figure 1. The roll angle φ is also added to the output vector.

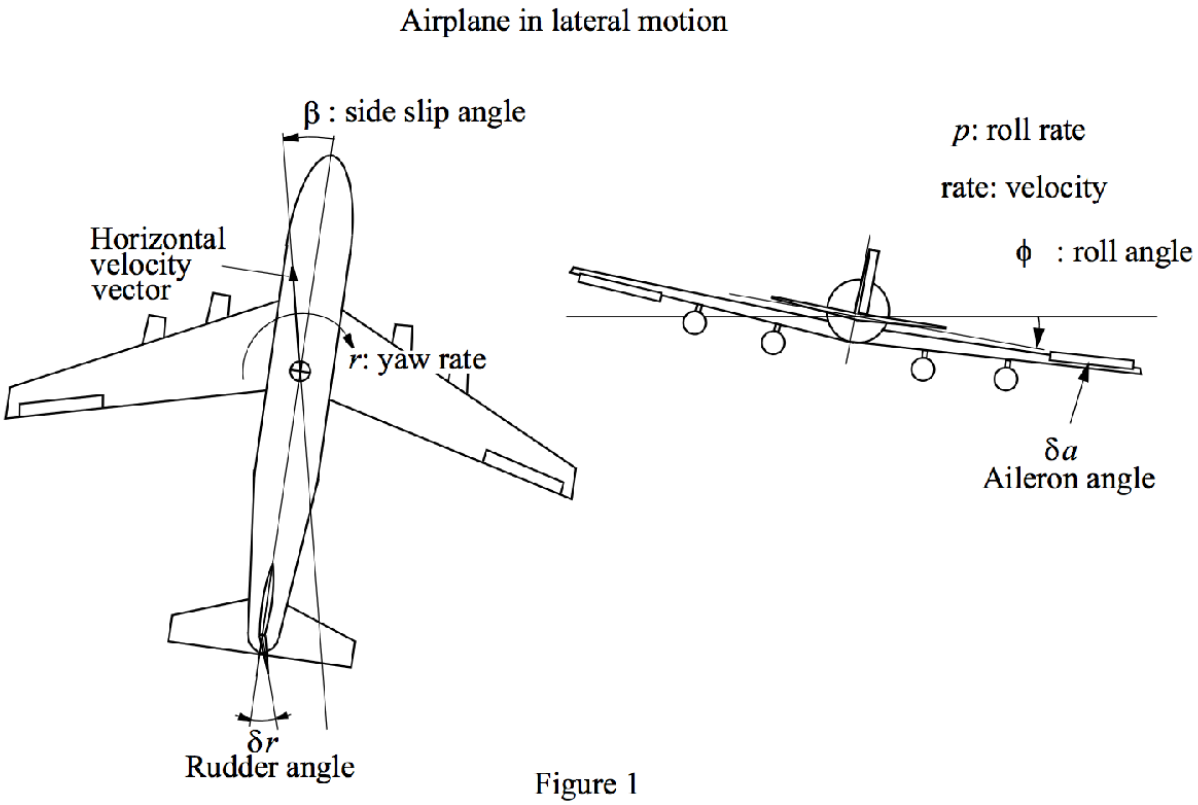


Figure 1

A linearized state space model of the lateral dynamics of a Boeing 747 in altitude 40000 ft and speed 774 ft/sec (850 km/h) is shown below:

$$\begin{pmatrix}\dot{\beta}\\ \dot{r}\\ \dot{p}\\ \dot{\varphi}\end{pmatrix} = \begin{pmatrix}-0.0558 & -0.9968 & 0.0802 & 0.0415\\ 0.5980 & -0.1150 & -0.0318 & 0\\ -3.0500 & 0.3880 & -0.4650 & 0\\ 0 & 0.0805 & 1 & 0\end{pmatrix}\begin{pmatrix}\beta\\ r\\ p\\ \varphi\end{pmatrix} + \begin{pmatrix}0.00729 & 10^{-5}\\ -0.47500 & 0.123\\ 0.15300 & 1.063\\ 0 & 0\end{pmatrix}\begin{pmatrix}\delta r\\ \delta a\end{pmatrix} \quad y = \begin{pmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\beta\\ r\\ p\\ \varphi\end{pmatrix}$$

where β is the side-slip angle (rad), r is the yaw rate (rad/s), p is the roll rate (rad/s), φ is the roll angle (rad), δ_r is the rudder deflection (rad) and δ_a is the aileron angle (rad).

```
format short g; clear all; close all; clc;

% Define system matrices:
A = [-0.0558 -0.9968  0.0802  0.0415;
```

```
      0.598  -0.115  -0.0318  0;
     -3.050   0.388  -0.4650  0;
      0       0.0805  1       0];
B = [0.00729   1e-5;
    -0.475     0.123;
     0.153     1.063;
     0         0];
C = [0 1 0 0;
     0 0 0 1];
D = zeros(2,2);

% Define state-space system:
sys_ss = ss(A,B,C,D)
```

```
sys_ss =

  A =
           x1       x2       x3       x4
   x1  -0.0558  -0.9968   0.0802   0.0415
   x2    0.598   -0.115  -0.0318        0
   x3    -3.05    0.388   -0.465        0
   x4        0   0.0805        1        0

  B =
           u1       u2
   x1  0.00729    1e-05
   x2   -0.475    0.123
   x3    0.153    1.063
   x4        0        0

  C =
       x1  x2  x3  x4
   y1   0   1   0   0
   y2   0   0   0   1

  D =
       u1  u2
   y1   0   0
   y2   0   0

Continuous-time state-space model.
```

Problem 1 Is the system controllable and observable?

Use the controllability and observability matrices $\mathbf{M}_c$ *and* $\mathbf{M}_o$ *(Matlab functions:* `ctrb` *and* `obsv`*). How would you test whether* $\mathbf{M}_c$ *has 4 linearly independent columns?*

Also use the test with the left and the right eigenvectors (first part of PBH-test, i.e. the criteria ***CC3*** *and* ***OC3****) and the diagonal test (****CC4*** *and* ***OC4****).*

Solution:

First the controllability of the system will be tested and then the observability of the system will be tested.

--- Controllability Theorem CC2:

To test for controllability we need to evaluate if the controllability matrix Mc has full row rank. The $\mathbf{M}_c$ matrix can be written from Eq. (3.230) in the textbook with $n = 4$ (dimension of the state space);

$$\mathbf{M}_c = \begin{pmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \mathbf{A}^3\mathbf{B} \end{pmatrix}$$

```
% Mc = [B A*B A^2*B A^3*B] is calculated by:
Mc = ctrb(A,B)
```

```
Mc =
      0.00729        1e-05      0.48534    -0.037354    -0.098535     0.058581     -0.40443    0.0049517
       -0.475        0.123     0.054119    -0.047942      0.29284   -0.0026226    -0.050301     0.025698
        0.153        1.063     -0.27768      -0.4466      -1.3302        0.303       1.0327     -0.32059
            0            0      0.11476       1.0729     -0.27332     -0.45046      -1.3066      0.30279
```

```
% Test if the controlabillity matrix has at least as many
% linearly independant columns as the number of states in the system:
if rank(Mc) >= size(A,1)
    disp('System is controllable!');
else
    disp('System is NOT controllable!');
end
```

```
System is controllable!
```

The rank of the $\mathbf{M}_c$ matrix is 4, so it has full row rank and the system is therefore controllable. We can also convince ourselves that the system is controllable by checking the validity of conditions **CC3** and **CC4**.

--- Controllability Theorem CC3 - Part 1:

First the left eigenvectors are computed with `eig()`as

```
[V,Dr,W] = eig(A); % or [W,D] = eig(A');
% V : Right eigenvectors, D : eigenvalues, W : Left eigenvectors

disp('Left eigenvectors:'); W
```

```
Left eigenvectors:
W =
    -0.088261 +    0.67683i    -0.088261 -    0.67683i     -0.47231 +          0i    -0.021083 +
      0.72641 +          0i      0.72641 +          0i     -0.84616 +          0i      0.97392 +
    -0.066988 -   0.032468i    -0.066988 +   0.032468i     -0.24439 +          0i      0.19129 +
    -0.029501 -  0.0048956i    -0.029501 +  0.0048956i     0.034837 +          0i      0.12022 +
```

If the system has no left eigenvectors such that $w_i^{\mathrm{T}}\mathbf{B} = 0$ (Eq. (3.247) in the textbook) then the system is controllable. If we apply this condition to our system we obtain:

$$\mathbf{W}^T\mathbf{B} = \begin{pmatrix} 0.3559 - 0.0000\,i & -0.0181 - 0.0345\,i \\ 0.3559 + 0.0000\,i & -0.0181 + 0.0345\,i \\ -0.3611 & 0.3639 \\ -0.4335 & 0.3231 \end{pmatrix}$$

```
disp('transpose(W)*B = ');
```

```
transpose(W)*B =
```

```
disp(W'*B);
```

```
      -0.35594 + 3.3539e-05i     0.018139 +   0.034507i
      -0.35594 - 3.3539e-05i     0.018139 -   0.034507i
       0.36109 +          0i     -0.36387 +          0i
       -0.4335 +          0i      0.32313 +          0i
```

therefore, since none of the elements are zero, the first part of **CC3** indicates that the system is controllable.

--- Controllability Theorem CC4:

The theorem holds for systems in diagonal form, so the first step is to find the modal matrix $\mathbf{M}$, which can be used to diagonalize the system at hand. The modal matrix is

$$\mathbf{M} = \begin{pmatrix} 0.1994 - 0.1063\,i & 0.1994 + 0.1063\,i & -0.0172 & 0.0067 \\ -0.0780 - 0.1333\,i & -0.0780 + 0.1333\,i & -0.0118 & 0.0404 \\ -0.0165 + 0.6668\,i & -0.0165 - 0.6668\,i & -0.4895 & -0.0105 \\ 0.6930 & 0.6930 & 0.8717 & 0.9991 \end{pmatrix}$$

```
% The modal matrix is the same as the normal (left) eigenvector matrix.
[M,Dr] = eig(A); M
```

```
M =
      0.19937 -    0.10627i      0.19937 +    0.10627i    -0.017178 +          0i    0.0067218 +
     -0.077977 -    0.13326i     -0.077977 +    0.13326i    -0.011828 +          0i     0.040422 +
     -0.016547 +    0.66677i     -0.016547 -    0.66677i     -0.48953 +          0i    -0.010525 +
      0.69301 +          0i      0.69301 +          0i      0.87174 +          0i       0.9991 +
```

The transformed $\mathbf{B}_t$ matrix is then

$$\mathbf{B}_t = \mathbf{M}^{-1}\mathbf{B}$$

```
Bt = M\B % equivalent to inv(M)*B
```

```
Bt =
      0.63546 -    0.89716i     -0.11936 -   0.015897i
      0.63546 +    0.89716i     -0.11936 +   0.015897i
       2.1477 + 1.7807e-16i      -2.1643 - 7.1086e-17i
      -2.7555 - 4.9356e-16i       2.0539 + 2.9207e-16i
```

It is clear that the $\mathbf{B}_t$ matrix does not have zero rows, hence the **CC4** test confirms that the system is controllable.

--- Observability Theorem OC2:

The observability matrix $\mathbf{M}_o$ is written as in Eq. (3.301) in the textbook with $n = 4$.

$$\mathbf{M}_o = \begin{pmatrix} 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0.5980 & -0.1150 & -0.0318 & 0 \\ 0 & 0.0805 & 1.0000 & 0 \\ -0.0051 & -0.5952 & 0.0664 & 0.0248 \\ -3.0019 & 0.3787 & -0.4676 & 0 \\ -0.5582 & 0.1013 & 0.0125 & -0.0002 \\ 1.8200 & 2.7673 & -0.0354 & -0.1246 \end{pmatrix}$$

The rank of the observability matrix is 4, and so it has full column rank and the system is therefore observable. In MatLab this can be computed as follows:

```
% Calculate observability matrix:
Mo = obsv(A,C)
```

```
Mo =
            0            1            0            0
            0            0            0            1
        0.598       -0.115      -0.0318            0
            0       0.0805            1            0
   -0.0051484      -0.5952     0.066404     0.024817
      -3.0019      0.37874     -0.46756            0
     -0.55817      0.10134     0.012454  -0.00021366
         1.82       2.7673    -0.035378     -0.12458
```

```
% Test if the observability matrix has full column rank,
% and if the system is therefore observable.
if rank(Mo) >= size(A,1)
    disp('System is observable!');
else
    disp('System is NOT observable!');
end
```

```
System is observable!
```

--- Observability Theorem OC3:

The right eigenvectors are calculated with with `eig()`. Note that the difference from the left eigenvectors, is the transpose of the A matrix.

```
[V, Dr] = eig(A); V
```

```
V =
      0.19937 -    0.10627i      0.19937 +    0.10627i    -0.017178 +          0i    0.0067218 +
     -0.077977 -    0.13326i    -0.077977 +    0.13326i    -0.011828 +          0i     0.040422 +
     -0.016547 +    0.66677i    -0.016547 -    0.66677i     -0.48953 +          0i    -0.010525 +
       0.69301 +          0i      0.69301 +          0i      0.87174 +          0i       0.9991 +
```

The Matlab code returns gives the matrix V whose columns are the right eigenvectors. If the system has no right eigenvectors such that $\mathbf{C}\mathbf{v}_i = 0$ (Eq. (3.313) in the textbook) then the system is observable. If we apply this condition to our system we obtain

$$\mathbf{CV} = \begin{pmatrix} -0.0780 + 0.1333\,i & -0.0780 - 0.1333\,i & -0.0118 & 0.0404 \\ 0.6930 & 0.6930 & 0.8717 & 0.9991 \end{pmatrix}$$

therefore, since none of the columns are zero the first part of **OC3** indicates that the system is observable.

Run the code to verify:

```
disp('C*V =')
```

```
C*V =
```

```
disp(C*V)
```

```
     -0.077977 -    0.13326i    -0.077977 +    0.13326i    -0.011828 +          0i     0.040422 +
       0.69301 +          0i      0.69301 +          0i      0.87174 +          0i       0.9991 +
```

--- Observability Theorem OC4:

By using the modal matrix $\mathbf{M}$ found above, the transformed output matrix $\mathbf{C}_t$ is obtained

$$\mathbf{C}_t = \mathbf{CM} = \begin{pmatrix} -0.0780 - 0.1333\,i & -0.0780 + 0.1333\,i & -0.0118 & 0.0404 \\ 0.6930 & 0.6930 & 0.8717 & 0.9991 \end{pmatrix}$$

It is clear that there are no zero columns in the $\mathbf{C}_t$ matrix, hence the **OC4** test confirms the observability of the system.

```
disp('C*M =')
```

```
C*M =
```

```
disp(C*M)
```

```
     -0.077977 -    0.13326i    -0.077977 +    0.13326i    -0.011828 +          0i     0.040422 +
       0.69301 +          0i      0.69301 +          0i      0.87174 +          0i       0.9991 +
```

Problem 2 Find the 4 transfer functions and determine the 4 sets of poles and zeros.

Solution:

To find the four transfer functions the ss2tf() function is used. When making transfer functions from a state space model with more than one input or output, the desired input/output needs to be specified.

$$G_{r\delta_r}(s) = \frac{-0.475\,(s + 0.4981\,)(s^2 + 0.02379\,s + 0.2381\,)}{(s + 0.5627\,)(s + 0.007278\,)(s^2 + 0.06587\,s + 0.8972\,)}$$

$$G_{\varphi\,\delta_r}(s) = \frac{2.6645 * 10^{-15}(s + 4.307 * 10^{13})(s - 4.44)(s + 2.694\,)}{(s + 0.5627\,)(s + 0.007278\,)(s^2 + 0.06587\,s + 0.8972\,)}$$

$$G_{r\delta_a}(s) = \frac{0.123\,(s + 0.4484\,)(s^2 - 0.2024\,s + 0.7605\,)}{(s + 0.5627\,)(s + 0.007278\,)(s^2 + 0.06587\,s + 0.8972\,)}$$

$$G_{\varphi\,\delta_a}(s) = \frac{1.4433 * 10^{-15}(s + 7.434 * 10^{14})(s^2 + 0.2159\,s + 0.9541\,)}{(s + 0.5627\,)(s + 0.007278\,)(s^2 + 0.06587\,s + 0.8972\,)}$$

The zeros and poles can be seen directly from the transfer functions when the tf's are factorized like above. In MatLab:

```
% Create state-space object
sys_ss = ss(A,B,C,D);
sys_ss.u = {'\delta_r','\delta_a'}; % Name inputs
sys_ss.y = {'r','\phi'}; % Name outputs

% Compute the transfer function matrix
sys_tf = zpk(minreal(tf(sys_ss)))
```

```
sys_tf =

  From input "\delta_r" to output...
          -0.475 (s+0.4981) (s^2 + 0.02379s + 0.2381)
   r:  -----------------------------------------------
       (s+0.5627) (s+0.007278) (s^2 + 0.06587s + 0.8972)

                     0.11476 (s-4.44) (s+2.694)
   \phi:  -------------------------------------------------
          (s+0.5627) (s+0.007278) (s^2 + 0.06587s + 0.8972)

  From input "\delta_a" to output...
           0.123 (s+0.4484) (s^2 - 0.2024s + 0.7605)
   r:  -----------------------------------------------
       (s+0.5627) (s+0.007278) (s^2 + 0.06587s + 0.8972)

                   1.0729 (s^2 + 0.2159s + 0.9541)
   \phi:  -------------------------------------------------
          (s+0.5627) (s+0.007278) (s^2 + 0.06587s + 0.8972)

Continuous-time zero/pole/gain model.
```

```
% The poles are the same for all four transfer functions
% as poles are determined only by the A matrix
disp('Poles (for all tf):'); disp(pole(sys_tf(1,1)));
```

```
Poles (for all tf):
    -0.032935 +    0.94665i
    -0.032935 -    0.94665i
```

```
     -0.56265 +          0i
    -0.007278 +          0i
```

```
% Compute the zeros for all four transfer functions:
for ii = 1:size(sys_tf,2)
    for jj = 1:size(sys_tf,1)
        disp(['Zeros from ' sys_tf.u{ii} ' to ' sys_tf.y{jj} ':'])
        disp(zero(sys_tf(jj,ii)));
    end
end
```

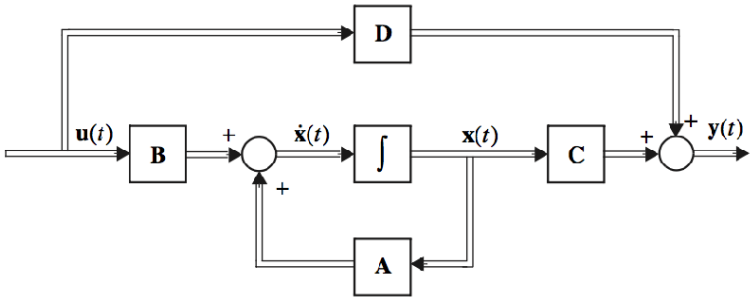
```
Zeros from \delta_r to r:
     -0.49808 +          0i
    -0.011893 +    0.48779i
    -0.011893 -    0.48779i
Zeros from \delta_r to \phi:
       4.4398
       -2.694
Zeros from \delta_a to r:
      0.10121 +    0.86618i
      0.10121 -    0.86618i
     -0.44845 +          0i
Zeros from \delta_a to \phi:
     -0.10797 +    0.97082i
     -0.10797 -    0.97082i
```

```
```

Note that all the poles have negative real parts, whereas some of the zeros have positive real parts.

Problem 3 Set up a detailed Simulink model of the airplane. Base the model on figure 2.3 in the textbook. Note: One can use the MatrixGain-block as well as the usual Gain-block, but be sure that it is set up for matrix-multiplication.

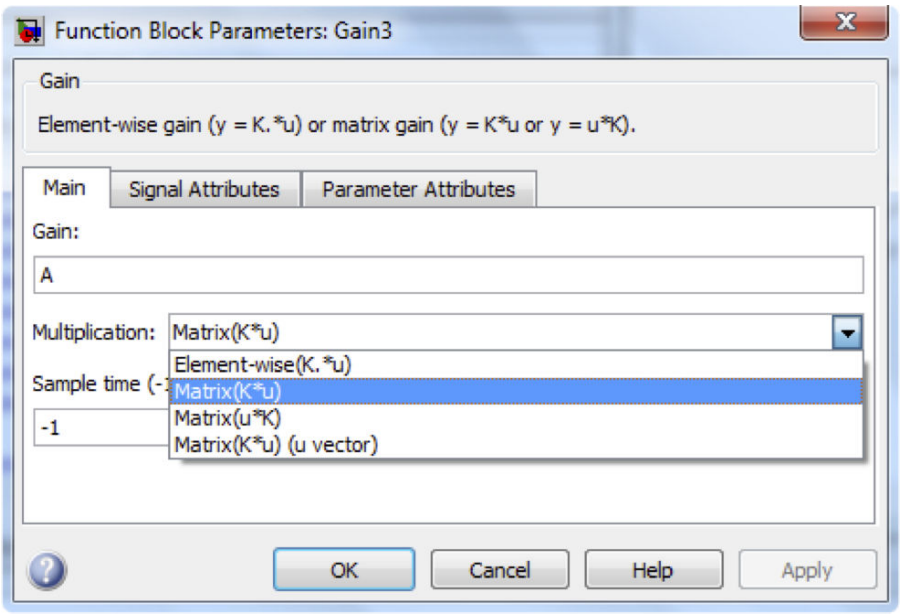
Fig. 2.3 Block diagram of a general continuous linear state space model



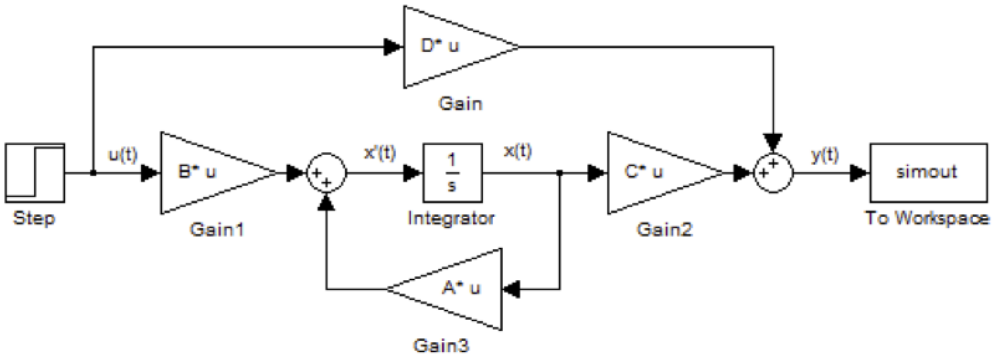
Solution:

--- Simulink model

The model of the system is entered in Simulink using the Gain-block. Remember to change the settings in the block to matrix gain as shown in Figure below.



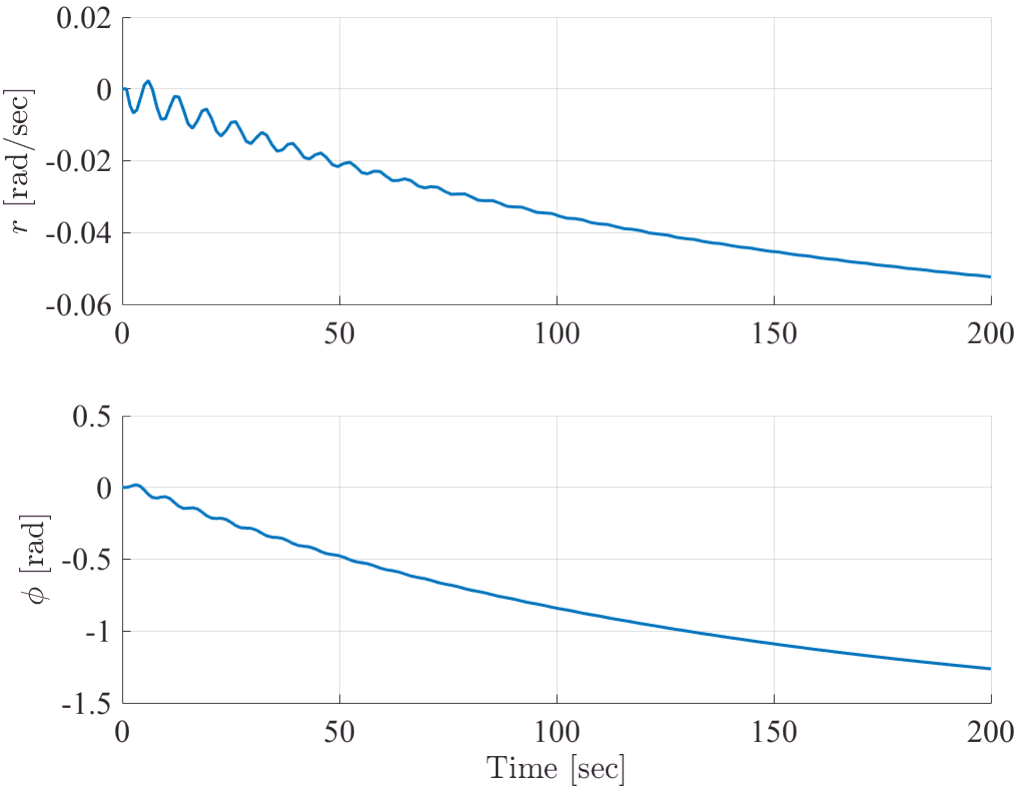
The block-diagram is entered as Fig. 2.3 in the textbook. The Simulink implementation of the block-diagram looks can be seen below




```
% To open simulink model, uncomment the following line and run code
% open('M10_Q3_Lat747.slx');
```

```
% Simulink model
u = [1*pi/180; 1*pi/180];
sim('M10_Q3_Lat747'); % Run Simulation

figure, h1 = subplot(2,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
plot(simout.time, simout.signals.values(:,1),'LineWidth',1.5)
ylabel('$$r$$ [rad/
sec]','FontName','times','FontSize',16,'Interpreter','latex');
h2 = subplot(2,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
```

```
plot(simout.time, simout.signals.values(:,2),'LineWidth',1.5)
ylabel('$$\phi$$
[rad]','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex');
```

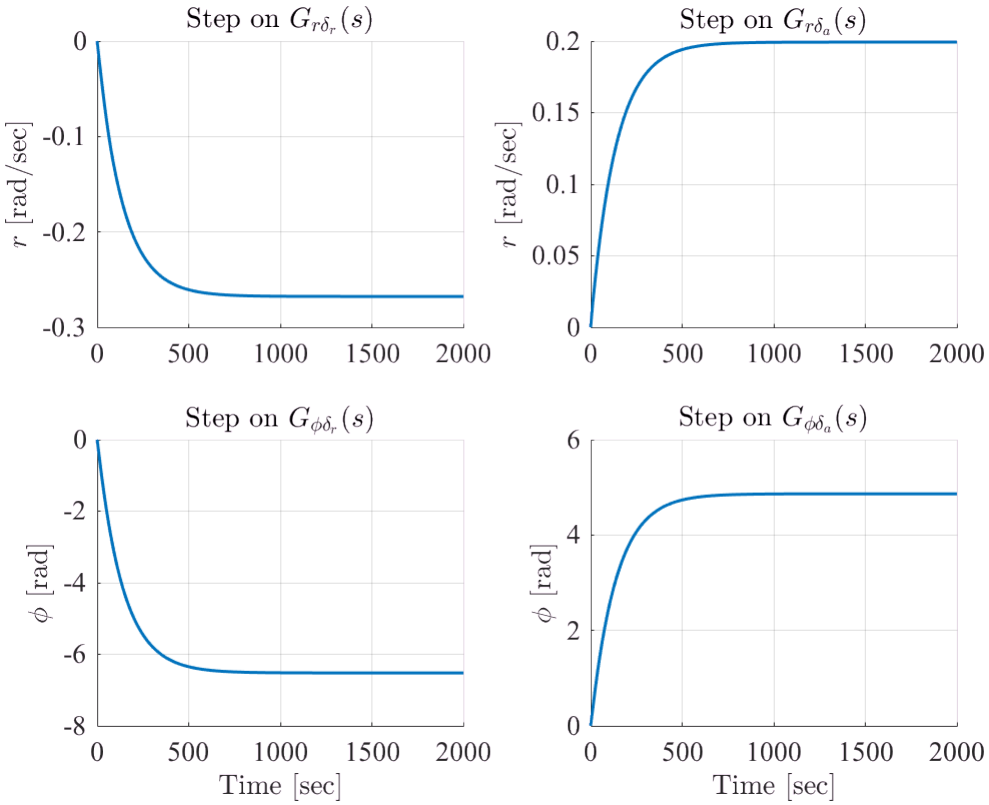


Problem 4 Find the 4 step responses (use the angle 1° for both inputs) and plot them in a 2 by 2 array using Matlab's subplot-function.

Solution:

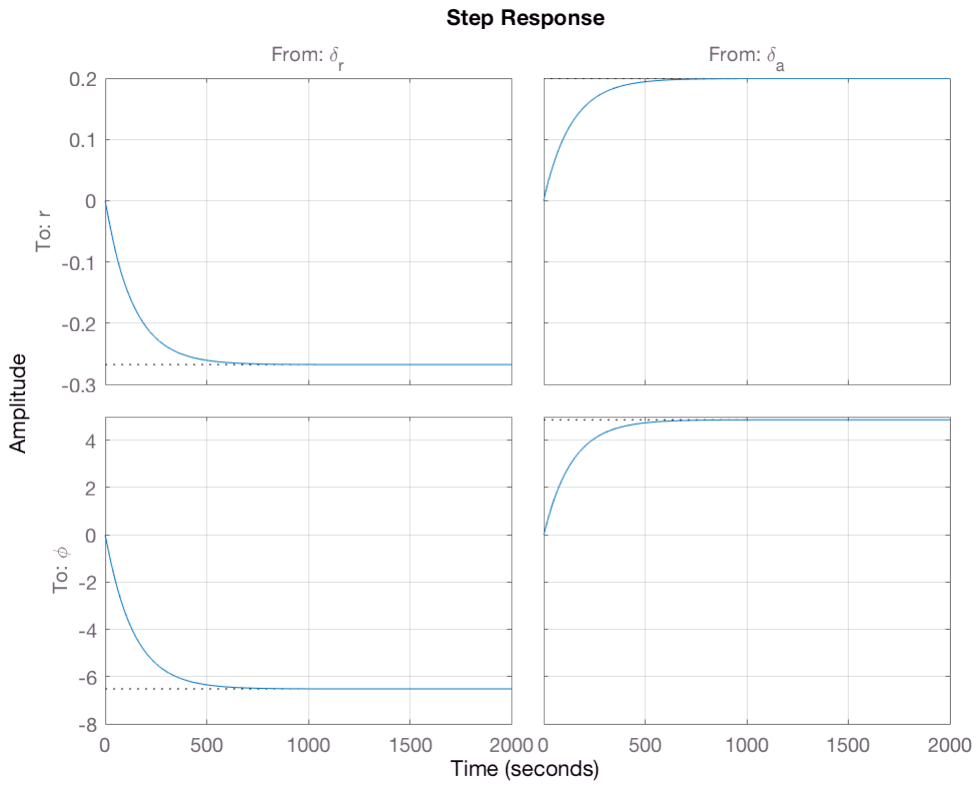
```
%% Q4 - Stepresponses
% Four stepresponses
tfinal = 2000;

figure, h3 = subplot(2,2,1); set(h3,'FontName','times','FontSize',14)
hold on, grid on
[y1,t1] = step(sys_tf(1,1)*2*pi/360,tfinal);
plot(t1,y1,'LineWidth',1.5)
title('Step on $$G_{r \delta_r}(s)$
$','FontName','times','FontSize',14,'Interpreter','latex');
ylabel('$$r$$ [rad/
sec]','FontName','times','FontSize',14,'Interpreter','latex');

h4 = subplot(2,2,3); set(h4,'FontName','times','FontSize',14)
```

```
hold on, grid on
[y2,t2] = step(sys_tf(2,1)*2*pi/360,tfinal);
plot(t2,y2,'LineWidth',1.5)
title('Step on $$G_{\phi \delta_r}(s)$
$','FontName','times','FontSize',14,'Interpreter','latex');
ylabel('$$\phi$$
[rad]','FontName','times','FontSize',14,'Interpreter','latex');
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex');

h5 = subplot(2,2,2); set(h5,'FontName','times','FontSize',14)
hold on, grid on
[y3,t3] = step(sys_tf(1,2)*2*pi/360,tfinal);
plot(t3,y3,'LineWidth',1.5)
title('Step on $$G_{r \delta_a}(s)$
$','FontName','times','FontSize',14,'Interpreter','latex');
ylabel('$$r$$ [rad/
sec]','FontName','times','FontSize',14,'Interpreter','latex');

h6 = subplot(2,2,4); set(h6,'FontName','times','FontSize',14)
hold on, grid on
[y4,t4] = step(sys_tf(2,2)*2*pi/360,tfinal);
plot(t4,y4,'LineWidth',1.5)
title('Step on $$G_{\phi \delta_a}(s)$
$','FontName','times','FontSize',14,'Interpreter','latex');
ylabel('$$\phi$$
[rad]','FontName','times','FontSize',14,'Interpreter','latex');
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex');
```



It was asked specifically that subplot was used, but in fact many plotting functions in matlab are capable of plotting matrix functions. A very similar plot can thus be produced simply by passing the transfer function matrix directly to step:

```
figure
step(sys_tf*2*pi/360, 2000), grid
```



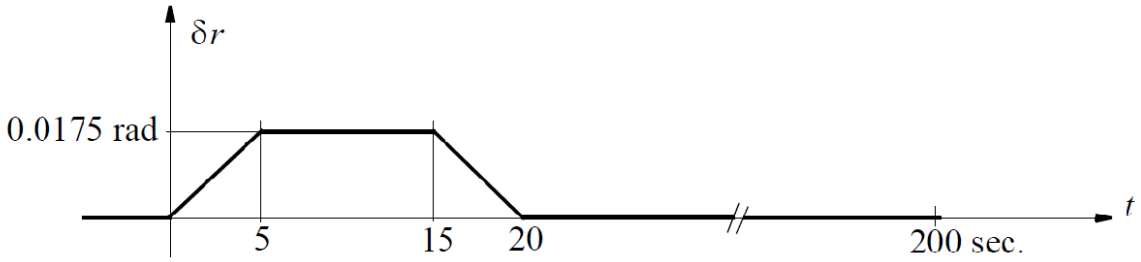
Problem 5 Last, simulate the response of the linear system by using the function on the figure below at both inputs, but one at a time.

Is it possible to change the airplane's heading by manipulating the ailerons only?

figure

step(sys_tf*2*pi/360, 2000), gridfigure

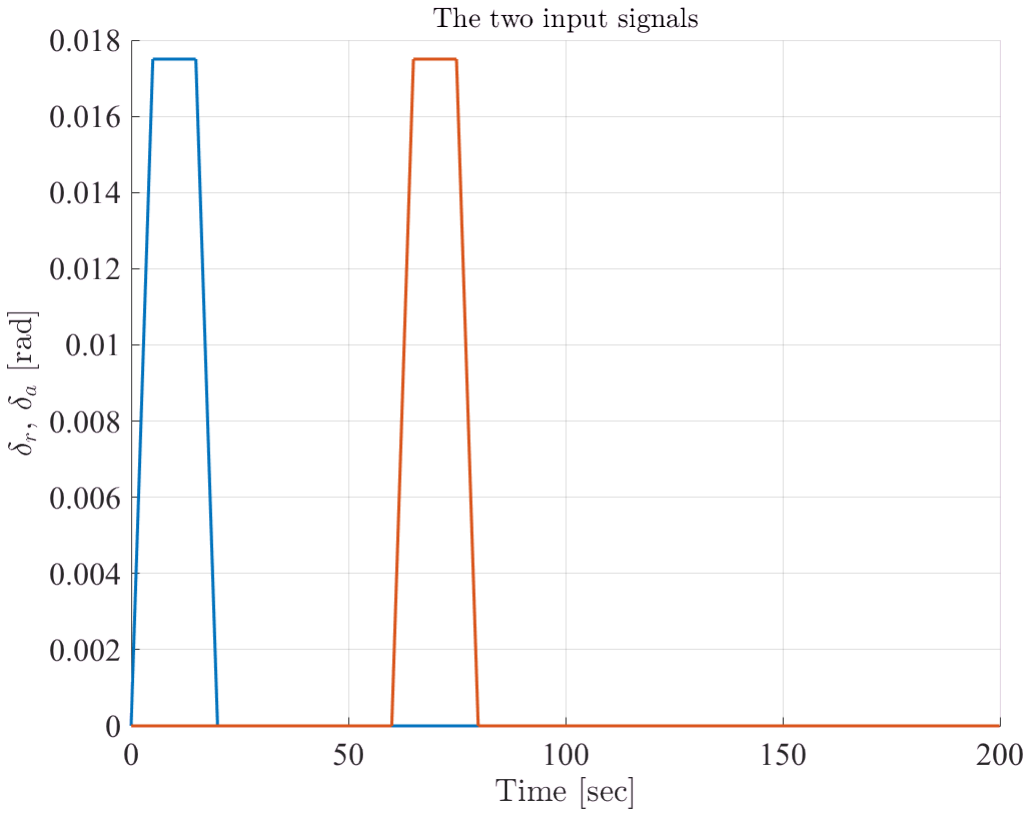
step(sys_tf*2*pi/360, 2000), grid



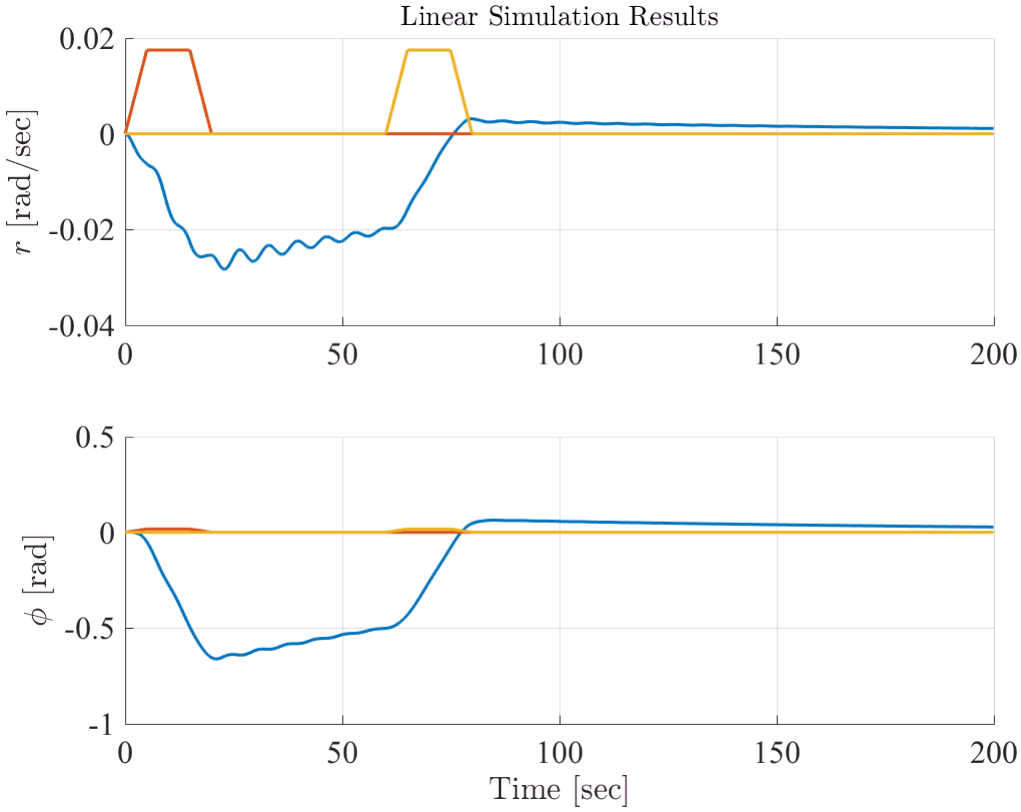
Solution:

The two ramp signals for δ_r and δ_a are generated in the same way as in Group Work Module 4+5. The two signals are started at different times, as it can be seen from the **"Two input signals"** plot. When these signals are used as input for the system the response looks like the graphs in the **"Linear Simulation Results"** plot. As it can be seen from the responses, heading alteration is achievable by actuating both the rudder and the ailerons.

```
% The step functions
t = 0:.1:199.9;
bias = 600;
ut1 = (0:50) * 0.0175/50;
ut2 = ones(1,98) * 0.0175;
ut3 = (0:50) * (-0.0175/50) + 0.0175;
ut4 = zeros(1,1800);
ut5 = zeros(1,bias);
ut6 = zeros(1,1800-bias);
uttot1 = [ut1 ut2 ut3 ut4];
uttot2 = [ut5 ut1 ut2 ut3 ut6];
u = [uttot1 ; uttot2];

% Check on the steps
figure, h7 = axes; set(h7,'FontName','times','FontSize',16)
hold on, grid on
plot(t,u,'LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex');
ylabel('$$\delta_r, \, \delta_a$$
[rad]','FontName','times','FontSize',16,'Interpreter','latex');
title('The two input
signals','FontName','times','FontSize',14,'Interpreter','latex');
```




```
% Simulate the step
y = lsim(A,B,C,D,u,t,[0 0 0 0]');
figure, h8 = subplot(2,1,1); set(h8,'FontName','times','FontSize',16)
hold on, grid on
plot(t,y(:,1),t,u,'LineWidth',1.5)
title('Linear Simulation
Results','FontName','times','FontSize',14,'Interpreter','latex');
ylabel('$$r$$ [rad/
sec]','FontName','times','FontSize',16,'Interpreter','latex');
h9 = subplot(2,1,2); set(h9,'FontName','times','FontSize',16)
hold on, grid on
plot(t,y(:,2),t,u,'LineWidth',1.5)
ylabel('$$\phi$$
[rad]','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex');
```


Linear Simulation Results
0.02
0
-0.02
-0.04
r [rad/sec]
0
50
100
150
200
0.5
0
-0.5
-1
ϕ [rad]
0
50
100
150
200
Time [sec]

Linear Control Design II - Group Work Problem Module 12 Solution

Jakob Mahler Hansen
Department of Electrical Engineering
September, 2011
Thomas Thougaard Paulsen
September, 2017

plot(t,y(:,1),t,u,'LineWidth',1.5)

Description

This group work module deals with the design of a full state feedback controller for the model of the aircraft analysed in Group Work Module 7 (the SISO model).

```
% Define system matrices:
A = [-0.0558 -0.9968  0.0802  0.0415;
      0.598  -0.115  -0.0318  0;
     -3.050   0.388  -0.4650  0;
      0       0.0805  1       0]
```

```
A =
   -0.0558   -0.9968    0.0802    0.0415
    0.5980   -0.1150   -0.0318         0
   -3.0500    0.3880   -0.4650         0
         0    0.0805    1.0000         0
```

```
B = [0.00729; -0.475; 0.153; 0]
```

```
B =
    0.0073
   -0.4750
    0.1530
         0
```

```
C = [0 1 0 0]
```

```
C =
     0     1     0     0
```

```
D = 0;
```

Problem 1

Transform the continuous time model given in Module 7 into the controller canonical form.

Solution:

First of all, the characteristic polynomial associated to the **A** matrix is found with the *poly()* function as.

$$P_{ch,\mathbf{A}} = det(\lambda\mathbf{I} - \mathbf{A}) = \lambda^4 + 0.6358\,\lambda^3 + 0.9389\,\lambda^2 + 0.5116\,\lambda + 0.0037$$

Then the **P** matrix can be calculated as in Eq. (3.353) on page 160 of the textbook.

Since the SISO system is controllable, it can be shown that all the **p** vectors are linearly independent, and therefore the inverse of **P** exists.

The **P** matrix is therefore given by

$$\mathbf{P} = \begin{bmatrix} -0.0077 & 0.2169 & 0.4900 & 0.0073 \\ -0.0563 & -0.1187 & -0.2479 & -0.4750 \\ 0.0045 & -1.3631 & -0.1804 & 0.1530 \\ -1.3726 & -0.2004 & 0.1148 & 0 \end{bmatrix}$$

The system matrices in controller canonical form can be calculated from (3.358), (3.361), (3.362) and (3.363) in the textbook by using the **P** matrix.

$$\mathbf{A}_{CC} = \mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.0037 & -0.5116 & -0.9389 & -0.6358 \end{bmatrix}$$

$$\mathbf{B}_{CC} = \mathbf{P}^{-1}\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{C}_{CC} = \mathbf{C}\mathbf{P} = [-0.0563 \quad -0.1187 \quad -0.2479 \quad -0.4750\,]$$

$$\mathbf{D}_{CC} = \mathbf{D} = [0]$$

```
% In Matlab, the useage of the poly() function is shown
chapo = poly(A)

% Here the P matrix is defined from determining the p vectors
p1 = B;
p2 = A*p1 + chapo(2)*p1;
p3 = A*p2 + chapo(3)*p1;
p4 = A*p3 + chapo(4)*p1;
P = [p4 p3 p2 p1]

% By using the P matrix the system can be written in the controller
% canonical form
Acc = P^(-1)*A*P
```

```
Acc =
   -0.0000    1.0000   -0.0000   -0.0000
    0.0000    0.0000    1.0000    0.0000
   -0.0000    0.0000    0.0000    1.0000
   -0.0037   -0.5116   -0.9389   -0.6358
```

```
Bcc = P^(-1)*B
```

```
Bcc =
    0.0000
   -0.0000
    0.0000
    1.0000
```

```
Ccc = C*P
```

```
Ccc =
   -0.0563   -0.1187   -0.2479   -0.4750
```

```
Dcc = D
```

```
Dcc = 0
```

Problem 2

It is desired to assign the eigenvalues through state feedback such that the damping ratio for the least damped eigenvalues is increased to approximately 0.7, and such that natural frequencies and time constants are approximately unchanged. A reasonable assignment could be:

$$\lambda = \begin{cases} -0.01 \\ -0.6 \\ -0.5 \pm 0.5\,j \end{cases}$$

Find the gain matrix **K** for the controller

$$u = -\mathbf{K}\mathbf{x} + r$$

by applying the method from Section 4.3.1 in the textbook. Check that the eigenvalues of the closed-loop system have been assigned as desired.

Solution:

The gain matrix **K** for the controller can be found by considering the closed loop characteristic polynomial. Including the controller into the system the close-loop system matrix becomes

$$\mathbf{A}_{\mathbf{K}CC} = \mathbf{A}_{CC} - \mathbf{B}_{CC}\mathbf{K}_{CC}$$

and its characteristic polynomial is

$$P_{ch,\mathbf{A},\mathbf{K}\,CC} = \prod_{i=1}^{n} (\lambda - \lambda_{cl,i}) = \lambda^4 + 1.6100\,\lambda^3 + 1.1160\,\lambda^2 + 0.3110\,\lambda + 0.0030$$

where $\lambda_{cl,i}$ is the i'th desired eigenvalue.

The elements in the feedback gain matrix are found as the difference between the coefficients of the open and closed loop characteristic polynomials, that is

$$\mathbf{K}_{CC} = \begin{bmatrix} \alpha_0 - a_0 & \alpha_1 - a_1 & \alpha_2 - a_2 & \alpha_3 - a_3 \end{bmatrix}$$

$$\mathbf{K}_{CC} = [-0.0007 \quad -0.2006 \quad 0.1771 \quad 0.9742]$$

where α_i and a_i are the coefficients of the i'th order part of the closed and the open loop characteristic polynomials, respectively.

To get the gain matrix for the original system the $\mathbf{K}_{CC}$ matrix is transformed from canonical form back to the original system coordinates by using the inverse of **P**

$$\mathbf{K}_1 = \mathbf{K}_{CC}\mathbf{P}^{-1}$$

$$\mathbf{K}_1 = [-0.589 \quad -1.9909 \quad 0.2145 \quad 0.0862]$$

The controller $\mathbf{K}_1$ is inserted in the negative feedback loop, and it changes the input to the system as

$$u = -\mathbf{K}_1\mathbf{x} + r$$

To check that the desired eigenvalues have been assigned correctly the eigenvalues of the closed loop system are calculated. This is achieved with the *eig()* function

$$\lambda_{cl} = \text{eig}(\mathbf{A} - \mathbf{B}\mathbf{K}_1) = \begin{cases} -0.01 \\ -0.6 \\ -0.5 - 0.5\,j \\ -0.5 + 0.5\,j \end{cases}$$

The desired eigenvalues are obtained: the system damping is now larger, leading to smaller oscillations.

```
% In Matlab, the polynomium leading to the desired eigenvalues is
% calculated from the function poly()
alpha = poly([-0.01 -0.6 -0.5+0.5*i -0.5-0.5*i]);
% Controller gains in controller canonical form
Kcc = fliplr(alpha-chapo);
% Controller gains for the states
K = Kcc(1:4)*P^(-1)
```

```
K =
   -0.5869   -1.9909    0.2145    0.0862
```

```
% Checking the closed loop eigenvalues
eig(A-B*K)
```

```
ans =
  -0.0100 + 0.0000i
  -0.5000 + 0.5000i
  -0.5000 - 0.5000i
  -0.6000 + 0.0000i
```

Problem 3

Add the controller to the Simulink open-loop model and carry out a simulation with the function shown on Figure 1 as input to the closed-loop system. Plot the states, the output and the rudder angle.

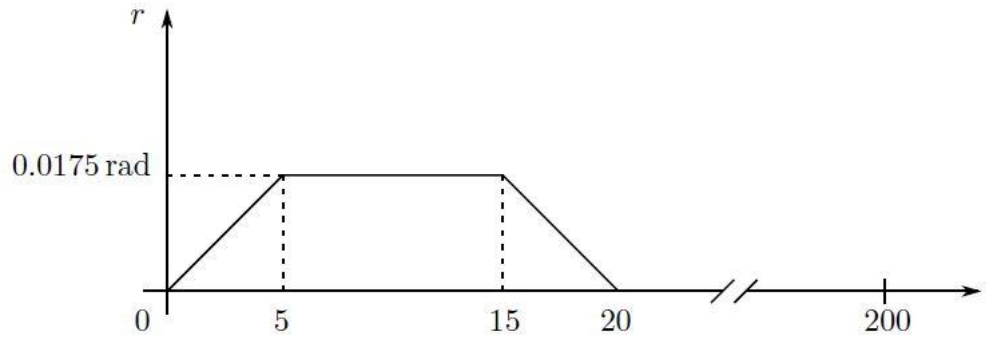


Figure 1 : Input signal to the closed loop system .

Solution:

The controller is placed in the negative feedback loop from the states to a summation point at the input of the system. See Figure 2 for the Simulink implementation.

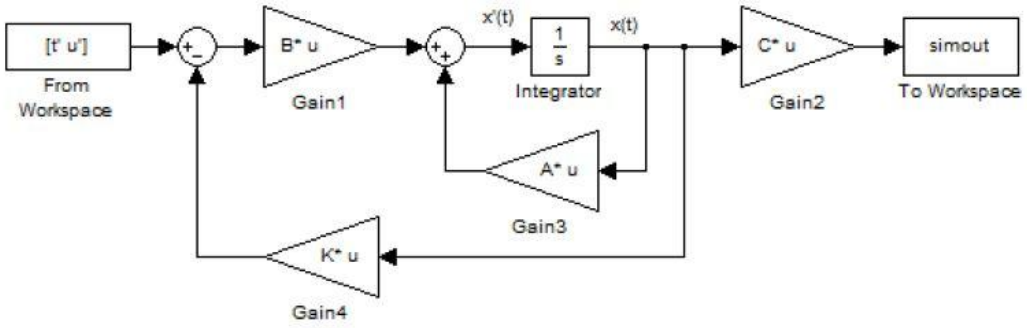


Figure 2 : Block diagram with the designed controller $\mathbf{K}_1$.

The response of the system is tested with and without the controller to compare its characteristics.

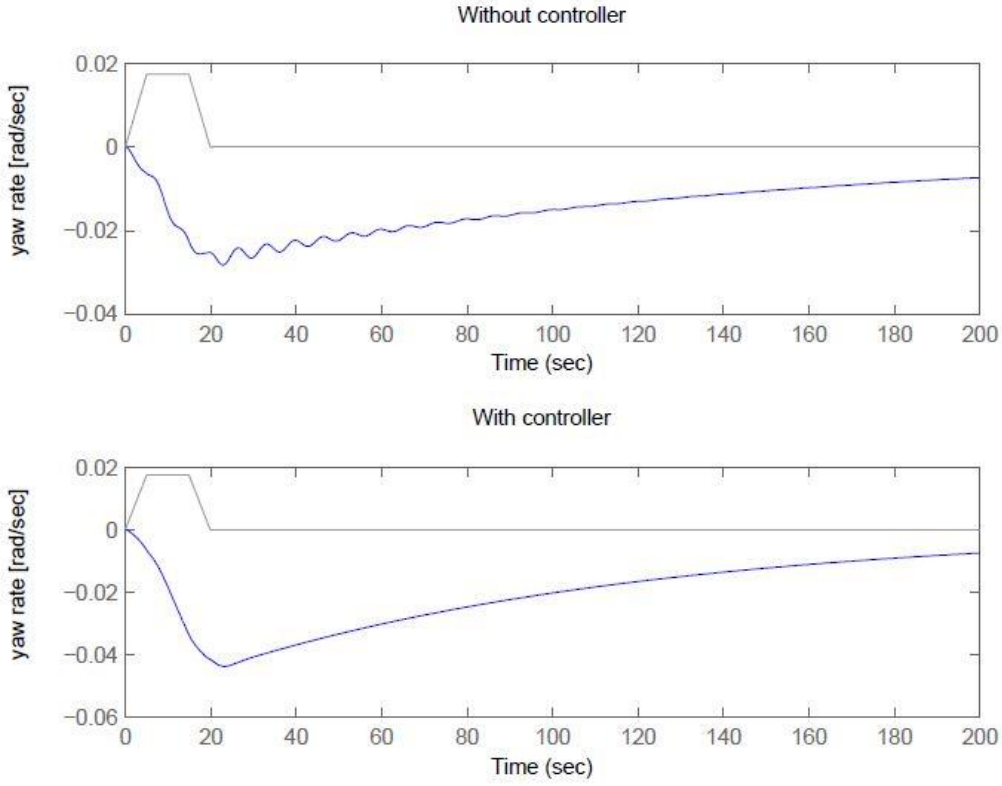
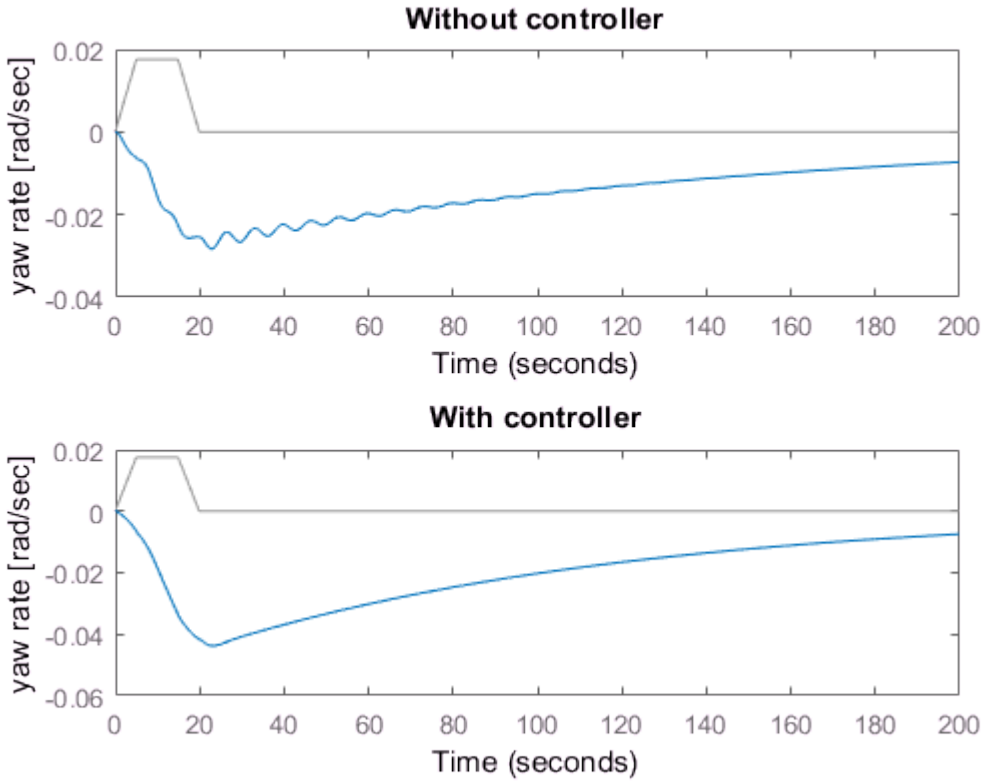


Figure 3 : Response of the system without (top) and with (bottom) controller $\mathbf{K}_1$.

Figure 3 shows that the controlled response is less oscillatory, giving more pleasant flight conditions. On the other hand, the magnitude of the response is almost doubled.

```
% Simulation of state responses
t = 0:.1:199.9; % Time vector
ut1 = (0:50) * 0.0175/50; % Rising input
ut2 = ones(1,98) * 0.0175; % Constant input
ut3 = (0:50) * (-0.0175/50) + 0.0175; % Decending input
ut4 = zeros(1,1800); % Zero input
u = [ut1 ut2 ut3 ut4];

figure(1)
subplot(2,1,1), lsim(A,B,C,D,u,t,[0 0 0 0]');
title('Without controller');
ylabel('yaw rate [rad/sec]');
subplot(2,1,2), lsim(A-B*K,B,C,D,u,t,[0 0 0 0]');
title('With controller');
ylabel('yaw rate [rad/sec]');
```



Problem 4

Repeat Problems 2 and 3 with the largest time constant decreased by a factor 50 and by applying Ackermann's formula (the Matlab's function is called *acker()*). The Ackermann's formula has to be applied to the original model.

Solution:

The largest time constant is desired to be decreased by a factor 50, to obtain a faster system's response. Hence, the new desired eigenvalues of the system are

$$\lambda = \begin{cases} -0.01 \cdot 50 \\ -0.6 \\ -0.5 \pm 0.5\,j \end{cases}$$

The new pole placement is carried out with the Matlab *acker()* function. The function calculates the feedback gain matrix $\mathbf{K}_2$, such that the SISO system with the feedback law

$$u = -\mathbf{K}_2\mathbf{x}$$

has closed loop poles at the desired positions. The gain matrix is

$$\mathbf{K}_2 = [0.4474 \quad -3.0480 \quad 0.0860 \quad 0.0163\,]$$

With this new controller gain plugged into the system the maneuver is accomplished in much less time, as shown in Figure 4.

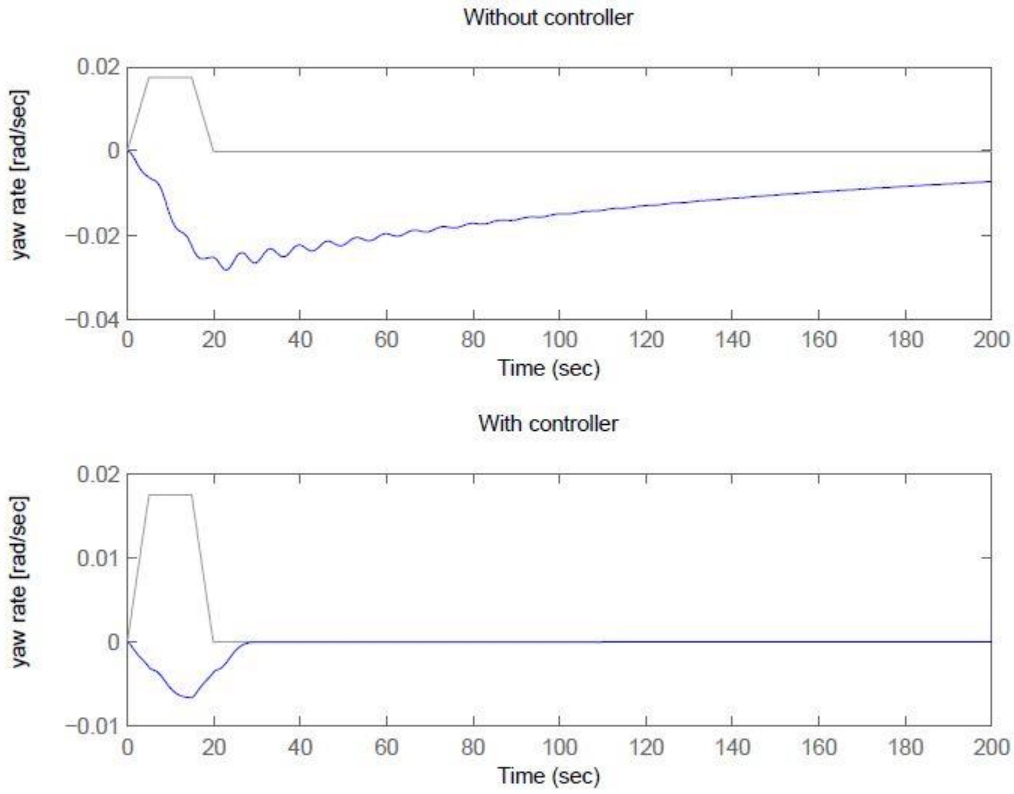


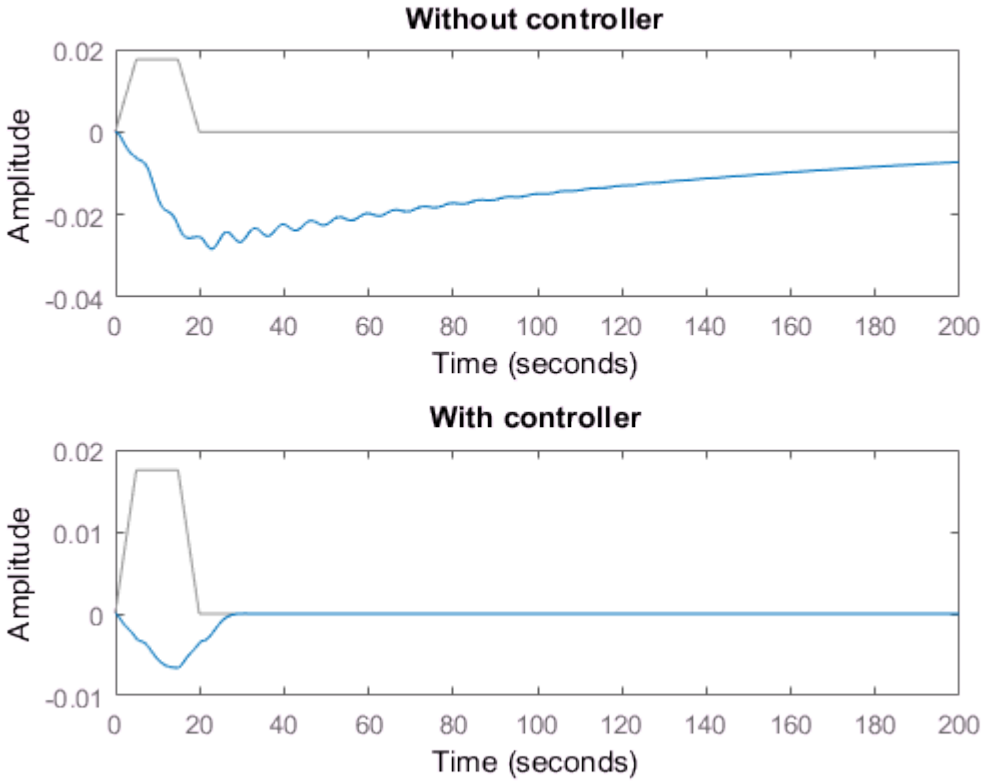
Figure 4 : Response of the system without (top) and with (bottom) controller $\mathbf{K}_2$.

Note that the magnitude of the response is now smaller than for the nonlinear model.

```
eigClose = [-0.01*50 -0.6 -0.5+0.5*i -0.5-0.5*i];
KQ4 = acker(A,B,eigClose)
```

```
KQ4 =
    0.4474   -3.0480    0.0860    0.0163
```

```
figure(2)
subplot(2,1,1), lsim(A,B,C,D,u,t,[0 0 0 0]')
title('Without controller');
subplot(2,1,2),lsim(A-B*KQ4,B,C,D,u,t,[0 0 0 0]')
title('With controller');
```



Problem 5

Repeat Problem 4 for a discrete time controller. Do not forget to convert the eigenvalues to the discrete domain. The discrete time controller must be tested together with the continuous time plant. To enable interaction between the continuous and discrete time parts of the control loop do not forget to insert digital-to-analogue (DAC) and analogue-to-digital (ADC) converters into the system. The DAC is usually a zero-order hold, and such a block is available in Simulink's discrete library. The ADC is implemented in Simulink as a discrete transfer function (same library) with 1 as numerator and denominator, as shown in Figure 5. All discrete blocks use the parameter *Sample Time*, which has to be chosen in a suitable manner.

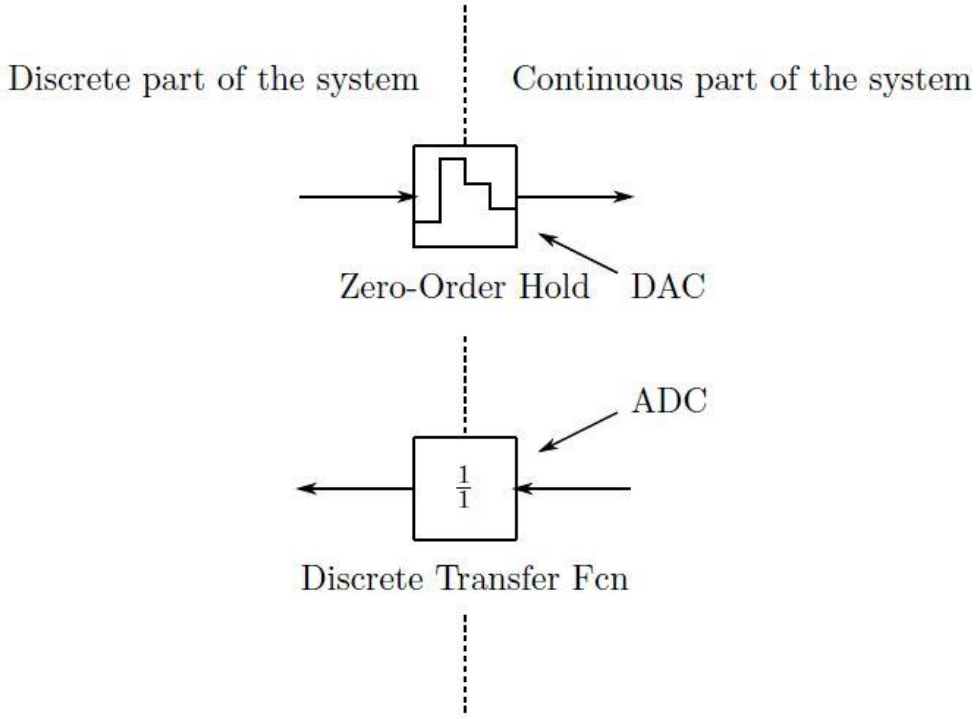


Figure 5 : Simulink discretization blocks .

Solution:

To discretize the model a DAC (Digital to Analog Converter in the form of a Zero-order hold) and a ADC (Analog to Digital Converter in the form of a Discrete Transfer function) are inserted as seen on Figure 6.

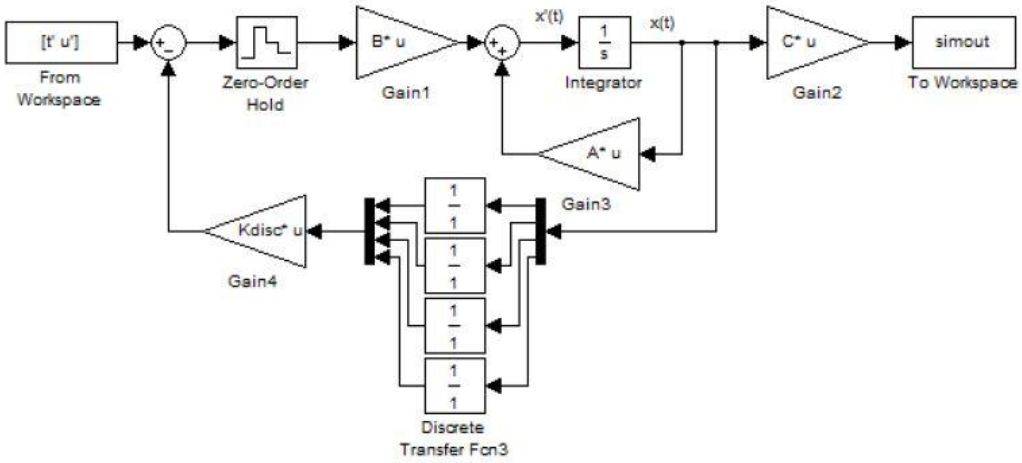


Figure 6 : Block diagram of the discrete system with controller .

The sample time of the system needs to be speciffied in the DAC and ADC blocks. The ADC only works for one input to one output, so the system lines are disassembled with a demux-block. When calculating gain matrix for the discrete system the system matrices as well as the eigenvalues needs to be converted to discrete time. First of all a sample time needs to be chosen, therefore $Ts = 0.1$ is selected.

The discrete matrices are found with the Matlab *c2d()* function.

$$\mathbf{F} = e^{\mathbf{A}*Ts} = \begin{bmatrix} 0.9902 & -0.09852 & 0.008162 & 0.004133 \\ 0.05968 & 0.9855 & -0.002846 & 0.000124 \\ -0.2956 & 0.05251 & 0.9533 & -0.00062 \\ -0.0147 & 0.0104 & 0.09766 & 1 \end{bmatrix}$$

$$\mathbf{G} = \int_0^{Ts} e^{\mathbf{A}Ts}\mathbf{B}\,dt = \begin{bmatrix} 0.003138 \\ -0.04718 \\ 0.01369 \\ 0.0005229 \end{bmatrix}$$

The eigenvalues of the discrete time system are

$$\lambda_F = e^{\lambda_A Ts} = \begin{Bmatrix} 0.9512 \\ 0.9418 \\ 0.9500 + 0.0475\,i \\ -0.9500 + 0.0475\,i \end{Bmatrix}$$

The discrete controller gain matrix is found using *acker()* function, as in the continuous case,

$$\mathbf{K}_{disc} = [0.3155 \quad -2.8363 \quad 0.0850 \quad 0.0159]$$

The responses for the system with and without the controller are depicted in Figure 7.

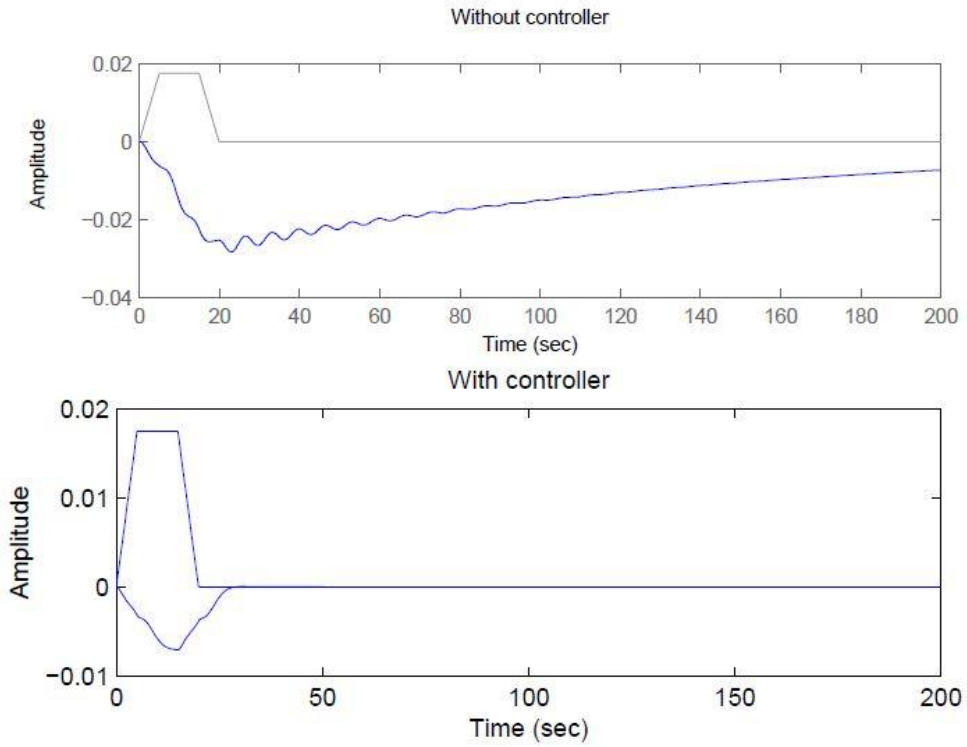


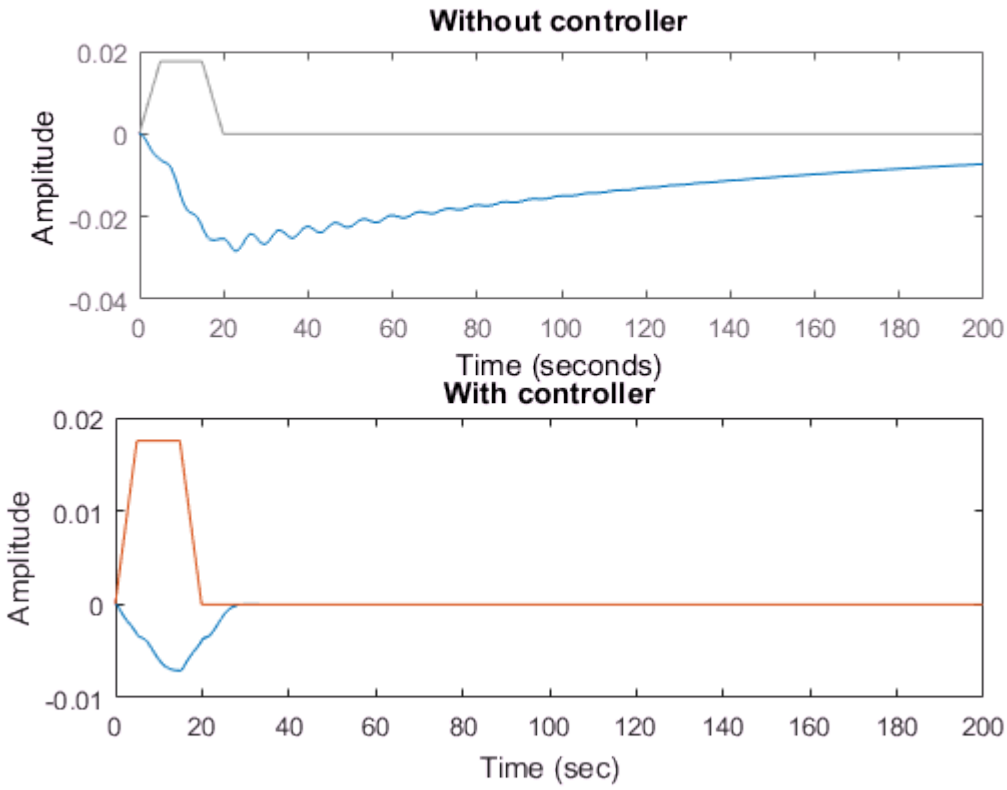
Figure 7 : The responses of the discrete system with and without the controller .

```
Ts = 0.1;
F = expm(A*Ts); % Matrix product in exponent
```

```
sys = c2d(ss(A,B,C,D),Ts,'zoh');
G = get(sys,'b');
eigDisc = exp(eigClose*Ts);

Kdisc = acker(F,G,eigDisc)
```

```
Kdisc =
    0.3155   -2.8363    0.0850    0.0159
```

```
figure(3)
subplot(2,1,1), lsim(sys,u,t,[0 0 0 0]')
title('Without controller');
sim('Q5');
subplot(2,1,2), plot(simout.time,simout.signals.values);
hold on
plot(t,u);
hold off;
title('With controller');
xlabel('Time (sec)');
ylabel('Amplitude');
```



Problem 6

Compare the system responses from Problem 4 and 5, and discuss the effect of discretization on the system performance.

Solution:

The response of the discrete system with controller resembles the continuous system with controller.

Linear Control Design II - Group Work Problem Module 13 solution

Description

Problems

The assignment deals with the multi-variable controller design for the TITO (two-inputs two-outputs) model of the aircraft introduced in Group Work Module 10.

The state space representation of the model is

$$\begin{pmatrix}\dot{\beta}\\ \dot{r}\\ \dot{p}\\ \dot{\phi}\end{pmatrix} = \begin{pmatrix}-0.0558 & -0.9968 & 0.0802 & 0.0415\\ 0.598 & -0.115 & -0.0318 & 0\\ -3.05 & 0.388 & -0.465 & 0\\ 0 & 0.0805 & 1.0 & 0\end{pmatrix}\begin{pmatrix}\beta\\ r\\ p\\ \phi\end{pmatrix} + \begin{pmatrix}0.00729 & 0.00001\\ -0.475 & 0.123\\ 0.153 & 1.063\\ 0 & 0\end{pmatrix}\begin{pmatrix}\delta r\\ \delta a\end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\beta\\ r\\ p\\ \phi\end{pmatrix}$$

Problem 1 Assess if the open loop system is controllable. Is the system stabilizable?.

Solution:

State controller for airplane model

In this module a full state controller is designed for the linear model of the airplane. Furthermore, the effects of model changes into the controller performance are investigated**.**

The linear model of the airplane is given by

$$\begin{pmatrix}\dot{\beta}\\ \dot{r}\\ \dot{p}\\ \dot{\phi}\end{pmatrix} = \begin{pmatrix}-0.0558 & -0.9968 & 0.0802 & 0.0415\\ 0.598 & -0.115 & -0.0318 & 0\\ -3.05 & 0.388 & -0.465 & 0\\ 0 & 0.0805 & 1.0 & 0\end{pmatrix}\begin{pmatrix}\beta\\ r\\ p\\ \phi\end{pmatrix} + \begin{pmatrix}0.00729 & 0.00001\\ -0.475 & 0.123\\ 0.153 & 1.063\\ 0 & 0\end{pmatrix}\begin{pmatrix}\delta r\\ \delta a\end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix}0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\beta\\ r\\ p\\ \phi\end{pmatrix}$$

The first step in control design is to evaluate the controllability and stabilizability properties of the system at hand. If the system is not fully controllable i.e. the controllability matrix is not full row rank then to assess the stabilizability of the system becomes of paramount importance in order to solve the control problem. Remember that a system is said to be stabilizable if all uncontrollable states are stable.

First, using the controllability theorem CC2 we check for controllability, that is

$$\mathbf{M}_c = \begin{pmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A^2B} & \mathbf{A^3B} \end{pmatrix}$$

$$\mathbf{M}_c = \begin{pmatrix} 0.00729 & 0.00001 & 0.48534 & -0.037354 & -0.098535 & 0.058581 & -0.40443 & 0.0049517 \\ -0.475 & 0.123 & 0.054119 & -0.047942 & 0.29284 & -0.0026226 & -0.050301 & 0.025698 \\ 0.153 & 1.063 & -0.27768 & -0.4466 & -1.3302 & 0.303 & 1.0327 & -0.32059 \\ 0 & 0 & 0.11476 & 1.0729 & -0.27332 & -0.45046 & -1.3066 & 0.30279 \end{pmatrix}$$

The controllability matrix $\mathbf{M}_c$ has full row rank; hence all states are controllable and there is no need to further investigate for stabilizability, since a controllable system is obviously stabilizable.

Problem 2 A multi-variable full state feedback controller $\mathbf{u} = -\mathbf{K}\mathbf{x} + \mathbf{r}$ for the aircraft is to be designed. The continuous time closed-loop system's eigenvalues have to be placed at the following location in the left half plane

$$\lambda = \begin{cases} -0.5 \\ -0.6 \\ -0.5 \pm 0.5\mathrm{j} \end{cases}$$

Design a discrete time controller and use the sampling period $0.1\,s$. Find the gain matrix $\mathbf{K}$ for the controller by applying the Matlab function *place*. Check the position of the eigenvalues of the closed-loop system. Are the design requirements fulfilled?

Solution:

1.1 Continuous Time State Feedback Controller

The design of the continuous time state feedback controller is done by pole placement through the MATLAB function `place`. Since the system at hand is TITO (two inputs - two outputs) the function `place` will assign both eigenvalues and eigenvectors, guaranteeing the orthogonality of the assigned eigenvectors. For the desired set of closed loop eigenvalues:

$$\lambda = \begin{cases} -0.5 \\ -0.6 \\ -0.5 \pm 0.5\mathrm{j} \end{cases}$$

the obtained continuous controller is:

$$\mathbf{K} = \begin{pmatrix} -1.4256 & -1.7645 & -0.63117 & -0.36287 \\ -2.3647 & 0.11052 & 0.67682 & 031429 \end{pmatrix}$$

```
format short g
A=[-.0558 -.9968 .0802 .0415;
```

```
   .598 -.115 -.0318 0;
   -3.05 .388 -.4650 0;
   0 .0805 1 0];
B=[.0729  .0001;
   -4.75   1.23;
   1.53   10.63;
   0      0]/10;
C=[0 1 0 0;
   0 0 0 1];
D=[0 0;
   0 0];

%% controlability
Mc = [B A*B A^2*B A^3*B];
rank(Mc)
```

```
ans =
     4
```

```
%%Continuous controller

pl = [-0.5 -0.6 -0.5+1i*0.5 -0.5-1i*0.5];
K = place(A,B,pl);
lambda = eig(A-B*K);
```

1.2 Discrete Time State Feedback Controller

The discrete time state feedback controller is design based on the discretized model, given by

$$\mathbf{F} = \exp(\mathbf{A}T)$$
$$\mathbf{G} = \int_0^T \exp(\mathbf{A}t)\mathbf{B}\,\mathrm{d}t$$
$$\lambda_{\mathbf{F}} = \exp(\lambda_{\mathbf{A}}T)$$

where $\mathbf{F}$ is the discrete time state matrix, $\mathbf{G}$ the discrete time input matrix, and $\lambda_{\mathbf{F}}$ the discrete time eigenvalues. The function \texttt{place} is also used for the discrete time system, and the controller is obtained as follow:

$$\mathbf{K} = \begin{pmatrix} -1.4184 & -1.6498 & -0.59416 & -0.34536 \\ -2.3737 & 0.24586 & 0.63373 & 0.29318 \end{pmatrix}$$

Problem 3 Add the controller to the Simulink model and carry out a simulation with the function in the following Figure as reference signal for both inputs, but one at a time. Plot the states, outputs and control signals, i.e. the rudder angle and the aileron angle.

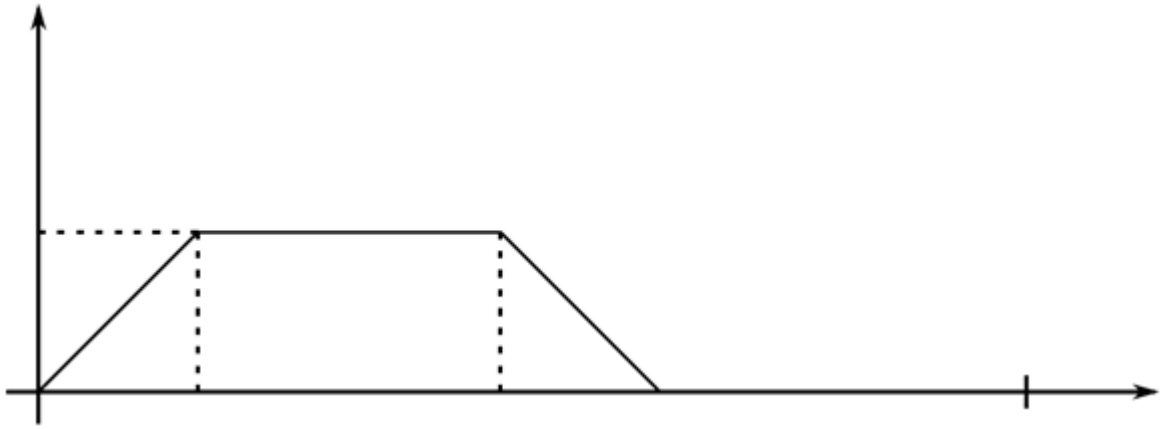


Figure 1: Reference signal for the closed-loop system.

Solution:

```
%%
%
% Generation of reference-signals
%
t=0:.1:150;
t1=0:.1:5;
r1=.0175*t1/5;
r3=.0175*(4-(t1+15)/5);
ra=[zeros(800,1)' r1 .0175*ones(99,1)' r3 zeros(1,500)];
r=[r1 .0175*ones(99,1)' r3 zeros(800,1)' zeros(1,500)];
inp = [t' r' ra'];
[ts,xs,ys] = sim('lat747mod10',150,[],inp);

%%%%%%%%
%
% plotting the results
%
%%%%%%%%
% continuous controller

figure, h1 = subplot(4,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
plot(t,r,'b',t,ra,'-.g','LineWidth',1.5)
ylabel('$\mathbf{r}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
l = legend('$\delta_r$','$\delta_a$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')

h2 = subplot(4,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(ts,ys(:,1),'k',ts,ys(:,2),'--r','LineWidth',1.5)
```

```
ylabel('$\mathbf{y}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$r$','$\phi$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')

h3 = subplot(4,1,3:4); set(h3,'FontName','times','FontSize',16)
hold on, grid on
plot(ts,xs(:,1),'k',ts,xs(:,2),'--
r',ts,xs(:,3),'-.g',ts,xs(:,4),':b','LineWidth',1.5)
ylabel('$\mathbf{x}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$\beta$','$r$','$p$','$\phi$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')
```

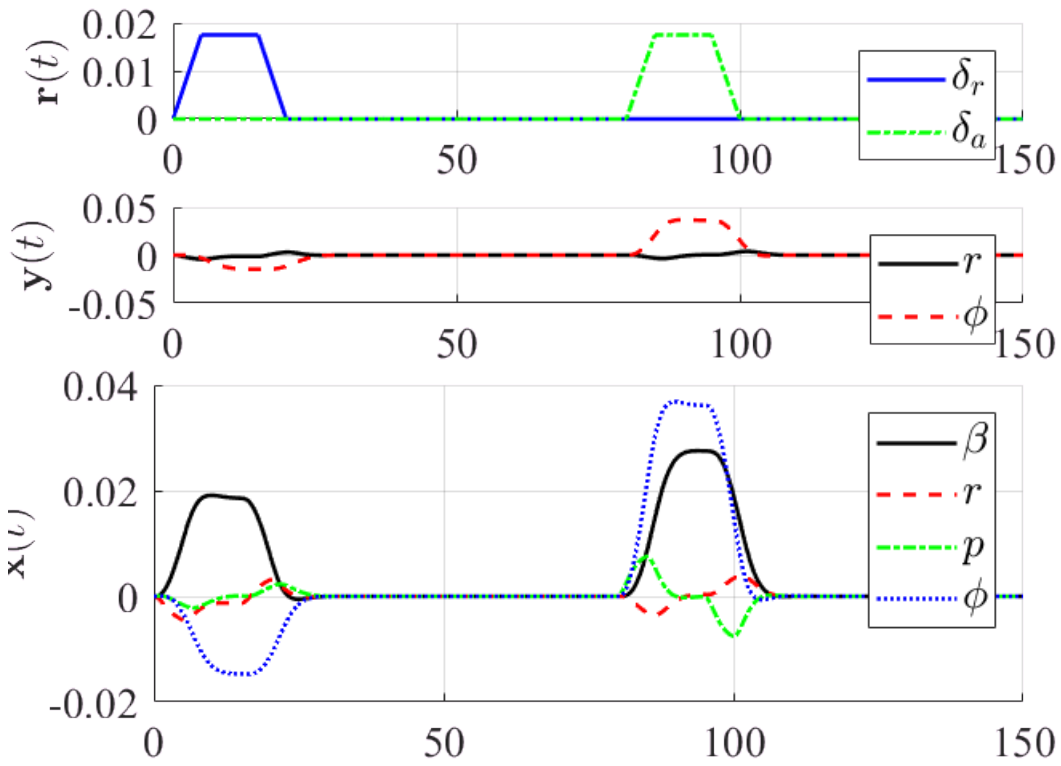


Figure 2: Simulation of the continuous time state feedback controller.

The fulfilment of the closed loop requirement (position of the eigenvalues) is checked by solving the closed loop eigenvalue problem $det(\mathbf{I}\lambda - \mathbf{A} + \mathbf{BK}) = 0$. Figure 2 illustrates the performance of the closed loop system in response to the reference signal.

Simulink implementation and simulation

To implement the discrete time controller together with the continuous time system in SIMULINK requires the inclusion of AD and DA converters, as shown in Figure 3. The simulation results of the sampled data system are

shown in Figure 4. Comparing Figure 4 with Figure 2 it is worth noting that the discrete time controller ensures the same performance of the continuous time controller.

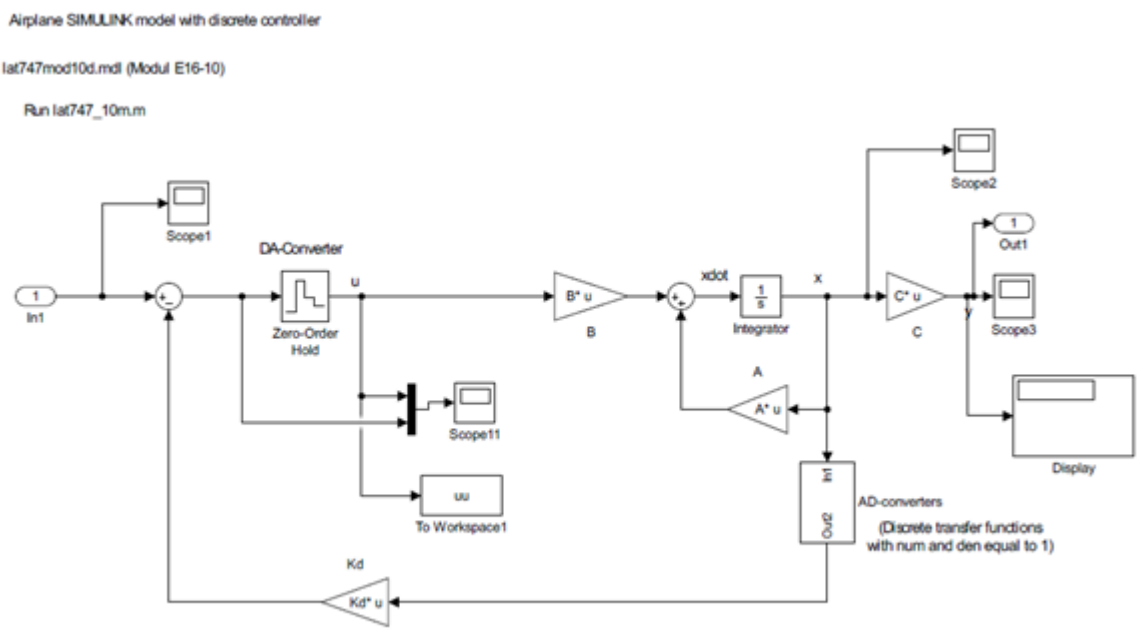


Figure 3: SIMULINK block diagram with AD and DA converters included.

```
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Discrete Controller design
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ev=[-.5 -.6 -.5+1i*.5 -.5-1i*.5];
T=0.1;
[F,G]=c2d(A,B,T);
evd=exp(T*ev);
Kd=place(F,G,evd);
lambdaA = log(eig(F-G*Kd))/T;

[td,xd,yd] = sim('lat747mod10d',150,[],inp);

%%%%%%%%
%
% plotting the results
%
%%%%%%%%
% discrete model

figure, h4 = subplot(4,1,1); set(h4,'FontName','times','FontSize',16)
hold on, grid on
```

```
plot(t,r,'b',t,ra,'-.g','LineWidth',1.5)
ylabel('$\mathbf{r}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
l = legend('$\delta_r$','$\delta_a$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')

h5 = subplot(4,1,2); set(h5,'FontName','times','FontSize',16)
hold on, grid on
plot(td,yd(:,1),'k',td,yd(:,2),'--r','LineWidth',1.5)
ylabel('$\mathbf{y}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$r$','$\phi$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')

h6 = subplot(4,1,3:4); set(h6,'FontName','times','FontSize',16)
hold on, grid on
plot(td,xd(:,1),'k',td,xd(:,2),'--
r',td,xd(:,3),'-.g',td,xd(:,4),':b','LineWidth',1.5)
ylabel('$\mathbf{x}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$\beta$','$r$','$p$','$\phi$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')
```

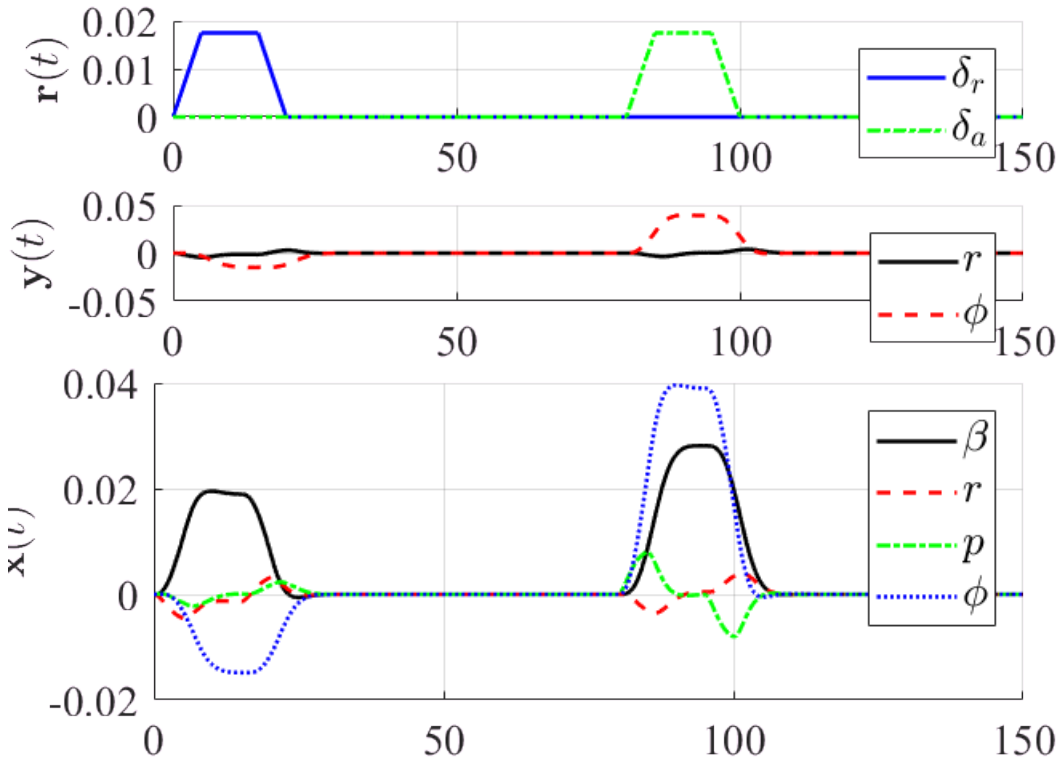


Figure 4: Simulation of the sampled data system.

Problem 4 Now suppose that the aircraft's rudder and aileron servos are slower than we thought. Add a $1\ s$ time constant to each of the continuous control inputs and see what happens. How could the problem be solved?

Solution:

Actuators with Slower Dynamics

To illustrate how the performance of the controller depends on the systems it is design for, actuators with slower dynamics are included into the system. This is achieved by including two low pass filters (one for each input) with time constant of one second before the input matrix $\mathbf{B}$. Figure 5 clearly shows that the controller is affected by the slower dynamics of the actuators and, therefore, the performance shown in Figure 4 can no longer be guaranteed. Hence a new controller should be designed for this new system dynamics.

The design of a new controller passes through the expansion of the system dynamical model as mentioned in problem 1.

$$\mathbf{x} = \begin{bmatrix} \beta & r & p & \phi & z_1 & z_2 \end{bmatrix}^T$$

$$\overline{\mathbf{A}} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \mathbf{0}_{2\times 4} & -\mathbf{I}_{2\times 2} \end{array}\right]$$

$$\overline{\mathbf{B}} = \begin{bmatrix} \mathbf{0}_{4\times 2} \\ \mathbf{I}_{2\times 2} \end{bmatrix}$$

$$\overline{\mathbf{C}} = \begin{bmatrix} \mathbf{C} & \mathbf{0}_{2\times 2} \end{bmatrix}$$

```
%% Model with added timeconstants

[tdn,xdn,ydn] = sim('lat747mod10dn',150,[],inp);

%%%%%%%%
%
% plotting the results
%
%%%%%%%%
% discrete model

figure, h7 = subplot(4,1,1); set(h7,'FontName','times','FontSize',16)
hold on, grid on
plot(t,r,'b',t,ra,'-.g','LineWidth',1.5)
ylabel('$\mathbf{r}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
l = legend('$\delta_r$','$\delta_a$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')

h8 = subplot(4,1,2); set(h8,'FontName','times','FontSize',16)
hold on, grid on
plot(tdn,ydn(:,1),'k',tdn,ydn(:,2),'--r','LineWidth',1.5)
```

```
ylabel('$\mathbf{y}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$r$','$\phi$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')

h9 = subplot(4,1,3:4); set(h9,'FontName','times','FontSize',16)
hold on, grid on
plot(tdn,xdn(:,1),'k',td,xdn(:,2),'--
r',tdn,xdn(:,3),'-.g',tdn,xdn(:,4),':b','LineWidth',1.5)
ylabel('$\mathbf{x}(t)
$','FontName','times','FontSize',16,'interpreter','latex')
hold off
l = legend('$\beta$','$r$','$p$','$\phi$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'interpreter','latex')
```

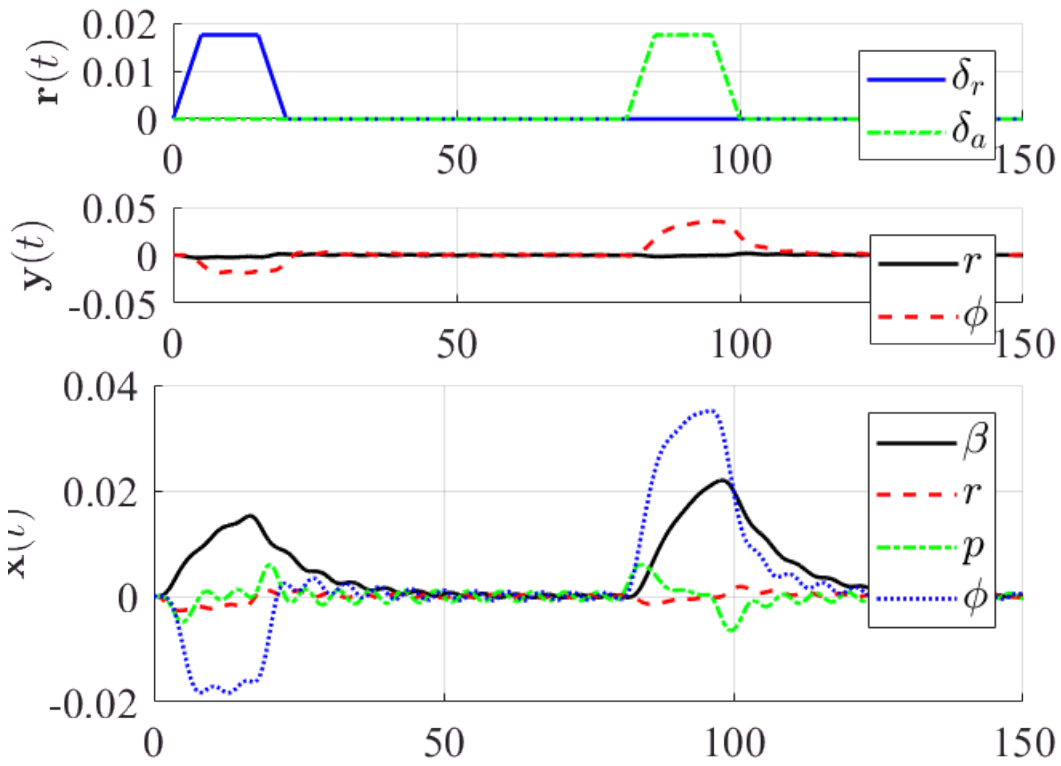


Figure 5: Simulation of the discrete time controller on the system with slower actuators.

Problem 5 Find the stationary output $\mathbf{y}_0$ for the closed-loop system and for the constant reference inputs $\mathbf{r}(k) = [0.1 \;\; 0]^{\mathrm{T}}$ and $\mathbf{r}(k) = [0 \;\; 0.1]^{\mathrm{T}}$. Does y_0 depend on the eigenvalue placement? (`Hint: For a stationary state the discrete state vector is constant, i.e.` $\mathbf{x}(k+1) = \mathbf{x}(k)$).

Solution:

Stationary States

The following equations how the stationary states depend on the controller. Since the controller is designed based on the desired closed loop eigenvalues, it is evident that also the stationary states are dependent on them.

$$\mathbf{x}(k+1) = \mathbf{x}(k) \Rightarrow \mathbf{x}_0 = (\mathbf{F} - \mathbf{GK})\mathbf{x}_0 + \mathbf{G}\mathbf{r}_0$$
$$(\mathbf{I} - \mathbf{F} + \mathbf{GK})\mathbf{x}_0 = \mathbf{G}\mathbf{r}_0$$
$$\mathbf{x}_0 = \mathbf{Q}^{-1}\mathbf{G}\mathbf{r}_0, \mathbf{Q} = (\mathbf{I} - \mathbf{F} + \mathbf{GK})$$
$$\mathbf{y}_0 = \mathbf{C}\mathbf{x}_0 = \mathbf{C}\mathbf{Q}^{-1}\mathbf{G}\mathbf{r}_0$$

```
% Computation of stationary states x0 and outputs y0:
%


r0=[.1 0]';
x0=inv(eye(4,4)-F+G*Kd)*G*r0
```

```
x0 =
       0.1082
   -0.0079812
   0.00064249
    -0.084668
```

```
y0=C*x0
```

```
y0 =
   -0.0079812
    -0.084668
```

```

r0=[0 .1]';
x0=inv(eye(4,4)-F+G*Kd)*G*r0
```

```
x0 =
      0.16043
    0.0025628
  -0.00020631
      0.22332
```

```
y0=C*x0
```

```
y0 =
    0.0025628
      0.22332
```

Linear Control Design II - Group Work Problem Module 14 Solution

Roberto Galeazzi, Dimitros Papageorgiou & Andreas S. Pedersen
Technical University of Denmark
September, 2014
Thomas Paulsen
September, 2017

Description

Consider the airplane model (MIMO model).

```
% System matrices:
A = [-0.0558 -0.9968  0.0802  0.0415;
      0.598  -0.115  -0.0318  0;
     -3.050   0.388  -0.4650  0;
      0       0.0805  1       0]
```

```
A = 4×4
   -0.0558   -0.9968    0.0802    0.0415
    0.5980   -0.1150   -0.0318         0
   -3.0500    0.3880   -0.4650         0
         0    0.0805    1.0000         0
```

```
B = [0.00729 1e-5; -0.475 0.123; 0.153 1.063; 0 0]
```

```
B = 4×2
    0.0073    0.0000
   -0.4750    0.1230
    0.1530    1.0630
         0         0
```

```
C = [0 1 0 0;0 0 0 1]
```

```
C = 2×4
     0     1     0     0
     0     0     0     1
```

```
D = [0 0;0 0];
```

Problem 1

Is the system observable? Is it detectable?

Solution:

```
% Check observability
rank(obsv(A,C))
```

```
ans = 4
```

Problem 2

In module 15 a multivariable state feedback controller for the aircraft was designed to assign the following closed-loop continuous eigenvalues

$$\lambda = \begin{cases} -0.5 \\ -0.6 \\ -0.5 \pm 0.5\,j \end{cases}$$

Design a discrete time observer for the system and use the sampling period $Ts = 0.1\text{s}$. Find the observer gain matrix **L** that assigns the eigenvalues of the estimation error dynamics to be 4-5 times faster than those of the closed-loop system (in continuous time). Use the Matlab function *place()* and remember that an observer can be designed by solving the dual problem of controller design.

Check the eigenvalues of the estimation error.

Solution:

We discretize the system using *c2d()*:

$$\mathbf{F} = \begin{bmatrix} 0.9902 & -0.0990 & 0.0082 & 0.0041 \\ 0.0600 & 0.9958 & -0.0029 & 0.0001 \\ -0.2956 & 0.0528 & 0.9533 & -0.0006 \\ -0.0147 & 0.0104 & 0.0977 & 1.0000 \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} 0.0031 & -0.0002 \\ -0.0474 & 0.0121 \\ 0.0137 & 0.1041 \\ 0.0005 & 0.0053 \end{bmatrix}$$

In the solution we use eigenvalues 5 times faster than the closed-loop system.

$$\lambda_d = \mathrm{e}^{5\lambda_{cl}Ts}$$

The observer-gain is found in matlab

$$\mathbf{L} = place\,(\mathbf{F}^T, \mathbf{C}^T, \lambda_d)^T = \begin{bmatrix} 0.1476 & -0.4685 \\ 0.3162 & -0.1297 \\ -0.3285 & 1.3029 \\ -0.0251 & 0.5943 \end{bmatrix}$$

```
% Discrete system
Ts = 0.1;
[F,G] = c2d(A,B,Ts);

% Observer design
lambda = [-0.5 -0.6 -0.5+0.5i -0.5-0.5i];
P_obsv_d = exp(5*lambda*Ts);
```

```
L = place(F',C',P_obsv_d)'
```

```
L = 4×2
    0.1476   -0.4695
    0.3055   -0.1293
   -0.3282    1.3050
   -0.0252    0.5947
```

```
log(eig(F-L*C))/Ts
```

```
ans = 4×1 complex
  -2.5000 + 2.5000i
  -2.5000 - 2.5000i
  -2.5000 + 0.0000i
  -3.0000 + 0.0000i
```

Problem 3

Add the observer to the open loop Simulink model of the aircraft and carry out a simulation with the function on Figure 1 at both inputs, but one at a time. Plot the states, the state estimates and the estimation error.

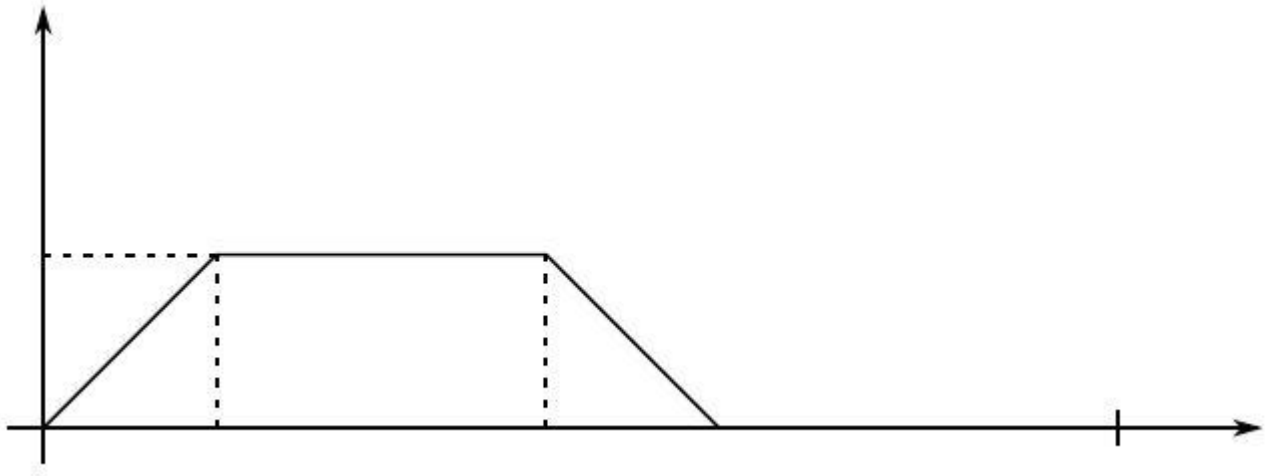


Figure 1 : Input signal for the closed loop system .

Solution:

The simulink implementation of the discrete-time full-order observer can be seen in Figure 2.

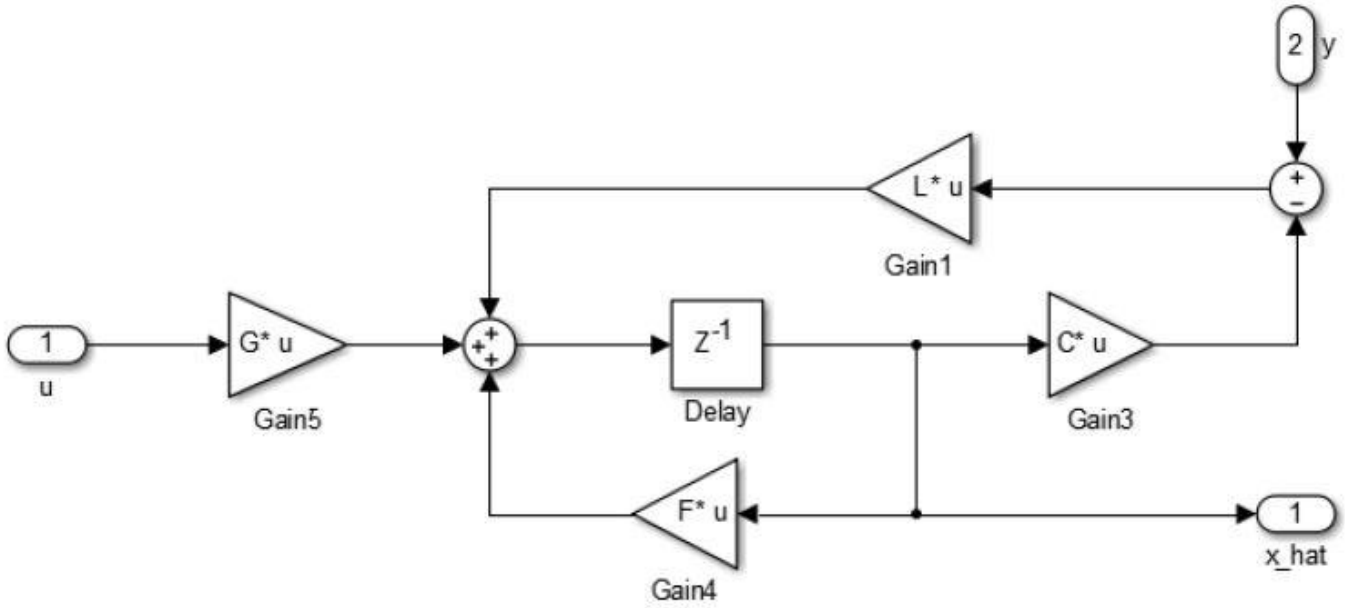
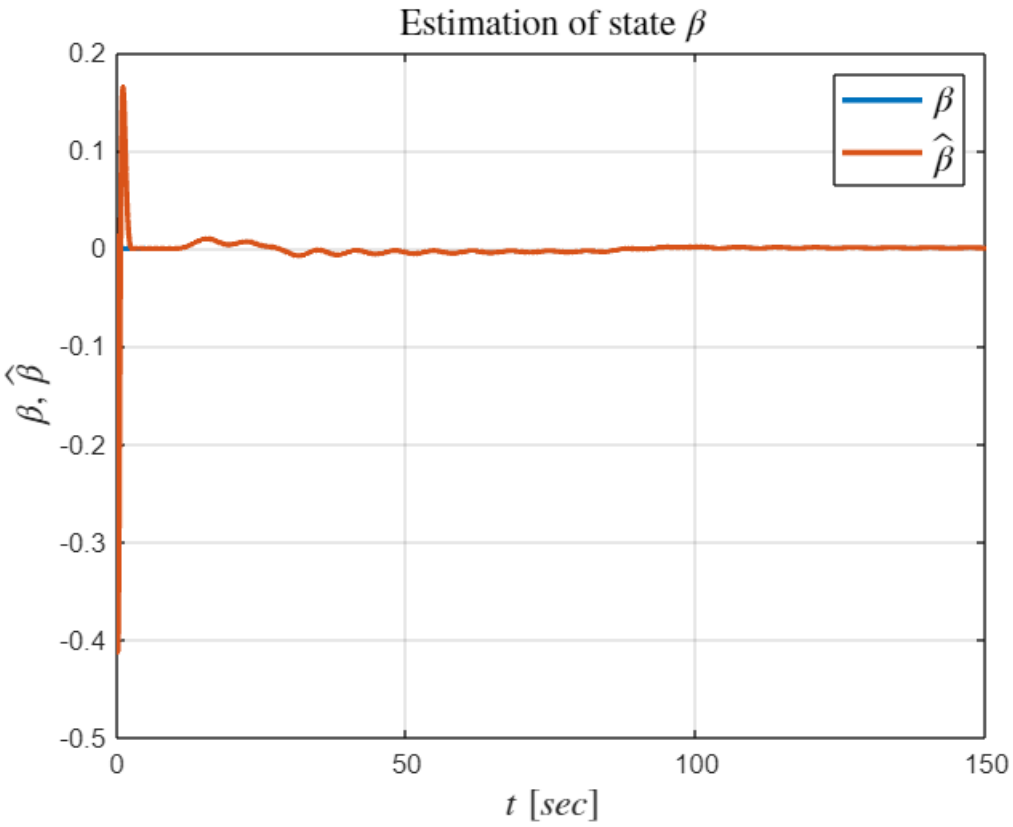
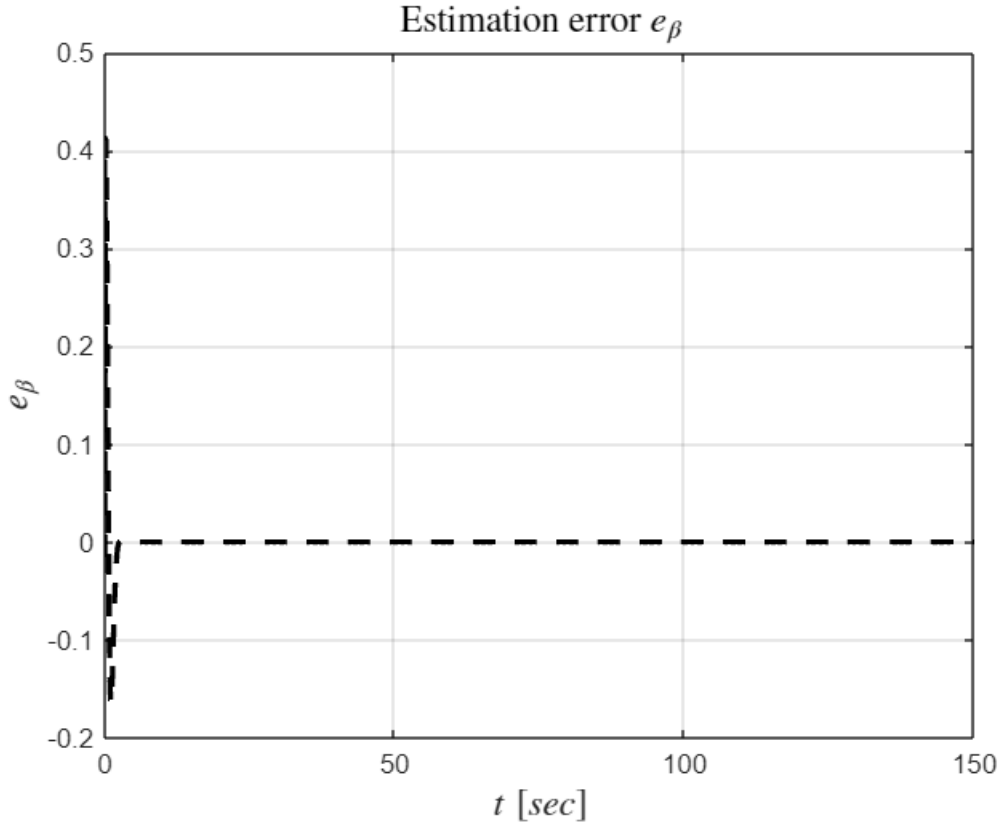


Figure 2 : Full order observer

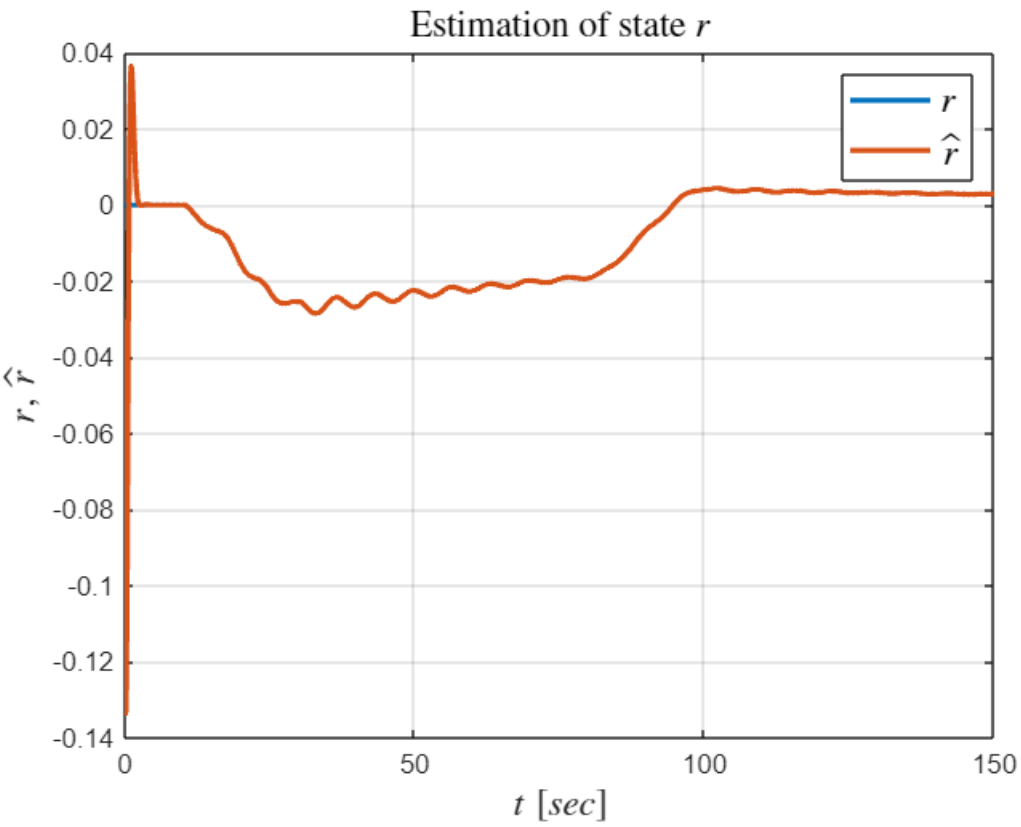
```
%% Simulation
t=0:.1:150;
t1=0:.1:5;
r1=.0175*t1/5;
r3=.0175*(4-(t1+15)/5);
ra=[zeros(800,1)' r1 .0175*ones(99,1)' r3 zeros(1,500)];
r=[zeros(1,100) r1 .0175*ones(99,1)' r3 zeros(800,1)' zeros(1,400)];
inp = [t' r' ra'];

%x0 = -[0.015;-0.03;-0.08;-0.7];
%x_esht_0 = [0.015;-0.03;-0.08;-0.7];
x0 = [0;0;0;0];
x_esht_0 = [0.015;-0.03;-0.08;-0.7];

sim('obsvOpenLoopSimul');

%% Plots
t = states.time;
figure
plot(t,states.signals.values(:,[1,5]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$\beta, \; \hat{\beta}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$\beta$', '$\hat{\beta}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $\beta$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



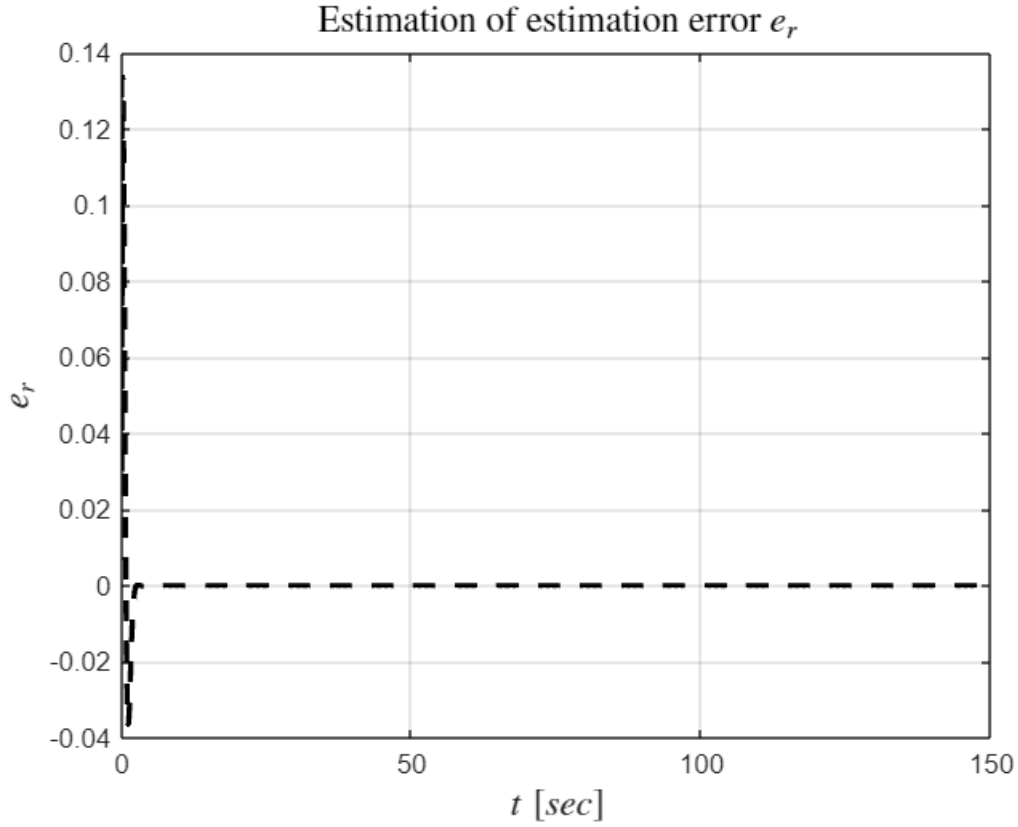

```
figure
plot(t,error.signals.values(:,1), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{\beta}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation error $e_{\beta}$', 'Interpreter', 'latex', 'FontSize',
14);
grid;
```



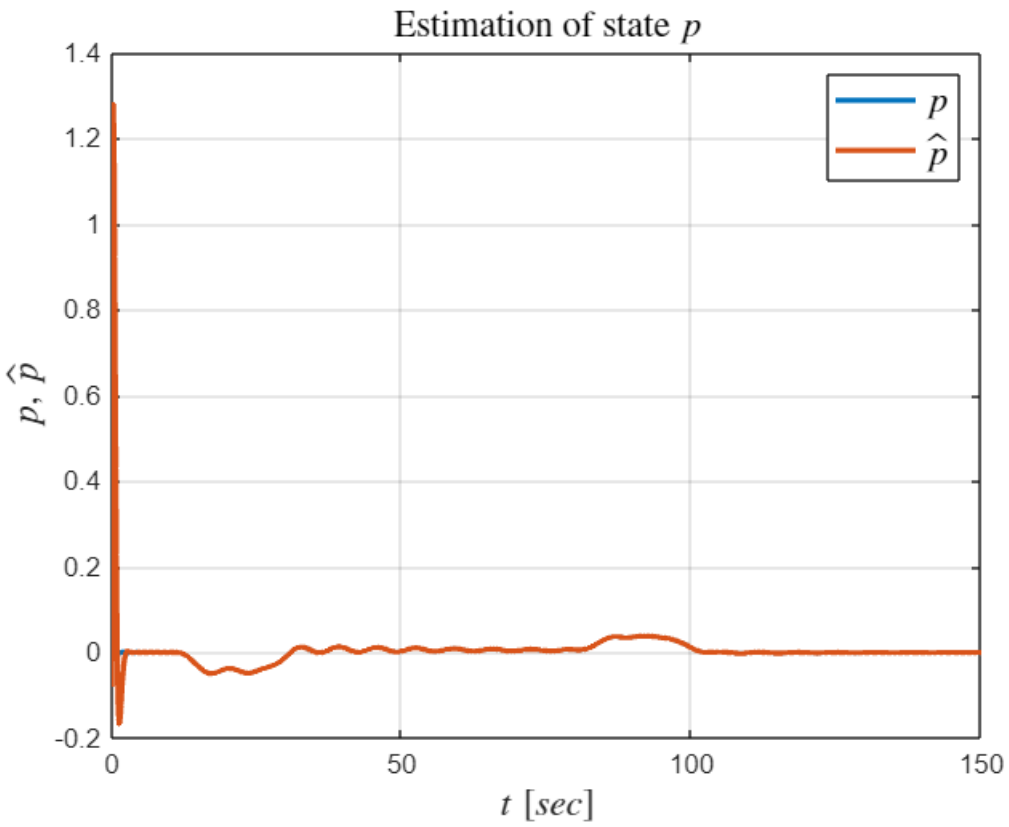

```
figure
plot(t,states.signals.values(:,[2,6]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$r, \; \hat{r}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$r$', '$\hat{r}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $r$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



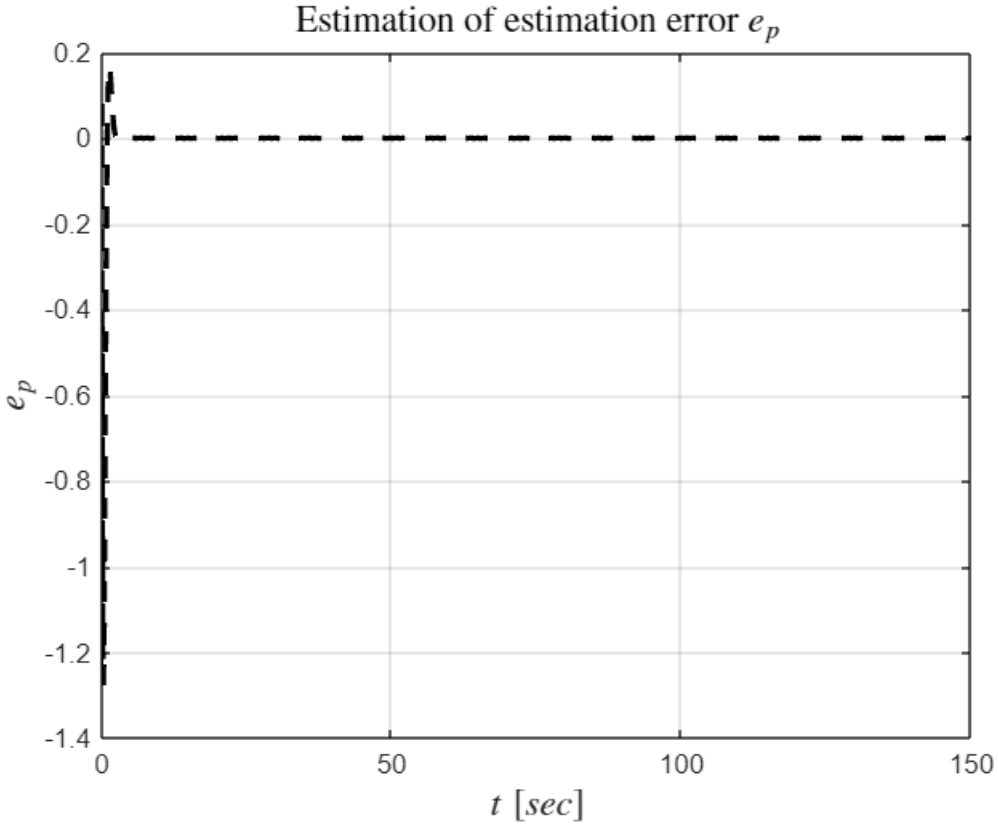

```
figure
plot(t,error.signals.values(:,2), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{r}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{r}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```



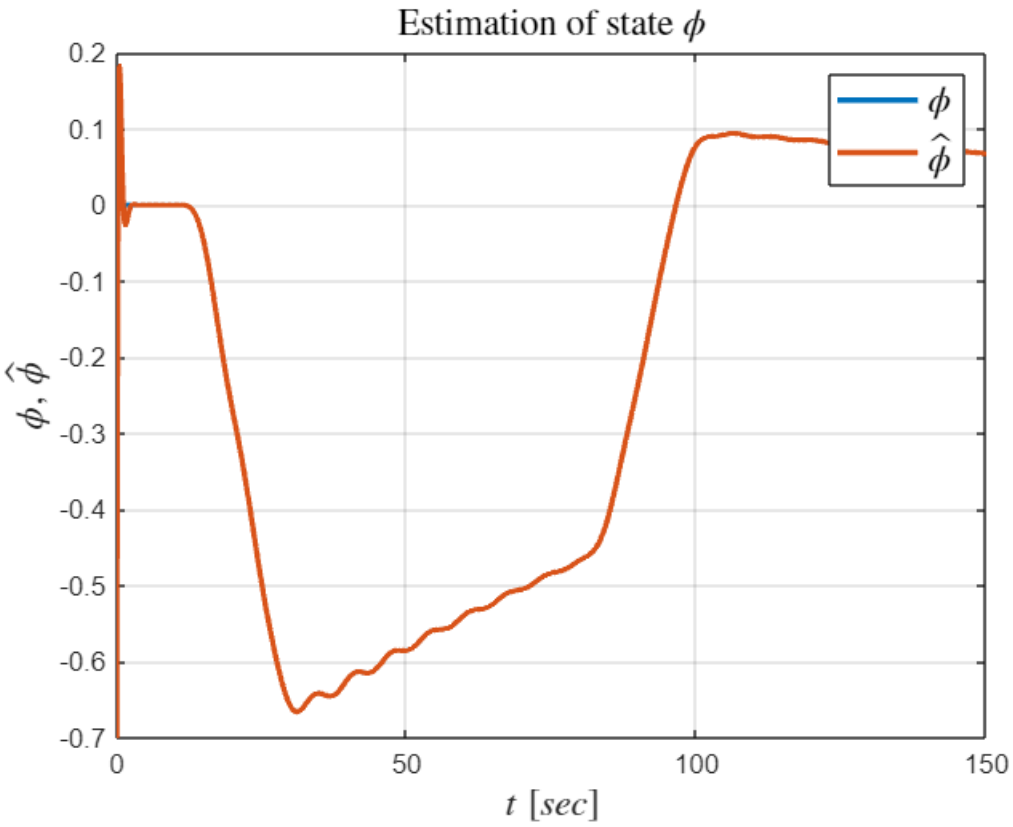

```
figure
plot(t,states.signals.values(:,[3,7]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$p, \; \hat{p}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$p$', '$\hat{p}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $p$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



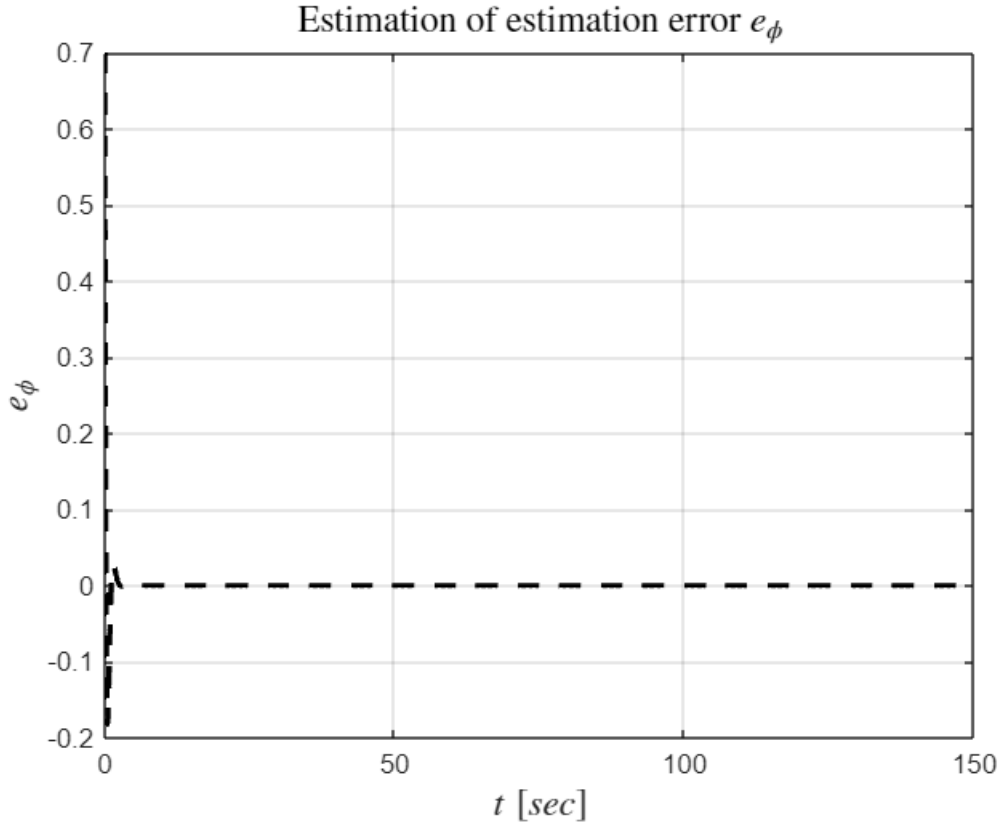

```
figure
plot(t,error.signals.values(:,3), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{p}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{p}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```




```
figure
plot(t,states.signals.values(:,[4,8]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$\phi, \; \hat{\phi}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$\phi$', '$\hat{\phi}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $\phi$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```




```
figure
plot(t,error.signals.values(:,4), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{\phi}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{\phi}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```



Problem 4

Add the discrete controller designed in the module 15 to the Simulink model and instead of using the true states of the system, use their estimates provided by the observer. Carry out simulations and plot the states, the state estimates, the estimation error, the outputs and the control signals. Compare the performance of the closed-loop systems with and without the observer.

Solution:

The simulink implementation of the closed-loop system can be seen in Figure 3.

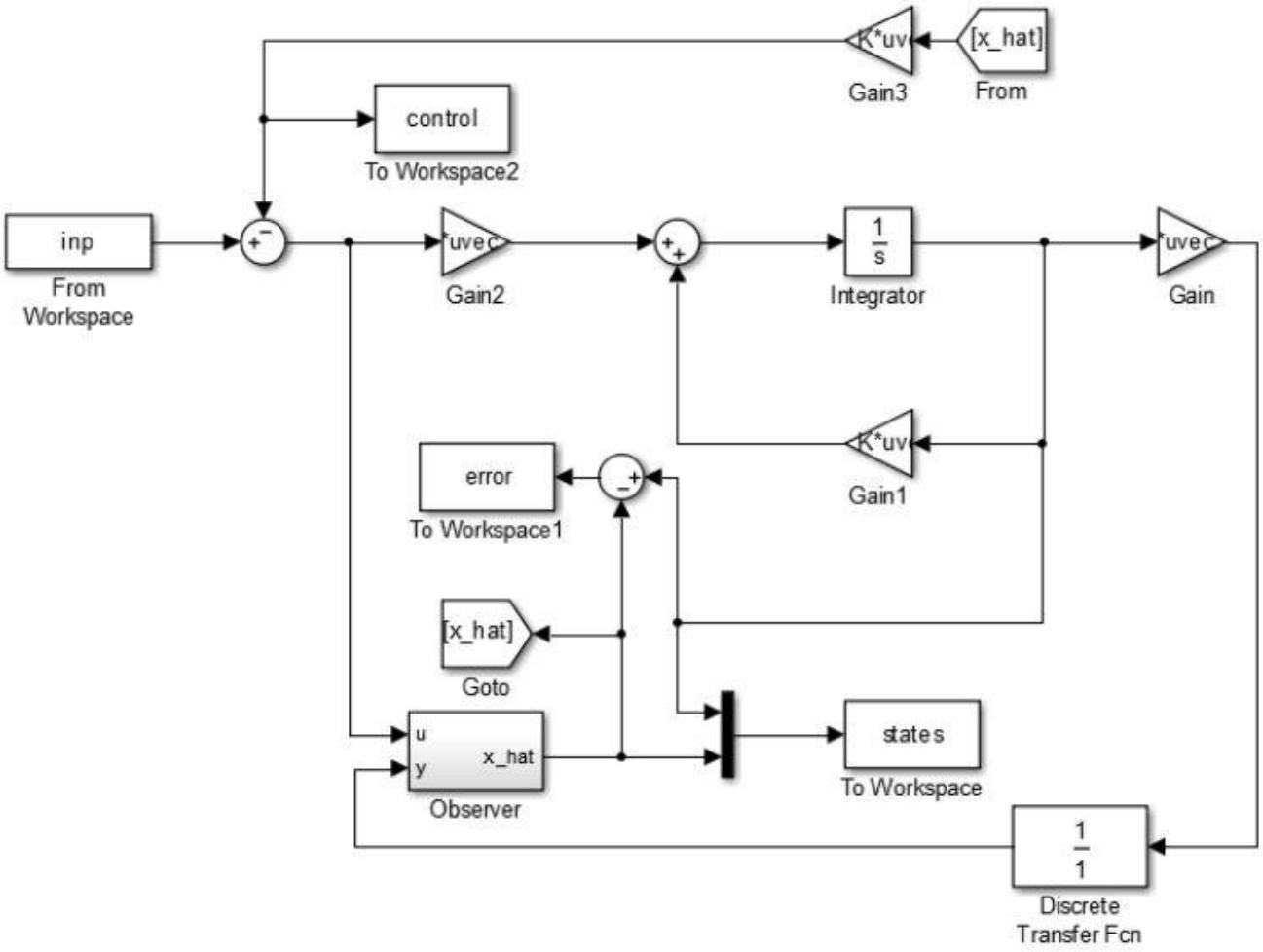
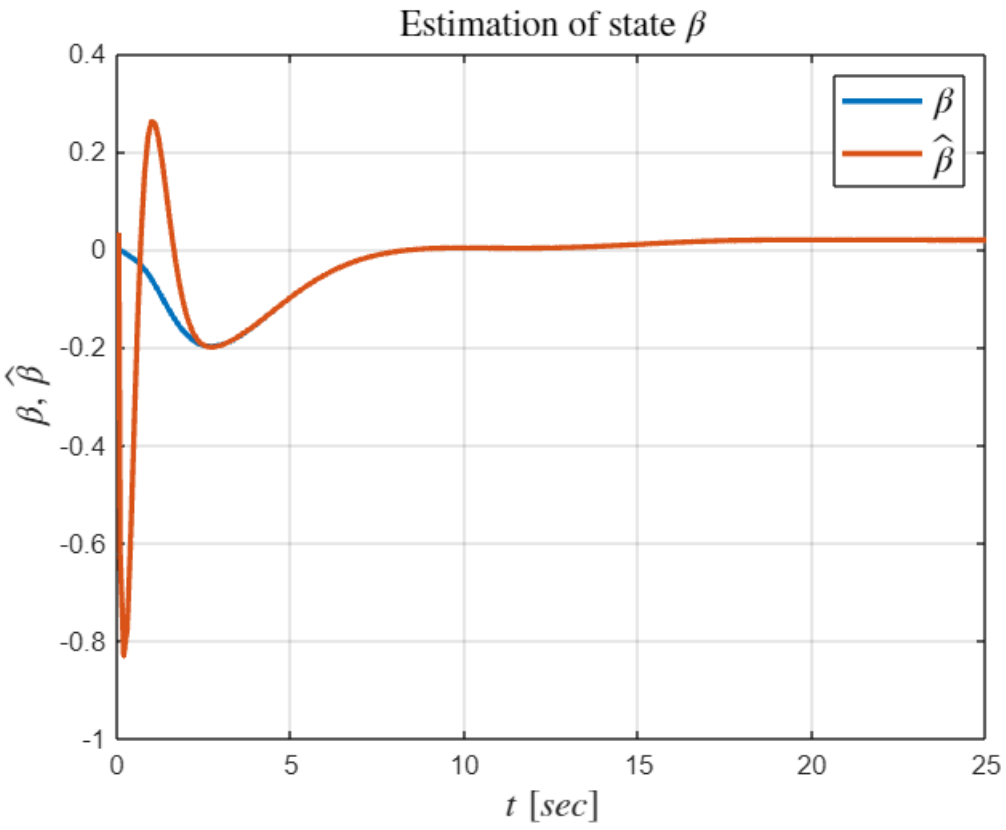


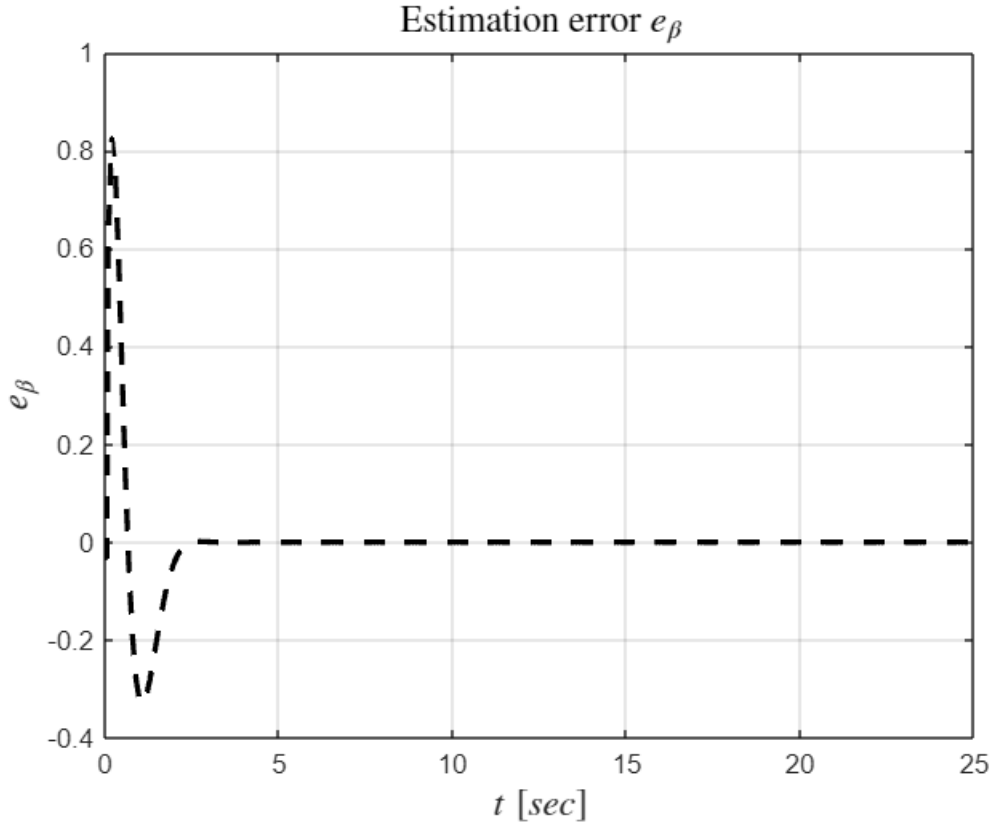
Figure 3 : Observer full − state feedback controller .

```
%% Simulation
t=0:.1:150;
t1=0:.1:5;
r1=.0175*t1/5;
r3=.0175*(4-(t1+15)/5);
ra=[zeros(800,1)' r1 .0175*ones(99,1)' r3 zeros(1,500)];
r=[zeros(1,100) r1 .0175*ones(99,1)' r3 zeros(800,1)' zeros(1,400)];
inp = [t' r' ra'];

% Discrete controller
ev=[-.5 -.6 -.5+1i*.5 -.5-1i*.5];
evd=exp(Ts*ev);
Kd=place(F,G,evd);

%x0 = -[0.015;-0.03;-0.08;-0.7];
%x_esht_0 = [0.015;-0.03;-0.08;-0.7];
x0 = [0;0;0;0];
x_esht_0 = [0.015;-0.03;-0.08;-0.7]*2;
```

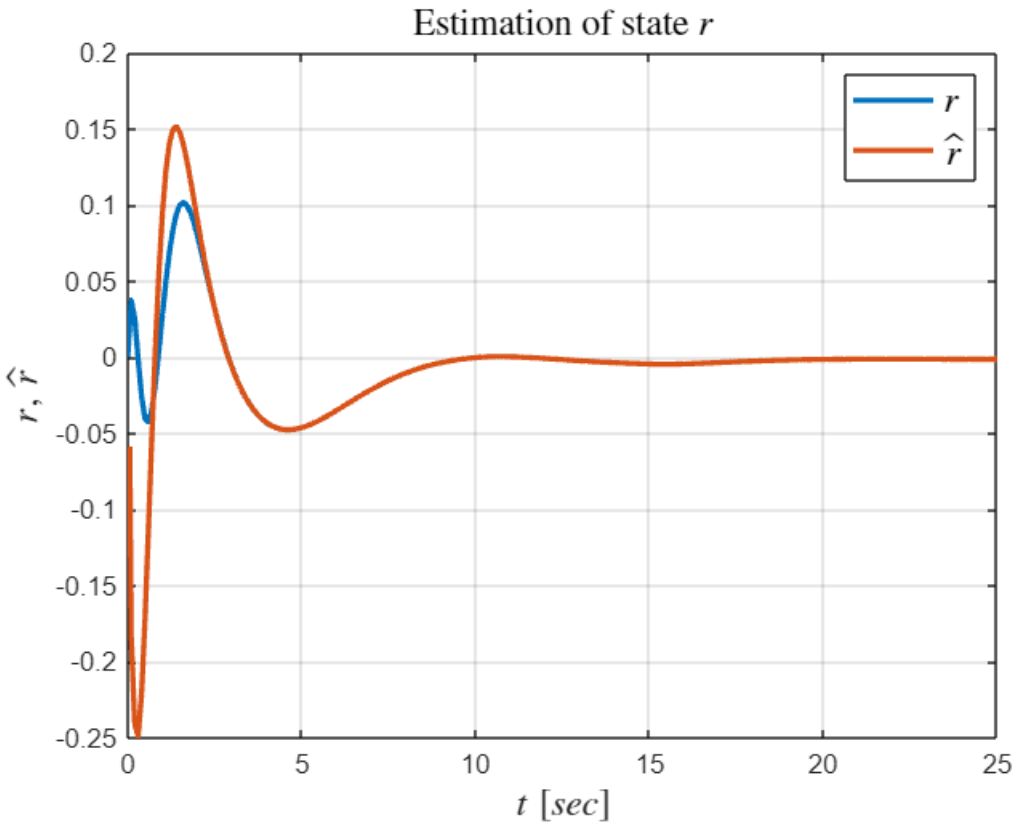
```
sim('obsvSimul');

%% Plots
t = states.time;
figure;
plot(t,states.signals.values(:,[1,5]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$\beta, \; \hat{\beta}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$\beta$', '$\hat{\beta}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $\beta$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



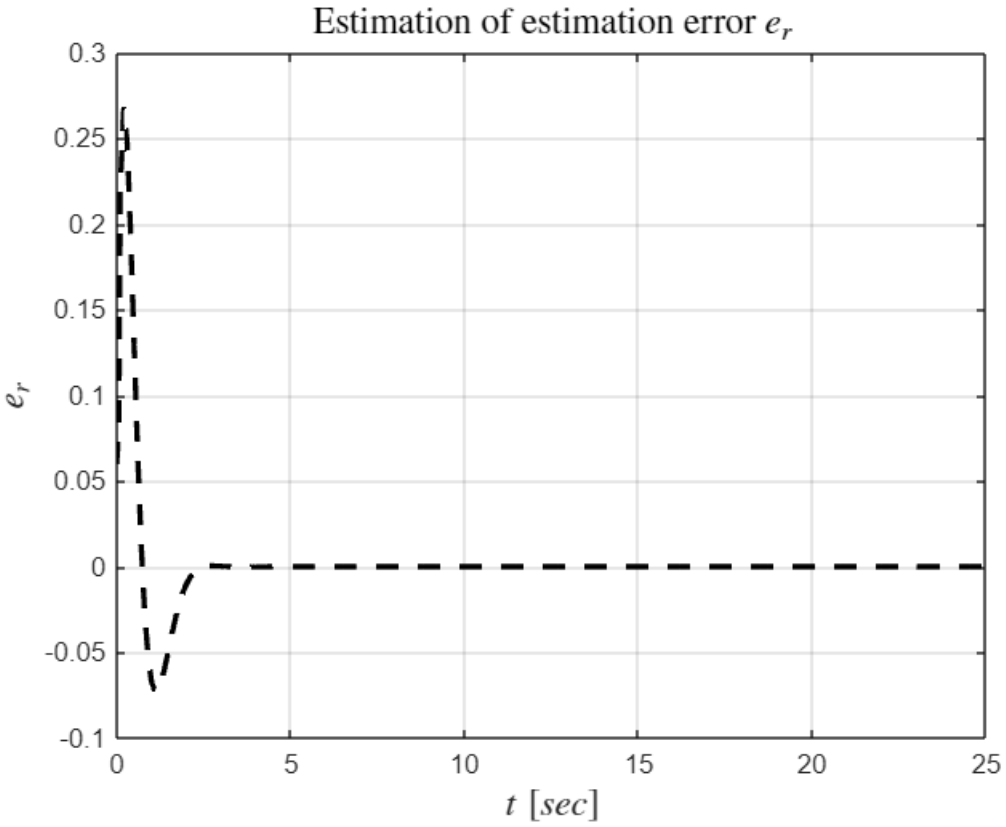

```
figure
plot(t,error.signals.values(:,1), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{\beta}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation error $e_{\beta}$', 'Interpreter', 'latex', 'FontSize',
14);
grid;
```



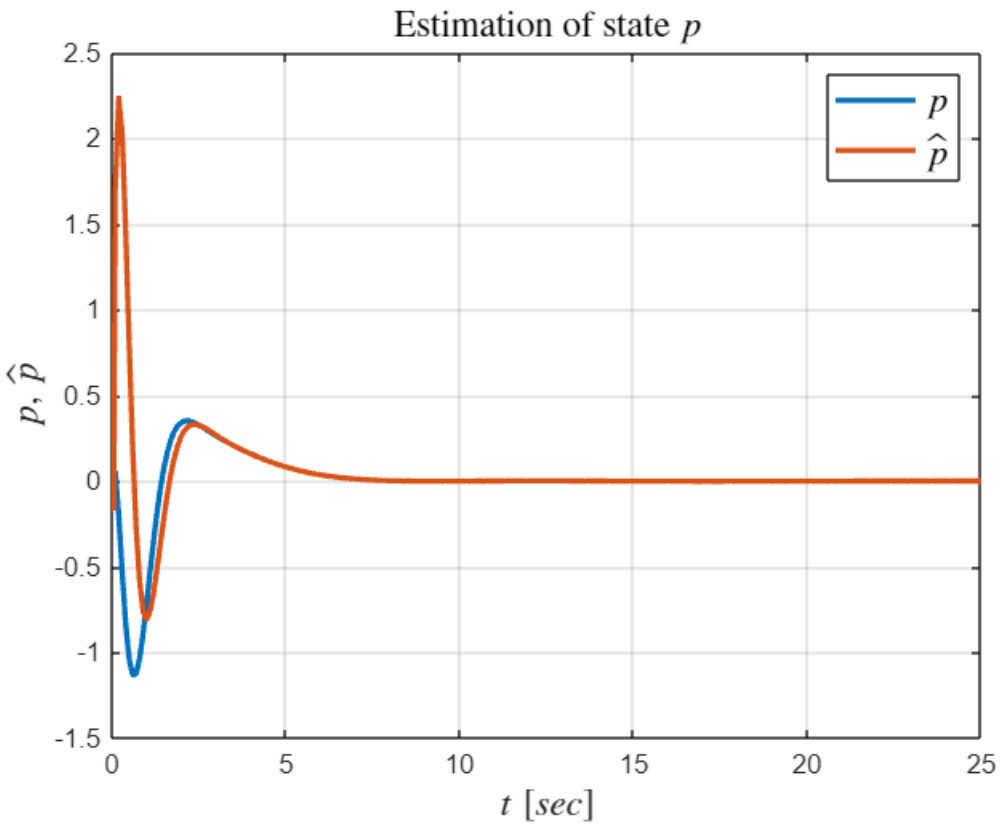

```
figure
plot(t,states.signals.values(:,[2,6]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$r, \; \hat{r}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$r$', '$\hat{r}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $r$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



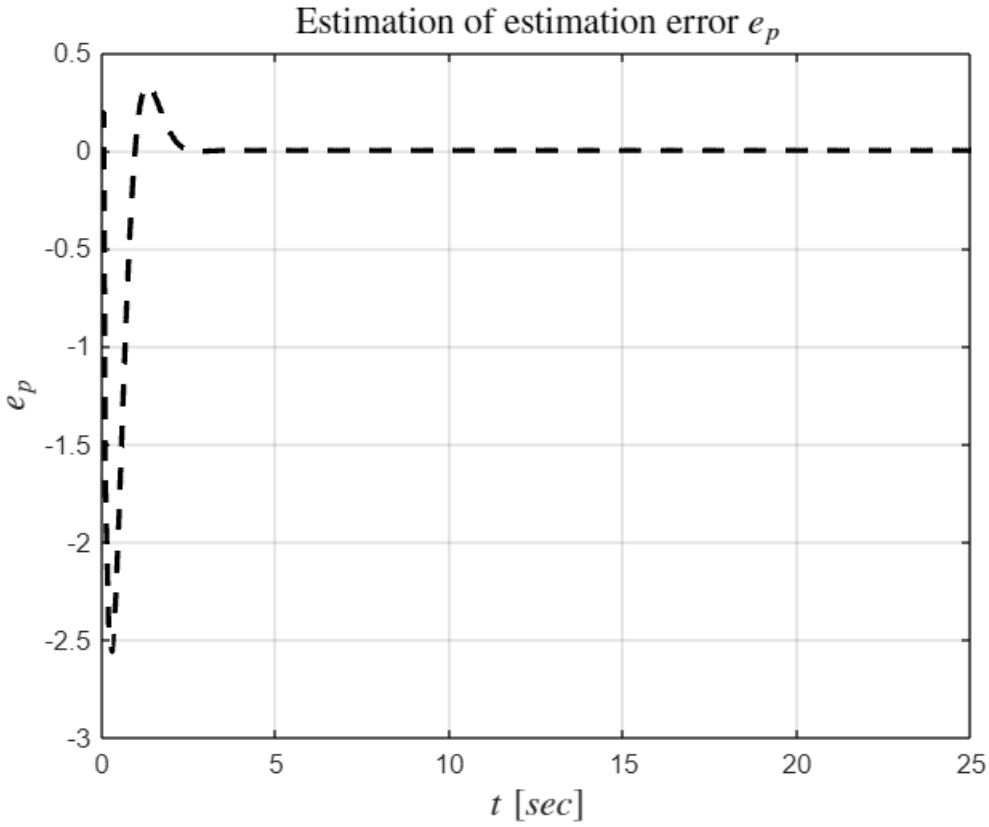

```
figure
plot(t,error.signals.values(:,2), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{r}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{r}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```



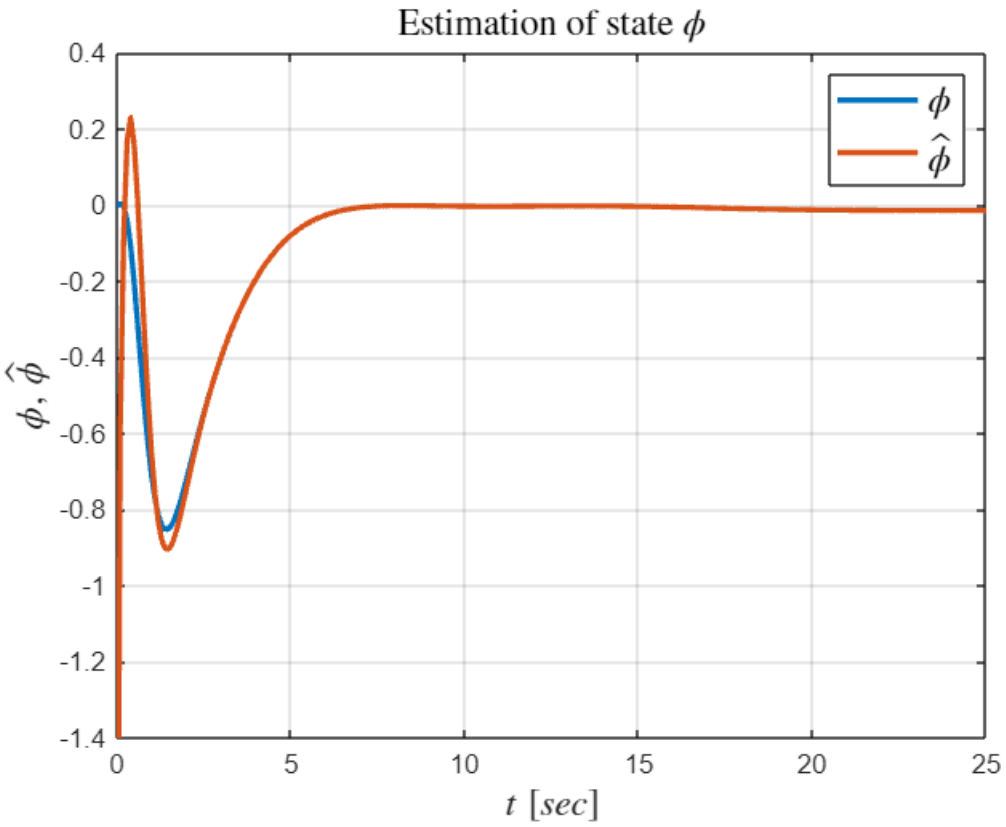

```
figure
plot(t,states.signals.values(:,[3,7]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$p, \; \hat{p}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$p$', '$\hat{p}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $p$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



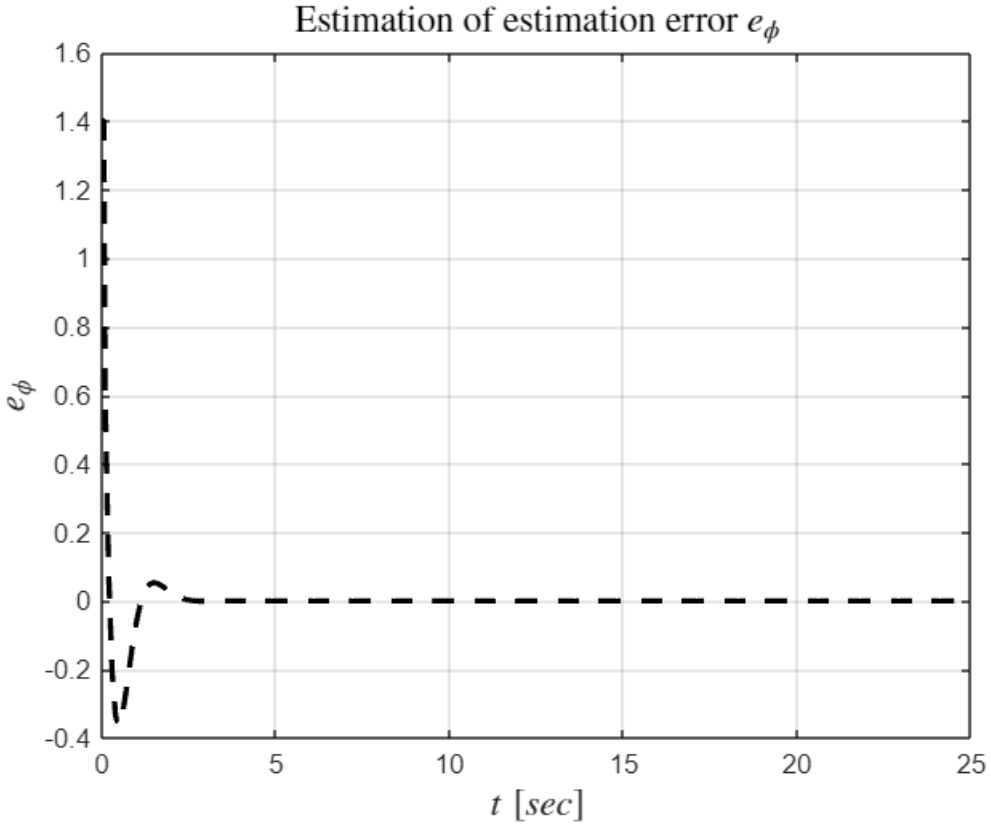

```
figure
plot(t,error.signals.values(:,3), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{p}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{p}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```




```
figure
plot(t,states.signals.values(:,[4,8]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$\phi, \; \hat{\phi}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$\phi$', '$\hat{\phi}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $\phi$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```




```
figure
plot(t,error.signals.values(:,4), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{\phi}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{\phi}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```



Problem 5

Do the estimations obtained from the observer depend on how good the controller is? Can we estimate any of the states if the system is very fast or unstable?

Solution:

As the controller input is also used in the observer, the estimations are independent on the controller performance. If the system is much faster than the observer, disturbance responses can be difficult to estimate.

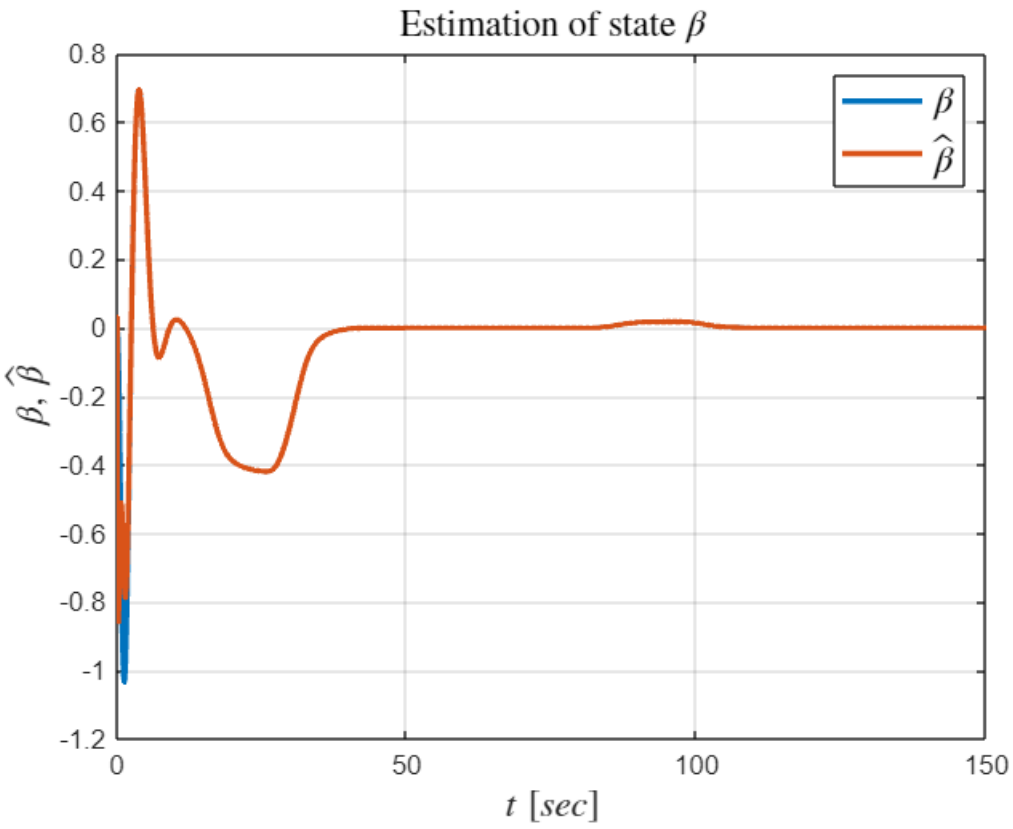
Problem 6

Repeat Problem 2-4 using the controller with integral action.

Solution:

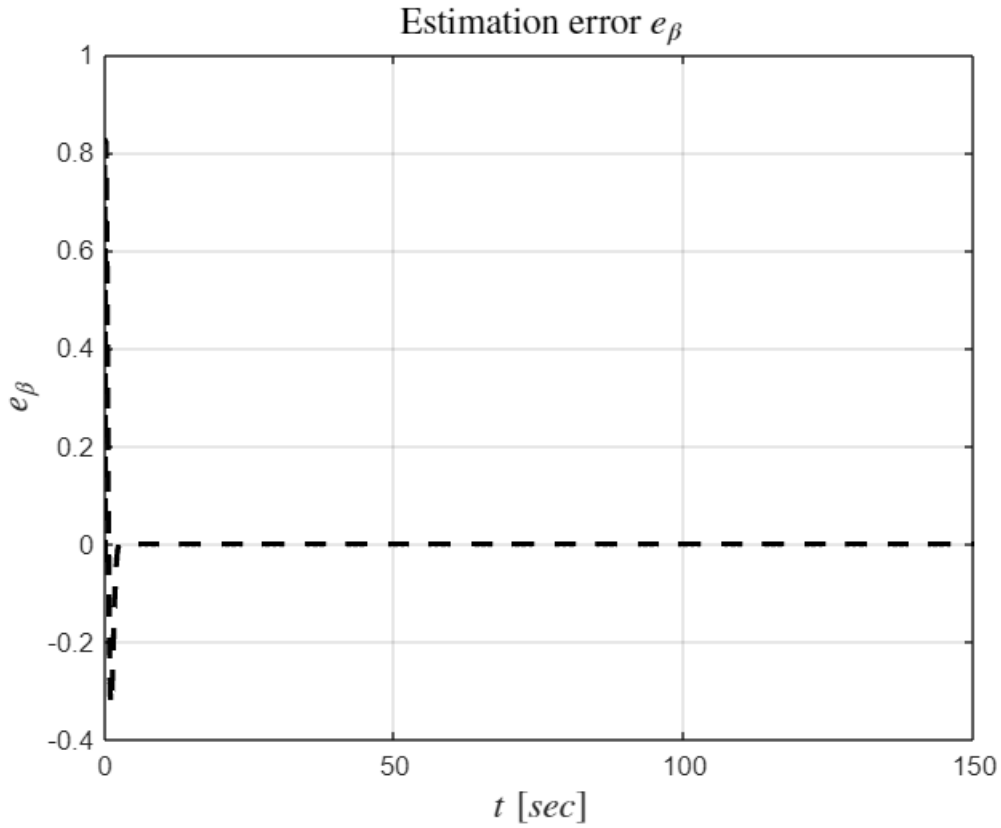
```
%% Integral control

% First augmenting the system with integrators
Fi=[F zeros(4,2); -C*Ts eye(2)];
Gi=[G; zeros(2,2)];
Ci=[C zeros(2,2)];

% Then designing the controllers
ev1=[-.5 -.6 -.5+1i -.5-1i -.7 -.7];
evd1=exp(ev1*Ts);
```

```
K1=place(Fi,Gi,evd1);
Kd=K1(:,1:4);
Ki=-K1(:,5:6);

% Evaluate solution
cl_eva1 = log(eig(Fi-Gi*K1))/Ts;

% Simulate
sim('obsvIntegralSimul');

%% Plots
t = states.time;
figure;
plot(t,states.signals.values(:,[1,5]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$\beta, \; \hat{\beta}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$\beta$', '$\hat{\beta}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $\beta$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```

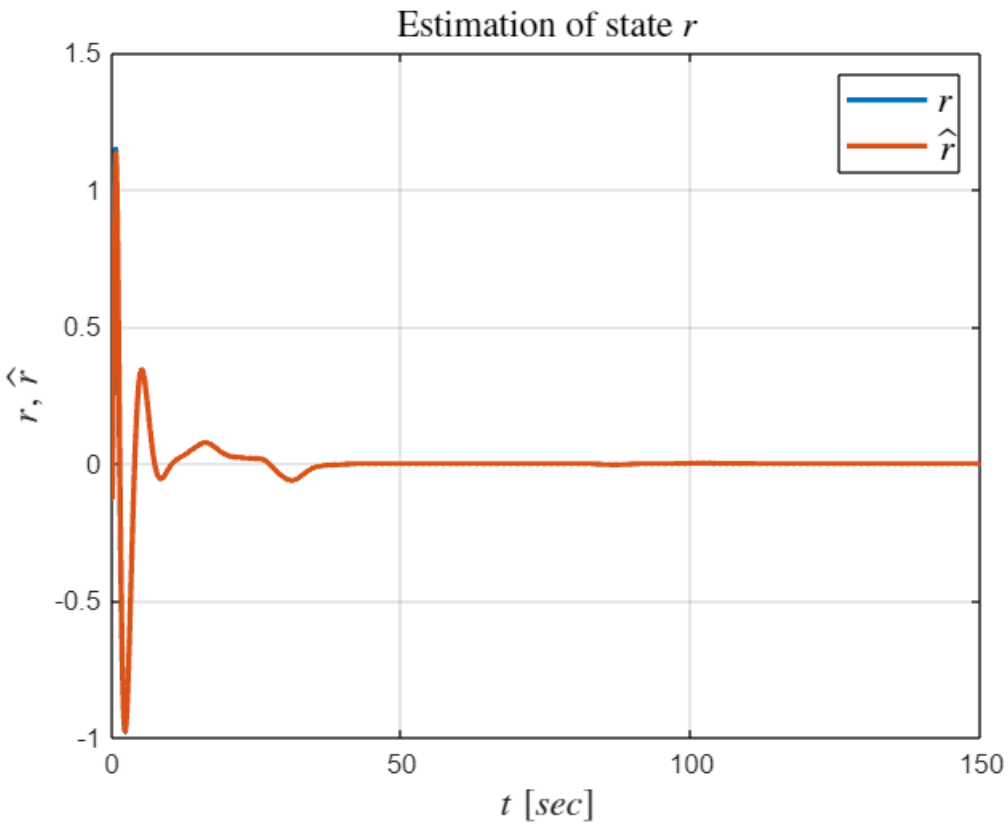



```
figure
plot(t,error.signals.values(:,1), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{\beta}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation error $e_{\beta}$', 'Interpreter', 'latex', 'FontSize',
14);
```

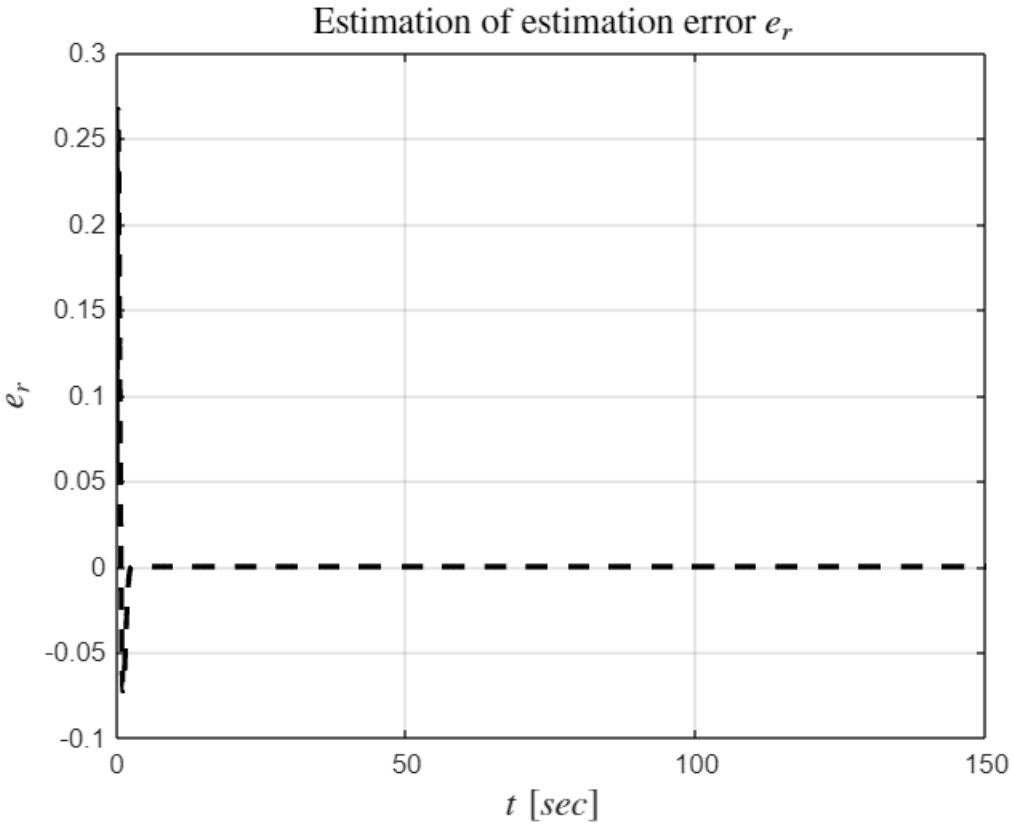
```
grid;
```



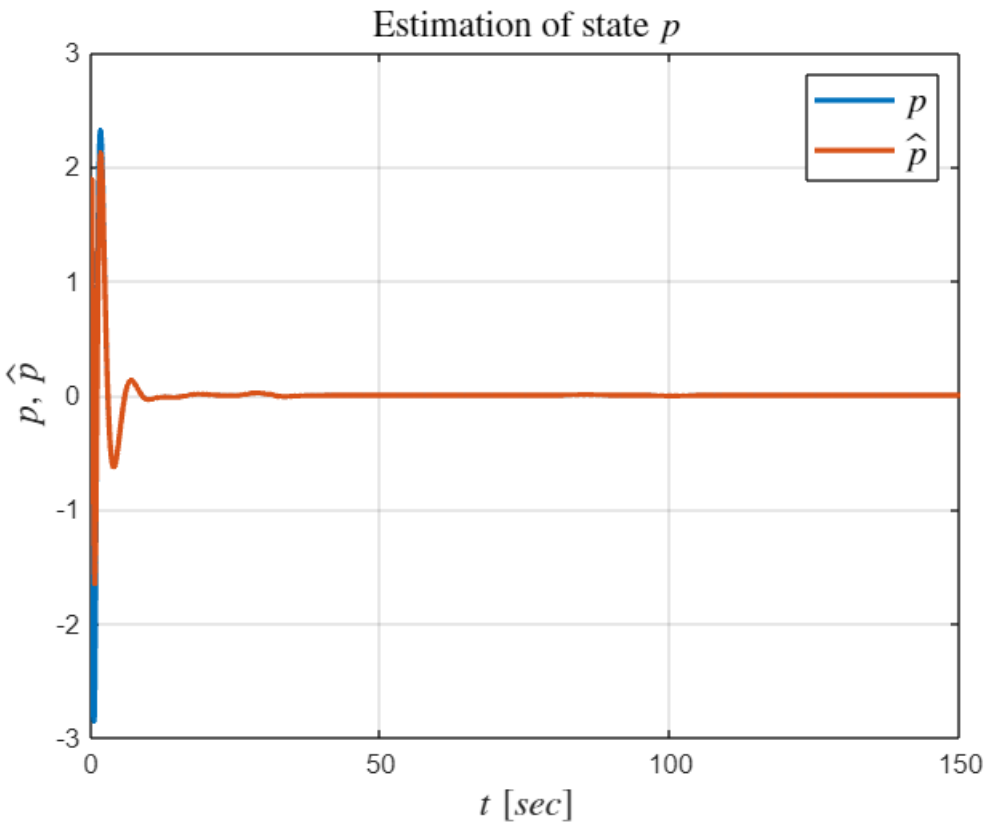

```
figure
plot(t,states.signals.values(:,[2,6]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$r, \; \hat{r}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$r$', '$\hat{r}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $r$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



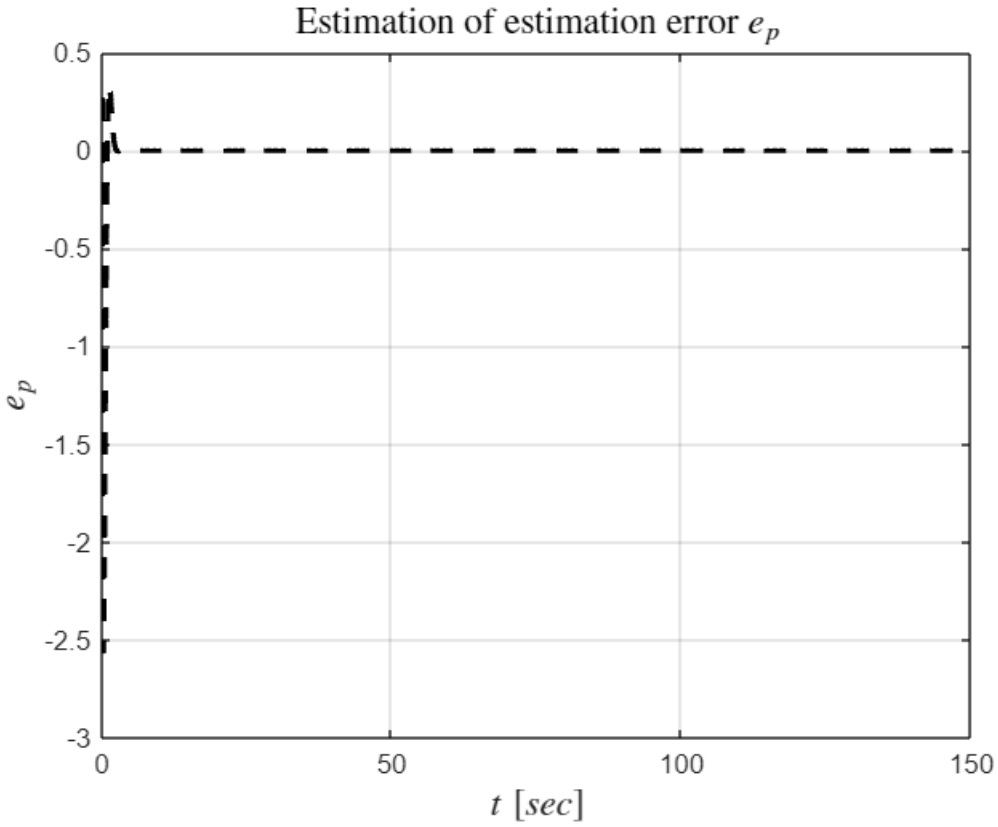

```
figure
plot(t,error.signals.values(:,2), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{r}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{r}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```



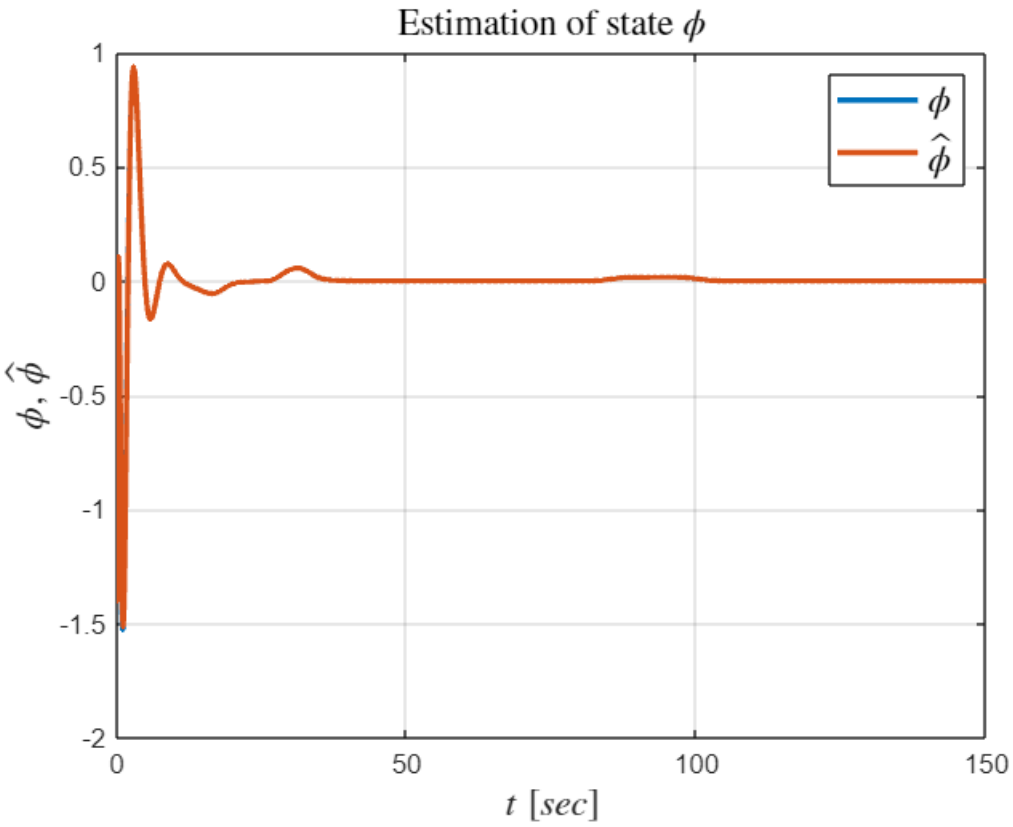

```
figure
plot(t,states.signals.values(:,[3,7]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$p, \; \hat{p}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$p$', '$\hat{p}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $p$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



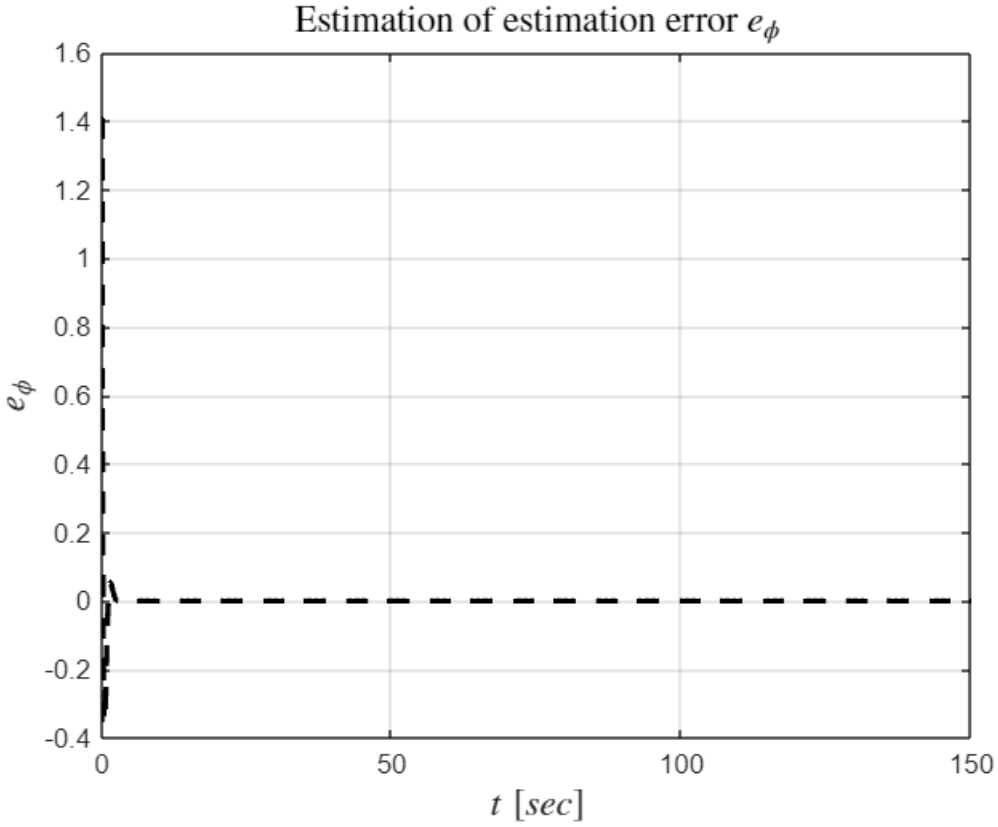

```
figure
plot(t,error.signals.values(:,3), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{p}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{p}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```



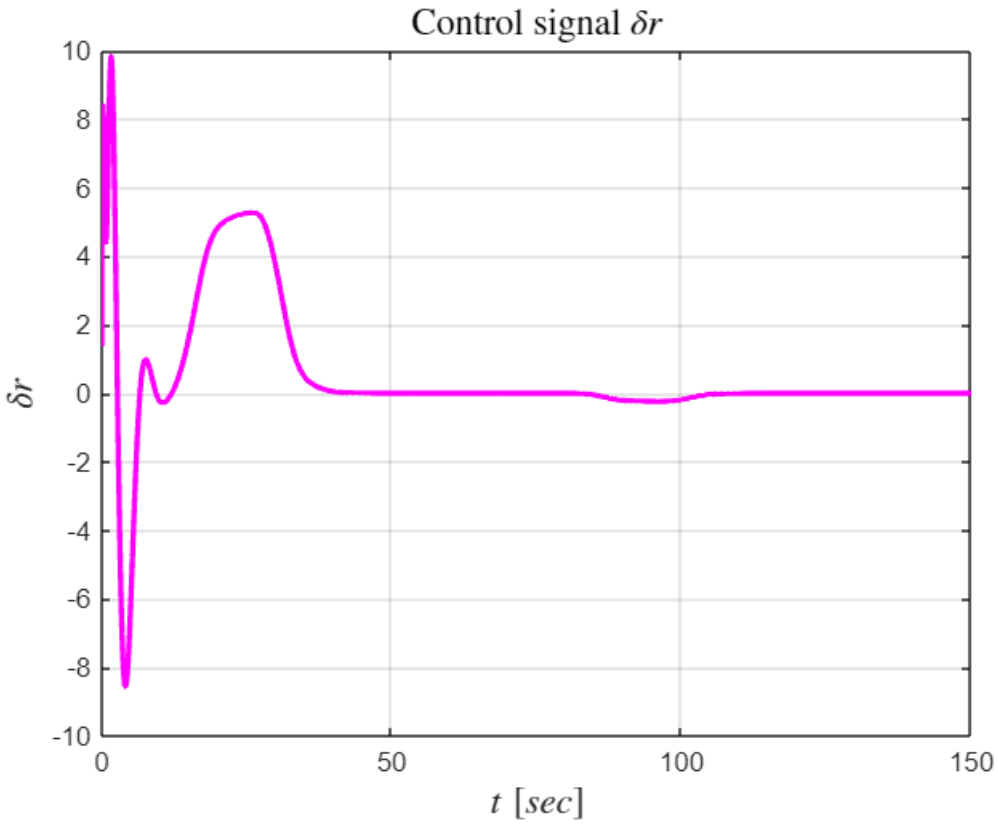

```
figure
plot(t,states.signals.values(:,[4,8]), 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$\phi, \; \hat{\phi}$', 'Interpreter', 'latex', 'FontSize', 14);
legend({'$\phi$', '$\hat{\phi}$'}, 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of state $\phi$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```



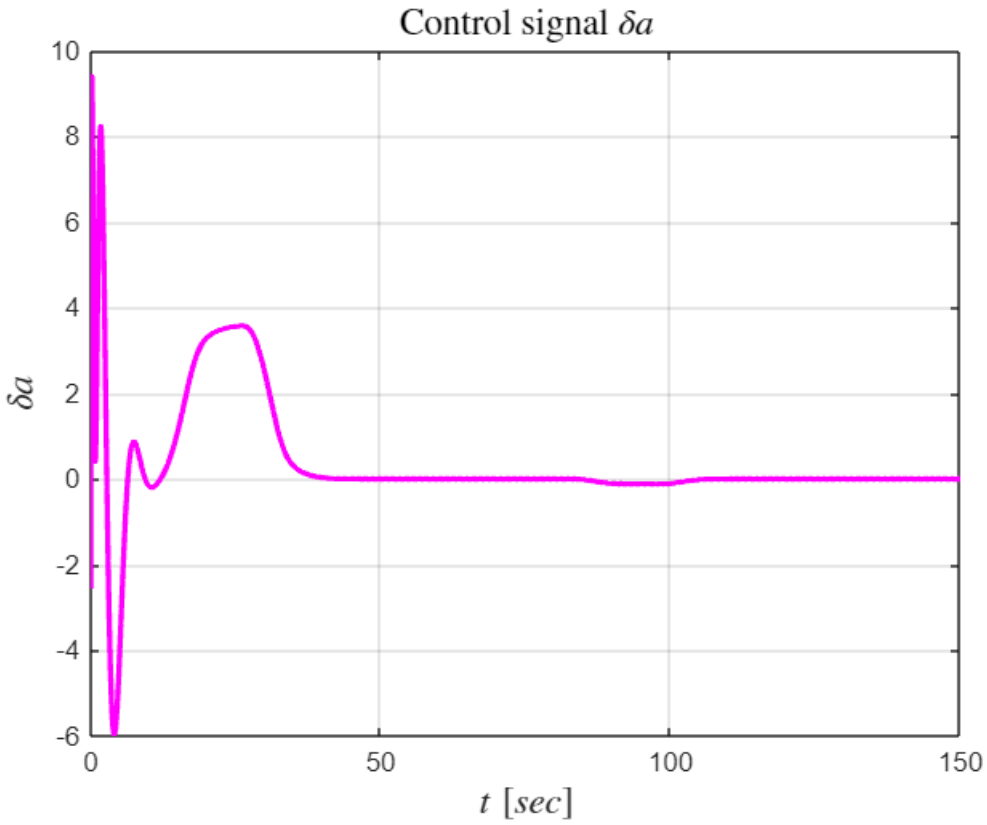

```
figure
plot(t,error.signals.values(:,4), '--k', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$e_{\phi}$', 'Interpreter', 'latex', 'FontSize', 14);
title('Estimation of estimation error $e_{\phi}$', 'Interpreter', 'latex',
'FontSize', 14);
grid;
```




```
t = control.time;
figure
plot(t,control.signals.values(:,1), 'm', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$\delta r$', 'Interpreter', 'latex', 'FontSize', 14);
title('Control signal $\delta r$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```




```
figure
plot(t,control.signals.values(:,2), 'm', 'LineWidth', 2);
xlabel('$t \; [sec]$', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('$\delta a$', 'Interpreter', 'latex', 'FontSize', 14);
title('Control signal $\delta a$', 'Interpreter', 'latex', 'FontSize', 14);
grid;
```


Control signal δa
10
8
6
4
2
0
-2
-4
-6
δa
0
50
100
150
t [sec]

Linear Control Design II - Group Work Problem Module 17

Description

The dynamics of a wind turbine operating in the full load region reads

$$J_r\dot{\omega}_r = \frac{1}{2}\rho A R C_P(\beta,\lambda)\frac{v^2}{\lambda} - K_d\theta - B_d\left(\omega_r - \frac{\omega_g}{N_g}\right)$$

$$J_g\dot{\omega}_g = \frac{K_d}{N_g}\theta + \frac{B_d}{N_g}\left(\omega_r - \frac{\omega_g}{N_g}\right) - T_g$$

$$\dot{\theta} = \omega_r - \frac{\omega_g}{N_g}$$

$$y_1 = \omega_r$$

$$y_2 = P_e = \omega_g T_g$$

where all quantities of interest are defined Group work Module 1. Based on the operating point and linearized model determined in Module 9 P1 and P2 address the following problems.

Problems

P1 Set the generator torque to its constant operating point, $T_g = T_{g,OP}$. Under the assumption that the system state is fully accessible (i.e. all state variables are available to the controller), design a discrete time full state feedback controller with integral action that uses the pitch angle β to regulate the rotor angular speed ω_r to its nominal value $\omega_{r,OP}$ despite variations in wind speed. The controller should assign eigenvalues such that

- the settling time of ω_r to step changes in wind speed of ± 2 m/s is between 0.5 and 1 second
- the response of ω_r to step changes in wind speed is not critically damped

P1.Solution

The system is single input single output because the generator torque is set to be at its constant operating value. It is assumed that the wind speed is piecewise constant around the steady state value. Hence only an integrator is needed.

```
clear all
close all
clc

% Get relevant parameters for the wind turbine. Still need to define x0 and
vOP in the Simulink model
load('relevantWindTurbineParameters.mat')
% Get the operating point and the linearized model from Module 10 P1 and P2
load('LinearModel.mat')
```

```
% Set Simulation Time
TIME_SIM = 25;

Dv = 2; %[m/s]

% Adapt the system matrices to match the requirements (SISO)
Bb = Bb(:,1); % Removing Tg
CcO = Cc; % Cc for problem 4
Cc = Cc(1,:); % Only measure wr
Dd = Dd(:,1);

% Find eigenvalues and asses asymptotical stability
lbd = eig(Aa)
```

```
lbd = 3×1 complex
  -0.2071 + 0.0000i
  -0.6797 +13.6692i
  -0.6797 -13.6692i
```

```
disp('The system in asymptotically stable')
```

```
The system in asymptotically stable
```

```
% time constants
tau = -1./lbd(imag(lbd)==0)
```

```
tau = 4.8291
```

```
% natural frequencies and damping ratios
wn = sqrt(real(lbd(2))^2+imag(lbd(2))^2)
```

```
wn = 13.6861
```

```
z = -real(lbd(2))/wn
```

```
z = 0.0497
```

```
% Assess Controllablity
Mc = ctrb(Aa,Bb);
if rank(Mc) == rank(Aa)
    disp('The system is controllable')
else
    disp('The system is not controllable')
end
```

```
The system is controllable
```

```
% Choosing a sampling time for the discrete controller based on openloop
% dynamics and closed loop requirements.
```

```
T = 2*pi/wn;
T_settling_min = 0.5;
tau_settling_min = 0.5/5; % assuming a first order system behaviour of the
closed loop response

Ts = min([tau(1)/10 T/10 tau_settling_min/10]) % sampling time
```

```
Ts = 0.0100
```

```


[Ff,Gg] = c2d(Aa,Bb,Ts) % Discretize the LTI continuous system
```

```
Ff = 3×3
    0.9960    0.0000   -0.1451
    2.0182    0.9792  165.8303
    0.0099   -0.0001    0.9907
Gg = 3×1
   -0.0110
   -0.0097
   -0.0001
```

```

```

For integral control, we have to augment the system first. Note that the equations (4.102)-(4.107) in the book has to be modified slightly, as the derivation assumes that Ts=1.

```

% Augment system matrices
Fau = [Ff zeros(size(Aa,1),size(Cc,1));
       -Cc*Ts eye(size(Cc,1))]
```

```
Fau = 4×4
    0.9960    0.0000   -0.1451         0
    2.0182    0.9792  165.8303         0
    0.0099   -0.0001    0.9907         0
   -0.0100         0         0    1.0000
```

```

% Augment Gg
Gau = [Gg;
       zeros(size(Gg,2),size(Cc,1))]
```

```
Gau = 4×1
   -0.0110
   -0.0097
   -0.0001
         0
```

```
% Assess Controllablity - We have to make sure that the new augmented
% system is fully controllable
Mc = ctrb(Fau,Gau);
```

```
if rank(Mc) == rank(Fau)
    disp('The system is controllable')
else
    disp('The system is not controllable')
end
```

```
The system is controllable
```

Now we can start making the controller with full state feedback and integral control described in chapter 5

```
%  ---- Desired characteristics for the closed loop system ----
% Not critically damped refers to 2nd order systems with damping above or
% below 1. We accept a little bit of overshoot but not to much thus damping
% is chosen to be 1/sqrt(2) < zeta < 1 - here zeta = 0.8 is chosen.

% The settling time should be between 0.5-1s for step changes in wind speed
% of +-2m/s. The speed of the response in a 2nd order system is related to
% the natural frequency and damping ratio: Tsettling = 4/(zeta*omega_n)
(approximately).

% We need to find 4 eigenvalues (3 for the original system and one for the
integrator) and the dominant complex pair
% are based on the design requirements and the others are chosen a bit
faster.

% First natural frequency and damping factor
Tsettling = 0.75; % Settling time
zeta_cl_1 = 0.65;

wn_cl_1 = 4/(zeta_cl_1*Tsettling);
% Corresponding eigenvalue (alpha +- j*beta)
alpha_1 = -wn_cl_1*zeta_cl_1;
beta_1 = sqrt(wn_cl_1^2 - alpha_1^2);

% Second natural frequency and damping factor
wn_cl_2 = wn_cl_1*1.2; % Choose the second natural frequency to be bigger
than the first one.
zeta_cl_2 = 0.8; %zeta_cl_1; % Choose the same damping here

% Corresponding eigenvalues
alpha_2 = -wn_cl_2*zeta_cl_2;
beta_2 = sqrt(wn_cl_2^2 - alpha_2^2);


% Collect desired closed loop eigenvalues for the continuous system
lambda_ct_des = [alpha_1+1i*beta_1 alpha_1-1i*beta_1 alpha_2+1i*beta_2
alpha_2-1i*beta_2];

% Convert to discrete eigenvalues
```

```
lambda_dt_des = exp(lambda_ct_des*Ts)
```

```
lambda_dt_des = 1×4 complex
   0.9462 + 0.0591i   0.9462 - 0.0591i   0.9226 + 0.0546i   0.9226 - 0.0546i
```

```
% Calculate gains for full state feedback.
% acker() because we have a SISO system. Calculate for non-augmented system.
K1 = acker(Fau, Gau, lambda_dt_des);
K_FSF = K1(1:3) % Gain for full state feedback
```

```
K_FSF = 1×3
  -20.4400    0.1119  -62.8482
```

```
Ki = -K1(4) % Gain for integrator
```

```
Ki = -30.2224
```

```
% Make sure that the closed loop system has stable eigenvalues (between 0
and 1).
eig(Fau-Gau*K1)
```

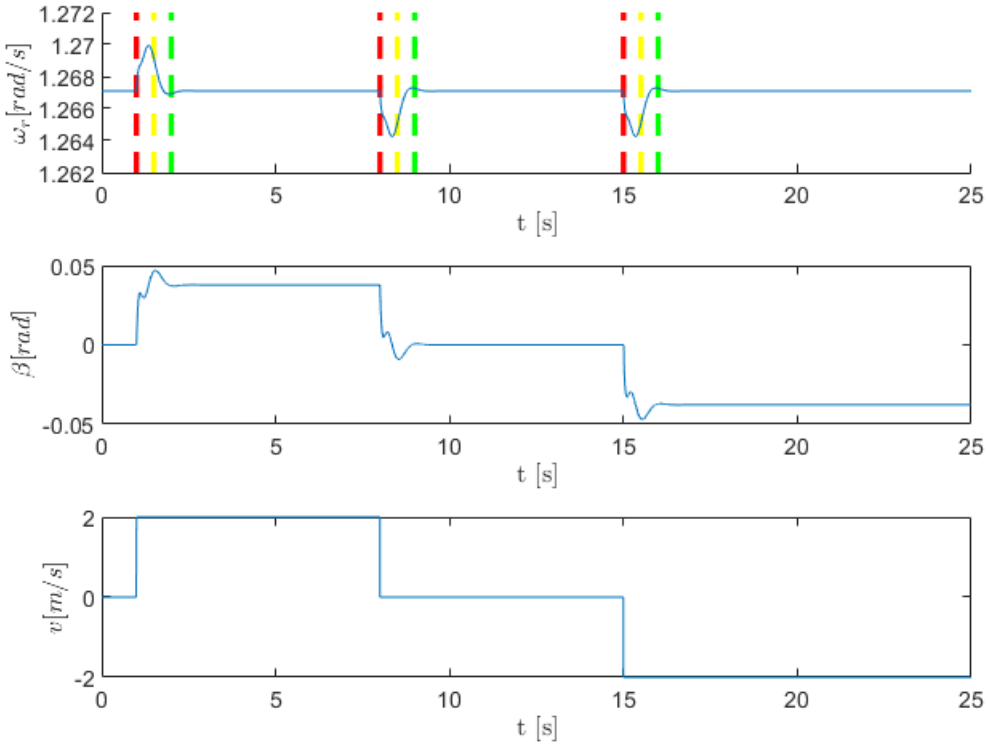
```
ans = 4×1 complex
   0.9226 + 0.0546i
   0.9226 - 0.0546i
   0.9462 + 0.0591i
   0.9462 - 0.0591i
```

P2 Implement the open loop linear system and the designed control system in Simulink. Evaluate the performance of your controller against the given requirements in response to wind speed step changes of ± 2 m/s.

Solution:

```
% Set Simulation Parameters

% Integrator step size in Simulink
STEP_SIZE = Ts/50;

% Sequence of disturbance i.e. windspeed changes
v_in = [zeros(floor(1/STEP_SIZE),1); Dv*ones(floor(7/STEP_SIZE),1);
zeros(floor(7/STEP_SIZE),1); -Dv*ones(floor(7/STEP_SIZE),1)];
t = 0:STEP_SIZE:TIME_SIM;

% Pad with -Dv at end
pad = length(t) - length(v_in);
v_in = [v_in; -Dv*ones(pad,1)];

% Find indiceses of windspeed changes and +4s and +6s (for plots)
```

```
stCh = find(diff(v_in) ~= 0);
ch1 = t(stCh(1));
ch14 = t(stCh(1)+floor(0.5/STEP_SIZE));
ch16 = t(stCh(1)+floor(1/STEP_SIZE));
ch2 = t(stCh(2));
ch24 = t(stCh(2)+floor(0.5/STEP_SIZE));
ch26 = t(stCh(2)+floor(1/STEP_SIZE));
ch3 = t(stCh(3));
ch34 = t(stCh(3)+floor(0.5/STEP_SIZE));
ch36 = t(stCh(3)+floor(1/STEP_SIZE));
w_OPup = w_OP*1.01;
w_OPlow = w_OP*0.99;

[~,~,yout] = sim('p2_linearWindTurbineModel',[0 TIME_SIM],[],[t' v_in]);

% Plot wr, windspeed and input
figure
subplot(3,1,1)

line([ch1 ch1],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
hold on
line([ch14 ch14],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch16 ch16],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
line([ch2 ch2],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch24 ch24],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch26 ch26],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
line([ch3 ch3],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch34 ch34],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch36 ch36],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
plot(t,yout(:,1))
ylim([1.262 1.272])
hold off
xlabel('t [s]','Interpreter','latex')
ylabel('$\omega_r [rad/s]$','Interpreter','latex')
subplot(3,1,2)
plot(t,yout(:,3))
hold on
hold off
xlabel('t [s]','Interpreter','latex')
ylabel('$\beta [rad]$','Interpreter','latex')
subplot(3,1,3)
plot(t,yout(:,2))
xlabel('t [s]','Interpreter','latex')
ylabel('$v [m/s]$','Interpreter','latex')
```



The red dashed lines marks the time instants where the step changes in the wind speed kick in. After 0.5 seconds transient ω_r settles back almost to the operating value (yellow dashed lines). After 1 seconds since the step change takes place ω_r is back to the required value. The response is also not critically damped but not very underdamped either which can be seen in the small overshoot between the yellow and green lines.

Plotting the input β gives a feeling of the physical feasability of the controller. When designing the controller one should always keep in mind the constraints introduced by the actuator.

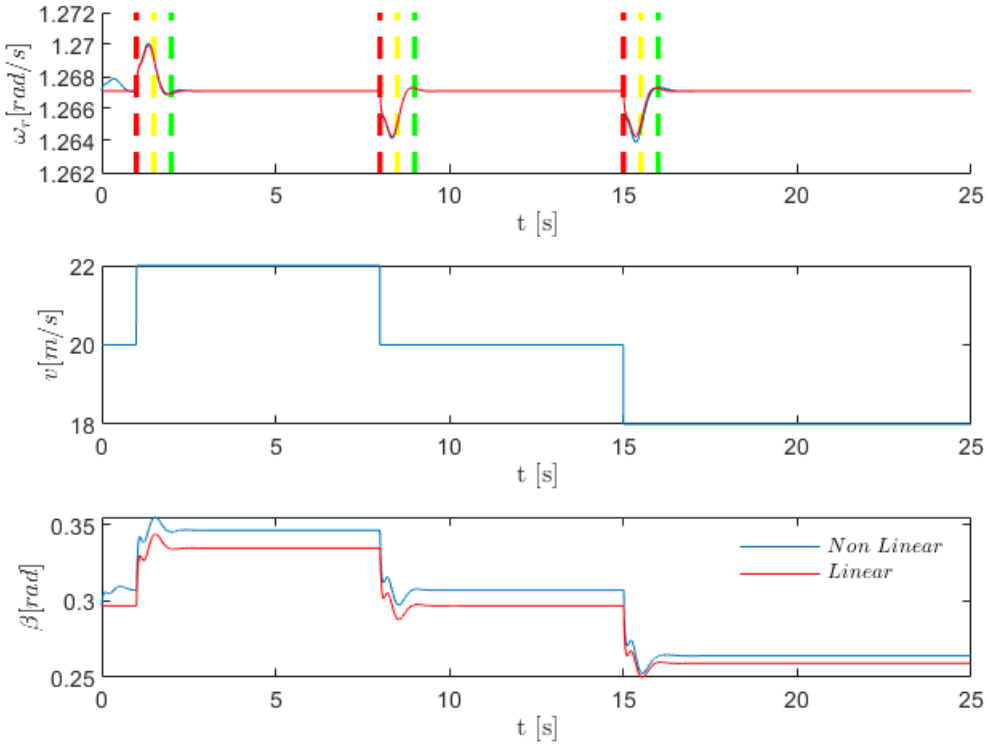
```
% Save yout structure to further plots
youtP2 = yout;
youtP2(:,2) = youtP2(:,2) + v_OP;
youtP2(:,3) = youtP2(:,3) + beta_OP;
```

P3 Implement the designed controller on the Simulink nonlinear model of the wind turbine provided in Module 9 and repeat the simulations performed in P2. Compare and discuss the performance of the control system on the linearized and nonlinear model.

Solution:

```
v_inNonLinear = v_in+v_OP;

[~,~,yout] = sim('p3_windTurbineModel.slx',[0 TIME_SIM],[],[t'
v_inNonLinear]);
youtP3 = yout;
```

```
% Plot wr, windspeed and input
figure
subplot(3,1,1)

line([ch1 ch1],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
hold on
line([ch14 ch14],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch16 ch16],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
line([ch2 ch2],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch24 ch24],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch26 ch26],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
line([ch3 ch3],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch34 ch34],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch36 ch36],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
plot(t,yout(:,1))
plot(t,youtP2(:,1),'Color','r')
hold off
xlabel('t [s]','Interpreter','latex')
ylabel('$\omega_r [rad/s]$','Interpreter','latex')
ylim([1.262 1.272])
subplot(3,1,2)
plot(t,yout(:,2))
xlabel('t [s]','Interpreter','latex')
ylabel('$v [m/s]$','Interpreter','latex')
subplot(3,1,3)
plot(t,yout(:,3))
hold on
plot(t,youtP2(:,3),'Color','r')
hold off
leg1 = legend('$Non \ Linear$','$Linear$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',8);
legend boxoff
xlabel('t [s]','Interpreter','latex')
ylabel('$\beta [rad]$','Interpreter','latex')
```



P4 Since only the rotor angular velocity and the electrical power are measured the controller designed in P1 cannot be implemented in reality unless an observer is also designed to estimate the unmeasured state variable. Design a discrete time full order observer that provides the state estimates of the measured and unmeasured state variables. The estimation error dynamics should settle to zero within 0.1 and 0.2 seconds.

Solution:

```
% The observer system estimates 3 states, which means that we have to
% design 3 eigenvalues with the desired characteristics for the estimation
% error dynamics
% We choose a complex pair of eigenvalues and a real eigenvalue

% For the real eigenvalue we just need to define the time constant tau
% based on the desired settling time, knowing that T_settling =~ 4*tau
t_settling_obs_1 = 0.15;
tau_obs_1 = t_settling_obs_1/4;
lambda_obs_1 = -1/tau_obs_1;

% For the complex pair, we can choose a slightly faster response,
% within the limits of the specifications
t_settling_obs_2 = 0.1;
% knowing that the settling time for complex eigenvalues can be
% approximated as T_settling = -4/alpha, we find
```

```
alpha_obs_2 = -4/t_settling_obs_2; %-4
% Now, we want the damping factor to be larger than 1/sqrt(2) =~ 0.7
% to avoid oscllatory behavior. We choose a value of 0.8
zeta_obs_2 = 0.8;
% With that, we can find the corresponding eigenvalue (alpha +- j*beta)
wn_obs_2 = -alpha_obs_2/zeta_obs_2;
beta_obs_2 = sqrt(wn_obs_2^2 - alpha_obs_2^2);

% Convert to discrete eigenvalues
lambda_ct_Obs_des = [lambda_obs_1 alpha_obs_2+1i*beta_obs_2
alpha_obs_2-1i*beta_obs_2];

lambda_dt_Obs_des = exp(lambda_ct_Obs_des*Ts)
```

```
lambda_dt_Obs_des = 1×3 complex
   0.7659 + 0.0000i   0.6404 + 0.1981i   0.6404 - 0.1981i
```

```
% Find Observer gain matrix
L = (place(Ff',CcO',lambda_dt_Obs_des)')
```

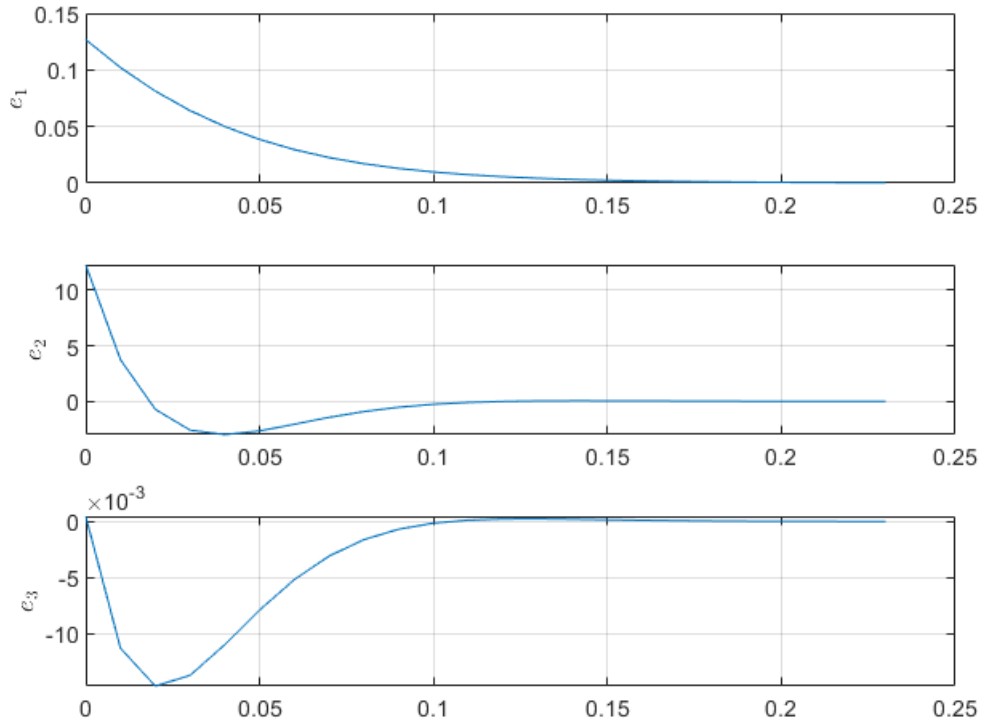
```
L = 3×2
    0.2307   -0.0097
    1.2545   16.9230
    0.0083    0.0215
```

```
% Make sure that the observer dynamics is stable and matches the desired one
check1 = eig(Ff-L*CcO)
```

```
check1 = 3×1 complex
   0.7659 + 0.0000i
   0.6404 + 0.1981i
   0.6404 - 0.1981i
```

```
% Check that the estimation error dynamics converges to zero within 0.1 s
and 0.2 s

tP4 = 0:Ts:1.5; %Simulate only for the needed amount of time

xe0 = 0.1*xOP;
Fe = Ff-L*CcO;
Ge = zeros(size(Ff,1),1);
Ce = eye(size(Ff,1));
De = Ge;
sys = ss(Fe,Ge,Ce,De,Ts);

[e,te,~] = initial(sys,xe0);
figure
subplot(3,1,1)
plot(te,e(:,1))
ylabel('$e_1$','Interpreter','latex')
grid on
subplot(3,1,2)
plot(te,e(:,2))
```

```
ylabel('$e_2$','Interpreter','latex')
grid on
subplot(3,1,3)
plot(te,e(:,3))
ylabel('$t \ [s]$','Interpreter','latex')
ylabel('$e_3$','Interpreter','latex')
grid on
```



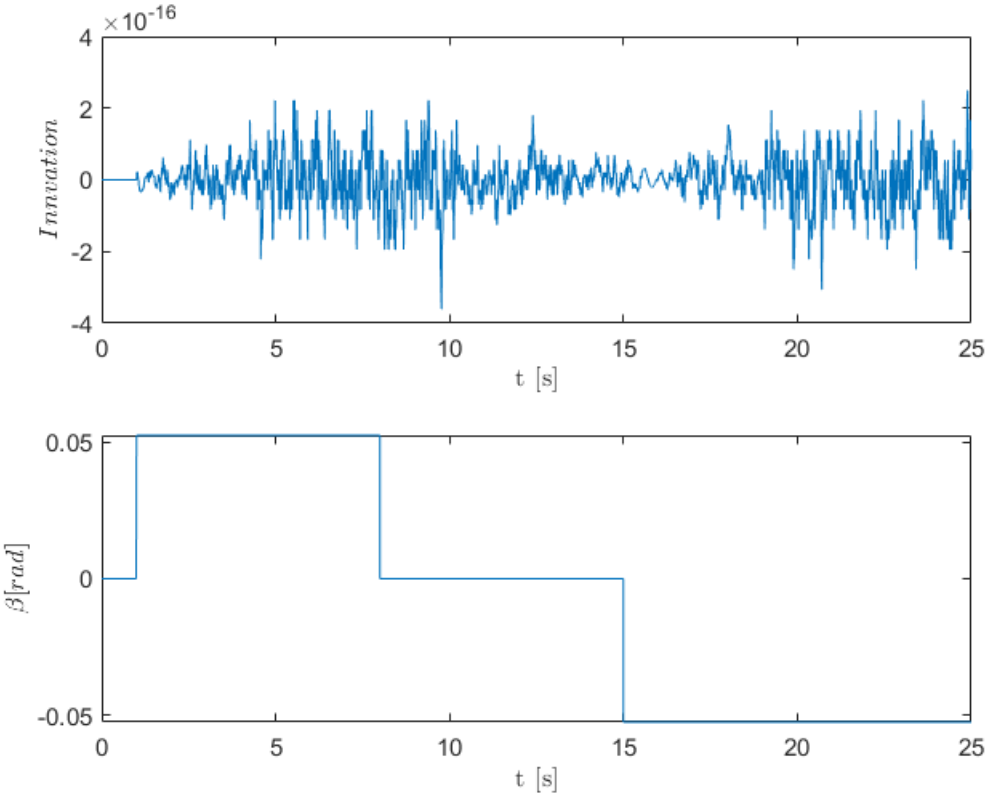
The plot above shows the estimation error zero-input response for 10 % initial bias in the states estimations. As required the estimation error settles to 0 within 0.1 and 0.2 s. Different design may lead to the same outcome.

P5 Implement the designed observer on the open loop linear model and evaluate its estimation performance around the operating point for step changes in pitch angle β of ± 3 deg .

Solution:

```
% Sequence of disturbance i.e. windspeed changes
DBeta = 3*pi/180; %[rad]
beta_in = [zeros(floor(1/STEP_SIZE),1); DBeta*ones(floor(7/STEP_SIZE),1);
zeros(floor(7/STEP_SIZE),1); -DBeta*ones(floor(7/STEP_SIZE),1)];

% Pad with -DBeta at end
pad = length(t) - length(beta_in);
beta_in = [beta_in; -DBeta*ones(pad,1)];

% Find indiceses of windspeed changes and +4s and +6s (for plots)
```

```
stCh = find(diff(beta_in) ~= 0);
ch1 = t(stCh(1));
ch14 = t(stCh(1)+floor(0.5/STEP_SIZE));
ch16 = t(stCh(1)+floor(1/STEP_SIZE));
ch2 = t(stCh(2));
ch24 = t(stCh(2)+floor(0.5/STEP_SIZE));
ch26 = t(stCh(2)+floor(1/STEP_SIZE));
ch3 = t(stCh(3));
ch34 = t(stCh(3)+floor(0.5/STEP_SIZE));
ch36 = t(stCh(3)+floor(1/STEP_SIZE));
w_OPup = w_OP+0.005;
w_OPlow = w_OP-0.005;

[~,~,yout] = sim('p5_linearWindTurbineModel',[0 TIME_SIM],[],[t' beta_in]);
% yout = [xh1, xh2, xh2, inn1, inn2, x1, x2, x3]

figure
subplot(2,1,1)
plot(t,yout(:,4))
xlabel('t [s]','Interpreter','latex')
ylabel('$Innvation$','Interpreter','latex')
subplot(2,1,2)
plot(t,beta_in)
xlabel('t [s]','Interpreter','latex')
ylabel('$\beta [rad]$','Interpreter','latex')
```

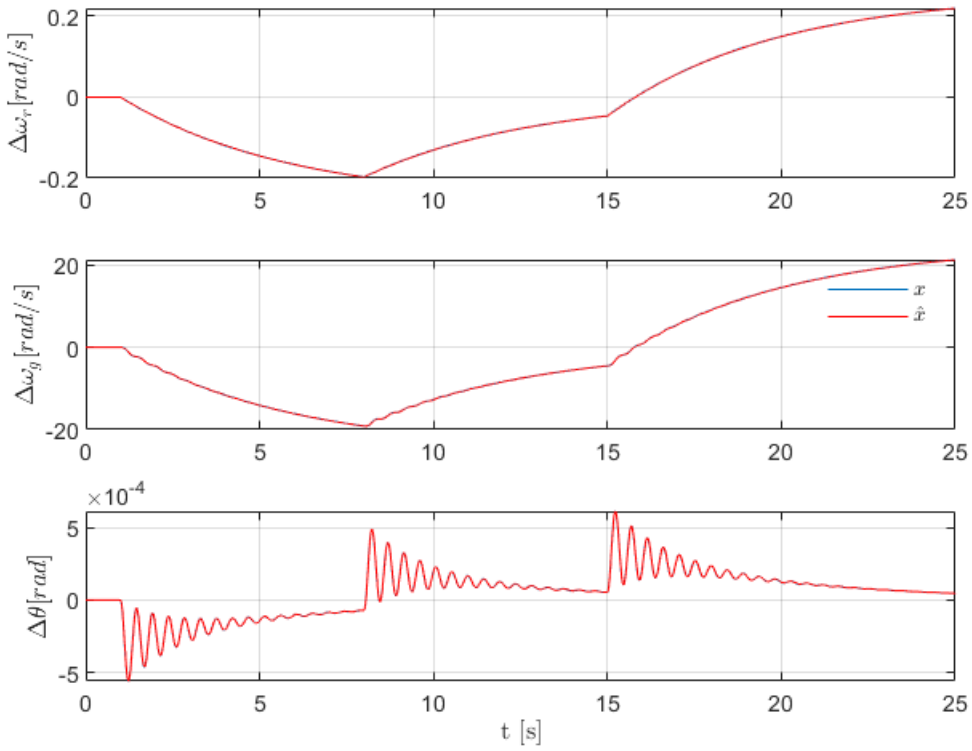


The innovations over ω_r measure the quality of the designed observer. The complete convergence occurs after 0.15 seconds and this fact might lead the reader thinking that the observer hasn't been correctly designed. Instead it is not the case since the innovations upper bound is in the order of 10^{-16} rad/s

If this profile (innovation) is compared against the response of θ in the next plot (scroll down), one can see that they have the same shape. The fast oscillatory behavior of θ creates small errors in the innovation signal.

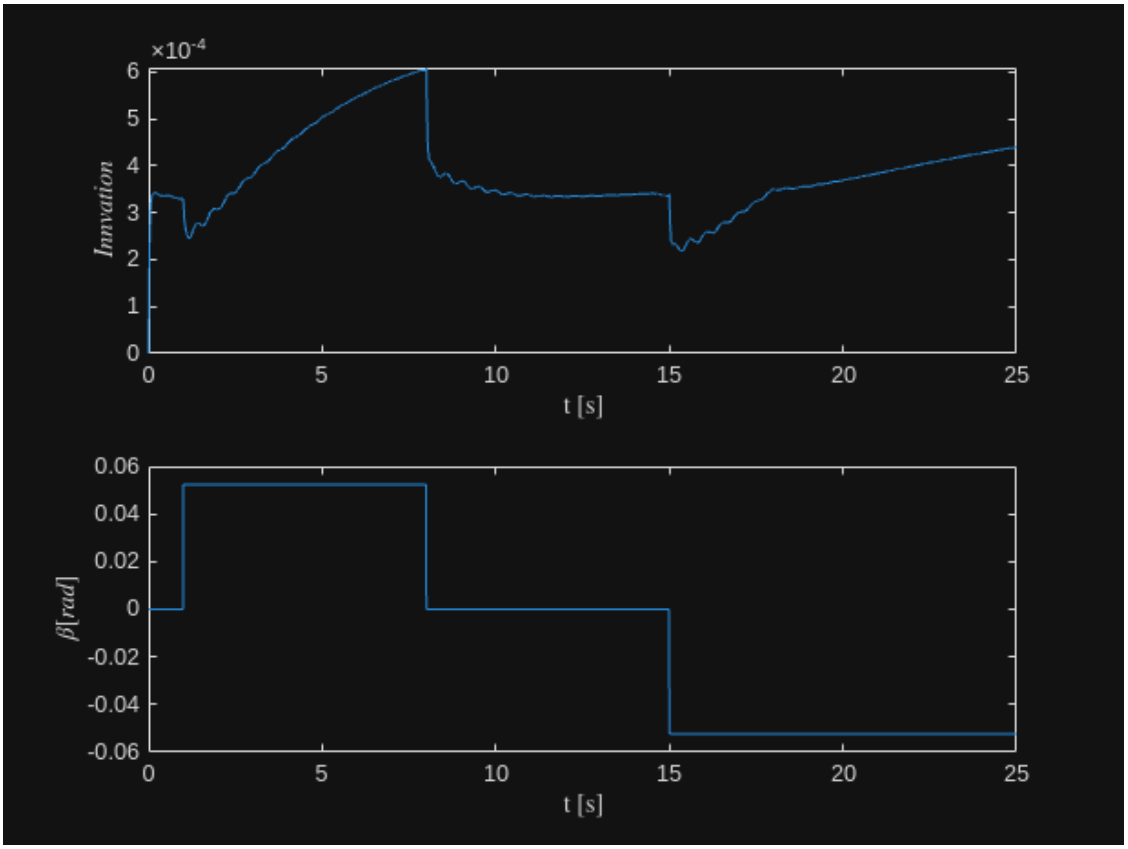
We can also visualize the true and estimated estates in the linear model and check that the observer converges to the true value

```
% Comparison of true and estimated states for the linear model
figure

subplot(3,1,1)
plot(t,yout(:,6))
hold on
plot(t,yout(:,1),'Color','r')
grid on
ylabel('$\Delta\omega_r [rad/s]$','Interpreter','latex')
hold off

subplot(3,1,2)
plot(t,yout(:,7))
hold on
plot(t,yout(:,2),'Color','r')
grid on
ylabel('$\Delta\omega_g [rad/s]$','Interpreter','latex')
hold off

leg1 = legend('$x$','$\hat{x}$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',8);
legend boxoff

subplot(3,1,3)
plot(t,yout(:,8))
hold on
plot(t,yout(:,3),'Color','r')
grid on
xlabel('t [s]','Interpreter','latex')
ylabel('$\Delta\theta [rad]$','Interpreter','latex')
hold off
```



Now let's simulate and plot the observer behavior on the non-linear model

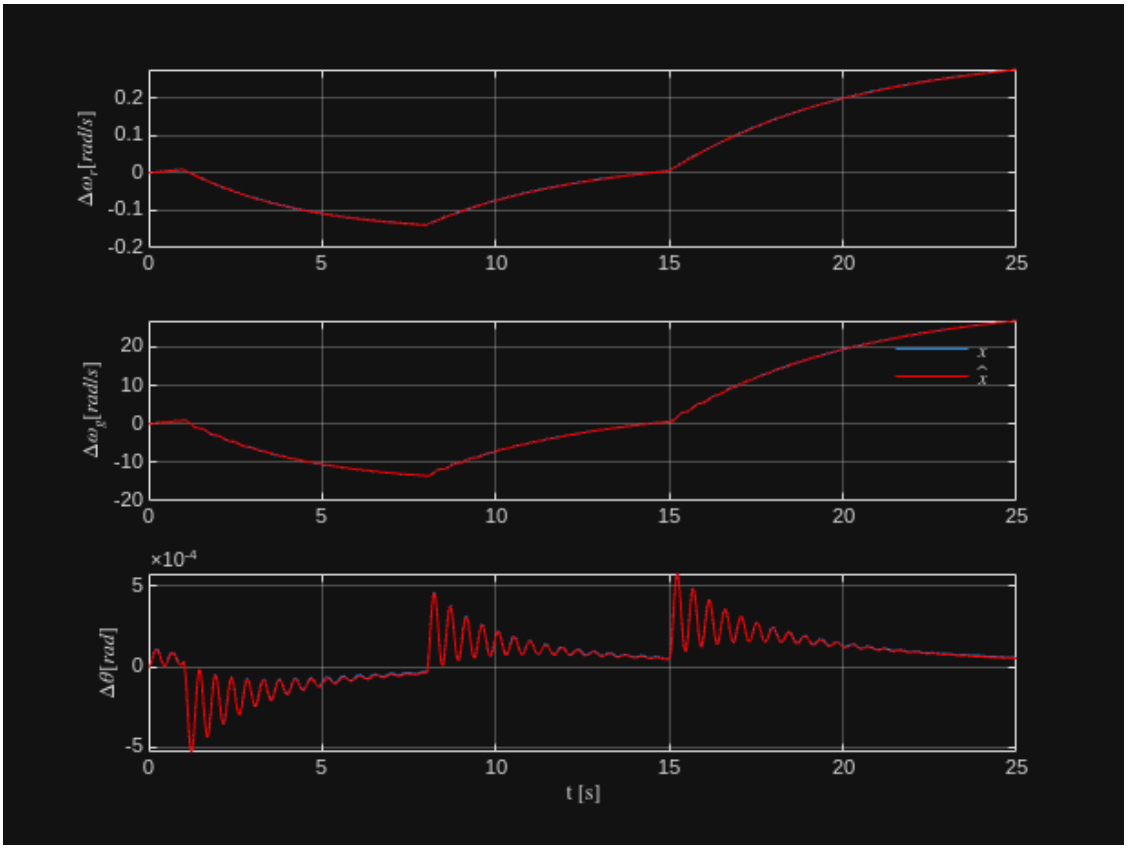
```
% Non-linear model
close all
[~,~,yout] = sim('p5_WindTurbineModel',[0 TIME_SIM],[],[t' beta_in]);
% yout = [xh1, xh2, xh2, inn1, inn2, w3, x1, x2, x3]
figure
subplot(2,1,1)
plot(t,yout(:,4))
xlabel('t [s]','Interpreter','latex')
ylabel('$Innvation$','Interpreter','latex')
subplot(2,1,2)
plot(t,beta_in)
xlabel('t [s]','Interpreter','latex')
ylabel('$\beta [rad]$','Interpreter','latex')
```



For the non-linear system a small offset is present in the innovation

```
% Comparison of true and estimated states
figure

subplot(3,1,1)
plot(t,yout(:,7))
hold on
plot(t,yout(:,1),'Color','r')
grid on
ylabel('$\Delta\omega_r [rad/s]$','Interpreter','latex')
hold off

subplot(3,1,2)
plot(t,yout(:,8))
hold on
plot(t,yout(:,2),'Color','r')
grid on
ylabel('$\Delta\omega_g [rad/s]$','Interpreter','latex')
hold off

leg1 = legend('$x$','$\hat{x}$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',8);
legend boxoff
```

```
subplot(3,1,3)
plot(t,yout(:,9))
hold on
plot(t,yout(:,3),'Color','r')
grid on
xlabel('t [s]','Interpreter','latex')
ylabel('$\Delta\theta [rad]$','Interpreter','latex')
hold off
```

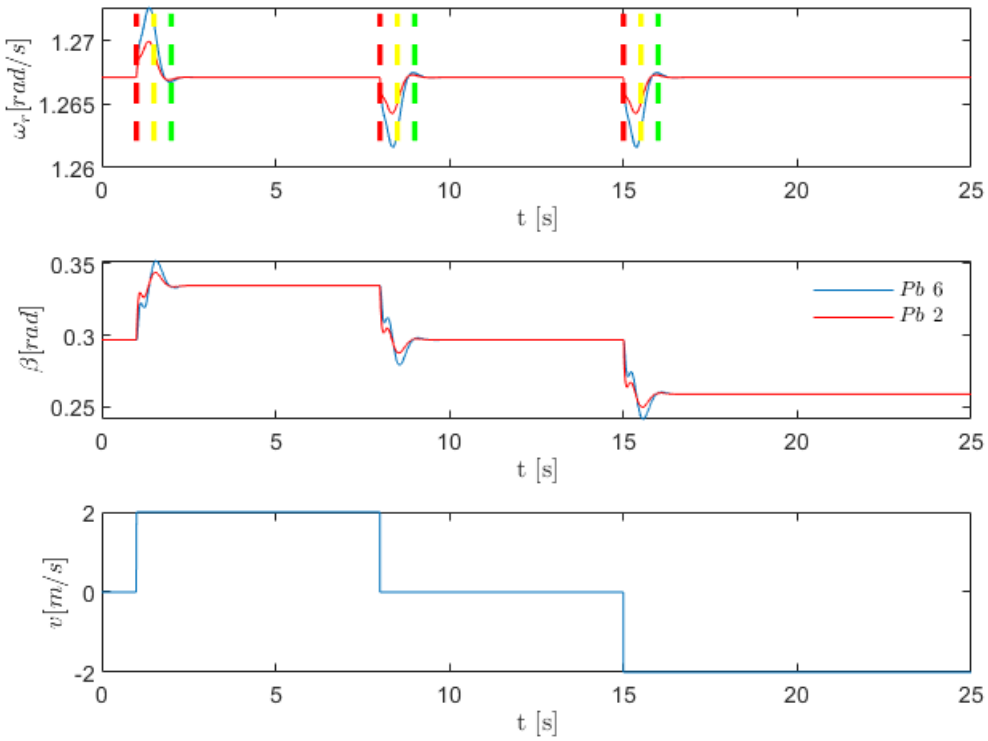


For the non-linear model, the observer still converges to the true states but with a small bias

P6 Now close the loop again around the linearized model by feeding the controller with the estimates provided by the observer and evaluate the control system performance against that obtained in P2.

```
close all
[~,~,yout] = sim('p6_linearWindTurbineModel',[0 TIME_SIM],[],[t' v_in]);
% Plot wr, windspeed and input
figure,
subplot(3,1,1)
plot(t,yout(:,1))
hold on
plot(t,youtP2(:,1),'Color','r')
line([ch1 ch1],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch14 ch14],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
```

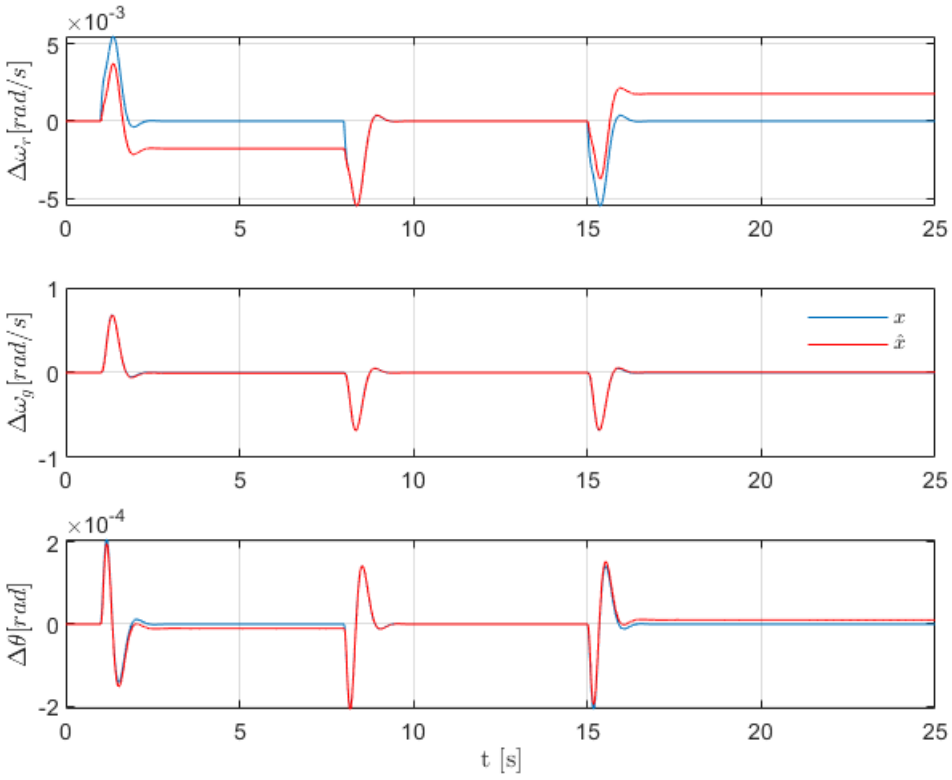
```
line([ch16 ch16],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
line([ch2 ch2],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch24 ch24],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch26 ch26],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
line([ch3 ch3],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch34 ch34],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch36 ch36],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
hold off
xlabel('t [s]','Interpreter','latex')
ylabel('$\omega_r [rad/s]$','Interpreter','latex')
subplot(3,1,2)
plot(t,yout(:,3)+beta_OP)
hold on
plot(t,youtP2(:,3),'Color','r')
hold off
xlabel('t [s]','Interpreter','latex')
ylabel('$\beta [rad]$','Interpreter','latex')
leg1 = legend('$Pb \ 6$','$Pb \ 2$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',8);
legend boxoff
subplot(3,1,3)
plot(t,yout(:,2))
xlabel('t [s]','Interpreter','latex')
ylabel('$v [m/s]$','Interpreter','latex')
```



Compared to **P2** the architecture has overall similar performance. Anyways in **P6** ω_r shows larger overshoots which require a slightly more intense control effort to compensate for. This is not surprising since the observer has no knowledge about the disturbance and needs the integrator compensating for it before it can return accurate states estimations.

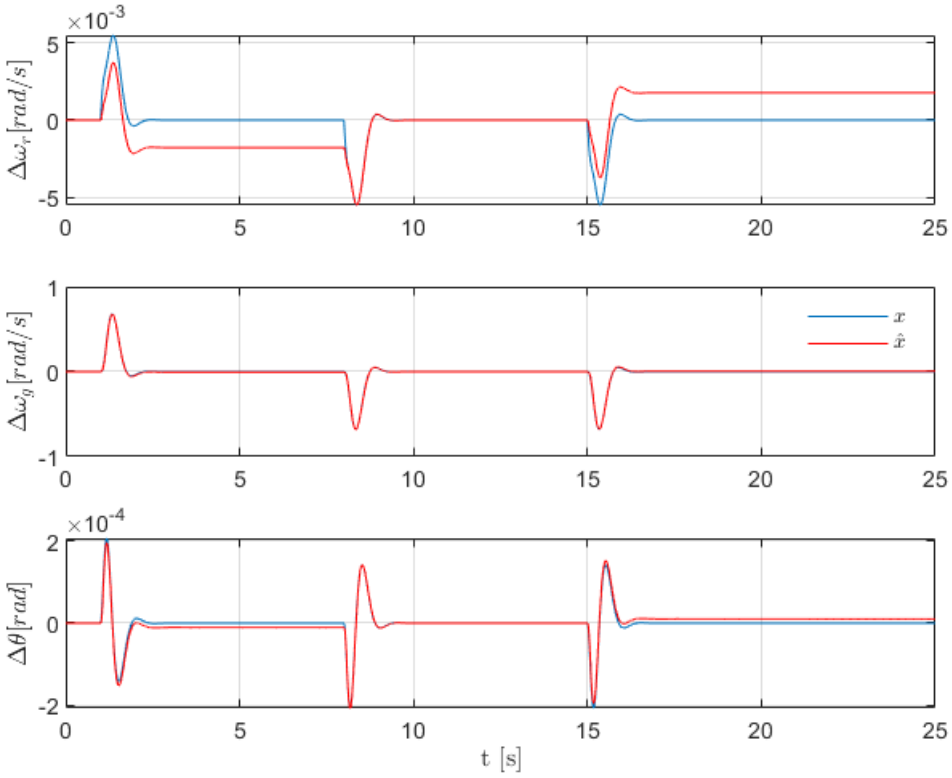
```
% Now compare the state estimates with the actual state

figure

subplot(3,1,1)
plot(t,yout(:,4))
hold on
plot(t,yout(:,7),'Color','r')
grid on
ylabel('$\Delta\omega_r [rad/s]$','Interpreter','latex')
hold off

subplot(3,1,2)
plot(t,yout(:,5))
hold on
plot(t,yout(:,8),'Color','r')
grid on
ylabel('$\Delta\omega_g [rad/s]$','Interpreter','latex')
hold off
```

```
leg1 = legend('$x$','$\hat{x}$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',8);
legend boxoff

subplot(3,1,3)
plot(t,yout(:,6))
hold on
plot(t,yout(:,9),'Color','r')
grid on
xlabel('t [s]','Interpreter','latex')
ylabel('$\Delta\theta [rad]$','Interpreter','latex')
hold off
```



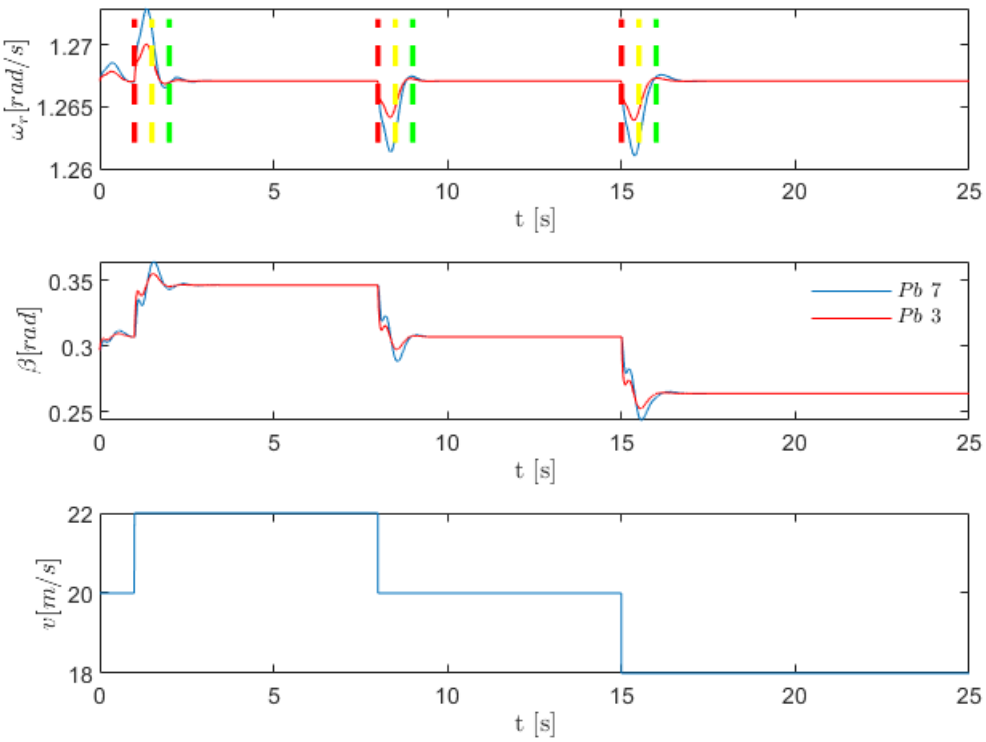
P7 Last, close the loop around the nonlinear system with the controller and observer and compare the performance against that obtained in P3

```
[~,~,yout] = sim('p7_windTurbineModel',[0 TIME_SIM],[],[t' v_inNonLinear]);
```




```
% Plot wr, windspeed and input
figure,
subplot(3,1,1)
plot(t,yout(:,1))
hold on
plot(t,youtP3(:,1),'Color','r')
line([ch1 ch1],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch14 ch14],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch16 ch16],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
line([ch2 ch2],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch24 ch24],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch26 ch26],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
line([ch3 ch3],[w_OPlow w_OPup],'Color','red','LineStyle','--','Linewidth',2)
line([ch34 ch34],[w_OPlow
w_OPup],'Color','yellow','LineStyle','--','Linewidth',2)
line([ch36 ch36],[w_OPlow
w_OPup],'Color','green','LineStyle','--','Linewidth',2)
hold off
xlabel('t [s]','Interpreter','latex')
ylabel('$\omega_r [rad/s]$','Interpreter','latex')
subplot(3,1,2)
plot(t,yout(:,3)+beta_OP)
```

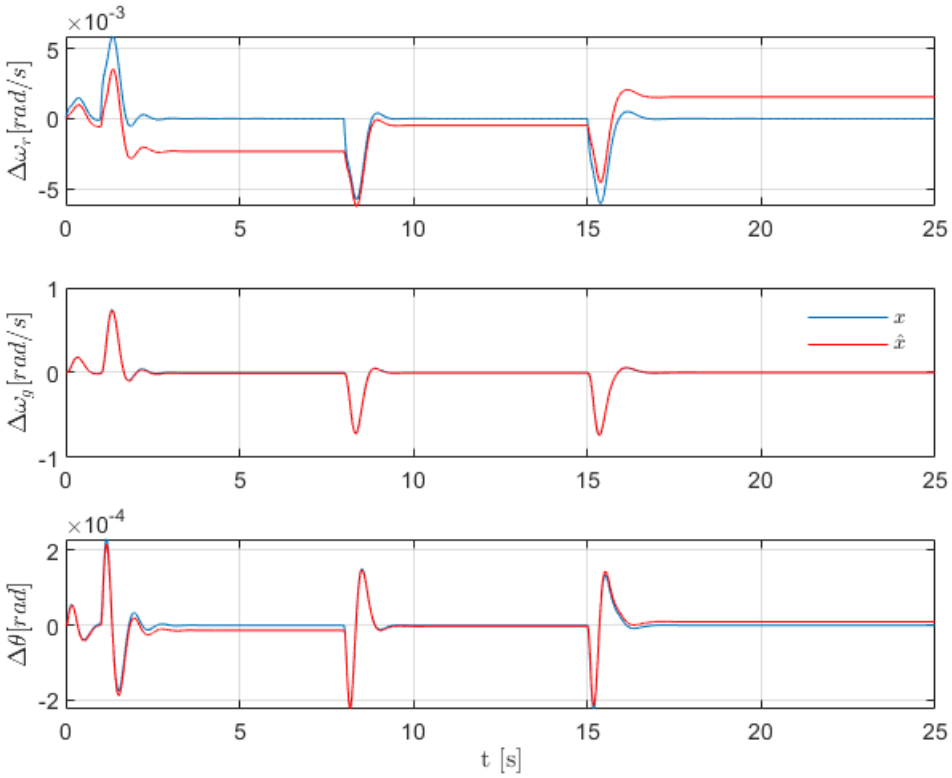
```
hold on
plot(t,youtP3(:,3),'Color','r')
hold off
leg1 = legend('$Pb \ 7$','$Pb \ 3$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',8);
legend boxoff
xlabel('t [s]','Interpreter','latex')
ylabel('$\beta [rad]$','Interpreter','latex')
subplot(3,1,3)
plot(t,yout(:,2))
xlabel('t [s]','Interpreter','latex')
ylabel('$v [m/s]$','Interpreter','latex')
```



Once again the observer based full state feedback with integral action performs slightly worse though it still respects the requirements. The reasons for performance degradation can be found on the unmodelled disturbance within the observer.

```
% Now compare the state estimates with the actual state

figure

subplot(3,1,1)
plot(t,yout(:,4))
hold on
plot(t,yout(:,7),'Color','r')
```

```
grid on
ylabel('$\Delta\omega_r [rad/s]$','Interpreter','latex')
hold off

subplot(3,1,2)
plot(t,yout(:,5))
hold on
plot(t,yout(:,8),'Color','r')
grid on
ylabel('$\Delta\omega_g [rad/s]$','Interpreter','latex')
hold off

leg1 = legend('$x$','$\hat{x}$');
set(leg1,'Interpreter','latex');
set(leg1,'FontSize',8);
legend boxoff

subplot(3,1,3)
plot(t,yout(:,6))
hold on
plot(t,yout(:,9),'Color','r')
grid on
xlabel('t [s]','Interpreter','latex')
ylabel('$\Delta\theta [rad]$','Interpreter','latex')
hold off
```



Linear Control Design II - Group Work Problem Module 18 Solution

Description

Problem 1 Consider the 2^{nd} order SISO system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$

where:

$$\mathbf{A} = \begin{bmatrix} -1 & 1 \\ 0 & 2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The optimal control law

$$u = -\mathbf{K}\mathbf{x}$$

is to be designed by minimizing the performance index J given by:

$$J = \int_0^\infty \left(\mathbf{x}^{\mathrm{T}}\mathbf{R}_1\mathbf{x} + u^{\mathrm{T}}\mathbf{R}_2 u\right) dt$$

Where:

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{R}_2 = [1]$$

1. Show analytically that the system cannot be stabilized by any gain matrix $\mathbf{K}$.
2. Draw a block diagram of the system and use it to explain your results.
3. Use the Matlab function `lqr` to design the optimal controller. Discuss the numerical results in relation to your analytical findings.

Solution:

By drawing the block diagram of the system or by calculating the controllability matrix of it like:

$$\mathbf{Q} = [\mathbf{B} \quad \mathbf{A}\mathbf{B}]$$

it can be seen that the system is not controllable. Thus, one cannot make an LQR (or any other state feedback regulator) for it.

Another approach is to try to find the eigenfrequencies of the closed loop system. Let $\mathbf{K}$ be the controller gain:

$$\mathbf{K} = [k_1 \quad k_2]$$

Then:

$$\det(\lambda \mathbf{I} - (\mathbf{A} - \mathbf{B}\mathbf{K})) = (\lambda + 1 + k_1)(\lambda - 2) = 0$$

which implies that the closed loop poles are at $\lambda_1 = -1 - k_1$ and $\lambda_2 = 2$. The second pole is the right half plane and cannot be moved by the controller i.e. its expression does not depend on k_1 or k_2. Thus the regulator is unstable no matter what the value of $\mathbf{K}$ is.

If one, for example, does try to make an LQR controller and uses the weighting matrices:

$$\mathbf{R_1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{R_2} = [1]$$

and Matlab function *K=lqr(A, B, R1, R2)* then the following result is obtained:

$$\mathbf{K} = [\text{NaN} \quad \text{NaN}]$$

and the warning *Matrix is singular to working precision*.

Problem 2 Consider the 2^{nd} order SISO system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$

where:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The optimal control law

$$u = -\mathbf{K}\mathbf{x}$$

is to be designed by minimizing the performance index J given by:

$$J = \int_0^\infty \left(\mathbf{x}^{\mathrm{T}}\mathbf{R}_1\mathbf{x} + u^{\mathrm{T}}\mathbf{R}_2 u\right) dt$$

where:

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{R}_2 = [1]$$

1. Determine the optimal feedback gain matrix $\mathbf{K}$ for this system.

Solution:

The optimal gain for this exercise is obtained by using the Matlab routine K=lqr(A, B, R1, R2). One can also, in this simple case, solve the Riccati equation directly:

$$\mathbf{A}^{\mathrm{T}}\mathbf{B} + \mathbf{PA} - \mathbf{PB}\mathbf{R}_2^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P} + \mathbf{R}_1 = 0$$

```
% System Matrices
A = [0 1;0 -1];
B = [0;1];
C = [1 1];
D = 0;

% Weighting Matrices
R1 = [1 0;0 1];
R2 = 1;

% Compute the optimal gain
K = lqr(A, B, R1,R2);
disp('Optimal gain matrix')
```

```
Optimal gain matrix
```

```
disp(K);
```

```
           1           1
```

```
[K,P,E] = lqr(A, B, R1,R2);
disp('Solution of the algebraic Riccati equation')
```

```
Solution of the algebraic Riccati equation
```

```
disp(P);
```

```
           2           1
           1           1
```

```
disp('Closed loop eigenvalues')
```

```
Closed loop eigenvalues
```

```
disp(E);
```

```
          -1
          -1
```

The solution is:

$$\mathbf{P} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

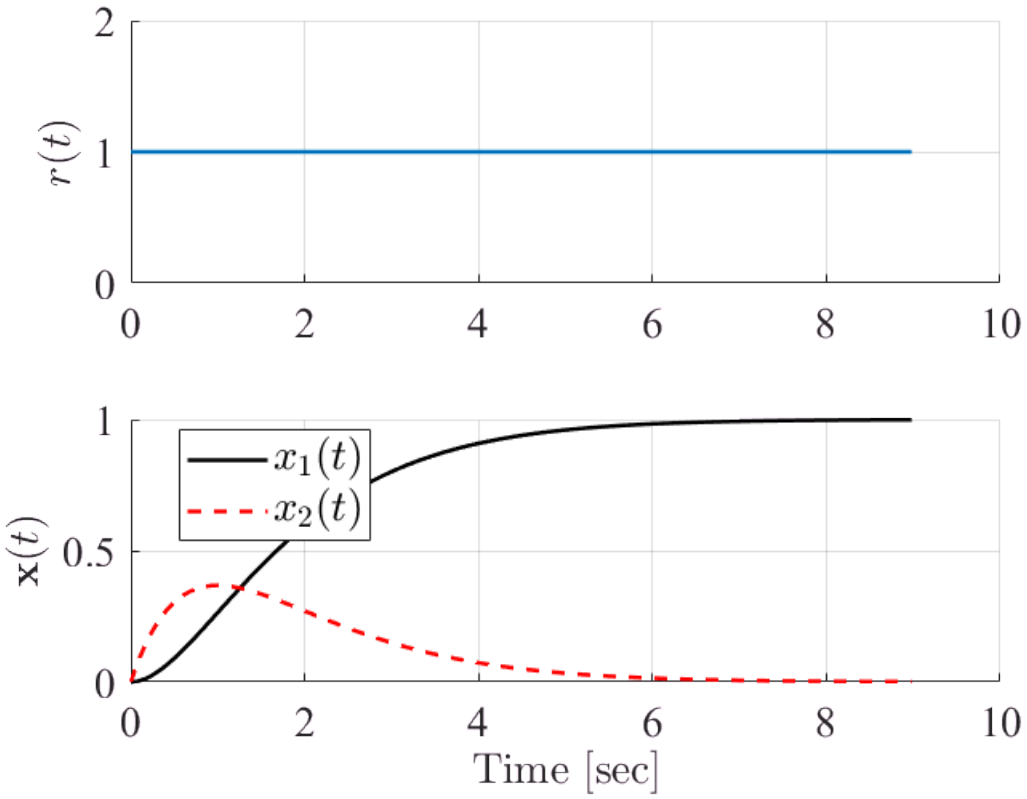
from which the optimal gain matrix $\mathbf{K}$ can be computed:

$$\mathbf{K} = [1][0 \quad 1]\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} = [1 \quad 1]$$

Explicitly, the optimal control feedback law is:

$$u = -\mathbf{K}x = -x_1 - x_2$$

```
% Plot state responses: connect feedback loop
Ak = A - B*K;
k1 = K(1);
k2 = K(2);
Bk = B*k2;

[y,x,t] = step(Ak,Bk,C,D);

figure, h1 = subplot(2,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
u = square(2*pi*0.05*t);
plot(t,u,'LineWidth',1.5);
ylabel('$r(t)$','FontName','times','FontSize',16,'Interpreter','latex')

h2 = subplot(2,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(t,[1 0]*x','-k',t,[0 1]*x','--r','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
ylabel('$\mathbf{x}(t)
$','FontName','times','FontSize',16,'Interpreter','latex')
l = legend('$x_1(t)$','$x_2(t)$','Location','Best');
set(l,'FontName','times','FontSize',16,'Interpreter','latex');
```



Problem 3 Consider the 3^{rd} order SISO system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$
$$y = \mathbf{C}\mathbf{x} + \mathbf{D}u$$

where:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{C} = [1 \quad 0 \quad 0] \qquad \mathbf{D} = [0]$$

The performance index J is given by:

$$J = \int_0^\infty \left(\mathbf{x}^{\mathrm{T}}\mathbf{R}_1\mathbf{x} + u^{\mathrm{T}}\mathbf{R}_2 u\right) dt$$

where:

$$\mathbf{R}_1 = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R}_2 = [0.01]$$

1. Why would one choose the matrix elements of the state weighting matrix in this way?

Assume that the following control law is designed

$$u = k_1(r - x_1) - (k_2x_2 + k_3x_3)$$

where r is the reference input to the system.

2. What control objective is fulfilled with the given feedback law?

3. Determine the optimal steady state feedback gain matrix $\mathbf{K}$ for this system.

4. Plot the step response of the closed loop system.

Solution:

The output of this system is state x_1, which is also the main objective of the control strategy. The other states are secondary. This is the reason for the structure of the control system and why the weighting matrix has been selected in the suggested way. Making the top left element of $\mathbf{R_1}$ equals 100 i.e. $\mathbf{R}_1(1,1) = 100$ ensures that this state has a faster response. Remember that the law for weighting the states is**:**

$$\mathbf{R_1}(i,i) = \frac{1}{(t_1 - t_0)\max\ [x_i(t)]^2}$$

In the problem case, $t_1 - t_0 = 1$. $x_i(t)$ is the state error with respect to the selected nominal operational point i.e. $\Delta x = x - x_0$. In this particular case, $x_0 = 0$ is the linearization point. If Δx_1 needs to be small then $\mathbf{R_1}(1,1)$ has to be large.

The optimal control matrix can be found using the Matlab routine *K = lqr(A, B, R1, R2)*:

```
% Determination of optimal gain for 2X2 system

% System Matrices
A = [0 1 0;0 0 1;0 -2 -3];
B = [0;0;1];
C = [1 0 0];
D = 0;

% Weighting Matrices
R1 = [100 0 0;0 1 0;0 0 1];
R2 = [0.01];

% Compute the optimal gain
K = lqr(A, B, R1,R2);
```

```
disp('Optimal gain matrix')
```

```
Optimal gain matrix
```

```
disp(K);
```

```
          100        53.12       11.671
```

$$\mathbf{K} = \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix} = [100 \quad 53.12 \quad 11.6711\,]$$

Equivalently one can use the Matlab command *[K P E] = lqr(A,B,C,D)*, where $\mathbf{P}$ is the matrix in the Riccati equation and $\mathbf{E}$ are the eigenvalues of the system.

```
[K,P,E] = lqr(A, B, R1,R2);
disp('Solution of the algebraic Riccati equation')
```

```
Solution of the algebraic Riccati equation
```

```
disp(P);
```

```
        55.12       14.671            1
       14.671       7.0267       0.5312
            1       0.5312      0.11671
```

```
disp('Closed loop eigenvalues')
```

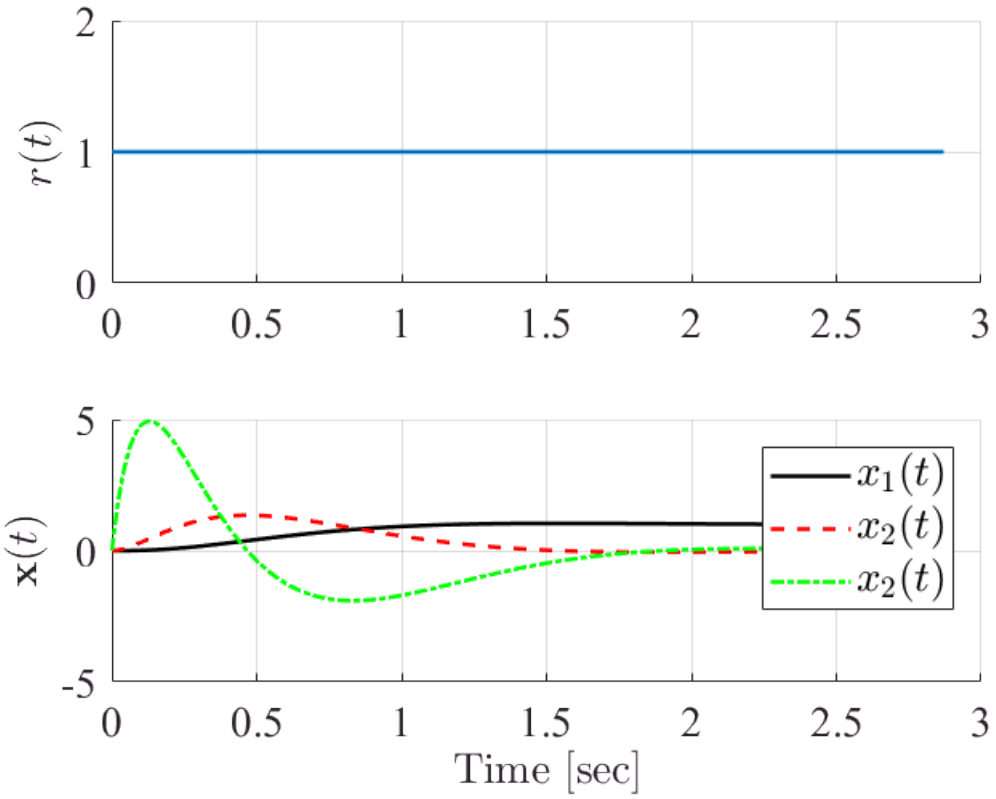
```
Closed loop eigenvalues
```

```
disp(E);
```

```
      -10.243 +          0i
      -2.2141 +     2.2047i
      -2.2141 -     2.2047i
```

```
% Plot state responses: connect feedback loop
Ak = A - B*K;
k1 = K(1);
k2 = K(2);
k3 = K(3);
Bk = B*k1;

[y,x,t] = step(Ak,Bk,C,D);

figure, h1 = subplot(2,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
u = square(2*pi*0.05*t);
plot(t,u,'LineWidth',1.5);
ylabel('$r(t)$','FontName','times','FontSize',16,'Interpreter','latex')

h2 = subplot(2,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(t,[1 0 0]*x','-k',t,[0 1 0]*x','--r',t,[0 0 1]*x','-.g','LineWidth',1.5)
```

```
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
ylabel('$\mathbf{x}(t)
$','FontName','times','FontSize',16,'Interpreter','latex')
l = legend('$x_1(t)$','$x_2(t)$','$x_2(t)$','Location','NorthEast');
set(l,'FontName','times','FontSize',16,'Interpreter','latex');
```



The optimal LQR gains are optimal independent of how they are used in the regulator. Thus the unusual configuration of the control system configuration used in this problem has no effect on the optimal gains found: an optimal solution is optimal.

The step response can be found using the Matlab command *[x, y, t] = step(AK, Bk1, C, D)* where *AK=A-B*K* and *Bk1 = B*k1*. The last statement ensures that the input and feedback to the system is via the state x_1, which is that state that is to be controlled most accurately.

Linear Control Design II - Group Work Problem Module 19 Solution

Part A - Time Dependent Discrete LQR Control: first order system.

The purpose of this exercise is to illustrate in detail what how a time dependent optimal controller is designed for a simple first order system. Consider a discrete time controller for a system given by:

$$x(k+1) = 0.3679\ x(k) + 0.6321\ u(k)$$

with the boundary condition that $x(0) = 1$. For this system find an optimal controller to minimizes the performance index:

$$J = \frac{1}{2}[x(10)^2] + \frac{1}{2}\sum_{k=0}^{9}[x^2(k) + u^2(k)]$$

Note that in this example the weighting matrices have been selected to be:

$$\mathbf{S} = 1 \qquad \mathbf{R_1} = 1 \qquad \mathbf{R_2} = 1$$

The minimum value of the index J is also to be determined.

To solve this problem it is required that it be solved backwards from the final time. The boundary condition on the matrix $\mathbf{P}(N)$ is:

$$\mathbf{P}(N) = \mathbf{S} \Rightarrow \mathbf{P}(N) = \mathbf{P}(10) = \mathbf{S} = 1$$

The Riccati equation has to be solved in order to calculate the time varying optimal gain. In this case it can be written as:

$$\mathbf{P}(k) = 1 + 0.3679 \cdot \mathbf{P}(k+1)[1 + 0.6321 \cdot 1 \cdot 0.6321 \cdot \mathbf{P}(k+1)]^{-1}0.3679$$

which can be simplified to:

$$\mathbf{P}(k) = 1 + 0.1354 \cdot \mathbf{P}(k+1)[1 + 0.3996 \cdot \mathbf{P}(k+1)]^{-1}$$

One can now calculate backwards to the desired start time:

$$\mathbf{P}(9) = 1 + 0.1354 \cdot 1 \cdot [1 + 0.3996 \cdot 1]^{-1} = 1.0967$$
$$\mathbf{P}(8) = \ldots$$

Problem 1 Write a Matlab program to carry though these calculations to find $\mathbf{P}(0)$. What is especially noticeable about the values of $\mathbf{P}(k)$ during this process? What is the steady state value of $\mathbf{P}(k)$ i.e. $\mathbf{P}_{SS}$? Note that this value must be positive as $\mathbf{P}(k)$ must be positive definite.

Calculate now the feedback gains necessary (at the different sampling times) to accomplish the desired control.

Solution:

We can find the values of $\mathbf{P}(k)$ by running the recursion

$$\mathbf{P}(k-1) = 1 + 0.1354 \cdot \mathbf{P}(k)[1 + 0.3996 \cdot \mathbf{P}(k)]^{-1}$$

with the boundary condition $\mathbf{P}(10) = \mathbf{S} = 1$

Matlab program to carry though the calculations to find $\mathbf{P}(k)\ \forall k \in [1..10]$:

```
% Make row-vector with the elements P(k) (k=[10..1]).
S = 1;
P(11) = S;
for i=10:-1:1
    P(i) = 1 + 0.1354*P(i+1)*(1 + 0.3996*P(i+1))^(-1);
end
P
```

```
P =
    1.1037    1.1037    1.1037    1.1037    1.1037    1.1037    1.1037    1.1037 ···
```

We note that $\mathbf{P}(8) < \mathbf{P}(9) < \mathbf{P}(10)$ etc. (This actually holds all the way down to $k = 0$, but the change in $\mathbf{P}$ is insignificant).

We can find the steady state value of $\mathbf{P}(k)$, $\mathbf{P}_{SS}$ either by solving the equation:

$$\mathbf{P}_{SS} = 1 + 0.1354 \cdot \mathbf{P}_{SS}[1 + 0.3996 \cdot \mathbf{P}_{SS}]^{-1}, \qquad \mathbf{P}_{SS} > 0$$

Or we can simply note that the value of $\mathbf{P}(0)$ and $\mathbf{P}(1)$ are equal to within 5 decimals. Thus

$$\mathbf{P}_{SS} = 1.1037$$

```
% Steady state solution (you can choose any of the first seven values)
Pss = P(1);
disp('The steady state value of the Riccati equation is:'); disp([' Pss = '
num2str(Pss)]);
```

```
The steady state value of the Riccati equation is:
 Pss = 1.1037
```

Problem 2 The optimal control law is:

$$u(k) = -\mathbf{K}(k)x(k)$$

Use this law to calculate the states $x(1), x(2), x(3), \ldots, x(9)$. What is the sequence of inputs necessary to give this state response?

Solution:

The optimal control law is:

$$u(k) = -\mathbf{K}(k)x(k)$$

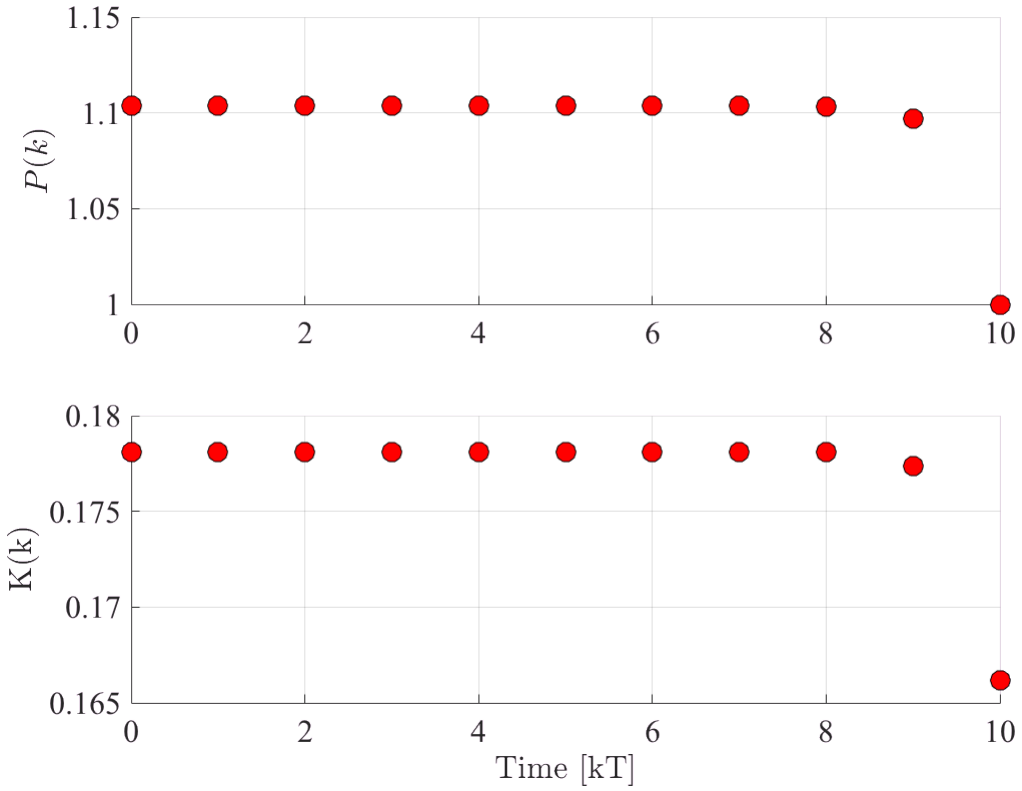
from the book Eq. (5.116) page 331 we have:

$$\mathbf{K}(k) = (\mathbf{R}_2 + \mathbf{G}^T\mathbf{P}(k)\mathbf{G})^{-1}\mathbf{G}^T\mathbf{P}(k)\mathbf{F}$$

where $\mathbf{G} = 0.6321$ and $\mathbf{F} = 0.3679$

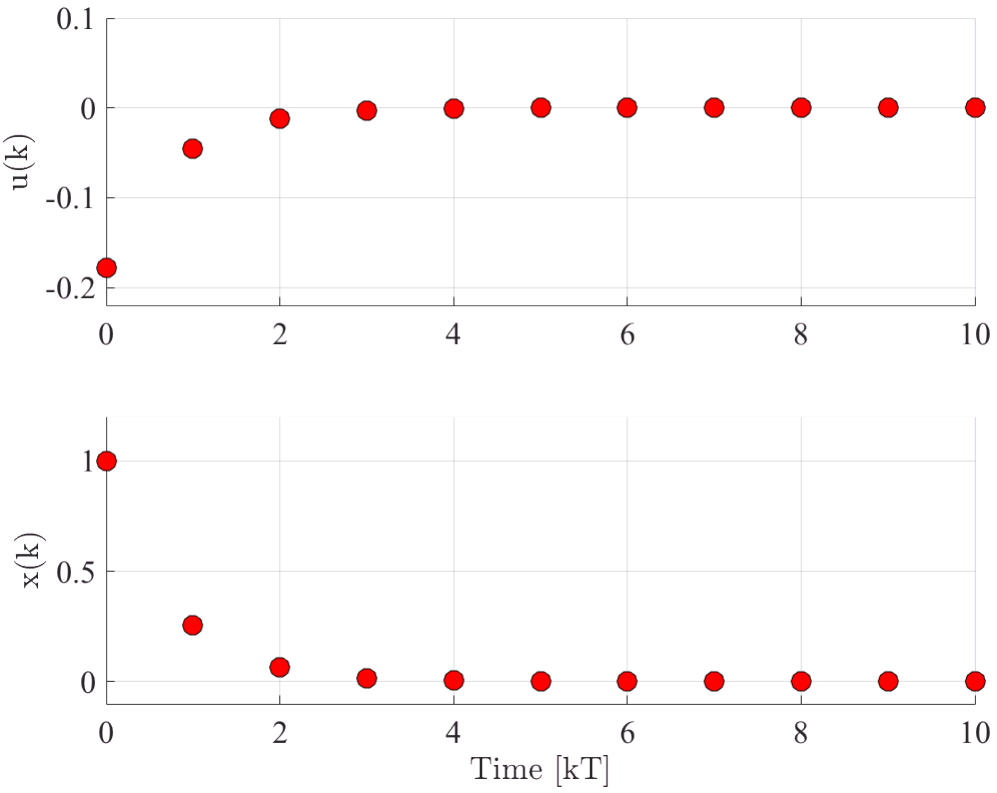
Using the initial condition $x(0) = 1$ and knowing that $x(k+1) = \mathbf{F}x(k) + \mathbf{G}u(k)$, we can calculate the input and state sequences:

```
F = 0.3679;
G = 0.6321;
R1 = 1;
R2 = 1;

% calculate values for u(k) and x(k), note that k=i-1:
x(1) = 1; % initial condition x(0) = 1
for i=1:10,
    K(i) = (R2+G'*P(i)*G)^(-1)*G'*P(i)*F; % Eq. (5.116) page 331
    u(i) = -K(i)*x(i); % Eq. (5.115) page 331
    x(i+1) = F*x(i) + G*u(i); % Time increment
end
i = 11;
K(i) = (R2+G'*P(i)*G)^(-1)*G'*P(i)*F; % Eq. (5.116) page 331
u(i) = -K(i)*x(i);

%Plotting
figure, h1 = subplot(2,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
i = 0:10;
% v = [0 9 1 1.2]; axis(v);
plot(i,P,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',10)
ylabel('$P(k)$','FontName','times','FontSize',16,'Interpreter','latex');

h2 = subplot(2,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
% v = [0 9 0 0.2]; axis(v);
plot(i,K,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',10)
ylabel('K(k)','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('Time [kT]','FontName','times','FontSize',16,'Interpreter','latex')
```




```
G = 0.6321;
R1 = 1;
R2 = 1;

% calculate values for u(k) and x(k), note that k=i-1:
x(1) = 1; % initial condition x(0) = 1
for i=1:10,
    K(i) = (R2+G'*P(i)*G)^(-1)*G'*P(i)*F; % Eq. (5.116) page 331
    u(i) = -K(i)*x(i); % Eq. (5.115) page 331
    x(i+1) = F*x(i) + G*u(i); % Time increment
end
i = 11;
K(i) = (R2+G'*P(i)*G)^(-1)*G'*P(i)*F; % Eq. (5.116) page 331
u(i) = -K(i)*x(i);

%Plotting
figure, h1 = subplot(2,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
i = 0:10;
% v = [0 9 1 1.2]; axis(v);
plot(i,P,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',10)
ylabel('$P(k)$','FontName','times','FontSize',16,'Interpreter','latex');

h2 = subplot(2,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
% v = [0 9 0 0.2]; axis(v);
```

```
plot(i,K,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',10)
ylabel('K(k)','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('Time [kT]','FontName','times','FontSize',16,'Interpreter','latex')

figure, h3 = subplot(2,1,1); set(h3,'FontName','times','FontSize',16)
hold on, grid on
v = [0 10 -0.22 0.1]; axis(v);
plot(i,u,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',10)
ylabel('u(k)','FontName','times','FontSize',16,'Interpreter','latex');

h4 = subplot(2,1,2); set(h4,'FontName','times','FontSize',16)
hold on, grid on
v = [0 10 -0.1 1.2]; axis(v);
plot(i,x,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',10)
ylabel('x(k)','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('Time [kT]','FontName','times','FontSize',16,'Interpreter','latex')
```



Problem 3 What is the minimum value of the performance index $J_{\min} = \frac{1}{2} x^{\mathrm{T}}(0)\mathbf{P}(0)x(0)$?

Solution:

We can either calculate the minimum value of the performance index by computing

$J_{min} = \frac{1}{2}x(10)^2 + \sum_{k=0}^{9}[x(k)^2 + u(k)^2]$

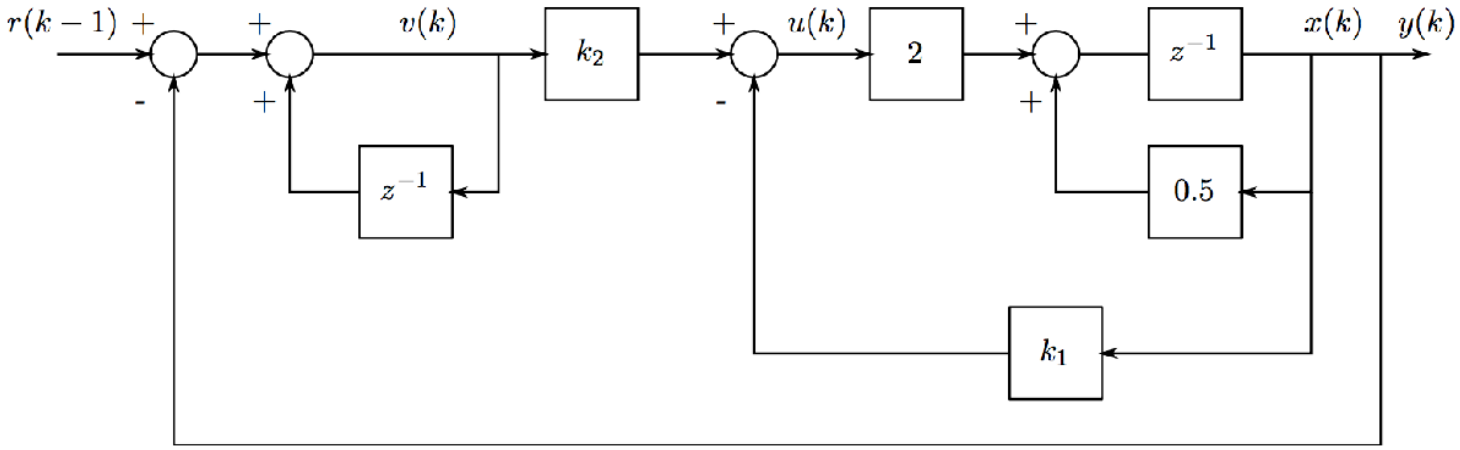
as we know the optimal $u(k)$ and resulting $x(k)$, or we can simply calculate $J_{min} = \frac{1}{2}x(0)^T\mathbf{P}(0)x(0)$:

```
Jmin = 0.5*x(1)'*P(1)*x(1);
disp('The minimum value of the performance index is'); disp(Jmin);
```

```
The minimum value of the performance index is
    0.5519
```

Part B - An integral LQR Regulator for a Servo System.

It is desired to design a steady state optimal controller for the servo loop in the Figure below:



The sample period for the loop is $T = 0.1\text{s}$. Clearly, k_1 is the integral gain and k_2 is the proportional gain.

Problem 1 Write down the state equations of the system, including the inputs and states. Show that the state equations of the system are:

$$\begin{bmatrix} x(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} 2 \\ -2 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k)$$

Solution:

We are interested in the state equations of the open loop system, so ignore k_1, k_2 for the time being.

We read of the state equations from the block diagram:

$$\begin{aligned} x(k+1) &= 2u(k) + 0.5x(k) \\ v(k) \quad &= v(k-1) + r(k-1) - y(k) = v(k-1) + r(k-1) - x(k) \end{aligned}$$

note that the time index of v does not mach that of x, so we time shift and get:

$$x(k+1) = 2u(k) + 0.5x(k)$$
$$v(k+1) = v(k) + r(k) - x(k+1) = v(k) + r(k) - 2u(k) - 0.5x(k)$$

in matrix form this is:

$$\begin{bmatrix} x(k+1) \\ v(k+1) \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} 2 \\ -2 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k)$$

Problem 2 Write down the state equations for the system when $t \to \infty$. The reference input is to be held constant.

Solution:

Assuming that we find an input $u(k)$ that can stabilise the system, as $k \to \infty$ the system will reach the steady state:

$$\begin{bmatrix} x(\infty) \\ v(\infty) \end{bmatrix} = \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x(\infty) \\ v(\infty) \end{bmatrix} + \begin{bmatrix} 2 \\ -2 \end{bmatrix} u(\infty) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r$$

By solving for $x(\infty)$ and $u(\infty)$ we get:

$$x(\infty) = r, \qquad u(\infty) = \frac{r}{4}$$

Note that we do not have any information about $v(\infty)$. This makes sense.

While $v(k+1) - v(k) \to -0.5x(\infty) - 2u(\infty) = 0$ as $k \to \infty$ the value $v(\infty)$ is dependant on the starting conditions of x and v.

Problem 3 Define the new incremental variables:

$$x_e(k) = x(k) - x(\infty)$$
$$v_e(k) = v(k) - v(\infty)$$
$$u_e(k) = u(k) - u(\infty)$$

Now write down the state equations of the system in terms of the new variables above. What is the new feedback law?

Solution:

We define new variables:

$$
\begin{aligned}
x_e(k) &= x(k) - x(\infty) \\
v_e(k) &= v(k) - v(\infty) \\
u_e(k) &= u(k) - u(\infty)
\end{aligned}
$$

Note that x_e becomes the difference between the state and the reference.

$$
\begin{aligned}
\begin{bmatrix} x_e(k+1) \\ v_e(k+1) \end{bmatrix} &= \begin{bmatrix} x(k+1) \\ v(k+1) \end{bmatrix} - \begin{bmatrix} x(\infty) \\ v(\infty) \end{bmatrix} \\
&= \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x(k) \\ v(k) \end{bmatrix} + \begin{bmatrix} 2 \\ -2 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k) - \begin{bmatrix} x(\infty) \\ v(\infty) \end{bmatrix} \\
&= \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x_e(k) + x(\infty) \\ v_e(k) + v(\infty) \end{bmatrix} + \begin{bmatrix} 2 \\ -2 \end{bmatrix} [u_e(k) + u(\infty)] + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k) - \begin{bmatrix} x(\infty) \\ v(\infty) \end{bmatrix} \\
&= \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x_e(k) \\ v_e(k) \end{bmatrix} + \begin{bmatrix} 2 \\ -2 \end{bmatrix} u_e(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k) + \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} x(\infty) + \begin{bmatrix} 2 \\ -2 \end{bmatrix} u(\infty) \\
&= \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x_e(k) \\ v_e(k) \end{bmatrix} + \begin{bmatrix} 2 \\ -2 \end{bmatrix} u_e(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k) + \begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} r + \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} r \\
&= \begin{bmatrix} 0.5 & 0 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} x_e(k) \\ v_e(k) \end{bmatrix} + \begin{bmatrix} 2 \\ -2 \end{bmatrix} u_e(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k) + \begin{bmatrix} 0 \\ -1 \end{bmatrix} r
\end{aligned}
$$

By holding $r(k)$ constant the reference disappears completely from the incremental system.

The control law for the incremental system is:

$$
\begin{aligned}
u(k) &= k_2 v(k) - k_1 x(k) \\
u_e(k) + u(\infty) &= k_2 v_e(k) + k_2 v(\infty) - k_1 x_e(k) - k_1 x(\infty) \\
u_e(k) &= k_2 v_e(k) + k_2 v(\infty) - k_1 x_e(k) - k_1 x(\infty) - u(\infty) \\
&= k_2 v_e(k) - k_1 x_e(k) + k_2 v(\infty) - k_1 x(\infty) - u(\infty) \\
&= k_2 v_e(k) - k_1 x_e(k) + k_2 \frac{u(\infty) + k_1 x(\infty)}{k_2} - k_1 x(\infty) - u(\infty) \\
&= k_2 v_e(k) - k_1 x_e(k) \\
&= -\mathbf{K} \begin{bmatrix} x_e(k) \\ v_e(k) \end{bmatrix} = -\begin{bmatrix} k_1 & -k_2 \end{bmatrix} \begin{bmatrix} x_e(k) \\ v_e(k) \end{bmatrix}
\end{aligned}
$$

Problem 4 Design a optimal steady state controller for the system above such that the performance index:

$$J = \frac{1}{2}\sum_{k=0}^{\infty}\left[\begin{bmatrix} x_e(k) \\ v_e(k) \end{bmatrix}^{\mathrm{T}} \mathbf{R}_1 \begin{bmatrix} x_e(k) \\ v_e(k) \end{bmatrix} + u_e^{\mathrm{T}}(k)\mathbf{R}_2 u_e(k)\right]$$

is minimized where:

$$\mathbf{R}_1 = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{R_2} = 1$$

Find the optimal feedback gain matrix for the servo system.

Solution:

We can find the controller that minimises the given performance index by solving the discrete time Riccati Equation:

$$\mathbf{P} = \mathbf{R}_1 + \mathbf{F}^T\mathbf{P}\mathbf{F} - \mathbf{F}^T\mathbf{P}\mathbf{G}[(\mathbf{R}_2 + \mathbf{G}^T\mathbf{P}\mathbf{G})^{-1}]\mathbf{G}^T\mathbf{P}\mathbf{F}$$

$$\mathbf{K} = (\mathbf{R}_2 + \mathbf{G}^T\mathbf{P}\mathbf{G})^{-1}\mathbf{G}^T\mathbf{P}\mathbf{F}$$

Note that this controller penalizes deviation between output and reference, and deviation form the steady state value of u.

The controller gain can be found through iteration, or with Matlab's `dlqr` or `dare` functions.

```
% Define System
F = [0.5 0 ; -0.5 1];
G = [2; -2];
Gr = [0; 1];
C = [1 0];
D = 0;

% Define Weights
R1 = [100 0 ; 0 1];
R2 = 1;

% Use the dlqr package to compute the controller gains K and the infinite
horizon ricatti solution P
[K,P,S] = dlqr(F, G, R1, R2)
```

```
K = 1×2
    0.2494   -0.0475
P = 2×2
  100.0624   -0.0119
   -0.0119   10.5168
S = 2×1
    0.9049
    0.0012
```

```
% Use the dare to compute the controller gains K and the infinite horizon
ricatti solution P
P = dare(F,G,R1,R2);
K = inv(R2 + G'*P*G)*G'*P*F;
```

```
K,P
```

```
K = 1×2
    0.2494   -0.0475
P = 2×2
  100.0624   -0.0119
   -0.0119   10.5168
```

```
% Use iteration to determine the controller gains K and the infinite horizon
ricatti solution P

% Iterate forward to a solution in the first case
% Start with the solution of the SS Riccati Equation with P = [0 0;0 0].
P = [0 0;0 0];
P = R1 + F'*P*F - F'*P*G*((R2 + G'*P*G)^-1)*G'*P*F; % Eq. (5.114) page 331

% Iterate for many steps and check whether or not P is constant.
for i = 1:1000
   P = R1 + F'*P*F - F'*P*G*((R2 + G'*P*G)^-1)*G'*P*F; % Eq. (5.114) page 331
end
disp('Solution of the Riccati equation:'); disp(P);
```

```
Solution of the Riccati equation:
  100.0624   -0.0119
   -0.0119   10.5168
```

```
K = inv(R2 + G'*P*G)*G'*P*F; % Eq. (5.116) page 331
disp('Optimal controller K:'); disp(K);
```

```
Optimal controller K:
    0.2494   -0.0475
```

In any case the values of $\mathbf{P}$ and $\mathbf{K}$ are:

$$\mathbf{P} = \begin{bmatrix} 100.0624 & -0.0119 \\ -0.0119 & 10.5168 \end{bmatrix}, \qquad \mathbf{K} = [0.2494 \quad -0.0475]$$

or in terms of k_1, k_2:

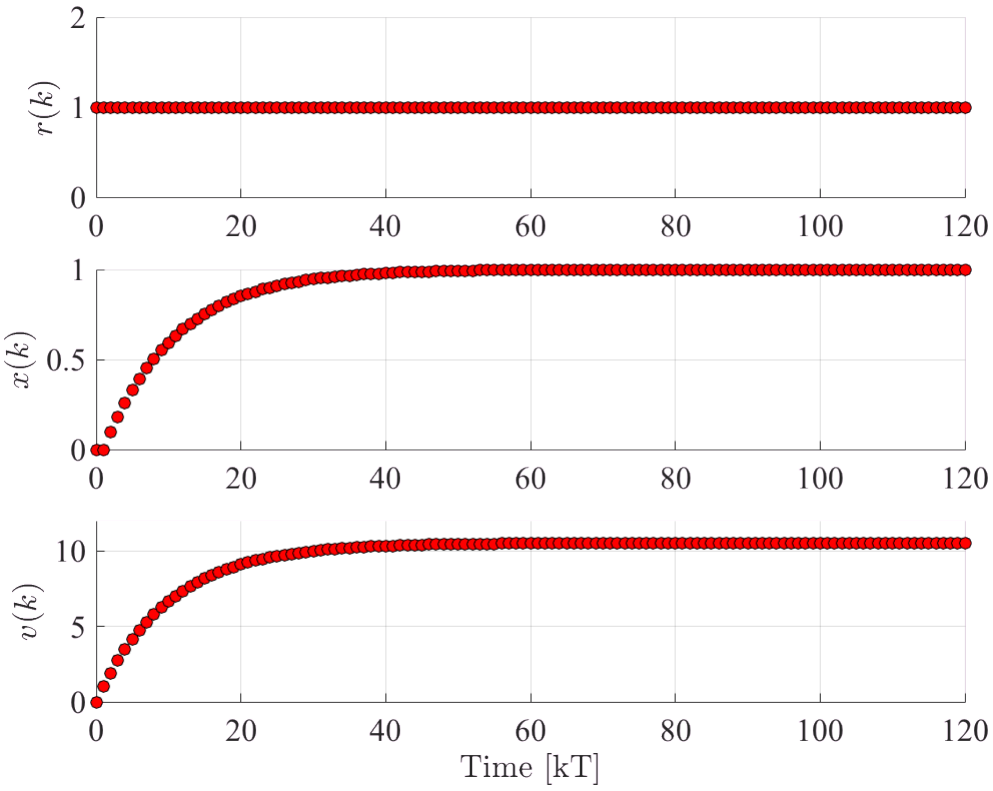
$$k_1 = 0.2494$$
$$k_2 = 0.0475$$

Problem 5 Simulate the servo control system response to a step change in the reference input. Plot the input to the system on one plot and in two different plots below, the output of the integrator $v(k)$ and the system output $y(k)$. Explain your results.

Solution:

In the MatLab code below, a simulation of the closed loop system with a constant reference is performed. Note how the steady state error is 0, i.e. after a while the system reaches the reference.

```
%% MODULE 14 - Problem 2 - An Integral Regulator for a Servo System
T = 0.1;

% Simulate the system in closed loop system with feedback gain k1, k2,
% starting condition of x(0) = v(0) = 0 and constant reference r(k) = r =
1.');
k = 0:120;
r = ones(1,length(k));
x = zeros(2,length(k));
u = ones(1,length(k));
for i = 1:length(k)-1
    u(i) = -K*x(:,i);
    x(:,i+1) = F*x(:,i) + G*u(i) + Gr*r(i);
end
y = x(1,:);
v = x(2,:);

% Plotting
figure, h1 = subplot(3,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
plot(k,r,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',6);
ylabel('$r(k)$','FontName','times','FontSize',16,'Interpreter','latex')

h2 = subplot(3,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(k,y,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',6);
ylabel('$x(k)$','FontName','times','FontSize',16,'Interpreter','latex')

h3 = subplot(3,1,3); set(h3,'FontName','times','FontSize',16)
hold on, grid on
plot(k,v,'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',6);
ylabel('$v(k)$','FontName','times','FontSize',16,'Interpreter','latex')
xlabel('Time [kT]','FontName','times','FontSize',16,'Interpreter','latex')
ylim([0 12]);
```



The steady state value $v(\infty) = \frac{u(\infty) + k_1 x(\infty)}{k_2} = 10.5168$ also matches with the simulation (if run for long enough).

```
disp(['v_ss = ' num2str(v(end))])
```

```
v_ss = 10.5167
```

Linear Control Design II - Group Work Problem Module 20

Roberto Galeazzi, Remus Mihail Prunesco & Leon Nagel
Department of Electrical Engineering
Technical University of Denmark
October, 2015
Thomas Paulsen
September, 2017

Description

Module 20 focuses on experiments with white noise. It is often necessary to calculate the average and root mean square (rms) values of a stochastic signal during simulations. Problem 6.5 in the textbook addresses the solution of this issue, which is based on the construction of two differential equations that can do this on line. These solutions are given below:

$$\frac{dv_{avg}}{dt} = \frac{1}{t}(-v_{avg} + u) \qquad (1)$$

where *v_avg* is the average value and *u* is the input. The rms value of a signal can be found from equation:

$$\frac{dv_{rms}}{dt} = \frac{1}{2t}(-v_{rms} + \frac{1}{v_{rms}}u^2) \qquad (2)$$

Problem 1

Given the formulas in Problem 6.5, show how Equations 1 and 2 can be obtained.

a. What is the mathematical relation between the rms value of a signal and the statistical parameters of the signal (mean and variance)?

b. Equations 1 and 2 are two scalar linear time-varying systems. Are they asymptotically stable? Do they represent a practical solution to the averaging problem?

c. What is their main failing?

Solution:

The derivation of the differential equations are done by differentiation of the two equation given in problem 6.5 on page 426 leading to

$$\frac{d}{dt}(V_{avg}) = \frac{d}{dt}\left(\frac{1}{t}\int_{t_0}^{t}(v(t)dt)\right) \Rightarrow \dot{V}_{avg} = -\frac{1}{t^2}\int_{t_0}^{t}(v(t)dt) + \frac{1}{t}(v(t) - v(t_0)) \Rightarrow \dot{V}_{avg} = -\frac{1}{t}V_{avg} + \frac{1}{t}v(t)$$

By realising that

$$-\frac{1}{t^2}\int_{t_0}^{t}(v(t)dt) = -\frac{1}{t}V_{avg}$$

and assuming $v(t_0)$=0. This can be done similarly for V_{rms}

$$v(t)^2dt\Big)^{\frac{1}{2}} \Rightarrow \dot{V}_{rms} = \frac{1}{2}\left(\frac{1}{t}\int_{t_0}^{t}v(t)^2dt\right)^{-\frac{1}{2}}\left[-\frac{1}{t^2}\int_{t_0}^{t}v(t)^2dt + \frac{1}{t}v(t)^2 - v(t_0)^2\right] \Rightarrow \dot{V}_{rms} = \frac{1}{2V_{rms}}\left[-\frac{1}{t}V_{rms}^2 + \frac{1}{t}v(t)^2\right] \Rightarrow \dot{V}_{rms} = -$$

this is done similarly by using the deffinition of V_{rms} and substitution for the integral expression along with the assumption $v(t_0)^2$=0.

1 a)

The deffinition of the rms value is given in the book on page 358 as:

$$X_{rms} = \sqrt{E\{X^2\}}$$

the deffinition of the variance *sigma_x*^2 is given on page 358 in the book as:

$$\sigma_x^2 = E\{X^2\} - E\{X\}^2 \Rightarrow E\{X^2\} = \sigma_x^2 + E\{X\}^2$$

by substitution of *E{X*^2*}* into the expression of the RMS and exchanging $E\{X\}^2 = m^2$ the relation is found as:

$$X_{rms} = \sqrt{\sigma_x^2 + m^2}$$

The rms value of a stochastic signal is actually the square root of its squared mean value (D.C. power) plus its variance (A.C. power).

1 b)

When analysing state stability the input can be set to zero resulting in:

$$\dot{V}_{avg} = -\frac{1}{t}V_{avg} \Rightarrow \frac{dV_{avg}}{dt} = -\frac{1}{t}V_{avg}$$

$$\dot{V}_{rms} = -\frac{1}{2t}V_{rms} \Rightarrow \frac{dV_{rms}}{dt} = -\frac{1}{2t}V_{rms}$$

this can be solved by seperation of the variables and integration as:

$$\int_{V_{avg}(0)}^{V_{avg}(t)}\left(\frac{1}{V_{avg}}dV_{avg}\right) = -\int_{t_0}^{t}\left(\frac{1}{t}dt\right)$$

$$\int_{V_{rms}(0)}^{V_{rms}(t)}\left(\frac{1}{V_{rms}}dV_{rms}\right) = -\frac{1}{2}\int_{t_0}^{t}\left(\frac{1}{t}dt\right)$$

this leads to

$$\ln(V_{avg}(t)) - \ln(V_{avg}(0)) = -(\ln(t) - \ln(t_0)) \Rightarrow \ln\left(\frac{V_{avg}(t)}{V_{avg}(0)}\right) = \ln\left(\frac{t_0}{t}\right) \Rightarrow V_{avg}(t) = V_{avg}(0)\frac{t_0}{t}$$

$$\ln(V_{rms}(t)) - \ln(V_{rms}(0)) = -\frac{1}{2}(\ln(t) - \ln(t_0)) \Rightarrow \ln\left(\frac{V_{rms}(t)}{V_{rms}(0)}\right) = \frac{1}{2}\ln\left(\frac{t_0}{t}\right) \Rightarrow V_{rms}(t) = V_{rms}(0)\frac{1}{2}\frac{t_0}{t}$$

From which is can be seen that for $\lim_{t \to \inf} V_{avg}(t) \to 0$ and $\lim_{t \to \inf} V_{rms}(t) \to 0$ thus both differential equations are asymptotically stable.

1 c)

The infinite time required for the averaging and the cutoff of the averaging effect at large times is their main failing.

Problem 2

Use the Equations 1 and 2 to calculate the mean and variance of the signals from the Uniform, *U*(0.5,1/2), and Gaussian distributed (Random Number) noise generators of the Matlab/Simulink package. The notation *U*(0.5,1/2) means uniformly distributed with a mean value 0.5 and variance 1/2 (σ); whereas *N*(0,1) means normally distributed with a mean value of 0 and a variance of 1 (σ). Use a generator sample time of 1 s, simulate over 100 s and plot the results. Then use a sample time of 0.1 s and simulate over 10 s. The variable *t* can be obtained by using the clock in the Simulink *Source* menu. Simulate over a time interval which is sufficient to obtain a reasonably smooth response and try different noise generator seeds and sample times. Use the standard integration routine rk45 (Runga Kutta).

a. Do simulation results agree with what is expected? Are the obtained mean and standard deviations correct? Explain.

Solution:

The averaging differential equations are easily realized in Simulink if it is remembered that the terms which must be realized in front of the integrator are those that occur on the right hand side of the differential equations inside the internal feedback loop.

2 a)

The results of averaging the Random Number (Gaussian noise), or the Uniform (Distributed) Random Number, do change with the value of the seed selected. This is because there is some statistical variation in the output of the generators. This variation can be eliminated if the generators outputs for many different seeds are averaged. It is only the mean values (calculated over long time intervals) of the generator outputs which are well defined.

```
tc = 100;
vstd = 1;
vmean =0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% for uniform distribution sigma^2 = 1/12*(b-a)
%                           mu = 1/2*(a+b)
%                           a = mu - sigma*sqrt(3)
%                           b = mu + sigma*sqrt(3)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nmax = vmean +vstd*sqrt(3);
nmin = vmean -vstd*sqrt(3);

varexp = vstd^2;

for i = 1:1 % change the range of the counter i in ordet to try different
generator seeds
    seed = i*11;
    [t1,x1,y1] = sim('m13solm',[0.01 1000]);
    xs(i,:,:) = x1(:,:);
    ys(i,:,:) = y1(:,:);
end

xm = mean(xs,1);
ym = mean(ys,1);

varmeas = var(ys(1,:,3))
```

```
varmeas = 0.9817
```

```
mmeas = mean(ys(1,:,3))
```

```
mmeas = 7.3367e-04
```

```
mm = mean(ys(1,end/2:end,1))
```

```
mm = 0.0245
```

```
mrms = mean(ys(1,end/2:end,2))
```

```
mrms = 1.0007
```

```
vrmsexp = sqrt(vmean^2 + vstd^2)
```

```
vrmsexp = 1
```

```
vrmsmeas = sqrt(mmeas^2 + varmeas)
```

```
vrmsmeas = 0.9908
```

```
varmm = mrms^2-mm^2
```

```
varmm = 1.0008
```

Problem 3

Instead of t in the equations above, use a time constant τ=100 s. Simulate the differential equations using this simpliffication.

a. Do the results of this simulation agree with those found in Problem 2? Explain.

b. What is the practical application of your results?

Solution:

If one uses a time constant, , instead of the time in the differential equations then one is using a real low pass filter instead of the ideal one implied by the definitions from which the equations are derived. This has no effect on the results obtained for the average and rms values measured as long as the time constant is much longer than the sampling time of the noise generators. The time constant should be on the order of 20 to 50 times the sampling time.

3 a)

Thus the results of the simulation with a time constant do agree with those obtained with the ideal filters of the primary differential equations.

3 b)

The practical application of the results obtained is that it is possible to build true rms instruments which work in actual applications. In fact all real rms instruments work with some averaging time.

```
[tsc,xsc,ysc] = sim('m13solm3',[0.01 1000]);
varmeasc = var(ysc(:,3))
```

```
varmeasc = 0.9902
```

```
mmeasc = mean(ysc(:,3))
```

```
mmeasc = -0.0017
```

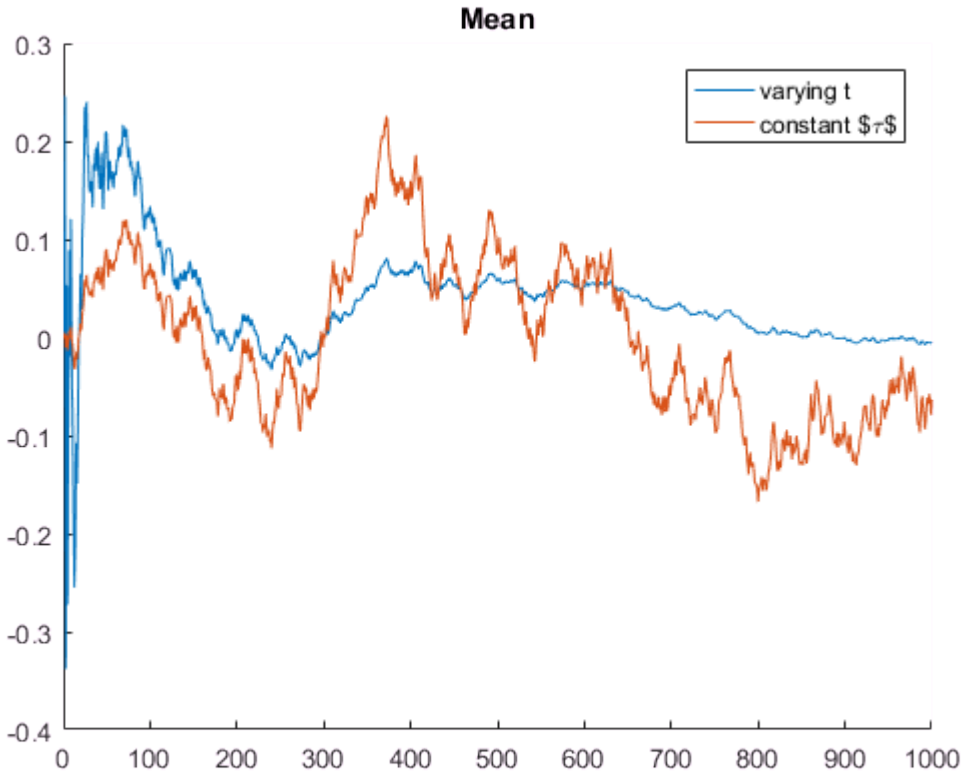
```
mmc = mean(ysc(end/2:end,1))
```

```
mmc = -0.0351
```

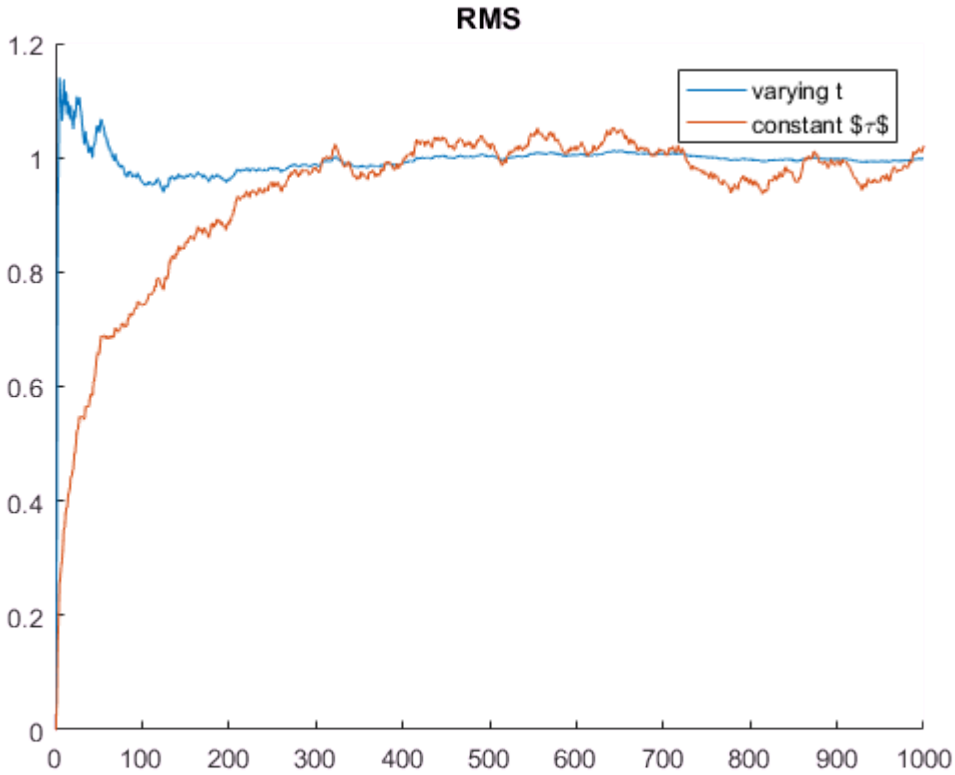
```
mrmsc = mean(ysc(end/2:end,2))
```

```
mrmsc = 0.9950
```

```


figure
hold all
plot(t1,ym(1,:,1));
plot(tsc,ysc(:,1))
legend('varying t','constant $\tau$');
title('Mean');
```




```
figure
hold all
plot(t1,ym(1,:,2));
plot(tsc,ysc(:,2))
legend('varying t','constant $\tau$');
title('RMS');
```



Problem 4

Add two different noise signals together (different seeds and variances) and use Equations 1 and 2 to find their means and standard deviations.

a. What do the results show about the rule for adding noise signals?

Solution:

When adding to different noise signals together the resulting noise signal will have mean and variance equal to the sum of the means and variances. This is described on page 368 in the book.

```
seed2 = 345;
vstd2 = 0.5;
vmean2 =0.5;
nmax2 = vmean2 +vstd2*sqrt(3);
nmin2 = vmean2 -vstd2*sqrt(3);
[tss,xss,yss] = sim('m13solm4',[0.01 5000]);

mmeass = mean(yss)
```

```
mmeass =
    0.4977    1.2255    0.5089    0.0044    0.5045
```

```
mms = mean(yss(end/2:end,1))
```

```
mms = 0.5064
```

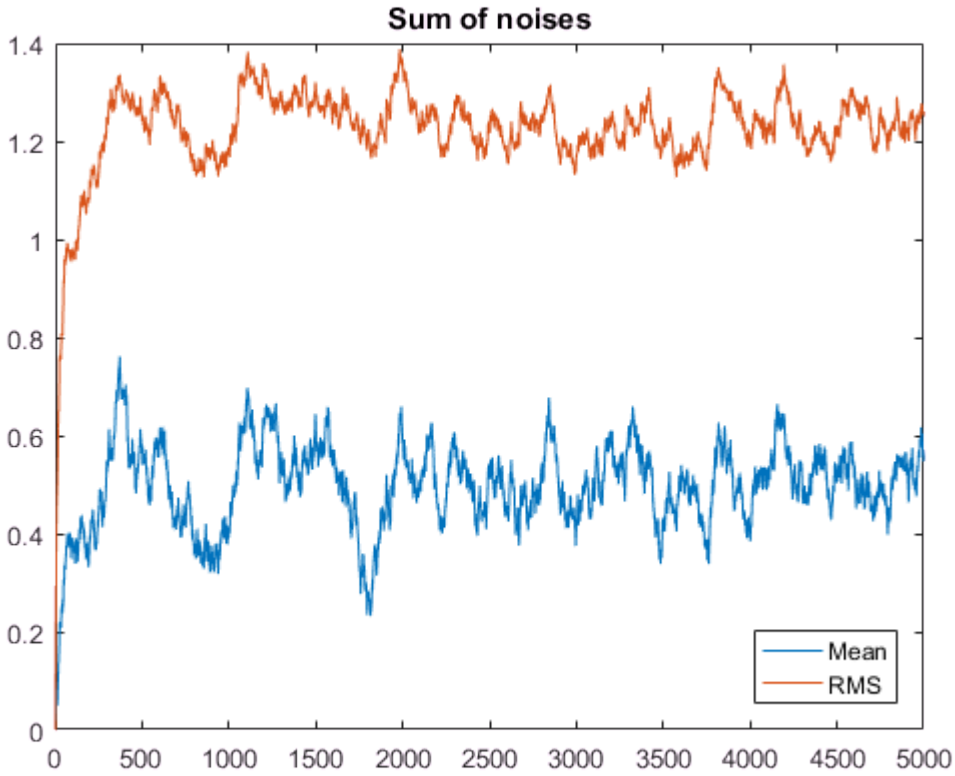
```
mrmss = mean(yss(end/2:end,2))
```

```
mrmss = 1.2277
```

```
exprmss = sqrt(vstd2^2+vstd^2+(vmean2+vmean)^2)
```

```
exprmss = 1.2247
```

```
figure
plot(tss,yss(:,1:2))
title('Sum of noises');
legend('Mean','RMS','location','southeast');
```



Problem 5

Assume that the available noise generators can deliver an approximation to white noise.

a. Construct a Wiener Process and use two such processes to construct a random walk process in two dimensions. Demonstrate graphically that this two dimensional process works as expected. Compare your results to those in Figures 6.11-6.13 in the textbook.

Solution:

5 a)

Result of the simulation is shown in Figure 1, the implementation can be found in the matlab simulink files.

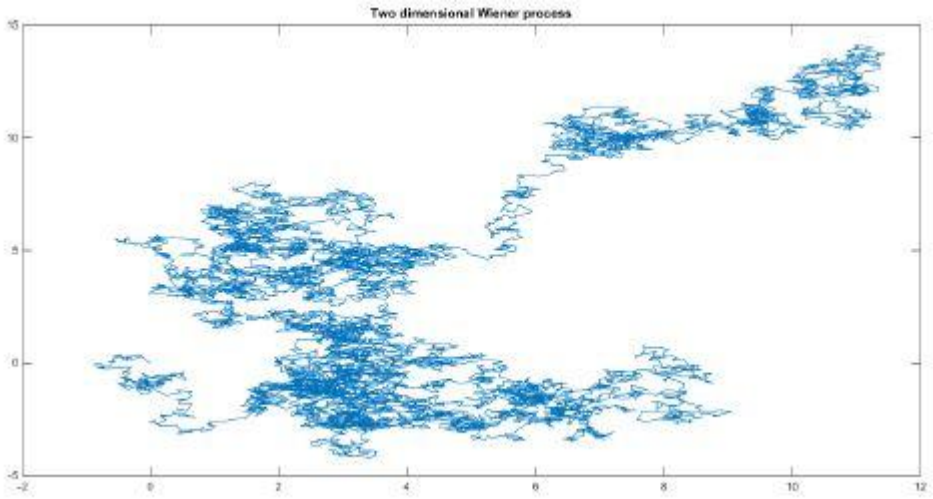
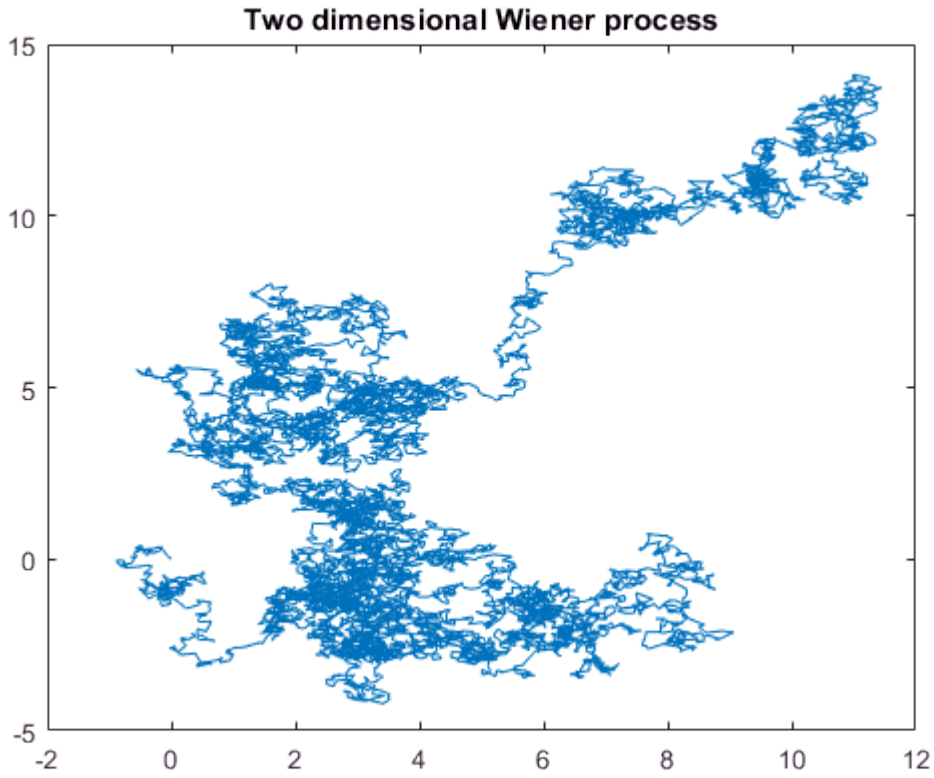


Figure 1 : Simulation of a two dimensional Wiener process .

```
seed61 = 123;
seed62 = 456;

[t6,x6,y6] = sim('m13sol6',[0.01 1000]);
figure
plot(x6(:,1),x6(:,2));
title('Two dimensional Wiener process');
```



Further Experiments with White Noise

In this exercise the work started in Group Work Exercise 13 will be continued with the study of a second order low pass filter and the noise calculated using the Lyapunov equation. Consider a second order low pass filter (ω_0=10 rad/s and ζ=0.707)

$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -\omega_0^2 x - 2\zeta\omega_0 v + \omega_0^2 \qquad (3) \\ y &= x \end{aligned}$$

driven by band-limited noise *u* with autocorrelation function

$$\mathbf{R}(\tau) = \sigma_u^2 \mathrm{e}^{-\beta|\tau|}$$

where σ_u^2=0.5 and β=10.

Problem 6

Using the Lyapunov equation find theoretical value of the covariance matrix **Q** as a function of the system parameters. Then solve numerically the Lyapunov equation using the Matlab command *lyap()*.

Solution:

The second order low pass filter is a time invariant stable system, hence the propagation of white noise through this system can be studied by setting up the steady state Lyapunov equation

$$\mathbf{A}\mathbf{Q}_{\inf} + \mathbf{Q}_{\inf}\mathbf{A}^{\mathrm{T}} + \mathbf{B}_v \mathbf{V} \mathbf{B}_v^{\mathrm{T}} = 0 \qquad (18)$$

where $\mathbf{Q}_{inf}$ is the steady state covariance matrix, which is symmetric and positive definite

$$\left(\mathbf{Q}_{\inf} = \mathbf{Q}_{\inf}^{\mathrm{T}} > 0\right)$$

The lyapunove equation is based on the assumption of white noise, however the input signal u is not white noise, but it is band-limited noise with a given autocorrelation function *R(*τ); therefore Equation 18 cannot be directly applied to the given system. However the band-limited noise *u* with autocorrelation *R*(τ) can be seen as the result of passing white noise ω with noise power equal to 1 through the filter

$$S(\omega) = \frac{2\sigma^2\beta}{\beta^2 + \omega^2}$$

this can also be written as

$$S(\omega) = G(\omega)\widetilde{S}$$

where $\widetilde{S} \in N_{iid}(0,1)$ and $G(\omega) = \dfrac{2\sigma^2\beta}{\beta^2 + \omega^2}$ if the variable ω^2 is rewritten as $(j\omega)^2$ then $G(\omega)$ can be expressed as

$$G(\omega) = \frac{\sqrt{2\beta}\,\sigma}{\beta + j\omega}\frac{\sqrt{2\beta}\,\sigma}{\beta - j\omega}$$

This expressed the fourier transform as

$$S(j\omega) = \widetilde{G}(j\omega)\widetilde{S}\widetilde{G}(-j\omega)$$

substituting $s = j\omega$ and considering only the stable filter results in the band-limited noise *u* being created as a white noise process sent through the filter:

$$H(s) = \frac{\sqrt{2\beta}\,\sigma_u}{s + \beta}$$

Hence the evaluation of the noise propagation through the second order low pass filter can still be done with the Lyapunov equation if first we augment the system as follows (augmented state vector $\mathbf{x}_a = [u, x, v]^{\mathrm{T}}$)

$$\dot{u} = -\beta u + \sqrt{2\beta}\,\sigma_u w$$
$$\dot{x} = v$$
$$\dot{v} = -\omega_0^2 x - 2\zeta\omega_0 v + \omega_0^2 u$$

The steady state Lyapunov equation in matrix form then reads ($v = 1$)

$$\begin{bmatrix} -\beta & 0 & 0 \\ 0 & 0 & 1 \\ \omega_0^2 & -\omega_0^2 & -2\zeta\omega_0 \end{bmatrix}\begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} + \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix}\begin{bmatrix} -\beta & 0 & \omega_0^2 \\ 0 & 0 & -\omega_0^2 \\ 0 & 1 & -2\zeta\omega_0 \end{bmatrix} + \begin{bmatrix} \sqrt{2\beta}\,\sigma_u \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} \sqrt{2\beta}\,\sigma_u & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

which gives rise to the following six algebraic equations in the unknowns q_{ij}

$$-2\beta q_{11} + 2\beta\sigma_u^2 = 0$$
$$q_{13} - \beta q_{12} = 0$$
$$-\beta q_{13} - 2\zeta\omega_0\beta q_{13} + \omega_0^2 q_{11} - \omega_0^2 q_{12} = 0$$
$$2q_{23} = 0$$
$$q_{33} - 2\zeta\omega_0 q_{23} + \omega_0^2 q_{12} - \omega_0^2 q_{22} = 0$$
$$-4\zeta\omega_0 q_{33} + 2\omega_0^2 q_{13} - 2\omega_0^2 q_{23} = 0$$

The solution can be found to be

$$q_{11} = \sigma_u^2$$

$$q_{12} = \frac{\omega_0^2}{\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2}\sigma_u^2$$

$$q_{13} = \frac{\beta\omega_0^2}{\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2}\sigma_u^2$$

$$q_{22} = \sigma_x^2 = \frac{\beta\omega_0 + 2\zeta\omega_0^2}{2\zeta(\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2)}\sigma_u^2$$

$$q_{23} = 0$$

$$q_{33} = \sigma_v^2 = \frac{\beta\omega_0^3}{2\zeta(\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2)}\sigma_u^2$$

and the covariance matrix $\mathbf{Q}_{\text{inf}}$ reads

$$= \begin{bmatrix} \sigma_u^2 & \frac{\omega_0^2}{\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2}\sigma_u^2 & \frac{\beta\omega_0^2}{\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2}\sigma_u^2 \\ \frac{\omega_0^2}{\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2}\sigma_u^2 & \sigma_x^2 = \frac{\beta\omega_0 + 2\zeta\omega_0^2}{2\zeta(\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2)}\sigma_u^2 & 0 \\ \frac{\beta\omega_0^2}{\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2}\sigma_u^2 & 0 & \sigma_v^2 = \frac{\beta\omega_0^3}{2\zeta(\beta^2 + 2\beta\zeta\omega_0 + \omega_0^2)}\sigma_u \end{bmatrix} = \begin{bmatrix} 0.5 & 0.14646 & 1.4646 \\ 0.14646 & 0.25003 & 0 \\ 1.4646 & 0 & 10.358 \end{bmatrix}$$

The diagonal elements of the covariance matrix $\mathbf{Q}_{\text{inf}}$ are the variances of the individual state variables indicating how large the deviations will be with respect to the mean values of the states. The off-diagonal terms provide instead an **absolute measure of linear correlation among the state variables**. In particular it can be noted that

the state variables *x* and *v* are correlated with the noise input *u*, and to positive variations of *u* correspond positive variations of *x* and *v* (this can be seen from $q_{12} > 0$ and $q_{13} > 0$). The correlation between *u* and *v*, and between *u* and *x* is certainly expected since the state variables are affected by the noise source

the state variables *x* and *v* are not correlated because they are orthogonal; in fact the integral/differential operator always introduces a phase shift of 90 degrees.

The same result shown in Equation 40 is obtained by using the Matlab command *lyap()* as illustrated in the following script

```
% System description
w = 10;
zeta = 0.707;
sigma_u = sqrt(0.5);
beta = 10;
V = 1;
```

```
A = [0 1;-w^2 -2*zeta*w];
B = [0 w^2]';
C = [1 0];

Aa = [-beta 0 0;0 0 1;w^2 -w^2 -2*zeta*w];
Ba = [sqrt(2*beta)*sigma_u 0 0]';
Q = Ba*V*Ba';

% Solution of the Lyapunov equation for calculating the noise amplitudes
lyap(Aa,Q)
```

```
ans =
    0.5000    0.1465    1.4646
    0.1465    0.2500    0.0000
    1.4646    0.0000   10.3576
```

Problem 7

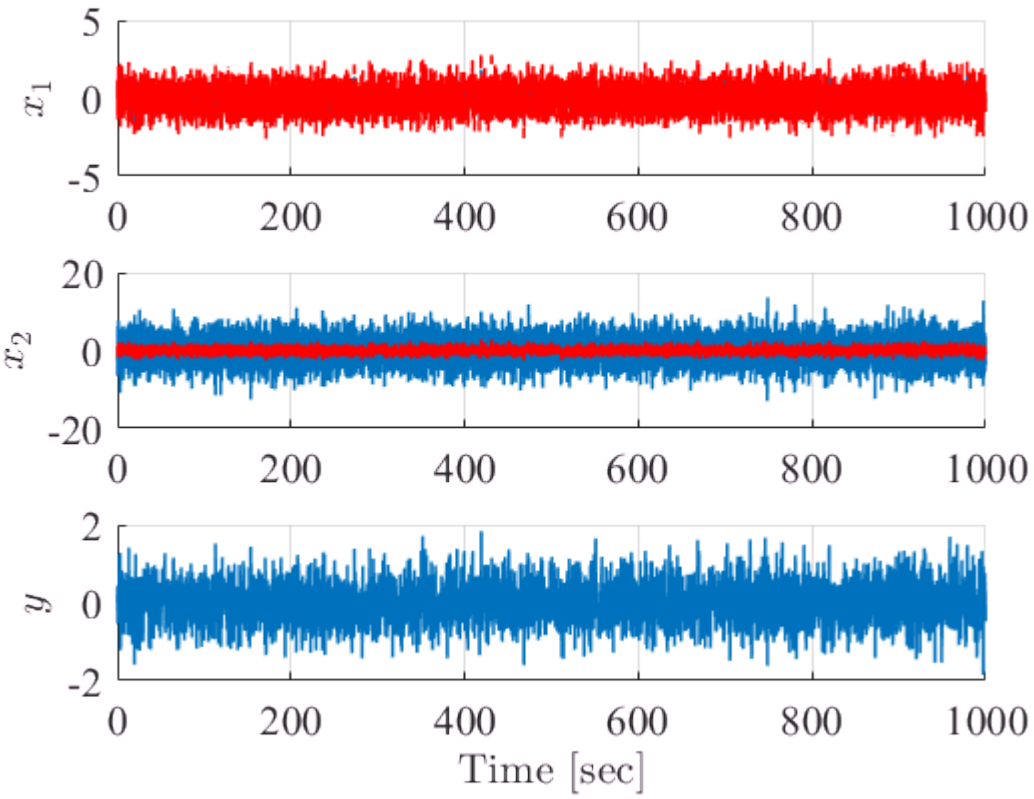
Implement a Simulink model representing Equation 3 and simulate it for at least 1000 seconds. Export the input, the states and the output, and calculate the statistical properties of these signals. Compare the obtained results with those of Problem 1 and comment on them.

Solution:

The calculation of the statistical properties of the state variables based on simulated/measured data is strongly affected by the length (number of samples) of the available data set, as shown in the following three test cases:

```
% Simulating the system
sim('gp14_simulink',1000);

u = logsout.getElement(1).Values.Data;
x = logsout.getElement(3).Values.Data(:,1);
v = logsout.getElement(3).Values.Data(:,2);
y = logsout.getElement(2).Values.Data;
mX = logsout.getElement(4).Values.Data;
mV = logsout.getElement(5).Values.Data;
rmsX = logsout.getElement(6).Values.Data;
rmsV = logsout.getElement(7).Values.Data;
t = logsout.getElement(1).Values.Time;

figure, h1 = subplot(3,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
plot(t,x,t,u,'--r','LineWidth',1); % state 1 output
ylabel('$$x_1$$','FontName','times','FontSize',16,'Interpreter','latex')
h2 = subplot(3,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(t,v,t,u,'--r','LineWidth',1); % state 2 output
ylabel('$$x_2$$','FontName','times','FontSize',16,'Interpreter','latex')
h3 = subplot(3,1,3); set(h3,'FontName','times','FontSize',16)
hold on, grid on
plot(t,y,'LineWidth',1); % original noise signal
```

```
ylabel('$$y$$','FontName','times','FontSize',16,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
```




```
% Numerical evaluation of the solution of the Lyapunov equation
Qmeas = cov([u x v])
```

```
Qmeas =
    0.4810    0.1356    1.4312
    0.1356    0.2359    0.0009
    1.4312    0.0009   10.0512
```

However for an ergodic process in the limit of time going to infinity the time average will match the ensemble average (Section 6.3.4 of the textbook), hence the longer we simulate/acquire data the closer the measured covariance matrix is to the theoretical covariance matrix $\mathbf{Q}_{\text{inf}}$.

Linear Control Design II - Group Work Problem Module 22 solution

Description

Problems

The purpose of this exercise is mainly to investigate the influence of noise on a physical dynamic system. It is also intended to show how state feedback regulators must be configured to work in real dynamic nonlinear systems. The parameters selected for the hot air balloon are close to those actually in use. The student should look at the notes at the end of the exercise in order to obtain extra hints to aid in solving the problems.

This group problem concerns a height regulator for a hot air balloon. Thus only the vertical degree of freedom of the hot air balloon's movement is of interest here. The state variables for the balloon are the altitude, $h\ [m]$, the vertical velocity, $w\ [ms^{-1}]$ and the temperature of the heated air in the balloon envelope, $T\ [K]$. To cause the balloon to rise, it is provided with a gas burner (heat source) which can provide a variable heat quantity, $q\ [kJ]$, which is the only input to the system. The state equations which describe the vertical movement of the hot air balloon are:

$$\frac{dT}{dt} = -\frac{1}{\tau_1}(T - T_a) + cq \qquad (1)$$

$$M\frac{dw}{dt} = Vg(\rho_a - \rho) - Mg - bw \qquad (2)$$

$$\frac{dh}{dt} = w \qquad (3)$$

where $\tau_1 = 300\,s$, $T_a(h)\ [K]$ is the ambient temperature of the air surrounding the balloon, $c = 1.10^{-3}[KkJ^{-1}s^{-1}]$ is a constant, $\rho_a(h)\ [kgm^{-3}]$ is the ambient air density, $\rho\ [kgm^{-3}]$ is the density of the air in the balloon envelope, $M = 300\,kg$ is the total mass of the balloon and its load, $V = 2000\,m^3$ is the volume of the envelope, $g = 9.81\,ms^{-2}$ is the acceleration of gravity and $\frac{b}{M} = \frac{1}{50} = \frac{1}{\tau_2}$ is the air resistance coefficient divided by the balloon mass, implying that the air resistance coefficient is $b = 6s^{-1}$.

A hot air balloon rises because heating the air inside the envelope causes the density of the air to decrease. This decreases the effective weight of the balloon and its load. At the same time as the

balloon rises, it moves into air which decreases in temperature and density while at the same time the heat losses increase. Equations(1) and (2) above take these effects into account. What is missing is a description of the relation of the density/temperature product of the envelope air to that of the surrounding air (Boyle's Law) and how the density and temperature vary with altitude. These relationships for this problem are given by:

$$\rho T = \rho_a T_a \qquad (4)$$

$$\rho_a(h) = \rho_a(0) - ph = \rho_a(0) - 0.795 \cdot 10^{-4}h \qquad (5)$$

$$T_a(h) = T_a(0) - fh = T_a(0) - 6 \cdot 10^{-3}h \qquad (6)$$

where $\rho_a(0) = 1.22[kgm^{-3}]$, $T_a(0) = 290\,K$, $p = 0.795 \times 10^{-4} kgm^{-4}$ and $f = 6 \times 10^{-3} Km^{-1}$.

The desired operating altitude of the balloon is $h_s = 500\,m$ at an envelope temperature of $T_s\ [K]$.

Problem 1 Given the required height, determine the ambient temperature and density at which the balloon will operate in steady state. On this basis the heat input to the burner can be determined. These variables define the nominal operating point (equilibrium linearization point) of the system, q_0, h_s and T_s.

Solution:

The first problem is to construct a Simulink model for the hot air ballon control object. This is best done by reformulating the system equations slightly so as to use immediately the information given in the exercise text equations (4) to (6) for the ambient density and temperature.

From equation (4) it is clear that

$$\rho = \rho_a \frac{T_a}{T}$$

This makes it possible to write down the state equations as

$$\frac{dT}{dt} = -\frac{1}{\tau_1}(T - T_a) + cq$$

$$\frac{dw}{dt} = \frac{gV}{M}(\rho_a - \rho) - g - \frac{b}{M}w$$

$$= \frac{gV}{M}\rho_a(1 - \frac{T_a}{T}) - g - \frac{1}{\tau_2}w$$

$$\frac{dh}{dt} = w$$

The second form of equation $\frac{dw}{dt}$ is convenient as both ρ_a and T_a are functions of and can be driven directly from the altitude state variable in the block diagram. These functions can be conveniently written into the “User Defined Function Block” in Simulink and driven by h.

```
clc; close all;clear all
tau1 = 300; %sec
c = 1e-3;   %K/kJ sec
M = 300;    %kg
V = 2000;   %m3
g = 9.81;   %m/s2
b = 6;      %sec^-1

rhoa0 = 1.22; %kg/m3
Ta0 = 290;    %K
p = 0.795e-4;
f = 6e-3;
```

```
hs = 500;     %m
ws = 0;
offset = 0.001;

%% Part 1
% We start out by inserting the values in equation (5) and (6)
rhoah = rhoa0 - p*hs;
Tah = Ta0 - f*hs;

% Because we are working in steady state the time derivatives in eq. (1),
% (2) and (3) are equal to zero. Inserting this in eq. (2), along with rho
% from (4) we get:
% 0 = V*g*(rhoah - rhoa0*Ta0/T) - M*g => (isolate T)
Ts = Tah/(1-M/(rhoah*V));

% Finding q from eq. (1):
% 0 = -1/tau1*(T-Tah) + c*q => (isolate q)
qs = (Ts-Tah)/(c*tau1);

ws = 0;
```

Otherwise the state equations can be realized in a straight forward way in Simulink. However, if one does not proceed in a logical fashion, this can be difficult to do without making many

errors. The following guide lines are given here to aid in constructing and debugging simulation models:

- Place the blocks in the diagram in a logical fashion so that they can easily be recognized and connected. In particular in physical models the subsystem models should be placed so that it is clear what subsystem drives another.
- Remember that the simulation object represents a real physical object and thus must be provided with inputs and initial conditions which are physically reasonable: Simulink cannot solve equations which are not mathematically consistant.
- In assembling sets of equations for subsystems, it is quite possible to simulate and test them one at a time before putting them into a larger system.
- Before starting a long simulation study of a block diagram, be careful to test it before final results are made about the system being simulated.
- When a simulation block diagram gives incorrect results, it is nearly always a simple error in its construction. Thus when models do not work correctly, study the block diagram and its constants and initial conditions carefully to make certain that they are correct. If necessary, study the pieces of the diagram seperately using simulation or other methods, for example, steady state operating point, eigenvalues, short term operation, etc.

In connection with the Simulink exercise for the hot air ballon some difficulty was experienced with failure to provide reasonable inputs and starting conditions for the ballon: suggestions about these variables were provided at the end of the exercise text. Study the exercise text before starting model building.

Using linearization, it can be shown that:

$$\mathbf{A} = \begin{bmatrix} -\frac{1}{\tau_1} & 0 & -\frac{f}{\tau_1} \\ g\rho_a \frac{V}{M}\left(\frac{T_a(0) - f h_s}{T_s^2}\right) & -\frac{b}{M} & g\frac{V}{M}\left[\frac{\rho_a f}{T_s} - p\left(1 - \frac{T_a}{T_s}\right)\right] \\ 0 & 1 & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} -\frac{1}{\tau_1} & 0 & -\frac{f}{\tau_1} \\ \frac{g}{c q_0 \tau_1}\left(1 - \frac{M}{\rho_a V}\right) & -\frac{1}{\tau_2} & g\frac{V}{M}\left[\frac{\rho_a f}{T_s} - p\left(1 - \frac{T_a}{T_s}\right)\right] \\ 0 & 1 & 0 \end{bmatrix} \qquad (7)$$

$$\mathbf{B} = \begin{bmatrix} c \\ 0 \\ 0 \end{bmatrix} \qquad (8)$$

for the state vector $[\Delta T \quad \Delta w \quad \Delta h]^{\mathrm{T}}$ and the input variable Δq.

Problem 2 As a homework exercise (not during the group work period) show that the linearized matrices in equations (7) and (8) are correct and draw a block diagram for the linearized system.

Solution:

Linearization of the ballon system can be performed using hand calculations and this was done to find the equations (7) and (8) in the exercise text. This is often a good way of understanding a system and of finding errors.

Problem 3 Construct a Simulink model for the nonlinear system of equations (1) to (6). Use the Matlab routine `trim` to find the steady state equilibrium operating point of the nonlinear system: that is to say find q_0, h_s and T_s. These variables can also be found analytically, try this at home.

Use the Simulink model to make sure that the equilibrium linearization point found is correct. Simulate over 10 and $100 \ \ s$. What do these simulations suggest about the stability of the hot air balloon?

The model is to be linearized around the operating point found. This can be done using the Matlab function `linmod`. The result of this linearization should be checked with the linearized equations (7) and (8) above.

Solution:

To find the equilibrium operating conditions for the ballon, it was suggested that one use the Matlab function ‘trim’. Extra written documentation for this function can be found on the world web net using the search words “Simulink + trim”. The operating point for the execise can be found using an operating point close to the actual operating point. The difficulty is that the ballon model is nonlinear and unstable and very sensitive to the initial conditions used for the ‘trim’ routine (which is a multidimensional search routine). This means that the routine can find many local minima if it has very inaccurate initial guesses.

In using the trim function, the Matlab documentation should carefully read before trying to use the function. It is necessary to specify in the arguments of the function what constraint has to be placed on the optimization.

In the case of the ballon, the constraint on has to be set at $500\,m$ and the other operating point specifications set close to their true values. There are at least two methods of using 'trim' that work in a straight foward fashion.

If a constant and correct heat input is used in the model, then the trim statement,

```
x = [0 0 0];
qin = 0;
qswitch = 0;
[x,u,y,dx] = trim('Q3',[300 0 500]')
```

```
x =
  328.7855
   -0.0000
  499.8535
u = 139.2821
y = 328.7855
dx =
   1.0e-12
    0.0026
    0.6761
   -0.0000
```

If it is desired to find a more accurate value of the heat input then the command,

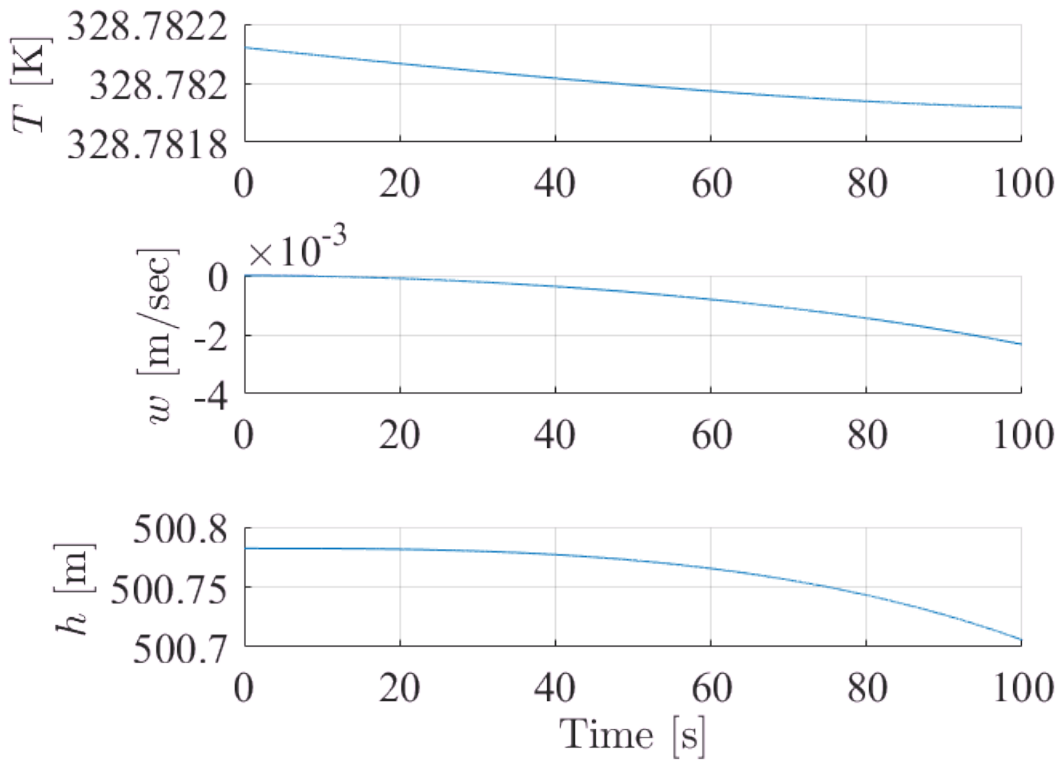
```
[x,u,y,dx] = trim('Q3',[328 0 500]',[139],[328 0 500]') %[deltaT deltaw
deltah]
```

```
x =
  328.7821
   -0.0000
  500.7821
u = 139.2894
y = 328.7821
dx =
   1.0e-14
    0.0139
    0.9095
   -0.0000
```

It should be noted that while the operating point is more or less correctly given by the numbers above, the control object is unstable and this means that if the ballon is simulated over a very long time, 100 - 500 sec, this instability will show itself. The trajectory of the control object will diverge from $h = 500\,m$ either positive or negative due to small numerical errors.

```
qswitch = -1;
qin = qs; %Simulate at linearization point
sim('Q3',100);

figure, h1 = subplot(3,1,1); set(h1,'FontName','times','FontSize',16);
hold on, grid on;
plot(T.time,T.signals.values);
ylabel('$T$ [K]','FontName','times','FontSize',16,'Interpreter','latex');
```

```
h2 = subplot(3,1,2); set(h2,'FontName','times','FontSize',16);
hold on, grid on;
plot(w.time,w.signals.values);
ylabel('$w$ [m/sec]','FontName','times','FontSize',16,'Interpreter','latex');

h3 = subplot(3,1,3); set(h3,'FontName','times','FontSize',16);
hold on, grid on;
plot(h.time,h.signals.values);
ylabel('$h$ [m]','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('Time [s]','FontName','times','FontSize',16,'Interpreter','latex');
```




```
% If we simulate with qin = qs then we stay at the same height, because the
% temperature is held.
% If we simulate for qin < qs, then we go towards the ground. The
% simulation only holds as long as we are above ground. If we simulate for
% qin > qs, then we rise.


% Linearize the model:
```

```
x =
  328.7834
    0.0000
  500.4254
u = 0
y = 328.7834
dx =
```

```
   1.0e-13
    0.0003
    0.6670
    0.0000
```

```
qswitch = 0;
[A,B,C,D] = linmod('Q3',x',qs)
```

```
A =
   -0.0033         0   -0.0000
    0.2049   -0.0200    0.0007
         0    1.0000         0
B =
   1.0e-03
    1.0000
         0
         0
C =
     1     0     0
D = 0
```

Problem 4 Calculate the eigenfrequencies of the linearized system at the desired operating point. Is the system stable? Why do balloon skippers constantly turn on and off the gas burner? Is automatic control desirable?

Solution:

The Matlab function “linmod” was to be used to linearize the system at its desired operating point. Extra written documentation for this function was provided during the exercise period. Linmod requires that the correct operating conditions be set for the model. The results of carrying out the linearization should again agree nearly exactly with those given in the exercise text. When using linmod, it should be remembered that the system has to be linearized around the selected operating point.

The eigenfrequencies of the hot air ballon show that it is an unstable control object and can only be held at a constant altitude using continuous watchfulness on the part of the ballon skipper.

Automatic control is thus desireable. The eigenfrequencies found for the ballon states at $500\,m$ altitude are

```
evals = eig(A);
T=evals(1) %%rad/ses
```

```
T = -0.0409
```

```
w=evals(2) %% rad/sec
```

```
w = 0.0026
```

```
h=evals(3) %% rad/sec
```

```
h = 0.0150
```

These eigenfrequencies indicate that the baloon is unstable, though very slow. That the system is slow means that it can be controlled manually by the ballon skipper without too much difficulty. However, the control of the temperature in the envelope is extremely sensitive but also extremely slow so constant attention to the envelope temperature is necessary.

Problem 5 In the first case it is desirable that a very mild control be used on the simulation model in order to keep the balloon around its desired operating point. Design an LQR feedback loop using the linearized model and the weight matrices:

$$\mathbf{R_1} = \begin{bmatrix} 1 \cdot 10^{-7} & 0 & 0 \\ 0 & 1 \cdot 10^{-7} & 0 \\ 0 & 0 & 1 \cdot 10^{-7} \end{bmatrix}, \mathbf{R_2} = [1]$$

What is unusual about the selection of these weight matrices? What does this imply about the nature of the control system selected?

Solution:

The unusual characteristic of the weighting matrices in the exercise is that they provide a very, very mild control: the matrix element of $\mathbf{R}_1$ are very small while at the same time those of $\mathbf{R}_2$ are very large. This is required here as it is desired that the effects of noise be evaluated for the system in a nearly open loop condition. The LQR gains found using the gain matrix found are very small and are just sufficient to move the poles of the ballon into the left half plane.

```
R1 = [[1e-7 0 0];[0 1e-7 0];[0 0 1e-8]];
R2 = 1;

[Pmatrix,eigs,K] = care(A,B,R1,R2);
T=eigs(1)
```

```
T = -0.0409
```

```
w=eigs(2)
```

```
w = -0.0150
```

```
h=eigs(3)
```

```
h = -0.0026
```

which are all very, very slow, but stable.

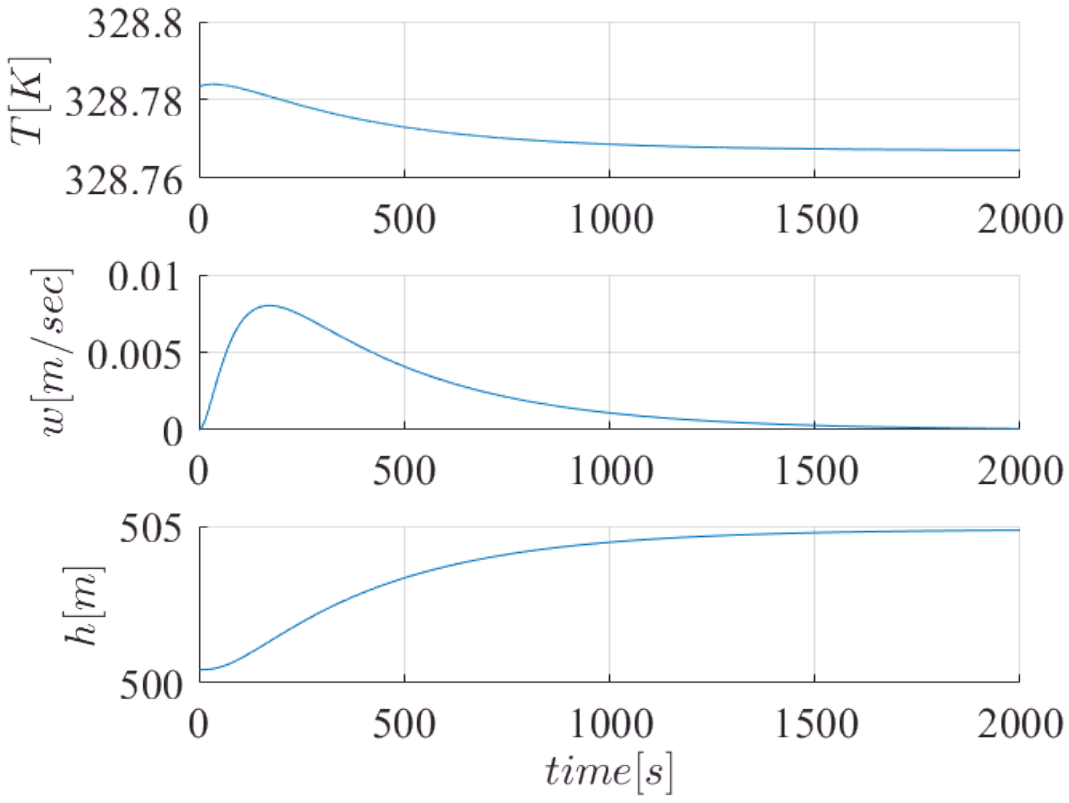
Problem 6 Use the mild LQR regulator designed above and confirm that it works as intended: that it can keep the nonlinear system at the desired operating point. Simulate the system over $2000 \quad s$. What happens if the initial conditions of the integrators do not agree with the values of q_0, w_s, h_s and T_s found above? What variables is the control system particularly sensitive to?

Solution:

Simulating the ballon control object with the mild feedback system designed. results in a stable system which is again very slow. If the initial conditions for the integration are different that those correstponding to the found using `trim`, the system at as though a step function has been inserted into that state as an input at the output of the relevant integrator. After this step, the ballon will return to its equilibrium operating condition, assuming that the step is not too large. Again it should be remembered that the Simulink model is a “true” representation

of the physical characteristics of the ballon. Using small positive initial conditions turns the ballon into an inert mass sitting on the ground, and it will not move.

```
qswitch = -1;
qin = qs; %Simulate at linearization point
sim('Q6',2000);

figure, h1 = subplot(3,1,1); set(h1,'FontName','times','FontSize',16);
hold on, grid on;
plot(T.time,T.signals.values);
ylabel('$T [K]$','FontName','times','FontSize',16,'Interpreter','latex');

h2 = subplot(3,1,2); set(h2,'FontName','times','FontSize',16);
hold on, grid on;
plot(w.time,w.signals.values);
ylabel('$w [m/sec]$','FontName','times','FontSize',16,'Interpreter','latex');

h3 = subplot(3,1,3); set(h3,'FontName','times','FontSize',16);
hold on, grid on;
plot(h.time,h.signals.values);
ylabel('$h [m]$','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('$time [s]$','FontName','times','FontSize',16,'Interpreter','latex');
```



Problem 7 State process noise is now to be inserted into the system. Use the noise scaling matrix $\mathbf{B}_v$:

$$\mathbf{B_v} = \begin{bmatrix} \frac{1}{\tau_1} & 0 & 0 \\ 0 & \frac{1}{\tau_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Use the state noise intensities $[\mathbf{V}]_{11} = 2K^2$, $[\mathbf{V}]_{22} = 0.1m^2s^{-2}$ and $[\mathbf{V}]_{33} = 0.01m^2$.

Simulate the feedback loop with these noise intensities and find the noise amplitudes on the states. Use the noise generator 'Random Number' in Simulink with a sample time of $1s$ and different seeds for each generator. Simulate over a period of $10000\ \ s$ and use only the last $5000\ \ s$ to find the output noise amplitudes. Do not attempt to make exact measurements here: use inspection and the $\pm 3\sigma$ rule. Evaluate the correctness of the simulation results using the Matlab function $covar$. Do the results of the simulation agree approximately with the theoretical calculations?

Measurements are to be made on the system in connection with the design of a full state feedback system for the balloon. The sensor measurement noise amplitudes are given here as maximum peak to peak values: T measurement: $\pm 1K$, w measurement: $\pm 1ms^{-1}$, h measurement: $\pm 5m$. Find the corresponding noise intensities.

Simulate the balloon with these noise intensities added to the state noise already present. How do the different noise intensities add? Is it desirable to use an LQR regulator on the system directly?

Solution:

Inserting noise generators into the $\mathbf{B_v}$ matrix and thereafter into the summing points in front of the integrators in the ballon object results in noise on the states of the system. Using the $\pm 3\sigma$ rule, the noise intensities (variances) which can be observed in the Simulink model and visual inspection are close to those which can be calculated theoretically $(\pm 10\%)$.

The calculation of the noise intensities on the ballon states is carried out using the “covar” function in the Matlab Control Toolbox. If the linearized $\mathbf{A}$ and $\mathbf{B}$ matrices and $\mathbf{K}$ are in the workspace then the necessary statements are

```
% first calculate the dynamic and input matrices
A = [-3.333e-3 0 -2e-5;0.20493 -0.02 7.477e-4;0 1 0];
B = [1e-3 0 0]';
Bv = diag([1/300 1/50 1]);
C = diag([1 1 1]);
D = [0];

% mild feedback gains
K = [35.2012 3.5953 0.12841];
Ak = A - B*K;

% assemble overall system
sys = ss(Ak,Bv,C,D);
```

```
% define state noise inputs
V1 = diag([2, 0.1 0.01]);
```

It is not the intention that these calculations be made with the full nonlinear system only with the linearized one. To compare these results with the full nonlinear system the intention is that the noise on the states is to be made by inspection, using the plus minus 3 sigma rule.

This results in the state covariance (Q) matrix of the system being calculated as

```
% calculate output (state) noise covariance
Q = covar(sys,V1)
```

```
Q =
    0.0065    0.0011   -1.6403
    0.0011    0.0120   -0.0050
   -1.6403   -0.0050  433.3937
```

It is the square root of the diagonal elements of this matrix that are the standard deviations of the state variations around their selected operating points. It should be noticed for example that the standard deviation of the height is $\sigma_h = \sqrt{433.39} = 20.82m$ which is large but occurs over a time period of on the order of 1000 sec = 16.67 min.

To this noise is added that which is caused by the measurements. The state noise observed after the measurements is on the order of

$$T : 6.5 \times 10^{-3} + (\frac{1}{3})^2 = 0.118\,K^2$$

$$w : 0.0120 + (\frac{1}{3})^2 = 0.123\,(m/sec)^2$$

$$h : 433.39 + 5^2 = 459.39m^2$$

The corresponding maximum peak to peak amplitudes can be found from the above by taking the square roots of the variances above and multiplying by 6.

```
%output (state) noise covariances can be converted to
%   amplitudes by taking the square roots of the diagonal
%   elements of Q
%
%The correct answer is:
%
%Q =
%
%    0.0002    0.0000   -0.0481
%    0.0000    0.0002   -0.0050
%   -0.0481   -0.0050   12.7988
```

Linear Control Design II - Group Work Problem Module 23

Problems

The group problem in this module is based on the air balloon model constructed in module 22. The purpose in this module is to design and test an LQR regulator for this system and then a LQG regulator (which includes the LQR regulator and the Kalman filter).

The main idea is to test the characteristics of an LQR regulator in the presence of noise and then to test an LQG regulator and compare it to the LQR regulator. The desired operating altitude is $h = 500\,\mathrm{m}$ with the same ambient conditions used earlier.

Problem 1 Design an LQR regulator for the balloon using the following weighting matrices

$$\mathbf{R}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10 \end{bmatrix}, \qquad \mathbf{R}_2 = 1$$

Based on the chosen weights which type of control action do you expect? Test the LQR regulator using a small step input around the 500 m operating point. Use a step up and then a step down to form a small symmetrical pulse: plus, minus and zero input. These steps should be at least 500 sec from each other to allow the system to reach equilibrium between pulses.

Solution:

1. First the constants are defined:

```
%  Initialize constants
clear all; close all; clc;
tau1=300;   %s
tau2=50;    %s
c=1e-3;     %(K/Kj*sec)
M=300;      %kg
V=2000;     %m^3
g=9.81;     %m/s^2
b=6;        %hz

rhoa0=1.22;  %kg/m^3;
Ta0=290;     %K
p=0.795e-4;
f=6e-3;
offset = 0.001;
```

2. The solution for the steady-state with $h_s = 500$ is computed:

```
hs = 500;   %m,   linearizing point
ws = 0;     %m/s, linearizing point
```

```
%% Solution From Module 16:
Tah=Ta0-f*hs;
rhoah=rhoa0-p*hs;
rho=-(M*g)/(V*g)+rhoah;

Ts=rhoah*Tah/rho;
q0=(1/tau1*(Ts-Tah))/c;

%% Linearisation point:
us = [q0]
```

```
us = 139.2866
```

```
xs = [Ts ws hs]'
```

```
xs =
  328.7860
         0
  500.0000
```

```
ys = xs;
```

3. The linearized system model is defined:

```
A = [    -1/tau1                              0     -f/tau1;
          g/(c*q0*tau1)*(1-M/(rhoah*V))  -b/M    g*V/M*((rhoah*f)/Ts-p*(1-
Tah/Ts));
          0                                    1      0
];
B=[c 0 0]';
Bv = diag([1/300 1/50 1]);
C = diag([1 1 1]);
D=0;
sys = ss(A,B,C,D); sys.u = 'q'; sys.y = {'T','w','h'}
```

```
sys =

  A =
              x1         x2         x3
   x1  -0.003333          0     -2e-05
   x2     0.2049      -0.02  0.0007478
   x3          0          1          0

  B =
            q
   x1  0.001
   x2      0
   x3      0

  C =
      x1  x2  x3
   T   1   0   0
   w   0   1   0
   h   0   0   1

  D =
      q
```

```
   T  0
   w  0
   h  0

Continuous-time state-space model.
```

3. Design Linear Quadratic Regulator:

```
% Design LQR
R1 = diag([1 1 10]);
R2 = 1;
[Kp1,S,~] = lqr(A,B,R1,R2); K = Kp1
```

```
K =
  156.9725   62.6696    3.7273
```

4. Test the LQR regulator (Simulation)

The LQR is tested using a small step input around the 500 m operating point. We Use a step up and then a step down to form a small symmetrical pulse: plus, minus and zero input. These steps should be at least 500 sec from each other to allow the system to reach equilibrium between pulses.

```
% Simulation
STEP_SIZE = 1;
stoc = 1;      % 0: Deterministic, 1: Stochastic

% Init Stochastics
tc = 2*STEP_SIZE;    % Define correlation time
V1 = diag([2 0.1 0.01]); % Process noice covariance
V2 = diag([1/3^2 1/3^2 (10/3)^2]); % Measurement Noice Covariance

% Generate Input Signal
delta_u= (0.1*us).*[zeros(1,500) ones(1,500) zeros(1,500) -1*ones(1,500)
zeros(1,500)];
% u     = us*ones(1,2500)+(0.1*us).*signal;
t=1:2500;

% Simulate System
x = xs;
sim('M17_Q1',2500, [],[t' delta_u']);
```

```
Unable to load block diagram 'M17_Q1'
```

```
figure, h0 = subplot(4,1,1); set(h0,'FontName','times','FontSize',16);
hold on, grid on;
plot(q.time,[us+r.signals.values,q.signals.values]);
ylabel('$q [kJ]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('ref', 'u')

h1 = subplot(4,1,2); set(h1,'FontName','times','FontSize',16);
```

```
hold on, grid on;
plot(sim_x.time,sim_x.signals.values(:,1));
ylabel('$T [K]$','FontName','times','FontSize',16,'Interpreter','latex');

h2 = subplot(4,1,3); set(h2,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,sim_x.signals.values(:,2));
ylabel('$w [m/sec]$','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('$time [s]$','FontName','times','FontSize',16,'Interpreter','latex');

h3 = subplot(4,1,4); set(h3,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,sim_x.signals.values(:,3));
ylabel('$h [m]$','FontName','times','FontSize',16,'Interpreter','latex');
```

Lets test that the noise enters the states correctly in the simulation

```
nstp = 20000;
t=1:nstp;
delta_u=repmat(0,1,nstp);
stoc = 1;      % 0: Deterministic, 1: Stochastic
sim('M17_Q1',nstp, [],[t' delta_u']);

% Estimated covariance matrix for the states
cov_est = cov(sim_x.signals.values)
```

```
cov_est =
    0.0002    0.0000   -0.0062
    0.0000    0.0009   -0.0055
   -0.0062   -0.0055    0.3626
```

```
% Calculated from linear model
cov_lyap = lyap(A-B*K,Bv*V1*Bv')
```

```
cov_lyap =
    0.0002    0.0000   -0.0061
    0.0000    0.0009   -0.0050
   -0.0061   -0.0050    0.3457
```

```
% Difference
cov_est - cov_lyap
```

```
ans =
   -0.0000    0.0000   -0.0001
    0.0000    0.0000   -0.0005
   -0.0001   -0.0005    0.0169
```

```
% Check the noise source !
% Notice that we must multiply with tc here to get the expected value,
% this is because we directly and discretely samples the continous noise
approximation, see page 410 in book.
```

```
cov_est_v1 = cov(sim_v1.signals.values) * tc
```

```
cov_est_v1 =
    1.9875    0.0037    0.0008
    0.0037    0.0996    0.0005
    0.0008    0.0005    0.0100
```

```
V1
```

```
V1 =
    2.0000         0         0
         0    0.1000         0
         0         0    0.0100
```

5. Test the slow LQR regulator (Simulation)

```
% Generate Input Signal
stoc = 1;      % 0: Deterministic, 1: Stochastic
delta_u= (0.1*us).*[zeros(1,500) ones(1,500) zeros(1,500) -1*ones(1,500)
zeros(1,500)];
t=1:2500;

% the old feedback gain (from Group work 16)
K = [35.2012 3.5953 0.12841];

sim('M17_Q1',2500, [],[t' delta_u']);
figure, h0 = subplot(4,1,1); set(h0,'FontName','times','FontSize',16);
hold on, grid on;
plot(q.time,[us+r.signals.values,q.signals.values]);
ylabel('$q [kJ]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('ref', 'u')

h1 = subplot(4,1,2); set(h1,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,sim_x.signals.values(:,1));
ylabel('$T [K]$','FontName','times','FontSize',16,'Interpreter','latex');

h2 = subplot(4,1,3); set(h2,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,sim_x.signals.values(:,2));
ylabel('$w [m/sec]$','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('$time [s]$','FontName','times','FontSize',16,'Interpreter','latex');

h3 = subplot(4,1,4); set(h3,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,sim_x.signals.values(:,3));
ylabel('$h [m]$','FontName','times','FontSize',16,'Interpreter','latex');
```

It is clear that the old feedback gain is not as well suited as the new one (found in part 1).

Problem 2 Design now a Kalman filter for this system using the noise assumptions in **Problem 7** of **Group Work Module 19**. The Matlab routine `kalman` or `lqe` can be used for this purpose. Look at the command syntax in the Matlab help. Test the Kalman filter in the Simulink model using the double pulse input used to test the LQR regulator. Plot both the state outputs and state estimates on the same plot. Below this, plot the state errors for the three states. Is the filtering sufficient? How can it be improved?

Solution:

1. Design Kalman Filter

```
% Kalman
Bv = diag([1/tau1 1/tau2 1]);
V1 = diag([2 0.1 0.01]);
V2 = diag([1/3^2 1/3^2 (10/3)^2]);

sys = ss(A-B*K,Bv,C,D);
[L,P,E] = lqe(A,Bv,C,V1,V2); L
```

```
L =
    0.0046    0.0103    0.0006
    0.0103    0.0418    0.0041
    0.0631    0.4095    0.0859
```

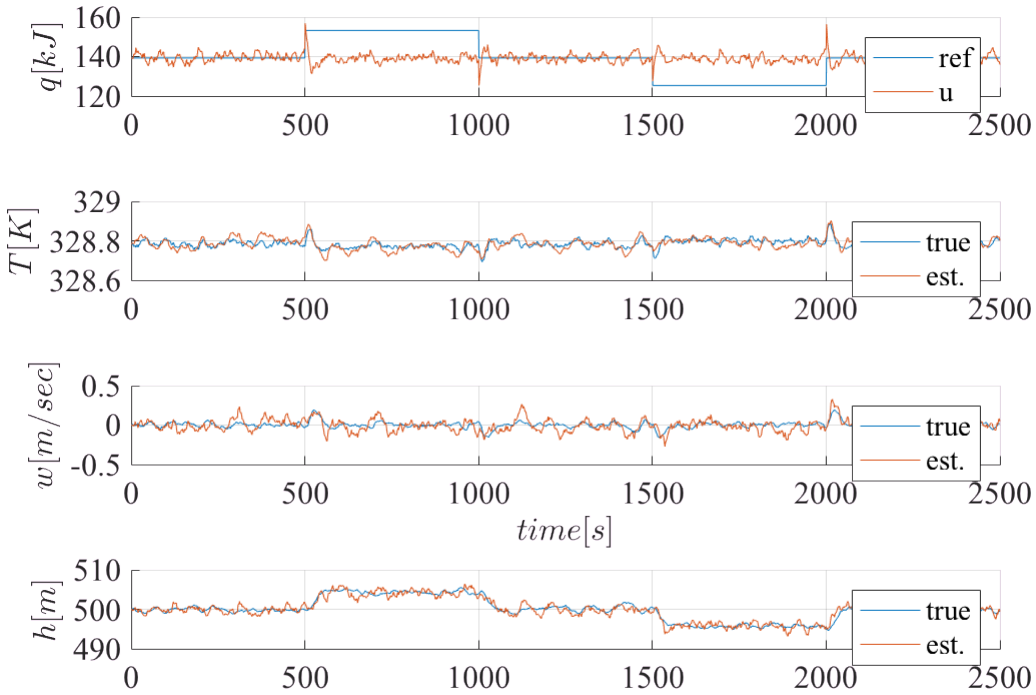
2. Test Kalman Filter (compare to true states in simulation) - Still LQR

```
% the feedback gain found in Problem 1
K = Kp1
```

```
K =
  156.9725   62.6696    3.7273
```

```
% Run stochastic simulation
stoc = 1;      % 0: Deterministic, 1: Stochastic
sim('M17_Q2',2500, [],[t' delta_u']);
figure, h0 = subplot(4,1,1); set(h0,'FontName','times','FontSize',16);
hold on, grid on;
plot(q.time,[us+r.signals.values,q.signals.values]);
ylabel('$q [kJ]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('ref', 'u')

h1 = subplot(4,1,2); set(h1,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,[sim_x.signals.values(:,1),sim_xhat.signals.values(:,1)]);
ylabel('$T [K]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('true', 'est.')

h2 = subplot(4,1,3); set(h2,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,[sim_x.signals.values(:,2),sim_xhat.signals.values(:,2)]);
ylabel('$w [m/sec]$','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('$time [s]$','FontName','times','FontSize',16,'Interpreter','latex');
```

```
legend('true', 'est.')

h3 = subplot(4,1,4); set(h3,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,[sim_x.signals.values(:,3),sim_xhat.signals.values(:,3)]);
ylabel('$h [m]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('true', 'est.')
```



Problem 3 Now use the state estimates to close the loop using the LQR regulator designed in **Problem 1**. It may be necessary to adjust the state weights to increase the feedback level to achieve stability. Does the Kalman filter improve the quality of the regulator (reduce the state error)?

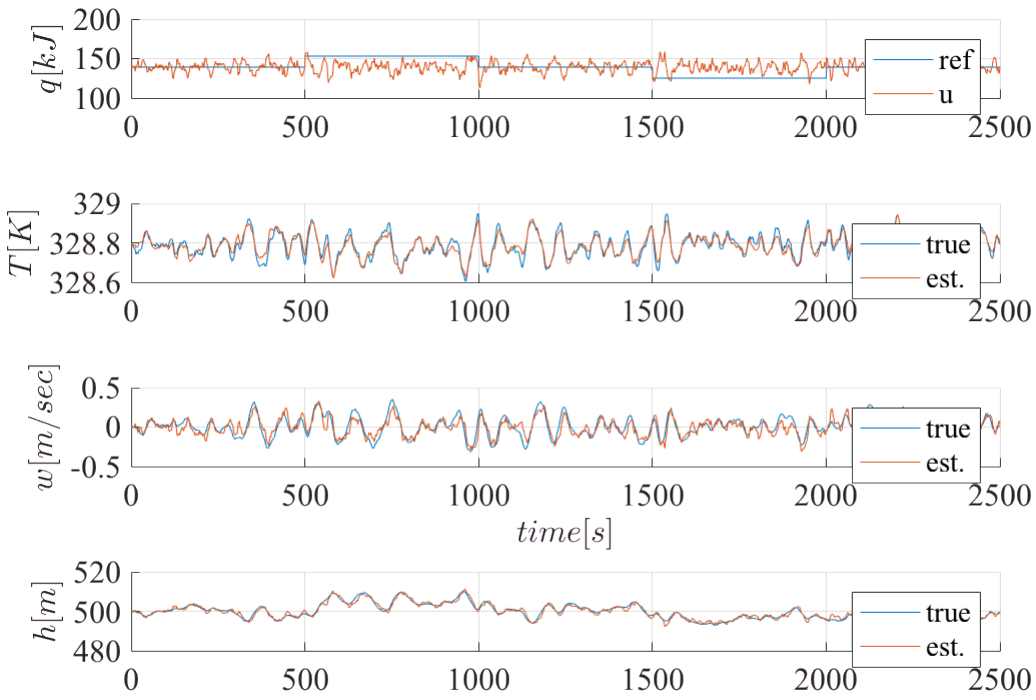
Solution:

```
% the feedback gain found in Problem 1
K = Kp1
```

```
K =
  156.9725   62.6696    3.7273
```

```
% Run stochastic simulation
stoc = 1;      % 0: Deterministic, 1: Stochastic
sim('M17_Q3',2500, [],[t' delta_u']);
figure, h0 = subplot(4,1,1); set(h0,'FontName','times','FontSize',16);
```

```
hold on, grid on;
plot(q.time,[us+r.signals.values,q.signals.values]);
ylabel('$q [kJ]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('ref', 'u')

h1 = subplot(4,1,2); set(h1,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,[sim_x.signals.values(:,1),sim_xhat.signals.values(:,1)]);
ylabel('$T [K]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('true', 'est.')

h2 = subplot(4,1,3); set(h2,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,[sim_x.signals.values(:,2),sim_xhat.signals.values(:,2)]);
ylabel('$w [m/sec]$','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('$time [s]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('true', 'est.')

h3 = subplot(4,1,4); set(h3,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,[sim_x.signals.values(:,3),sim_xhat.signals.values(:,3)]);
ylabel('$h [m]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('true', 'est.')
```



Problem 4 The LQR has been designed to work at an altitude of 500 m. It is of interest to find out how robust it is with respect to changes in its operating point. Change the reference for the system so that the system operates at 400 m without changing the feedback gains. Is the system still stable? What does this say about the robustness of LQR regulators. For this simulations use the LQR regulator alone without the Kalman filter.

Solution:

1 . Close loop with LQR

```
% Compute new values for system model
hs400 = 400;   %m,   linearizing point
ws400 = 0;     %m/s, linearizing point

%% Solution From Module 16:
Tah=Ta0-f*hs400;
rhoah=rhoa0-p*hs400;
rho=-(M*g)/(V*g)+rhoah;
Ts400=rhoah*Tah/rho;
q0=(1/tau1*(Ts400-Tah))/c;
us400=q0
```

```
us400 = 138.5090
```

```
xs400 = [Ts400 ws400 hs400]'
```

```
xs400 =
  329.1527
         0
  400.0000
```

```
% the feedback gain found in Problem 1
K = Kp1
```

```
K =
  156.9725   62.6696    3.7273
```

```
% Run stochastic simulation
stoc = 1;      % 0: Deterministic, 1: Stochastic
delta_u= (0.1*us).*[zeros(1,500) ones(1,500) zeros(1,500) -1*ones(1,500)
zeros(1,500)];
t=1:2500;
sim('M17_Q4',2500, [],[t' delta_u']);
figure, h0 = subplot(4,1,1); set(h0,'FontName','times','FontSize',16);
hold on, grid on;
plot(q.time,[us400+r.signals.values,q.signals.values]);
ylabel('$q [kJ]$','FontName','times','FontSize',16,'Interpreter','latex');
legend('ref', 'u')

h1 = subplot(4,1,2); set(h1,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,sim_x.signals.values(:,1));
```

```
ylabel('$T [K]$','FontName','times','FontSize',16,'Interpreter','latex');

h2 = subplot(4,1,3); set(h2,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,sim_x.signals.values(:,2));
ylabel('$w [m/sec]$','FontName','times','FontSize',16,'Interpreter','latex');
xlabel('$time [s]$','FontName','times','FontSize',16,'Interpreter','latex');

h3 = subplot(4,1,4); set(h3,'FontName','times','FontSize',16);
hold on, grid on;
plot(sim_x.time,sim_xhat.signals.values(:,3));
ylabel('$h [m]$','FontName','times','FontSize',16,'Interpreter','latex');
```