Example 3.1. **Second Order LTI System**

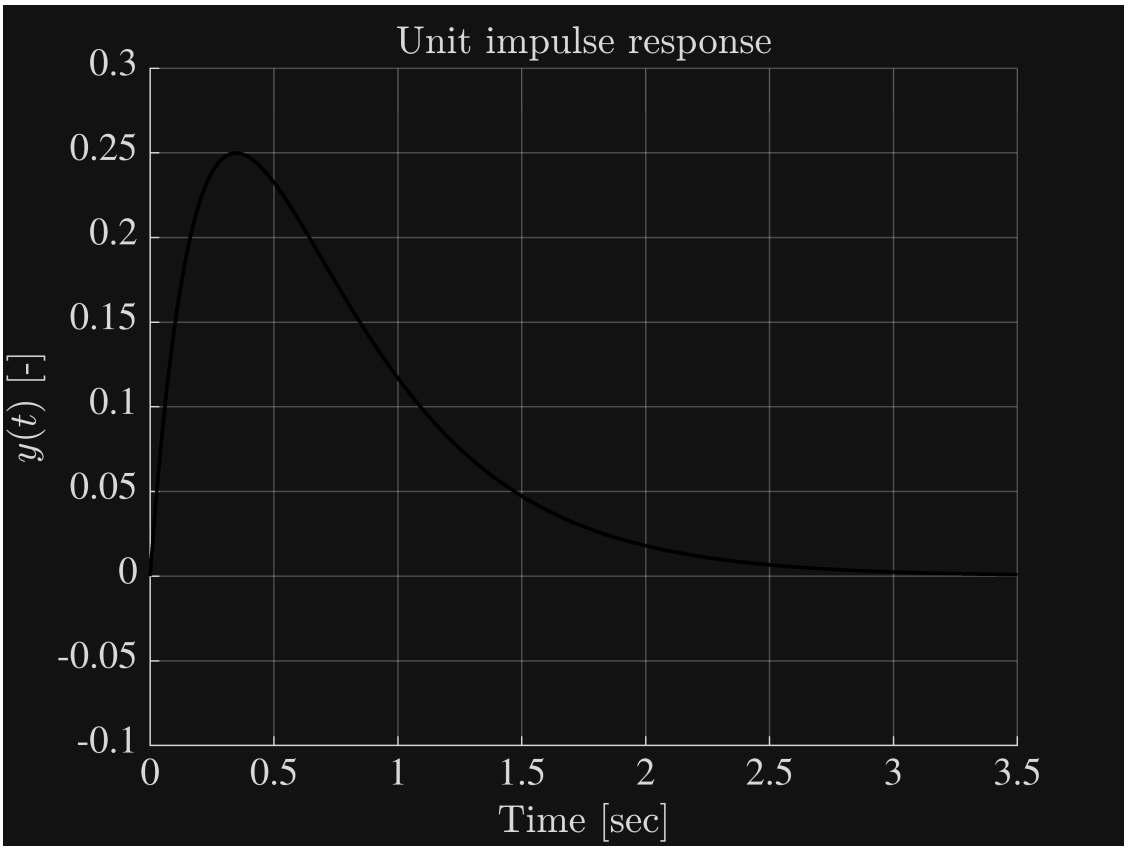
A second Order LTI-System is given by the equations.

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -8 & -6 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u,$$

$$y = \begin{bmatrix} 2 & 0 \end{bmatrix} \mathbf{x}.$$

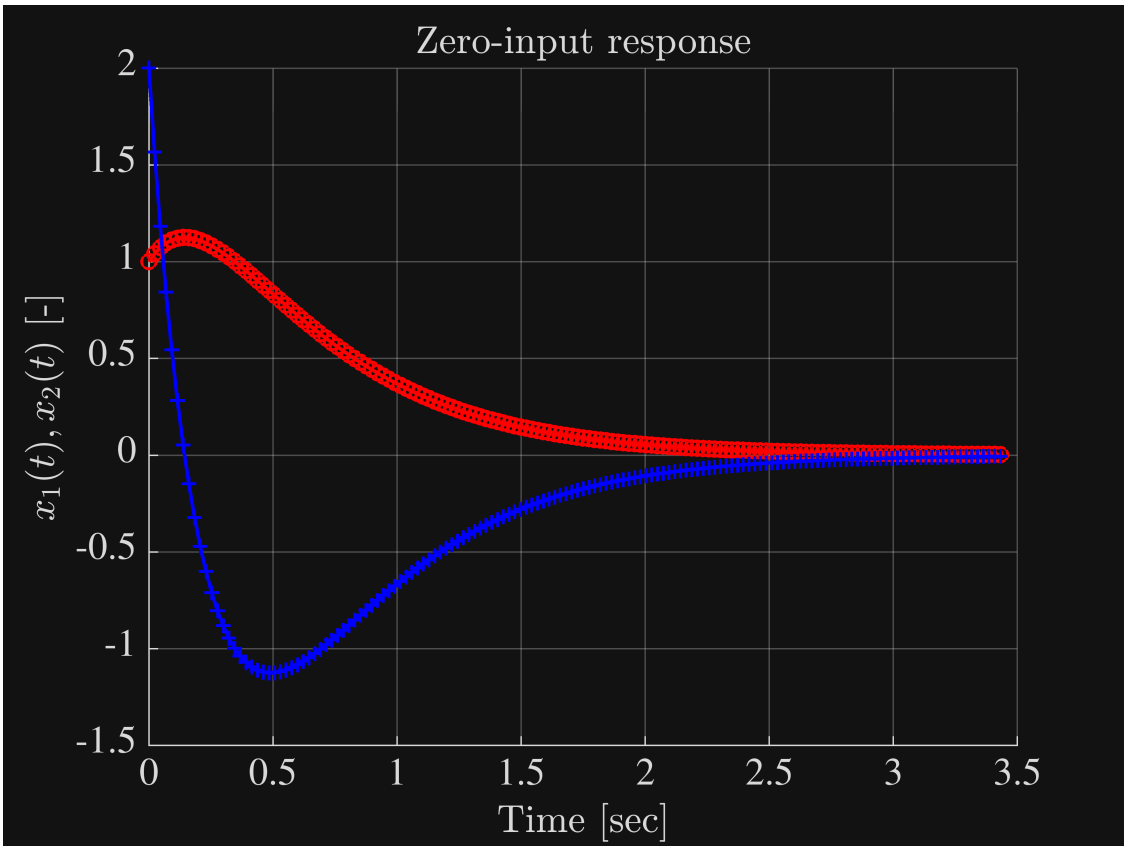
The system is SISO and thus u and y are scalars.

```
clear all
close all
clc

% Example 3.1 (Textbook) - Zero state and zero input solutions
A = [0 1;-8 -6];
B = [0 1]';
C = [2 0];
D = [];
sys = ss(A,B,C,D);

% impulse response
[y_imp t1] = impulse(sys);
figure, h1 = axes; set(h1,'FontName','times','FontSize',16)
hold on, grid on
plot(t1,y_imp,'k','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$$y(t)$$ [-]','FontName','times','FontSize',16,'interpreter','latex')
title('Unit impulse
response','FontName','times','FontSize',16,'interpreter','latex')
axis([0 3.5 -0.1 0.3])
```

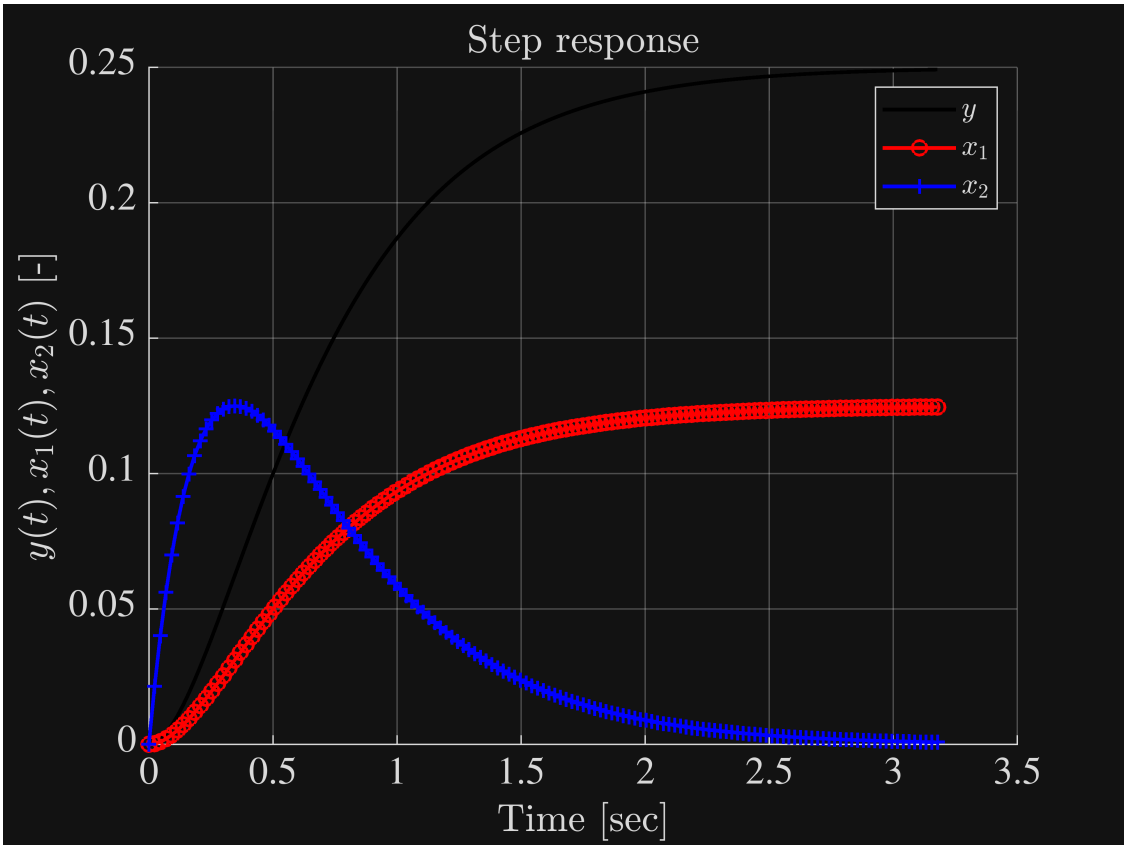



```
% zero-input response
[y_zi t2 x_zi] = initial(sys,[1 2]');
figure, h2 = axes; set(h2,'FontName','times','FontSize',16)
```

```
hold on, grid on
plot(t2,x_zi(:,1),'-or',t2,x_zi(:,2),'-+b','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$$x_1(t),x_2(t)$$
[-]','FontName','times','FontSize',16,'interpreter','latex')
title('Zero-input
response','FontName','times','FontSize',16,'interpreter','latex')
```




```
% step response
[y_step t3 x_step] = step(sys);
figure, h2 = axes; set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(t3,y_step,'k',t3,x_step(:,1),'-or',t3,x_step(:,2),'-+b','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$$y(t),x_1(t),x_2(t)$$
[-]','FontName','times','FontSize',16,'interpreter','latex')
title('Step response','FontName','times','FontSize',16,'interpreter','latex')
leg = legend('$y$','$x_1$','$x_2$');
set(leg,'FontName','times','FontSize',12,'interpreter','latex')
```


Step response
0.25
0.2
0.15
0.1
0.05
0
$y(t), x_1(t), x_2(t)$ [-]
0
0.5
1
1.5
2
2.5
3
3.5
Time [sec]
y
x_1
x_2

Two Tank System

```
clear all
close all
clc
```

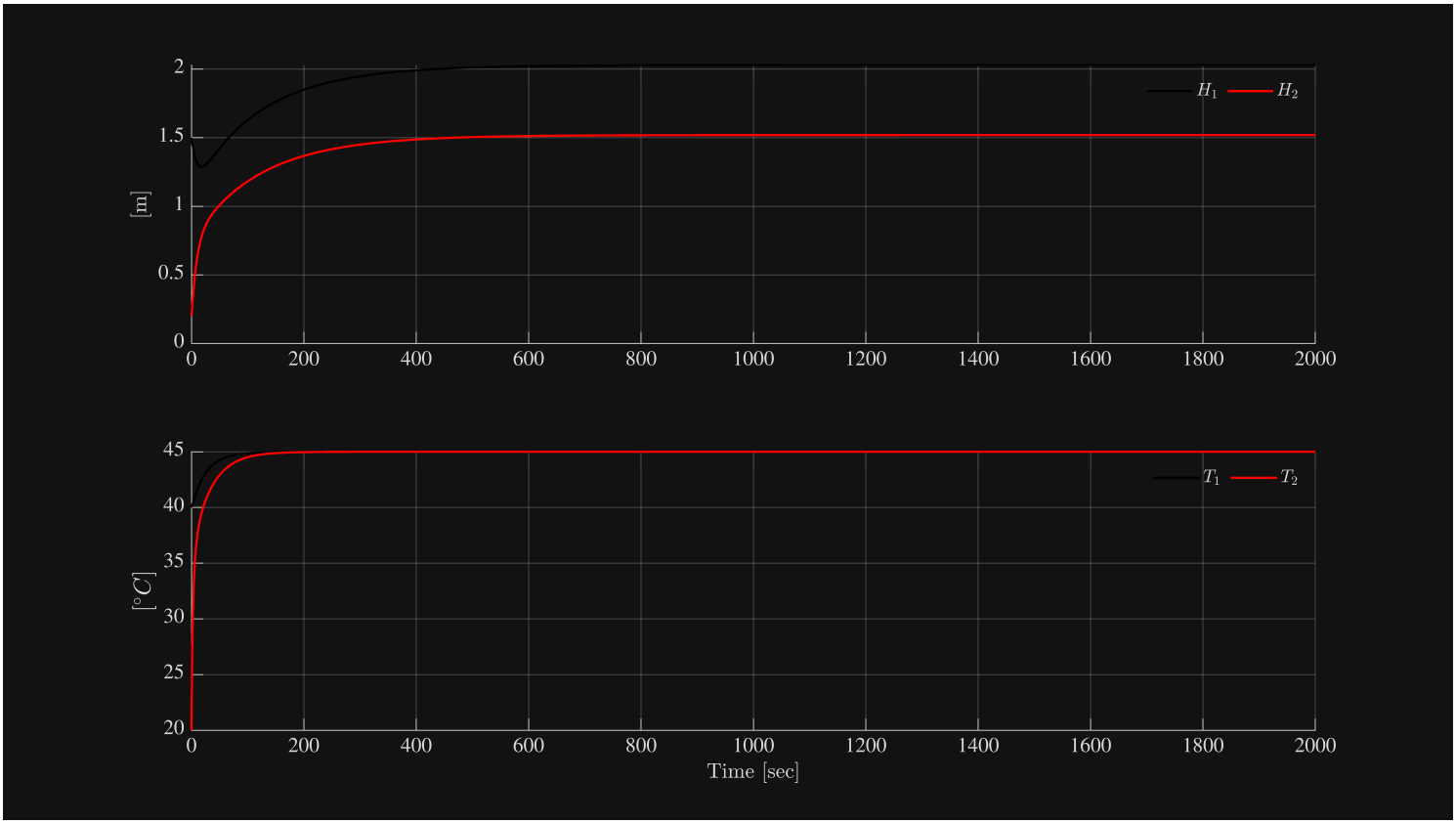
Configure simulation environment

```
SIM_TIME = 2000; % simulation time [sec]
STEP_SIZE = 0.05; % fundamental integration step size for running
simulations [sec]
```

Model parameters

```
At = 0.785; % [m^2]
Dv = 2.66; % [m^(1/2)/s]
C0 = 0.056; % [[m^(5/2)/s]]
ka = 0.004; % [m^3/(V*s)]
kh = 2; % [V/m]
kt = 0.1; % [V/C]

% Input and disturbance working point
uw0 = 5; % hot water valve [V]
u10 = uw0;
uc0 = 5; % cold water valve [V]
u20 = uc0;
Av0 = 0.0122; % area of outlet valve [m^2]
v10 = Av0;
Tw0 = 60; % temperature of the hot water [C]
v20 = Tw0;
Tc0 = 30; % temperature of the cold water [C]
v30 = Tc0;

% Simulate the system to evaluate the stationary state in connection with
% the chosen values of input and disturbances
x0 = [1.5,0.2,40,20]'; % initial conditions for the state variables x0 =
[H1,H2,T1,T2]'
sim('TwoTankSystem_SteadyState',SIM_TIME,[],[])
x0_S = xFinal'; % simulated steady state condition

% Plot of state responses
t = logsout.getElement('H1').Values.Time;
H1 = logsout.getElement('H1').Values.Data;
H2 = logsout.getElement('H2').Values.Data;
T1 = logsout.getElement('T1').Values.Data;
T2 = logsout.getElement('T2').Values.Data;

figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
```

```
hf1 = subplot(2,1,1); set(hf1,'FontName','times','FontSize',14)
hold on, grid on
plot(t,H1,'k',t,H2,'r','LineWidth',1.5)
ylabel('[m]','FontName','times','FontSize',14,'Interpreter','latex')
leg1 =
legend('$H_1$','$H_2$','Location','NorthEast','Orientation','horizontal');
set(leg1,'FontName','times','FontSize',12,'Interpreter','latex','box','off')
hf2 = subplot(2,1,2); set(hf2,'FontName','times','FontSize',14)
hold on, grid on
plot(t,T1,'k',t,T2,'r','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
ylabel('$[^\circ C]$','FontName','times','FontSize',16,'Interpreter','latex')
leg2 =
legend('$T_1$','$T_2$','Location','NorthEast','Orientation','horizontal');
set(leg2,'FontName','times','FontSize',12,'Interpreter','latex','box','off')
```



Stationary state analytically determined

```
x20 = ka^2/(Dv^2*v10^2)*(u10+u20)^2;
H20 = x20;
x10 = x20 + ka^2/C0^2*(u10+u20)^2;
H10 = x10;
x30 = (u10*v20+u20*v30)/(u10+u20);
T10 = x30;
x40 = x30;
```

```
T20 = x40;
x0_A = [x10 x20 x30 x40]'; % analytical steady state condition

% Compare the simulated result with the analytically calculated one
disp([x0_S x0_A])
```

```
    2.0295    2.0295
    1.5193    1.5193
   45.0000   45.0000
   45.0000   45.0000
```

Numerically linearize the nonlinear model around the stationary state

```
u0 = [u10 u20 v10 v20 v30]';
y0 = [kh*x0(1);kt*x0(4)];

% Another method to determine the stationary state is to use the command
% TRIM (read help)
% [xss,uss,yss,dx] = trim('TwoTankSystem_Linearize',x0,u0,y0);

% Numerical linearization of a nonlinear system implemented in Simulink can
% be achieved through the command LINMOD (read help)
[An,Bn,Cn,Dn] = linmod('TwoTankSystem_Linearize',x0_S,u0);

% Present the linearized model
disp('Numerically linearized model of the Two Tank System')
```

```
Numerically linearized model of the Two Tank System
```

```
disp('matrix A')
```

```
matrix A
```

```
disp(An)
```

```
   -0.0499    0.0499         0         0
    0.0499   -0.0667         0         0
   -0.0000         0   -0.0251         0
    0.0000   -0.0000    0.0335   -0.0335
```

```
disp('matrix B')
```

```
matrix B
```

```
disp(Bn(:,1:2))
```

```
    0.0051    0.0051
         0         0
    0.0377   -0.0377
         0         0
```

```
disp('matrix Bv')
```

```
matrix Bv
```

```
disp(Bn(:,3:5))
```

```
         0         0         0
   -4.1767         0         0
         0    0.0126    0.0126
         0         0         0
```

```
disp('matrix C')
```

```
matrix C
```

```
disp(Cn)
```

```
         0    2.0000         0         0
         0         0         0    0.1000
```

```
disp('matrix D')
```

```
matrix D
```

```
disp(Dn(:,1:2))
```

```
     0     0
     0     0
```

```
disp('matrix Dv')
```

```
matrix Dv
```

```
disp(Dn(:,3:5))
```

```
     0     0     0
     0     0     0
```

Linear model analytically determined

```
a11 = -C0/(2*At*sqrt(x10-x20));
a12 = C0/(2*At*sqrt(x10-x20));
a21 = C0/(2*At*sqrt(x10-x20));
a22 = -C0/(2*At*sqrt(x10-x20)) - Dv*v10/(2*At*sqrt(x20));
a33 = -ka*(u10+u20)/(At*x10);
a43 = C0*sqrt(x10-x20)/(At*x20);
a44 = -C0*sqrt(x10-x20)/(At*x20);

A = [a11 a12 0 0;a21 a22 0 0;0 0 a33 0;0 0 a43 a44];

b11 = ka/At;
b12 = b11;
b31 = ka*(v20-x30)/(At*x10);
b32 = ka*(v30-x30)/(At*x10);

B = [b11 b12;0 0;b31 b32;0 0];

bv21 = -Dv*sqrt(x20)/At;
bv32 = ka*u10/(At*x10);
bv33 = ka*u20/(At*x10);
```

```
Bv = [0 0 0;bv21 0 0;0 bv32 bv33;0 0 0];

C = [0 kh 0 0;0 0 0 kt];
D = zeros(2,2);
Dv = zeros(2,3);

TwoTank = ss(A,[B Bv],C,[D Dv]);

% Present the linearized model
disp('Analytically computed linearized model of the Two Tank System')
```

```
Analytically computed linearized model of the Two Tank System
```

```
disp('matrix A')
```

```
matrix A
```

```
disp(A)
```

```
   -0.0499    0.0499         0         0
    0.0499   -0.0667         0         0
         0         0   -0.0251         0
         0         0    0.0335   -0.0335
```

```
disp('matrix B')
```

```
matrix B
```

```
disp(B)
```

```
    0.0051    0.0051
         0         0
    0.0377   -0.0377
         0         0
```

```
disp('matrix Bv')
```

```
matrix Bv
```

```
disp(Bv)
```

```
         0         0         0
   -4.1767         0         0
         0    0.0126    0.0126
         0         0         0
```

```
disp('matrix C')
```

```
matrix C
```

```
disp(Cn)
```

```
         0    2.0000         0         0
         0         0         0    0.1000
```

```
disp('matrix D')
```

```
matrix D
```

```
disp(D)
```

```
     0     0
     0     0
```

```
disp('matrix Dv')
```

```
matrix Dv
```

```
disp(Dv)
```

```
     0     0     0
     0     0     0
```

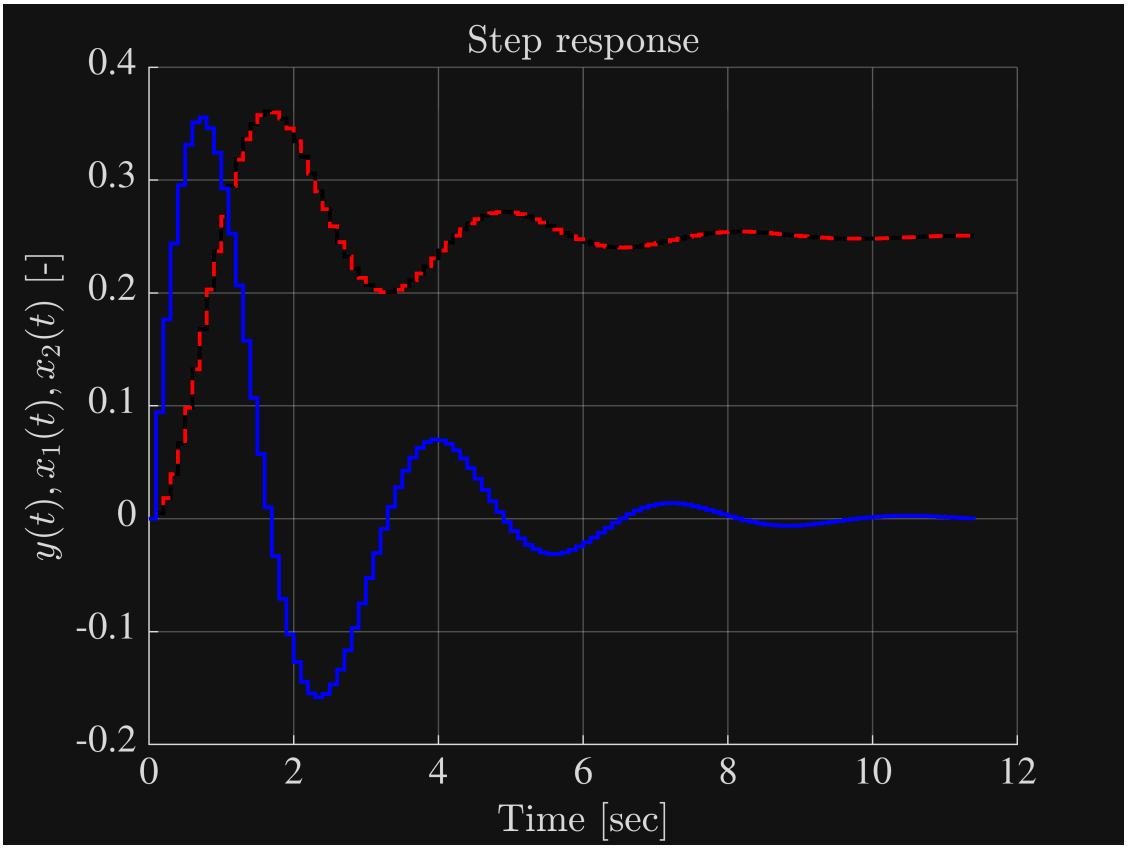
Make a simulation comparing the output at steady state of the linearized and nonlinear system (why don't you try?)

Example 3.6. **Time Response of a Discrete Time System**

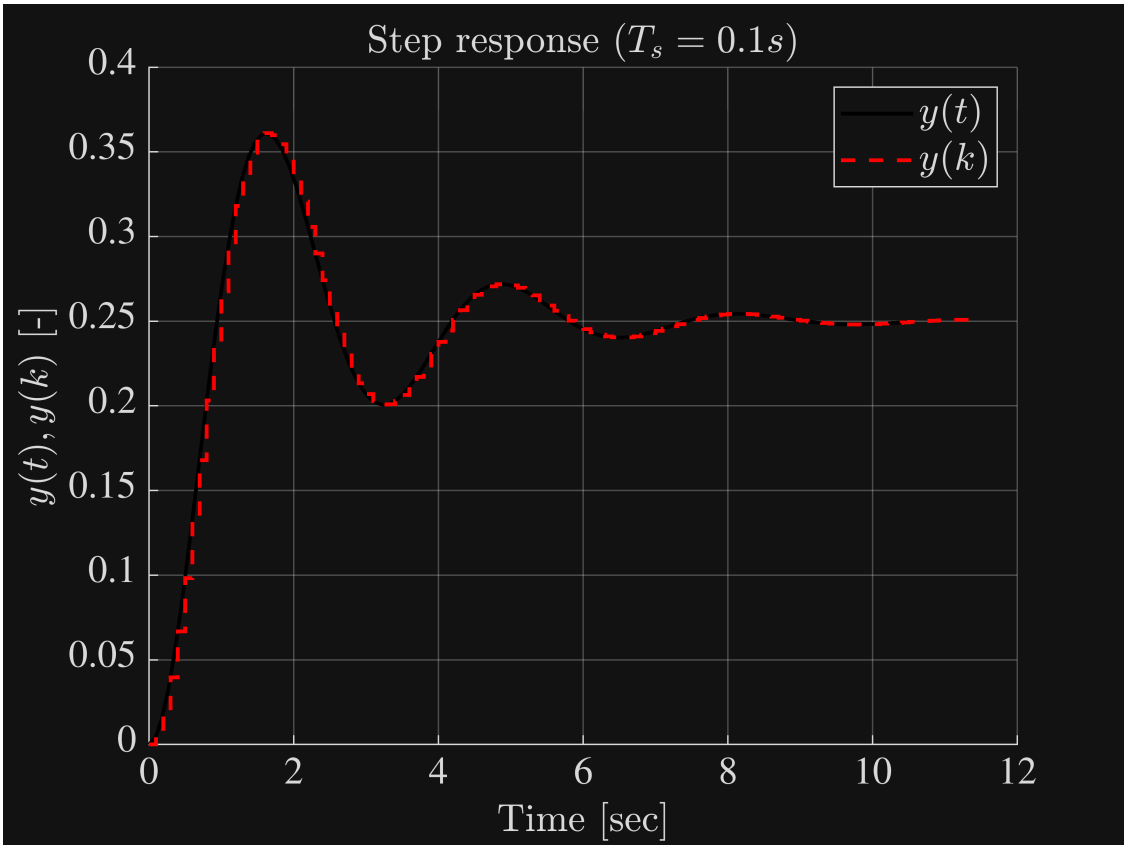
Suppose that a system is described by the matrices:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -4 & -1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{C} = [1\ 0], \mathbf{D} = 0.$$

```
clear all
close all
clc

% Example 3.6 (Textbook) - Discretization of Continuous Time System
A = [0 1;-4 -1];
B = [0 1]';
C = [1 0];
D = 0;

lambda_A = eig(A);
[Wn1 Z1] = damp(A);

% applying formulas page 75
Wn2 = sqrt(real(lambda_A).^2+imag(lambda_A).^2);
Z2 = -real(lambda_A)./Wn2;

% Discretization of the system (Ts = 0.1 sec)
Ts = 0.1;
sys = ss(A,B,C,D);
sysd = c2d(sys,Ts);

lambda_F1 = eig(sysd.a);
% applying Frobenius theorem
Lambda_F = exp(Ts*lambda_A);

% Discrete time system responses
% zero state response
[y_zs td x_zs] = step(sysd);
figure, h1 = axes; set(h1,'FontName','times','FontSize',16)
hold on, grid on
stairs(td,y_zs,'k','LineWidth',1.5)
stairs(td,x_zs(:,1),'--r','LineWidth',1.5)
stairs(td,x_zs(:,2),'b','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$$y(t),x_1(t),x_2(t)$$
[-]','FontName','times','FontSize',16,'interpreter','latex')
title('Step response','FontName','times','FontSize',16,'interpreter','latex')
```

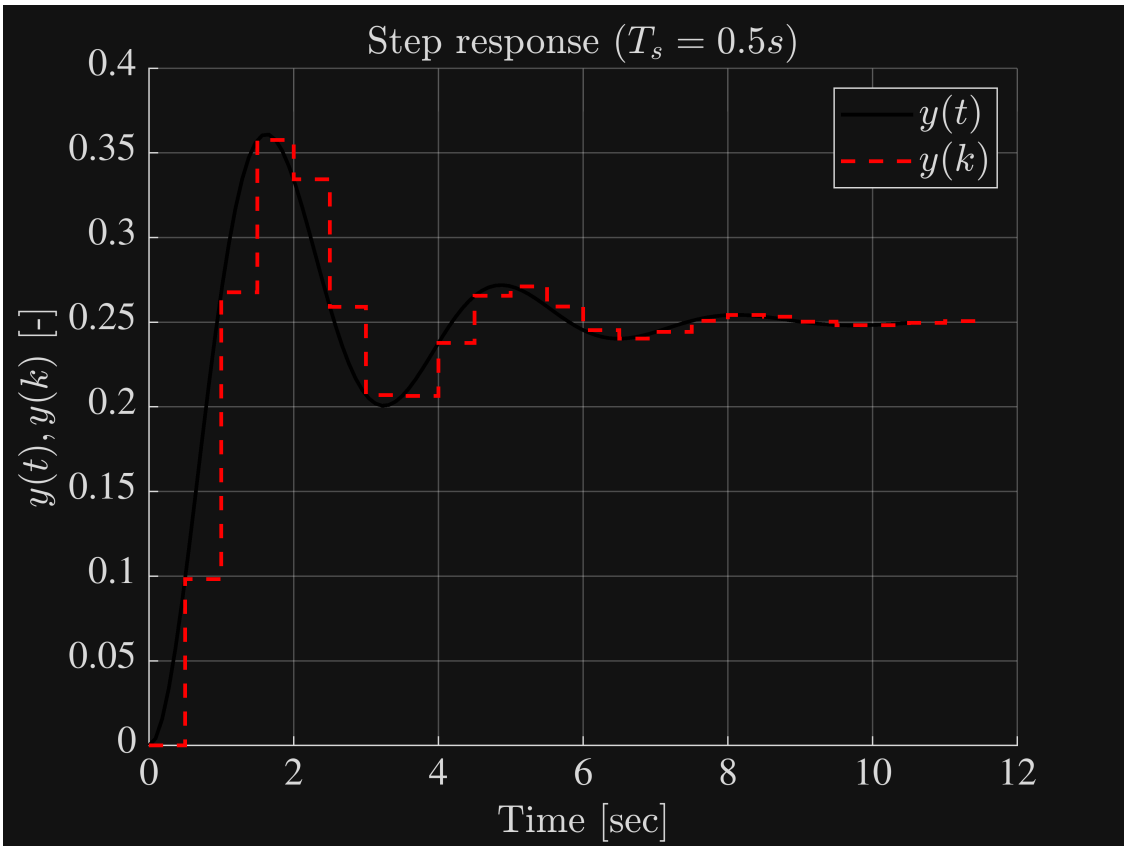



```
% Discrete Vs continuous time system response
[yC_zs tc xC_zs] = step(sys);
figure, h2 = axes; set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(tc,yC_zs,'k','LineWidth',1.5)
stairs(td,y_zs,'--r','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$$y(t),y(k)$$
[-]','FontName','times','FontSize',16,'interpreter','latex')
title('Step response ($T_s =
0.1s$)','FontName','times','FontSize',16,'interpreter','latex')
l1 = legend('$$y(t)$$','$$y(k)$$');
set(l1,'FontName','times','FontSize',16,'interpreter','latex')
```

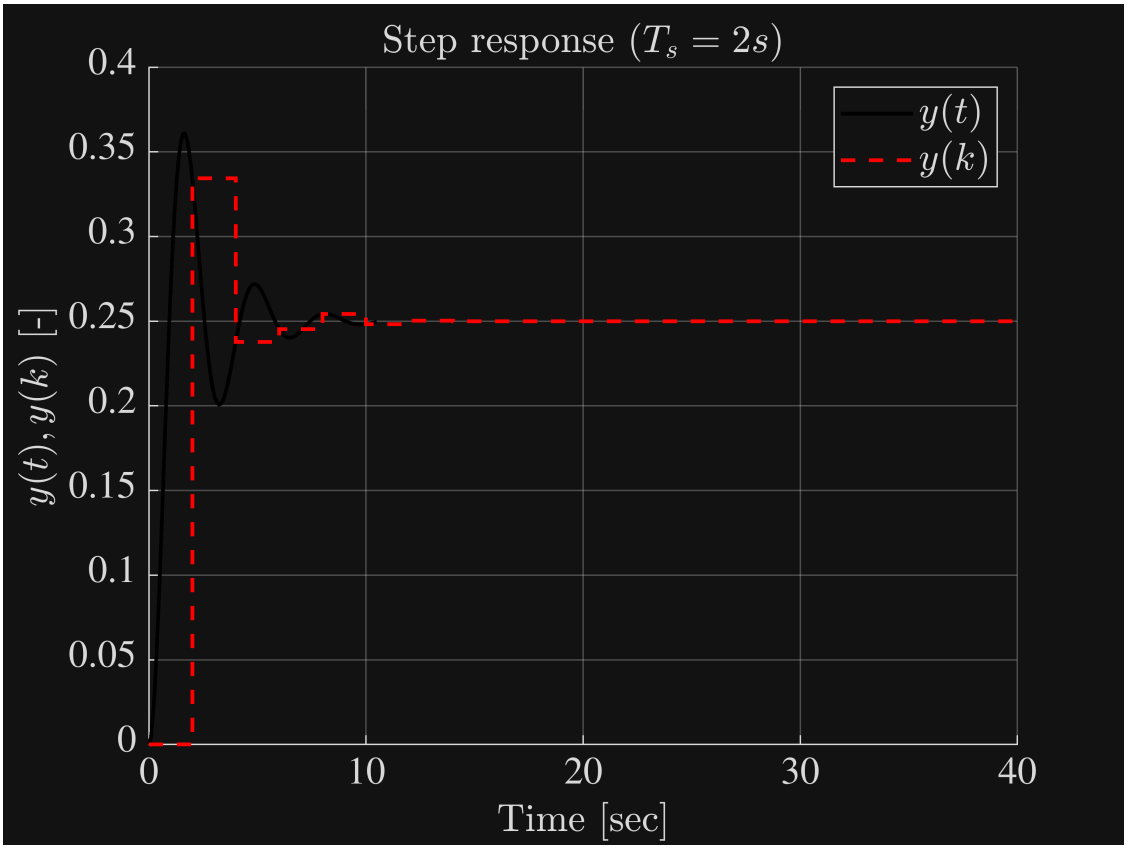



```
% Transfer function (Ts = 0.1 sec)
[nH dH] = ss2tf(sysd.a,sysd.b,sysd.c,sysd.d);
H = tf(nH,dH,Ts);
[r p k] = residue(nH,dH); % partial fraction expansion for SISO systems
Wn = abs(log(p(1)))/Ts;
Z = -cos(angle(log(p(1))));

% Discretization of the system (Ts = 0.5 sec)
Ts = 0.5;
sys = ss(A,B,C,D);
sysd = c2d(sys,Ts);

lambda_F2 = eig(sysd.a);

[y_zs td x_zs] = step(sysd);
figure, h3 = axes; set(h3,'FontName','times','FontSize',16)
hold on, grid on
plot(tc,yC_zs,'k','LineWidth',1.5)
stairs(td,y_zs,'--r','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$$y(t),y(k)$$
[-]','FontName','times','FontSize',16,'interpreter','latex')
title('Step response ($T_s =
0.5s$)','FontName','times','FontSize',16,'interpreter','latex')
l2 = legend('$$y(t)$$','$$y(k)$$');
```

```
set(l2,'FontName','times','FontSize',16,'interpreter','latex')
```




```
% Discretization of the system (Ts = 2 sec)
Ts = 2;
sys = ss(A,B,C,D);
sysd = c2d(sys,Ts);

lambda_F3 = eig(sysd.a);

[y_zs td x_zs] = step(sysd);
figure, h4 = axes; set(h4,'FontName','times','FontSize',16)
hold on, grid on
plot(tc,yC_zs,'k','LineWidth',1.5)
stairs(td,y_zs,'--r','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',16,'interpreter','latex')
ylabel('$$y(t),y(k)$$
[-]','FontName','times','FontSize',16,'interpreter','latex')
title('Step response ($T_s =
2s$)','FontName','times','FontSize',16,'interpreter','latex')
l3 = legend('$$y(t)$$','$$y(k)$$');
set(l3,'FontName','times','FontSize',16,'interpreter','latex')
```


Step response ($T_s = 2s$)
$y(t)$
$y(k)$
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
$y(t), y(k)$ [-]
0
10
20
30
40
Time [sec]

Example 3.9. **Stability of a Phase Variable System**

In this example consider a continuous time 4th order system with three different eigenvalue locations,

$$\text{a.}\begin{Bmatrix} -1.5 \\ -2 \\ -0.3 \pm j1.5 \end{Bmatrix} \quad \text{b.}\begin{Bmatrix} 0 \\ -2 \\ -0.3 \pm j1.5 \end{Bmatrix} \quad \text{c.}\begin{Bmatrix} 0 \\ 0 \\ -0.3 \pm j1.5 \end{Bmatrix}$$

The matrices are

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{C} = [1 \quad 0 \quad 0 \quad 0], \quad \mathbf{D} = 0.$$

As a matter of fact only the system matrix **A** is important here since the stability will be investigated according to stability definitions 1 and 2.

The characteristic polynomial of **A** is

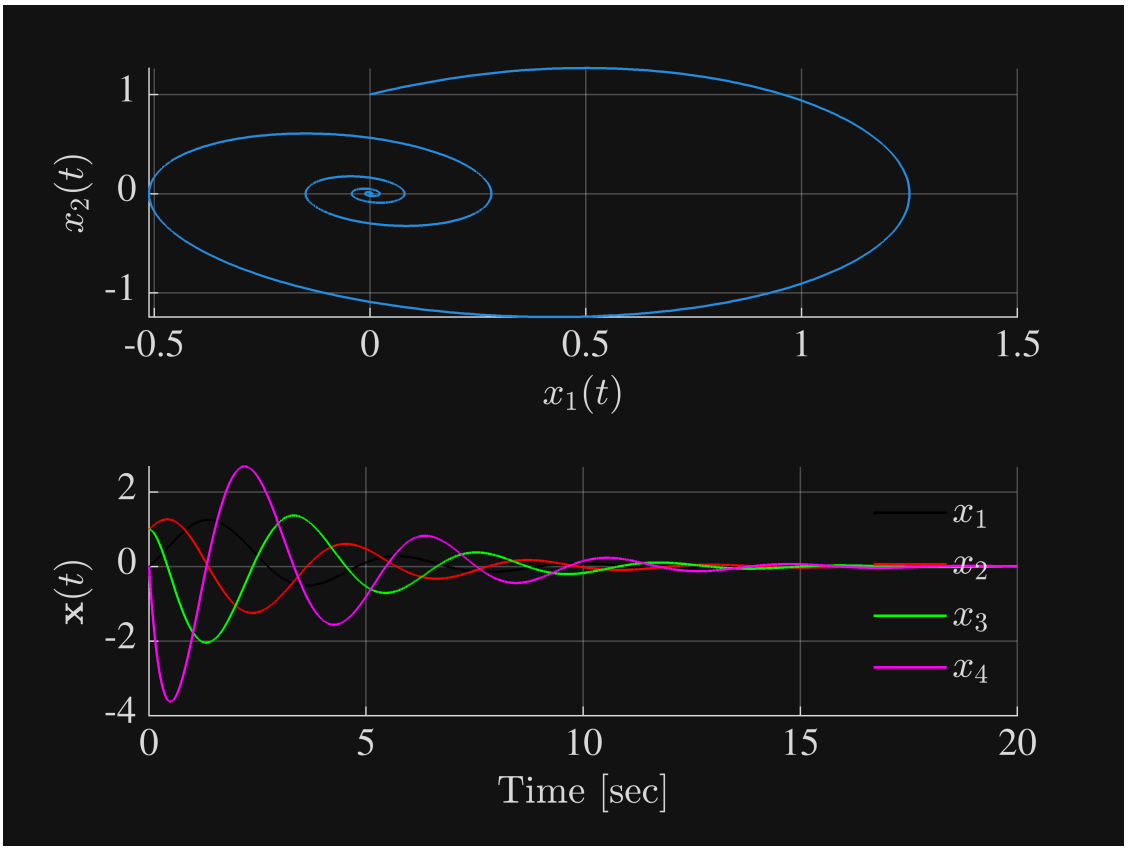
$$P(\mathbf{A}) = \lambda^4 + a_3\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0$$

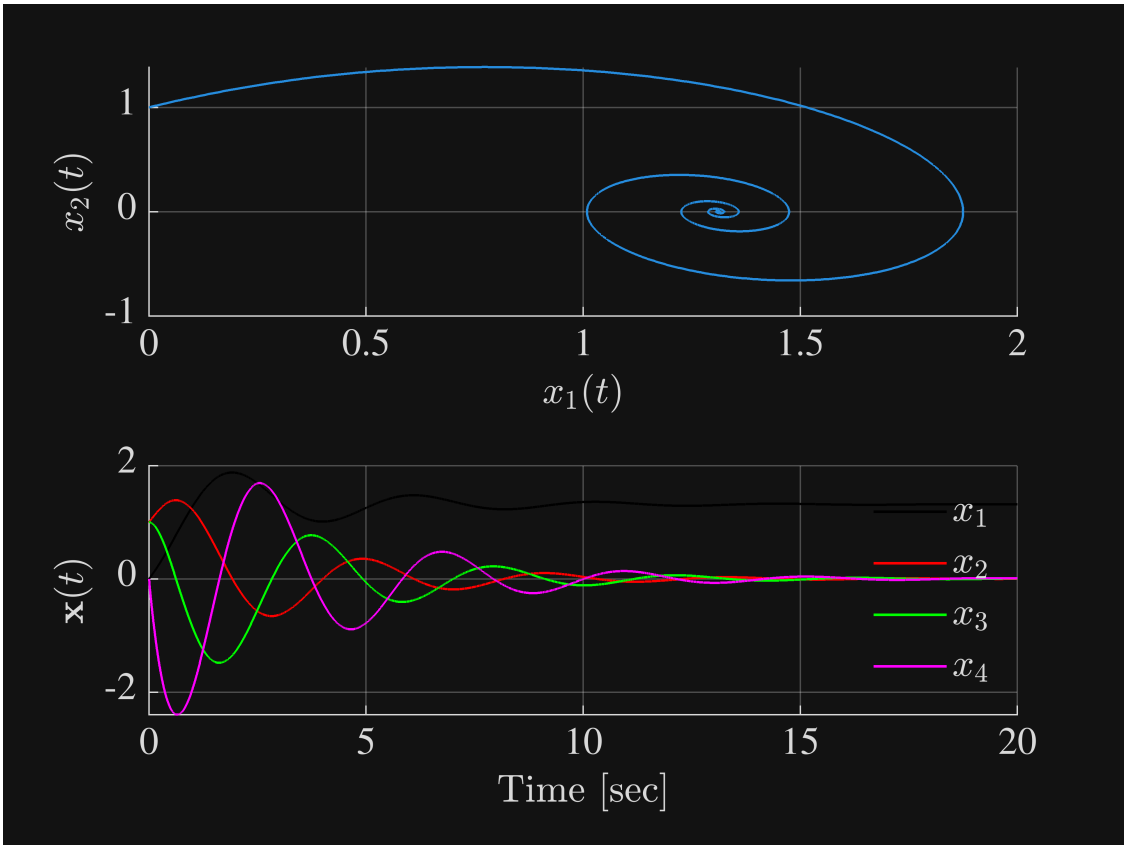
with the coefficients

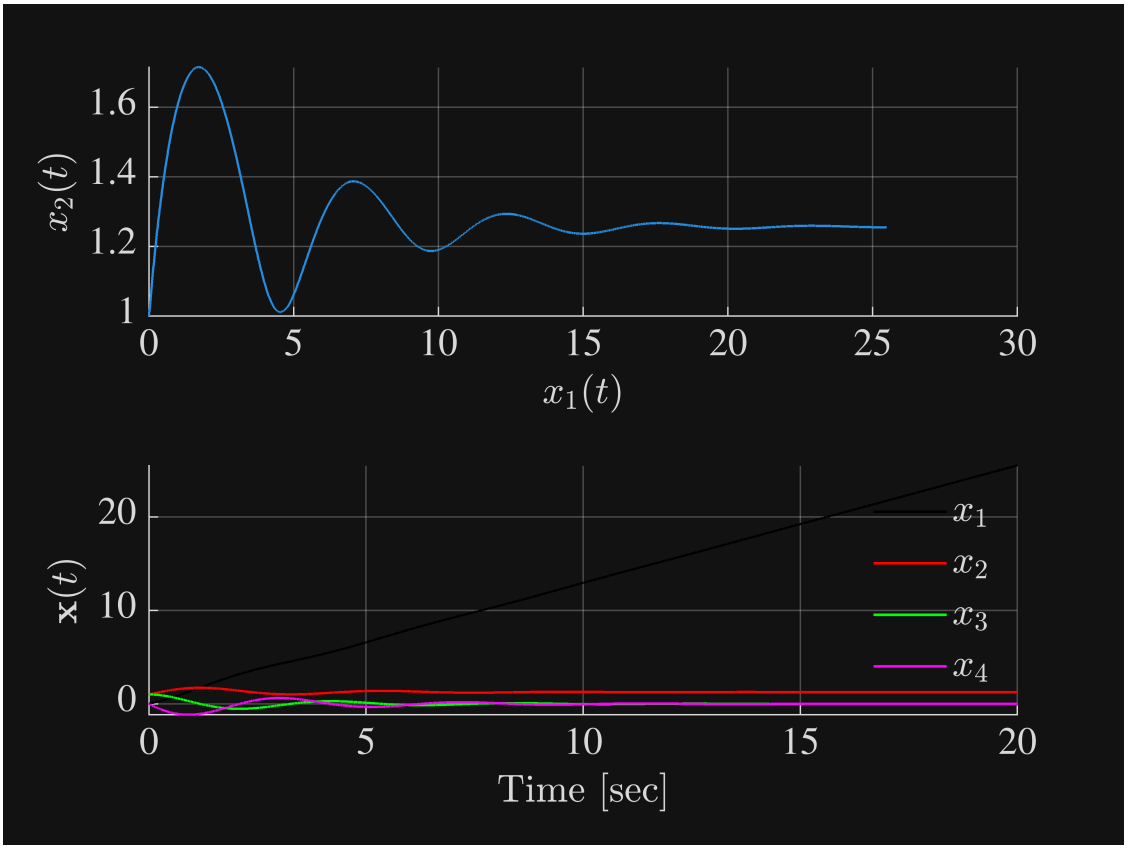
	a_3	a_2	a_1	a_0
System a	4.1	7.44	9.99	7.02
System b	2.6	3.54	4.68	0
System c	0.6	2.34	0	0

```
clear all
close all
clc

% Example 3.9 (Textbook)

a0 = [7.02 0 0]';
a1 = [9.99 4.68 0]';
a2 = [7.44 3.54 2.34]';
a3 = [4.1 2.6 0.6]';

% Input, Output and Feedofrward matrices
B = [0 0 0 1]';
C = [1 0 0 0];
D = 0;

% time, input and initial conditions vector
t = (0:0.01:20)';
u = zeros(size(t,1),1);
x0 = [0 1 1 0]';

for ii = 1:length(a0)
    A = [0 1 0 0; ...
         0 0 1 0; ...
         0 0 0 1; ...
         -a0(ii) -a1(ii) -a2(ii) -a3(ii)];

    lambda_A(:,ii) = eig(A);

    sys = ss(A,B,C,D);
    [y,t,x] = lsim(sys,u,t,x0);

    figure, h1 = subplot(2,1,1); set(h1,'FontName','times','FontSize',16)
    hold on, grid on
    plot(x(:,1),x(:,2),'LineWidth',1)
    xlabel('$$x_1(t)$
$','FontName','times','FontSize',16,'Interpreter','latex')
    ylabel('$$x_2(t)$
$','FontName','times','FontSize',16,'Interpreter','latex')
    h2 = subplot(2,1,2); set(h2,'FontName','times','FontSize',16)
    hold on, grid on
    plot(t,x(:,1),'k',t,x(:,2),'r',t,x(:,3),'g',t,x(:,4),'m','LineWidth',1)
    xlabel('Time
[sec]','FontName','times','FontSize',16,'Interpreter','latex')
    ylabel('$$\mathbf{x}(t)$
$','FontName','times','FontSize',16,'Interpreter','latex')
    l1 = legend('$$x_1$$','$$x_2$$','$$x_3$$','$$x_4$$');

set(l1,'FontName','times','FontSize',16,'Interpreter','latex','box','off');
```

```
end
```


1
0
-1
$x_2(t)$
-0.5
0
0.5
1
1.5
$x_1(t)$
2
0
-2
-4
$\mathbf{x}(t)$
x_1
x_2
x_3
x_4
0
5
10
15
20
Time [sec]


1
0
-1
$x_2(t)$
0
0.5
1
1.5
2
$x_1(t)$
2
0
-2
$\mathbf{x}(t)$
x_1
x_2
x_3
x_4
0
5
10
15
20
Time [sec]


1.6
1.4
1.2
1
$x_2(t)$
0
5
10
15
20
25
30
$x_1(t)$
20
10
0
$\mathbf{x}(t)$
x_1
x_2
x_3
x_4
0
5
10
15
20
Time [sec]

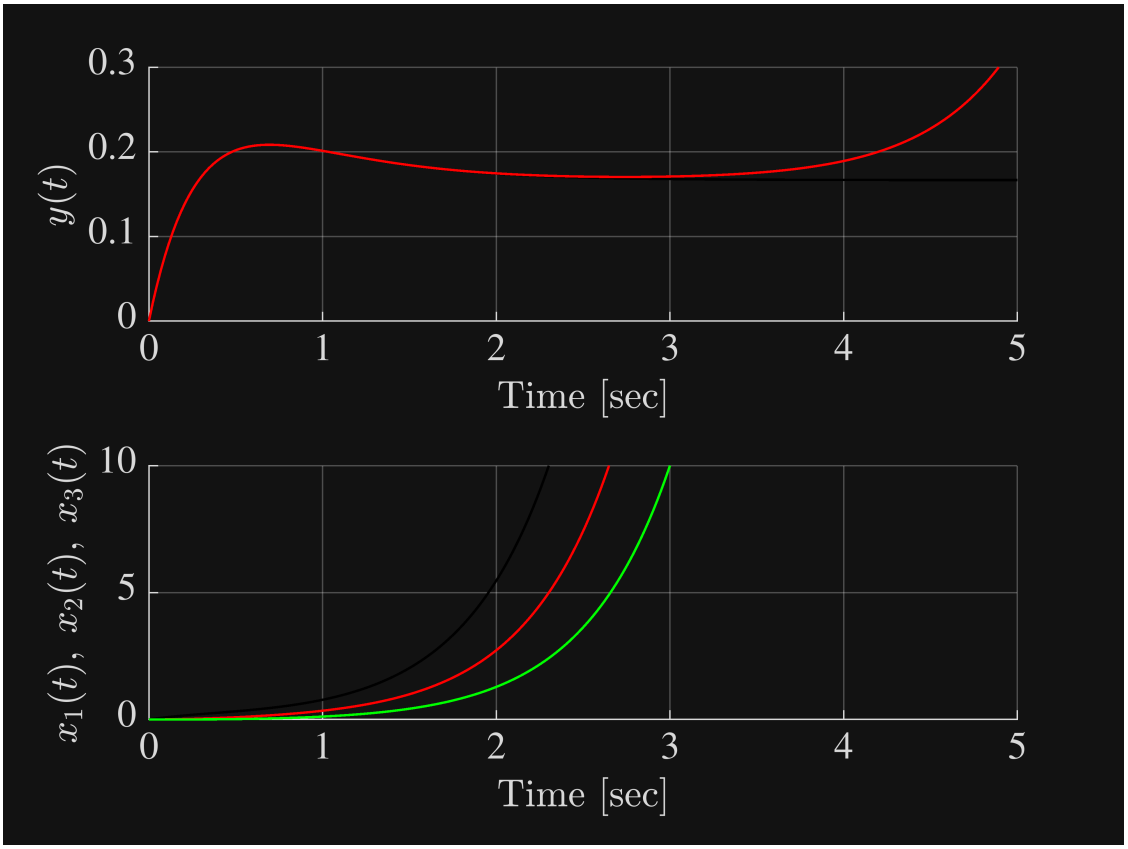
Example 3.12. **Externally Stable and Internally Unstable System**

Consider the SISO continuous time LTI system,

$$\dot{\mathbf{x}} = \begin{bmatrix} -3 & 4 & 12 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u,$$

$$y = [1 \quad -1 \quad -2]\mathbf{x}.$$

```
clear all
close all
clc

% Example 3.12 (Textbook)

A = [-3 4 12;...
      1 0 0;...
      0 1 0];

B = [1 0 0]';
C = [1 -1 -2];
D = 0;

% Internal stability
lambda_A = eig(A);

% External stability
[num den] = ss2tf(A,B,C,D);
G = tf(num,den);
[z p k] = tf2zp(num,den);

% Simulate responses
sys = ss(A,B,C,D);
A(1,2) = 4.001;
lambda_Aus = eig(A);
sys_us = ss(A,B,C,D);
[y,t,x] = step(sys);
[y_us,t_us] = step(sys_us);
figure, h1 = subplot(2,1,1); set(h1,'FontName','times','FontSize',16)
hold on, grid on
plot(t,y,'k',t_us,y_us,'r','LineWidth',1)
ax1 = axis;
axis([ax1(1) 5 ax1(3) 0.3]);
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
ylabel('$$y(t)$$','FontName','times','FontSize',16,'Interpreter','latex')
h2 = subplot(2,1,2); set(h2,'FontName','times','FontSize',16)
hold on, grid on
plot(t,x(:,1),'k',t,x(:,2),'r',t,x(:,3),'g','LineWidth',1)
ax2 = axis;
axis([ax2(1) 5 ax2(3) 10]);
xlabel('Time [sec]','FontName','times','FontSize',16,'Interpreter','latex')
ylabel('$$x_1(t), \; x_2(t), \; x_3(t)$
$','FontName','times','FontSize',16,'Interpreter','latex')
```


0.3
0.2
0.1
0
$y(t)$
0
1
2
3
4
5
Time [sec]
10
5
0
$x_1(t),\ x_2(t),\ x_3(t)$
0
1
2
3
4
5
Time [sec]

Control Design for a Mass-Spring-Damper System

System Description

A mass-spring-damper system is described by a second order differential equation with constant coefficients, as

$$m\ddot{x} + b\dot{x} + kx = F$$

where m, b, and k are the mass, friction coefficient and the spring constant. Let $\mathbf{x} = [x, \dot{x}]^{\mathrm{T}}$ be the state vector including the position and the linear velocity of the mass, then the system in state space form is described by

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{1}{m}F$$

Defining $\omega_n = \sqrt{\frac{k}{m}}$ and $\zeta = \frac{b}{2\sqrt{km}}$ the system reads

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\omega_n^2 x_1 - 2\zeta\omega_n x_2 + \alpha F$$

By setting $\omega_n = 1,\ \zeta = 0.2,\ \alpha = 1$ implement the state space model in Matlab

```
wn = 1;
zeta = 0.2;
alpha = 1;

A = [0 1;-wn^2 -2*zeta*wn];
B = [0 alpha]';
C = eye(2);
D = zeros(2,1);
msd = ss(A,B,C,D);
```

Plot the the two transfer functions from force to position and from force to velocity

```
[numG,denG] = ss2tf(msd.a,msd.b,msd.c,msd.d);
G_fp = tf(numG(1,:),denG)
```

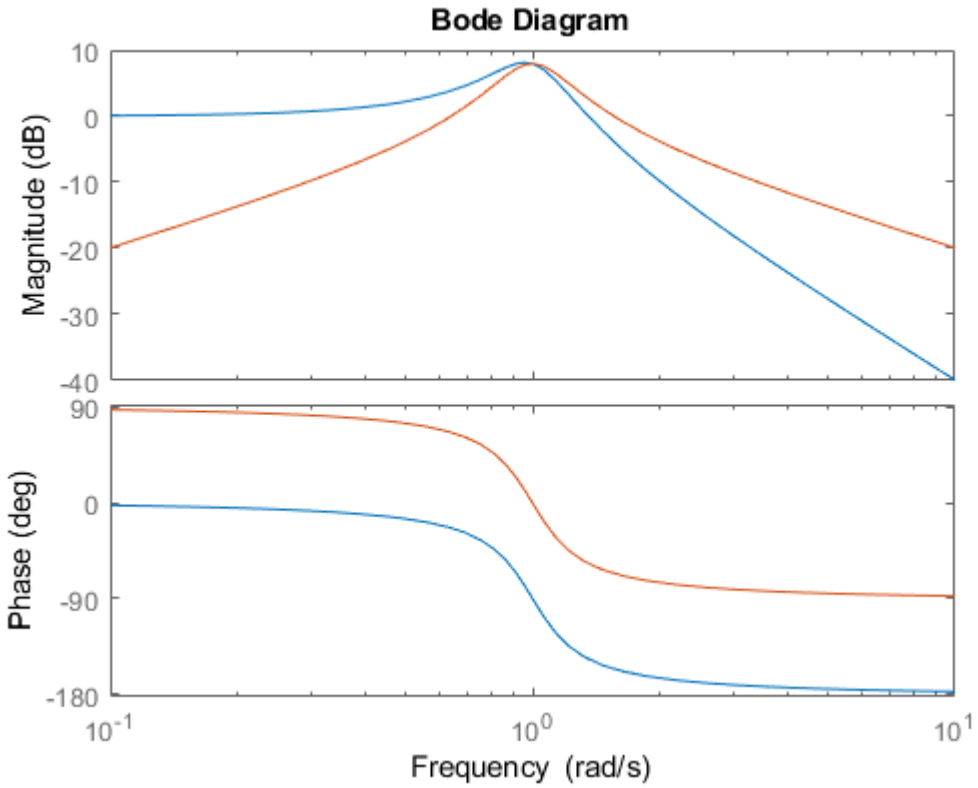
```
G_fp =

        1
  ---------------
  s^2 + 0.4 s + 1

Continuous-time transfer function.
```

```
G_fv = tf(numG(2,:),denG)
```

```
G_fv =

        s
```

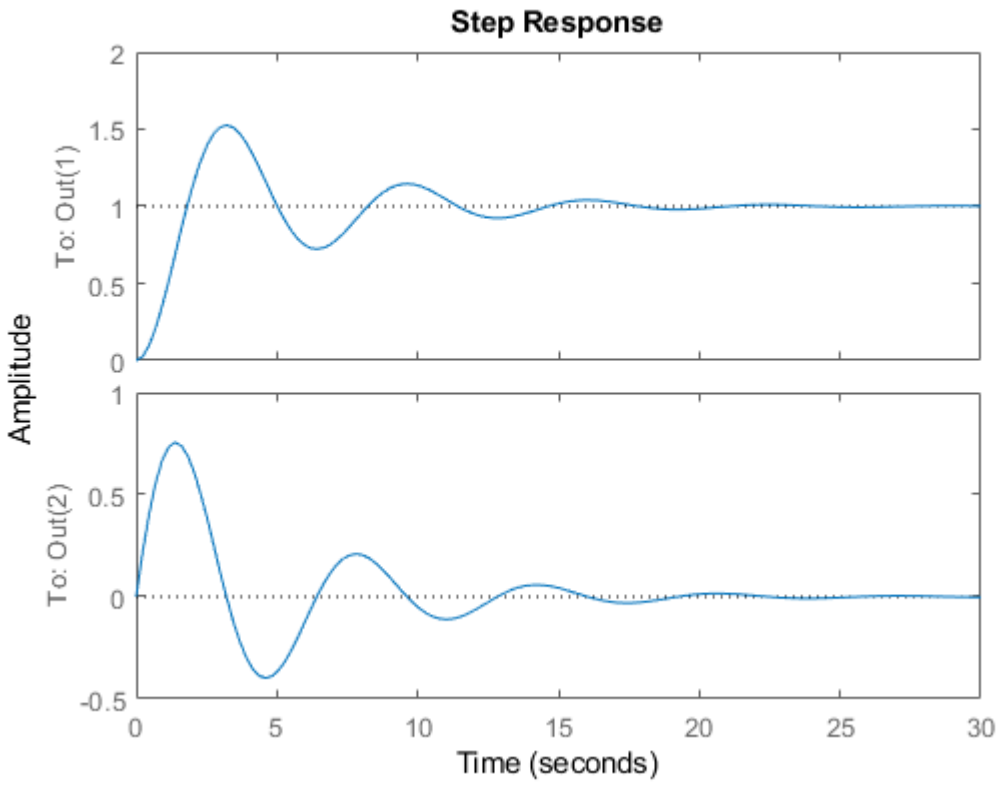
```
  ---------------
  s^2 + 0.4 s + 1

Continuous-time transfer function.
```

```
figure, bode(G_fp,G_fv)
```



Plot also the step response for a unit step force input

```
figure, step(msd)
```



Control System Design

Three control objectives will be pursued:

1. Reduce the oscillatory behavior of the response without increasing the speed of the response

The oscillatory behavior of the response is linked to the damping ratio ζ, whereas the speed of the response is linked to the natural frequency ω_n. Therefore to meet the control objective we need to increase the value of the damping ratio to $\zeta_{\text{new}} > \zeta$ and leave unchanged the natural frequency.

The following state feedback controller is proposed

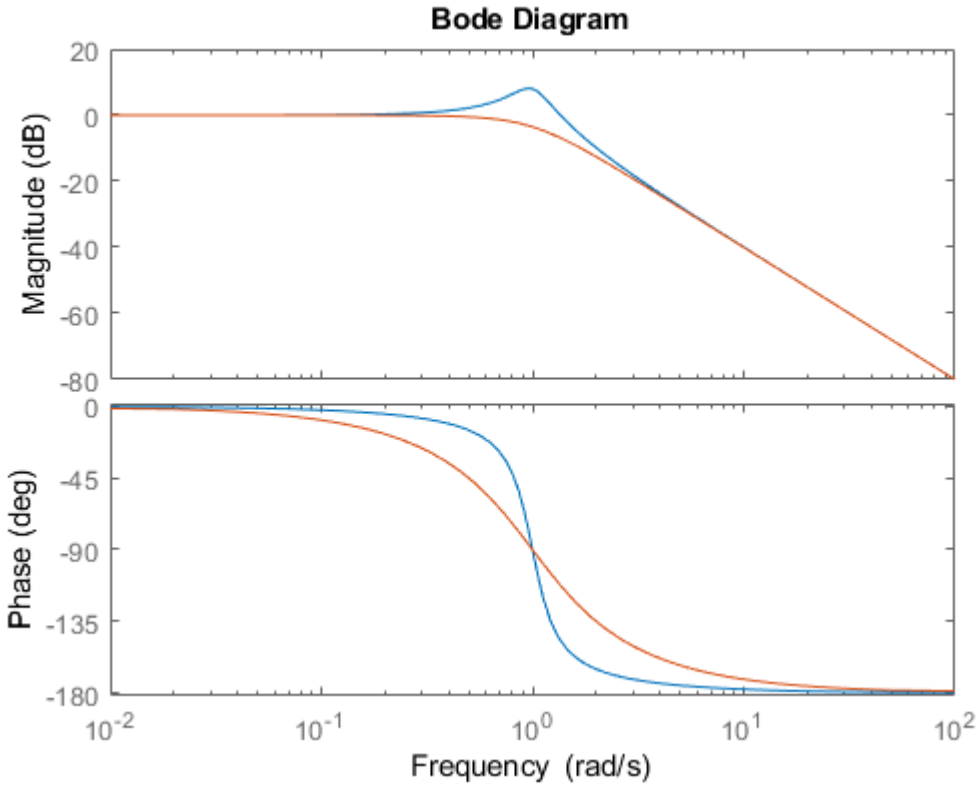
$$F = -\frac{2}{\alpha}(\zeta_{\text{new}} - \zeta)\omega_n + r = -[k_1 \; k_2][x_1 \; x_2]^{\text{T}} + r$$

where $k_1 = 0$ and $k_2 = 2(\zeta_{\text{new}} - \zeta)\omega_n/\alpha$.

```
zeta_new = 0.75;
k1 = 0;
k2 = 2/alpha*(zeta_new - zeta)*wn;
A_cl = [0 1;A(2,1)-k1 A(2,2)-k2];
msd_cl = ss(A_cl,B,C,D);
[numG_cl,denG_cl] = ss2tf(msd_cl.a,msd_cl.b,msd_cl.c,msd_cl.d);
G_fp_cl = tf(numG_cl(1,:),denG_cl)
```

```
G_fp_cl =

        1
  --------------
  s^2 + 1.5 s + 1

Continuous-time transfer function.
```

```
figure, bode(G_fp,G_fp_cl)
```




```
damp(A)
```

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

 -2.00e-01 + 9.80e-01i     2.00e-01       1.00e+00          5.00e+00
 -2.00e-01 - 9.80e-01i     2.00e-01       1.00e+00          5.00e+00
```

```
damp(A_cl)
```

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

 -7.50e-01 + 6.61e-01i     7.50e-01       1.00e+00          1.33e+00
 -7.50e-01 - 6.61e-01i     7.50e-01       1.00e+00          1.33e+00
```

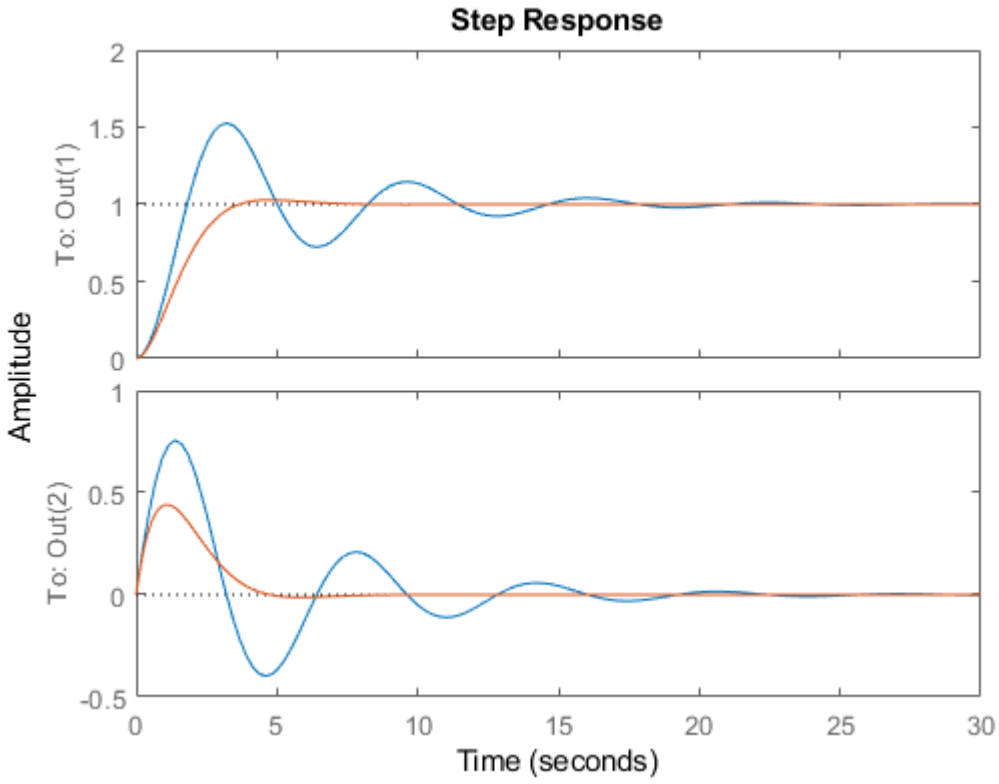
```
bandwidth(G_fp)
```

```
ans = 1.5090
```

```
bandwidth(G_fp_cl)
```

```
ans = 0.9383
```

```
figure, step(msd,msd_cl)
```



2. Increase the speed of the response without changing the oscillatory behavior of the response

To meet the control objective we need to increase the value of the natural frequency to $\omega_{\text{new}} > \omega$ and leave the damping ratio unchanged.

The following state feedback controller is proposed

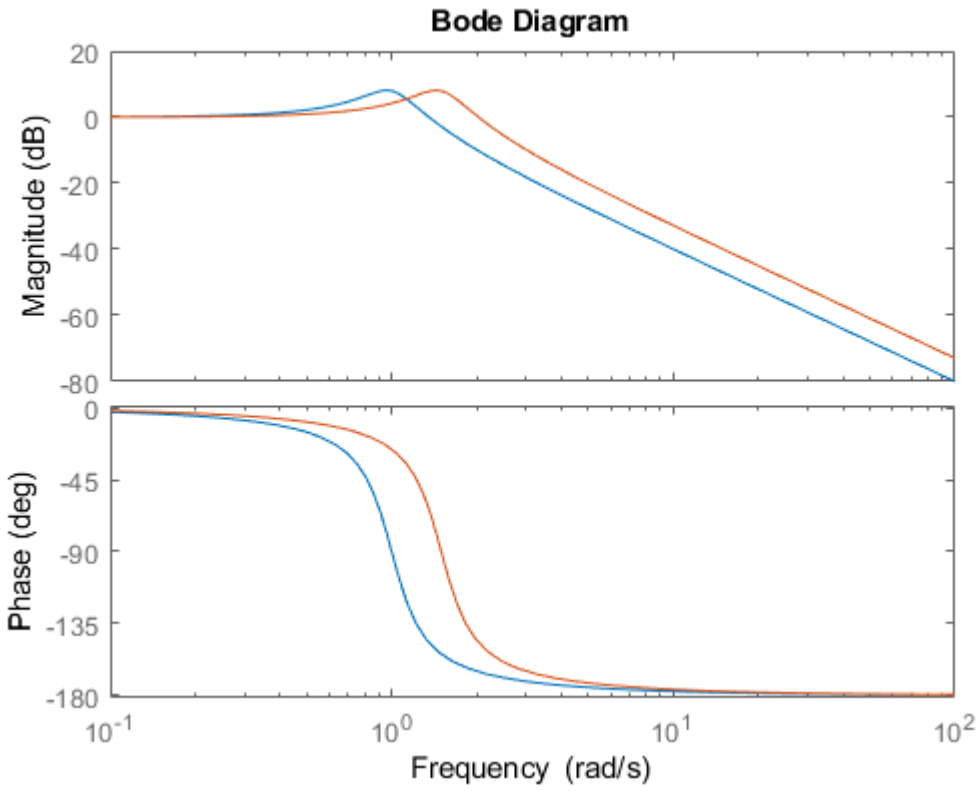
$$F = -\frac{1}{\alpha}((\omega_{n,\text{new}}^2 - \omega_n^2) + 2\zeta(\omega_{n,\text{new}} - \omega_n)) + r = -[k_1\ k_2][x_1\ x_2]^{\text{T}} + r$$

where $k_1 = (\omega_{n,\text{new}}^2 - \omega_n^2)/\alpha$ and $k_2 = 2\zeta(\omega_{n,\text{new}} - \omega_n)/\alpha$.

```
wn_new = 1.5;
k1 = (wn_new^2 - wn^2)/alpha;
k2 = 2/alpha*zeta*(wn_new - wn);
A_cl = [0 1;A(2,1)-k1 A(2,2)-k2];
msd_cl = ss(A_cl,B*wn_new^2,C,D);
[numG_cl,denG_cl] = ss2tf(msd_cl.a,msd_cl.b,msd_cl.c,msd_cl.d);
G_fp_cl = tf(numG_cl(1,:),denG_cl)
```

```
G_fp_cl =

        2.25
  ------------------
  s^2 + 0.6 s + 2.25

Continuous-time transfer function.
```

```
figure, bode(G_fp,G_fp_cl)
```




```
damp(A)
```

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

 -2.00e-01 + 9.80e-01i     2.00e-01       1.00e+00          5.00e+00
 -2.00e-01 - 9.80e-01i     2.00e-01       1.00e+00          5.00e+00
```

```
damp(A_cl)
```

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

 -3.00e-01 + 1.47e+00i     2.00e-01       1.50e+00          3.33e+00
 -3.00e-01 - 1.47e+00i     2.00e-01       1.50e+00          3.33e+00
```

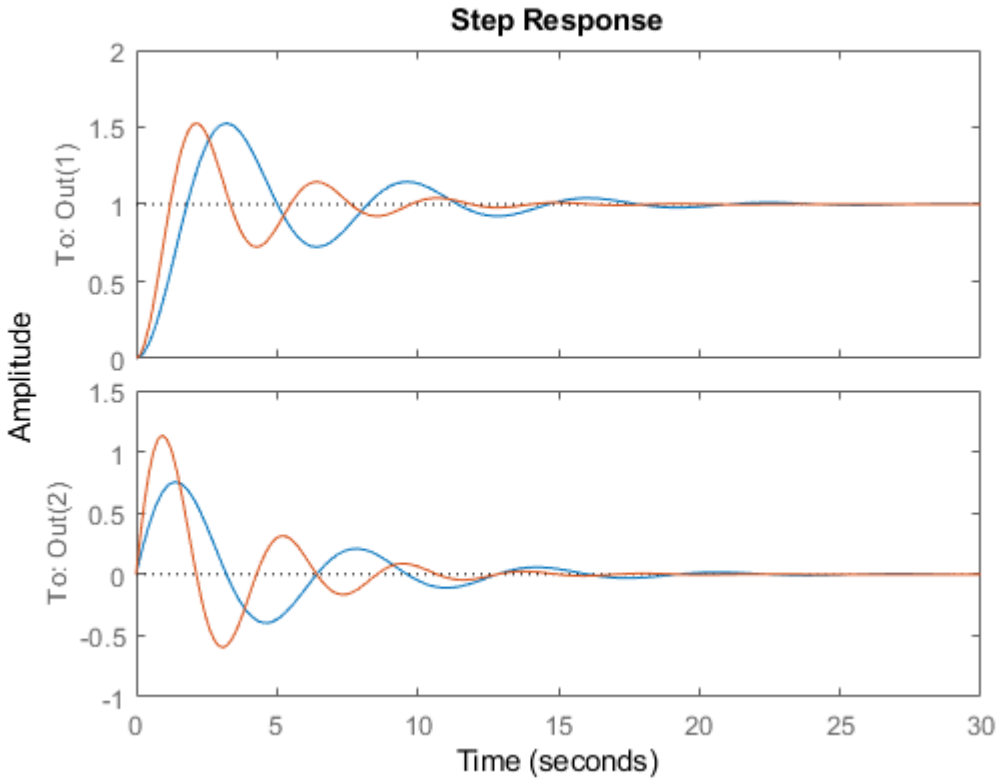
```
bandwidth(G_fp)
```

```
ans = 1.5090
```

```
bandwidth(G_fp_cl)
```

```
ans = 2.2635
```

```
figure, step(msd,msd_cl)
```



3. Reduce the oscillatory behavior of the response and increase the speed of the response

Example 4.16. **Observer Control of a Marginally stable System**

Many mechanical and electrical systems (or combinations of them) can be described as un- or underdamped harmonic oscillators. For example hydraulic/mechanical motors, tuned circuits (electrical or mechanical/electrical), etc. The state and output equations of such a system are easy to write down. Choosing the position and the velocity as the states, the equations will be

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u,$$

$$y = [1\ 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

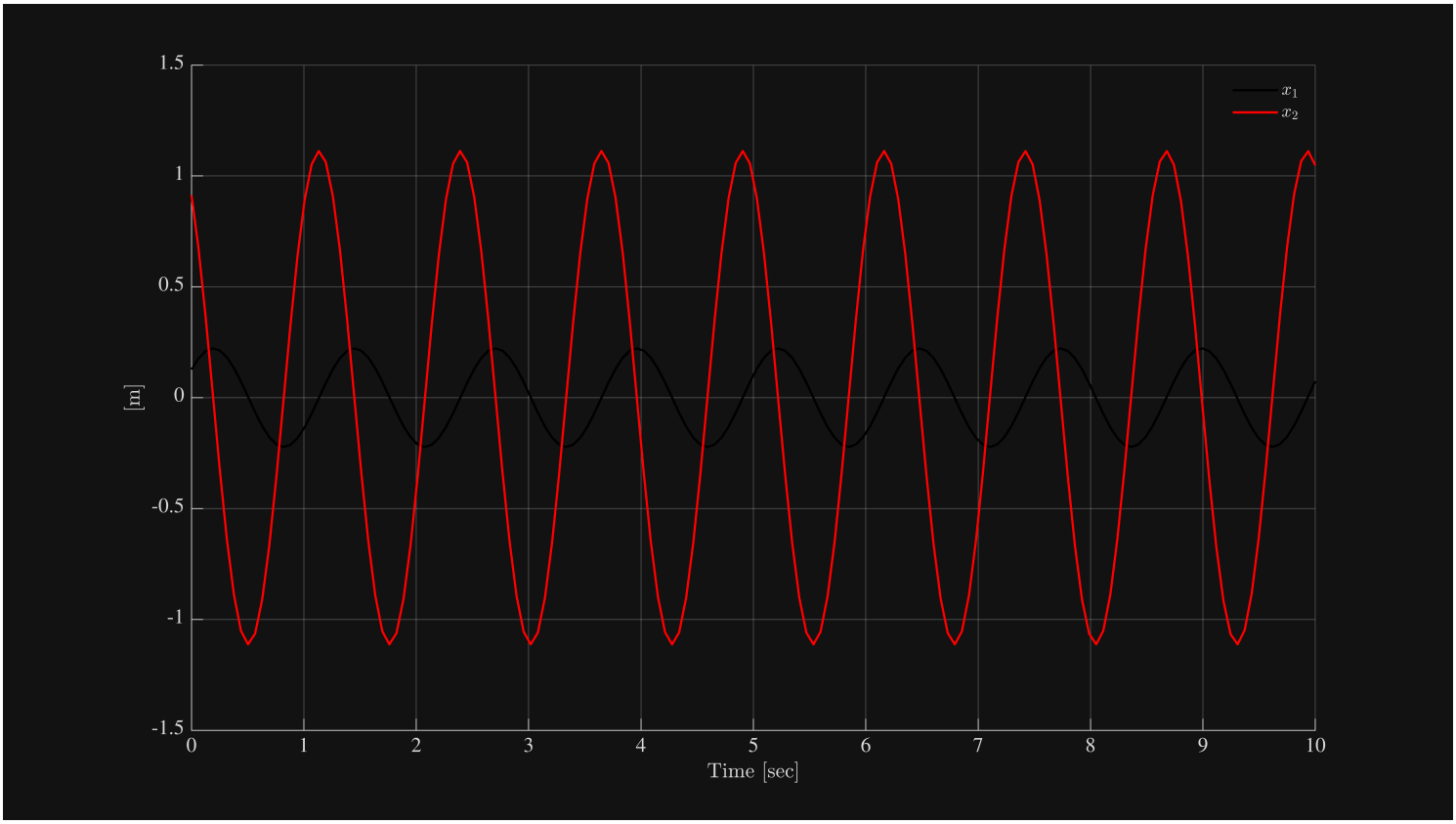
Example 4.16

Observer design for marinally stable and unstable system

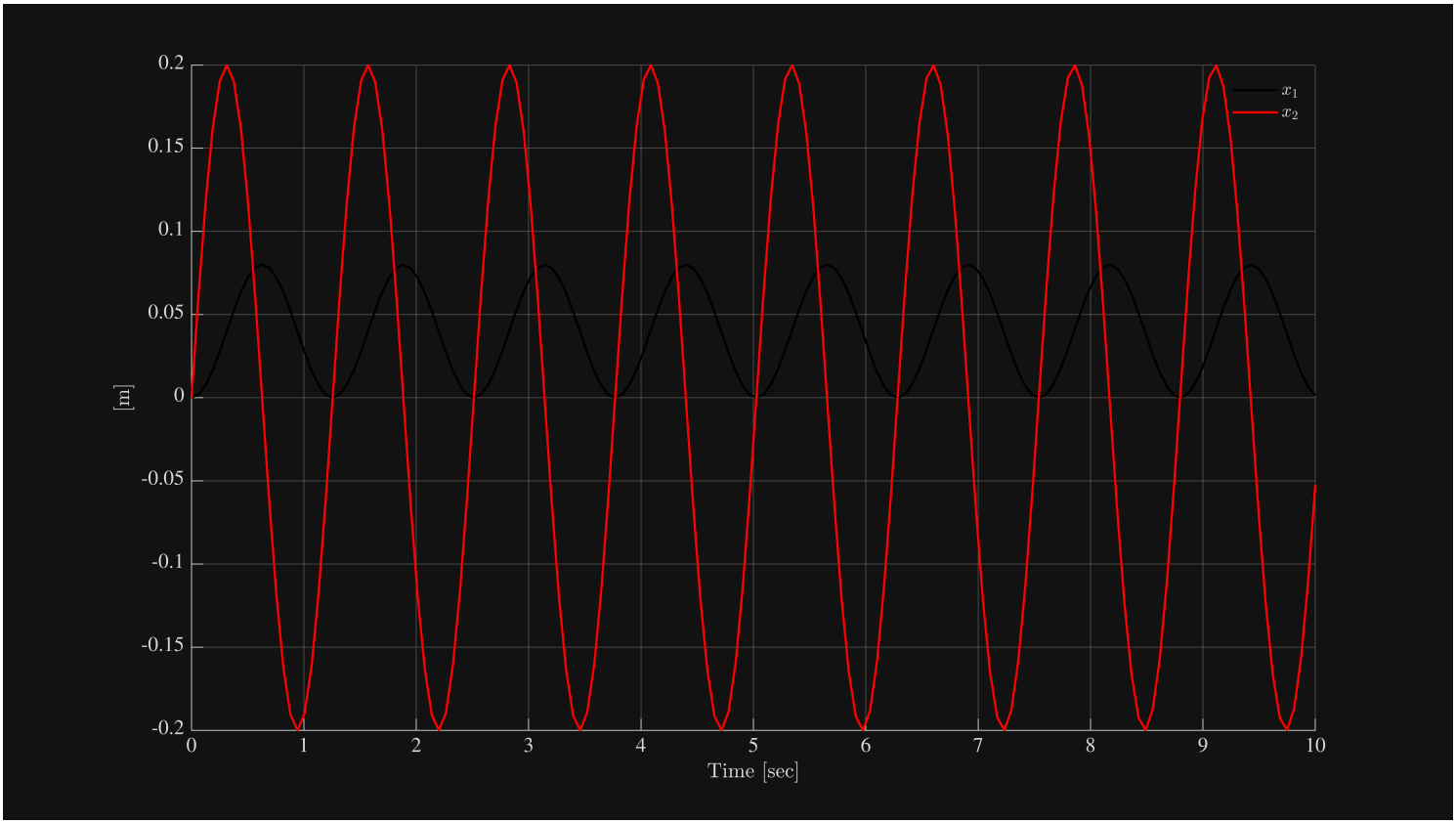
```
clear all
clc
close all
```

Marginally stable system

```
wn = 5; % [rad/s]
A = [0 1;-wn^2 0];
B = [0 1]';
C = [1 0];
x0 = [-1.5 0.25]';

sys_ol = ss(A,B,C,0);
x0 = rand(2,1);

% zero input resposnse
[~,t,x] = initial(sys_ol,x0,10);
figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
hf1 = axes; set(hf1,'FontName','times','FontSize',14)
hold on, grid on
plot(t,x(:,1),'k',t,x(:,2),'r','LineWidth',1.5)
ylabel('[m]','FontName','times','FontSize',14,'Interpreter','latex')
leg1 = legend('$x_1$','$x_2$','Location','NorthEast');
set(leg1,'FontName','times','FontSize',12,'Interpreter','latex','box','off')
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
```




```
% zero state resposnse
[~,t,x] = step(sys_ol,10);
figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
hf1 = axes; set(hf1,'FontName','times','FontSize',14)
hold on, grid on
plot(t,x(:,1),'k',t,x(:,2),'r','LineWidth',1.5)
ylabel('[m]','FontName','times','FontSize',14,'Interpreter','latex')
leg1 = legend('$x_1$','$x_2$','Location','NorthEast');
set(leg1,'FontName','times','FontSize',12,'Interpreter','latex','box','off')
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
```




```
lambda_A = eig(A);
```

Oberserver design

```
wn_obs = 5*wn;
zeta_obs = 0.7;
a = -wn_obs*zeta_obs;
b = sqrt(wn_obs^2-a^2);
lambda_obs_des = [a+b*1i a-b*1i]';

L = acker(A',C',lambda_obs_des)';
lambda_obs = eig(A-L*C);
```

Simulate the observer with the system

```
xhat0 = x0;
sim('Example_4_16_model')
```

```
xhat0 = 5*x0;
sim('Example_4_16_model')
```

Unstable system

```
clear all
```

```
wn_sys = 2;
zeta_sys = -0.3;
a = -zeta_sys*wn_sys;
b = sqrt(wn_sys^2-a^2);

A = [a -b;b a];
B = [0 1]';
C = [1 0];
x0 = [-1.5 0.25]';

lambda_A = eig(A);

wn_obs = 5*wn_sys;
zeta_obs = 0.7;
ao = -wn_obs*zeta_obs;
bo = sqrt(wn_obs^2-a^2);
lambda_obs_des = [ao+bo*1i ao-bo*1i]';

L = acker(A',C',lambda_obs_des)';
lambda_obs = eig(A-L*C);
```

Simulate the observer with the system

```
xhat0 = x0;
sim('Example_4_16_model')
```

```
xhat0 = 5*x0;
sim('Example_4_16_model')
```

TwoTankSystem

```
clear all
close all
clc
```

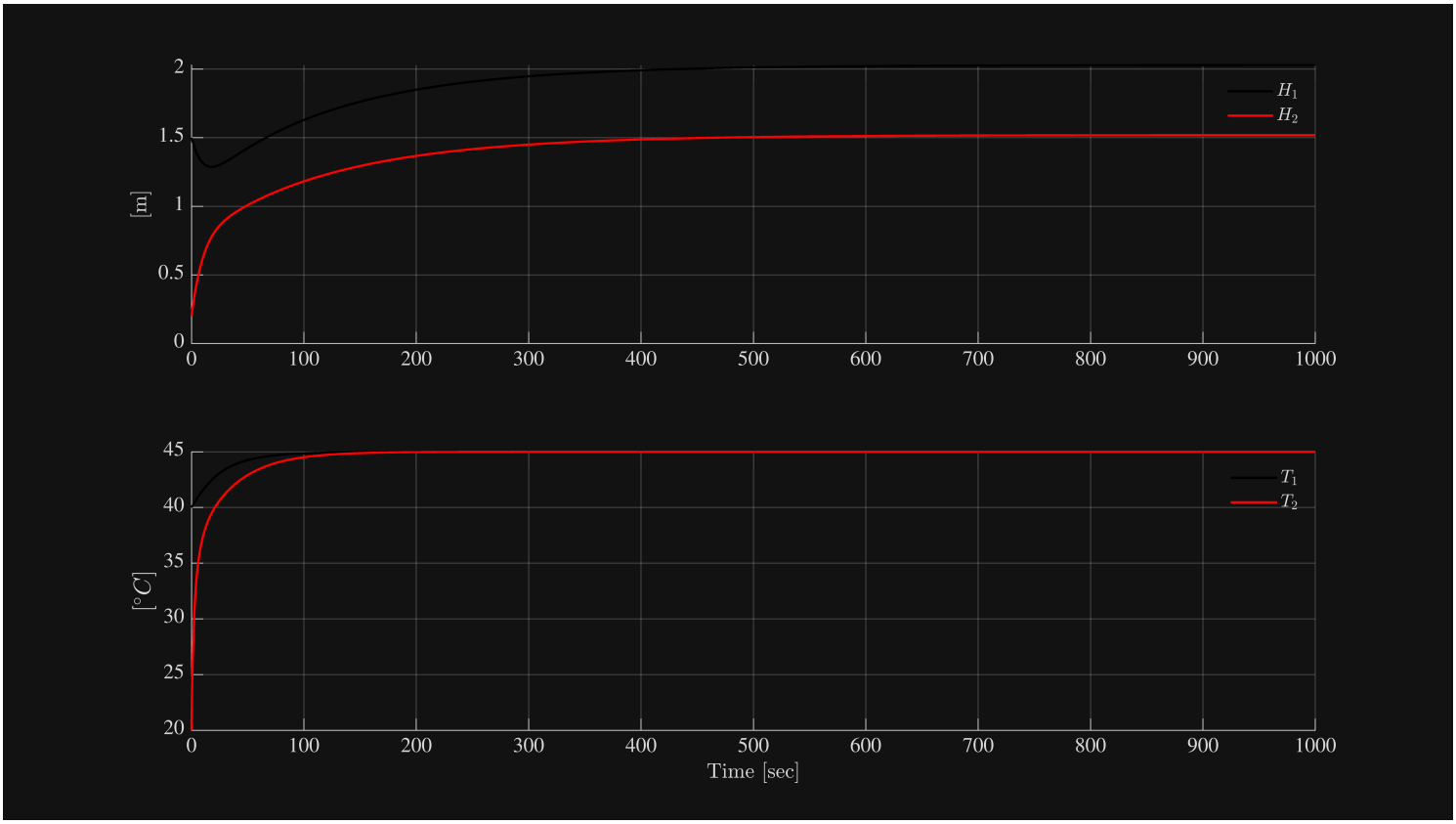
Simulation paramters

```
SIM_TIME = 1000; % simulation time [sec]
STEP_SIZE = 0.05; % fundamental sampling time for running simulations [sec]
```

Model parameters

```
At = 0.785; % [m^2]
Dv = 2.66; % [m^(1/2)/s]
C0 = 0.056; % [[m^(5/2)/s]]
ka = 0.004; % [m^3/(V*s)]
kh = 2; % [V/m]
kt = 0.1; % [V/C]

% Input and disturbance working point
uw0 = 5; % hot water valve [V]
u10 = uw0;
uc0 = 5; % cold water valve [V]
u20 = uc0;
Av0 = 0.0122; % area of outlet valve [m^2]
v10 = Av0;
Tw0 = 60; % temperature of the hot water [C]
v20 = Tw0;
Tc0 = 30; % temperature of the cold water [C]
v30 = Tc0;

% Simulate the system to evaluate the stationary state in connection with
% the chosen values of input and disturbances
x0 = [1.5,0.2,40,20]'; % initial conditions for the state variables x0 =
[H1,H2,T1,T2]'
sim('TwoTankSystem_SteadyState',SIM_TIME,[],[])
x0 = xFinal'; % simulated steady state condition

% Plot of state responses
t = logsout.getElement('H1').Values.Time;
H1 = logsout.getElement('H1').Values.Data;
H2 = logsout.getElement('H2').Values.Data;
Tm1 = logsout.getElement('T1').Values.Data;
Tm2 = logsout.getElement('T2').Values.Data;

figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
hf1 = subplot(2,1,1); set(hf1,'FontName','times','FontSize',14)
```

```
hold on, grid on
plot(t,H1,'k',t,H2,'r','LineWidth',1.5)
ylabel('[m]','FontName','times','FontSize',14,'Interpreter','latex')
leg1 = legend('$H_1$','$H_2$','Location','NorthEast');
set(leg1,'FontName','times','FontSize',12,'Interpreter','latex','box','off')
hf2 = subplot(2,1,2); set(hf2,'FontName','times','FontSize',14)
hold on, grid on
plot(t,Tm1,'k',t,Tm2,'r','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
ylabel('$[^\circ C]$','FontName','times','FontSize',16,'Interpreter','latex')
leg2 = legend('$T_1$','$T_2$','Location','NorthEast');
set(leg2,'FontName','times','FontSize',12,'Interpreter','latex','box','off')
```



Numerically linearize the nonlinear model around the stationary state

```
% Numerical linearization of a nonlinear system implemented in Simulink can
% be achieved through the command LINMOD (read help)
u0 = [u10 u20 v10 v20 v30]';
[A,B,C,D] = linmod('TwoTankSystem_Linearize',x0,u0);
Bv = B(:,3:5);
B = B(:,1:2);
Dvv = D(:,3:5);
D = D(:,1:2);
x0_lin = zeros(size(A,1),1);
```

```
% Present the linearized model
disp('Numerically linearized model of the Two Tank System')
```

```
Numerically linearized model of the Two Tank System
```

```
disp('matrix A')
```

```
matrix A
```

```
disp(A)
```

```
   -0.0499    0.0499         0         0
    0.0499   -0.0667         0         0
   -0.0000         0   -0.0251         0
    0.0000   -0.0000    0.0335   -0.0335
```

```
disp('matrix B')
```

```
matrix B
```

```
disp(B)
```

```
    0.0051    0.0051
         0         0
    0.0377   -0.0377
         0         0
```

```
disp('matrix Bv')
```

```
matrix Bv
```

```
disp(Bv)
```

```
         0         0         0
   -4.1762         0         0
         0    0.0126    0.0126
         0         0         0
```

```
disp('matrix C')
```

```
matrix C
```

```
disp(C)
```

```
         0    2.0000         0         0
         0         0         0    0.1000
```

```
disp('matrix D')
```

```
matrix D
```

```
disp(D)
```

```
     0     0
     0     0
```

```
disp('matrix Dv')
```

```
matrix Dv
```

```
disp(Dvv)
```

```
     0     0     0
     0     0     0
```

Discretize the system

```
% Choose the sampling time
lambda = eig(A); % eigenvalues
tau = 1./abs(lambda); % time constants

Ts = 1; % sampling time [sec] chosen approximately 10 times smaller than the
smallest time constant
[F,G] = c2d(A,B,Ts);
[~,Gv] = c2d(A,Bv,Ts);

% Present the discretized linear model
disp('Discretized linear model of the Two Tank System')
```

```
Discretized linear model of the Two Tank System
```

```
disp('matrix F')
```

```
matrix F
```

```
disp(F)
```

```
    0.9525    0.0471         0         0
    0.0471    0.9366         0         0
   -0.0000   -0.0000    0.9752         0
    0.0000   -0.0000    0.0326    0.9670
```

```
disp('matrix G')
```

```
matrix G
```

```
disp(G)
```

```
    0.0050    0.0050
    0.0001    0.0001
    0.0372   -0.0372
    0.0006   -0.0006
```

```
disp('matrix Gv')
```

```
matrix Gv
```

```
disp(Gv)
```

```
   -0.1003         0         0
   -4.0416         0         0
    0.0000    0.0124    0.0124
    0.0000    0.0002    0.0002
```

```
disp('matrix C')
```

```
matrix C
```

```
disp(C)
```

```
         0    2.0000         0         0
         0         0         0    0.1000
```

```
disp('matrix D')
```

```
matrix D
```

```
disp(D)
```

```
     0     0
     0     0
```

```
disp('matrix Dv')
```

```
matrix Dv
```

```
disp(Dv)
```

```
    2.6600
```

```
% Compare the discretized linear model with the continuous time nonlinear
% model
u0 = [u10 u20 v10 v20 v30]';
v0 = u0(3:5,1);
u0 = u0(1:2,1);
y0 = C*x0;

% Step changes in inputs and disturbances
uw1 = 1.1*uw0;
Tw = 100;
uc1 = 0.9*uc0;
Tc = 400;
Av1 = 1.05*Av0;
Tv = 700;
Tw1 = 1.1*Tw0;
Te1 = 1100;
Tc1 = 1.2*Tc0;
Te2 = 1500;
```

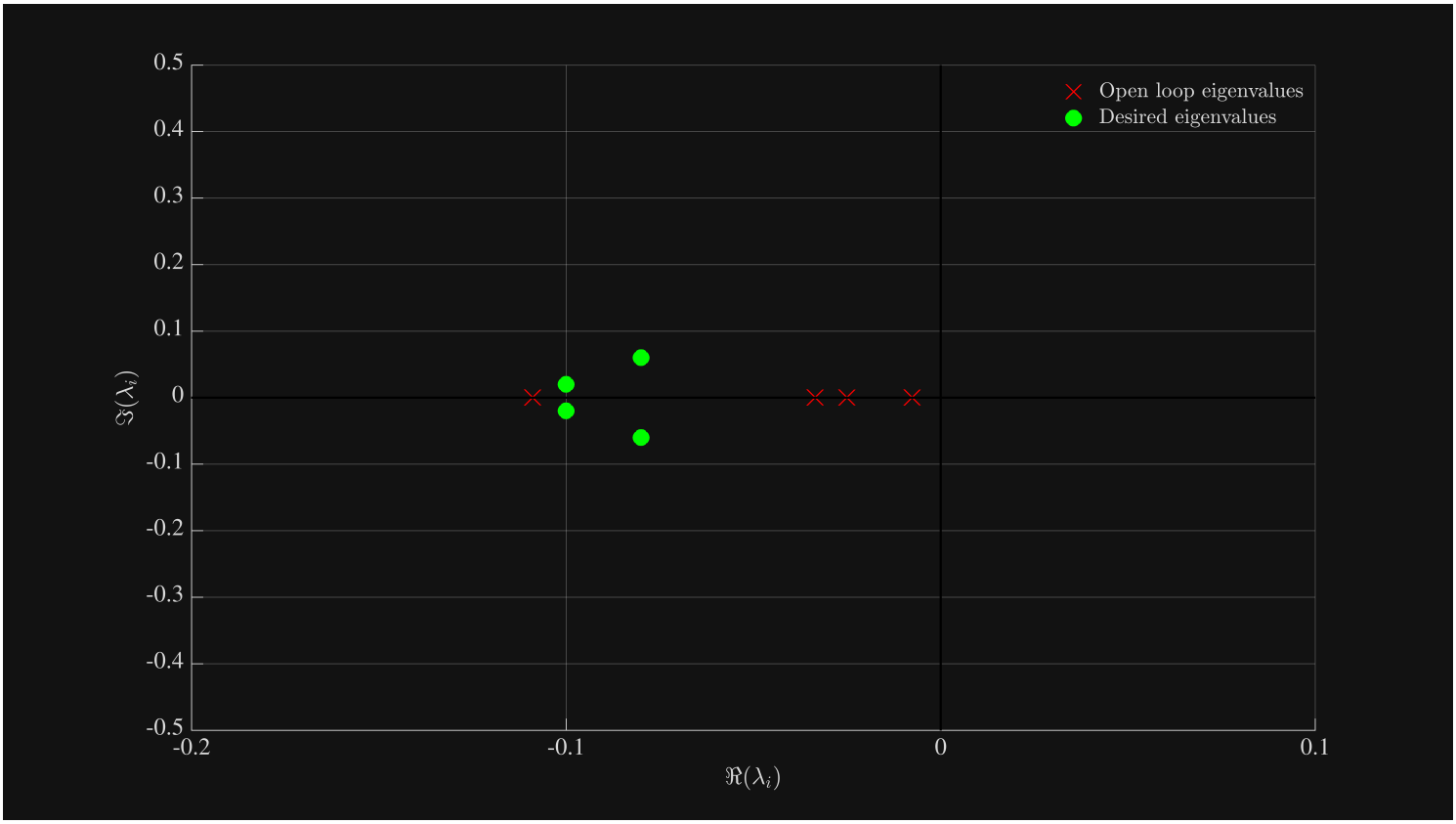
System stability

```
lambda_OL = eig(A); % open loop eigenvalues
if sum(real(lambda_OL) < 0) == size(A,1)
    disp('Open loop system is asymptotically stable')
elseif sum(real(lambda_OL) < 0) < size(A,1)
    tmp = size(A,1) - sum(real(lambda_OL) < 0);
    if sum(real(lambda_OL) > 0) >= 1 || tmp > 1
        disp('Open loop system is unstable')
```

```
    elseif sum(real(lambda_OL) > 0) == 0 && tmp == 1
        disp('Open loop system is marginally stable')
    end
end
```

```
Open loop system is asymptotically stable
```

```
TwoTankSys = ss(A,B,C,D);
[w,z] = damp(TwoTankSys);

figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
h1 = axes; set(h1,'FontName','times','FontSize',16)
hold on, grid on
h1 =
plot(real(lambda_OL),imag(lambda_OL),'x','MarkerSize',15,'MarkerEdgeColor','r
','LineWidth',1.5);
line([floor(min(real(lambda_OL))) ceil(max(real(lambda_OL)))+0.1],[0
0],'Color','k','LineWidth',1.5)
line([0 0],[floor(min(imag(lambda_OL)))-1 ceil(max(imag(lambda_OL)))
+1],'Color','k','LineWidth',1.5)
xlabel('$\Re(\lambda_i)
$','FontName','times','FontSize',16,'Interpreter','latex')
ylabel('$\Im(\lambda_i)
$','FontName','times','FontSize',16,'Interpreter','latex')
xlim([-0.2 0.1])
ylim([-0.5 0.5])

lambda_CL_des = [-0.1+0.02*1i -0.1-0.02*1i -0.08+0.06*1i -0.08-0.06*1i]'; %
desired eigenvalues for the closed loop system
h2 =
plot(real(lambda_CL_des),imag(lambda_CL_des),'o','MarkerSize',10,'MarkerEdgeC
olor','g','MarkerFaceColor','g','LineWidth',1.5);
leg = legend([h1 h2],'Open loop eigenvalues','Desired eigenvalues');
set(leg,'FontName','times','FontSize',14,'Interpreter','latex','box','off')
```



Controllability analysis

```
Mc = ctrb(A,B);
if rank(Mc) == size(A,1)
    disp('Open loop system is controllable')
else
    dimCs = size(A,1) - rank(Mc); % dimension of controllability subspace
    disp('Open loop system is not controllable and the controllability
subspace has dimension ',dimCs)
end
```

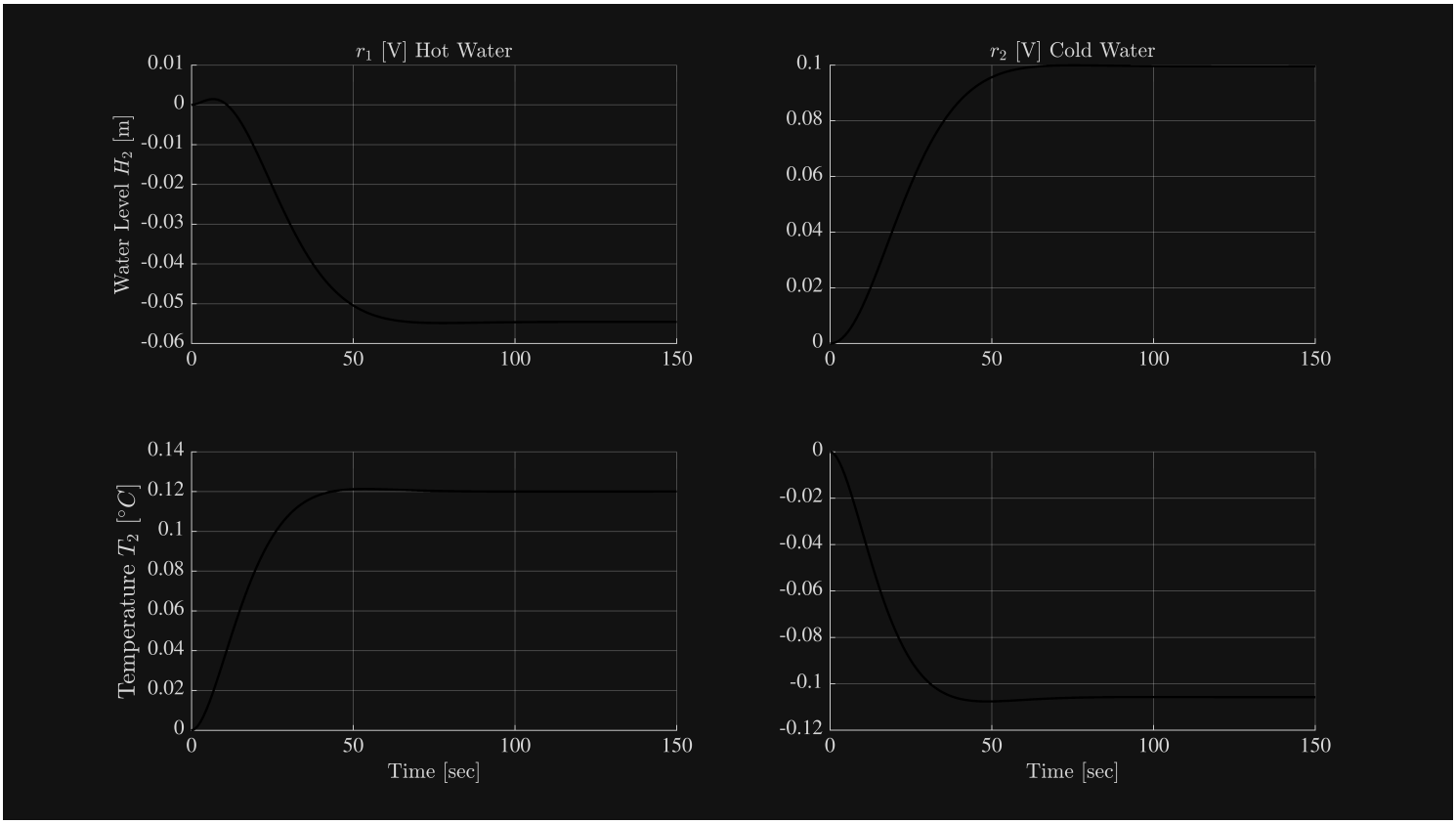
```
Open loop system is controllable
```

Full state feedback control design (Continuous Time)

```
K_fs = place(A,B,lambda_CL_des); % continuous time feedback controller gain
K_CT = K_fs;
Ak = A-B*K_fs;
lambda_CL = eig(Ak);
if sum(lambda_CL-lambda_CL_des) < 10^-6
    disp('Closed loop specifications fulfilled')
end
```

Closed loop specifications fulfilled

```
TwoTankSys_CL = ss(Ak,B,C,D);

[y_CL,t_CL] = step(TwoTankSys_CL,150);
figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
hf1 = subplot(2,2,1); set(hf1,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL,y_CL(:,1,1)/kh,'k','LineWidth',1.5)
ylabel('Water Level $H_2$
[m]','FontName','times','FontSize',14,'Interpreter','latex')
title('$r_1$ [V] Hot
Water','FontName','times','FontSize',14,'Interpreter','latex')
hf2 = subplot(2,2,2); set(hf2,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL,y_CL(:,1,2)/kh,'k','LineWidth',1.5)
title('$r_2$ [V] Cold
Water','FontName','times','FontSize',14,'Interpreter','latex')
hf3 = subplot(2,2,3); set(hf3,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL,y_CL(:,2,1)/kt,'k','LineWidth',1.5)
ylabel('Temperature $T_2$ $[^\circ C]
$','FontName','times','FontSize',16,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
hf4 = subplot(2,2,4); set(hf4,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL,y_CL(:,2,2)/kt,'k','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
```



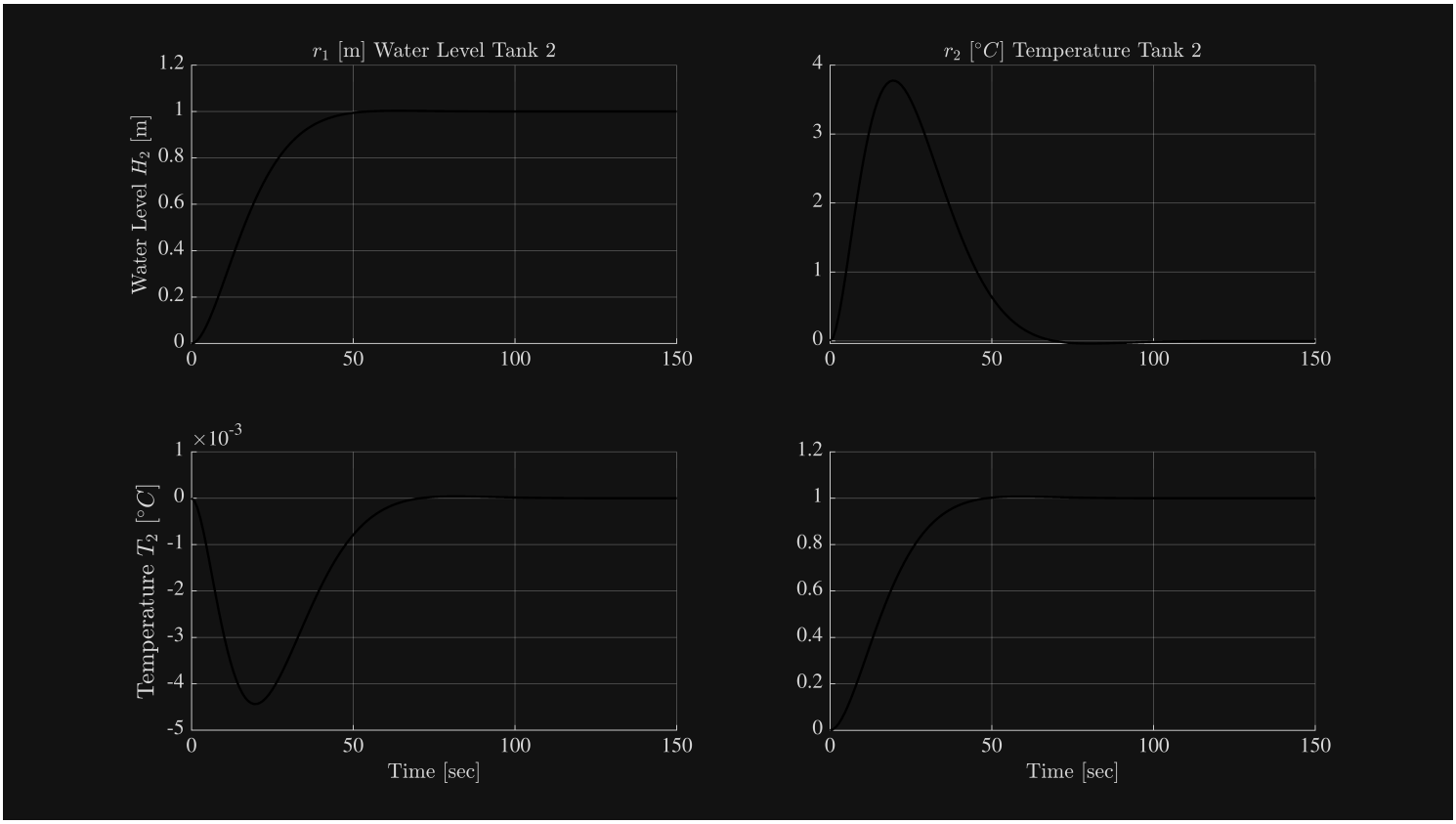
Set point changes

```
r0 = [0 0]';
r1 = [0.5 -0.5]';
Tref1 = 100;
Tref2 = 500;
```

Full state feedback + reference feedforward (Continuous Time)

```
kappa = -C/Ak*B; % steady state gain
N = kappa^-1; % reference feedforward gain
Bff = B*N;

TwoTankSys_CL_FF = ss(Ak,Bff,C,D);

[y_CL_FF,t_CL_FF] = step(TwoTankSys_CL_FF,150);
figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
hf1 = subplot(2,2,1); set(hf1,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL_FF,y_CL_FF(:,1,1),'k','LineWidth',1.5)
ylabel('Water Level $H_2$
[m]','FontName','times','FontSize',14,'Interpreter','latex')
title('$r_1$ [m] Water Level Tank
2','FontName','times','FontSize',14,'Interpreter','latex')
```

```
hf2 = subplot(2,2,2); set(hf2,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL_FF,y_CL_FF(:,1,2),'k','LineWidth',1.5)
title('$r_2$ [$^\circ C$] Temperature Tank
2','FontName','times','FontSize',14,'Interpreter','latex')
hf3 = subplot(2,2,3); set(hf3,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL_FF,y_CL_FF(:,2,1),'k','LineWidth',1.5)
ylabel('Temperature $T_2$ $[^\circ C]
$','FontName','times','FontSize',16,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
hf4 = subplot(2,2,4); set(hf4,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL_FF,y_CL_FF(:,2,2),'k','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
```



Set point changes

```
H_ref0 = y0(1)/kh;
T_ref0 = y0(2)/kt;
H_ref1 = H_ref0 + 0.5;
T_ref1 = T_ref0 + 1;
r0 = [H_ref0 T_ref0]';
Tref1 = 100;
Tref2 = 500;
```

Full state feedback control with integral action (Continuous Time)

```
lambda_CL_des = [-0.095+0.03*1i -0.095-0.03*1i -0.08+0.06*1i -0.08-0.06*1i
-0.05+0.085*1i -0.05-0.085*1i]';
Aof = [A zeros(4,2);-C zeros(2,2)];
Bof = [B;zeros(2,2)];
Br = [zeros(4,2);eye(2)];
Cof = [C zeros(2)];
K_of = place(Aof,Bof,lambda_CL_des)
```

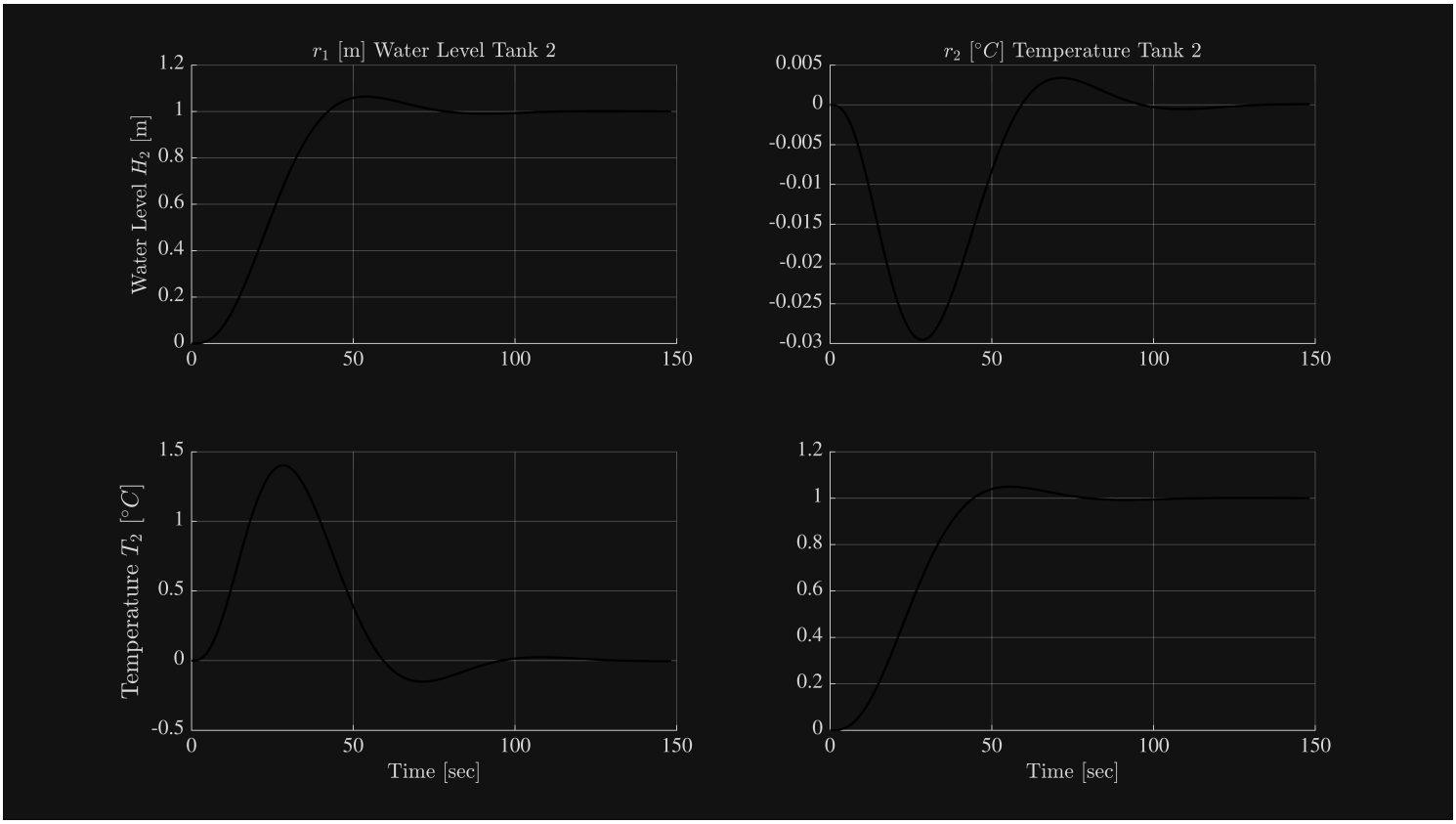
```
K_of = 2×6
   17.8742   42.0101    1.7305    3.3677   -1.6218   -1.1813
    2.7720    7.5225   -2.7691   -8.0574    0.1323    4.7237
```

```
Ak_of = Aof-Bof*K_of;
K = K_of(:,1:4);
Ki = -K_of(:,5:6);
u0 = u0(1:2,1);
xi0 = inv(Ki)*(u0+K*x0);
xi0_lin = zeros(size(C,1),1);
lambda_CL = eig(Ak_of);
if sum(lambda_CL-lambda_CL_des) < 10^-6
    disp('Closed loop specifications fulfilled')
end
```

```
Closed loop specifications fulfilled
```

```
Br(5,1) = kh;
Br(6,2) = kt;
TwoTankSys_CL_i = ss(Ak_of,Br,Cof,zeros(2,2));

[y_CL_i,t_CL_i] = step(TwoTankSys_CL_i);

figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
hf5 = subplot(2,2,1); set(hf5,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL_i,y_CL_i(:,1,1)/kh,'k','LineWidth',1.5)
ylabel('Water Level $H_2$
[m]','FontName','times','FontSize',14,'Interpreter','latex')
title('$r_1$ [m] Water Level Tank
2','FontName','times','FontSize',14,'Interpreter','latex')
hf6 = subplot(2,2,2); set(hf6,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL_i,y_CL_i(:,1,2)/kh,'k','LineWidth',1.5)
title('$r_2$ [$^\circ C$] Temperature Tank
2','FontName','times','FontSize',14,'Interpreter','latex')
hf7 = subplot(2,2,3); set(hf7,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL_i,y_CL_i(:,2,1)/kt,'k','LineWidth',1.5)
ylabel('Temperature $T_2$ $[^\circ C]
$','FontName','times','FontSize',16,'Interpreter','latex')
```

```
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
hf8 = subplot(2,2,4); set(hf8,'FontName','times','FontSize',14)
hold on, grid on
plot(t_CL_i,y_CL_i(:,2,2)/kt,'k','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
```



Full order observer design

```
% Observability test
Mo = obsv(A,C);
if rank(Mo) == size(A,1)
    disp('Open loop system is observable')
else
    dimOs = size(A,1) - rank(Mo); % dimension of controllability subspace
    disp('Open loop system is not observable and the observability subspace
has dimension ')
    disp(dimOs)
end
```

```
Open loop system is observable
```

```
lambda_obs_des = 5*lambda_OL;
L = place(A',C',lambda_obs_des)';

lambda_obs = eig(A-L*C);
if sum(lambda_obs-lambda_obs_des) < 10^-6
```

```
    disp('Observer dynamics specifications fulfilled')
end
```

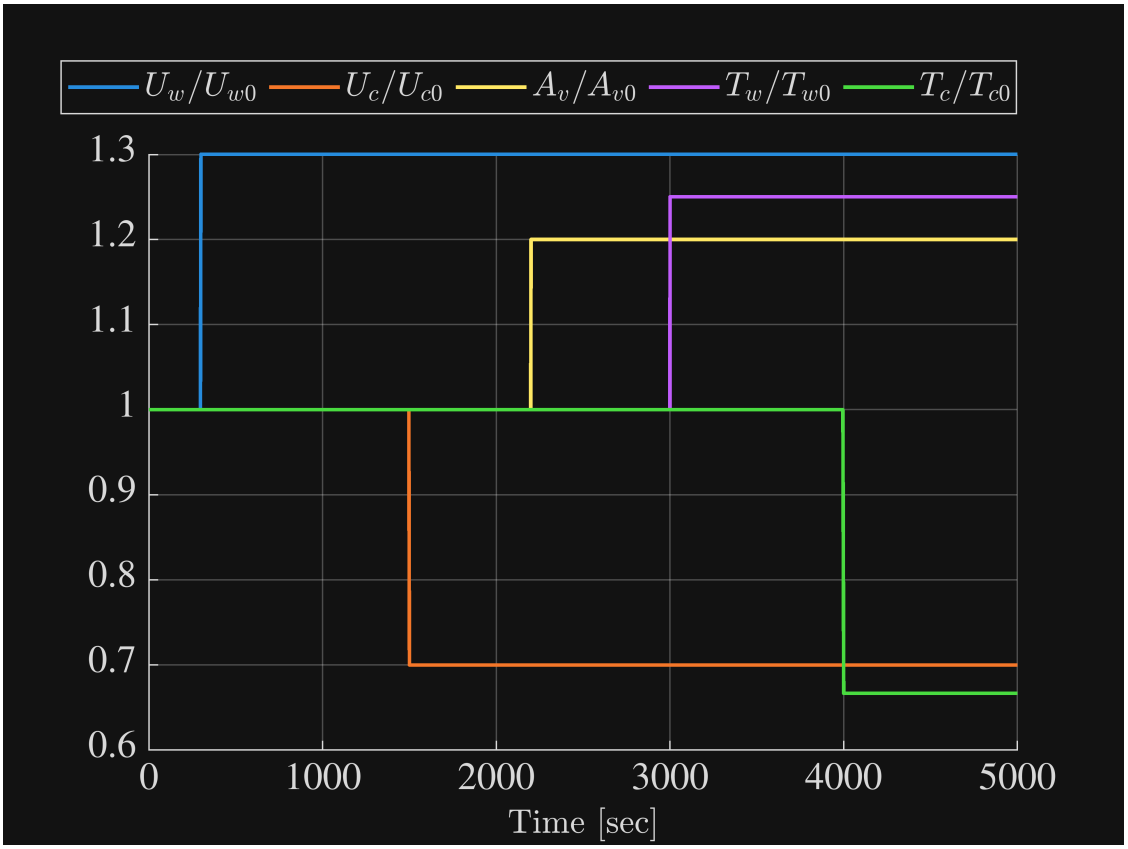
```
Observer dynamics specifications fulfilled
```

Simulate observer

```
xhat0 = 1.0*x0-x0;
Y0 = C*x0;

% input steps
SIM_TIME = 5000;
time = (0:STEP_SIZE:SIM_TIME)';
Tw = 300; % step time
uw0 = 5;
uw1 = 6.5; % [V]
Tc = 1500; % step time
uc0 = 5;
uc1 = 3.5; % [V]
% disturbance step
Tv = 2200; % step time
Av0 = 0.0122;
Av1 = 1.2*0.0122; % 20% more [m^2]
T1 = 3000; % step time
T2 = 4000; % step time
Tw0 = 60; % [C]
Tw1 = 75; % [C]
Tc0 = 30; % [C]
Tc1 = 20; % [C]
```

Plot

```
figure, h1 = axes; set(h1,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,[uw0*ones(length(time(1:Tw/STEP_SIZE)),1); uw1*ones(length(time(Tw/
STEP_SIZE+1:end)),1)]./uw0,'LineWidth',1.5)
plot(time,[uc0*ones(length(time(1:Tc/STEP_SIZE)),1); uc1*ones(length(time(Tc/
STEP_SIZE+1:end)),1)]./uc0,'LineWidth',1.5)
plot(time,[Av0*ones(length(time(1:Tv/STEP_SIZE)),1); Av1*ones(length(time(Tv/
STEP_SIZE+1:end)),1)]./Av0,'LineWidth',1.5)
plot(time,[Tw0*ones(length(time(1:T1/STEP_SIZE)),1); Tw1*ones(length(time(T1/
STEP_SIZE+1:end)),1)]./Tw0,'LineWidth',1.5)
plot(time,[Tc0*ones(length(time(1:T2/STEP_SIZE)),1); Tc1*ones(length(time(T2/
STEP_SIZE+1:end)),1)]./Tc0,'LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
leg1 = legend('$U_w/U_{w0}$','$U_c/U_{c0}$','$A_v/A_{v0}$','$T_w/
T_{w0}$','$T_c/T_{c0}$');
set(leg1,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal','Location','northoutside')
```




```
H1 = logsout{3}.Values.Data(:,1);
```

```
Brace indexing is not supported for variables of this type.
```

```
H2 = logsout{3}.Values.Data(:,2);
Tp1 = logsout{3}.Values.Data(:,3);
Tp2 = logsout{3}.Values.Data(:,4);
H1_hat = logsout{1}.Values.Data(:,1);
H2_hat = logsout{1}.Values.Data(:,2);
T1_hat = logsout{1}.Values.Data(:,3);
T2_hat = logsout{1}.Values.Data(:,4);

figure, h2 = subplot(2,2,1); set(h2,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,H1,time,H1_hat,'--r','LineWidth',1.5)
ylabel('[m]','FontName','times','FontSize',14,'Interpreter','latex')
leg2 = legend('$H_1$','$\hat{H}_1$');
set(leg2,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal')
h3 = subplot(2,2,2); set(h3,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,H2,time,H2_hat,'--r','LineWidth',1.5)
ylabel('[m]','FontName','times','FontSize',14,'Interpreter','latex')
leg3 = legend('$H_2$','$\hat{H}_2$');
set(leg3,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal')
```

```
h4 = subplot(2,2,3); set(h4,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,Tp1,time,T1_hat,'--r','LineWidth',1.5)
ylabel('[$^\circ C$]','FontName','times','FontSize',14,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
leg4 = legend('$T_1$','$\hat{T}_1$');
set(leg4,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal')
h5 = subplot(2,2,4); set(h5,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,Tp2,time,T2_hat,'--r','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
leg5 = legend('$T_2$','$\hat{T}_2$');
set(leg5,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal')
```

Changed output equation

```
C1 = [0 kh 0 0;0 0 kt 0;0 0 0 kt];
Mo1 = obsv(A,C1);
if rank(Mo1) == size(A,1)
    disp('Open loop system is observable')
else
    dimOs = size(A,1) - rank(Mo1); % dimension of controllability subspace
    disp('Open loop system is not observable and the observability subspace
has dimension ')
    disp(dimOs)
end
```

```
Open loop system is observable
```

```
lambda_obs_des = 5*lambda_OL;
L1 = place(A',C1',lambda_obs_des)';

lambda_obs = eig(A-L1*C1);
if sum(lambda_obs-lambda_obs_des) < 10^-6
    disp('Observer dynamics specifications fulfilled')
end
```

```
Observer dynamics specifications fulfilled
```

```
C2 = [0 kh 0 0;0 0 kt 0];
Mo2 = obsv(A,C2);
if rank(Mo2) == size(A,1)
    disp('Open loop system is observable')
else
    dimOs = size(A,1) - rank(Mo2); % dimension of controllability subspace
    disp('Open loop system is not observable and the observability subspace
has dimension ')
```

```
    disp(dimOs)
end
```

```
Open loop system is not observable and the observability subspace has dimension
     1
```

Simulate observer

```
H1 = logsout{4}.Values.Data(:,1);
```

```
Brace indexing is not supported for variables of this type.
```

```
H2 = logsout{4}.Values.Data(:,2);
Tp1 = logsout{4}.Values.Data(:,3);
Tp2 = logsout{4}.Values.Data(:,4);
H1_hat = logsout{3}.Values.Data(:,1);
H2_hat = logsout{3}.Values.Data(:,2);
T1_hat = logsout{3}.Values.Data(:,3);
T2_hat = logsout{3}.Values.Data(:,4);

figure, h2 = subplot(2,2,1); set(h2,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,H1,time,H1_hat,'--r','LineWidth',1.5)
ylabel('[m]','FontName','times','FontSize',14,'Interpreter','latex')
leg2 = legend('$H_1$','$\hat{H}_1$');
set(leg2,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal')
h3 = subplot(2,2,2); set(h3,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,H2,time,H2_hat,'--r','LineWidth',1.5)
ylabel('[m]','FontName','times','FontSize',14,'Interpreter','latex')
leg3 = legend('$H_2$','$\hat{H}_2$');
set(leg3,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal')
h4 = subplot(2,2,3); set(h4,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,Tp1,time,T1_hat,'--r','LineWidth',1.5)
ylabel('[$^\circ C$]','FontName','times','FontSize',14,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
leg4 = legend('$T_1$','$\hat{T}_1$');
set(leg4,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal')
h5 = subplot(2,2,4); set(h5,'FontSize',16,'FontName','times')
hold on, grid on
plot(time,Tp2,time,T2_hat,'--r','LineWidth',1.5)
xlabel('Time [sec]','FontName','times','FontSize',14,'Interpreter','latex')
leg5 = legend('$T_2$','$\hat{T}_2$');
set(leg5,'FontName','times','FontSize',14,'Interpreter','latex','Orientation'
,'Horizontal')
```

Observer design for disturbance rejection

```
clc
close all
clear all

% System dynamics
zeta = 0.1;
wn1 = 0.3;

A = [0 1;-wn1^2 -2*zeta*wn1];
B = [0 1]';
Bv = [0 0.5]';
C = [1 0];
x0 = [0 0]';

% Disturbance dynamics
disturbance = 'sinusoidal';
switch disturbance
    case 'constant'
        Aw = 0;
        Cw = 1;
        w0 = 0.8;
    case 'sinusoidal'
        wn2 = 0.6;
        Aw = [0 1;-wn2^2 0];
        Cw = [1 0];
        w0 = [0.8 0]';
end

% Augmented system x_a = [x w]'
Axw = Bv*Cw;
Aa = [A Axw;zeros(size(Aw,1),size(A,2)) Aw];
Ba = [B' zeros(1,size(Aw,1))]';
Ca = [C zeros(1,size(Aw,1))];

% Controllability and observability analysis
Mc = ctrb(Aa,Ba);
if rank(Mc) == size(Aa,1)
    disp('System is controllable')
else
    dim_Anc = size(Aa,1)-rank(Mc);
    disp('System uncontrollable subspace has dimension '), disp(dim_Anc)
    disp('Controllable subspace decomposition')
    [Aac,Bac,Cac,Q,~] = ctrbf(Aa,Ba,Ca)
    Ac = Aac(dim_Anc+1:end,dim_Anc+1:end);
    Bc = Bac(dim_Anc+1:end,1);
    lambda_ctrl_des = [-0.5+0.5*1i,-0.5-0.5*1i];
    Kt1_1 = acker(Ac,Bc,lambda_ctrl_des)
    lambda_ctrl = eig(Ac-Bc*Kt1_1)
```

```
    switch disturbance
        case 'constant'
            Kt2_1 = 0;
            Kt2_2 = 0.5;
        case 'sinusoidal'
            Kt2_1 = [0 0];
            Kt2_2 = [0 0.5];
    end
    Kt1 = [Kt2_1 Kt1_1];
    Kt2 = [Kt2_2 Kt1_1];
    K1 = Kt1*Q
    K2 = Kt2*Q
end
```

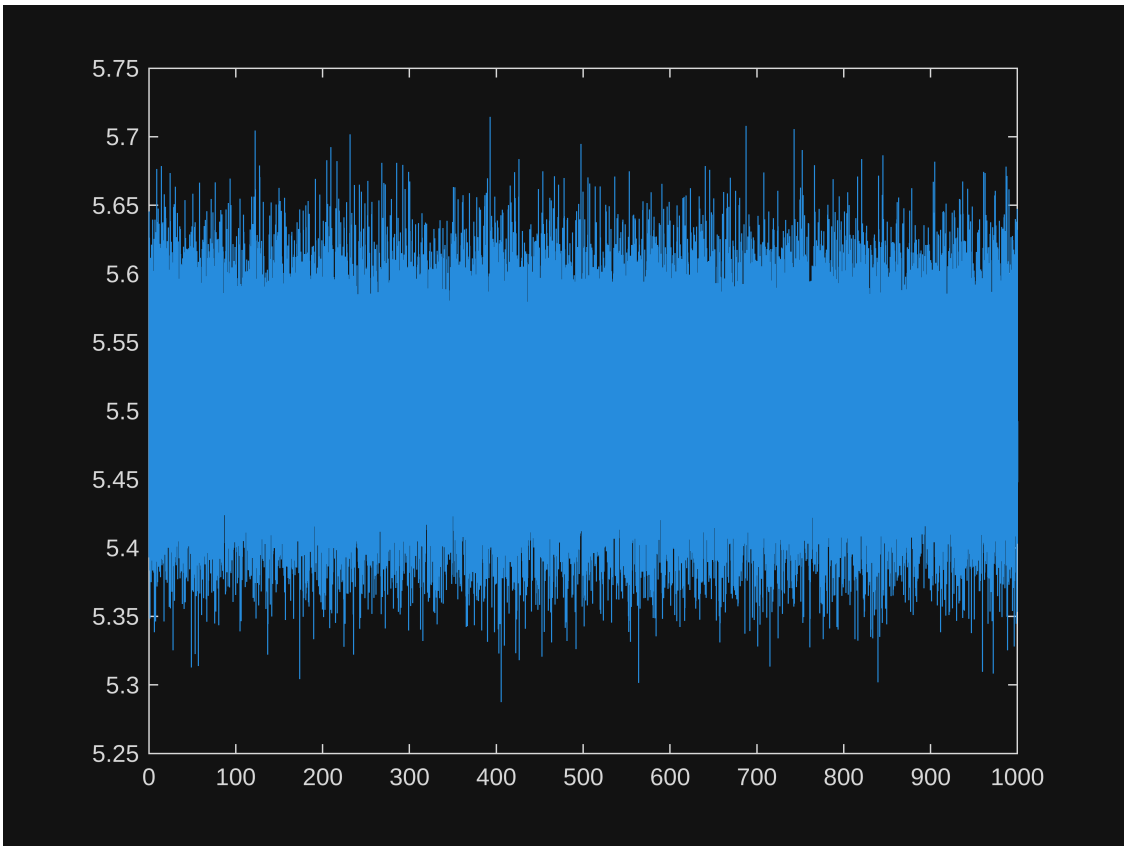
```
System uncontrollable subspace has dimension
     2
Controllable subspace decomposition
Aac = 4×4
         0   -0.3600         0         0
    1.0000         0         0         0
         0         0         0    1.0000
         0    0.5000   -0.0900   -0.0600
Bac = 4×1
     0
     0
     0
     1
Cac = 1×4
     0     0     1     0
Q = 4×4
     0     0     0     1
     0     0     1     0
     1     0     0     0
     0     1     0     0
Kt1_1 = 1×2
    0.4100    0.9400
lambda_ctrl = 2×1 complex
  -0.5000 + 0.5000i
  -0.5000 - 0.5000i
K1 = 1×4
    0.4100    0.9400         0         0
K2 = 1×4
    0.4100    0.9400    0.5000         0
```

```
Mo = obsv(Aa,Ca);
if rank(Mo) == size(Aa,1)
    disp('System is observable')
    disp('Observer design')
    switch disturbance
        case 'constant'
           lambda_obs_des = [5*lambda_ctrl_des, -2]
           xhat0 = rand(3,1);
        case 'sinusoidal'
           lambda_obs_des = [5*lambda_ctrl_des, -2, -2.35]
           xhat0 = rand(4,1);
    end
```

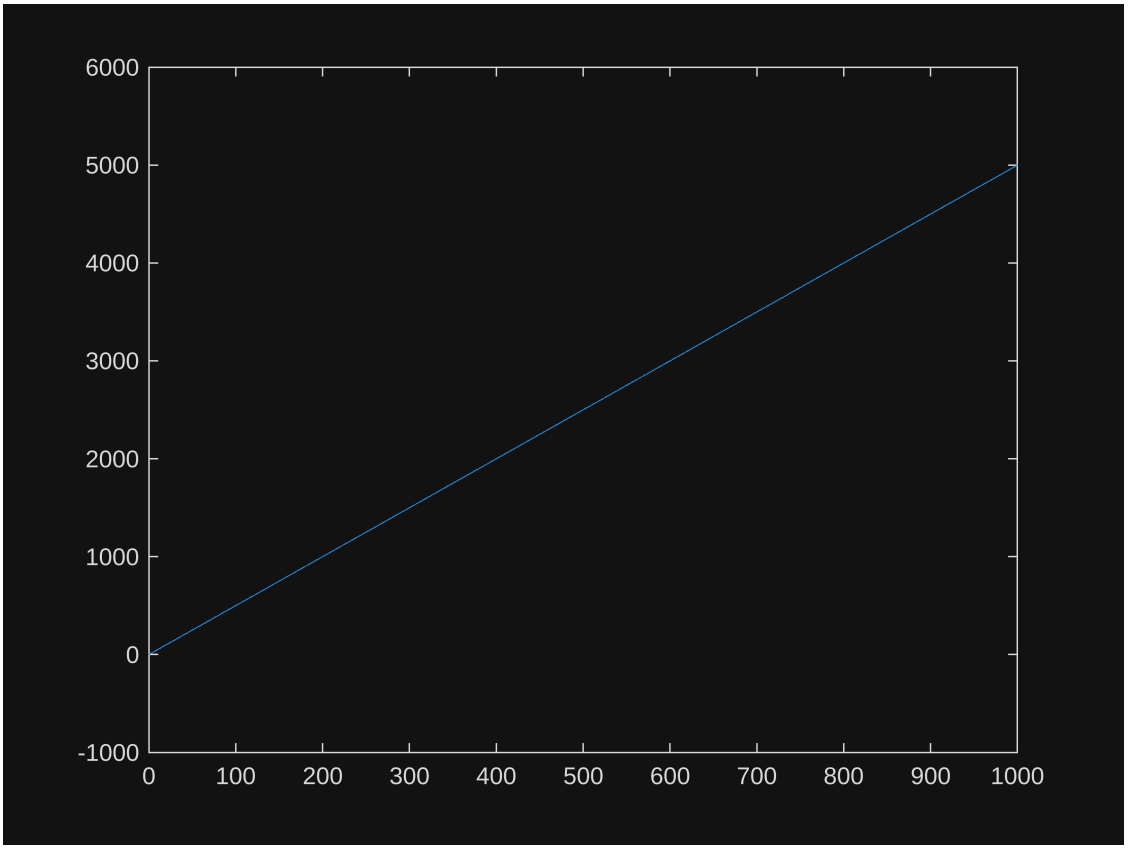
```
    L = acker(Aa',Ca',lambda_obs_des)';
    lambda_obs = eig(Aa-L*Ca)
else
    disp('System observable subspace has dimension '), size(Aa,1)-rank(Mo)
end
```

```
System is observable
Observer design
lambda_obs_des = 1×4 complex
  -2.5000 + 2.5000i  -2.5000 - 2.5000i  -2.0000 + 0.0000i  -2.3500 + 0.0000i
lambda_obs = 4×1 complex
  -2.5000 + 2.5000i
  -2.5000 - 2.5000i
  -2.3500 + 0.0000i
  -2.0000 + 0.0000i
```

```
clear all
clc
close all

% Generation of measured data
SIM_TIME = 1000; % simulation time [s]
STEP_SIZE = 0.001;
Ts = 0.01; % sampling time [s]
time = (0:Ts:SIM_TIME)';
b0 = 0.5*ones(length(time),1); % constant bias [m/s]
v0 = 5*ones(length(time),1); % true velocity [m/s]
v_noise = 0.05*randn(length(time),1);
v_m = v0 + b0 + v_noise; % measured velocity

x = zeros(length(time),1);
x_noise = randn(length(time),1);

% True position
for ii = 2:length(v0)
    x(ii,1) = x(ii-1,1) + Ts*v0(ii-1,1);
end
x_m = x + x_noise; % measured position

figure, plot(time,v_m)
```




```
figure, plot(time,x_m)
```




```
% Continuous-time model for the Kalman filter
A=[0 -1;0 0];
B=[1;0];
M=[0;1];
C=[1 0];
D=0;
sysc=ss(A,B,C,D);

% Noise intensities
V=1; % noise intensity of position measurement
q=0.1; W=q^2; % noise intensity of fictitious process noise on bias state

% Compute Discrete-time model
sysd = c2d(sysc,Ts,'zoh');
[F,G,C,D] = ssdata(sysd);

% Noise variances
Vd = V/Ts; % measurement noise variance
Wd_approx = M*W*M'*Ts; % first order approximation from discretization of
process noise
Wd=[1/3*q^2*Ts^3 -1/2*q^2*Ts^2;-1/2*q^2*Ts^2 q^2*Ts];
% Discrete-time Kalman filter using dlqe:
[Ld1,Pd1,Zd1]=dlqe(F,eye(2),C,Wd,Vd)
```

```
Ld1 = 2×1
```

```
    0.0045
   -0.0010
Pd1 = 2×2
    0.4482   -0.1002
   -0.1002    0.0448
Zd1 = 2×2
    0.4462   -0.0998
   -0.0998    0.0447
```

```
[F_kf,G_kf,C_kf,D_kf] = destim(F,G,C,D,Ld1,1,1)
```

```
F_kf = 2×2
    0.9955   -0.0100
    0.0010    1.0000
G_kf = 2×2
    0.0100    0.0045
         0   -0.0010
C_kf = 3×2
    0.9955         0
    0.9955         0
    0.0010    1.0000
D_kf = 3×2
         0    0.0045
         0    0.0045
         0   -0.0010
```

```
xhat0 = zeros(2,1);
% Discrete-time Kalman filter using lqed (from continuous-time data):
[Ld2,Pd2,Zd2]=lqed(A,M,C,W,V,Ts) %==> same solution !
```

```
Ld2 = 2×1
    0.0045
   -0.0010
Pd2 = 2×2
    0.4482   -0.1002
   -0.1002    0.0448
Zd2 = 2×2
    0.4462   -0.0998
   -0.0998    0.0447
```

```


% Run simulation
sim('Bias_estimation_KF',SIM_TIME);
```

Example 7.1. **Time Dependent Kalman Filter**

This example concerns a Kalman filter designed around a first order low pass filter. The system equations are

$$\dot{x}(t) = -ax(t) + bu(t) + v_1(t), \tag{7.21}$$

$$y(t) = x(t) + v_2(t), \tag{7.22}$$

where a and b are constants and $v_1(t) \in N(0, V_1)$ and $v_2(t) \in N(0, V_2)$ are white noise sources such that

$$E\{v_1(t)\} = 0, R_{v_1}(t_1 - t_2) = \sigma_{v_1}^2 \delta(t_1 - t_2), \tag{7.23}$$

$$E\{v_2(t)\} = 0, R_{v_2}(t_1 - t_2) = \sigma_{v_2}^2 \delta(t_1 - t_2). \tag{7.24}$$

The initial conditions for the system are $E\{x(0)\} = 0$ and $E\{x^2(0)\} = q_0$. The Riccati equation for this system can be written as

$$\dot{q} = -2aq + \sigma_{v_1}^2 - \left(\frac{1}{\sigma_{v_2}^2}\right) q^2. \tag{7.25}$$

In this simple case the Riccati equation can be directly integrated to find $q(t)$:

$$\begin{aligned} q(t) &= q_1 + \frac{q_1 + q_2}{[(q_0 + q_2)/(q_0 + q_1)]e^{2at} - 1}, \\ &\Rightarrow q(t) \to q_1, \quad \text{for} \quad t \to \infty, \end{aligned} \tag{7.26}$$

Example 7.1 Time dependent Kalman filter

```
close all
clear all
clc

a = -1;
b = 1;
sigma_1 = 0.5; % standard deviation process noise
sigma_2 = 0.1; % standard deviation measurement noise
tc = 0.01; % correlation time of the white noise
x0 = 0;
q0 = 1.5;

% Simulation and plot time,y,'.-b',
sim('Example7_1.slx')
```

```
Warning: Square root of a negative number in 'Example7_1/Sqrt'. Consider setting the 'Output signal type' to complex.
```

```
v1 = logsout.getElement('v1').Values.Data;
v2 = logsout.getElement('v2').Values.Data;
x = logsout.getElement('x').Values.Data;
time = logsout.getElement('x').Values.Time;
y = logsout.getElement('y').Values.Data;
xhat = logsout.getElement('xhat').Values.Data;
inn = logsout.getElement('inn').Values.Data;
u = logsout.getElement('u').Values.Data;
Q = logsout.getElement('Q').Values.Data;
L = logsout.getElement('L').Values.Data;

sigma_1_cl = std(v1);
sigma_2_cl = std(v2);

% Lyapunov equation

Qx_th = lyap(a,sigma_1^2)
```

```
Qx_th = 0.1250
```

```
Qx_cl = var(x)
```

```
Qx_cl = 0.1503
```

```
[Lss,Qe_th,~] = lqe(a,1,1,sigma_1^2,sigma_2^2)
```

```
Lss = 4.0990
Qe_th = 0.0410
```

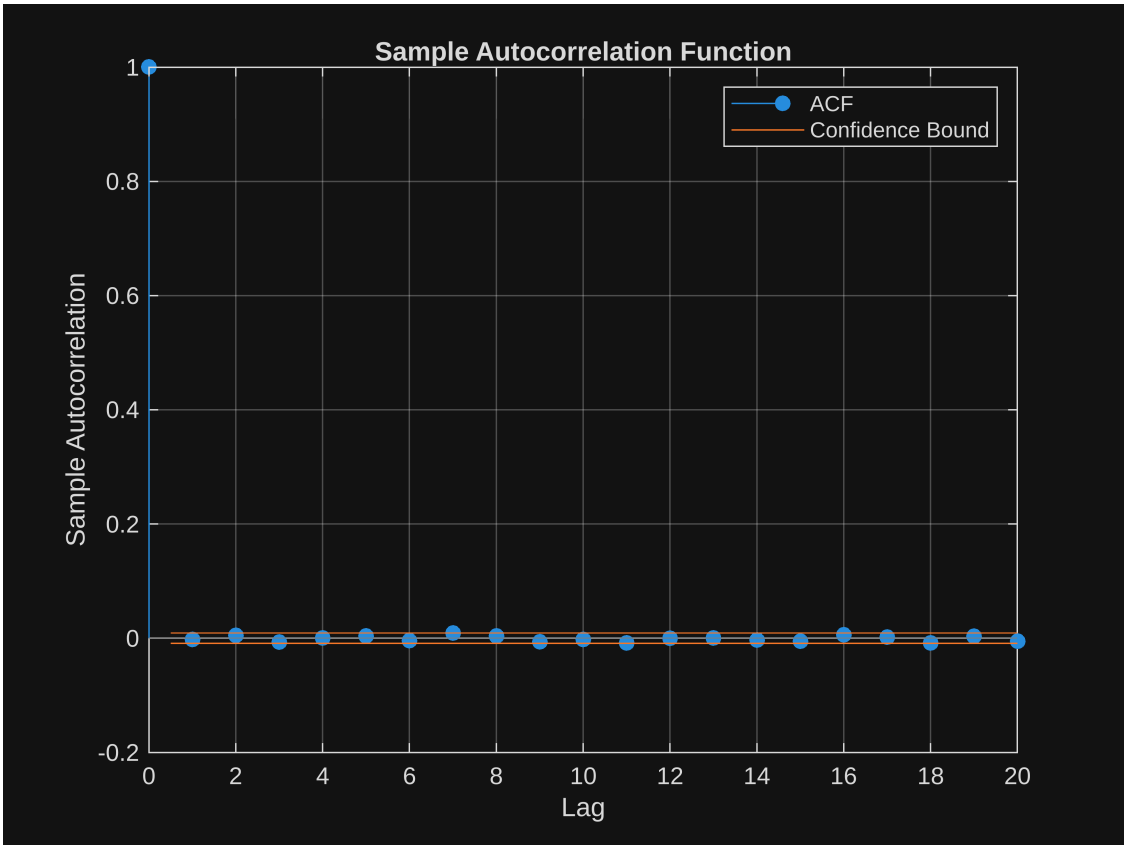
```
Qe_cl = var(x-xhat)
```

```
Qe_cl = 0.0408
```

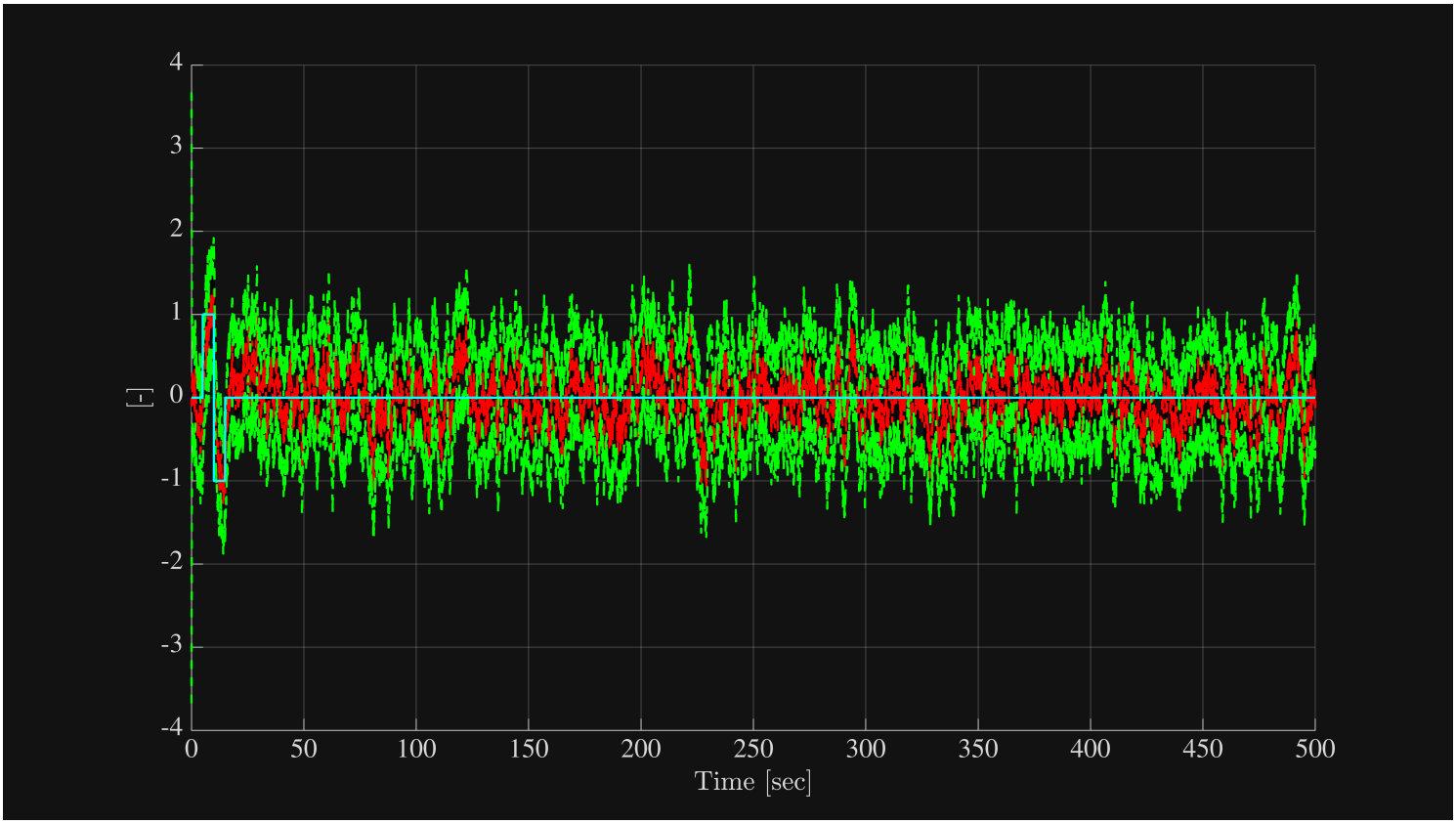
```
Qi_cl = var(inn)*tc
```

```
Qi_cl = 0.0104
```

```
figure, autocorr(inn)
```



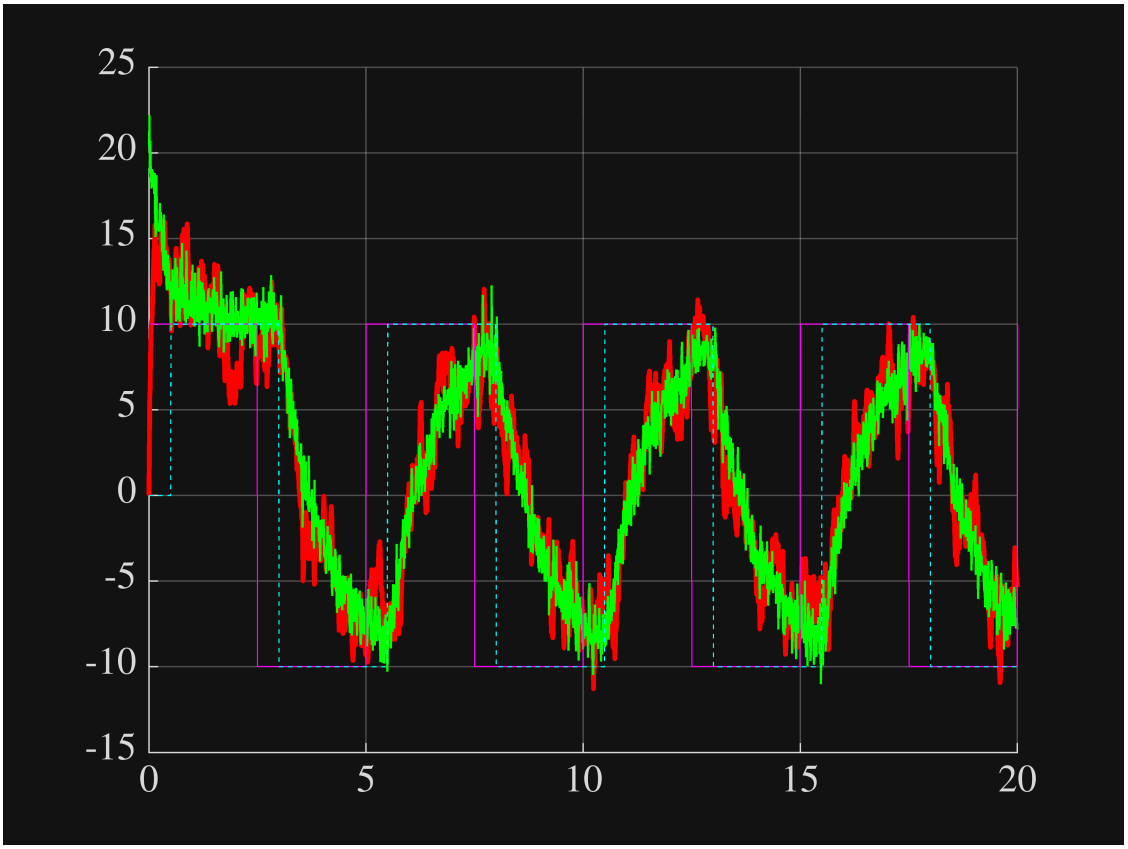

```
figure('units','normalized','outerposition',[0 0 1
1],'PaperOrientation','landscape','Renderer','Painter')
h1 = axes; set(h1,'FontName','times','FontSize',18)
hold on, grid on
plot(time,x,'-k',time,xhat,'--r',time,xhat+3*sqrt(Q),'--g',...
     time,xhat-3*sqrt(Q),'--g',time,u,'-c','LineWidth',1.5)
ylabel('[-]','FontName','times','FontSize',18,'Interpreter','latex')
xlabel('Time [sec]','FontName','times','FontSize',18,'Interpreter','latex')
```


4
3
2
1
0
-1
-2
-3
-4
[-]
0
50
100
150
200
250
300
350
400
450
500
Time [sec]

Kalman filtering of first order system with time delay

```
clc
close all
clear al

SIM_TIME = 20;
STEP_SIZE = 0.01;

% System parameter
alpha = -1; % [rad/s]
A = alpha;
B = 1;
Bw = 1;
C = 1;
D = 0;
x0 = 20;

tau = [0 0.1 0.5 1]; % time delay [s]
T = tau(3);

% Measurement noise
Vn = 1; % measurement noise intensity
Tc = 0.01; % correlation time of the noise

% Kalman filter design
A_kf = A;
B_kf = B;
Bw_kf = Bw;
C_kf = C;
D_kf = D;

Vw = 100; % process noise intensity

[L,P1,lambda] = lqe(A_kf,Bw_kf,C_kf,Vw,Vn);
xhat0 = 0;

% Generate measurements
sim('FirstOrderSystem_wDelay_KF',SIM_TIME);

time = logsout.getElement(1).Values.Time;
n = logsout.getElement(1).Values.Data;
x = logsout.getElement(3).Values.Data;
xhat = logsout.getElement(2).Values.Data;
u_tau = logsout.getElement(5).Values.Data;
u = logsout.getElement(6).Values.Data;

y = x+n*sqrt(Tc);

figure, h1 = axes; set(h1,'FontName','times','FontSize',16)
```

```
hold on, grid on
plot(time,x,'k','LineWidth',2)
plot(time,xhat,'r','LineWidth',2)
plot(time,y,'g','LineWidth',1)
plot(time,u,'m','LineWidth',0.5)
plot(time,u_tau,'--c','LineWidth',0.5)
```



Include 2nd order Padé approximation in the model of the Kalman filter

```
[numP,denP] = pade(T,2);
[Atau,Btau,Ctau,Dtau] = tf2ss(numP,denP);

A_kf = [A B*Ctau;zeros(2,1) Atau];
B_kf = [B*Dtau; Btau];
Bw_kf = [Bw; zeros(2,1)];
C_kf = [C zeros(1,2)];
D_kf = 0;

Vw = 1;

[L,P2,lambda] = lqe(A_kf,Bw_kf,C_kf,Vw,Vn);
xhat0 = zeros(3,1);

% Generate measurements
sim('FirstOrderSystem_wDelay_KF',SIM_TIME);
```

```
time = logsout.getElement(1).Values.Time;
n = logsout.getElement(1).Values.Data;
x = logsout.getElement(3).Values.Data;
xhat = logsout.getElement(2).Values.Data;
u_tau = logsout.getElement(5).Values.Data;
u = logsout.getElement(6).Values.Data;

y = x+n*sqrt(Tc);

figure, h1 = axes; set(h1,'FontName','times','FontSize',16)
hold on, grid on
plot(time,x,'k','LineWidth',2)
plot(time,xhat,'r','LineWidth',2)
plot(time,y,'g','LineWidth',1)
plot(time,u,'m','LineWidth',0.5)
plot(time,u_tau,'--c','LineWidth',0.5)
```