Solving ODEs in MATLAB

- Ordinary Differential Equations

$$\ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} + \frac{1}{m}F$$

1. ODEs Solver

Differential Equations

Example:

$$\boxed{\dot{x} = ax}$$

Where $a = -\frac{1}{T}$

T is the Time constant

Note! $\dot{x} = \frac{dx}{dt}$

The Solution can be proved to be (will not be shown here):

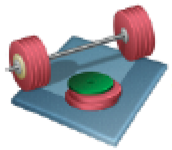
$$\boxed{x(t) = e^{at}x_0}$$

Use the following: $T = 5$

$$x(0) = 1$$

$$0 \leq t \leq 25$$

```
T = 5;
a = -1/T;
x0 = 1;
t = [0:1:25];

x = exp(a*t)*x0;

plot(t,x);
grid
```



Students: Try this example

1. ODEs Solver

Differential Equations

$$x(t) = e^{at}x_0$$

$T = 5$

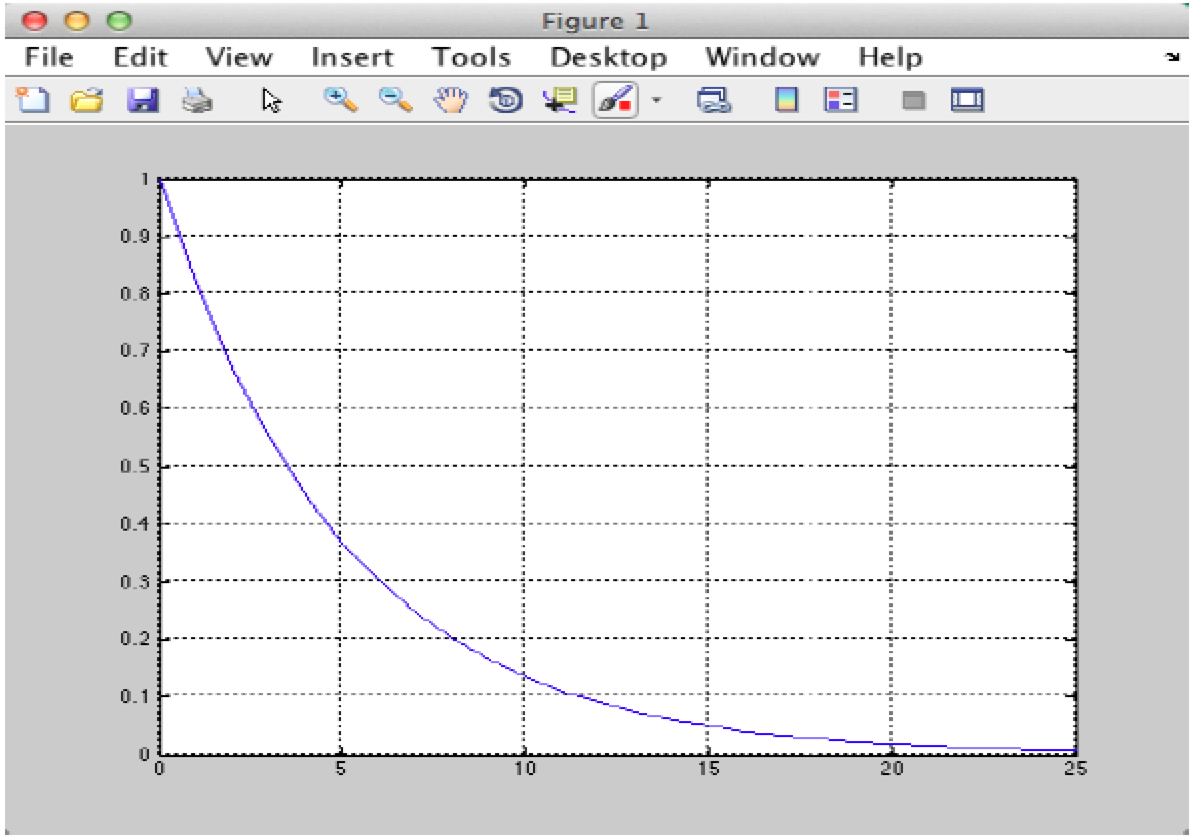
$x(0) = 1$

$a = -\frac{1}{T}$

$0 \leq t \leq 25$

```
T = 5;
a = -1/T;
x0 = 1;
t = [0:1:25];

x = exp(a*t)*x0;

plot(t,x);
grid
```

Problem with this method:
We need to solve the ODE before we can plot it!!



1. ODEs Solver

Using ODE Solvers in MATLAB

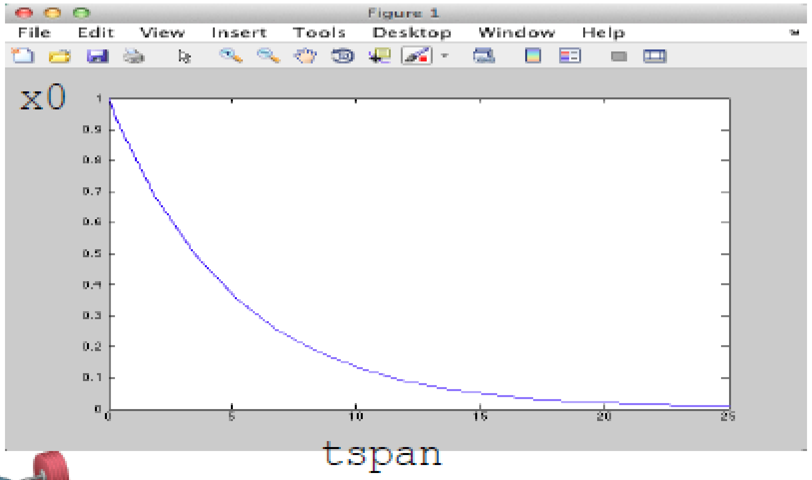
Example: $\dot{x} = ax$

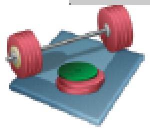
Step 1: Define the differential equation as a MATLAB function (`mydiff.m`):

```
function dx = mydiff(t,x)
T = 5;
a = -1/T;
dx = a*x;
```

Step 2: Use one of the built-in ODE solver (ode23, ode45, ...) in a Script.

```
clear
clc

tspan = [0 25];
x0 = 1;

[t,x] = ode23(@mydiff,tspan,x0);
plot(t,x)
```



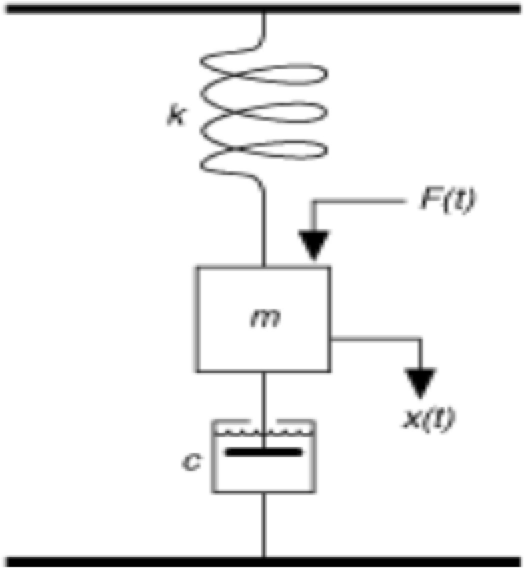


Students: Try this example. Do you get the same result?

1. ODEs Solver

Higher Order ODEs

Mass-Spring-Damper System



Example (2.order differential equation):

$$\boxed{\ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} + \frac{1}{m}F}$$

x – position, $\dot{x}$ – speed/velocity, $\ddot{x}$ – acceleration

c - damping constant, m - mass, k - spring constant, F – force

In order to use the ODEs in MATLAB we need reformulate a higher order system into a system of first order differential equations

1. ODEs Solver

Higher Order ODEs

Mass-Spring-Damper System:

$$\boxed{\ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} + \frac{1}{m}F}$$

In order to use the ODEs in MATLAB we need reformulate a higher order system into a system of first order differential equations

We set:

$$\boxed{\begin{aligned} x_1 &= x \\ x_2 &= \dot{x} = \dot{x}_1 \end{aligned}}$$

This gives:

$$\boxed{\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{x} \end{aligned}}$$

Finally:

$$\boxed{\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}F \end{aligned}}$$

Now we are ready to solve the system using MATLAB

1. ODEs Solver

Step 1: Define the differential equation as a MATLAB function (`mass_spring_damper_diff.m`):

```
function dx = mass_spring_damper_diff(t,x)

k = 1;
m = 5;
c = 1;
F = 1;

dx = zeros(2,1); %Initialization

dx(1) = x(2);
dx(2) = -(k/m)*x(1)-(c/m)*x(2)+(1/m)*F;
```

Students: Try with different values for k, m, c and F

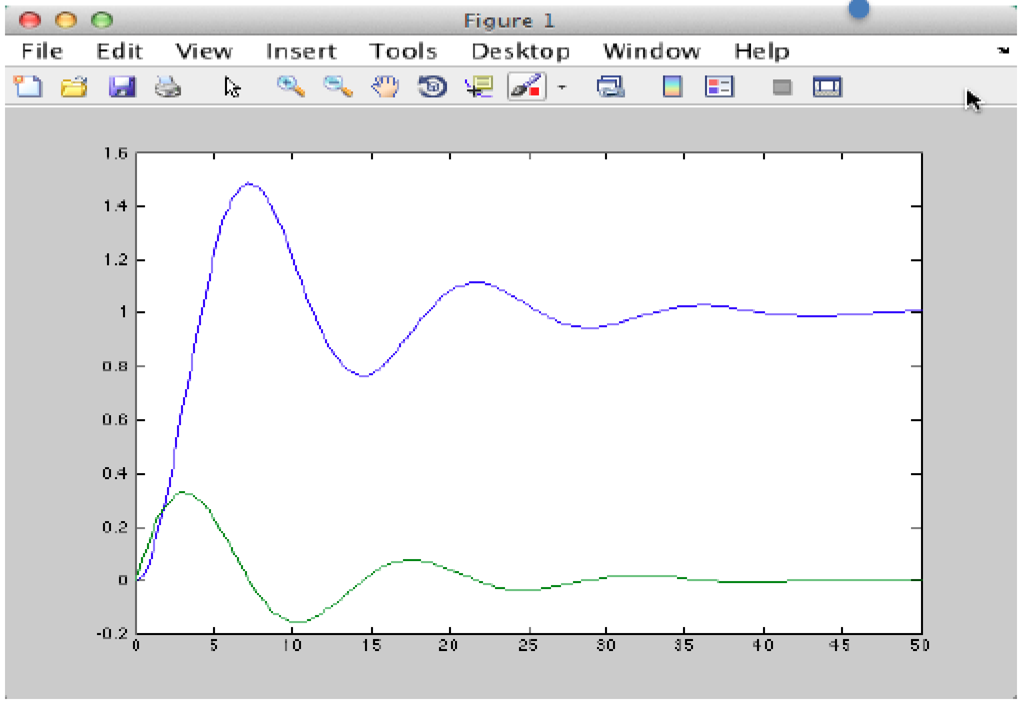
Students: Try this example

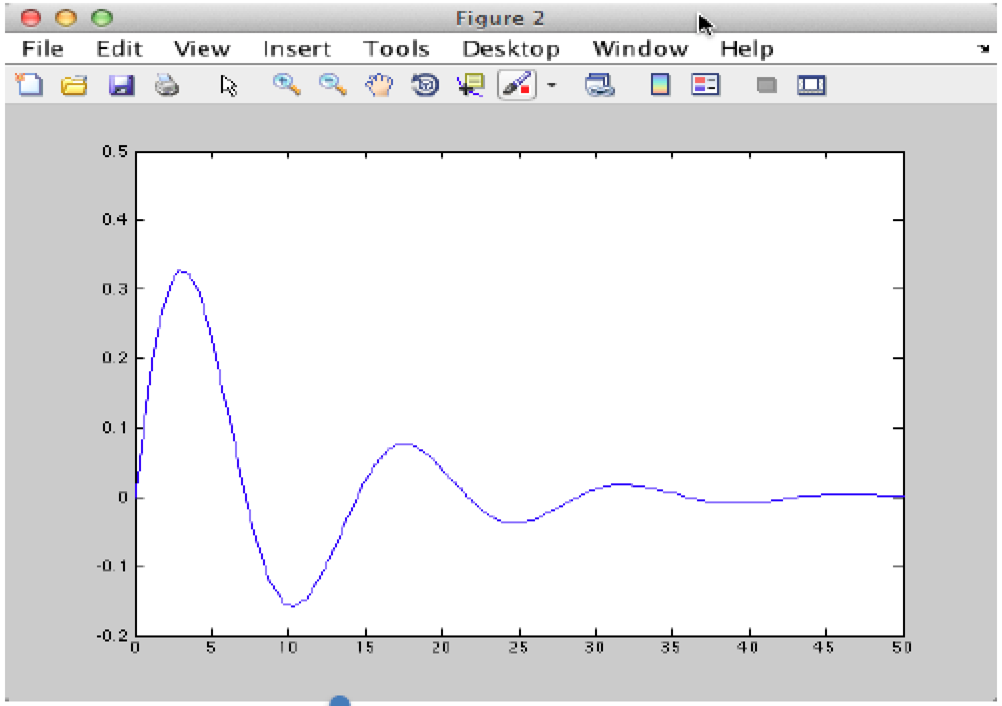
Step 2: Use the built-in ODE solver in a script.

```
clear
clc

tspan = [0 50];
x0 = [0;0];

[t,x] = ode23(@mass_spring_damper_diff,tspan,x0);
plot(t,x)
```

1. ODEs Solver

```
...
[t,x] = ode23(@mass_spring_damper_diff,tspan,x0);
plot(t,x)
```



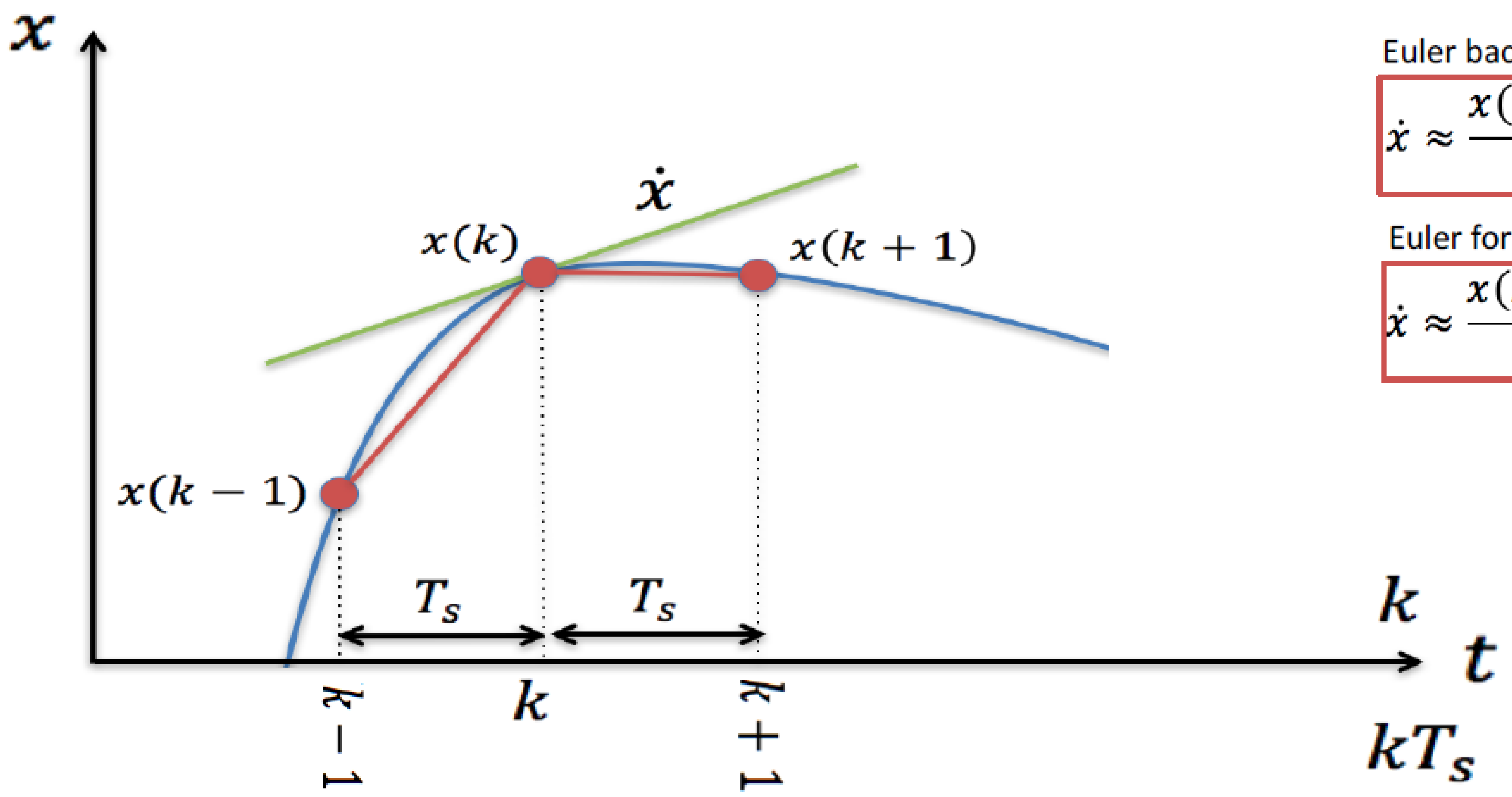



```
...
[t,x] = ode23(@mass_spring_damper_diff,tspan,x0);
plot(t,x(:,2))
```

2. Runge-Kutta method

Discrete Systems

Discrete Approximation of the time derivative



Euler backward method:

$$\dot{x} \approx \frac{x(k) - x(k-1)}{T_s}$$

Euler forward method:

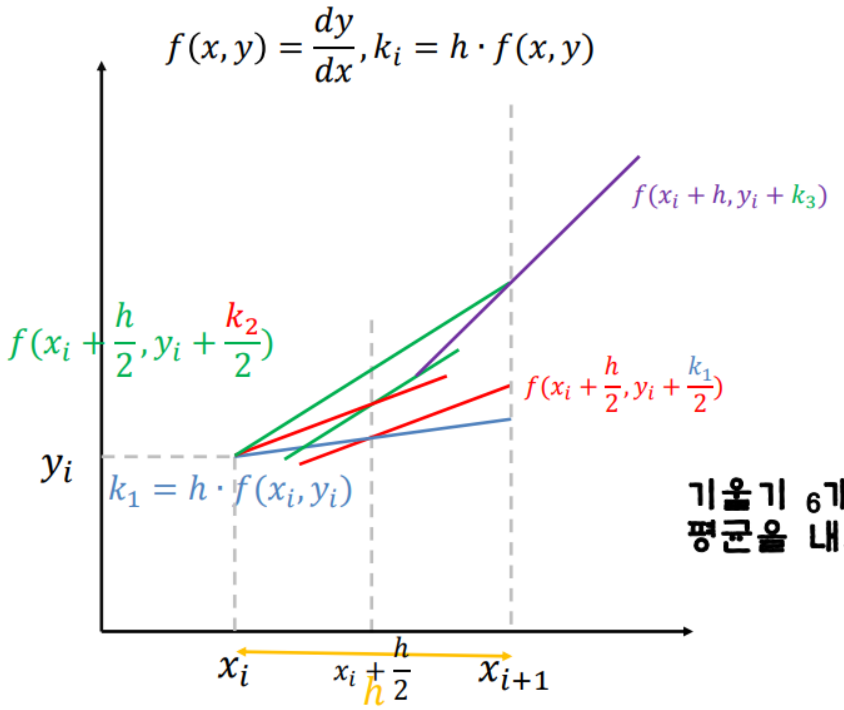
$$\dot{x} \approx \frac{x(k+1) - x(k)}{T_s}$$

2. Runge-Kutta method

Runge-Kutta method:

A method of numerically integrating ordinary differential equations by using a trial step at the midpoint of an interval to cancel out lower-order error terms.

4th order Runge-Kutta



기울기 6개를 가지고
평균을 내서 구함!

$$k_1 = f(x_n, y_n)$$

$$k_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$

$$k_4 = f(x_n + h, y_n + hk_3)$$

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

code

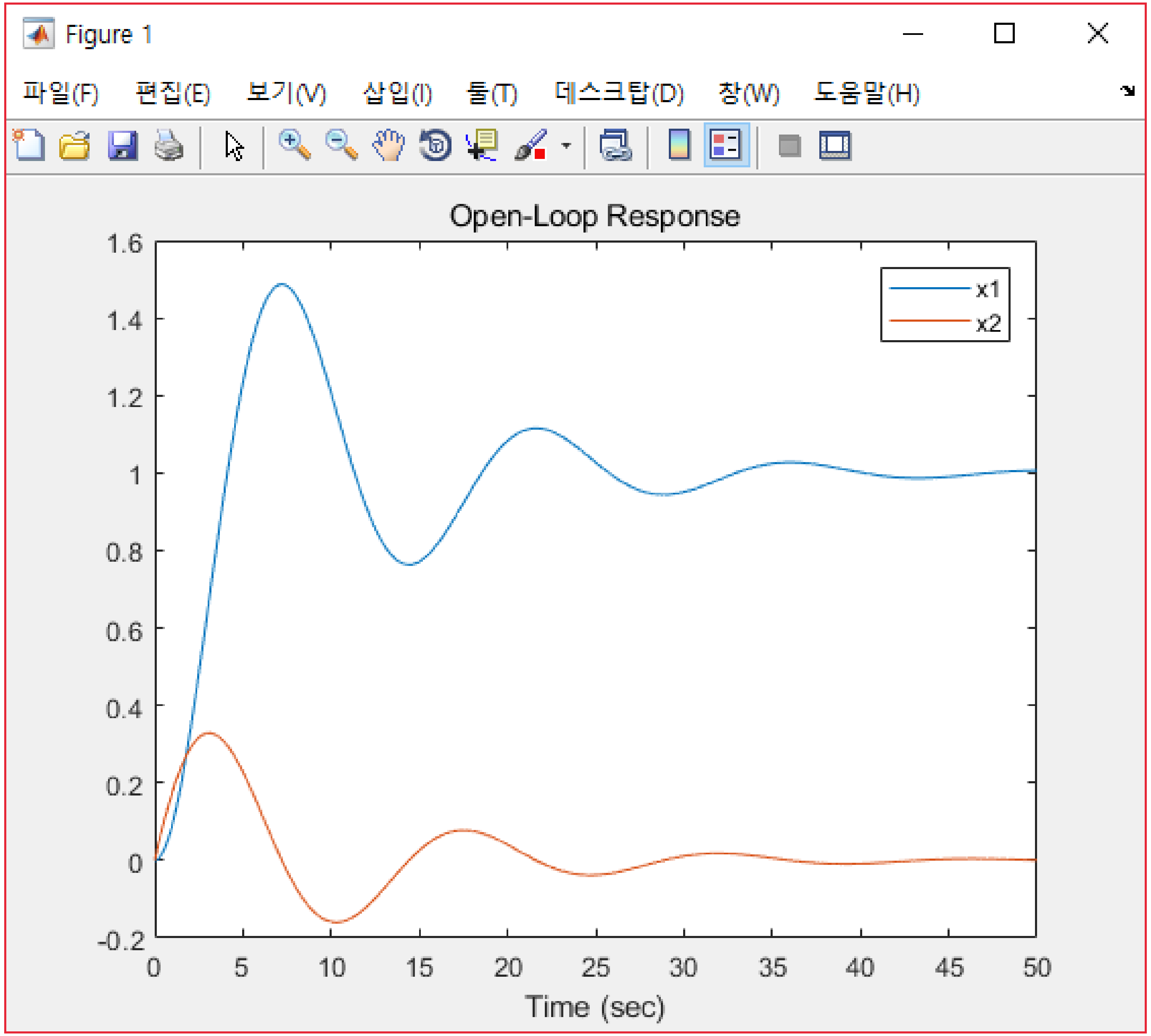
```
function dx=rk(x,u,T)
k1 = f(x,u)*T;
k2 = f(x+k1*0.5,u)*T;
k3 =f(x+k2*0.5,u)*T;
k4 = f(x+k3,u)*T;
dx=x + ((k1+k4)/6+(k2+k3)/3);
```

2. Runge-Kutta method

<table>
<tr><td rowspan="2">Runge-Kutta
2nd order</td><td>$$y_{i+1} = y_i + (1-b)k_1 + k_2 \quad (b \neq 0)$$</td></tr>
<tr><td>$k_1 = h \cdot f(x_i, y_i)$
$k_2 = h \cdot f\left(x_i + \frac{h}{2b}, y_i + \frac{k_1}{2b}\right)$</td></tr>
<tr><td rowspan="2">Runge-Kutta
3rd order</td><td>$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 4k_2 + k_3)$$</td></tr>
<tr><td>$k_1 = h \cdot f(x_i, y_i)$
$k_2 = h \cdot f\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$
$k_3 = h \cdot f(x_i + h, y_i - 2k_1 + 2k_2)$</td></tr>
<tr><td rowspan="2">Runge-Kutta
4th order</td><td>$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$</td></tr>
<tr><td>$k_1 = h \cdot f(x_i, y_i)$
$k_2 = h \cdot f\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$
$k_3 = h \cdot f\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$
$k_4 = h \cdot f(x_i + h, y_i + k_3)$</td></tr>
<tr><td colspan="2">$$f(x, y) = \frac{dy}{dx}$$</td></tr>
</table>

2. Runge-Kutta method

```
clear all;
clc;
k=1;
m=5;
c=1;
F=1;
A = [0 1;-k/m -c/m];
B = [0;1/m];
C = [1 0];
t = 0:0.01:50;
u = zeros(size(t));
u(:) = F;
x0 = [0 0];
sys = ss(A,B,C,0);
[y,t,x] = lsim(sys,u,t,x0);
figure(1)
plot(t,x)
title('Open-Loop Response')
legend('x1','x2')
xlabel('Time (sec)')
```



2. Runge-Kutta method

Main.m

```
X(:,1) = [0;0];
Tf=50;
Ti=0.01;
t=0:Ti:Tf;
sample_size = size(t,2);
F = 1;
U = F;
for i=1:sample_size-1
    X(:,i+1) = rk(X(:,i), U,Ti);
end
figure(2)
plot(t,X(1,:))
hold on;
plot(t,X(2,:))
hold off;
```

rk.m %Runge-Kutta

```
function dx=rk(x,u,T)
k1 = plant(x,u)*T;
k2 = plant(x+k1*0.5,u)*T;
k3 =plant(x+k2*0.5,u)*T;
k4 = plant(x+k3,u)*T;
dx=x + ((k1+k4)/6+(k2+k3)/3);
```

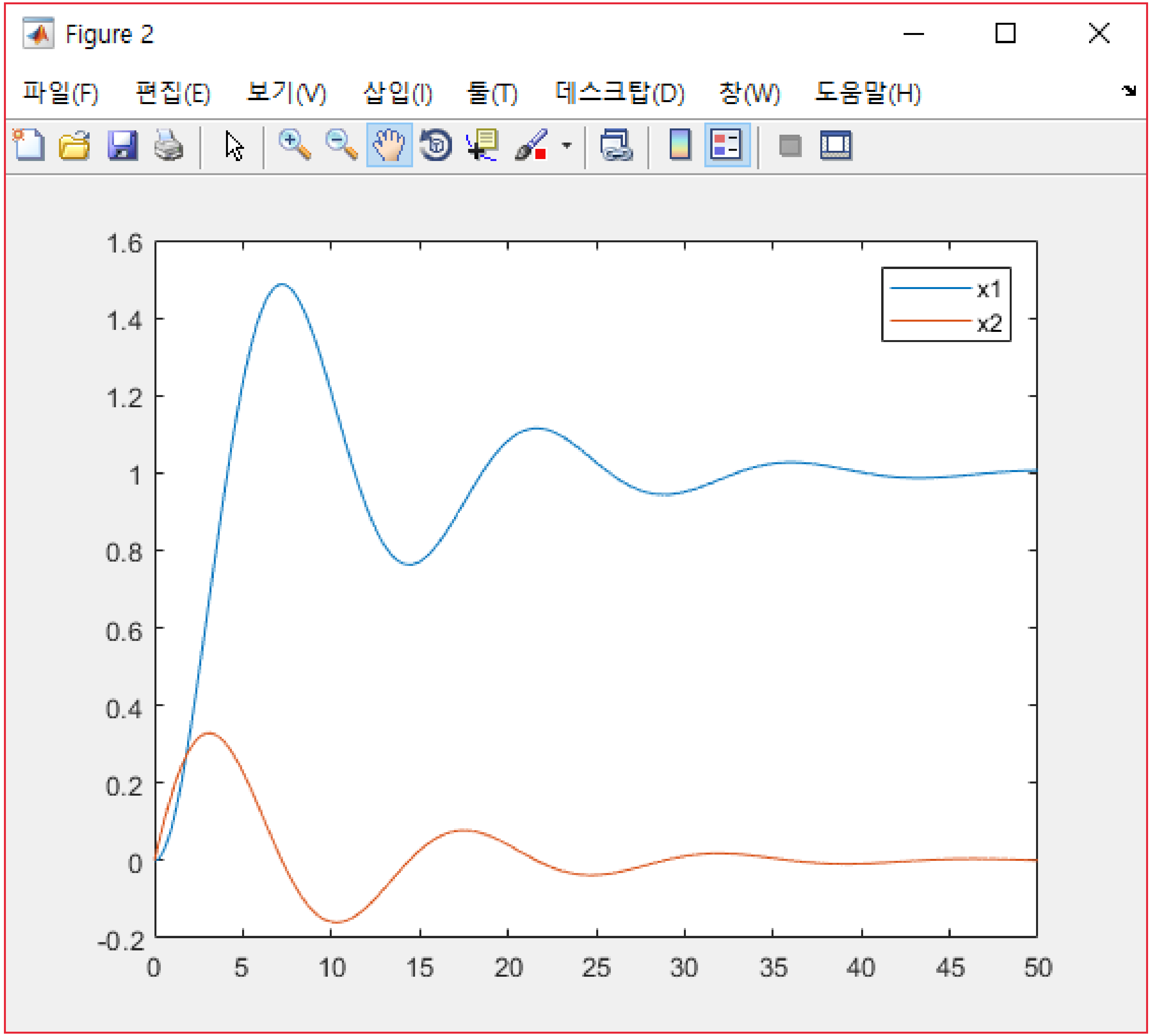
plant.m %System

```
function dx=plant(x,u)
k=1;
m=5;
c=1;
dx(1,1) = x(2);
dx(2,1) = -(c/m)*x(2)-(k/m)*x(1)+(1/m)*u;
```

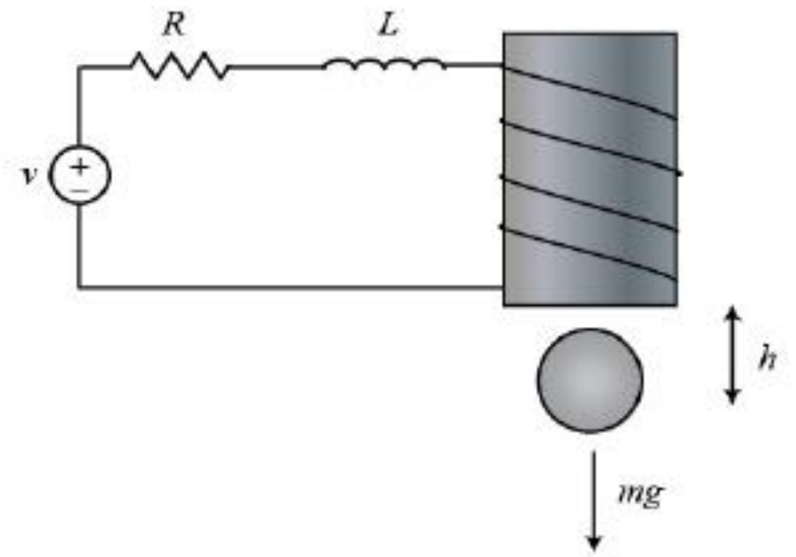
$$\dot{x}_1 = x_2$$

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

2. Runge-Kutta method



3. State feedback control



Magnetically suspended ball

$$M\frac{d^2h}{dt^2} = Mg - \frac{Ki^2}{h}$$

$$V = L\frac{di}{dt} + iR$$

where $M = 0.05Kg, K = 0.0001, L = 0.01H, R = 1Ohm, g = 9.81m/sec^2$.
The system is at equilibrium (the ball is suspended in midair) whenever $h = Ki^2/Mg$ (at which point dh/dt = 0). We linearize the equations about the point $h = 0.01m$ (where the nominal current is about 7 amp) and get the state space equations:

$$\dot{x} = Ax + Bu$$

y = Cx

```
A = [  0  1     0
     980  0  -2.8
       0  0  -100];
B = [0
     0
     100];
C = [1   0   0];
```

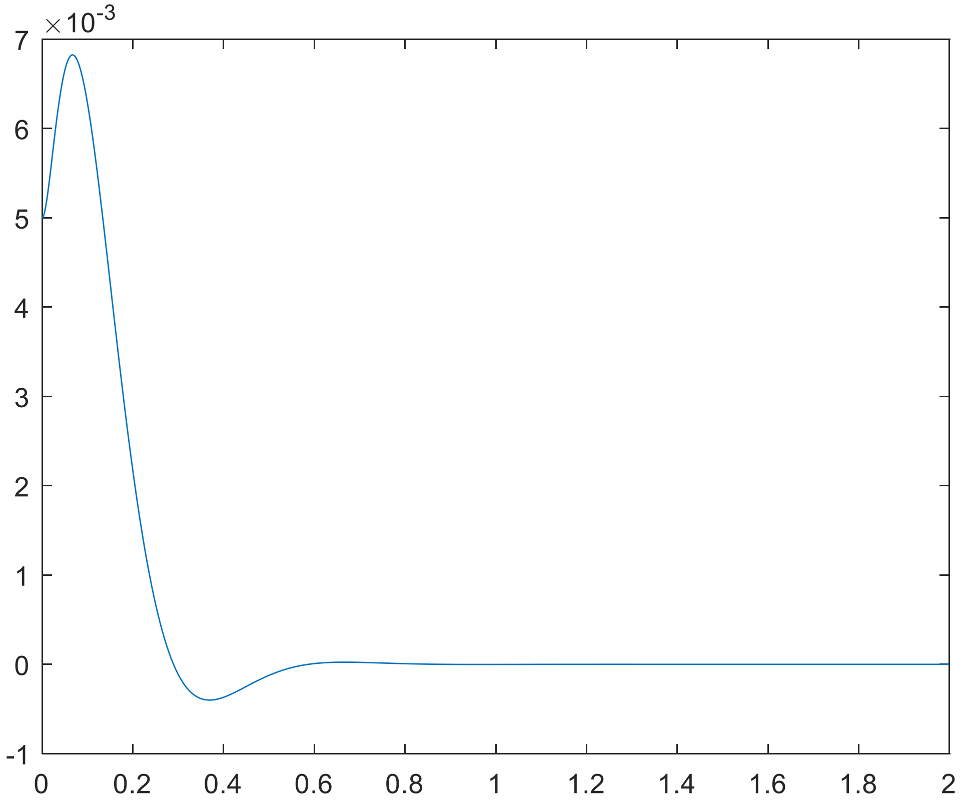
Linear system simulation test

```
A = [ 0 1 0;980 0 -2.8;0 0 -100];
B = [0 0 100]';
C = [1 0 0];
t = 0:0.01:2;
u = zeros(size(t));
x0 = [0.01 0 0]';
sys = ss(A,B,C,0);
[y,t,x] = lsim(sys,u,t,x0);
plot(t,y)
title('Open-Loop Response to Non-Zero
Initial Condition')
xlabel('Time (sec)')
ylabel('Ball Position (m)')
```

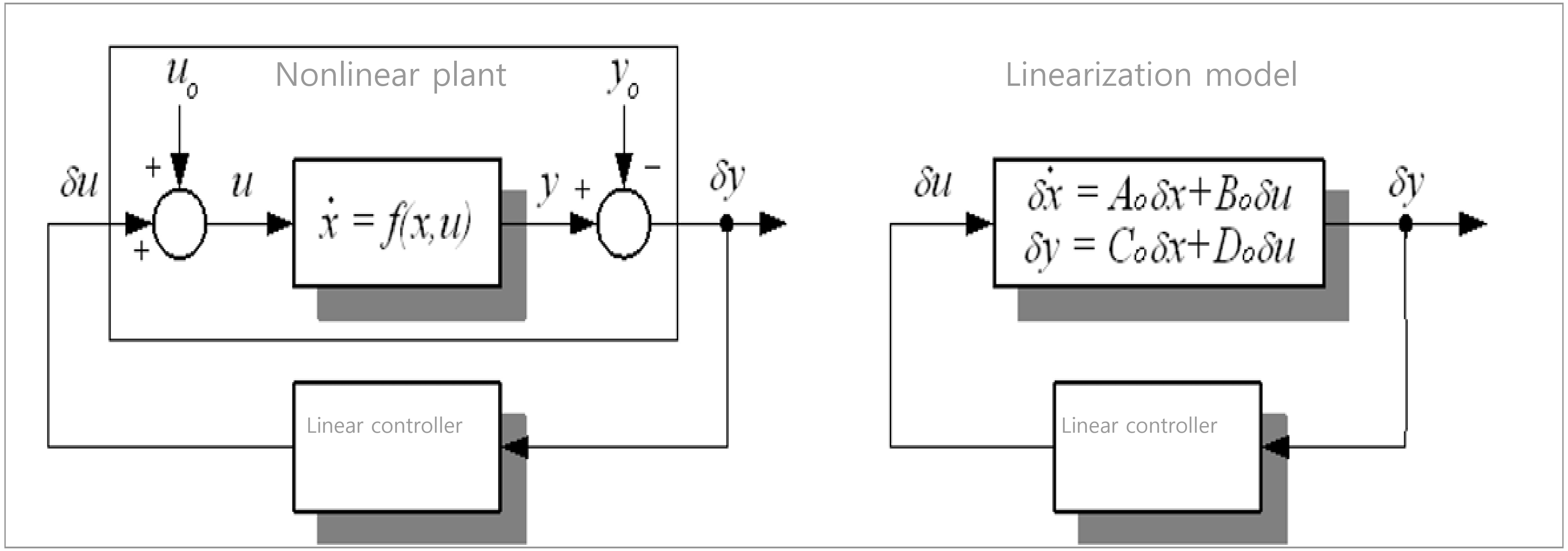
Linear system simulation test

```
%%% PROGRAM
%%% Plant equation %%%
function dx=plant(x,u)
dx(1,1) = x(2);
dx(2,1) = 980*x(1)-2.8*x(3);
dx(3,1) = -100*x(3)+100*u;
end
%%% Differential Eq. solution
using Runge-Kutta Method %%%
function dx=rk5(x,u,T)
k1=plant(x,u)*T;
k2=plant(x+k1*0.5,u)*T;
k3=plant(x+k2*0.5,u)*T;
k4=plant(x+k3,u)*T;
dx=x +((k1+k4)/6+(k2+k3)/3);
end
```

```
%%% Main program %%%
clear
A = [ 0 1 0; 980 0 -2.8; 0 0 -100];
B = [0 0 100]';
C = [1 0 0];
p1 = -10 + 10i;
p2 = -10 - 10i;
p3 = -50;
K = place(A,B,[p1 p2 p3]);
X(:,1)=[0.005 0 0]';
Tf=2;
Ti=0.001;
t=0:Ti:Tf;
sample_size = size(t,2);
for i=1:sample_size-1
U=-K*X(:,i);
X(:,i+1) = rk5(X(:,i),U,Ti);
end;
plot(t,X(1,:))
```


×10^-3
7
6
5
4
3
2
1
0
-1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Linear control for nonlinear systems



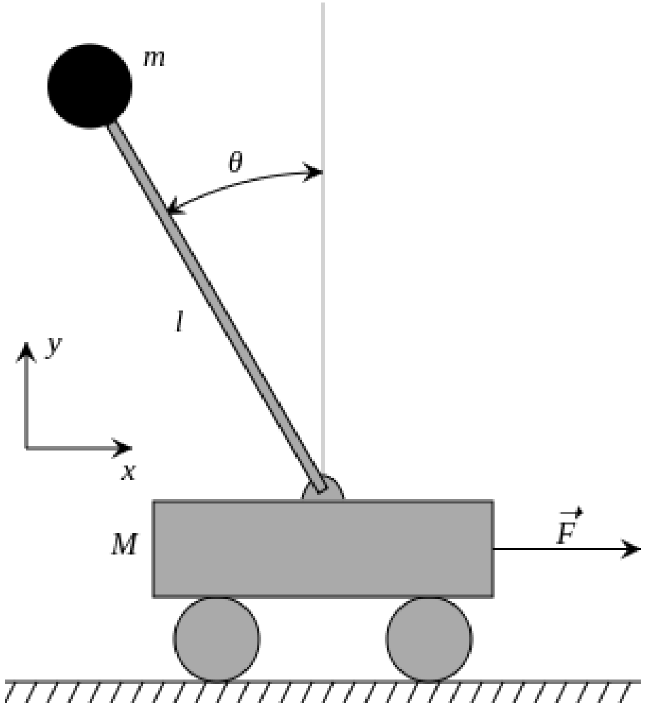
3. State feedback control

main.m

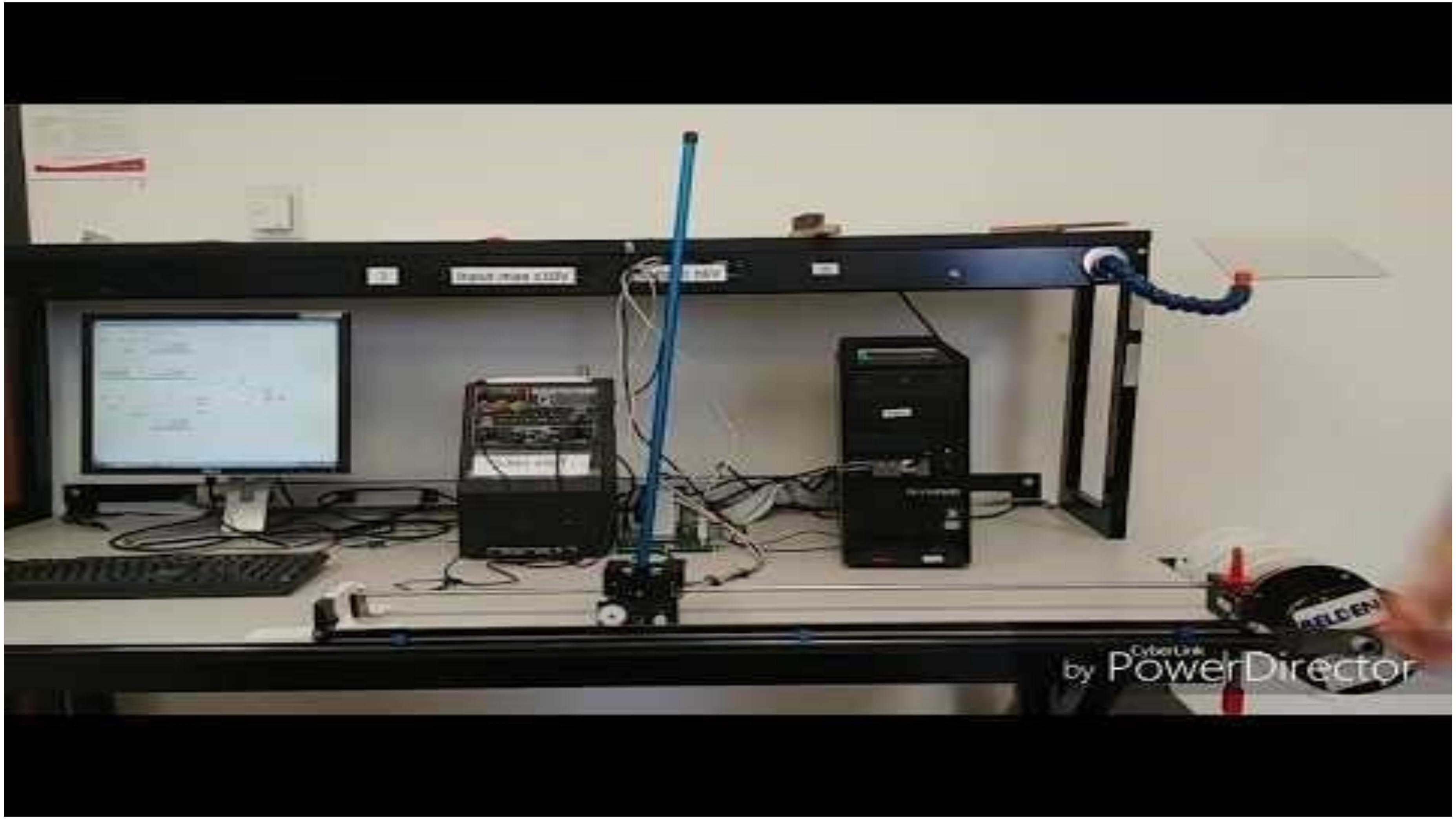
```
A = [ 0 1 0; 980 0 -2.8; 0 0 -100];
B = [0 0 100]';
C = [1 0 0];
p1 = -30 + 10i;
p2 = -30 - 10i;
p3 = -100;
K = place(A,B,[p1 p2 p3]);
X(:,1)=[-0.01 0 0]';
Tf=2;
Ti=0.001;
t=0:Ti:Tf;
sample_size = size(t,2);
Ut=[];
u0=7;
for i=1:sample_size-1
U=u0-K*(X(:,i)-[0.01 0 7]');
X(:,i+1) = rk5(X(:,i),U,Ti);
Ut=[Ut U];
end;
figure(1)
plot(t,X(1,:))
```

```
function dx=plant(x,u)
%if x(1) == 0
%     x(1) = 0.0001;
%end
M=0.05;
K=0.0001;
g=9.81;
dx(1,1) = x(2);
dx(2,1) = g-K/M*x(3)^2/x(1);
%dx(2,1) = 980*x(1)-2.8*x(3);
dx(3,1) = -100*x(3)+100*u;

%%% Differential Eq. solution
using Runge-Kutta Method %%%
function dx=rk5(x,u,T)
k1=plant(x,u)*T;
k2=plant(x+k1*0.5,u)*T;
k3=plant(x+k2*0.5,u)*T;
k4=plant(x+k3,u)*T;
dx=x +((k1+k4)/6+(k2+k3)/3);
```

4. Inverted pendulum control → HW#1



$$
\begin{aligned}
\dot{x} &= f(x) + g(x)u \\
f(x) &= \begin{bmatrix} x_2 \\ \frac{sin(x_3)x_4^2 - g*sin(x_3)cos(x_3)}{2-cos(x_3)^2} \\ x_4 \\ \frac{-sin(x_3)cos(x_3)x_4^2 + 2g*sin(x_3)}{2-cos(x_3)^2} \end{bmatrix} \\
g(x) &= \begin{bmatrix} 0 \\ \frac{1}{2-cos(x_3)^2} \\ 0 \\ \frac{-cos(x_3)}{2-cos(x_3)^2} \end{bmatrix} \\
where \quad g &= 9.8m/s^2
\end{aligned}
$$


CyberLink
by PowerDirector

1. Plot the time response.$(0 \leq t \leq 5)$
   Set the initial condition $x_1 = x_2 = x_4 = u = 0, \quad x_3 = 0.2rad.$
   (Hint: Use Runge-kutta method to solve the differential equation)

2. The Equilibrium Point is $x_1 = x_2 = x_3 = x_4 = u = 0.$
   Is the equilibrium point stable point?

3. Linearlize the nonlinear system at the equilibrium point. (Hint : syms, jacobian(x,y))

   Find A, B matrices.

   $$\dot{x} = Ax + Bu \tag{7}$$

   Example
   ?syms u v
   ?jacobian(u*exp(v),[u;v])

4. Design the controller using the linearlized model. Simulate to erect the pendulum using the nonlinear model.
   Initial condition is $x_1 = x_2 = x_4 = 0, x_3 = 0.2$ (Hint: acker)

Linearization using jacobian

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, \cdots, x_n, u) \\ f_2(x_1, x_2, \cdots, x_n, u) \\ \vdots \\ f_n(x_1, x_2, \cdots, x_n, u) \end{bmatrix}$$

$$\begin{aligned} \delta\dot{x}(t) &\approx A_0 \delta x(t) + B_0 \delta u(t) \\ \delta y(t) &\approx C_0 \delta x(t) + D_0 \delta u(t) \end{aligned}$$

$$A_0 = \frac{\partial}{\partial x} f(x_0, u_0), \quad B_0 = \frac{\partial}{\partial u} f(x_0, u_0)$$

$$C_0 = \frac{\partial}{\partial x} g(x_0, u_0), \quad D_0 = \frac{\partial}{\partial u} g(x_0, u_0)$$

$$\delta u = u - u_0, \ \delta x = x - x_0, \ \delta y = y - y_0$$

```
syms X1 X2 X3 X4 U;
fx=[X2;
(sin(X3)*X4^2-9.8*sin(X3)*cos(X3))/(2-
(cos(X3))^2)+1/(2-(cos(X3))^2)*U;
X4;
(-sin(X3)*cos(X3)*X4^2+2*9.8*sin(X3))/(2-
(cos(X3))^2)+(-cos(X3))/(2-(cos(X3))^2)*U];
jacobian(fx,[X1 X2 X3 X4 U]);
```

```
clear
global A B;
A = [0 1 0 0;0 0 -9.8 0;
0 0 0 1;0 0 19.6 0];
B = [0 1 0 -1]';
C = [1 0 0 0];
% p1 = -1 + i;
% p2 = -1 - i;
% p3 = -2;
% p4 = -5;
p1=-10;
p2=-60;
p3=-10+0.4i;
p4=-10-0.4i;

K = place(A,B,[p1 p2 p3 p4]);
Q=10*eye(4);R=1;
[K,S,E] = lqr(A,B,Q,R);
```

```
X(:,1)=[0 0 0.5 0]';

Tf=10;
Ti=0.01;

t=0:Ti:Tf;

sample_size = size(t,2);

for i=1:sample_size-1
   U=-K*X(:,i);
   X(:,i+1) = rk6(X(:,i),U,Ti);
  end;

plot(t,X(:,:)')
```

```
function DX =rk6(X, U, T)

k1 = plant1(X, U)*T;
k2 = plant1(X + k1 * 0.5, U)*T;
k3 = plant1(X + k2 * 0.5, U)*T;
k4 = plant1(X + k3, U)*T;
DX = X + ((k1 + k4)/6.0 + (k2 + k3) / 3.0);


function DX = plant1(X,U)
%global A B;
DX = X;
DX(1) = X(2);
DX(2)=(sin(X(3))*X(4)^2-9.8*sin(X(3))*cos(X(3)))/(2-
(cos(X(3)))^2)+1/(2-(cos(X(3)))^2)*U;
DX(3) = X(4);
DX(4)=(-sin(X(3))*cos(X(3))*X(4)^2+2*9.8*sin(X(3)))/(2-
(cos(X(3)))^2)+(-cos(X(3)))/(2-(cos(X(3)))^2)*U;

%DX = A*X+B*U;
```