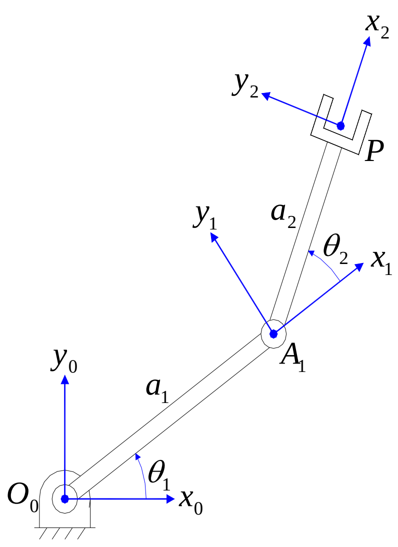
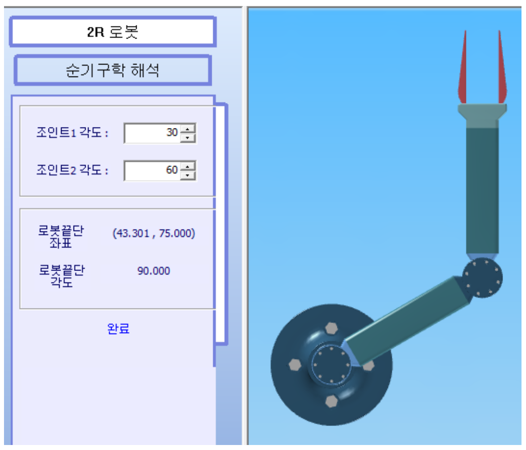
CH. 1: Analysis and MATLAB Practice of Planar Robots

[1] Position Analysis and MATLAB Practice

(1) 2-DOF serial robot





조인트1 각도 : 30

조인트2 각도 : 60

로봇끝단 좌표 (43.301 , 75.000)

로봇끝단 각도 90.000

Fig. 1: Planar 2-DOF serial robot.

1) D-H Parameters

Joint i	α_i	a_i	d_i	θ_i
1	0	a_1	0	θ_1
2	0	a_2	0	θ_2

- Link lengths: $a_1 = 100$, $a_2 = 100$ mm

- Homogeneous transformation matrix:

(Formula) ${}^{i-1}A_i = \text{Trans}(z_{i-1}, d_i)\text{Rot}(z_{i-1}, \theta_i)\text{Trans}(x_i, a_i)\text{Rot}(x_i, \alpha_i)$

$${}^{0}A_1 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & a_1c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & a_1s\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^{1}A_2 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_2c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$ {}^0A_2 = {}^0A_1\,{}^1A_2 = \begin{bmatrix} c\theta_{12} & -c\theta_{12} & 0 & a_1c\theta_1 + a_2c\theta_{12} \\ s\theta_{12} & s\theta_{12} & 0 & a_1s\theta_1 + a_2s\theta_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

2) Forward kinematics

$$ \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = {}^0A_2 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1c\theta_1 + a_2c\theta_{12} \\ a_1s\theta_1 + a_2s\theta_{12} \\ 0 \\ 1 \end{bmatrix} \tag{2} $$

3) Inverse kinematics

$$ {}^0A_2 = \begin{bmatrix} c\theta_{12} & -c\theta_{12} & 0 & a_1c\theta_1 + a_2c\theta_{12} \\ s\theta_{12} & s\theta_{12} & 0 & a_1s\theta_1 + a_2s\theta_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\phi & -s\phi & 0 & p_x \\ s\phi & c\phi & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

- From (1,4) and (2,4) elements of Eq. (3),

$$ p_x = a_1c\theta_1 + a_2c\theta_{12} \tag{4} $$

$$ p_y = a_1s\theta_1 + a_2s\theta_{12} \tag{5} $$

- θ_1 can be eliminated by taking the square of Eq. (4) and Eq. (5) and taking the sum.

$$ p_x^2 + p_y^2 = a_1^2 + a_2^2 + 2a_1a_2c\theta_2 \tag{6} $$

- Solving Eq. (6) for θ_2 gives

$$ \theta_2 = \cos^{-1}\kappa \tag{7} $$

where, $\kappa = \dfrac{p_x^2 + p_y^2 - a_1^2 - a_2^2}{2a_1a_2}$.

- The solution of Eq. (7) can be classified into three cases. (1) two distinct real roots $|\kappa| < 1$, (2) one double root $|\kappa| = 1$, (3) no real roots $|\kappa| > 1$. As shown in Fig. 2, there are two solutions, θ_2^* and $-\theta_2^*$ ($\pi \geq \theta_2^* \geq 0$). When $\theta_2 = \theta_2^*$, the solution is called elbow-down configuration. When $\theta_2 = -\theta_2^*$, it is called elbow-up configuration. When $|\kappa| = 1$, the arm is stretched-out or stretched –in. When $|\kappa| > 1$, it is out of workspace.
- Once θ_2 is obtained, θ_1 can be determined by expanding Eq. (4) and Eq. (5) as follow.

$$(a_1 + a_2 c\theta_2)c\theta_1 - (a_2 s\theta_2)s\theta_1 = p_x \tag{8}$$

$$(a_2 s\theta_2)c\theta_1 + (a_1 + a_2 c\theta_2)s\theta_1 = p_y \tag{9}$$

- Solving Eq. (8) and Eq. (9) for $c\theta_1$ and $s\theta_1$ yields

$$\begin{aligned} c\theta_1 &= \frac{(a_1 + a_2 c\theta_2)p_x + a_2 s\theta_2 p_y}{\Delta} \\ s\theta_1 &= \frac{-a_2 s\theta_2 p_x + (a_1 + a_2 c\theta_2)p_y}{\Delta} \end{aligned} \tag{10}$$

where $\Delta = a_1^2 + a_2^2 + 2a_1a_2c\theta_2$.

- A single solution of θ_1 for given θ_2 is obtained by

$$\theta_1 = \text{Atan2}(s\theta_1, c\theta_1) \tag{11}$$

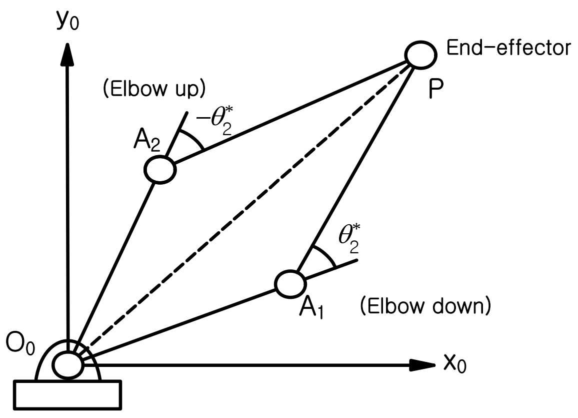


Fig. 2: Two possible inverse kinematics solutions.

4) MATLAB Example

[skm_2dof_cal.m]

```
% Main Function %
function []=skm_2dof_cal()
clc;
D2R=pi/180; R2D=180/pi;
a1= 100; a2=100;
% [alpha, a, d] %
param=[0,a1,0; 0,a2,0];

% (1) Forward Kinematics %
th=D2R*[0,90]';
T02=forward_kinematics(th, param);
pos=T02(1:2,4);

disp('(1) Forward Kinematics:');
disp('Joint angles [deg]:');
disp(R2D*th');
disp('Position [mm]:');
disp(pos');

% (2) Inverse Kinematics %
pos=[100, 100]';
[th]=inverse_kinematics(pos, param);
disp('(2) Inverse Kinematics:');
disp('Position [mm]:');
disp(pos');
disp('Joint angles [deg]:');
disp(R2D*th');

% Forward & Inverse Kinematics %
%-------- Forward Kinematics
function T=forward_kinematics(th, param)
T=Transformation(th(1,1), param(1,:))*Transformation(th(2,1), param(2,:));
%-------- Inverse Kinematics
function [th, w_index]=inverse_kinematics(pos, param)
w_index=1;    % In workspace %
a1=param(1,2); a2=param(2,2);

% (1) Calculate th2 %
px=pos(1,1); py=pos(2,1);
kapha=(px^2+py^2-a1^2-a2^2)/(2*a1*a2);
if abs(kapha)>1
       w_index=-1;    % Out of Workspace %
       kapha=1;
end
th2=+acos(kapha); % Elbow Down %
%th2=-acos(kapha); % Elbow Up %

% (2) Calculate th1 %
delta=a1^2+a2^2+2*a1*a2*cos(th2);
```

```
cth1=(px*(a1+a2*cos(th2))+py*a2*sin(th2))/delta;
sth1=(-px*a2*sin(th2)+py*(a1+a2*cos(th2)))/delta;
th1=atan2(sth1, cth1);
th=[th1, th2]';

% Homogeneous Transformation Matrix %
%-------- D-H Transformation Matrix
function [T]=Transformation(th, param)
alpha=param(1); a=param(2); d=param(3);
T=Trans([0,0,d]')*Rotz(th)*Trans([a,0,0]')*Rotx(alpha);
%------- Rotation matrix about the X-axis
function [T]=Rotx(th)
cx=cos(th); sx=sin(th);
T=[1,0,0,0; 0,cx,-sx,0; 0,sx,cx,0; 0,0,0,1];
%------- Rotation matrix about the Z-axis
function [T]=Rotz(th)
cz=cos(th); sz=sin(th);
T=[cz,-sz,0,0; sz,cz,0,0; 0,0,1,0; 0,0,0,1];
%-------- Translation along the x, y, z axes
function [T]=Trans(p)
T=[1,0,0,p(1,1); 0,1,0,p(2,1); 0,0,1,p(3,1); 0,0,0,1];
```

[실행결과]

```
(1) Forward Kinematics:
Joint angles [deg]:
         0        90
Position [mm]:
      100       100
(2) Inverse Kinematics:
Position [mm]:
      100       100
Joint angles [deg]:
         0        90
```

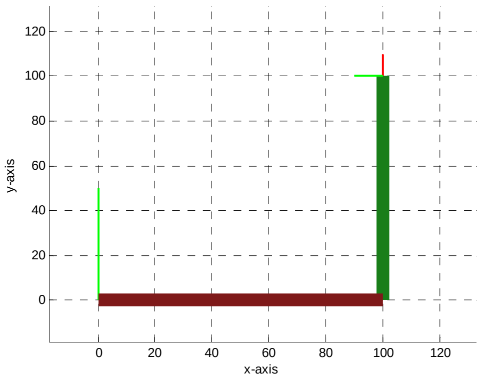
[skm_2dof.m & skm_2dof_vrml]

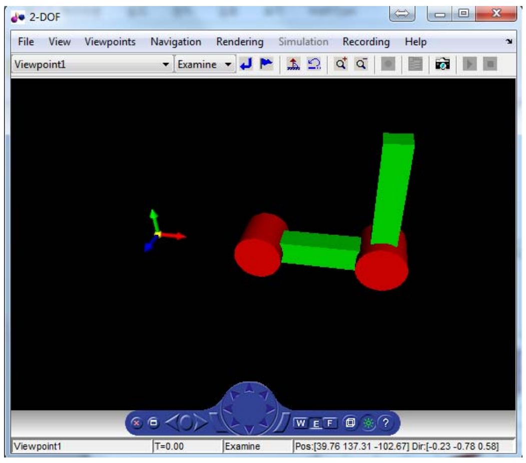
```
%% Main Function %%
function skm_2dof

%------- Figure settings
figure('KeyPressFcn',@key_callback);

%------- Figure Settings
%view(125,25); axis square
axis([-200, 200, -200, 200]);
xlabel('x-axis'); ylabel('y-axis');
grid on;

%% Keyboard Interrrupt %%
```

```
function key_callback(obj,event)
cla;                                            % Erase the current figure %
persistent pos ang th

D2R=pi/180; R2D=180/pi;
dpos=5; dth=5*D2R;
a1= 100; a2=100;
param=[0,a1,0; 0,a2,0];

if isempty(pos), pos=[a1,a2]'; end
if isempty(ang), ang=90*D2R; end
if isempty(th), th=[0,90*D2R]'; end

key=event.Character;
switch key
    case '1', th(1,1)=th(1,1)+dth; index=1;          % +th1 Movement %
    case 'q', th(1,1)=th(1,1)-dth; index=1;          % -th1 Movement %
    case '2', th(2,1)=th(2,1)+dth; index=1;          % +th2 Movement %
    case 'w', th(2,1)=th(2,1)-dth; index=1;          % -th2 Movement %

    case 'a', pos(1,1)=pos(1,1)+dpos; index=2;       % +X Movement %
    case 'z', pos(1,1)=pos(1,1)-dpos; index=2;       % -X Movement %
    case 's', pos(2,1)=pos(2,1)+dpos; index=2;       % +Y Movement %
    case 'x', pos(2,1)=pos(2,1)-dpos; index=2;       % -Y Movement %

    case 'p', pos=[a1,a2]'; th=[0,90*D2R]'; index=1;    % Original Pos & Orien %
    otherwise
        disp('Not Proper Key'); index=1;
end

% Draw the Fixed Frame %
sc_f=50; draw_frame(eye(4,4), sc_f);            % Fixed Frame %

switch index
    case 1, % Forward Kinematics %
        T02=forward_kinematics(th, param);
        pos=T02(1:2,4);
    case 2, % Inverse Kinematics %
        [th_c, w_index]=inverse_kinematics(pos, param);
        if w_index==1, th=th_c; end
end

% Draw a 2-DOF Planar Robot %
draw_2dof_robot(th, param);

clc;
disp('px[mm], py[mm]');
disp(pos');
disp('th1[deg], th2[deg]');
disp(R2D*th');
disp('Transformation Matrix');
T02=forward_kinematics(th,param);
disp(T02);

%% Forward & Inverse Kinematics
```

```
%-------- Forward Kinematics
function T=forward_kinematics(th, param)
T=Transformation(th(1,1), param(1,:))*Transformation(th(2,1), param(2,:));

%-------- Inverse Kinematics
function [th, w_index]=inverse_kinematics(pos, param)
w_index=1;    % In workspace %
a1=param(1,2); a2=param(2,2);

% (1) Calculate th2 %
px=pos(1,1); py=pos(2,1);
kapha=(px^2+py^2-a1^2-a2^2)/(2*a1*a2);
if abs(kapha)>1
        w_index=-1;    % Out of Workspace %
        kapha=1;
end
th2=+acos(kapha); % Elbow Down %
%th2=-acos(kapha); % Elbow Up %

% (2) Calculate th1 %
delta=a1^2+a2^2+2*a1*a2*cos(th2);
cth1=(px*(a1+a2*cos(th2))+py*a2*sin(th2))/delta;
sth1=(-px*a2*sin(th2)+py*(a1+a2*cos(th2)))/delta;
th1=atan2(sth1, cth1);
th=[th1, th2]';

%-------- D-H Transformation Matrix
function [T]=Transformation(th, param)
alpha=param(1); a=param(2); d=param(3);
T=Trans([0,0,d]')*Rotz(th)*Trans([a,0,0]')*Rotx(alpha);

%% Draw Primitives %%
%-------- Draw a 3-DOF Planar Robot
function draw_2dof_robot(th, param)
% Draw the First Link %
T01=Transformation(th(1,1), param(1,:));
p1=[0,0,0]'; p2=T01(1:3,4);
lw=10; clr=[0.5,0.1,0.1];
draw_line(p1,p2,lw,clr);

% Draw the Second Link %
T02=T01*Transformation(th(2,1), param(2,:));
p1=T01(1:3,4); p2=T02(1:3,4);
lw=10; clr=[0.1,0.5,0.1];
draw_line(p1,p2,lw,clr);

% Draw the Moving Frame & Box %
sc_f=50; draw_frame(T02, sc_f/5);
sc_b=5; draw_box(T02, sc_b);

%-------- Draw a Frame
function draw_frame(T, sc)
p1=T*[0,0,0,1]'; p2=T*[sc,0,0,1]'; lw=2; clr=[1,0,0];
draw_line(p1,p2,lw,clr);
p1=T*[0,0,0,1]'; p2=T*[0,sc,0,1]'; lw=2; clr=[0,1,0];
```

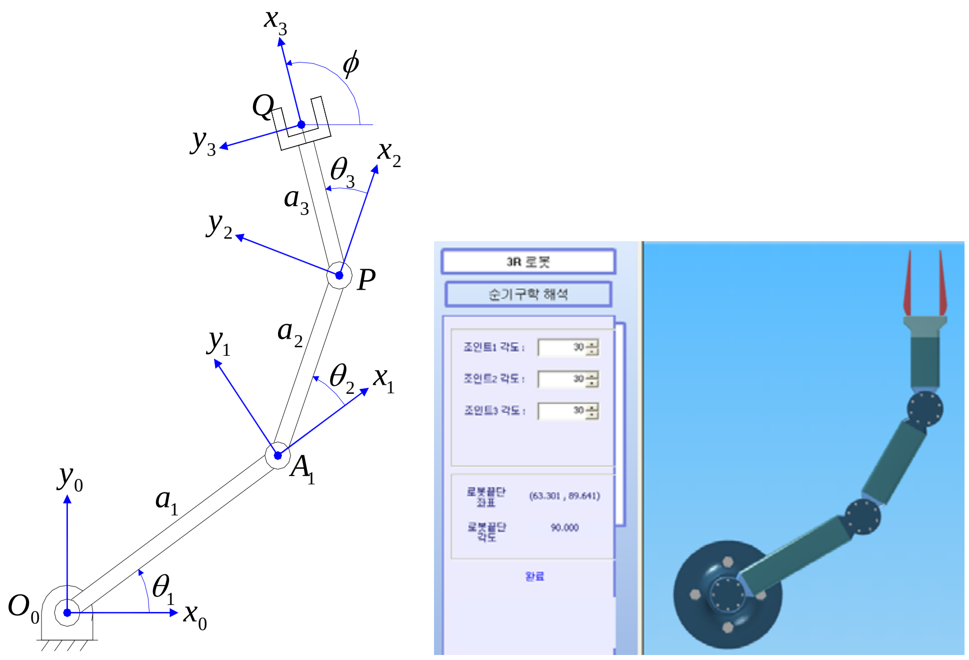
```
draw_line(p1,p2,lw,clr);
p1=T*[0,0,0,1]'; p2=T*[0,0,sc,1]'; lw=2; clr=[0,0,1];
draw_line(p1,p2,lw,clr);

%------- Draw a Line
function draw_line(p1, p2, lw, clr)
line([p1(1,1),p2(1,1)],[p1(2,1),p2(2,1)],[p1(3,1),p2(3,1)],'LineWidth',lw,'Color',clr);

%------ Draw a Box %%% Multifaceted Patches
function draw_box(T, sc)
vt0=[0,0,0,1; sc,0,0,1; sc,sc,0,1; 0,sc,0,1; 0,0,sc,1; sc,0,sc,1; sc,sc,sc,1; 0,sc,sc,1]';
fm=[1,2,6,5; 2,3,7,6; 3,4,8,7; 4,1,5,8; 1,2,3,4; 5,6,7,8];
vt1=T*vt0; vm=vt1(1:3,:)';
patch('Vertices',vm,'Faces',fm,...
        'FaceVertexCData',gray(8),'FaceColor','flat');

%% Homogeneous Transformation Matrix %%
%------- Rotation matrix about the X-axis
function [T]=Rotx(th)
cx=cos(th); sx=sin(th);
T=[1,0,0,0; 0,cx,-sx,0; 0,sx,cx,0; 0,0,0,1];

%------- Rotation matrix about the Y-axis
function [T]=Roty(th)
cy=cos(th); sy=sin(th);
T=[cy,0,sy,0; 0,1,0,0; -sy,0,cy,0; 0,0,0,1];

%------- Rotation matrix about the Z-axis
function [T]=Rotz(th)
cz=cos(th); sz=sin(th);
T=[cz,-sz,0,0; sz,cz,0,0; 0,0,1,0; 0,0,0,1];

%-------- Translation along the x, y, z axes
function [T]=Trans(p)
T=[1,0,0,p(1,1); 0,1,0,p(2,1); 0,0,1,p(3,1); 0,0,0,1];
```





Viewpoint1 | T=0.00 | Examine | Pos:[39.76 137.31 -102.67] Dir:[-0.23 -0.78 0.58]

(2) 3-DOF serial robot



3R 로봇

순기구학 해석

조인트1 각도 : 30

조인트2 각도 : 30

조인트3 각도 : 30

로봇끝단 좌표 (63.301 , 89.641)

로봇끝단 각도 90.000

완료

Fig. 3: Planar 3-DOF serial robot.

1) D-H Parameters

Joint i	α_i	a_i	d_i	θ_i
1	0	a_1	0	θ_1
2	0	a_2	0	θ_2
3	0	a_3	0	θ_3

- Link lengths: $a_1 = 100$, $a_2 = 70$, $a_3 = 50$ mm

- Homogeneous transformation matrix:

$$
{}^0A_1 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & a_1 c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & a_1 s\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^1A_2 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_2 c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2 s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$ {}^2A_3 = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_3c\theta_3 \\ s\theta_{31} & c\theta_3 & 0 & a_3s\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

$$ {}^0A_3 = {}^0A_1\,{}^1A_2\,{}^2A_3 = \begin{bmatrix} c\theta_{123} & -c\theta_{123} & 0 & a_1c\theta_1 + a_2c\theta_{12} + a_3c\theta_{123} \\ s\theta_{123} & s\theta_{123} & 0 & a_1s\theta_1 + a_2s\theta_{12} + a_3s\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

2) Forward kinematics

$$ \begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = {}^0A_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1c\theta_1 + a_2c\theta_{12} + a_3c\theta_{123} \\ a_1s\theta_1 + a_2s\theta_{12} + a_3s\theta_{123} \\ 0 \\ 1 \end{bmatrix} \tag{2} $$

3) Inverse kinematics

$$ {}^0A_3 = \begin{bmatrix} c\theta_{123} & -c\theta_{123} & 0 & a_1c\theta_1 + a_2c\theta_{12} + a_3c\theta_{123} \\ s\theta_{123} & s\theta_{123} & 0 & a_1s\theta_1 + a_2s\theta_{12} + a_3s\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\phi & -s\phi & 0 & q_x \\ s\phi & c\phi & 0 & q_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3} $$

Since $c\theta_{123} = c\phi$ and $s\theta_{123} = s\phi$,

$$ \theta_{123} = \theta_1 + \theta_2 + \theta_3 = \phi \tag{4} $$

When $p_x = q_x - a_3c\phi$ and $p_y = q_y - a_3s\phi$ are calculated, the 3-DOF serial robot problem is reduced to that of the 2-DOF serial robot.

$$ p_x = a_1c\theta_1 + a_2c\theta_{12} \tag{5} $$

$$ p_y = a_1s\theta_1 + a_2s\theta_{12} \tag{6} $$

4) MATLAB Example

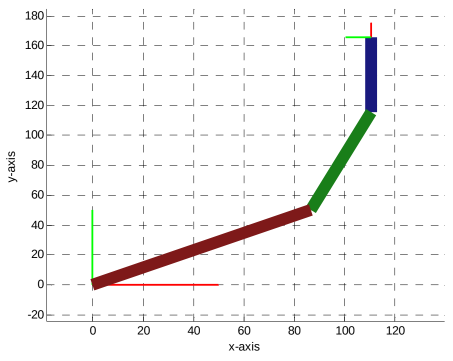
[skm_3dof.m & skm_3dof_vrml]

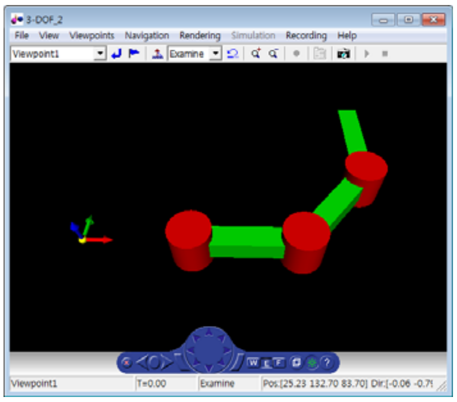
```
%% Main Function %%
function skm_3dof

%------- Figure settings
figure('KeyPressFcn',@key_callback);

%------- Figure Settings
%view(125,25); axis square
axis([-200, 200, -200, 200]);
xlabel('x-axis'); ylabel('y-axis');
grid on;

%% Keyboard Interrrupt %%
function key_callback(obj,event)
cla;                                    % Erase the current figure %
persistent pos ang th

D2R=pi/180; R2D=180/pi;
dpos=5; dth=5*D2R;
a1= 100; a2=70; a3=50;
param=[0,a1,0; 0,a2,0;0,a3,0];

if isempty(pos), pos=[a1,a2+a3]'; end
if isempty(ang), ang=90*D2R; end
if isempty(th), th=[0,90*D2R,0]'; end

key=event.Character;
switch key
    case '1', th(1,1)=th(1,1)+dth; index=1;         % +th1 Movement %
    case 'q', th(1,1)=th(1,1)-dth; index=1;         % -th1 Movement %
    case '2', th(2,1)=th(2,1)+dth; index=1;         % +th2 Movement %
    case 'w', th(2,1)=th(2,1)-dth; index=1;         % -th2 Movement %
    case '3', th(3,1)=th(3,1)+dth; index=1;         % +th3 Movement %
    case 'e', th(3,1)=th(3,1)-dth; index=1;         % -th3 Movement %

    case 'a', pos(1,1)=pos(1,1)+dpos; index=2;        % +X Movement %
    case 'z', pos(1,1)=pos(1,1)-dpos; index=2;        % -X Movement %
    case 's', pos(2,1)=pos(2,1)+dpos; index=2;        % +Y Movement %
    case 'x', pos(2,1)=pos(2,1)-dpos; index=2;        % -Y Movement %
    case 'd', ang=ang+dth; index=2;                     % +ANG Movement %
    case 'c', ang=ang-dth; index=2;                     % -ANG Movement %

    case 'p', pos=[a1,a2+a3]'; ang=90*D2R; th=[0,90*D2R,0]'; index=1;     % Original Pos & Orien %
    otherwise
        disp('Not Proper Key'); index=1;
end

% Draw the Fixed Frame %
sc_f=50; draw_frame(eye(4,4), sc_f);                % Fixed Frame %
```

```
switch index
       case 1, % Forward Kinematics %
              T03=forward_kinematics(th, param);
              pos=T03(1:2,4); ang=atan2(T03(2,1),T03(1,1));
       case 2, % Inverse Kinematics %
              [th_c, w_index]=inverse_kinematics(pos, ang,param);
              if w_index==1, th=th_c; end
end

% Draw a 3-DOF Planar Robot %
sc_r=100; draw_3dof_robot(th, param, sc_r);


clc;
disp('px[mm], py[mm], ang[deg]');
disp([pos(1,1), pos(2,1), ang*R2D]);
disp('th1[deg], th2[deg], th3[deg]');
disp([th(1,1), th(2,1), th(3,1)]*R2D);
disp('Transformation Matrix');
T03=forward_kinematics(th,param);
disp(T03);

%% Forward & Inverse Kinematics
%-------- Forward Kinematics
function T=forward_kinematics(th, param)
T=Transformation(th(1,1), param(1,:))*Transformation(th(2,1), param(2,:))*Transformation(th(3,1),
param(3,:));

%-------- Inverse Kinematics
function [th, w_index]=inverse_kinematics(pos, ang, param)
w_index=1;    % In workspace %
a1=param(1,2); a2=param(2,2); a3=param(3,2);

% (0) Calculate T03 %
T03=Trans([pos(1,1),pos(2,1),0]')*Rotz(ang);

% (1) Calculate th2 %
qx=T03(1,4); qy=T03(2,4);
cphi=T03(1,1); sphi=T03(2,1);
px=qx-a3*cphi; py=qy-a3*sphi;
kapha=(px^2+py^2-a1^2-a2^2)/(2*a1*a2);
if abs(kapha)>1
       w_index=-1;    % Out of Workspace %
       kapha=1;
end
th2=+acos(kapha); % Elbow Down %
%th2=-acos(kapha); % Elbow Up %

% (2) Calculate th1 %
delta=a1^2+a2^2+2*a1*a2*cos(th2);
cth1=(px*(a1+a2*cos(th2))+py*a2*sin(th2))/delta;
sth1=(-px*a2*sin(th2)+py*(a1+a2*cos(th2)))/delta;
th1=atan2(sth1, cth1);

% (3) Calcuate th3 %
phi=atan2(sphi, cphi);
```

```
th3=phi-(th1+th2);

th=[th1, th2, th3]';

%-------- D-H Transformation Matrix
function [T]=Transformation(th, param)
alpha=param(1); a=param(2); d=param(3);
T=Trans([0,0,d]')*Rotz(th)*Trans([a,0,0]')*Rotx(alpha);

%% Draw Primitives %%
%-------- Draw a 3-DOF Planar Robot
function draw_3dof_robot(th, param, sc)
% Draw the First Link %
T01=Transformation(th(1,1), param(1,:));
p1=[0,0,0]'; p2=T01(1:3,4);
lw=10; clr=[0.5,0.1,0.1];
draw_line(p1,p2,lw,clr);

% Draw the Second Link %
T02=T01*Transformation(th(2,1), param(2,:));
p1=T01(1:3,4); p2=T02(1:3,4);
lw=10; clr=[0.1,0.5,0.1];
draw_line(p1,p2,lw,clr);

% Draw the Third Link %
T03=T02*Transformation(th(3,1), param(3,:));
p1=T02(1:3,4); p2=T03(1:3,4);
lw=10; clr=[0.1,0.1,0.5];
draw_line(p1,p2,lw,clr);

% Draw the Moving Frame & Box %
sc_f=50; draw_frame(T03, sc_f/5);
sc_b=5; draw_box(T03, sc_b);

%-------- Draw a Frame
function draw_frame(T, sc)
p1=T*[0,0,0,1]'; p2=T*[sc,0,0,1]'; lw=2; clr=[1,0,0];
draw_line(p1,p2,lw,clr);
p1=T*[0,0,0,1]'; p2=T*[0,sc,0,1]'; lw=2; clr=[0,1,0];
draw_line(p1,p2,lw,clr);
p1=T*[0,0,0,1]'; p2=T*[0,0,sc,1]'; lw=2; clr=[0,0,1];
draw_line(p1,p2,lw,clr);

%------- Draw a Line
function draw_line(p1, p2, lw, clr)
line([p1(1,1),p2(1,1)],[p1(2,1),p2(2,1)],[p1(3,1),p2(3,1)],'LineWidth',lw,'Color',clr);

%------ Draw a Box %%% Multifaceted Patches
function draw_box(T, sc)
vt0=[0,0,0,1; sc,0,0,1; sc,sc,0,1; 0,sc,0,1; 0,0,sc,1; sc,0,sc,1; sc,sc,sc,1; 0,sc,sc,1]';
fm=[1,2,6,5; 2,3,7,6; 3,4,8,7; 4,1,5,8; 1,2,3,4; 5,6,7,8];
vt1=T*vt0; vm=vt1(1:3,:)';
patch('Vertices',vm,'Faces',fm,...
          'FaceVertexCData',gray(8),'FaceColor','flat');
```

```
%% Homogeneous Transformation Matrix %%
%------- Rotation matrix about the X-axis
function [T]=Rotx(th)
cx=cos(th); sx=sin(th);
T=[1,0,0,0; 0,cx,-sx,0; 0,sx,cx,0; 0,0,0,1];

%------- Rotation matrix about the Y-axis
function [T]=Roty(th)
cy=cos(th); sy=sin(th);
T=[cy,0,sy,0; 0,1,0,0; -sy,0,cy,0; 0,0,0,1];

%------- Rotation matrix about the Z-axis
function [T]=Rotz(th)
cz=cos(th); sz=sin(th);
T=[cz,-sz,0,0; sz,cz,0,0; 0,0,1,0; 0,0,0,1];

%-------- Translation along the x, y, z axes
function [T]=Trans(p)
T=[1,0,0,p(1,1); 0,1,0,p(2,1); 0,0,1,p(3,1); 0,0,0,1];
```




3-DOF_2
File View Viewpoints Navigation Rendering Simulation Recording Help
Viewpoint1
Examine
Viewpoint1
T=0.00
Examine
Pos:[25.23 132.70 83.70] Dir:[-0.06 -0.7

(3) 5-bar robot

1) Inverse kinematics (Type I)

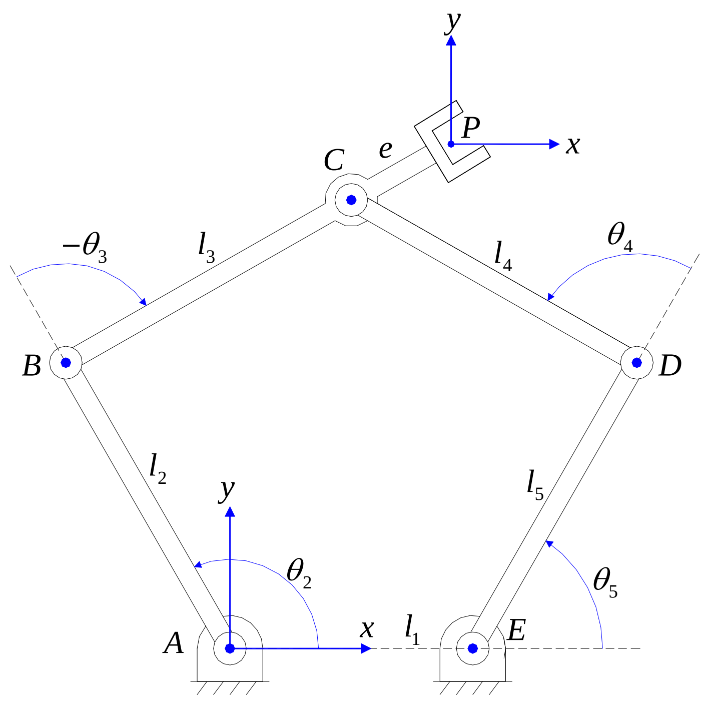


Fig. 4: Planar 5-bar (Type I) robot.

- Kinematic parameters: $l_1 = \overline{AE} = 74$, $l_2 = \overline{AB} = 100$, $l_3 = \overline{BP} = 100 + 28 = 128$ ($e = \overline{CP} = 28$ mm), $l_4 = \overline{CD} = 100$, $l_5 = \overline{DE} = 100$

- The left vector-loop of the mechanism is first analyzed. The end-effector position is given by

$$p_x = l_2 c\theta_2 + l_3 c\theta_{23} \tag{1}$$

$$p_y = l_2 s\theta_2 + l_3 s\theta_{23} \tag{2}$$

- Taking the sum of squares of Eq. (1) and Eq. (2), θ_2 can be eliminated.

$$p_x^2 + p_y^2 = l_2^2 + l_3^2 + 2l_2 l_3 c\theta_3 \tag{3}$$

- Solve Eq. (3) for θ_3 gives

$$\theta_3 = (\text{sign})\cos^{-1}\kappa_1 \quad \text{(Elbow up)} \tag{4}$$

where, $\kappa_1 = \dfrac{p_x^2 + p_y^2 - l_2^2 - l_3^2}{2l_2 l_3}$. If the home position is the elbow-down configuration, sign is positive. Otherwise, sign is negative. Here, the elbow-up configuration is selected, therefore, $\theta_3 = -\cos^{-1}\kappa_1$.

- θ_2 can be solved by expanding Eq. (1) and Eq. (2) with respect to θ_3 as follow.

$$(l_2 + l_3 c\theta_3)c\theta_2 - (l_3 s\theta_3)s\theta_2 = p_x \tag{5}$$

$$(l_3 s\theta_3)c\theta_2 + (l_2 + l_3 c\theta_3)s\theta_2 = p_y \tag{6}$$

- Solving Eq. (5) and Eq. (6) for $c\theta_2$ and $s\theta_2$ yields,

$$\begin{aligned} c\theta_2 &= \frac{(l_2 + l_3 c\theta_3)\,p_x + l_3 s\theta_2\, p_y}{\Delta_1} \\ s\theta_2 &= \frac{-l_3 s\theta_3\, p_x + (l_2 + l_3 c\theta_3)\,p_y}{\Delta_1} \end{aligned} \tag{7}$$

where $\Delta_1 = l_2^2 + l_3^2 + 2l_2 l_3 c\theta_3$ 이다.

- A single solution of θ_2 for given θ_3 is obtained by

$$\theta_2 = \text{Atan2}(s\theta_2, c\theta_2) \tag{8}$$

- Next, the right vector-loop of the mechanism is analyzed. The point C is calculated by

$$c_x = p_x - e\,c\theta_{23},\ c_y = p_y - e\,s\theta_{23} \tag{9}$$

- Calculating the right vector-loop gives

$$c_x = l_1 + l_5 c\theta_5 + l_4 c\theta_{54} \tag{10}$$

$$c_y = \quad l_5 s\theta_5 + l_4 s\theta_{54} \tag{11}$$

- Taking the square of Eq. (10) and Eq. (11) and taking the sum, θ_5 can be eliminated.

$$(c_x - l_1)^2 + c_y^2 = l_5^2 + l_4^2 + 2l_5 l_4 c\theta_4 \tag{12}$$

- Solve Eq. (12) for θ_4 yields,

$$\theta_4 = +\cos^{-1}\kappa_2 \quad \text{(Elbow down)} \tag{13}$$

where, $\kappa_2 = \dfrac{(c_x - l_1)^2 + c_y^2 - l_5^2 - l_6^2}{2l_5 l_6}$.

- θ_5 can be solved by expanding Eq. (10) and Eq. (11) with respect to θ_4 as follow.

$$(l_5 + l_4 c\theta_4)c\theta_5 - (l_4 s\theta_4)s\theta_5 = c_x - l_1 \tag{14}$$

$$(l_4 s\theta_4)c\theta_5 + (l_5 + l_4 c\theta_4)s\theta_5 = c_y \tag{15}$$

- Solving Eq. (14) and Eq. (15) for $c\theta_5$ and $s\theta_5$ yields,

$$\begin{aligned} c\theta_5 &= \frac{(l_5 + l_4 c\theta_4)(c_x - l_1) + l_4 s\theta_4 c_y}{\Delta_2} \\ s\theta_5 &= \frac{-l_4 s\theta_4 (c_x - l_1) + (l_5 + l_4 c\theta_4)c_y}{\Delta_2} \end{aligned} \tag{16}$$

where $\Delta_2 = l_5^2 + l_4^2 + 2l_5 l_4 c\theta_4$.

- A single solution of θ_5 for given θ_4 is obtained by

$$\theta_5 = \text{Atan2}(s\theta_5, c\theta_5) \tag{17}$$

2) Inverse kinematics (Type II)

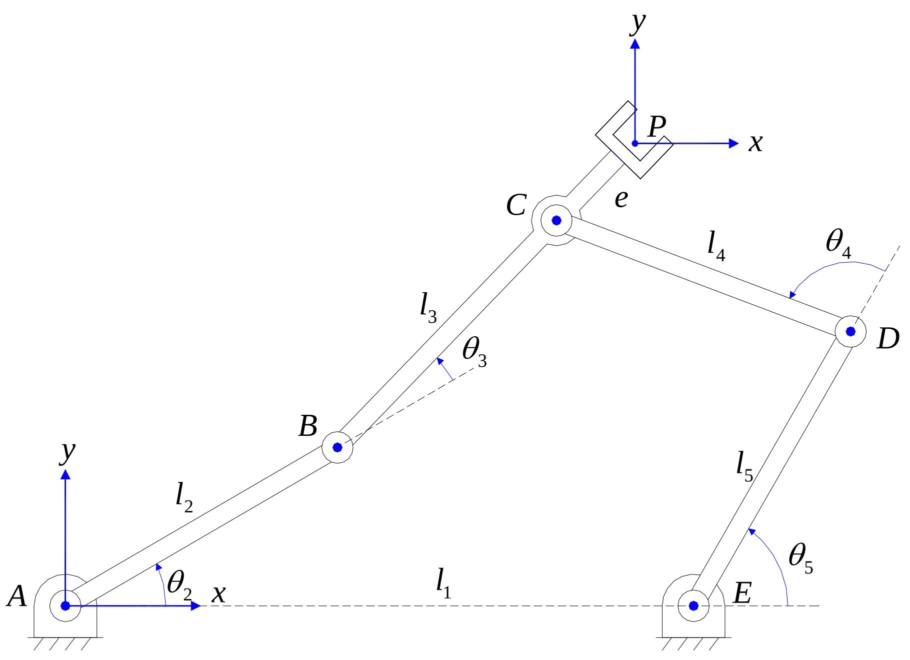
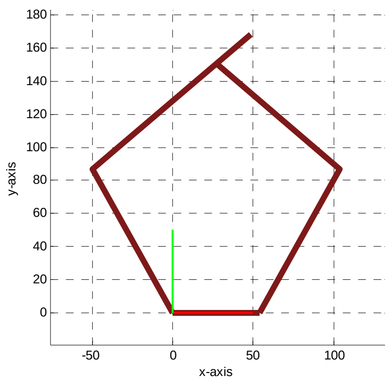


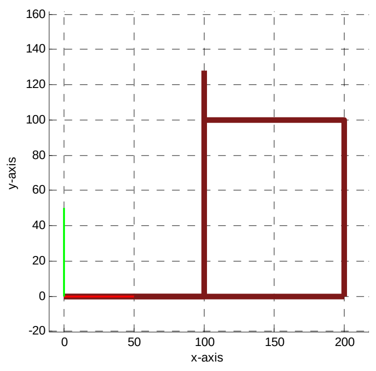
Fig. 5: Planar 5-bar (Type II) robot.

- Kinematic parameters: $l_1 = \overline{AE} = 200$, $l_2 = \overline{AB} = 100$, $l_3 = \overline{BP} = 100 + 28 = 128$ ($e = \overline{CP} = 28$ mm), $l_4 = \overline{CD} = 100$, $l_5 = \overline{DE} = 100$
- The analysis is the same with that of the 5-bar (Type I) robot except that the left vector-loop is the elbow-down configuration. θ_3 is determined by

$$\theta_3 = +\cos^{-1}\kappa_1 \quad \text{(Elbow down)} \tag{4'}$$

4) MATLAB Example





(4) 3-DOF parallel robot

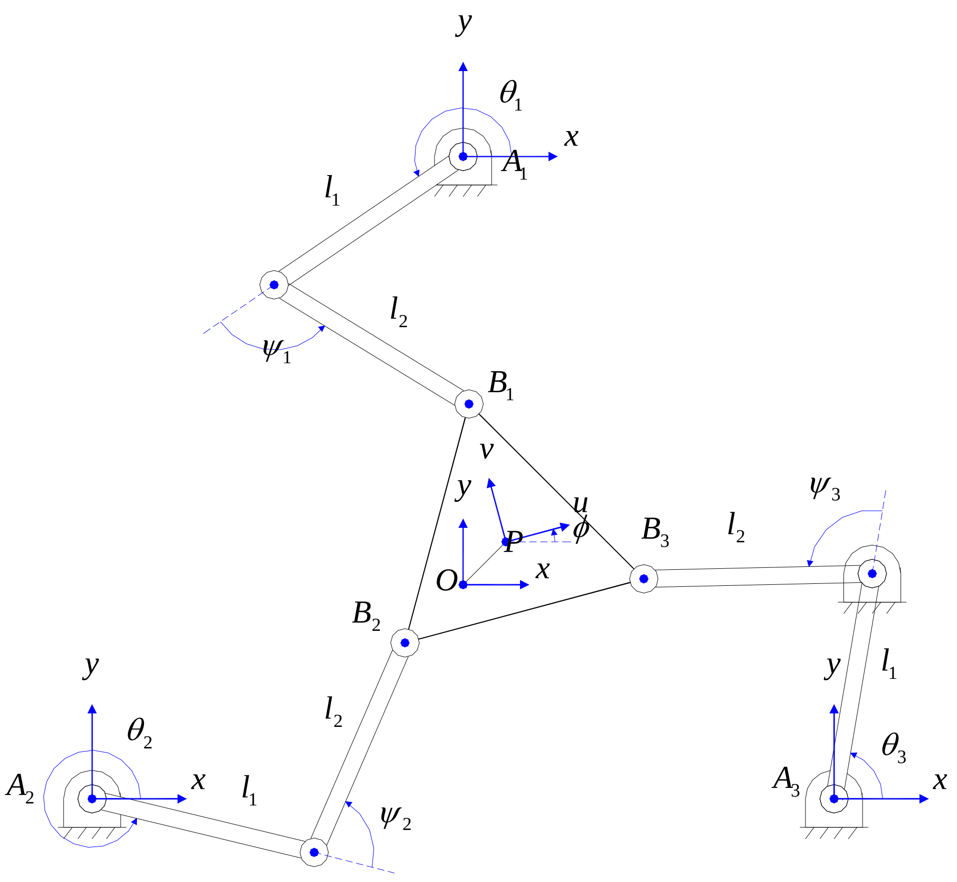


Fig. 6: Planar 3-DOF 3-RRR parallel robot.

- Kinematic parameters: $a = \overline{OA_i} = 150$ mm, $b = \overline{PB_i} = 35$ mm, $l_1 = l_2 = 100$ mm

- The reference and moving frames are defined by

  $\{O\}:\ O - xy,\ \{P\}:\ P - uv$

- The local frame of the *i*th leg is defined by

  $\{A_i\}:\ A_i - xy$

- The position vector of A_i in the reference frame is given by

$$\boldsymbol{a}_1 = r_a \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ \boldsymbol{a}_2 = r_a \begin{bmatrix} -\sqrt{3}/2 \\ -1/2 \end{bmatrix},\ \boldsymbol{a}_3 = r_a \begin{bmatrix} +\sqrt{3}/2 \\ -1/2 \end{bmatrix} \tag{1}$$

- The position vector of B_i in the moving frame is given by

$${}^P\boldsymbol{b}_1 = r_b \begin{bmatrix} 0 \\ 1 \end{bmatrix},\ {}^P\boldsymbol{b}_2 = r_b \begin{bmatrix} -\sqrt{3}/2 \\ -1/2 \end{bmatrix},\ {}^P\boldsymbol{b}_3 = r_b \begin{bmatrix} +\sqrt{3}/2 \\ -1/2 \end{bmatrix} \tag{2}$$

- The vector of $\boldsymbol{d}_i = \overrightarrow{A_i B_i}$ in the reference frame is written by

$$\boldsymbol{d}_i = \boldsymbol{p} + R\,{}^P\boldsymbol{b}_i - \boldsymbol{a}_i \tag{3}$$

where $\boldsymbol{p} = \overrightarrow{OP}$ and the rotation matrix is $R = \begin{bmatrix} c\phi & -s\phi \\ s\phi & c\phi \end{bmatrix}$.

- Expressing $\boldsymbol{d}_i = \overrightarrow{A_i B_i}$ in terms of joint angles gives

$$d_{xi} = l_1 c\theta_i + l_2 c(\theta_i + \psi_i) \tag{4}$$

$$d_{yi} = l_1 s\theta_i + l_2 s(\theta_i + \psi_i) \tag{5}$$

- Rewriting Eq. (4) and Eq. (5) yields

$$d_{xi} - l_1 c\theta_i = l_2 c(\theta_i + \psi_i) \tag{4'}$$

$$d_{yi} - l_1 s\theta_i = l_2 s(\theta_i + \psi_i) \tag{5'}$$

- Taking the sum of squares of Eq. (4’) and Eq. (5’), ψ_i can be eliminated by

$$e_{1i} s\theta_i + e_{2i} c\theta_i + e_{3i} = 0 \tag{6}$$

where,

$$e_{1i} = -2l_1 d_{yi}\,,$$

$$e_{2i} = -2l_1 d_{xi}\,,$$

$$e_{3i} = d_{xi}^2 + d_{yi}^2 + l_1^2 - l_2^2 \tag{7}$$

- Eq. (7) can be written by

$$(e_{3i} - e_{2i})t_i^2 + 2e_{1i}t_i + (e_{3i} + e_{2i}) = 0 \tag{8}$$

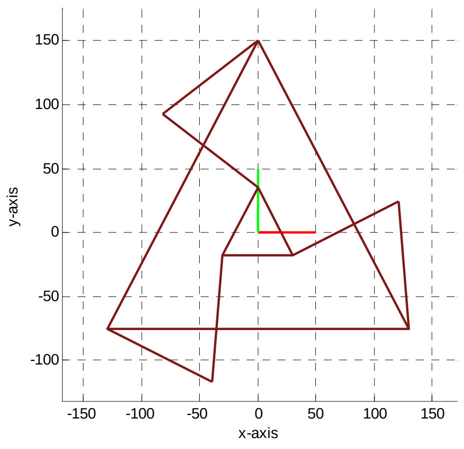
where, $t_i = \tan\dfrac{\theta_i}{2}$.

- Solving Eq. (8) for θ_i gives

$$\theta_i = 2\tan^{-1}\frac{-e_{1i} \pm \sqrt{e_{1i}^2 + e_{2i}^2 - e_{3i}^2}}{e_{3i} - e_{2i}} \tag{9}$$

- In Eq. (9), negative sign is chosen because each leg has the elbow-down configuration in the local frame.

4) MATLAB Example



[2] Velocity and Statics Analysis and MATLAB Practice

(1) 2-DOF serial robot

- Taking the derivative of position vectors with respect to time, the Jacobian matrix for the 2-DOF serial robot can be obtained by

$$p_x = a_1 c\theta_1 + a_2 c\theta_{12} \tag{1}$$

$$p_y = a_1 s\theta_1 + a_2 s\theta_{12} \tag{2}$$

- The velocity relation is given by

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \tag{3}$$

where $[v_x,\ v_y]^T$ is the velocity at the end-effector, $[\dot{\theta}_1,\ \dot{\theta}_2]^T$ is the joint angular velocity, and the Jacobian matrix is

$$J = \begin{bmatrix} -a_1 s\theta_1 - a_2 s\theta_{12} & -a_2 s\theta_{12} \\ a_1 c\theta_1 + a_2 c\theta_{12} & a_2 c\theta_{12} \end{bmatrix} \tag{4}$$

- The statics relation can be derived from the principle of virtual works.

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = J^T \begin{bmatrix} f_x \\ f_y \end{bmatrix} \tag{5}$$

where $[f_x,\ f_y]^T$ is the force at the end-effector, and $[\tau_1,\ \tau_2]^T$ is the joint torque vector.

(2) 3-DOF serial robot

- Taking the derivative of position vectors with respect to time, the Jacobian matrix for the 3-DOF serial robot can be obtained by

$$q_x = a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} \tag{1}$$

$$q_y = a_1 s\theta_1 + a_2 s\theta_{12} + a_3 s\theta_{123} \tag{2}$$

$$\phi = \theta_1 + \theta_2 + \theta_3 \tag{3}$$

- The velocity relation is given by

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = J \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} \tag{4}$$

where $[v_x,\ v_y,\ \omega_z]^T$ is the velocity at the end-effector, $[\dot{\theta}_1,\ \dot{\theta}_2, \dot{\theta}_3]^T$ is the joint angular velocity, and the Jacobian matrix is

$$J = \begin{bmatrix} -a_1 s\theta_1 - a_2 s\theta_{12} - a_3 s\theta_{123} & -a_2 s\theta_{12} - a_3 s\theta_{123} & -a_3 s\theta_{123} \\ a_1 c\theta_1 + a_2 c\theta_{12} + a_3 c\theta_{123} & a_2 c\theta_{12} + a_3 c\theta_{123} & a_3 c\theta_{123} \\ 1 & 1 & 1 \end{bmatrix} \tag{5}$$

- The statics relation can be derived from the principle of virtual works.

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = J^T \begin{bmatrix} f_x \\ f_y \\ n_z \end{bmatrix} \tag{6}$$

where $[f_x,\ f_y,\ n_z]^T$ is the force and moment at the end-effector, and $[\tau_1,\ \tau_2,\ \tau_3]^T$ is the joint torque vector.

(3) 5-bar robot

- The vectors are defined by

$$\boldsymbol{r}_1 = \overrightarrow{AB}, \quad \boldsymbol{r}_2 = \overrightarrow{ED}, \quad \boldsymbol{u}_1 = \frac{\overrightarrow{BC}}{\left\|\overrightarrow{BC}\right\|}, \quad \boldsymbol{u}_2 = \frac{\overrightarrow{DC}}{\left\|\overrightarrow{DC}\right\|}$$

- The force at the end-effector ($\boldsymbol{f} = [f_x, f_y]^T$) is in static equilibrium with the joint force (f_1, f_2) at points B and C.

$$\begin{aligned} \boldsymbol{f} &= \boldsymbol{u}_1 f_1 + \boldsymbol{u}_2 f_2 \\ &= \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \end{aligned} \tag{1}$$

- The joint force has the relation with the joint torque ($\boldsymbol{\tau} = [\tau_1, \tau_2]^T$) given by

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} (\boldsymbol{r}_1 \times \boldsymbol{u}_1) \cdot \boldsymbol{k} & 0 \\ 0 & (\boldsymbol{r}_2 \times \boldsymbol{u}_2) \cdot \boldsymbol{k} \end{bmatrix}^{-1} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{2}$$

- Then, the statics relation between the force at the end-effector and the joint torque is derived by

$$\begin{aligned} \boldsymbol{f} &= \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 \end{bmatrix} \begin{bmatrix} (\boldsymbol{r}_1 \times \boldsymbol{u}_1) \cdot \boldsymbol{k} & 0 \\ 0 & (\boldsymbol{r}_2 \times \boldsymbol{u}_2) \cdot \boldsymbol{k} \end{bmatrix}^{-1} \boldsymbol{\tau} \\ &= J\, \boldsymbol{\tau} \end{aligned} \tag{3}$$

where the Jacobian matrix is given by

$$J = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 \end{bmatrix} \begin{bmatrix} (\boldsymbol{r}_1 \times \boldsymbol{u}_1) \cdot \boldsymbol{k} & 0 \\ 0 & (\boldsymbol{r}_2 \times \boldsymbol{u}_2) \cdot \boldsymbol{k} \end{bmatrix}^{-1} \tag{4}$$

- The velocity relation can be derived from the principle of virtual works.

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = J^T \begin{bmatrix} v_x \\ v_y \end{bmatrix} \qquad (5)$$

where $[v_x,\ v_y]^T$ is the velocity at the end-effector, and $[\dot{\theta}_1,\ \dot{\theta}_2]^T$ is the joint angular velocity.

(4) 3-DOF parallel robot

- The vectors are defined by

$$\boldsymbol{r}_i = \overrightarrow{A_iM_i}\,,\quad \boldsymbol{u}_i = \frac{\overrightarrow{M_iB_i}}{\left\|\overrightarrow{M_iB_i}\right\|}\,,\quad \boldsymbol{r}_{ei} = \overrightarrow{PB_i}$$

- The force and moment at the end-effector ($\boldsymbol{f} = [f_x, f_y]^T$, and n_z) are in static equilibrium with the joint force (f_i) at point B_i .

$$\begin{bmatrix} \boldsymbol{f} \\ n_z \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 \\ (\boldsymbol{r}_{e1} \times \boldsymbol{u}_1)\cdot \boldsymbol{k} & (\boldsymbol{r}_{e2} \times \boldsymbol{u}_2)\cdot \boldsymbol{k} & (\boldsymbol{r}_{e3} \times \boldsymbol{u}_3)\cdot \boldsymbol{k} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} \qquad (1)$$

- The joint force has the relation with the joint torque ($\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]^T$) given by

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} (\boldsymbol{r}_1 \times \boldsymbol{u}_1)\cdot \boldsymbol{k} & 0 & 0 \\ 0 & (\boldsymbol{r}_2 \times \boldsymbol{u}_2)\cdot \boldsymbol{k} & 0 \\ 0 & 0 & (\boldsymbol{r}_3 \times \boldsymbol{u}_3)\cdot \boldsymbol{k} \end{bmatrix}^{-1} \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \qquad (2)$$

- Therefore, the statics relation between the force at the end-effector and the joint torque is given by

$$\boldsymbol{f} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 \\ (\boldsymbol{r}_{e1} \times \boldsymbol{u}_1)\cdot \boldsymbol{k} & (\boldsymbol{r}_{e2} \times \boldsymbol{u}_2)\cdot \boldsymbol{k} & (\boldsymbol{r}_{e3} \times \boldsymbol{u}_3)\cdot \boldsymbol{k} \end{bmatrix} \begin{bmatrix} (\boldsymbol{r}_1 \times \boldsymbol{u}_1)\cdot \boldsymbol{k} & 0 & 0 \\ 0 & (\boldsymbol{r}_2 \times \boldsymbol{u}_2)\cdot \boldsymbol{k} & 0 \\ 0 & 0 & (\boldsymbol{r}_3 \times \boldsymbol{u}_3)\cdot \boldsymbol{k} \end{bmatrix}^{-1} \boldsymbol{\tau} \qquad (3)$$

$$= J\,\boldsymbol{\tau}$$

where the Jacobian matrix is

$$J = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 \\ (\boldsymbol{r}_{e1} \times \boldsymbol{u}_1)\cdot \boldsymbol{k} & (\boldsymbol{r}_{e2} \times \boldsymbol{u}_2)\cdot \boldsymbol{k} & (\boldsymbol{r}_{e3} \times \boldsymbol{u}_3)\cdot \boldsymbol{k} \end{bmatrix} \begin{bmatrix} (\boldsymbol{r}_1 \times \boldsymbol{u}_1)\cdot \boldsymbol{k} & 0 & 0 \\ 0 & (\boldsymbol{r}_2 \times \boldsymbol{u}_2)\cdot \boldsymbol{k} & 0 \\ 0 & 0 & (\boldsymbol{r}_3 \times \boldsymbol{u}_3)\cdot \boldsymbol{k} \end{bmatrix}^{-1} \qquad (4)$$

- The velocity relation can be derived from the principle of virtual works.

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = J^T \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \qquad (5)$$

where $[v_x,\ v_y, \omega_z]^T$ is the linear and angular velocity at the end-effector, and $[\dot{\theta}_1,\ \dot{\theta}_2, \dot{\theta}_3]^T$ is the joint angular velocity.

(5) MATLAB practice

- When the 2-DOF serial robot with $a_1 = 0.1,\ a_2 = 0.1\,[\mathrm{m}]$ is at the position of $\theta_1 = 0^\circ$ and $\theta_2 = 90^\circ$, determined the joint velocity and torque to generate the end-effector velocity and force, $\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$ [m/sec] and $\begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$ [N].

[skm_2dof_Jacobian.m]

```
function []=skm_2dof_Jacobian()
clc;
D2R=pi/180;
% Robot parameters %
a1= 0.1; a2=0.1;        % [m]
param=[0,a1,0; 0,a2,0];

th=D2R*[0,90]';

[J]=Jacobian_Matrix(th, param);

% Joint Velocities %
v=[0.1,0.1]';      % [m/s]
th_dot=inv(J)*v;

% Joint torques %
f=[0,10]';              % [N]
tau=J'*f;

disp('Jacobian Matrix');
disp(J);
disp('Joint velocity [rad/s]');
disp(th_dot');
disp('Joint torques [Nm]');
disp(tau');

function [J]=Jacobian_Matrix(th, param)
a1=param(1,2); a2=param(2,2);
th1=th(1,1); th2=th(2,1);
% Jacobian Matrix %
J=[-a1*sin(th1)-a2*sin(th1+th2), -a2*sin(th1+th2);...
       a1*cos(th1)+a2*cos(th1+th2), a2*cos(th1+th2)];
```

[3] Dynamics Analysis and MATLAB Practice

(1) Introduction

1) Dynamics equation

- In general, most mechanical systems can be modeled with the second-order differential equations.

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = F_{ext}(t) \tag{1}$$

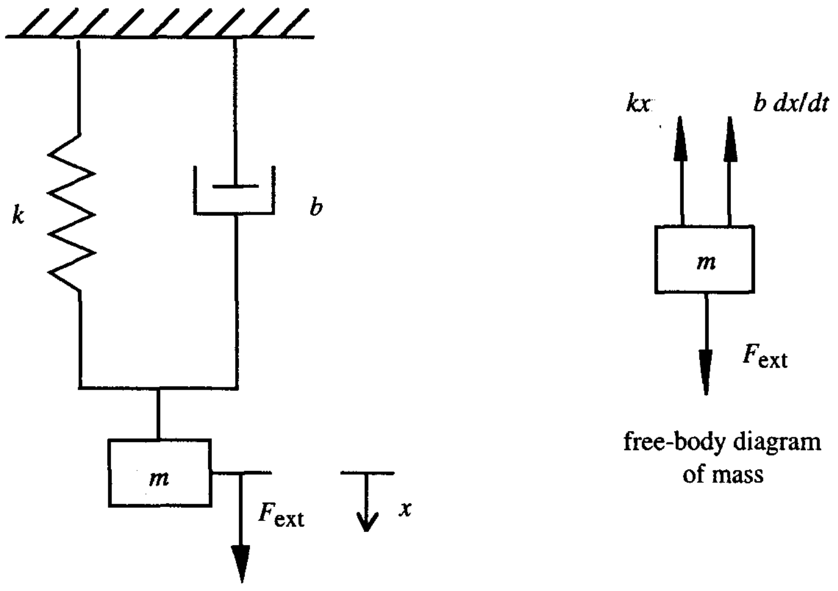
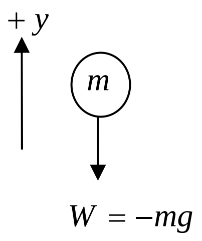


Fig. 7: Equation of motions and free-body diagram.

- Considering a falling sphere, its dynamics equation is modeled by

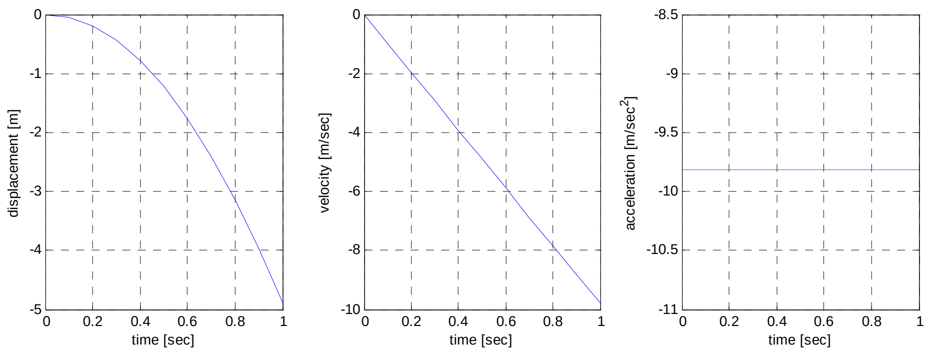
$$m\frac{d^2y}{dt^2} = -mg \tag{2}$$



- The dynamic simulation is to determine the position, velocity, and acceleration of an object for given external force. The motion can be calculated by taking the integral of dynamic equation with respect to time. For example,

$$\begin{aligned} a(t) &= \frac{d^2y}{dt^2} = -g\,, \\ v(t) &= \int_0^t a(t)\,dt = \int_0^t -g\,dt = -gt\,, \\ s(t) &= \int_0^t v(t)\,dt = \int_0^t (-gt)\,dt = -\frac{1}{2}gt^2 \end{aligned} \tag{3}$$

- For $g = 9.81\,\mathrm{m/sec^2}$, the position, velocity, and acceleration of a falling sphere are plotted by



- The following is the MATLAB program of the simulation. Save it as "plot_sva.m" and type "plot_sva" in the command window to run the program.

[plot_sva.m]

```
function plot_sva
g=9.81;         % Gravity=9.81 m/s^2 %
t=0:0.1:1;
s=-0.5*g*t.^2; v=-g*t; a=-g*ones(1,11);

figure(1)
subplot(1,3,1), plot(t,s);
xlabel('time [sec]'); ylabel('displacement [m]'); grid on;
subplot(1,3,2), plot(t,v);
xlabel('time [sec]'); ylabel('velocity [m/sec]'); grid on;
subplot(1,3,3), plot(t,a);
xlabel('time [sec]'); ylabel('acceleration [m/sec^2]'); grid on;
```

2) State-space representation

- The state-space representation is expressed by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t)) \tag{4}$$

- For example, an n th order differential equation is

$$a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \cdots + a_1 \frac{dy}{dt} + a_0 y = u \tag{5}$$

- Defining n state variables ($a_n = 1$) as

$$\begin{aligned} x_1 &= y \\ x_2 &= \dot{y} = \dot{x}_1 \\ x_3 &= \ddot{y} = \dot{x}_2 \\ &\vdots \\ x_n &= \frac{d^{n-1} y}{dt^{n-1}} = \dot{x}_{n-1} \end{aligned} \tag{6}$$

- Rewriting Eq. (6) gives,

$$\begin{aligned}
&\dot{x}_1 = x_2 \\
&\dot{x}_2 = x_3 \\
&\vdots \\
&\dot{x}_{n-1} = x_n \\
&\dot{x}_n = \frac{d^n y}{dt^n} = -a_0 y - a_1 \frac{dy}{dt} - \cdots - a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + u \\
&\quad = -a_0 x_1 - a_1 x_2 - \cdots - a_{n-1}x_n + u
\end{aligned} \tag{7}$$

- In a matrix form,

$$\begin{bmatrix} \dot{x}_1 \\ . \\ . \\ . \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & . & . & 0 \\ 0 & 0 & 1 & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & 1 \\ -a_0 & . & . & . & -a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ . \\ . \\ . \\ x_n \end{bmatrix} + \begin{bmatrix} 0 \\ . \\ . \\ 0 \\ 1 \end{bmatrix} u \tag{8}$$

Or,

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + \boldsymbol{b}u \tag{9}$$

- The output vector y is written by

$$y = [1,0,...,0]\boldsymbol{x} = \boldsymbol{c}^T \boldsymbol{x} \tag{10}$$

[Example 1]

Express $m\dfrac{d^2 y}{dt^2} = -mg$ in the state-space representation.

(Solution)

State-space variables:

$$\begin{aligned} x_1 &= y \\ x_2 &= \dot{y} = \dot{x}_1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{y} = -g \end{aligned}$$

The matrix form is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} g \quad \Rightarrow \quad \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}u$$

[Example 2]

Express $m\dfrac{d^2y}{dt^2} + b\dfrac{dy}{dt} + ky = u$ in the state-space representation.

(Solution)

State-space variables:

$$\begin{aligned} x_1 &= y \\ x_2 &= \dot{y} = \dot{x}_1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{y} = -\frac{k}{m}y - \frac{b}{m}\frac{dy}{dt} + \frac{u}{m} = -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{1}{m}u \end{aligned}$$

The matrix form is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{k}{m} & -\dfrac{b}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix} u \quad \Rightarrow \quad \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}u$$

3) Dynamic simulation using MATLAB

- Let's do the dynamic simulation for the falling stone by using MATLAB. Use the state-space equation ($\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}u$) in Example 1, and use the numerical integration function, "ode45". The usage of the function is given by

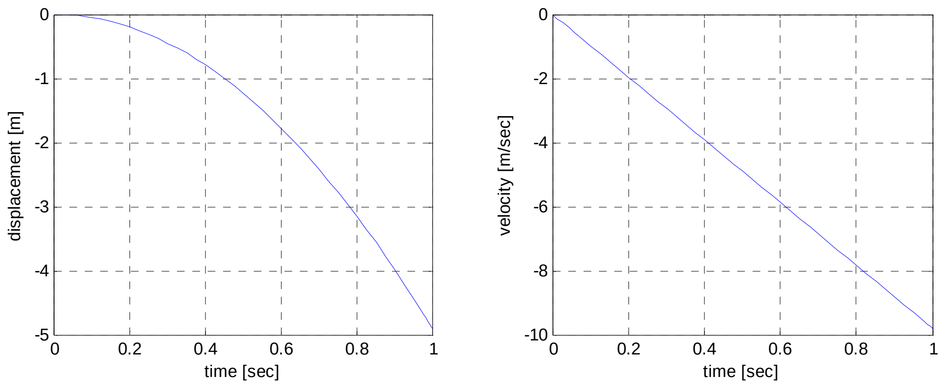
  [T, Y] = solver(odefun,tspan,y0,options)

- Open new function and save as "falling_sphere.m". Run the program and check the result.

[falling_sphere.m]

```
function [T, Y]=falling_sphere

g=9.81;         % Gravity=9.81 m/s^2 %
A=[0,1;0,0]; b=[0; -1]; u=g;
```

```
tspan=[0,1];        % t0=0, tf=1 sec %
y0=[0;0];            % s0=0, v0=0 %
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4]);
[T,Y] = ode45(@(t,y) falling_stone(t, y, A, b, u),tspan,y0,options);

figure(1)
subplot(1,2,1), plot(T,Y(:,1));
xlabel('time [sec]'); ylabel('displacement [m]'); grid on;
subplot(1,2,2), plot(T,Y(:,2));
xlabel('time [sec]'); ylabel('velocity [m/sec]'); grid on;

function dy=falling_stone(t,y, A, b, u)
dy=A*y+b*u;
```



(2) 1-DOF equation of motion

- The state-space representation for 1-DOF equation of motion is given by

$$f = m\ddot{x} + b\dot{x} + kx \tag{1}$$

- State-space variables:

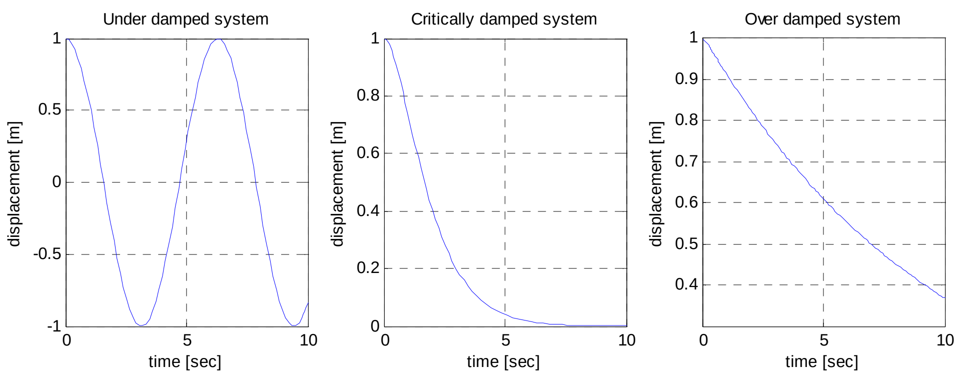
$$\begin{aligned} x_1 &= x \\ x_2 &= \dot{x} = \dot{x}_1 \end{aligned} \quad \Rightarrow \quad \begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \ddot{y} = -\frac{k}{m}x - \frac{b}{m}\dot{x} + \frac{u}{m} = -\frac{k}{m}x_1 - \frac{b}{m}x_2 + \frac{1}{m}f \end{aligned} \tag{2}$$

- The matrix form is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} f \quad \Rightarrow \quad \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\, f \tag{3}$$

(Ex) Forward dynamics simulation of a 1-DOF mechanical system

$f = 0$ (free vibration), $x(0) = 1[\text{m}]$, $\dot{x}(0) = 0$



[mechanical_system.m]

```
function [T, Y]=mechanical_system

f=0;        % Free Vibration %
tspan=[0,10];       % t0=0, tf=1 sec %
```

```
y0=[1;0];            % s0=0, v0=0 %
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4]);

%% (1) Under Damped
% mass-damper-spring constants %
m=1; b=0; k=1;
% State-space Equation %
A=[0,1;-k/m,-b/m]; b=[0; 1/m]; u=f;
[T1,Y1] = ode45(@(t,y) equation_of_motion(t, y, A, b, u),tspan,y0,options);

%% (2) Critically Damped
% mass-damper-spring constants %
m=1; b=2; k=1;
% State-space Equation %
A=[0,1;-k/m,-b/m]; b=[0; 1/m]; u=f;
[T2,Y2] = ode45(@(t,y) equation_of_motion(t, y, A, b, u),tspan,y0,options);

%% (3) Over Damped
% mass-damper-spring constants %
m=1; b=10; k=1;
% State-space Equation %
A=[0,1;-k/m,-b/m]; b=[0; 1/m]; u=f;
[T3,Y3] = ode45(@(t,y) equation_of_motion(t, y, A, b, u),tspan,y0,options);

figure(1)
subplot(1,3,1), plot(T1,Y1(:,1));
title('Under damped system');
xlabel('time [sec]'); ylabel('displacement [m]'); grid on;
subplot(1,3,2), plot(T2,Y2(:,1));
title('Critically damped system');
xlabel('time [sec]'); ylabel('displacement [m]'); grid on;
subplot(1,3,3), plot(T3,Y3(:,1));
title('Over damped system');
xlabel('time [sec]'); ylabel('displacement [m]'); grid on;

function dy=equation_of_motion(t, y, A, b, u)
dy=A*y+b*u;
```

(3) Dynamic Analysis of the 2-DOF serial robot

- a_{c1} and a_{c2} denote the distances from joints to mass centers, and the mass moments of inertia are I_{c1} and I_{c2}.

- Dynamic equation using Lagrangian method:

$$\begin{aligned}\tau_1 =& (m_1 a_{c1}^2 + m_2(a_1^2 + a_{c2}^2) + 2m_2 a_1 a_{c2} c\theta_2 + I_{c1} + I_{c2})\ddot{\theta}_1 + (m_2 a_{c2}^2 + m_2 a_1 a_{c2} c\theta_2 + I_{c2})\ddot{\theta}_2 \\ &- m_2 a_1 a_{c2} s\theta_2 (2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^2) + g_c(m_1 a_{c1} c\theta_1 + m_2(a_1 c\theta_1 + a_{c2} c\theta_{12}))\end{aligned} \tag{1}$$

$$\tau_2 = (m_2 a_{c2}^2 + m_2 a_1 a_{c2} c\theta_2 + I_{c2})\ddot{\theta}_1 + (m_2 a_{c2}^2 + I_{c2})\ddot{\theta}_2 + m_2 a_1 a_{c2} s\theta_2 \dot{\theta}_1^2 + m_2 g_c a_{c2} c\theta_{12} \qquad (2)$$

- Dynamic equation for an ideal case:

$$a_{c1} = \frac{a_1}{2},\ \ a_{c2} = \frac{a_2}{2},\ \ I_{c1} = \frac{m_1 a_1^2}{12},\ \ I_{c2} = \frac{m_2 a_2^2}{12}$$

$$\begin{aligned}\tau_1 = &\left((\frac{1}{3}m_1 + m_2)a_1^2 + m_2 a_1 a_2 c\theta_2 + \frac{1}{3}m_2 a_2^2 \right)\ddot{\theta}_1 + \left(\frac{1}{2}m_2 a_1 a_2 c\theta_2 + \frac{1}{3}m_2 a_2^2 \right)\ddot{\theta}_2 \\ &- m_2 a_1 a_2 s\theta_2 (\dot{\theta}_1\dot{\theta}_2 + \frac{1}{2}\dot{\theta}_2^2) + g_c \left((\frac{1}{2}m_1 + m_2)a_1 c\theta_1 + \frac{1}{2}m_2 a_2 c\theta_{12} \right)\end{aligned} \qquad (3)$$

$$\tau_2 = \left(\frac{1}{2}m_2 a_1 a_2 c\theta_2 + \frac{1}{3}m_2 a_2^2 \right)\ddot{\theta}_1 + \left(\frac{1}{3}m_2 a_2^2 \right)\ddot{\theta}_2 + \left(\frac{1}{2}m_2 a_1 a_2 s\theta_2 \right)\dot{\theta}_1^2 + \frac{1}{2}m_2 g_c a_2 c\theta_{12} \qquad (4)$$

- In a matrix form,

$$\begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} = \begin{bmatrix}M_{11} & M_{12} \\ M_{21} & M_{22}\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1 \\ \ddot{\theta}_2\end{bmatrix} + \begin{bmatrix}C_{11} & C_{12} \\ C_{21} & C_{22}\end{bmatrix}\begin{bmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix} + \begin{bmatrix}G_1 \\ G_2\end{bmatrix} \qquad (3)$$

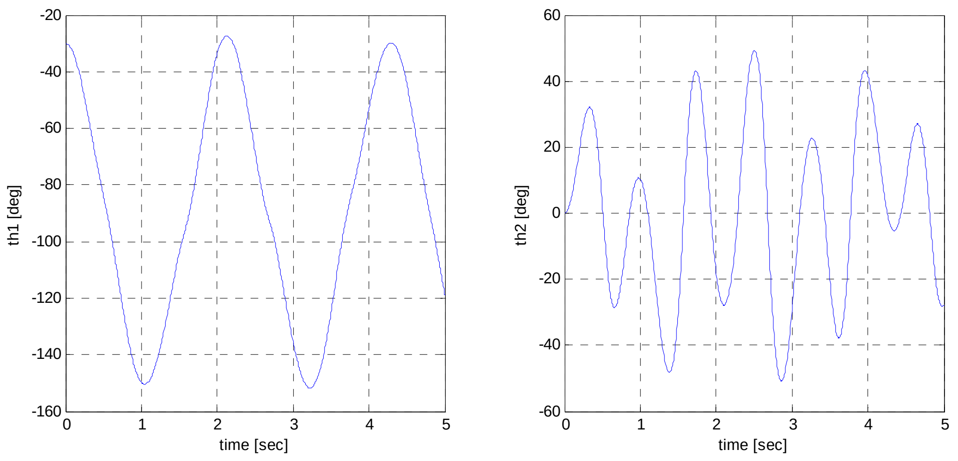
The state-space representation is

For $\boldsymbol{\theta} = \begin{bmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix}$,

$$\begin{aligned}\boldsymbol{x}_1 &= \boldsymbol{\theta} \\ \boldsymbol{x}_2 &= \dot{\boldsymbol{\theta}} = \dot{\boldsymbol{x}}_1\end{aligned} \quad \Rightarrow \quad \begin{aligned}\dot{\boldsymbol{x}}_1 &= \boldsymbol{x}_2 \\ \dot{\boldsymbol{x}}_2 &= \ddot{\boldsymbol{\theta}} = M^{-1}(\boldsymbol{\theta})\left[-C(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} - \boldsymbol{G}(\boldsymbol{\theta}) + \boldsymbol{\tau}\right] \\ &= M^{-1}(\boldsymbol{x}_1)\left[-C(\boldsymbol{x}_1,\boldsymbol{x}_2)\boldsymbol{x}_2 - \boldsymbol{G}(\boldsymbol{x}_1) + \boldsymbol{\tau}\right]\end{aligned} \qquad (4)$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{\theta}$ are 2x1 vectors.

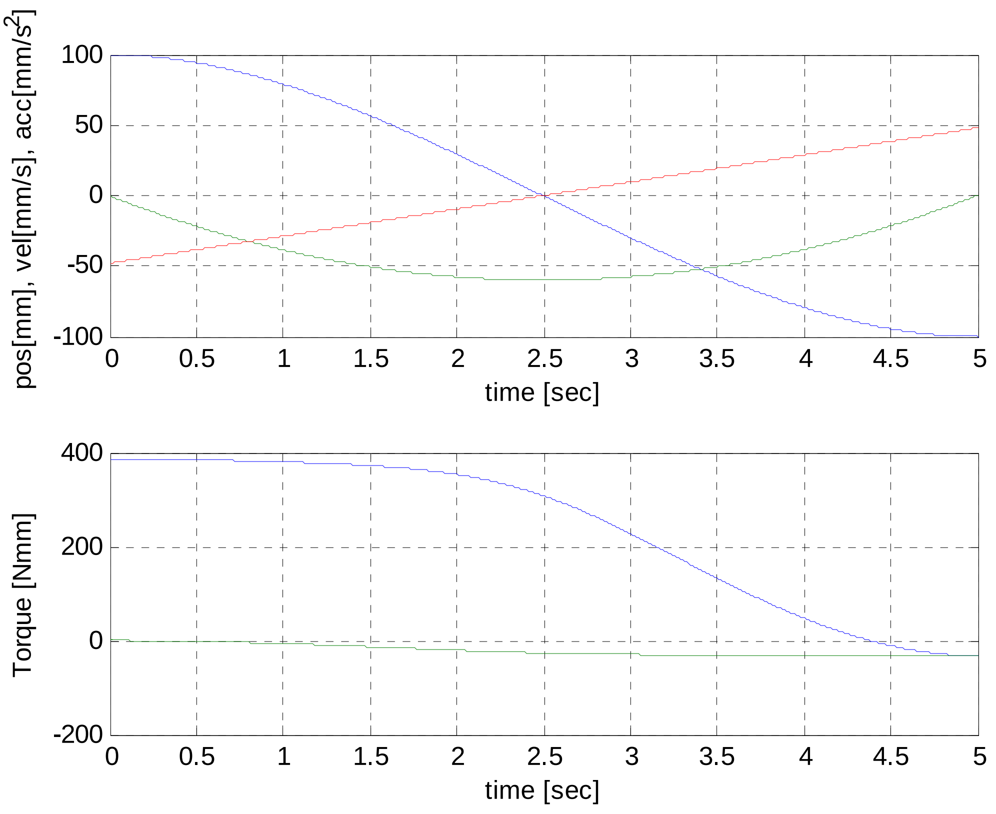
[Ex1] Forward dynamics simulation for the 2-DOF serial robot with $a_1 = 1,\ a_2 = 0.5$ [m], $m_1 = 1,\ \mathrm{m}_2 = 0.5$ [kg], the initial condition $\theta_1(0) = -30^\circ, \theta_2(0) = 0^\circ$.



[Ex2] Inverse Dynamics Analysis for the 2-DOF serial robot with the following mass properties:

m1=0.363[kg], a1=100, ac1=87.74[mm], Ic1=313.923[kgmm^2],
m2=0.075[kg], a2=100, ac2=41.68[mm], Ic2=109.864[kgmm^2],
g=9806.65[kg m/s^2],

$$p_x(0)=100, p_y(0)=100 \rightarrow p_x(5)=-100, p_y(5)=100 \text{ [mm]}$$



[Ex3] Dynamic Analysis of the 3-DOF serial robot

- Considering only gravity terms, the actuator torques are determined by

$$\tau_1 = g_c(m_1 a_{c1} c\theta_1 + m_2(a_1 c\theta_1 + a_{c2} c\theta_{12}) + m_3(a_1 c\theta_1 + a_2 c\theta_{12} + a_{c3} c\theta_{123}))$$

$$\tau_2 = g_c(m_2 a_{c2} c\theta_{12} + m_3(a_2 c\theta_{12} + a_{c3} c\theta_{123}))$$

$$\tau_3 = m_3 g_c a_{c3} c\theta_{123}$$

[Reference] Trajectory Planning using a Cubic Polynomial

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{1}$$

- The initial and final conditions:

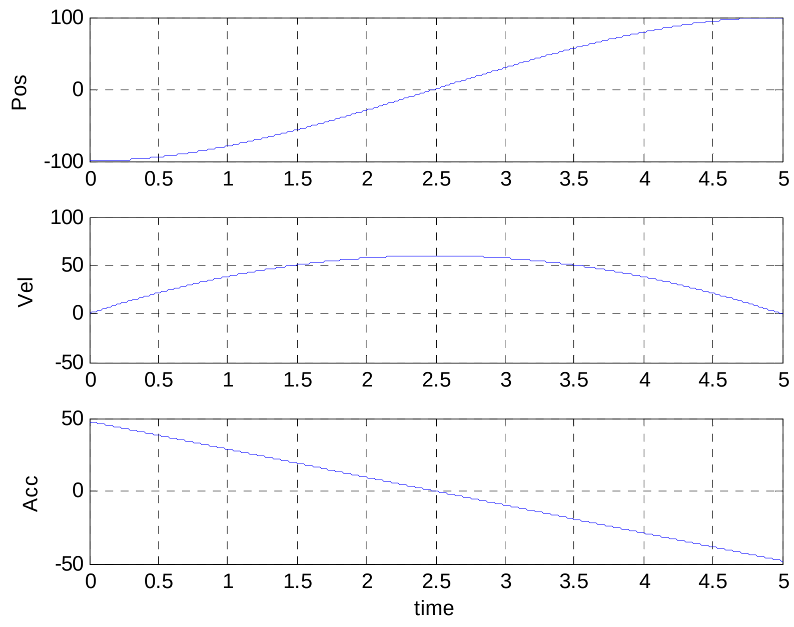
$$\begin{matrix} \theta(0) = \theta_0 & \dot{\theta}(0) = \dot{\theta}_0 \\ \theta(t_f) = \theta_f , & \dot{\theta}(t_f) = \dot{\theta}_f \end{matrix} \tag{2}$$

- The coefficients are determined by

$$\begin{aligned} a_0 &= \theta_0 \\ a_1 &= \dot{\theta}_0 \\ a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta}_f \\ a_3 &= -\frac{2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta}_f + \dot{\theta}_0) \end{aligned} \tag{3}$$

- The position, velocity, and acceleration are calculated with the following functions:

$$\begin{aligned} \theta(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ \dot{\theta}(t) &= a_1 + 2a_2 t + 3a_3 t^2 \\ \ddot{\theta}(t) &= 2a_2 + 6a_3 t \end{aligned} \tag{4}$$



(4) MATLAB practice

- The following program is the inverse dynamics analysis of the 2-DOF serial robot.

[dyn_2dof.m]

```
function []=dyn_2dof()
clear all;

a1=100; a2=100;
x0=[a1,a2]';
xf=[-a1,a2]';
tf=5; dt=0.001;

x=zeros(2,1); x_dot=zeros(2,1); x_ddot=zeros(2,1);
[c] = coeff_3rd(x0, zeros(2,1), xf, zeros(2,1), tf);

n=1;
for t=0:dt:tf
        for ch=1:2
                x(ch,1)=c(ch,1)+c(ch,2)*t+c(ch,3)*t^2+c(ch,4)*t^3;
                x_dot(ch,1)=c(ch,2)+2*c(ch,3)*t+3*c(ch,4)*t^2;
                x_ddot(ch,1)=2*c(ch,3)+6*c(ch,4)*t;
        end
        [th]=inverse_kinematics(x);
        [th_dot, th_ddot]=derivatives(th, dt);

        % Required Torques %
        [T]=dynamics(th_ddot, th_dot, th);
        x_data(n,1)=t;
        y_data1(n,:)=[x(1,1), x_dot(1,1), x_ddot(1,1)];
        y_data2(n,:)=T;
        n=n+1;
end

figure(1)
subplot(2,1,1), plot(x_data, y_data1);
xlabel('time [sec]'); ylabel('pos[mm], vel[mm/s], acc[mm/s^2]'); grid on;
subplot(2,1,2), plot(x_data, y_data2);
xlabel('time [sec]'); ylabel('Torque [Nmm]'); grid on;

function [v, a]=derivatives(x, dt)
global MAX_CH
persistent xp vp

if isempty(xp), xp=x; end
if isempty(vp), vp=zeros(MAX_CH,1); end

v=(x-xp)/dt;
a=(v-vp)/dt;

% Back substitutions %
```

```
xp=x; vp=v;

function [th,w_index]=inverse_kinematics(xd)

w_index=1;    % In workspace %
a1=100; a2=100;

% (1) Calculate th2 %
px=xd(1,1); py=xd(2,1);
kapha=(px^2+py^2-a1^2-a2^2)/(2*a1*a2);
if abs(kapha)>1
        w_index=-1;    % Out of Workspace %
        kapha=1;
end
th2=+acos(kapha); % Elbow Down %
%th2=-acos(kapha); % Elbow Up %

% (2) Calculate th1 %
delta=a1^2+a2^2+2*a1*a2*cos(th2);
cth1=(px*(a1+a2*cos(th2))+py*a2*sin(th2))/delta;
sth1=(-px*a2*sin(th2)+py*(a1+a2*cos(th2)))/delta;
th1=atan2(sth1, cth1);
if th1<0, th1=th1+2*pi; end % 0<th<360 %

th=[th1, th2]';



function [T]=dynamics(th_ddot, th_dot, th)
% Robot Parameters %
m1=0.363; a1=100; ac1=87.74; Ic1=313.923;
m2=0.075; a2=100; ac2=41.68; Ic2=109.864;
g=9806.65;

% State Variables %
th1=th(1,1); th2=th(2,1);
th1_dot=th_dot(1,1); th2_dot=th_dot(2,1);

M(1,1)=m1*ac1^2+m2*(a1^2+ac2^2)+2*m2*a1*ac2*cos(th2)+Ic1+Ic2;
M(1,2)=m2*ac2^2+m2*a1*ac2*cos(th2)+Ic2;
M(2,1)=M(1,2);
M(2,2)=m2*ac2^2+Ic2;

C(1,1)=-m2*a1*ac2*sin(th2)*(2*th1_dot*th2_dot+th2_dot^2);
C(2,1)=m2*a1*ac2*sin(th2)*th1_dot^2;

G(1,1)=g*(m1*ac1*cos(th1)+m2*(a1*cos(th1)+ac2*cos(th1+th2)));
G(2,1)=m2*g*ac2*cos(th1+th2);

T=0.001*(M*th_ddot+C+G);

%---------------------- Trajectory Functions
function [c] = coeff_3rd(sp, sv, ep, ev, tsec)
MAX_CH=2;
c=zeros(MAX_CH,4);
```

```
for ch=1:MAX_CH
        c(ch,1)=sp(ch,1);
        c(ch,2)=sv(ch,1);
        c(ch,3)=(3.0*(ep(ch,1)-sp(ch,1))-(2.0*sv(ch,1)+ev(ch,1))*tsec)/(tsec*tsec);
        c(ch,4)=(-2.0*(ep(ch,1)-sp(ch,1))+(sv(ch,1)+ev(ch,1))*tsec)/(tsec*tsec*tsec);
end
```

[forward_2dof.m]

```
function [T, Y]=forward_2dof()

D2R=pi/180; R2D=180/pi;

tspan=[0,5];              % t0=0, tf=5 sec %
y0=D2R*[-30;0;0;0]; % th1_0=-30, th2_0=0, th1_dot_0=0, th2_dot_0=0 %
options = odeset('RelTol',1e-6,'AbsTol',[1e-6 1e-6 1e-6 1e-6]);
[T,Y] = ode45(@equation_of_motion,tspan,y0,options);

figure(1)
subplot(1,2,1), plot(T,Y(:,1)*R2D);
xlabel('time [sec]'); ylabel('th1 [deg]'); grid on;
subplot(1,2,2), plot(T,Y(:,2)*R2D);
xlabel('time [sec]'); ylabel('th2 [deg]'); grid on;

% simulation %
figure(2)
axis([-1.5, 1.5, -2, 1]); axis square
xlabel('x-axis'); ylabel('y-axis');
L1=1; L2=0.5; param=[0,L1,0; 0,L2,0];
size_Y=size(Y);
for n=1:size_Y(1)
        cla;
        th(1,1)=Y(n,1); th(2,1)=Y(n,2);
        draw_2dof_robot(th, param);
        pause(0.02);
end

%% Dynamics
function dy=equation_of_motion(t, y)

% Robot Parameters %
m1=1; m2=0.5; L1=1; L2=0.5; g=9.81;
% State Variables %
th1=y(1,1); th2=y(2,1); th1_dot=y(3,1); th2_dot=y(4,1);

M(1,1)=((1/3)*m1+m2)*L1^2+m2*L1*L2*cos(th2)+(1/3)*m2*L2^2;
M(1,2)=(1/2)*m2*L1*L2*cos(th2)+(1/3)*m2*L2^2;
M(2,1)=M(1,2);
M(2,2)=(1/3)*m2*L2^2;
C(1,1)=-m2*L1*L2*sin(th2)*(th1_dot*th2_dot+(1/2)*th2_dot^2);
C(2,1)=(1/2)*m2*L1*L2*sin(th2)*th1_dot^2;
G(1,1)=g*(((1/2)*m1+m2)*L1*cos(th1)+(1/2)*m2*L2*cos(th1+th2));
G(2,1)=(1/2)*m2*g*L2*cos(th1+th2);
T(1,1)=0;
T(2,1)=0;
```

```
dy(1:2,1)=[th1_dot; th2_dot];
dy(3:4,1)=inv(M)*(-C-G+T);

%% Draw Functions
%-------- Draw a 3-DOF Planar Robot
function draw_2dof_robot(th, param)
% Draw the First Link %
T01=Transformation(th(1,1), param(1,:));
p1=[0,0,0]'; p2=T01(1:3,4);
lw=4; clr=[0.5,0.1,0.1];
draw_line(p1,p2,lw,clr);

% Draw the Second Link %
T02=T01*Transformation(th(2,1), param(2,:));
p1=T01(1:3,4); p2=T02(1:3,4);
lw=4; clr=[0.1,0.5,0.1];
draw_line(p1,p2,lw,clr);

%-------- Draw a Frame
function draw_frame(T, sc)
p1=T*[0,0,0,1]'; p2=T*[sc,0,0,1]'; lw=2; clr=[1,0,0];
draw_line(p1,p2,lw,clr);
p1=T*[0,0,0,1]'; p2=T*[0,sc,0,1]'; lw=2; clr=[0,1,0];
draw_line(p1,p2,lw,clr);
p1=T*[0,0,0,1]'; p2=T*[0,0,sc,1]'; lw=2; clr=[0,0,1];
draw_line(p1,p2,lw,clr);

%------- Draw a Line
function draw_line(p1, p2, lw, clr)
line([p1(1,1),p2(1,1)],[p1(2,1),p2(2,1)],[p1(3,1),p2(3,1)],'LineWidth',lw,'Color',clr);

%------ Draw a Box %%% Multifaceted Patches
function draw_box(T, sc)
vt0=[0,0,0,1; sc,0,0,1; sc,sc,0,1; 0,sc,0,1; 0,0,sc,1; sc,0,sc,1; sc,sc,sc,1; 0,sc,sc,1]';
fm=[1,2,6,5; 2,3,7,6; 3,4,8,7; 4,1,5,8; 1,2,3,4; 5,6,7,8];
vt1=T*vt0; vm=vt1(1:3,:)';
patch('Vertices',vm,'Faces',fm,...
        'FaceVertexCData',gray(8),'FaceColor','flat');

%% Homogeneous Transformation Matrix %%
%------- Rotation matrix about the X-axis
function [T]=Rotx(th)
cx=cos(th); sx=sin(th);
T=[1,0,0,0; 0,cx,-sx,0; 0,sx,cx,0; 0,0,0,1];
%------- Rotation matrix about the Y-axis
function [T]=Roty(th)
cy=cos(th); sy=sin(th);
T=[cy,0,sy,0; 0,1,0,0; -sy,0,cy,0; 0,0,0,1];
%------- Rotation matrix about the Z-axis
function [T]=Rotz(th)
cz=cos(th); sz=sin(th);
T=[cz,-sz,0,0; sz,cz,0,0; 0,0,1,0; 0,0,0,1];
%-------- Translation along the x, y, z axes
function [T]=Trans(p)
```

```
T=[1,0,0,p(1,1); 0,1,0,p(2,1); 0,0,1,p(3,1); 0,0,0,1];
function [T]=Transformation(th, param)
alpha=param(1); a=param(2); d=param(3);
T=Trans([0,0,d]')*Rotz(th)*Trans([a,0,0]')*Rotx(alpha);
```