A Matlab Program for Analysis of Kinematics

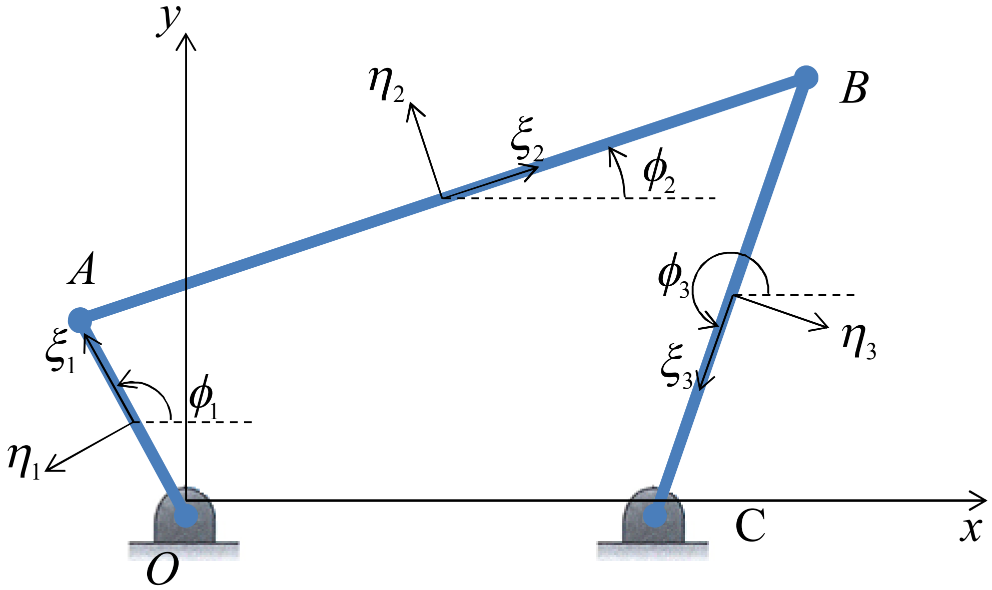
Example of a four-bar linkage mechanism

$\overline{OA} = 80\text{mm}$

$\overline{AB} = 260\text{mm}$

$\overline{BC} = 180\text{mm}$

$\overline{OC} = 180\text{mm}$



Constraint equations:

$$-x_1 + 40\cos\phi_1 = 0$$
$$-y_1 + 40\sin\phi_1 = 0$$
$$x_1 + 40\cos\phi_1 - x_2 + 130\cos\phi_2 = 0$$
$$y_1 + 40\sin\phi_1 - y_2 + 130\sin\phi_2 = 0$$
$$x_2 + 130\cos\phi_2 - x_3 + 90\cos\phi_3 = 0$$
$$y_2 + 130\sin\phi_2 - y_3 + 90\sin\phi_3 = 0$$
$$x_3 + 90\cos\phi_3 - 180 = 0$$
$$y_3 + 90\sin\phi_3 = 0$$
$$\phi_1 - 2\pi t - \pi/2 = 0$$

To solve the 9 equations for 9 unknown $\boldsymbol{q}^T = [x_1, y_1, \phi_1, x_2, y_2, \phi_2, x_3, y_3, \phi_3]$

The Jacobian matrix and the right-side of the velocity equations

$$
\mathbf{J} = \begin{array}{c} \partial\Phi_1/\partial \\ \partial\Phi_2/\partial \\ \partial\Phi_3/\partial \\ \partial\Phi_4/\partial \\ \partial\Phi_5/\partial \\ \partial\Phi_6/\partial \\ \partial\Phi_7/\partial \\ \partial\Phi_8/\partial \\ \partial\Phi_9/\partial \end{array}
\begin{bmatrix}
-1 & 0 & -40\sin\phi_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 40\cos\phi_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -40\sin\phi_1 & -1 & 0 & -130\sin\phi_2 & 0 & 0 & 0 \\
0 & 1 & 40\cos\phi_1 & 0 & -1 & 130\cos\phi_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -130\sin\phi_2 & -1 & 0 & -90\sin\phi_3 \\
0 & 0 & 0 & 0 & 1 & 130\cos\phi_2 & 0 & -1 & 90\cos\phi_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -90\sin\phi_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 90\cos\phi_3 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\qquad
\beta = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2\pi \end{bmatrix}
$$

To solve $\mathbf{J}\dot{\boldsymbol{q}} = \boldsymbol{\beta}$ for the velocity $\dot{\boldsymbol{q}}$

The right-side of the acceleration equations

$$\boldsymbol{\gamma} = \begin{bmatrix} 40\cos\phi_1 \cdot \dot{\phi}_1^2 \\ 40\sin\phi_1 \cdot \dot{\phi}_1^2 \\ 50\cos\phi_1 \cdot \dot{\phi}_1^2 + 130\cos\phi_2 \cdot \dot{\phi}_2^2 \\ 50\sin\phi_1 \cdot \dot{\phi}_1^2 + 130\sin\phi_2 \cdot \dot{\phi}_2^2 \\ 130\cos\phi_2 \cdot \dot{\phi}_2^2 + 90\cos\phi_3 \cdot \dot{\phi}_3^2 \\ 130\sin\phi_2 \cdot \dot{\phi}_2^2 + 90\sin\phi_3 \cdot \dot{\phi}_3^2 \\ 90\cos\phi_3 \cdot \dot{\phi}_3^2 \\ 90\sin\phi_3 \cdot \dot{\phi}_3^2 \\ 0 \end{bmatrix}$$

To solve $\mathbf{J}\ddot{\boldsymbol{q}} = \boldsymbol{\gamma}$ for the acceleration $\ddot{\boldsymbol{q}}$

The procedure of m-file

Initialize and start at $t = 0$

Call nonlinear solver with initial position to solve $\boldsymbol{q}(t)$

Determine Jacobian matrix and $\boldsymbol{\beta}$ with $\boldsymbol{q}(t)$ to solve $\dot{\boldsymbol{q}}(t)$

Determine $\boldsymbol{\gamma}$ with $\dot{\boldsymbol{q}}(t)$ to solve $\ddot{\boldsymbol{q}}(t)$

Assume $\boldsymbol{q}(t + \Delta t)$ with intended motion
and set it as the new initial position of next iteration

Plot the time response of $\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)$ and $\ddot{\boldsymbol{q}}(t)$

Determine the positions of the bars and animate them

The code of m.file (1)

```
% Set up the time interval and the initial positions of the nine coordinates
T_Int=0:0.01:2;
X0=[0 50 pi/2   125.86 132.55 0.2531   215.86 82.55 4.3026];
global T
Xinit=X0;

% Do the loop for each time interval
for Iter=1:length(T_Int);
    T=T_Int(Iter);
    % Determine the displacement at the current time
    [Xtemp,fval] = fsolve(@constrEq4bar,Xinit);

    % Determine the velocity at the current time
    phi1=Xtemp(3); phi2=Xtemp(6); phi3=Xtemp(9);
    JacoMatrix=Jaco4bar(phi1,phi2,phi3);
    Beta=[0 0 0    0 0 0    0 0 2*pi]';
    Vtemp=JacoMatrix\Beta;

    % Determine the acceleration at the current time
    dphi1=Vtemp(3); dphi2=Vtemp(6); dphi3=Vtemp(9);
    Gamma=Gamma4bar(phi1,phi2,phi3,dphi1,dphi2,dphi3);
    Atemp=JacoMatrix\Gamma;

    % Record the results of each iteration
    X(:,Iter)=Xtemp; V(:,Iter)=Vtemp; A(:,Iter)=Atemp;

    % Determine the new initial position to solve the equation of the next
    % iteration and assume that the kinematic motion is with inertia
    if Iter==1
        Xinit=X(:,Iter);
    else
        Xinit=X(:,Iter)+(X(:,Iter)-X(:,Iter-1));
    end

end
```

The code of m.file (2)

```
% T vs displacement plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,X(i,:))
    set(gca,'xtick',[], 'FontSize', 5)
end
% Reset the bottom subplot to have xticks
set(gca,'xtickMode', 'auto')

% T vs velocity plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,V(i,:))
    set(gca,'xtick',[], 'FontSize', 5)
end
set(gca,'xtickMode', 'auto')

% T vs acceleration plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,A(i,:))
    AxeSup=max(A(i,:));
    AxeInf=min(A(i,:));
    if AxeSup-AxeInf<0.01
        axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
    end
    set(gca,'xtick',[], 'FontSize', 5)
end
set(gca,'xtickMode', 'auto')
```

The code of m.file (3)

```
% Determine the positions of the four revolute joints at each iteration
Ox=zeros(1,length(T_Int));
Oy=zeros(1,length(T_Int));
Ax=80*cos(X(3,:));
Ay=80*sin(X(3,:));
Bx=Ax+260*cos(X(6,:));
By=Ay+260*sin(X(6,:));
Cx=180*ones(1,length(T_Int));
Cy=zeros(1,length(T_Int));

% Animation
figure
for t=1:length(T_Int);
    bar1x=[Ox(t) Ax(t)];
    bar1y=[Oy(t) Ay(t)];
    bar2x=[Ax(t) Bx(t)];
    bar2y=[Ay(t) By(t)];
    bar3x=[Bx(t) Cx(t)];
    bar3y=[By(t) Cy(t)];

    plot (bar1x,bar1y,bar2x,bar2y,bar3x,bar3y);
    axis([-120,400,-120,200]);
    axis normal

    M(:,t)=getframe;
end
```

Initialization

```
1. % Set up the time interval and the initial positions of the nine coordinates
2. T_Int=0:0.01:2;
3. X0=[0 50 pi/2   125.86 132.55 0.2531   215.86 82.55 4.3026];
4. global T
5. Xinit=X0;
```

1. The sentence is notation that is behind symbol “%”.
2. Simulation time is set from 0 to 2 with $\Delta t = 0.01$.
3. Set the appropriate initial positions of the 9 coordinates which are used to solve nonlinear solver.
4. Declare a global variable T which is used to represent the current time t and determine the driving constraint for angular velocity.

Determine the displacement

```
10. [Xtemp,fval] = fsolve(@constrEq4bar,Xinit);
```

10. Call the nonlinear solver fsolve in which the constraint equations and initial values are necessary. The initial values is mentioned in above script. The constraint equations is written as a function (which can be treated a kind of subroutine in Matlab) as following and named as constrEq4bar. The fsolve finds a root of a system of nonlinear equations and adopts the trust-region dogleg algorithm by default.

```
a. function F=constrEq4bar(X)
b.
c. global T
d.
e. x1=X(1); y1=X(2); phi1=X(3);
f. x2=X(4); y2=X(5); phi2=X(6);
g. x3=X(7); y3=X(8); phi3=X(9);
h.
i. F=[ -x1+40*cos(phi1);
j.     -y1+40*sin(phi1);
k.     x1+40*cos(phi1)-x2+130*cos(phi2);
l.     y1+40*sin(phi1)-y2+130*sin(phi2);
m.     x2+130*cos(phi2)-x3+90*cos(phi3);
n.     y2+130*sin(phi2)-y3+90*sin(phi3);
o.     x3+90*cos(phi3)-180;
p.     y3+90*sin(phi3);
q.     phi1-2*pi*T-pi/2];
```

The equation of driving constraint is depended on current time T

Determine the velocity

```
14.     phi1=Xtemp(3); phi2=Xtemp(6); phi3=Xtemp(9);
15.     JacoMatrix=Jaco4bar(phi1,phi2,phi3);
16.     Beta=[0 0 0    0 0 0   0 0 2*pi]';
17.     Vtemp=JacoMatrix\Beta;
```

15. Call the function Jaco4bar to obtain the Jacobian Matrix depended on current values of displacement.
16. Declare the right-side of the velocity equations.
17. Solve linear equation by left matrix division “\” roughly the same as $\mathbf{J}^{-1}\boldsymbol{\beta}$. The algorithm adopts several methods such as LAPACK, CHOLMOD, and LU. Please find the detail in Matlab Help.

```
function JacoMatrix=Jaco4bar(phi1,phi2,phi3)

JacoMatrix=[    -1 0 -40*sin(phi1)      0 0 0                  0 0 0;
                0 -1 40*cos(phi1)       0 0 0                  0 0 0;
                1 0 -40*sin(phi1)       -1 0 -130*sin(phi2)    0 0 0;
                0 1 40*cos(phi1)        0 -1 130*cos(phi2)     0 0 0;
                0 0 0                   1 0 -130*sin(phi2)     -1 0 -90*sin(phi3);
                0 0 0                   0 1 130*cos(phi2)      0 -1 90*cos(phi3);
                0 0 0                   0 0 0                  1 0 -90*sin(phi3);
                0 0 0                   0 0 0                  0 1 90*cos(phi3);
                0 0 1                   0 0 0                  0 0 0];
```

Determine the acceleration

```
20.     dphi1=Vtemp(3); dphi2=Vtemp(6); dphi3=Vtemp(9);
21.     Gamma=Gamma4bar(phi1,phi2,phi3,dphi1,dphi2,dphi3);
22.     Atemp=JacoMatrix\Gamma;
```

21. Call the function Gamma4bar to obtain the right-side of the velocity equations depended on current values of velocity.
22. Solve linear equation to obtain the current acceleration.

```
function Gamma=Gamma4bar(phi1,phi2,phi3,dphi1,dphi2,dphi3)

Gamma=[ 40*cos(phi1)*dphi1^2;
        40*sin(phi1)*dphi1^2;
        40*cos(phi1)*dphi1^2+130*cos(phi2)*dphi2^2;
        40*sin(phi1)*dphi1^2+130*sin(phi2)*dphi2^2;
        130*cos(phi2)*dphi2^2+90*cos(phi3)*dphi3^2;
        130*sin(phi2)*dphi2^2+90*sin(phi3)*dphi3^2;
        90*cos(phi3)*dphi3^2;
        90*sin(phi3)*dphi3^2;
        0];
```

Determine next initial positions

```
29.    if Iter==1
30.        Xinit=X(:,Iter);
31.    else
32.        Xinit=X(:,Iter)+(X(:,Iter)-X(:,Iter-1));
33.    end
```

29.~33. Predict the next initial positions with assumption of inertia except the first time of the loop.

Plot time response

```
37.figure
38.for i=1:9;
39.    subplot(9,1,i)
40.    plot (T_Int,X(i,:))
41.    set(gca,'xtick',[], 'FontSize', 5)
42.end
43.% Reset the bottom subplot to have xticks
44.set(gca,'xtickMode', 'auto')
45.
46.% T vs velocity plot for the nine coordinates
47.figure
48.for i=1:9;
37.…
```

37. Create a blank figure .

39. Locate the position of subplot in the figure.

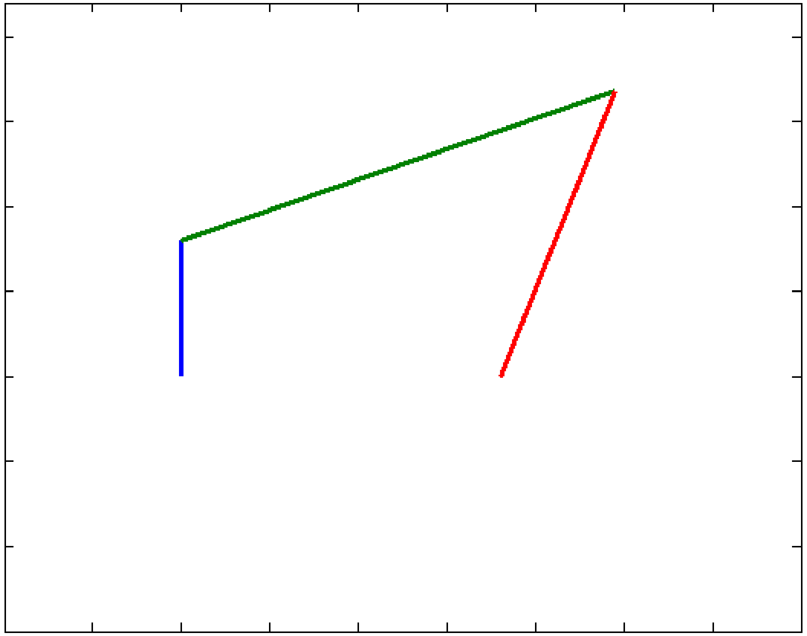
40. Plot the nine subplots for the time responses of nine coordinates.

41. Eliminate x-label for time-axis and set the font size of y-label.

44. Resume x-label at bottom because the nine subplots share the same time-axis.

47.~ It is similar to above.

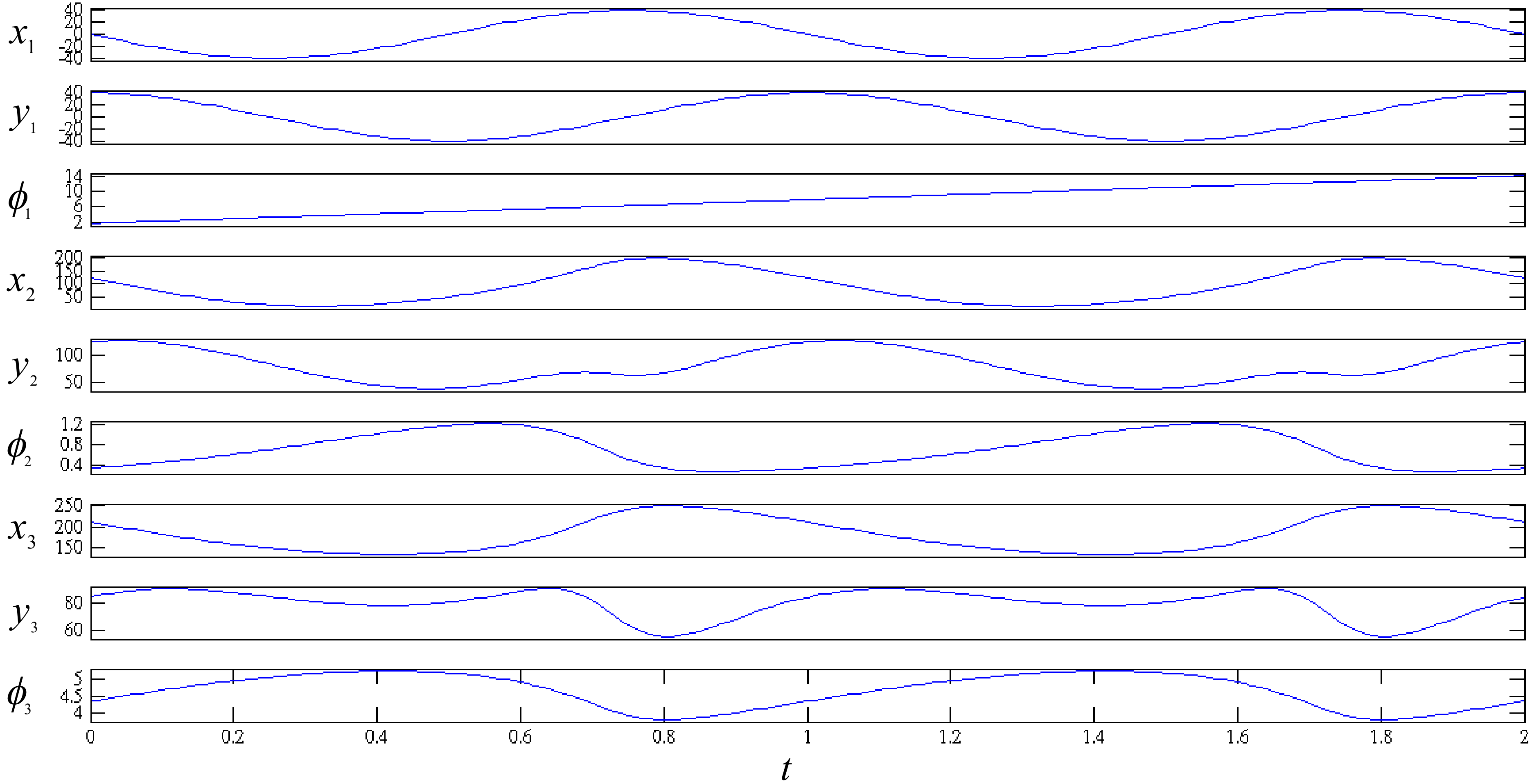
Animation



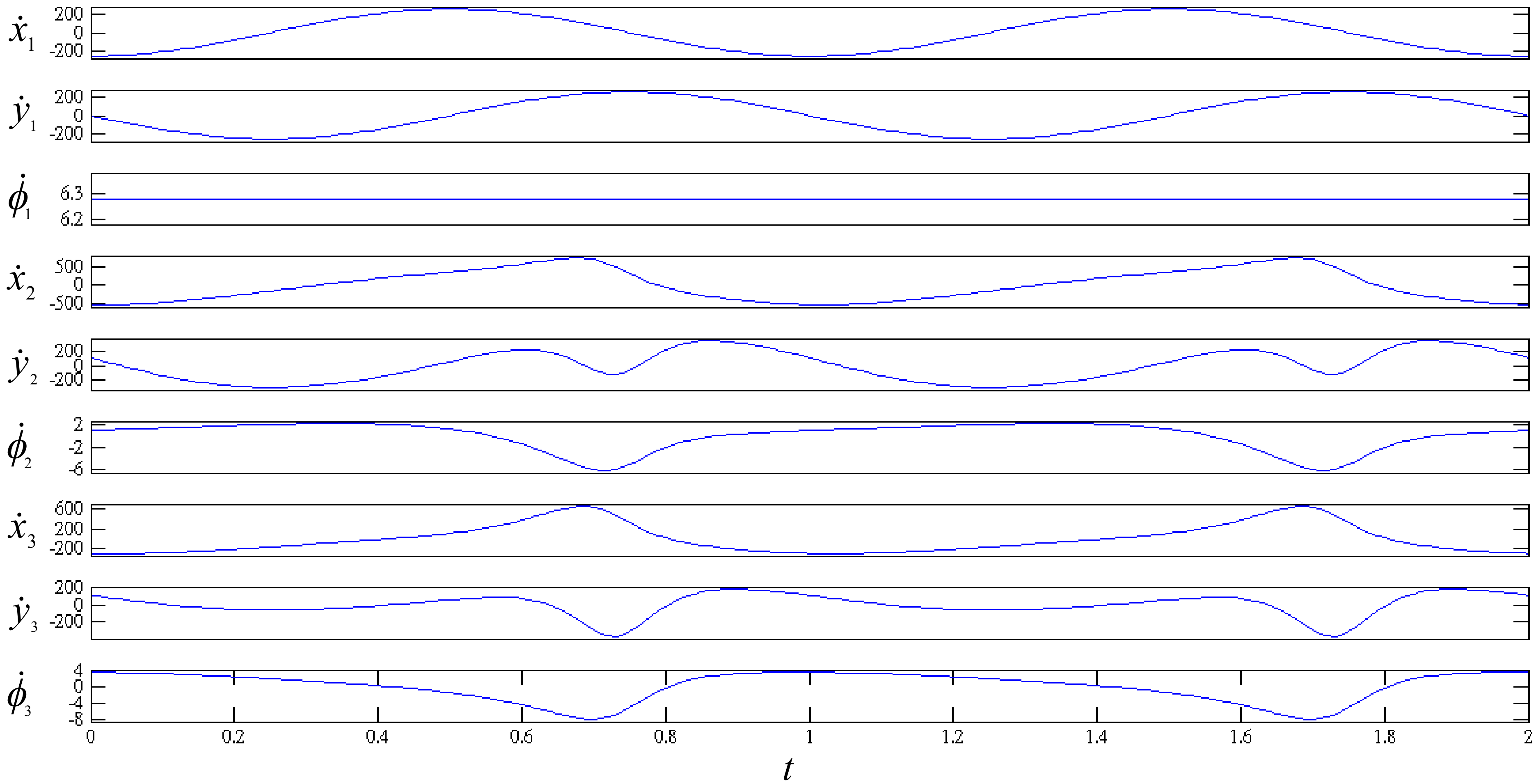
```
69.Ox=zeros(1,length(T_Int));
70.Oy=zeros(1,length(T_Int));
71.Ax=80*cos(X(3,:));
72.Ay=80*sin(X(3,:));
73.Bx=Ax+260*cos(X(6,:));
74.…

80.for t=1:length(T_Int);
81.    bar1x=[Ox(t) Ax(t)];
82.    bar1y=[Oy(t) Ay(t)];
83.    bar2x=[Ax(t) Bx(t)];
84.    bar2y=[Ay(t) By(t)];
85.    bar3x=[Bx(t) Cx(t)];
86.    bar3y=[By(t) Cy(t)];
87.
88.    plot (bar1x,bar1y,bar2x,bar2y,bar3x,bar3y)
89.    axis([-120,400,-120,200]);
90.    axis normal
91.
92.    M(:,t)=getframe;
93.end
```

69. Determine the displacement of revolute joint.
80. Repeat to plot the locations by continue time elapsing.
81. Determine the horizontal location of $\overline{OA}$.
88. Plot $\overline{OA}$, $\overline{AB}$, $\overline{BC}$, and $\overline{OC}$.
89. Set an appropriate range of axis.

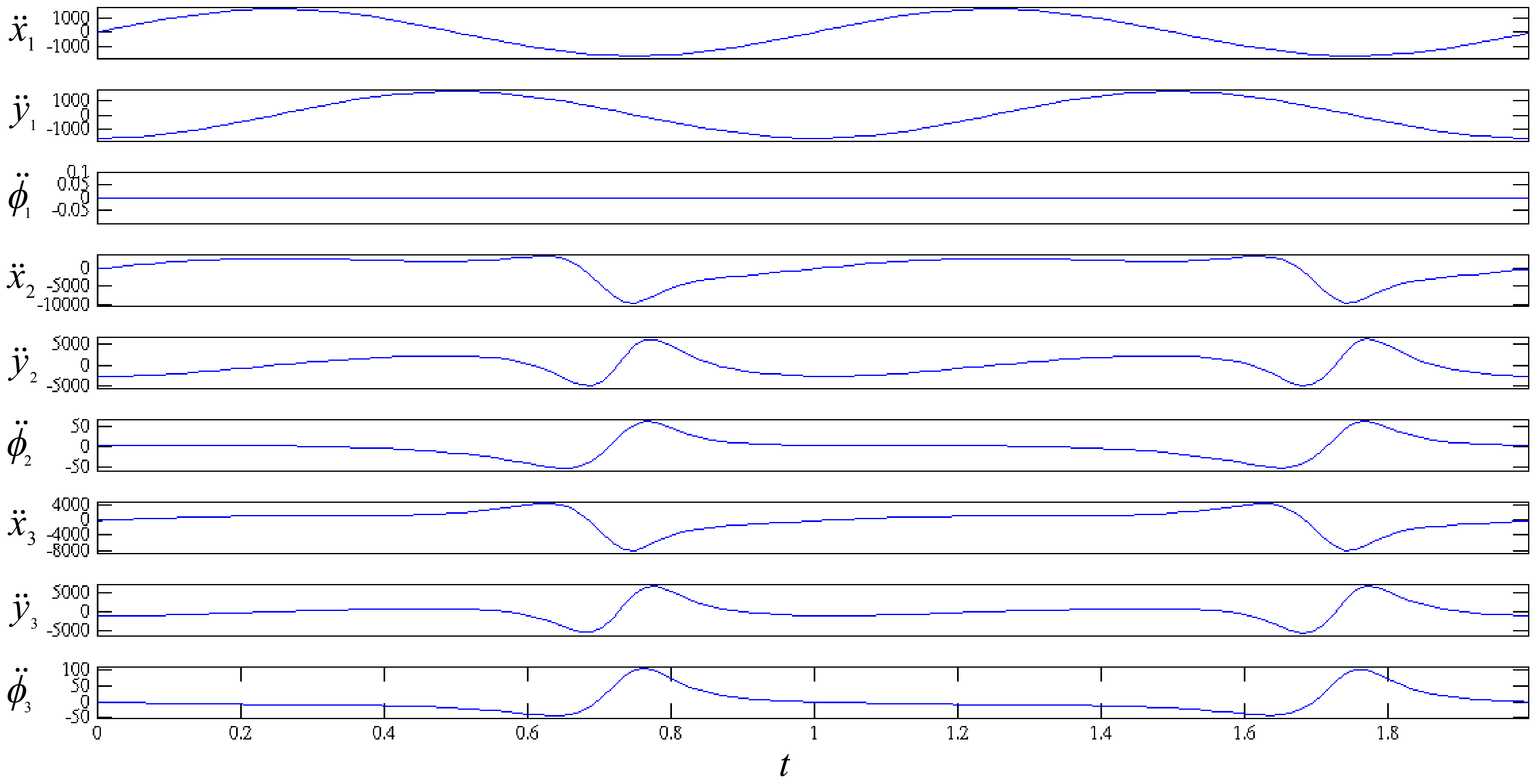
Time response of displacement



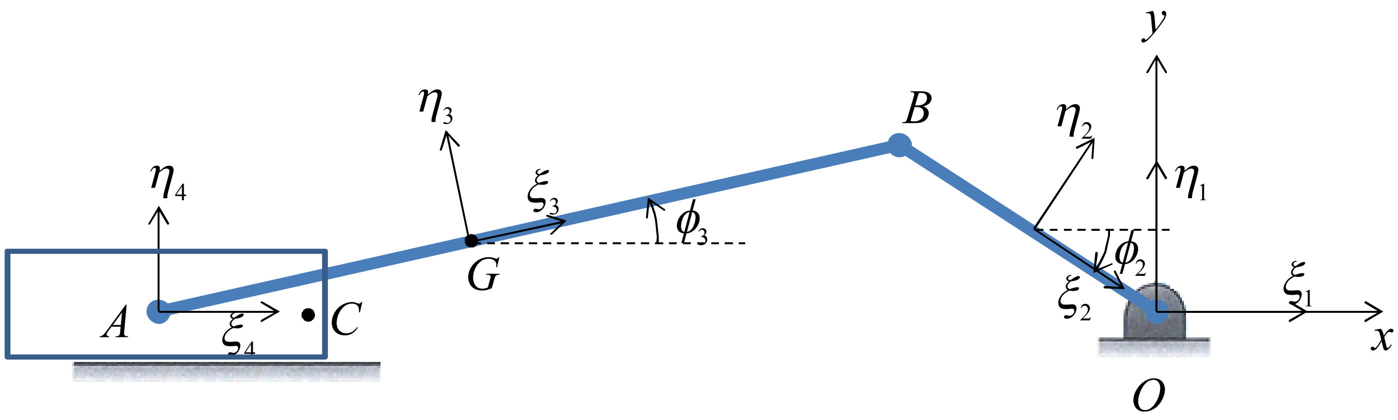
Time response of velocity



Time response of acceleration



Example of a slider-crank mechanism



Constraint equations :

$$x_1 = 0$$
$$y_1 = 0$$
$$\phi_1 = 0$$
$$x_4 - x_3 + 200\cos\phi_3 = 0$$
$$y_4 - y_3 + 200\sin\phi_3 = 0$$
$$x_3 + 300\cos\phi_3 - x_2 + 100\cos\phi_2 = 0$$
$$y_3 + 300\sin\phi_3 - y_2 + 100\sin\phi_2 = 0$$
$$x_2 + 100\cos\phi_2 - x_1 = 0$$
$$y_2 + 100\cos\phi_2 - y_1 = 0$$
$$100\cos\phi_4(y_1 - y_4 - 100\sin\phi_4) - 100\sin\phi_4(x_1 - x_4 - 100\cos\phi_4) = 0$$
$$\phi_4 - \phi_1 = 0$$
$$\phi_2 - 5.76 + 1.2t = 0$$

$\overline{AG} = 200\text{mm}$

$\overline{GB} = 300\text{mm}$

$\overline{BO} = 200\text{mm}$

To solve the 9 equations for 9 unknown $\boldsymbol{q}^T = [x_2, y_2, \phi_2, x_3, y_3, \phi_3, x_4, y_4, \phi_4]$

The Jacobian matrix and the right-side of the velocity equations

$\langle 1\rangle,\langle 2\rangle,\langle 3\rangle,\langle 6\rangle,\langle 10\rangle,\langle 14\rangle,\langle 18\rangle,\langle 22\rangle,\langle 26\rangle,\langle 34\rangle,\langle 36\rangle = 1$

$\langle 4\rangle,\langle 8\rangle,\langle 12\rangle,\langle 16\rangle,\langle 20\rangle,\langle 24\rangle,\langle 35\rangle = -1$

$\langle 5\rangle = -200\sin\phi_3$

$\langle 9\rangle = 200\cos\phi_3$

$\langle 13\rangle,\langle 23\rangle = -100\sin\phi_2$

$\langle 15\rangle = -300\sin\phi_3$

$\langle 17\rangle,\langle 27\rangle = 100\cos\phi_2$

$\langle 19\rangle = 300\cos\phi_3$

$\langle 28\rangle = -100\sin\phi_4$

$\langle 29\rangle = 100\cos\phi_4$

$\langle 31\rangle = -\langle 28\rangle$

$\langle 32\rangle = -\langle 29\rangle$

$\langle 33\rangle = 100[\cos\phi_4(x_4 - x_1) + \sin\phi_4(y_4 - y_1)]$

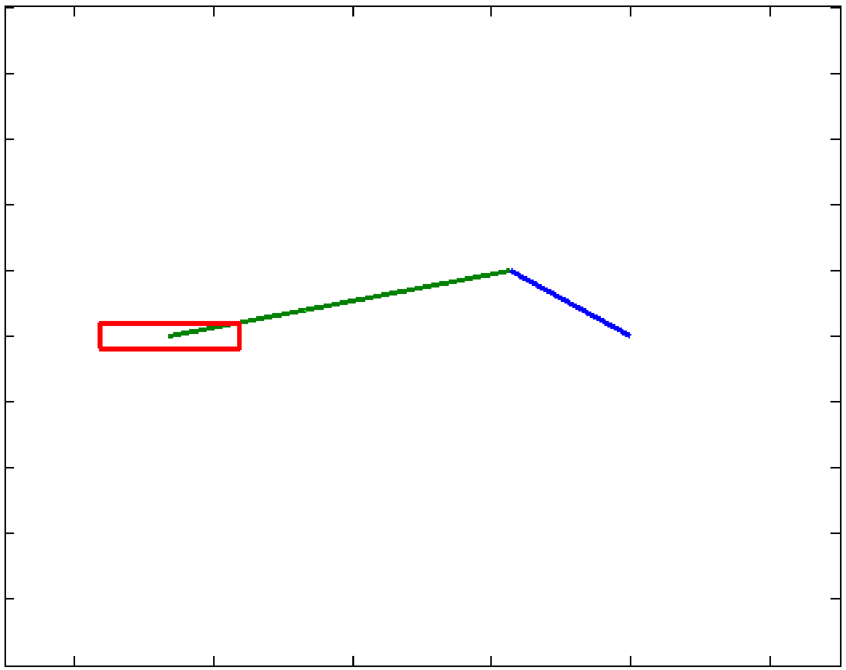
$$
\mathbf{J} = \begin{array}{c} \partial\Phi_1/\partial \\ \partial\Phi_2/\partial \\ \partial\Phi_3/\partial \\ \partial\Phi_4/\partial \\ \partial\Phi_5/\partial \\ \partial\Phi_6/\partial \\ \partial\Phi_7/\partial \\ \partial\Phi_8/\partial \\ \partial\Phi_9/\partial \\ \partial\Phi_{10}/\partial \\ \partial\Phi_{11}/\partial \\ \partial\Phi_{12}/\partial \end{array}
\begin{bmatrix}
\langle 1\rangle & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \langle 2\rangle & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \langle 3\rangle & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \langle 4\rangle & 0 & \langle 5\rangle & \langle 6\rangle & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \langle 8\rangle & \langle 9\rangle & 0 & \langle 10\rangle & 0 \\
0 & 0 & 0 & \langle 12\rangle & 0 & \langle 13\rangle & \langle 14\rangle & 0 & \langle 15\rangle & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \langle 16\rangle & \langle 17\rangle & 0 & \langle 18\rangle & \langle 19\rangle & 0 & 0 & 0 \\
\langle 20\rangle & 0 & 0 & \langle 22\rangle & 0 & \langle 23\rangle & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \langle 24\rangle & 0 & 0 & \langle 26\rangle & \langle 27\rangle & 0 & 0 & 0 & 0 & 0 & 0 \\
\langle 28\rangle & \langle 29\rangle & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \langle 31\rangle & \langle 32\rangle & \langle 33\rangle \\
0 & 0 & \langle 34\rangle & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \langle 35\rangle \\
0 & 0 & 0 & 0 & 0 & \langle 36\rangle & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\qquad
\beta = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1.2 \end{bmatrix}
$$

The right-side of the acceleration equations

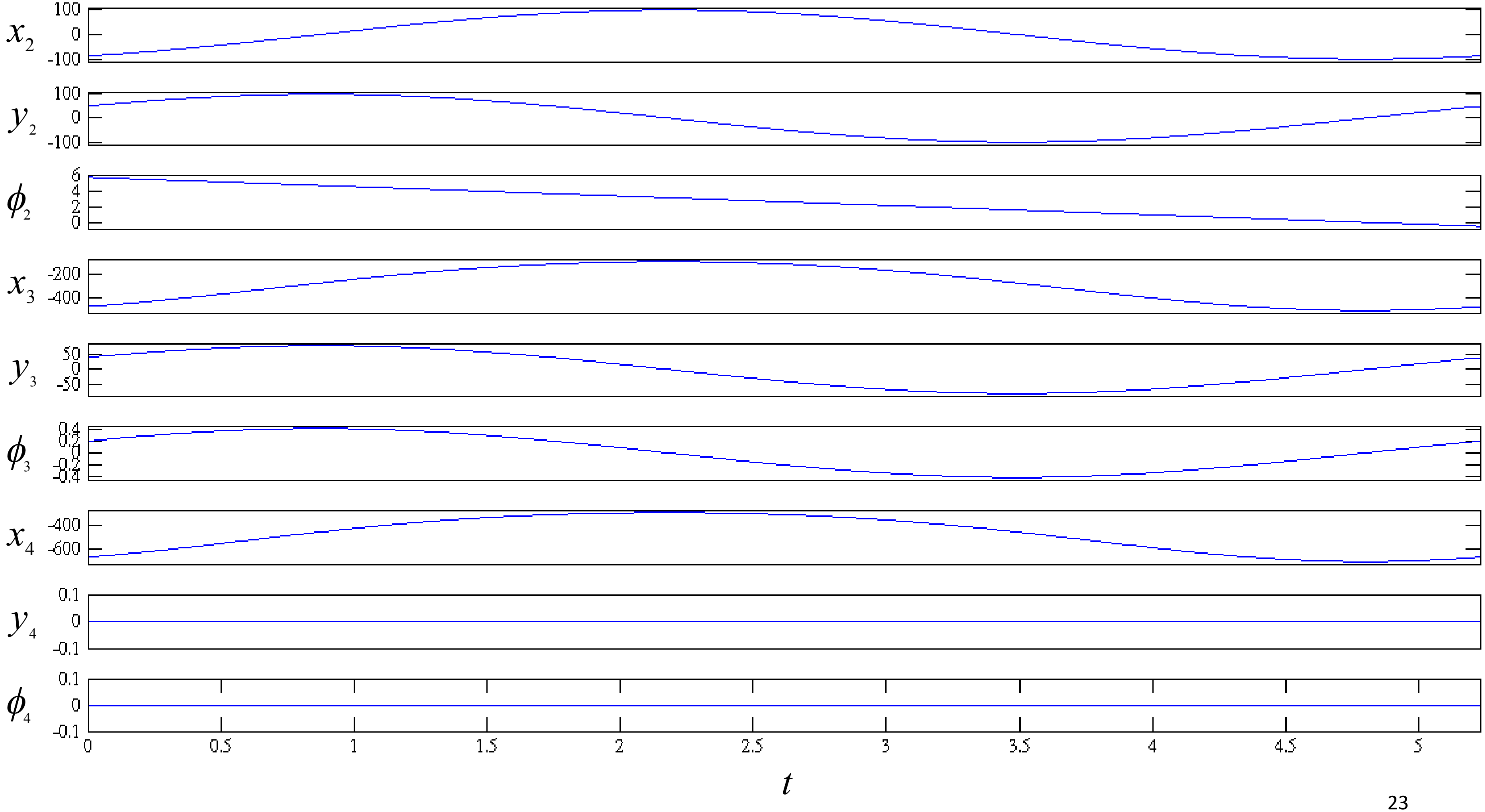
$$\boldsymbol{\gamma} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 200\cos\phi_3\dot{\phi}_3^{\,2} \\ 200\sin\phi_3\dot{\phi}_3^{\,2} \\ 300\cos\phi_3\dot{\phi}_3^{\,2} + 100\cos\phi_2\dot{\phi}_2^{\,2} \\ 300\sin\phi_3\dot{\phi}_3^{\,2} + 100\sin\phi_2\dot{\phi}_2^{\,2} \\ 100\cos\phi_2\dot{\phi}_2^{\,2} \\ 100\sin\phi_2\dot{\phi}_2^{\,2} \\ \gamma(10) \\ 0 \\ 0 \end{bmatrix}$$

where $\gamma(10) = 200\cos\phi_4(\dot{x}_1 - \dot{x}_4)\dot{\phi}_4 + 200\sin\phi_4(\dot{y}_1 - \dot{y}_4)\dot{\phi}_4 - \dot{\phi}_4^{\,2}\left[100\sin\phi_4(x_1 - x_4) - 100\cos\phi_4(y_1 - y_4)\right]$

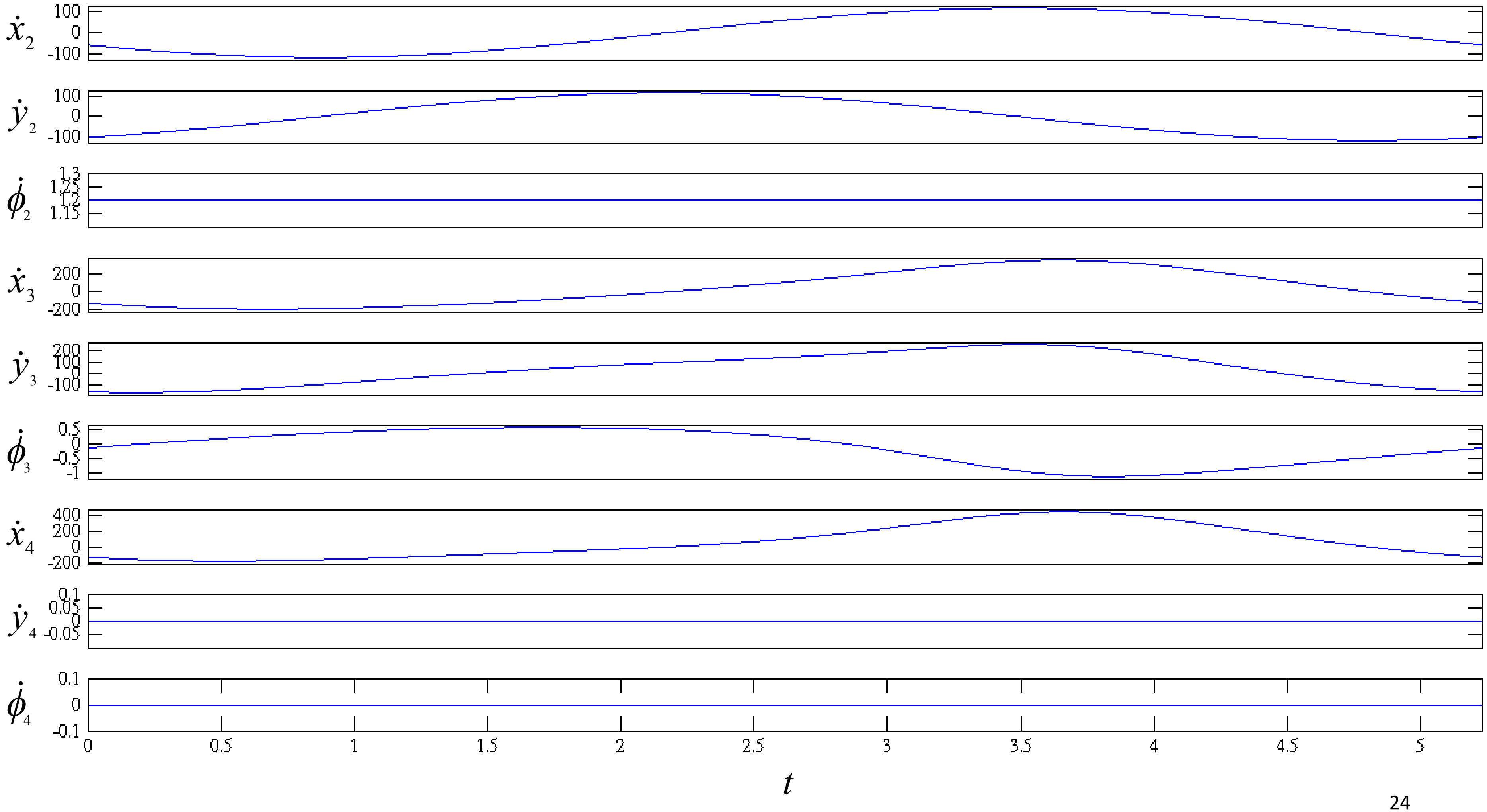
Animation



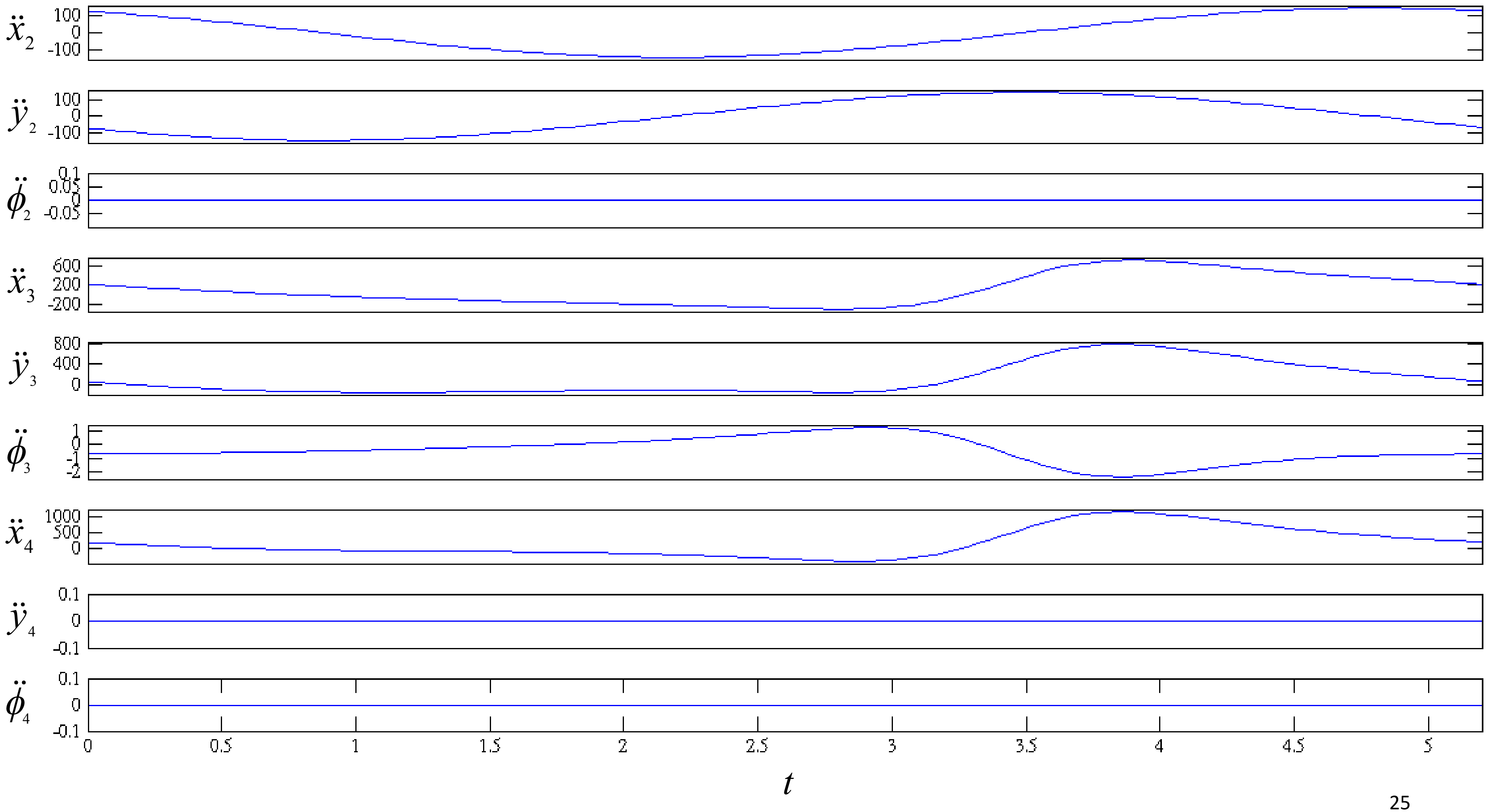
Time response of displacement



Time response of velocity



Time response of acceleration



Main program: kinematic4bar

```
% This script of matlab m.file is used to solve the kinematic problem of a
% four-bar linkage for the mechanical dynamics course.

% Designer:                        C. Y. Chuang
% Proto-type:                       13-July-2012
% Adviser:                         Prof. Yang

clear all
clc
% Set up the time interval and the initial positions of the nine coordinates
T_Int=0:0.01:2;
X0=[0 40 pi/2   125.86 132.55 0.2531   215.86 82.55 4.3026];
global T
Xinit=X0;

% Do the loop for each time interval
for Iter=1:length(T_Int);
    T=T_Int(Iter);
    % Determine the position at the current time
    [Xtemp,fval] = fsolve(@constrEq4bar,Xinit);

    % Determine the velocity at the current time
    phi1=Xtemp(3); phi2=Xtemp(6); phi3=Xtemp(9);
    JacoMatrix=Jaco4bar(phi1,phi2,phi3);
    Beta=[0 0 0    0 0 0    0 0 2*pi]';
    Vtemp=JacoMatrix\Beta;

    % Determine the acceleration at the current time
    dphi1=Vtemp(3); dphi2=Vtemp(6); dphi3=Vtemp(9);
    Gamma=Gamma4bar(phi1,phi2,phi3,dphi1,dphi2,dphi3);
    Atemp=JacoMatrix\Gamma;

    % Record the results of each iteration
    X(:,Iter)=Xtemp; V(:,Iter)=Vtemp; A(:,Iter)=Atemp;
```

```
    % Determine the new initial position to solve the equation of the next
    % iteration and assume that the kinematic motion is with inertia
    if Iter==1
        Xinit=X(:,Iter);
    else
        Xinit=X(:,Iter)+(X(:,Iter)-X(:,Iter-1));
    end

end

% T vs displacement plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,X(i,:),'linewidth',1)
    AxeSup=max(X(i,:));
    AxeInf=min(X(i,:));
    AxeSpac=0.05*(AxeSup-AxeInf);
    if AxeSup-AxeInf<0.01
        axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
    else
        axis([-inf,inf,AxeInf-AxeSpac,AxeSup+AxeSpac]);
    end
    set(gca,'xtick',[], 'FontSize', 5,'FontName','timesnewroman')
end
% Reset the bottom subplot to have xticks
set(gca,'xtickMode', 'auto')

% T vs velocity plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,V(i,:),'linewidth',1)
    AxeSup=max(V(i,:));
    AxeInf=min(V(i,:));
    AxeSpac=0.05*(AxeSup-AxeInf);
    if AxeSup-AxeInf<0.01
        axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
    else
        axis([-inf,inf,AxeInf-AxeSpac,AxeSup+AxeSpac]);
```

```
    end
    set(gca,'xtick',[], 'FontSize', 5,'FontName','timesnewroman')
end
set(gca,'xtickMode', 'auto')

% T vs acceleration plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,A(i,:),'linewidth',1)
    AxeSup=max(A(i,:));
    AxeInf=min(A(i,:));
    AxeSpac=0.05*(AxeSup-AxeInf);
    if AxeSup-AxeInf<0.01
        axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
    else
        axis([-inf,inf,AxeInf-AxeSpac,AxeSup+AxeSpac]);
    end
    set(gca,'xtick',[], 'FontSize', 5,'FontName','timesnewroman')
end
set(gca,'xtickMode', 'auto')

% Determine the positions of the four revolute joints at each iteration
Ox=zeros(1,length(T_Int));
Oy=zeros(1,length(T_Int));
Ax=80*cos(X(3,:));
Ay=80*sin(X(3,:));
Bx=Ax+260*cos(X(6,:));
By=Ay+260*sin(X(6,:));
Cx=180*ones(1,length(T_Int));
Cy=zeros(1,length(T_Int));

% Animation
figure
for t=1:length(T_Int);
    bar1x=[Ox(t) Ax(t)];
    bar1y=[Oy(t) Ay(t)];
    bar2x=[Ax(t) Bx(t)];
    bar2y=[Ay(t) By(t)];
    bar3x=[Bx(t) Cx(t)];
```

```
    bar3y=[By(t) Cy(t)];

    plot (bar1x,bar1y,bar2x,bar2y,bar3x,bar3y);
    axis([-100,350,-150,220]);
    axis normal

    M(:,t)=getframe;
end

movie2avi(M,'Kine4bar.avi','compression','None');
```

Function: constrEq4bar

```
function F=constrEq4bar(X)

global T

x1=X(1); y1=X(2); phi1=X(3);
x2=X(4); y2=X(5); phi2=X(6);
x3=X(7); y3=X(8); phi3=X(9);

F=[ -x1+40*cos(phi1);
    -y1+40*sin(phi1);
    x1+40*cos(phi1)-x2+130*cos(phi2);
    y1+40*sin(phi1)-y2+130*sin(phi2);
    x2+130*cos(phi2)-x3+90*cos(phi3);
    y2+130*sin(phi2)-y3+90*sin(phi3);
    x3+90*cos(phi3)-180;
    y3+90*sin(phi3);
    phi1-2*pi*T-pi/2];
```

Function: Jaco4bar

```
function JacoMatrix=Jaco4bar(phi1,phi2,phi3)

JacoMatrix=[  -1  0  -40*sin(phi1)      0  0  0                  0  0  0;
              0  -1  40*cos(phi1)       0  0  0                  0  0  0;
              1  0  -40*sin(phi1)       -1  0  -130*sin(phi2)    0  0  0;
              0  1  40*cos(phi1)        0  -1  130*cos(phi2)     0  0  0;
              0  0  0                   1  0  -130*sin(phi2)     -1  0  -90*sin(phi3);
              0  0  0                   0  1  130*cos(phi2)      0  -1  90*cos(phi3);
              0  0  0                   0  0  0                  1  0  -90*sin(phi3);
              0  0  0                   0  0  0                  0  1  90*cos(phi3);
              0  0  1                   0  0  0                  0  0  0];
```

Function: Gamma4bar

```
function Gamma=Gamma4bar(phi1,phi2,phi3,dphi1,dphi2,dphi3)

Gamma=[40*cos(phi1)*dphi1^2;
       40*sin(phi1)*dphi1^2;
       40*cos(phi1)*dphi1^2+130*cos(phi2)*dphi2^2;
       40*sin(phi1)*dphi1^2+130*sin(phi2)*dphi2^2;
       130*cos(phi2)*dphi2^2+90*cos(phi3)*dphi3^2;
       130*sin(phi2)*dphi2^2+90*sin(phi3)*dphi3^2;
       90*cos(phi3)*dphi3^2;
       90*sin(phi3)*dphi3^2;
       0];
```

Main program: kinematicSC

```
% This script of matlab m.file is used to solve the kinematic problem of a
% slider-crank mechanism for the mechanical dynamics course.

% Designer:                     C. Y. Chuang
% Proto-type:                    13-July-2012
% Adviser:                      Prof. Yang

clear all
clc
% Set up the time interval and the initial positions of the nine coordinates
T_Int=0:0.01:2*pi/1.2;
X0=[0 0 0    0 0 5.76   0 0 0    -500 0 0];;
global T
Xinit=X0;

% Do the loop for each time interval
for Iter=1:length(T_Int);
    T=T_Int(Iter);
    % Determine the position at the current time
    [Xtemp,fval] = fsolve(@constrEqSC,Xinit);

    % Determine the velocity at the current time
    x1=Xtemp(1); y1=Xtemp(2); phi1=Xtemp(3);
    phi2=Xtemp(6);
    phi3=Xtemp(9);
    x4=Xtemp(10); y4=Xtemp(11); phi4=Xtemp(12);
    JacoMatrix=JacoSC(x1,y1,phi1, phi2, phi3,x4,y4,phi4);
    Beta=[  0 0 0   0 0 0   0 0 0   0 0 1.2]';
    Vtemp=JacoMatrix\Beta;

    % Determine the acceleration at the current time
    dx1=Vtemp(1); dy1=Vtemp(2); dphi1=Vtemp(3);
    dphi2=Vtemp(6);
    dphi3=Vtemp(9);
    dx4=Vtemp(10); dy4=Vtemp(11); dphi4=Vtemp(12);
    Gamma=GammaSC(x1,y1,phi2,phi3,x4,y4,phi4,dx1,dy1,dphi2,dphi3,dx4,dy4,dphi4);
```

```
    Atemp=JacoMatrix\Gamma;

    % Record the results of each iteration
    X(:,Iter)=Xtemp; V(:,Iter)=Vtemp; A(:,Iter)=Atemp;

    % Determine the new initial position to solve the equation of the next
    % iteration and assume that the kinematic motion is with inertia
    if Iter==1
       Xinit=X(:,Iter);
    else
       Xinit=X(:,Iter)+(X(:,Iter)-X(:,Iter-1));
    end

end

% T vs displacement plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,X(i+3,:),'linewidth',1)
    AxeSup=max(X(i+3,:));
    AxeInf=min(X(i+3,:));
    AxeSpac=0.05*(AxeSup-AxeInf);
    if AxeSup-AxeInf<0.01
       axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
    else
       axis([-inf,inf,AxeInf-AxeSpac,AxeSup+AxeSpac]);
    end
    set(gca,'xtick',[], 'FontSize', 5,'FontName','timesnewroman')
end
% Reset the bottom subplot to have xticks
set(gca,'xtickMode', 'auto')

% T vs velocity plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,V(i+3,:),'linewidth',1)
    AxeSup=max(V(i+3,:));
    AxeInf=min(V(i+3,:));
```

```
    AxeSpac=0.05*(AxeSup-AxeInf);
    if AxeSup-AxeInf<0.01
        axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
    else
        axis([-inf,inf,AxeInf-AxeSpac,AxeSup+AxeSpac]);
    end
    set(gca,'xtick',[], 'FontSize', 5,'FontName','timesnewroman')
end
set(gca,'xtickMode', 'auto')

% T vs acceleration plot for the nine coordinates
figure
for i=1:9;
    subplot(9,1,i)
    plot (T_Int,A(i+3,:),'linewidth',1)
    AxeSup=max(A(i+3,:));
    AxeInf=min(A(i+3,:));
    AxeSpac=0.05*(AxeSup-AxeInf);
    if AxeSup-AxeInf<0.01
        axis([-inf,inf,(AxeSup+AxeSup)/2-0.1 (AxeSup+AxeSup)/2+0.1]);
    else
        axis([-inf,inf,AxeInf-AxeSpac,AxeSup+AxeSpac]);
    end
    set(gca,'xtick',[], 'FontSize', 5,'FontName','timesnewroman')
end
set(gca,'xtickMode', 'auto')

% Determine the positions of the four revolute joints at each iteration
Ox=zeros(1,length(T_Int));
Oy=zeros(1,length(T_Int));
Bx=-200*cos(X(6,:));
By=-200*sin(X(6,:));
Ax=Bx-500*cos(X(9,:));
Ay=By-500*sin(X(9,:));

% Animation
figure
for t=1:length(T_Int);
    bar2x=[Ox(t) Bx(t)];
    bar2y=[Oy(t) By(t)];
```

```
    bar3x=[Bx(t) Ax(t)];
    bar3y=[By(t) Ay(t)];
    blockx=[Ax(t)-100 Ax(t)+100 Ax(t)+100 Ax(t)-100 Ax(t)-100];
    blocky=[Ay(t)-20 Ay(t)-20 Ay(t)+20 Ay(t)+20 Ay(t)-20];

    plot (bar2x,bar2y,bar3x,bar3y,blockx,blocky);
    axis([-900,300,-500,500]);
    axis normal

    M(:,t)=getframe;
end

movie2avi(M,'KineSC.avi','compression','None');
```

Function: constrEqSC

```
function F=constrEqSC(X)

global T

x1=X(1);    y1=X(2);    phi1=X(3);
x2=X(4);    y2=X(5);    phi2=X(6);
x3=X(7);    y3=X(8);    phi3=X(9);
x4=X(10);   y4=X(11);   phi4=X(12);

F=[ x1;
    y1;
    phi1;
    x4-x3+200*cos(phi3);
    y4-y3+200*sin(phi3);
    x3+300*cos(phi3)-x2+100*cos(phi2);
    y3+300*sin(phi3)-y2+100*sin(phi2);
    x2+100*cos(phi2)-x1;
    y2+100*sin(phi2)-y1;
    (100*cos(phi4))*(y1-y4-100*sin(phi4))-(100*sin(phi4))*(x1-x4-100*cos(phi4))
    phi4-phi1;
    phi2-5.76+1.2*T];
```

Function: JacoSC

```
function JacoMatrix=JacoSC(x1,y1,phi1, phi2, phi3,x4,y4,phi4)

% The variable E means the element which is a vector of nonzero values in
% the sparse matrix JacoMatrix.

E([1 2 3 6 10 14 18 22 26 34 36])=1;
E([4 8 12 16 20 24 35])=-1;
E(5)=-200*sin(phi3);
E(9)=200*cos(phi3);
E([13 23])=-100*sin(phi2);
E(15)=-300*sin(phi3);
E([17 27])=100*cos(phi2);
E(19)=300*cos(phi3);
E(28)=-100*sin(phi4);
E(29)=100*cos(phi4);
E(31)=-E(28);
E(32)=-E(29);
E(33)=-100*( cos(phi4)*(x1-x4) +cos(phi4)*(y1-y4) );

JacoMatrix=[ E(1) 0 0        0 0 0          0 0 0          0 0 0;
             0 E(2) 0        0 0 0          0 0 0          0 0 0;
             0 0 E(3)        0 0 0          0 0 0          0 0 0;
             0 0 0           0 0 0          E(4) 0 E(5)    E(6) 0 0;
             0 0 0           0 0 0          0 E(8) E(9)    E(10) 0 0;
             0 0 0           E(12) 0 E(13)  E(14) 0 E(15)  0 0 0;
             0 0 0           0 E(16) E(17)  0 E(18) E(19)  0 0 0;
             E(20) 0 0       E(22) 0 E(23)  0 0 0          0 0 0;
             0 E(24) 0       0 E(26) E(27)  0 0 0          0 0 0;
             E(28) E(29) 0   0 0 0          0 0 0          E(31) E(32) E(33);
             0 0 E(34)       0 0 0          0 0 0          0 0 E(35);
             0 0 0           0 0 E(36)      0 0 0          0 0 0];
```

Function: GammaSC

```
function Gamma=GammaSC(x1,y1,phi2,phi3,x4,y4,phi4,dx1,dy1,dphi2,dphi3,dx4,dy4,dphi4)

gamma10=200*cos(phi4)*(dx1-dx4)*dphi4...
       +200*sin(phi4)*(dy1-dy4)*dphi4...
       -dphi4^2*(100*sin(phi4)*(x1-x4)-100*cos(phi4)*(y1-y4));

Gamma=[    0;
           0;
           0;
           200*cos(phi3)*dphi3^2;
           200*sin(phi3)*dphi3^2;
           300*cos(phi3)*dphi3^2+100*cos(phi2)*dphi2^2;
           300*sin(phi3)*dphi3^2+100*sin(phi2)*dphi2^2;
           100*cos(phi2)*dphi2^2;
           100*sin(phi2)*dphi2^2;
           gamma10;
           0;
           0];
```