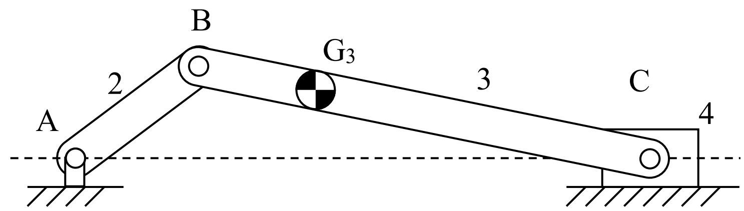
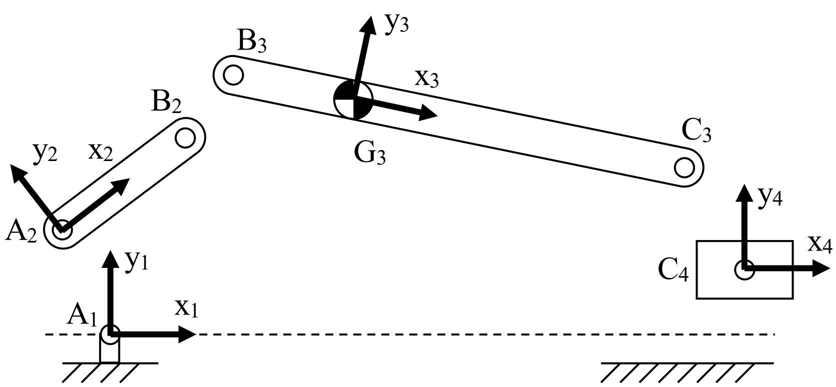
ME 481 - H06 **Name ________________________**

Use generalized coordinates {q} and joint constraints {Φ} for the slider crank shown below.

$$\{q\} = \begin{Bmatrix} x_2 \\ y_2 \\ \phi_2 \\ x_3 \\ y_3 \\ \phi_3 \\ x_4 \\ y_4 \\ \phi_4 \end{Bmatrix} = \begin{Bmatrix} \{r_2\}^G \\ \phi_2 \\ \{r_3\}^G \\ \phi_3 \\ \{r_4\}^G \\ \phi_4 \end{Bmatrix} \qquad \{\Phi\} = \begin{Bmatrix} \{r_2\}^A - \{r_1\}^A \\ \{r_3\}^B - \{r_2\}^B \\ \{r_4\}^C - \{r_3\}^C \\ \phi_4 \\ y_4 \\ \phi_2 - \omega_2 t \end{Bmatrix}$$



AB = R = 0.985 inch
BC = L = 4.33 inch
BG_3 = 1.1 inch
G_2 is at A_2 (balanced crank)
G_3 is on centerline of link 3
G_4 is at C_4 (simple piston model)
x_2 axis along centerline of link 2
x_3 axis along centerline of link 3
ω_2 = 1000 rpm CCW constant



BLUEPRINT INFORMATION

$$\{s_2\}'^A = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \qquad \{s_2\}'^B = \begin{Bmatrix} R \\ 0 \end{Bmatrix} \qquad \{s_3\}'^B = \begin{Bmatrix} -BG_3 \\ 0 \end{Bmatrix} \qquad \{s_3\}'^C = \begin{Bmatrix} L - BG_3 \\ 0 \end{Bmatrix}$$

$$\{s_4\}'^C = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \qquad \{r_1\}^A = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

example for B3 $\{r_3\}^B = \{r_3\} + [A_3]\{s_3\}'^B$ $\quad [A_3] = \begin{bmatrix} \cos\phi_3 & -\sin\phi_3 \\ \sin\phi_3 & \cos\phi_3 \end{bmatrix}$

ME 481 - H06 **Name** ______________________

1) Evaluate residuals $\{\Phi\}$ for rough estimates of generalized coordinates $\{q\}$ at t = 0.005 sec shown below. Comment on the relative precision of $\{\Phi\}$ versus $\{q\}$. Attach hardcopy of code.

$$\{q\} = \begin{Bmatrix} 0 \text{ inch} \\ 0 \text{ inch} \\ 0.4363 \text{ rad } (25°) \\ 2.0 \text{ inch} \\ 0.5 \text{ inch} \\ -0.1745 \text{ rad } (-10°) \\ 5.0 \text{ inch} \\ 0 \text{ inch} \\ 0 \text{ rad } (0°) \end{Bmatrix} \qquad \{\Phi\} = \begin{Bmatrix} 0 \text{ in} \\ 0 \text{ in} \\ 0.0240 \text{ in} \\ 0.2747 \text{ in} \\ -0.1809 \text{ in} \\ 0.0609 \text{ in} \\ 0 \text{ rad} \\ 0 \text{ in} \\ -0.0873 \text{ rad} \end{Bmatrix}$$

2) Use geometric equations to determine better estimates for $\{q\}$ at time t = 0.005 sec. Then evaluate new residuals. Comment on precision of $\{q\}$ and $\{\Phi\}$ between parts 1) and 2). Attach hardcopy of code.

$$\{q\} = \begin{Bmatrix} 0 \text{ inch} \\ 0 \text{ inch} \\ \omega_2 t = 0.5236 \text{rad} \\ 1.946 \text{ in} \\ 0.367 \text{ in} \\ -0.114 \text{rad } (-6.53°) \\ 5.155 \text{ in} \\ 0 \text{ inch} \\ 0 \text{ rad } (0°) \end{Bmatrix} \qquad \{\Phi\} = \begin{Bmatrix} 0 \text{ in} \\ 0 \text{ in} \\ 0.0001013 \text{ in} \\ -0.0004042 \text{ in} \\ -0.0000452 \text{ in} \\ 0.0003267 \text{ in} \\ 0 \text{ rad} \\ 0 \text{ in} \\ 0 \text{ rad} \end{Bmatrix}$$

ω_2 = 1000 rpm = 104.72 rad/s
$\phi_2 = \omega_2 t$ = 0.5236 rad = 30°

$\theta = \phi_2$
$R \sin\theta = L \sin\phi$ $\quad \phi = 6.53°$
$s = R\cos\theta + L\cos\phi$ = 5.115 in
$\phi_3 = -\phi$ $\quad x_4 = s$

$x_3^G = R\cos\theta + (BG_3)\cos\phi$ = 1.946 in
$y_3^G = R\sin\theta - (BG_3)\sin\phi$ = 0.367 in

3) Evaluate the Jacobian $\lfloor \Phi_q \rfloor$ for your better estimate of $\{q\}$ at t = 0.005 sec. Attach hardcopy of code.

$$[\Phi_q] = \begin{bmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & +0.4925 & +1 & 0 & -0.1251 & 0 & 0 & 0 \\ 0 & -1 & -0.8530 & 0 & +1 & -1.0929 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -0.3674 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -3.2090 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

det JAC = -4.3019

4) Use your code to perform a Newton-Raphson position solution at t = 0.010 sec. Calculate piston position x_4 and determinant of the Jacobian. Validate with geometric equations. Attach hardcopy of code.

x_4 (Newton-Raphson) 4.7376 inch $\det\lfloor\Phi_q\rfloor$ -4.2451

x_4 (geometric) 4.7376 inch

5) Compute piston velocity $\dot{x}_4$ and acceleration $\ddot{x}_4$ at t = 0.010 sec using a matrix solution with right-hand-side (RHS) vectors $\{\nu\}$ and $\{\gamma\}$. Validate with geometric equations. Attach hardcopy of code.

$\dot{x}_4$ (matrix) -99.69 ips $\ddot{x}_4$ (matrix) -4173 ips^2

$\dot{x}_4$ (geometric) -99.69 ips $\ddot{x}_4$ (geometric) -4173 ips^2

EXTRA CREDIT

Place a loop around your solution for part 5) using $0 \le t \le 0.06$ sec and provide MATLAB graphs for piston position x_4, velocity $\dot{x}_4$ and acceleration $\ddot{x}_4$ as functions of crank angle ϕ_2. Validate using results from geometric equations on the same MATLAB graphs.

EXTRA EXTRA CREDIT

Modify your slider crank code for part 5) to analyze the four bar in Notes_04_05. This should only require modifying the last three rows in your constraint vector, your Jacobian matrix and your acceleration RHS vector.

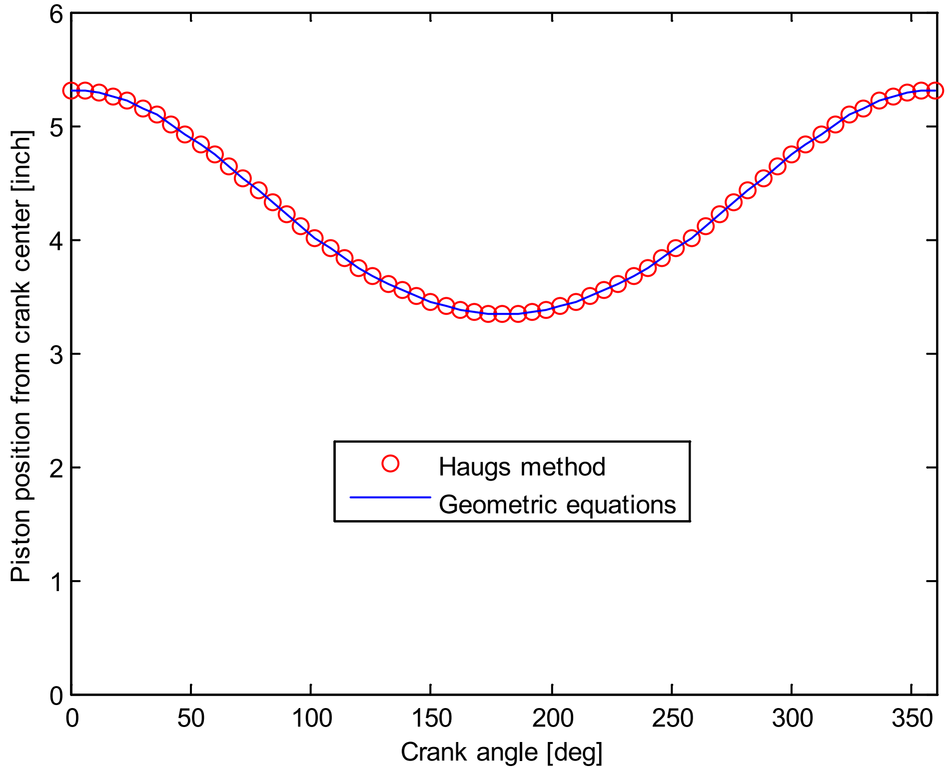
use	$\phi_2 = 65°$	$\dot{\phi}_2 = 10$ rad/sec CW	$\ddot{\phi}_2 = 2$ rad/sec^2 CCW
validation	$\phi_3 = 13.151°$	$\dot{\phi}_3 =$ +3.9013 rad/sec	$\ddot{\phi}_3 = +7.0627$ rad/sec^2
	$\phi_4 = -65.173°$	$\dot{\phi}_4 = -5.3533$ rad/sec	$\ddot{\phi}_4 =$ +69.7682 rad/sec^2

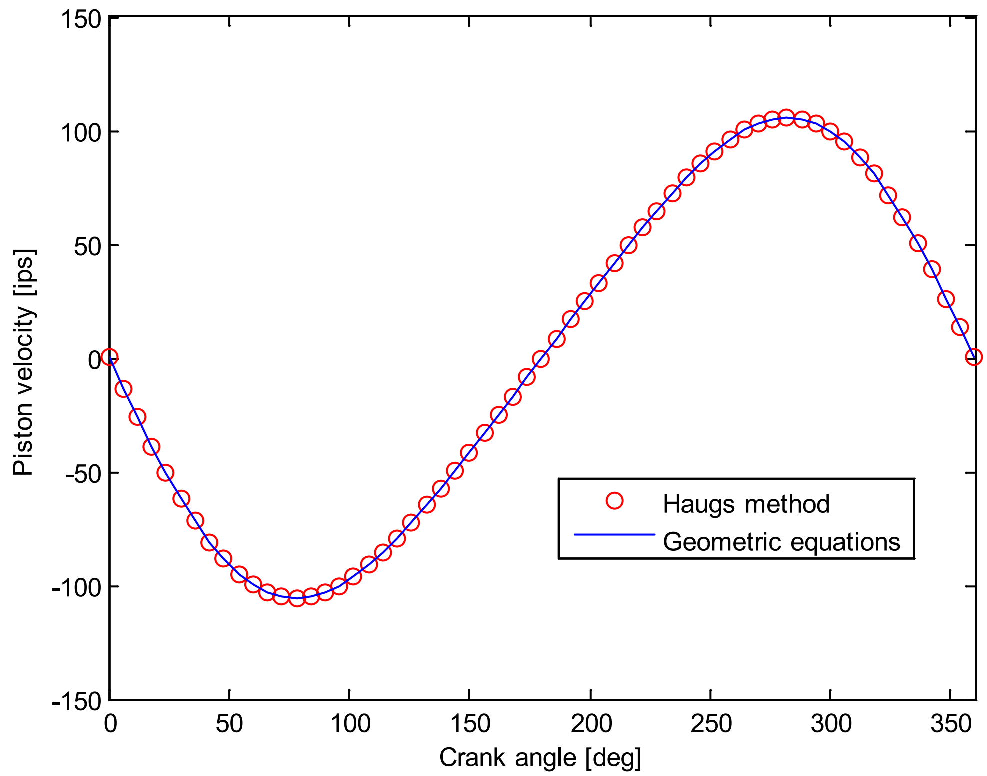
use PHI(9) = phi2 – phi2_start – w2*t – alpha2*t*t/2

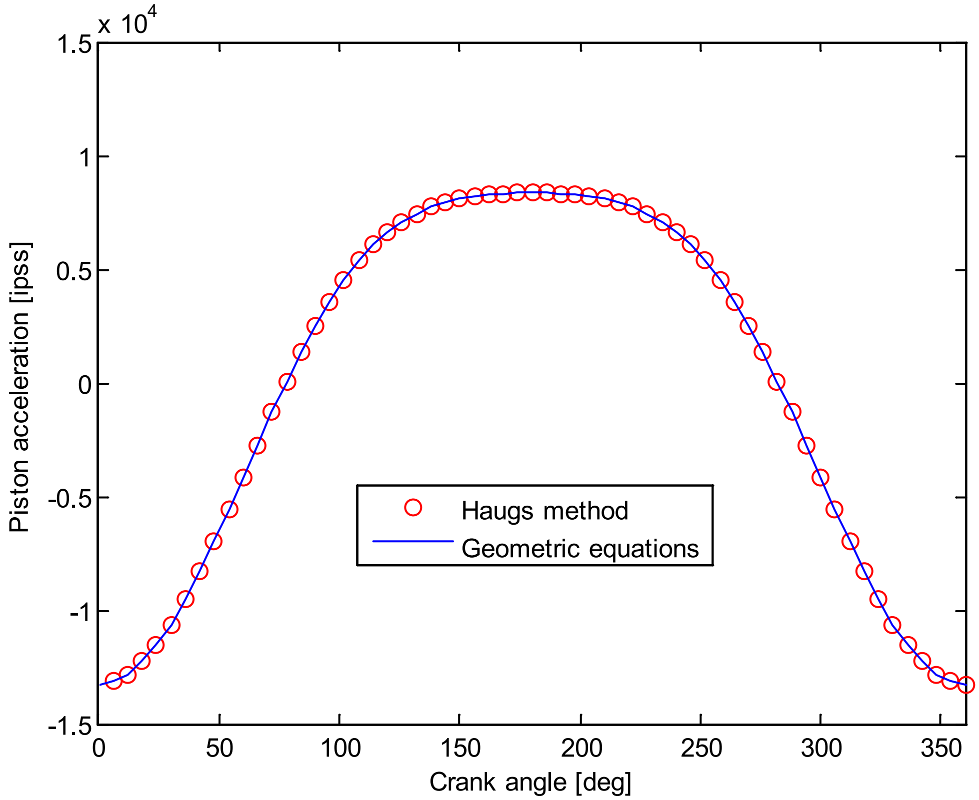
```
qd_transpose =
         0         0  -10.0000  251.4765  -39.4096    3.9013  116.6046   53.9475   -5.3533

qdd_transpose =
  1.0e+003 *
         0         0    0.0020   -1.7001   -2.6150    0.0071   -1.2309   -1.3273    0.0698

qdd(6) = 7.0627
qdd(9) = 69.7682
```

```
q_new_transpose =
         0         0    1.0472    1.5709    0.6363   -0.1983    4.7376         0         0

det_JAC = -4.2451

sGEO = 4.7376

vGEO = -99.6932

aGEO = -4.1730e+003

qd_transpose =
         0         0  104.7198  -91.9624   38.4724  -12.1491  -99.6932         0         0

qdd_transpose =
  1.0e+003 *
         0         0         0   -5.0889   -6.9781    2.1739   -4.1730         0         0

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

% h06.m - ME 481 H06 - joint constraint matrices
% HJSIII, 20.03.04

% constants
d2r = pi / 180;
Rmat = [ 0 -1 ;
         1  0 ];

% blueprint information - units [inch]
R = 0.985;
L = 4.33;
BG3 = 1.1;
r1A  = [  0      0 ]';
s2pA = [  0      0 ]';
s2pB = [  R      0 ]';
s3pB = [ -BG3    0 ]';
s3pC = [  L-BG3  0 ]';
s4pC = [  0      0 ]';

% crank speed = 1000 rpm CCW
w2 = +1000 * 2 * pi /60;  % [rad/sec]

% initial estimates for generalized coordinates
q = [ 0  0  25*d2r  2.0    0.5    -10*d2r    5.0    0  0 ]';  % rough, t=0.005
q = [ 0  0  30*d2r  1.946  0.367  -6.53*d2r  5.155  0  0 ]';  % manual, t=0.005
%t = 0.005;  % part 1, 2, 3
t = 0.010;  % part 4

% Newton-Raphson iteration loop
for iter = 1 : 10,

% generalized coordinates
  r2 = q(1:2);
  phi2 = q(3);
  r3 = q(4:5);
  phi3 = q(6);
  r4 = q(7:8);
  phi4 = q(9);

% transformation matrices
  A2 = [ cos(phi2)  -sin(phi2) ;
         sin(phi2)   cos(phi2) ];

  A3 = [ cos(phi3)  -sin(phi3) ;
         sin(phi3)   cos(phi3) ];

  A4 = [ cos(phi4)  -sin(phi4) ;
         sin(phi4)   cos(phi4) ];

% revolute points
  r2A = r2 + A2 * s2pA;
  r2B = r2 + A2 * s2pB;
  r3B = r3 + A3 * s3pB;
  r3C = r3 + A3 * s3pC;
  r4C = r4 + A4 * s4pC;
```

```
% evaluate constraints
  y4 = r4(2);

% fill constraint vector
  PHI = [  r2A-r1A   ;
           r3B-r2B   ;
           r4C-r3C   ;
           phi4      ;
           y4        ;
           phi2-w2*t ];
  PHI_transpose = PHI'

% Jacobian
  B2 = Rmat * A2;
  B3 = Rmat * A3;
  B4 = Rmat * A4;
  JAC = zeros(9,9);
  JAC(1:2,1:3) = [ +eye(2)  +B2*s2pA ];
  JAC(3:4,1:6) = [ -eye(2)  -B2*s2pB  +eye(2)  +B3*s3pB ];
  JAC(5:6,4:9) = [                    -eye(2)  -B3*s3pC  +eye(2)  +B4*s4pC ];
  JAC(7,9) = 1;
  JAC(8,8) = 1;
  JAC(9,3) = 1;
  JAC

% one step of Newton-Raphson
  q = q - inv(JAC)*PHI;
  q_new_transpose = q'

pause

% bottom - for iter
end

det_JAC = det( JAC )

% geometric equations
th = phi2;
thdot = w2;
thddot = 0;
phi = asin(  R*sin(th) / L );
sGEO = R*cos(th) + L*cos(phi)
phidot = R*thdot*cos(th) /L ./cos(phi);
vGEO = -R*thdot*sin(th) - L*phidot*sin(phi)
phiddot = ( R*thddot*cos(th) -R*thdot*thdot*sin(th) +L*phidot*phidot*sin(phi) ) ...
                 /L ./cos(phi);
aGEO = -R*thddot*sin(th) -R*thdot*thdot*cos(th) -L*phiddot*sin(phi) ...
                -L*phidot*phidot*cos(phi)

% matrix velocity
vel_rhs = zeros(9,1);
vel_rhs(9) = w2;
qd = inv(JAC) * vel_rhs;
qd_transpose = qd'

% matrix acceleration
phi2d = qd(3);
phi3d = qd(6);
acc_rhs = zeros(9,1);
acc_rhs(1:2) =  phi2d*phi2d*A2*s2pA;
acc_rhs(3:4) =  phi3d*phi3d*A3*s3pB - phi2d*phi2d*A2*s2pB;
acc_rhs(5:6) = -phi3d*phi3d*A3*s3pC;
qdd = inv(JAC) * acc_rhs;
qdd_transpose = qdd'

% bottom - h06
```


Piston position from crank center [inch]
Crank angle [deg]
Haugs method
Geometric equations
0
1
2
3
4
5
6
0
50
100
150
200
250
300
350


Piston velocity [ips]
Crank angle [deg]
Haugs method
Geometric equations
-150
-100
-50
0
50
100
150
0
50
100
150
200
250
300
350




```
% h06_ec.m - ME 481 H06 - joint constraint matrices - extra credit
% HJSIII, 13.02.07

% constants
d2r = pi / 180;
Rmat = [ 0 -1 ;
         1  0 ];


% blueprint information - units [inch]
R = 0.985;
L = 4.33;
BG3 = 1.1;
r1A  = [  0       0 ]';
s2pA = [  0       0 ]';
s2pB = [  R       0 ]';
s3pB = [ -BG3     0 ]';
s3pC = [  L-BG3   0 ]';
s4pC = [  0       0 ]';

% crank speed = 1000 rpm CCW
w2 = +1000 * 2 * pi /60;  % [rad/sec]

% initial estimates for generalized coordinates
q = [ 0  0  25*d2r  2.0     0.5     -10*d2r    5.0     0  0 ]';  % rough, t=0.005
q = [ 0  0  30*d2r  1.946   0.367   -6.53*d2r  5.155   0  0 ]';  % manual, t=0.005

% t = 0.005;  % part 1, 2, 3
% t = 0.010;  % part 4

% time loop
keep = [];
for t = 0 : 0.001 : 0.060,

% Newton-Raphson iteration loop
for iter = 1 : 10,

% generalized coordinates
  r2 = q(1:2);
```

```
  phi2 = q(3);
  r3 = q(4:5);
  phi3 = q(6);
  r4 = q(7:8);
  phi4 = q(9);

% transformation matrices
  A2 = [ cos(phi2)  -sin(phi2) ;
         sin(phi2)   cos(phi2) ];

  A3 = [ cos(phi3)  -sin(phi3) ;
         sin(phi3)   cos(phi3) ];

  A4 = [ cos(phi4)  -sin(phi4) ;
         sin(phi4)   cos(phi4) ];

% revolute points
  r2A = r2 + A2 * s2pA;
  r2B = r2 + A2 * s2pB;
  r3B = r3 + A3 * s3pB;
  r3C = r3 + A3 * s3pC;
  r4C = r4 + A4 * s4pC;

% evaluate constraints
  y4 = r4(2);

% fill constraint vector
  PHI = [  r2A-r1A   ;
           r3B-r2B   ;
           r4C-r3C   ;
           phi4      ;
           y4        ;
           phi2-w2*t ];

% Jacobian
  B2 = Rmat * A2;
  B3 = Rmat * A3;
  B4 = Rmat * A4;
  JAC = zeros(9,9);
  JAC(1:2,1:3) = [ +eye(2)  +B2*s2pA ];
  JAC(3:4,1:6) = [ -eye(2)  -B2*s2pB  +eye(2)  +B3*s3pB ];
  JAC(5:6,4:9) = [                    -eye(2)  -B3*s3pC  +eye(2)  +B4*s4pC ];
  JAC(7,9) = 1;
  JAC(8,8) = 1;
  JAC(9,3) = 1;

% one step of Newton-Raphson
  q = q - inv(JAC)*PHI;

% bottom - for iter
end

% geometric equations
th = phi2;
thdot = w2;
thddot = 0;
phi = asin(  R*sin(th) / L );
sGEO = R*cos(th) + L*cos(phi);

phidot = R*thdot*cos(th) /L /cos(phi);
vGEO = -R*thdot*sin(th) - L*phidot.*sin(phi);

phiddot = ( R*thddot*cos(th) -R*thdot*thdot*sin(th) +L*phidot*phidot*sin(phi) ) ...
                 /L /cos(phi);
aGEO = -R*thddot*sin(th) -R*thdot*thdot*cos(th) -L*phiddot*sin(phi) ...
                -L*phidot*phidot*cos(phi);

% matrix velocity
vel_rhs = zeros(9,1);
vel_rhs(9) = w2;
qd = inv(JAC) * vel_rhs;

% matrix acceleration
phi2d = qd(3);
phi3d = qd(6);
```

```
acc_rhs = zeros(9,1);
acc_rhs(1:2) =  phi2d*phi2d*A2*s2pA;
acc_rhs(3:4) =  phi3d*phi3d*A3*s3pB - phi2d*phi2d*A2*s2pB;
acc_rhs(5:6) = -phi3d*phi3d*A3*s3pC;
qdd = inv(JAC) * acc_rhs;

% bottom - for t
keep = [ keep ; q(3)  q(7)  qd(7)  qdd(7)  sGEO  vGEO  aGEO ];
end

% plot results
th_deg = keep(:,1) / d2r;
sMAT = keep(:,2);
vMAT = keep(:,3);
aMAT = keep(:,4);
sGEO = keep(:,5);
vGEO = keep(:,6);
aGEO = keep(:,7);

figure( 1 )
  plot( th_deg,sMAT,'ro', th_deg,sGEO,'b' )
  axis( [ 0 360  0 6 ] )
  xlabel( 'Crank angle [deg]' )
  ylabel( 'Piston position from crank center [inch]' )
  legend( 'Haugs method', 'Geometric equations' )

figure( 2 )
  plot( th_deg,vMAT,'ro', th_deg,vGEO,'b' )
  axis( [ 0 360  -150 150 ] )
  xlabel( 'Crank angle [deg]' )
  ylabel( 'Piston velocity [ips]' )
  legend( 'Haugs method', 'Geometric equations' )

figure( 3 )
  plot( th_deg,aMAT,'ro', th_deg,aGEO,'b' )
  axis( [ 0 360  -15000 15000 ] )
  xlabel( 'Crank angle [deg]' )
  ylabel( 'Piston acceleration [ipss]' )
  legend( 'Haugs method', 'Geometric equations' )

% bottom of h06_ec
```

```
% h06_ec_ec.m - ME 481 H06 - joint constraint matrices - extra extra credit
% HJSIII, 13.02.08

% constants
d2r = pi / 180;
Rmat = [ 0 -1 ;
         1  0 ];


% Notes_04_05 four bar
% blueprint information - units [cm]
AB = 30;
BC = 60;
CD = 45;
AD = 90;
BG3 = 23;
CG3 = BC - BG3;
DG4 = 24;
CG4 = CD - DG4;
r1A  = [   0       0 ]';
r1D  = [  AD       0 ]';
s2pA = [   0       0 ]';
s2pB = [  AB       0 ]';
s3pB = [ -BG3      0 ]';
s3pC = [  CG3      0 ]';
s4pC = [ -CG4      0 ]';
s4pD = [  DG4      0 ]';

% initial estimates for generalized coordinates
phi2 = 65 * d2r;
phi3 = 13.151 * d2r;
phi4 = -65.173 * d2r;

t = 0;
phi2_start = phi2;  % 65 deg
w2 = -10;           % 10 rad/sec CW
alpha2 = 2;         % 2 rad/sec/sec CCW

xG3 = AB*cos(phi2) + BG3*cos(phi3);
yG3 = AB*sin(phi2) + BG3*sin(phi3);

xG4 = AB*cos(phi2) + BC*cos(phi3) + CG4*cos(phi4);
yG4 = AB*sin(phi2) + BC*sin(phi3) + CG4*sin(phi4);

q = [ 0  0  phi2  xG3  yG3  phi3  xG4  yG4  phi4 ]';

% Newton-Raphson iteration loop
for iter = 1 : 10,

% generalized coordinates
  r2 = q(1:2);
  phi2 = q(3);
  r3 = q(4:5);
  phi3 = q(6);
  r4 = q(7:8);
  phi4 = q(9);

% transformation matrices
  A2 = [ cos(phi2)  -sin(phi2) ;
         sin(phi2)   cos(phi2) ];

  A3 = [ cos(phi3)  -sin(phi3) ;
         sin(phi3)   cos(phi3) ];

  A4 = [ cos(phi4)  -sin(phi4) ;
         sin(phi4)   cos(phi4) ];

% revolute points
  r2A = r2 + A2 * s2pA;
  r2B = r2 + A2 * s2pB;
  r3B = r3 + A3 * s3pB;
  r3C = r3 + A3 * s3pC;
  r4C = r4 + A4 * s4pC;
  r4D = r4 + A4 * s4pD;
```

```
% fill constraint vector
  PHI = [  r2A-r1A                    ;
           r3B-r2B                    ;
           r4C-r3C                    ;
           r4D-r1D                    ;
           phi2-phi2_start-w2*t-alpha2*t*t/2 ];

% Jacobian
  B2 = Rmat * A2;
  B3 = Rmat * A3;
  B4 = Rmat * A4;
  JAC = zeros(9,9);
  JAC(1:2,1:3) = [ +eye(2)  +B2*s2pA                                     ];
  JAC(3:4,1:6) = [ -eye(2)  -B2*s2pB  +eye(2)  +B3*s3pB                  ];
  JAC(5:6,4:9) = [                    -eye(2)  -B3*s3pC  +eye(2)  +B4*s4pC ];
  JAC(7:8,7:9) = [                                       +eye(2)  +B4*s4pD ];
  JAC(9,3) = 1;

% one step of Newton-Raphson
  q = q - inv(JAC)*PHI;

  phi_transpose = PHI'
  qnew_transpose = q'
  pause

% bottom - for iter
end

% matrix velocity
vel_rhs = zeros(9,1);
vel_rhs(9) = w2 + alpha2*t;
qd = inv(JAC) * vel_rhs;
qd_transpose = qd'

% matrix acceleration
phi2d = qd(3);
phi3d = qd(6);
phi4d = qd(9);
acc_rhs = zeros(9,1);
acc_rhs(1:2) = phi2d*phi2d*A2*s2pA;
acc_rhs(3:4) = phi3d*phi3d*A3*s3pB - phi2d*phi2d*A2*s2pB;
acc_rhs(5:6) = phi4d*phi4d*A4*s4pC - phi3d*phi3d*A3*s3pC;
acc_rhs(7:8) = phi4d*phi4d*A4*s4pD;
acc_rhs(9) = alpha2;
qdd = inv(JAC) * acc_rhs;
qdd_transpose = qdd'

% bottom of h06_ec_ec
```