McGill

Department of Mechanical Engineering

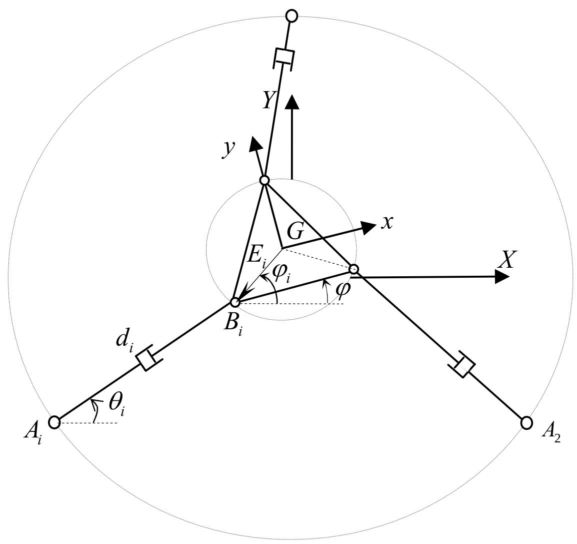
MECH 573: Mechanics of Robotic Systems

Winter 2006: TR 11:35-12:55 ENGTR 1090

Prof. Hamid D. Taghirad

Project Part 1 Due date: 2006/03/09

Consider the planar 3RPR manipulator shown in the following figure.



In this manipulator, the base points A_i's are located on a circle with radius R_A, with an equivalent angle of 120^o. Similarly, the moving platform points B_i's are located on a circle with radius R_B. The fixed coordinate frame is located at the center of circle R_A, whose x axis is parallel to A_1A_2 and its y axis along A_3. The moving platform position and orientation is represented by the vector $\mathbf{x} = [x_G, y_G, \varphi]^T$, and the input joint variables are the extendible limb lengths d_i's, $i=1,2$ and 3. The joint variables vector is represented by $\mathbf{q} = [d_1, d_2, d_3]^T$ and the angle of each limb is denoted with θ_i's. The vector E_i denotes the vector representation of GB_i.

1) Derive expressions for inverse kinematics, i.e. suppose $[x_G, y_G, \varphi]^T$ is given find d_i's.

2) Solve the forward kinematics meaning d_i's are given find expressions for $[x_G, y_G, \varphi]^T$.

3) Derive the Jacobian matrices J_x and J_q, and discuss on the inverse and direct kinematics singularities, the Jacobian matrices are defined as $J_x \dot{x} = J_q \dot{q}$.

4) Derive the stiffness matrix of the manipulator, assuming that the moving platform is at a central location. Under what conditions will the stiffness matrix become decoupled? Assume unit stiffness at each limb.

5) Develop a program in Matlab with the following numerical values for parameters, in which for a given moving platform trajectory $\boldsymbol{x(t)}$, the limb lengths $d_i(t)$'s are found, notice that due to multiple solution in inverse kinematics, the solution must be found in such a way that no jumps are observed in the manipulator trajectories. Then give the time trajectories found in the first part, and solve the forward kinematics of the manipulator. Compare the calculated $\boldsymbol{x(t)}$ with the original one to verify your programs. Submit your developed programs via WebCT, and provide the illustrative plots of $d_i(t)$'s, and possibly $\boldsymbol{x_d(t) - x(t)}$ in your project report, for verification purpose.

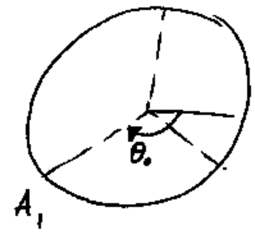
$R_B=1,\ R_A=10,\ \mathbf{x}(0)=[0,\ 0,\ 0]^T,\ \delta x=-0.5,\ \delta y=0.5,\ \omega=0.25,\ 0 < t < 1.$
$x_d(t)= \delta x\ (3-2t)\ t^2,\ y_d(t)= \delta y\ (3-2t)\ t^2,\ \varphi_d(t)= \omega\ t.$

6) For the above trajectories, evaluate the Jacobian matrix $J = J_q^{-1} J_x$ and its singular values (`svd` function in Matlab). Does the lowest singular value vanish in any configuration and the system experience any singularities? To further analyze the singularities of the mechanism, extend the trajectories, into a larger 3D workspace for the moving platform and give 3D plot of the locations where the system experience singularities, if any.

7) Try similar analysis as in (6) but examine the eigenvalues of the stiffness matrix instead of the singular values of the Jacobian matrix, what do you observe?

"Good Luck"

1) Kinematic analysis of 3RPR manipulator

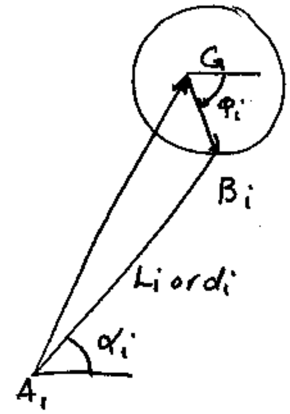


$$\theta_i = \left[\theta_0 \,,\ \theta_0 + 2\pi/3 \,,\ \theta_0 + 4\pi/3\right]^T \text{ where } \theta_0 = -5\pi/6$$

$$\theta_{A_i} = \theta_{A_0} + (i-1) * 2\pi/3 \,;$$

$$A_i = \left[RA\,\cos(\theta_{A_i}) \,;\ RA\,\sin(\theta_{A_i})\right]$$

Loop closure: $\overrightarrow{A_iG} = \overrightarrow{A_iB_i} + \overrightarrow{B_iG}$



$$\begin{cases} x_G - x_{A_i} = L_i\, c\alpha_i - (RB\cos\varphi_i) \\ y_G - y_{A_i} = L_i\, s\alpha_i - (RB\sin\varphi_i) \end{cases}$$

in which $\varphi_i = \theta_{B_i} + \varphi$; $\theta_{B_i} = \theta_{B_0} + (i-1) * 2\pi/3 \,;$

Cancel α_i :

$$\text{assume} \begin{cases} x_i = x_G + RB\cos\varphi_i - x_{A_i} \\ y_i = y_G + RB\sin\varphi_i - y_{A_i} \end{cases} \rightarrow \begin{cases} L_i\, c\alpha_i = x_i \\ L_i\, s\alpha_i = y_i \end{cases}$$

$$\Rightarrow \begin{cases} L_i^2 = x_i^2 + y_i^2 \rightarrow L_i = \sqrt{x_i^2 + y_i^2} \\ \alpha_i = \operatorname{Atan2}\left[y_i \,,\ x_i\right] \end{cases}$$

2) Forward Kinematics

$$\begin{cases} L_i\, c\alpha_i = x_G + RB\, c\varphi_i - x_{A_i} = x_G + X_i \\ L_i\, s\alpha_i = y_G + RB\, s\varphi_i - y_{A_i} = y_G + Y_i \end{cases} \quad \text{in which} \begin{cases} X_i = RB\, c\varphi_i - x_{A_i} \\ Y_i = RB\, s\varphi_i - y_{A_i} \end{cases}$$

Square & Sum

$$L_i^2 = x_G^2 + y_G^2 + 2X_i x_G + 2Y_i y_G + X_i^2 + Y_i^2$$

this siplifies to

$x_G^2 + y_G^2 + r_i x_G + s_i y_G + u_i = 0$ for $i = 1, 2, 3$

in which $\begin{cases} r_i = 2X_i = 2(RBc\varphi_i - x_{Ai}) \\ s_i = 2Y_i = 2(RBs\varphi_i - y_{Ai}) \\ u_i = X_i^2 + Y_i^2 - L_i^2 \end{cases}$

$$\underbrace{\begin{bmatrix} r_1 - r_2 & s_1 - s_2 \\ r_2 - r_3 & s_2 - s_3 \end{bmatrix}}_{R} \begin{bmatrix} x_G \\ y_G \end{bmatrix} = \underbrace{\begin{bmatrix} u_2 - u_1 \\ u_3 - u_2 \end{bmatrix}}_{u} \rightarrow v = \begin{bmatrix} x_G \\ y_G \end{bmatrix} = R^{-1} \cdot u$$

use pinv for robustness

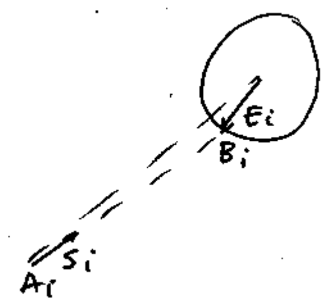
put back in the above equation

$$g_i = v_1^2 + v_2^2 + r_i \cdot v_1 + s_i v_2 + u_i = f(\varphi)$$

Solve $Res = \sum_{i=1}^{3} g_i = 0$ with fzero function.

3) Jacobian Analysis

loopclosure: $\vec{A_i G} + \vec{GB_i} = \vec{A_i B_i}$ differentiate:



$$v_G + \dot{\varphi}(k \times E_i) = \dot{L}_i \hat{s}_i + \dot{\alpha}_i (k \times \hat{s}_i) \cdot L_i$$

$$v_G + \dot{\varphi}(k \times E_i) = \dot{L}_i \hat{s}_i + \dot{\alpha}_i L_i (k \times \hat{s}_i)$$

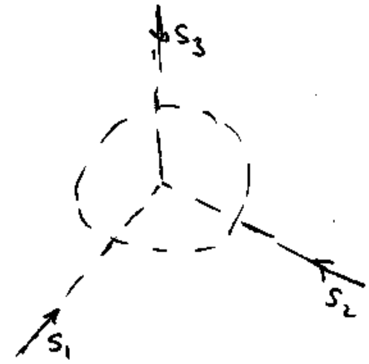
Dot multiply with $\hat{s}_i$ to cancel $\dot{\alpha}_i$

$$\hat{s}_i \cdot v_G + \dot{\varphi}\, k \cdot (E_i \times S_i) = \dot{L}_i$$

$$\Rightarrow J_x = \left[\begin{array}{c|c|c} S_{ix} & S_{iy} & E_{ix} S_{iy} - S_{ix} E_{iy} \end{array} \right] \quad ; \quad J_q = I_{3\times3}$$

J_x becomes singular when $E_i \parallel \hat{s}_i$! this happens at central location of the manipulator ! and when $\varphi = \pi$ $\Rightarrow$ workspace is limited here

4) Stiffness analysis



$$K = k J_x^T J_x = 1 \cdot J_x^T J_x \Big|_{@\text{central position}}$$

@central position $E_i \parallel S_i$; s_i = Symmetric

$$s_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} ; \quad S_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} ; \quad S_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$J_x = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix} \Rightarrow K = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The Stiffness matrix is decoupled but it is Singular.

5) Note: that since the manipulator is singular at central position, the ill conditioning may cause numerical problems at this configuration; i.e. better to start at $X_0 = [0, 0, Pi/4]$ instead of central location.

Programs are attached and can be found in the homepage of the course

6) The last part of the program uses plot3d to generate a 3D plot for Singular Conditions; check the attached programs & figures

Good Luck.

```
%   Copyright Hamid D. Taghirad 2006
%
% This program Verifies the solution of inverse and forward kinematic
% problem of a 3RPR manipulator. Furthermore it calculates the Jacobian
% and its singular configurations.
%
% Mechanics of Robotics Systems MECH 573
% Project Part I

clear all

%   /initial values/
deg2rad=pi/180;
rad2deg=180/pi;

dt=0.01;
Tf=1;
N=Tf/dt;

%
%   The coordinates of Ai's, Bi's, ai's and bi's at initial position
%   Consider the fixed coordinate is located at the center
%   at initial position



RA = 10;            % the Ai's circle radius
RB = 1;             % the Bi's circle radius
th0=-150*deg2rad;   % The orientation of A1 in its circle
x(:,1)=[0;0;0]; % pos/orientation of the first  moving platform center
alpha_old = [60;120;-90;]*deg2rad; % found from first iteration
x_old     = x(:,1);     % the initial position and orientation
for i=1:3;
     Ath(i)=th0+(i-1)*2*pi/3;
     A(:,i)=[RA*cos(Ath(i)); RA*sin(Ath(i))];
end

% The main loop
for i=2:N;
    t(i)=dt*(i-1);      % generating time
%   The desired trajectory of the end effector xd=3xN vector
%   And its derivative dxd=3xN vector
    wd=-pi;
    deltax=0;
    deltay=0;
    xd(:,i)=[x(1,1)+(3-2*t(i))*t(i)^2*deltax;x(2,1)+...
            (3-2*t(i))*t(i)^2*deltay;x(3,1)+wd*t(i)];
    dxd(:,i)=[6*(1-t(i))*t(i)*deltax;6*(1-t(i))*t(i)*deltay;wd];
%-----------------------------------------------------------------
%   Solve the inverse kinematics
%
    [L(:,i),alpha(:,i)]=InvKin_3RPR(xd(:,i),A,RB,th0,alpha_old);
```

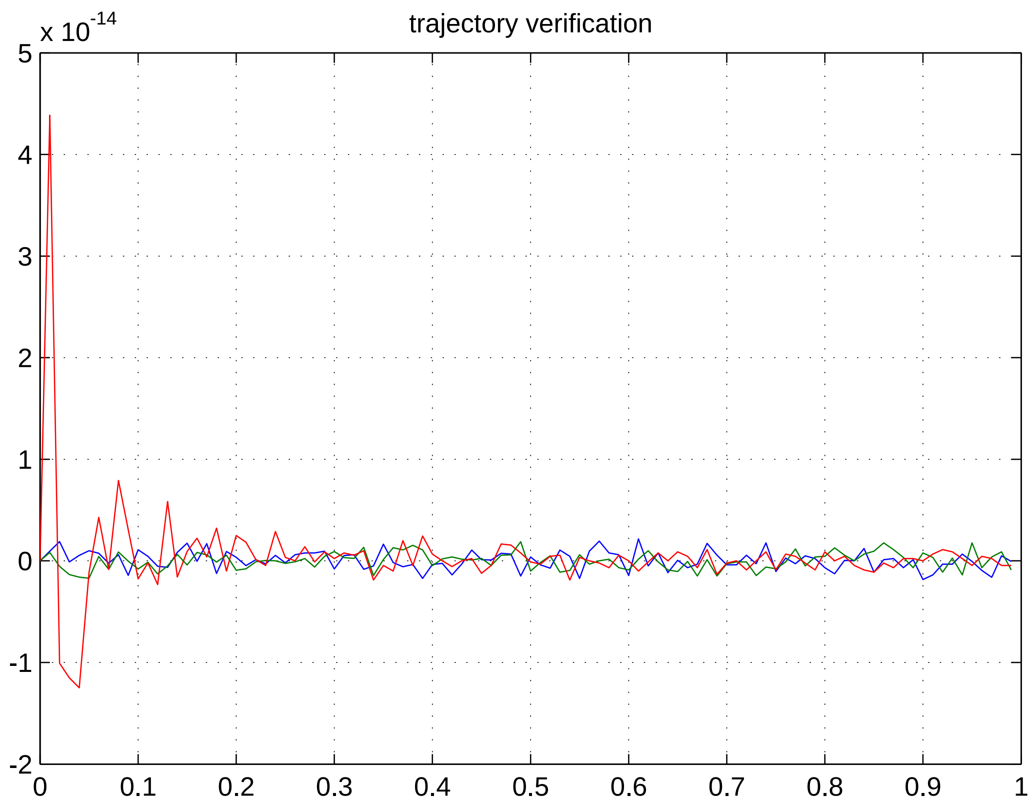
```
    B = Geometry_3RPR(xd(:,i),RB,th0);
    alpha_old=alpha(:,i);
%-----------------------------------------------------------------
%   Solve the Forward kinematics
%
    xc(:,i)=FK_3RPR(L(:,i),A,RB,Ath,x_old);
    x_old=xc(:,i);

%   Generate the Jacobians, and examine the inverse Kinematics at
%   velocity level
%
    J=Jacobian_3RPR(xd(:,i),B,alpha(:,i));
    Sen(:,i)=rcond(J'*J);
    dL(:,i)=J*dxd(:,i);
end


figure(1)
clf
plot(t,xd-xc),grid
title('trajectory verification')

figure(2)
clf
plot(t(2:N),L(1,2:N)),grid
title('Limb lengths')

echo on
pause       % Strike a key to continue
echo off

figure(1)
clf
plot(-xd(3,:)*rad2deg,Sen),
grid on
title('1/cond No. of the Jacobian')
xlabel('\phi degrees')
ylabel('rcond J(\phi)')

%   3D Workspace and singularity analysis
%
%   Define a 3D woekspace

xg=-5:1:5;
yg=-5:1:5;
phi=-pi:pi/10:pi;

clear Z
NN=1;
alpha_old = [60;120;-90;]*deg2rad; % found from first iteration
for i=1:max(size(xg));
    for j=1:max(size(yg));
```

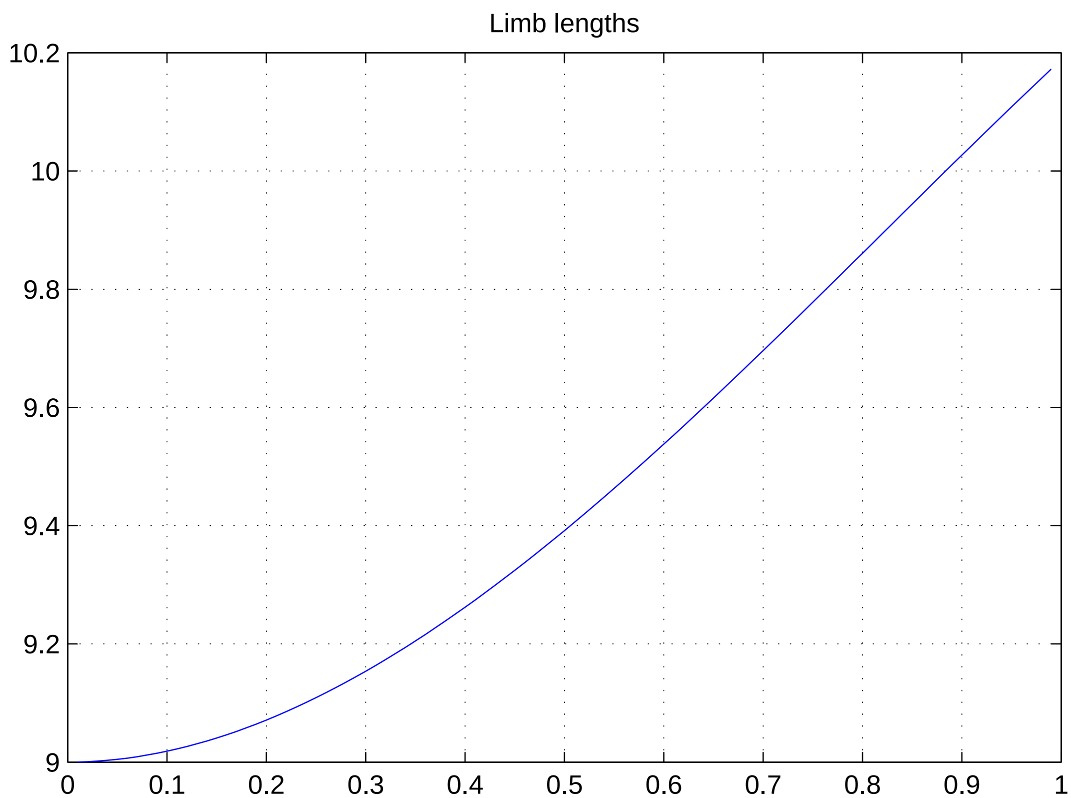
```
        for k=1:max(size(phi));
         Zi=[xg(i);yg(j);phi(k)];
         [Li, alphai]= InvKin_3RPR(Zi,A,RB,th0,alpha_old);
         Bi = Geometry_3RPR(Zi,RB,th0);
         alpha_old=alphai;
         Ji=Jacobian_3RPR(Zi,Bi,alphai);
            if rcond(Ji) < 1e-12 ;
            Z(:,NN)=[xg(i);yg(j);phi(k)];
            NN=NN+1;
            end
        end
     end
end

figure(2)
clf
plot3(Z(1,:),Z(2,:),Z(3,:)*rad2deg,'o')
axis square
grid on
xlabel('MP position x')
ylabel('MP position y')
zlabel('MP orientation phi (degrees)')
title('Locations where singularity occurs')

%   This concludes the program
%
```

```
function [L,alpha]=InvKin_3RPR(X,A,RB,th0,alpha_old);
%
%   Inverse Kinematics of the 3RPR parallel manipulator
%
%   Input arguments:
%   X   : The position and orientation vector of the moving
%         platform                                          1x3
%   A   : The position of the base coordinates:             2x4
%   RB  : The circle radius of B'i's                        1x1
%   th0 : The configuration angles of A1 and B1             1x1
%   alpha_old: The previous alpha to wrap the alpha         1x4

%   Output argument:
%   L     : The limb lengths                                1x4
%   alpha : The angle of the limbs                          1x4

threshold = pi/6;   % threshold to wrap alpha
xg=X(1);
yg=X(2);
phi=X(3);

for i= 1:3,
    Phi(i)=th0 + (i-1)*2*pi/3 + phi;
    x(i) = xg - A(1,i) + RB*cos(Phi(i));
    y(i) = yg - A(2,i) + RB*sin(Phi(i));
    L(i)=sqrt(x(i)^2+y(i)^2);
    alpha(i)=atan2(y(i),x(i));

    if abs(alpha(i) - alpha_old(i)) > threshold;
        if alpha(i) < alpha_old(i)
            alpha(i)=alpha(i) + 2*pi;
        else
            alpha(i)=alpha(i) - 2*pi;
        end
    end
end
L=L';
alpha=alpha';
```

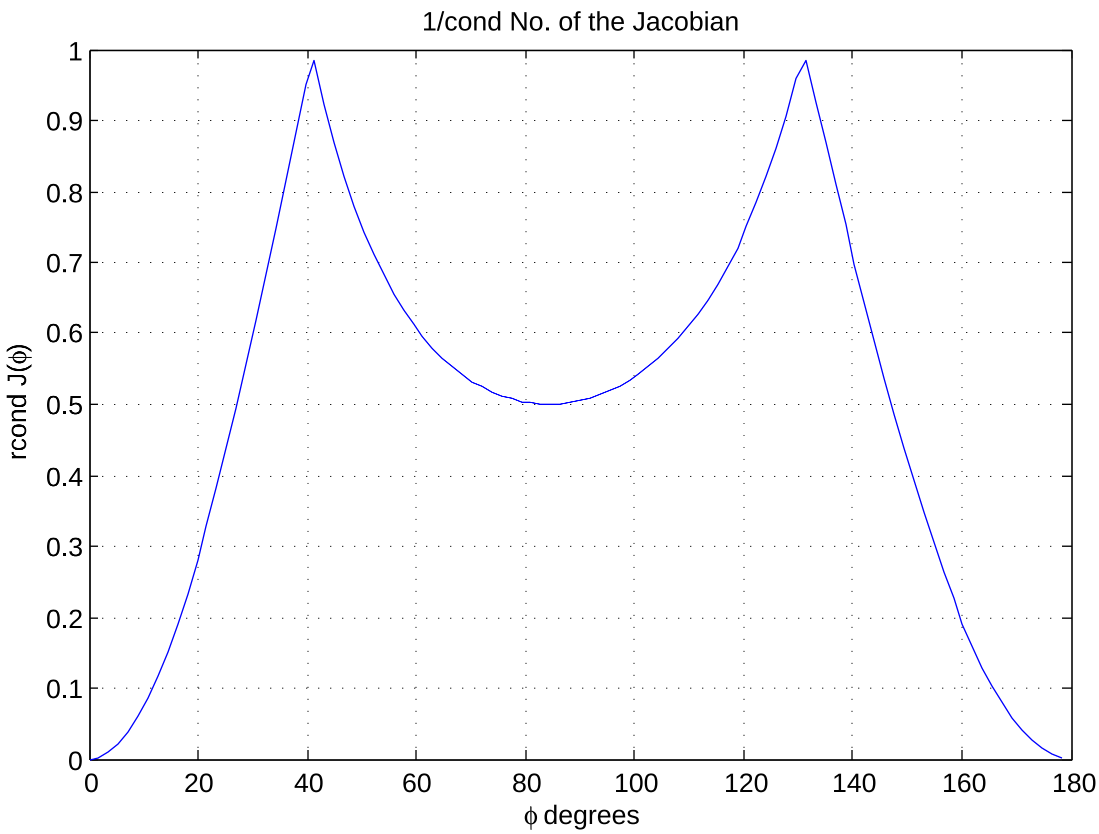
```
function X=FK_3RPR(L,A,RB,Ath,x_old);
%
%   Forward Kinematics of the 3RPR parallel manipulator
%
%   Input arguments:
%   L  : The limbs length                                    1x4
%   A  :  The position of Ai's                               2x3
%   RB  : The circle radius of Bi's                          1x1
%   Ath : The configuration angles of Ai's                   1x3
%   x_old: The previous X to wrap the phi                    1x4
%
%   Output argument:
%   X = [Gx;Gy;phi]                                          3x1
%   Gx:The x position of the center of the moving platform  1x1
%   Gy:The y position of the center of the moving platform  1x1
%   phi: The orientation of the moving platform              1x1
%
%   Mechanics of Robotics Systems MECH 573
%   Project Part I
%   Copyright Hamid D. Taghirad 2006
%
x=x_old(3);
X(3)=fzero(@FKfun, x);
    function out=FKfun(x)
        phi=x;
         for i=1:3;
           Phi(i)=Ath(i) + phi;
           xx(i)=RB*cos(Phi(i)) - A(1,i);
           yy(i)=RB*sin(Phi(i)) - A(2,i);
           r(i) = 2*xx(i);
           s(i) = 2*yy(i);
           u(i) = xx(i)^2 + yy(i)^2 - L(i)^2;
        end
        R=[r(1)-r(2) s(1)-s(2)
           r(2)-r(3) s(2)-s(3)];
        U=[u(2)-u(1)
           u(3)-u(2)];
        v=pinv(R)*U;
        out = v(1)^2+v(2)^2+r(1)*v(1)+s(1)*v(2)+u(1);
%         for i=1:3;
%         g(i) = v(1)^2+v(2)^2+r(i)*v(1)+s(i)*v(2)+u(i);
%         end
%         out=sum(g);
    end % End nested function
    %Calculates the X and Y coordinates
  X(1:2)=v;
end %end main function
```

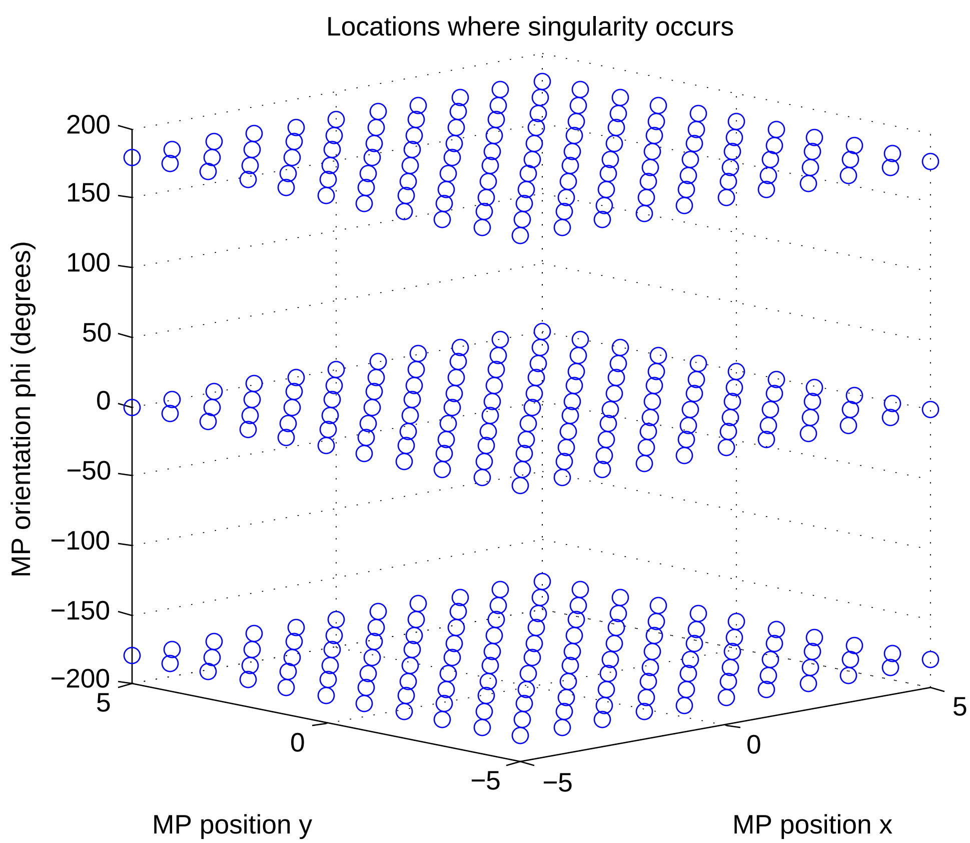
```
function J = Jacobian_3RPR(X,B,alpha);
%   Jacobian Matrix of the 3RPR parallel manipulator
%   function J = MJacobian(X,B,alpha);
%
%   Input arguments:
%
%   X    : The position/orientation of the Moving platform    3x1
%   B    : The Position of Bi's                               2x4
%   alpha: The limb absolute angles                           4x1
%
%   Output argument:
%   J : The Jx-Jacobian matrix of 3RPR manipulator            4x3

G=X(1:2,1);
for j=1:3,
     VE(:,j)=B(:,j)-G;
     VS(:,j)=[cos(alpha(j));sin(alpha(j))];
end

E1=VE(:,1); E2=VE(:,2); E3=VE(:,3);
S1=VS(:,1); S2=VS(:,2); S3=VS(:,3);

ExS= [E1(1)*S1(2)-E1(2)*S1(1)
      E2(1)*S2(2)-E2(2)*S2(1)
      E3(1)*S3(2)-E3(2)*S3(1)
     ];

 J=[VS(1,:)', VS(2,:)', ExS];
```


trajectory verification
x 10^-14
5
4
3
2
1
0
-1
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Limb lengths
10.2
10
9.8
9.6
9.4
9.2
9
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


1/cond No. of the Jacobian
rcond J(ϕ)
ϕ degrees
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
20
40
60
80
100
120
140
160
180


Locations where singularity occurs
MP orientation phi (degrees)
200
150
100
50
0
−50
−100
−150
−200
5
0
−5
−5
0
5
MP position y
MP position x