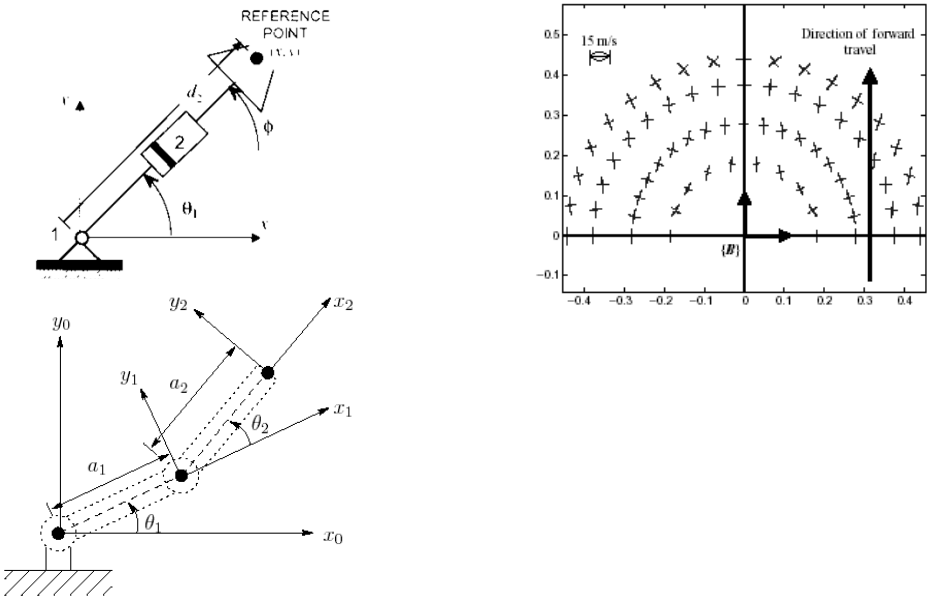
THE DEPARTMENT OF AUTOMOTIVE ENGINEERING
CLEMSON UNIVERSITY
AuE 8220: Autonomy: Mobility and Manipulation, Fall 2022

Homework #6: Jacobians (related design and control issues)
Assigned on: Nov. 15th, 2022, Due: Nov 22nd 2022 1:00 PM

Instructions:
Submit your scanned/printed work as a single PDF on Canvas by the due date/time noted above.

Problem 1:



(a) (b)

Figure 1: (a) A Planar RP and an RR manipulator and (b) Manipulability ellipsoids in the workspace (for illustration only)

For the following planar RP manipulator shown in Figure 1(a) assume that $0.5 \le d_2 \le 1.5$ and $0^o \le \theta_1 < 360^o$. For the planar RR manipulator assume that a_1=3, a_2=2 and $0^o \le \theta_1 < 360^o$ and $0^o \le \theta_2 < 360^o$ and use the elbow-up configuration. Create a fine grid of the workspace and compute the inverse-kinematics solution for use in the sub-parts below.

(i) Using the Isotropy Index Measure of Manipulability $w_I = \frac{\sigma_{\min}}{\sigma_{\max}}$ as a performance measure, create an estimate of this performance measure at various locations in the workspace, i.e. determine the values of the measure of manipulability on the fine grid and plot the resulting "manipulability surface" in a 3D plot (Method to be used: For the given values of a_1 and a_2: (i) select the positions of the end effector X and Y on a fine grid in the Cartesian workspace, (ii) determine the corresponding joint angles and (iii) compute the Jacobian based "quality measure". We can plot this as a 3D plot – for each X and Y location of the end-effector, let Z corresponds to the computed "quality measure"). Discuss the locations of the configuration(s) where this measure may be maximum/minimum. What are these maximum/minimum values?

(ii) Yoshikawa's measure of manipulability $w = \sqrt{det(\mathbf{J}\mathbf{J}^T)}$ can also be written as $w = det(\mathbf{J})$ for this special case of a planar 2R manipulator. Then analytically determine the conditions under which w is maximized? For the given values of a_1 and a_2, determine the "optimal/maximal" configuration(s)? What is the corresponding maximum singular values? Plot the manipulability ellipsoid at such configuration(s)?

We can also determine the "manipulability ellipsoid" corresponding to a given end-effector location. Plot the "scaled manipulability ellipsoids" at a suitable finely spaced grid within the workspace of this manipulator – see the example shown in Figure 1(b) where only a suitably scaled version of the principal axes of the corresponding manipulability ellipsoid are shown).

Solution:

(i) **Step 1**

To use the isotropy index of measure, we first find the Jacobian for the end-effector

$$x = a_1\cos\theta_1 + a_2\cos(\theta_1+\theta_2) = 3\cos\theta_1 + 2\cos(\theta_1+\theta_2)$$
$$y = a_1\sin\theta_1 + a_2\sin(\theta_1+\theta_2) = 3\sin\theta_1 + 2\sin(\theta_1+\theta_2)$$

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} \end{bmatrix}$$

The eigenvalue of the matrix JJ^T for each joint configuration can be determined using J. The axes of the velocity ellipsoid for each configuration of the workspace will be equal to the square root of each eigenvalue. Then, it is possible to obtain the Isotropy Index Measure of Manipulability.

Below shown is code for 1.1:

```
clear;
clc;
%% Part 1.1
% link lengths
global a1
global a2
%%assign value to the variables

a1 = 3;
a2 = 2;
r1 = a1 - a2;
r2 = a1 + a2;

% Workspace grid
r = r1:0.01:r2;
theta = 0:0.05*pi:1.95*pi;
```

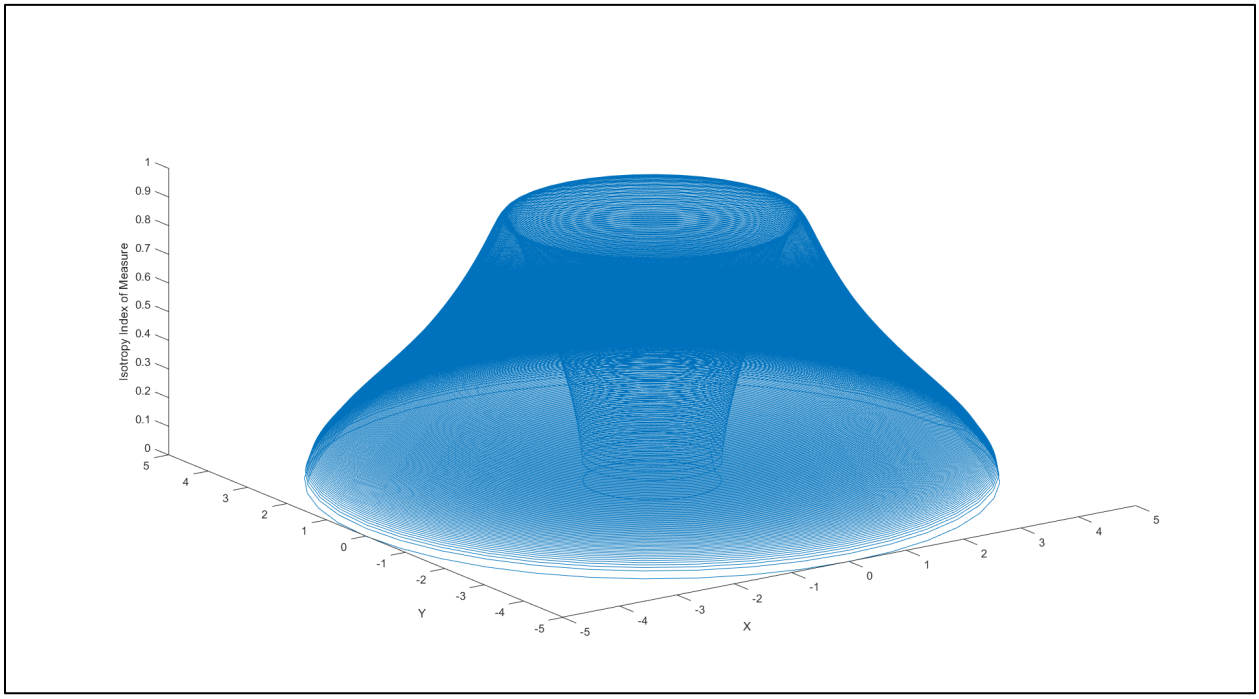
```
%Initial Values
measure = zeros(1,length(r)*length(theta));
x = zeros(1,length(r)*length(theta));
y = zeros(1,length(r)*length(theta));
k = 1;

%%calculate manipulability for each workspace configuration
for i = 1:1:length(r)
    for j = 1:1:length(theta)
        x(k) = r(i)*cos(theta(j));
        y(k) = r(i)*sin(theta(j));
        measure(k) = IIM(x(k),y(k));
        k = k+1;
    end
end
%%plot the result
plot3(x,y,measure)
clear
%%define global variables: link length
global a1
global a2
%%assign value to the variables
a1 = 3;
a2 = 2;
r1 = a1 - a2;
r2 = a1 + a2;
%%grid the workspace
r = r1:0.01:r2;
theta = 0:0.05*pi:1.95*pi;
%%initiate values needed for plotting
measure = zeros(1,length(r)*length(theta));
x = zeros(1,length(r)*length(theta));
y = zeros(1,length(r)*length(theta));
k = 1;
%%calculate manipulability for each workspace configuration
for i = 1:1:length(r)
    for j = 1:1:length(theta)
        x(k) = r(i)*cos(theta(j));
        y(k) = r(i)*sin(theta(j));
        measure(k) = IIM(x(k),y(k));
        k = k+1;
    end
end
%%plot the result
plot3(x,y,measure);
xlabel('X');
ylabel('Y');
zlabel('Isotropy Index of Measure');
```

Corresponding function is:

```
function measure = IIM(x, y)
%link lengths
global a1
global a2
```

```
%configuration calculation
g1 = acos((x.^2 + y.^2 - a1.^2 - a2.^2)/(2*a1*a2));
g2 = - acos((x.^2 + y.^2 - a1.^2 - a2.^2)/(2*a1*a2));
theta1 = atan2(y, x) - atan(a2*sin(g1)/(a1 + a2*cos(g1)));
theta2 = atan2(y, x) - atan(a2*sin(g2)/(a1 + a2*cos(g2)));

%choose the elbow up configuration
if a1*sin(theta1)>a1*sin(theta2)
theta2 = g1;
elseif a1*sin(theta1)<a1*sin(theta2)
theta1 = theta2;
theta2 = g2;
else
theta2 = g1;
end

%Jacobian calculation
J = [-a1*sin(theta1)-a2*sin(theta1+theta2), -a2*sin(theta1+theta2);...
a1*cos(theta1)+a2*cos(theta1+theta2), a2*cos(theta1+theta2)];

%calculate the eigen value of Jacobian
A = J*J';
[~, S, ~] = svd(A);

%get manipulability
measure = min(sqrt(S(1,1)),sqrt(S(2,2)))/max(sqrt(S(1,1)),sqrt(S(2,2)));
end
```

Plot:



The plot reveals that the measure's lowest value, 0, is found at the inner and outer rings of workspace. When the end effector is in this position, the velocity ellipsoid's short axis is compressed to 0.
The ring with a radius of 2.13 has the highest value at 0.92.

(ii & iii) Measure of manipulability is given as w = det(J)

From part I, using the Jacobian and the Matlab symbolic toolbox

$$w = 6\sin\theta_2$$
$$\dot{w} = 6\cos\theta 2$$

Code is:

```
syms t1 t2 real
 J = [-3*sin(t1) - 2*sin(t1+t2) -2*sin(t1+t2);3*cos(t1)+ 2*cos(t1 +t2) 2*cos(t1
+t2)];
 w = simplify(det(J));
```

We can see that:
For Optimal Configuration: $\theta_2 = 90^o$. 270^o is not possible as it gives a negative measure of manipulability and w > 0.

For Maximal Configuration: $\theta_2 = 0^o$ or 180^o

Using Matlab calculations for calculating singular values:
Assume that $\theta_1 = 0°$. The maximal singular value for $\theta_2 = 0°$ was determined using MATLAB and is 29. The highest singular value for 2=180° was determined using MATLAB and is 5. Calculated using MATLAB, the highest singular value for 2=90° is 14.5208.

We now calculate manipulability ellipsoids for each workspace configuration using the code shown below:

```
%define global variables: link lengths
global a1
global a2

%assign value to the variables
a1 = 3;
a2 = 2;
r1 = a1 - a2;
r2 = a1 + a2;

%grid the workspace
r = r1:0.5:r2;
theta = 0:0.05*pi:1.95*pi;

%initiate values needed for plotting
measure = zeros(1,length(r)*length(theta));
x = zeros(1,length(r)*length(theta));
y = zeros(1,length(r)*length(theta));
k = 1;
m = 0.1;
```

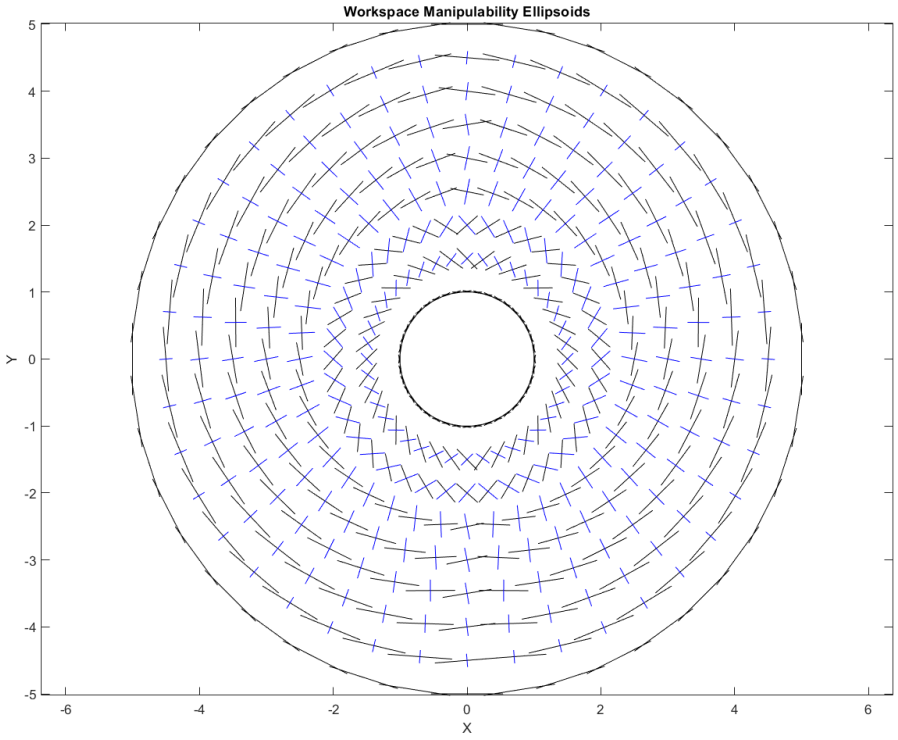
```
%%calculate manipulability for each workspace configuration
for i = 1:1:length(r)
    for j = 1:1:length(theta)
        x(k) = r(i)*cos(theta(j));
        y(k) = r(i)*sin(theta(j));
        draw_ellipsoid(x(k),y(k),m);
        k = k+1;
    end
end
figure
plot3(x,y,measure);
xlabel('X');
ylabel('Y');
zlabel('Measure');
```

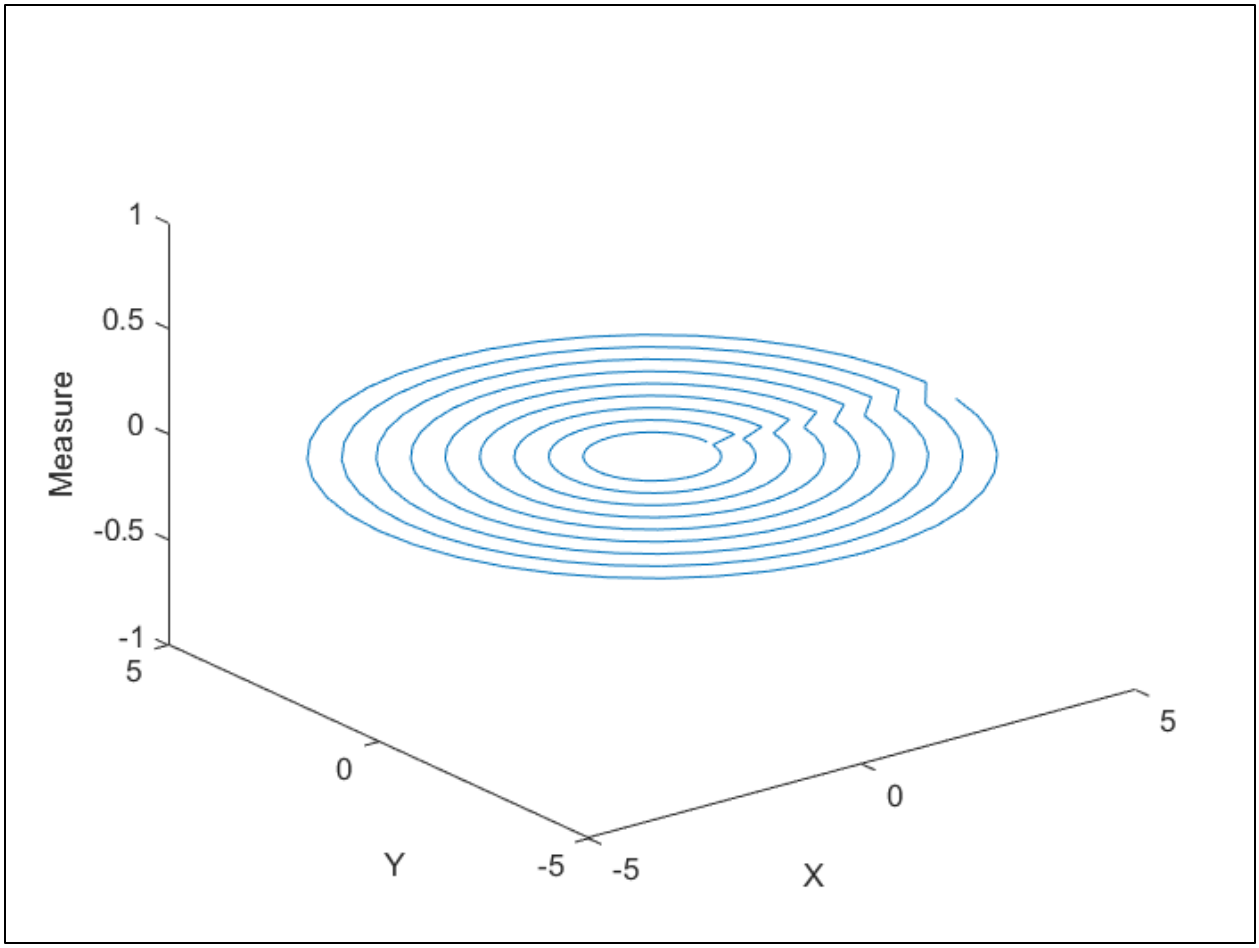
A separate draw_ellipsoid function is made that plots semi-major axis and semi-minor axis lines:

```
function [] = draw_ellipsoid(x, y, k)
%Link lengths
global a1
global a2

% configuration calculation
g1 = acos((x.^2 + y.^2 - a1.^2 - a2.^2)/(2*a1*a2));
g2 = - acos((x.^2 + y.^2 - a1.^2 - a2.^2)/(2*a1*a2));
theta1 = atan2(y, x) - atan(a2*sin(g1)/(a1 + a2*cos(g1)));
theta2 = atan2(y, x) - atan(a2*sin(g2)/(a1 + a2*cos(g2)));

% choose the elbow up configuration
if a1*sin(theta1)>a1*sin(theta2)
    theta2 = g1;
elseif a1*sin(theta1)<a1*sin(theta2)
    theta1 = theta2;
    theta2 = g2;
else
    theta2 = g1;
end
%Jacobian calculation
J = [-a1*sin(theta1)-a2*sin(theta1+theta2), -a2*sin(theta1+theta2);...
    a1*cos(theta1)+a2*cos(theta1+theta2), a2*cos(theta1+theta2)];

%calculate the eigen value of Jacobian
A = J*J';
[U, S, V] = svd(A);

%get length for axis
delta1 = sqrt(S(1,1));
delta2 = sqrt(S(2,2));

%get starting and ending point for each axis
x_1 = [x - delta1*U(1,1)*k, x, x + delta1*U(1,1)*k];
y_1 = [y - delta1*U(2,1)*k, y, y + delta1*U(2,1)*k];
x_2 = [x - delta2*U(1,2)*k, x, x + delta2*U(1,2)*k];
```

```
y_2 = [y - delta2*U(2,2)*k, y, y + delta2*U(2,2)*k];
plot(x_1,y_1,'k', x_2,y_2, 'b');
title('Workspace Manipulability Ellipsoids');
xlabel('X');
ylabel('Y');
axis equal
hold on
end
```

Below shown are the plots:





Problem 2:

In this problem we will use kinematic control using a two-degree-of-freedom manipulator to track desired EE position trajectories. The corresponding link lengths of the 2-link manipulator are assumed to be: L_1=4, L_2=4. The **end-effector position** $\underset{\sim}{\mathbf{x}}^d = (x, y)^T$ will be the manipulation variables of interest. This end-effector traverses TWO desired trajectories (1 full cycle in 10 seconds): (A) an ellipse with semi-major axis=1.5 oriented at an angle 30^0 w.r.t the horizontal and a semi-minor axis = 1, center at (0.5,0.5); and (B) a circle of radius 1.5 centered at (1,1).

The desired manipulation rates can be achieved by the manipulator using (an appropriately modified closed-loop variant of) resolved motion-rate control. Hence you decide to create 2 types of controllers that allows the actual manipulation rates $\dot{\underset{\sim}{\mathbf{x}}}$ to track the desired manipulation rates $\dot{\underset{\sim}{\mathbf{x}}}^d$

(i) Design a joint-space controller (whose error-dynamics time-constants are 3 seconds).

(ii) Design a closed-loop task-space controller such that the pole of the error dynamics along the X-axis is -5 and Y axis is at -10.

Simulate and plot the results of these various schemes for 3 full traversals of the ellipse of interest – i.e. simulate for 60 seconds in total using MATLAB/Simulink.

Solution:

Following is the information given:
Link Lengths $L_1 = 4$ and $L_2 = 4$
Circular and Elliptical end-effector trajectories

Time Constant = 3 seconds

The joint space controller was designed to track an ellipse and a circle. The algorithm is:

Step 1:

Determine the trajectory points using interpolation. It was done using polar coordinates of the geometrical form

Step 2:

Determine the velocities along x and y direction at each point

Step 3:

Initialize the start location and obtain the measurement to be fed into the controller

A closed-loop joint space and task space controllers are designed as follows:

$$\dot{\theta} = \dot{\theta^d} + K[\theta^d - \theta]$$
$$\dot{\theta} = \dot{x^d} + K[x^d - x]$$

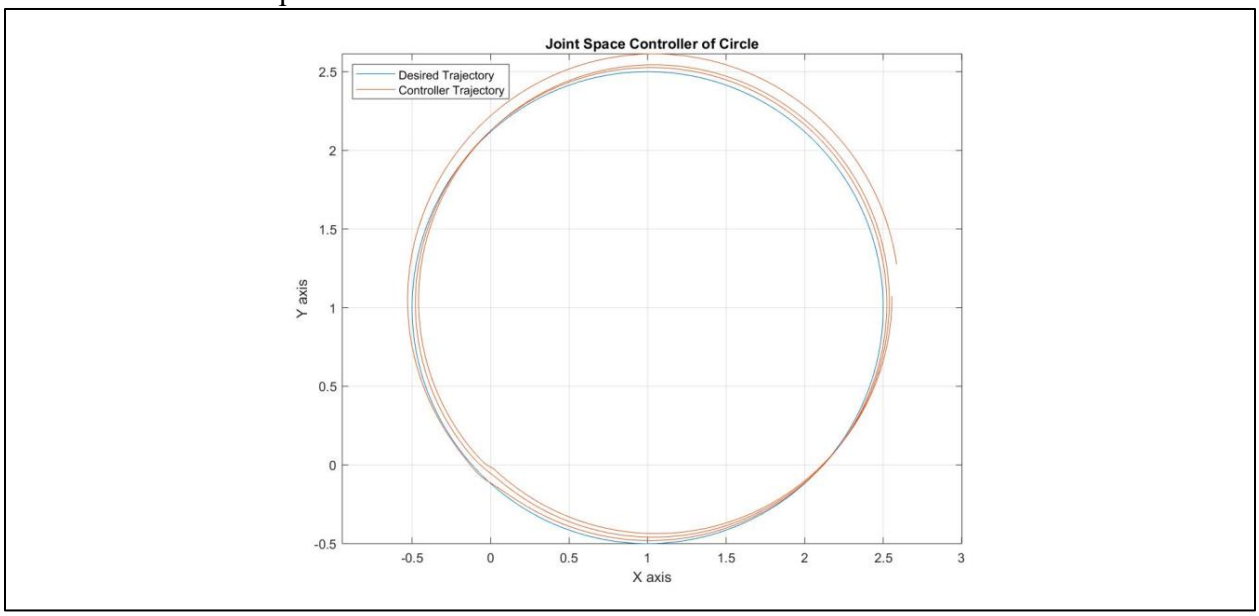
Where K is the gain and the terms in the square brackets are the error kinematics. Note that these are kinematic based controllers:

Below shown is the code for pseudo-Inverse based solution:

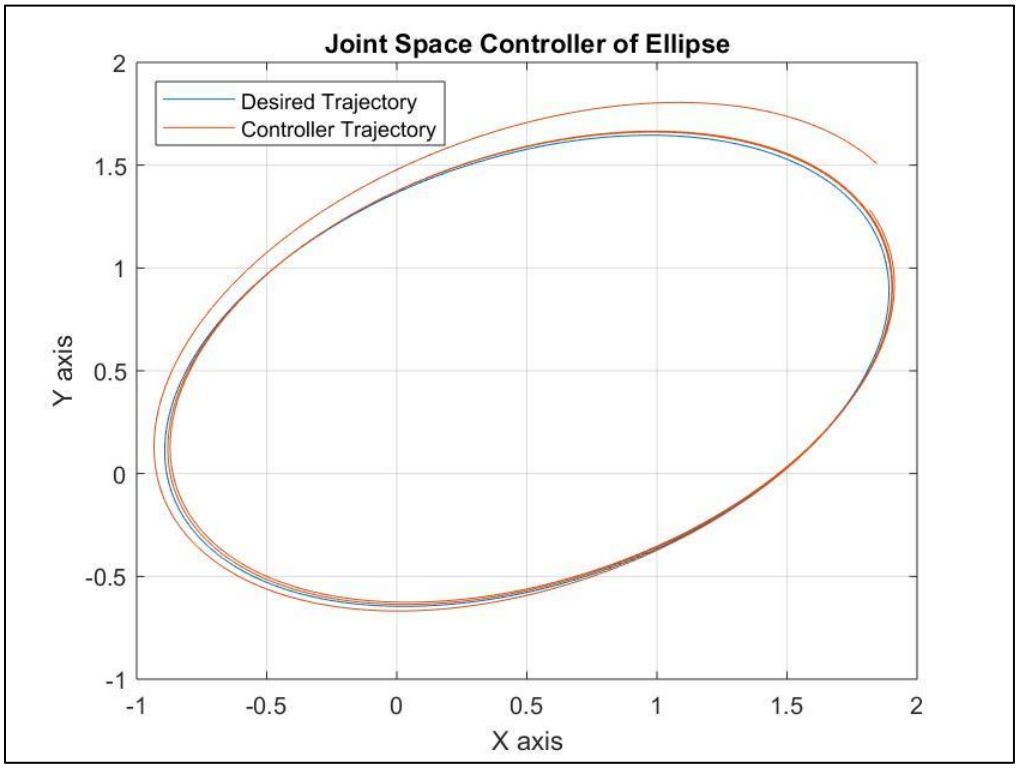
```
% % Circle
% clc;
% clear all;
% close all;
% t=0:0.01:10;
% len=length(t);
% w=2*pi/10;
% r=1.5;
% xc=[];yc=[];xc_dot=[];yc_dot=[];
% for i=1:len
%     xc=[xc,1+r*cos(w*t(i))];
%     xc_dot=[xc_dot,-r*sin(w*t(i))*w];
%     yc=[yc,1+r*sin(w*t(i))];
%     yc_dot=[yc_dot,r*cos(w*t(i))*w];
% end
% plot(xc,yc);
% grid on;
% axis equal;
% Ellipse
clc;
close all;
clear;
t = 0:.01:10;
len=length(t);
w = 2*pi/10;
a = 1.5; b = 1;
phi = deg2rad(30);
xc = a*cos(w*t);
yc = b*sin(w*t);
R = [cos(phi) -sin(phi);sin(phi) cos(phi)];
```

```
xdot=[-a*w*cos(phi)*sin(w*t)-b*w*sin(phi)*cos(w*t)
    -a*w*sin(phi)*sin(w*t)+b*w*cos(phi)*cos(w*t)];
xc_dot=xdot(1,:);
yc_dot=xdot(2,:);
plot(xc,yc);
grid on;
axis equal;
% Closed Loop Joint Space Controller
L1=4;L2=4;xe=xc(1);ye=yc(1);
[t1,t2]=RR_InvPosKin([L1,L2],xe,ye,1);
Xe=zeros(1,3*len);
Ye=zeros(1,3*len);
q=[t1;t2;t1+0.05;t2+0.05];
K=[1/3 0;...
    0 1/3];
for i=1:3*len
    [Xe(i),Ye(i)]=RR_ForwardPosKin([L1 L2],[q(1),q(2)]);
    t1=q(3);t2=q(4);
    J=[-L1*sin(t1)-L2*sin(t1+t2) -L2*sin(t1+t2);...
        L1*cos(t1)+L2*cos(t1+t2) L2*cos(t1+t2)];
    xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
    thetaDotDesired=J\xDot;
    thetaError=[q(1,1)-q(3,1);q(2,1)-q(4,1)];
    thetaDot=thetaDotDesired+K*thetaError;
    qDot=[thetaDotDesired;thetaDot];
    q=mod(q + qDot.*.01,2*pi);
end
hold on
plot(Xe,Ye,'r');
title('Pseudo Inverse - Ellipse');
legend('Desired', 'Actual');
xlabel('X');
ylabel('Y');
```

Below shown are the plots:



Joint Space Control

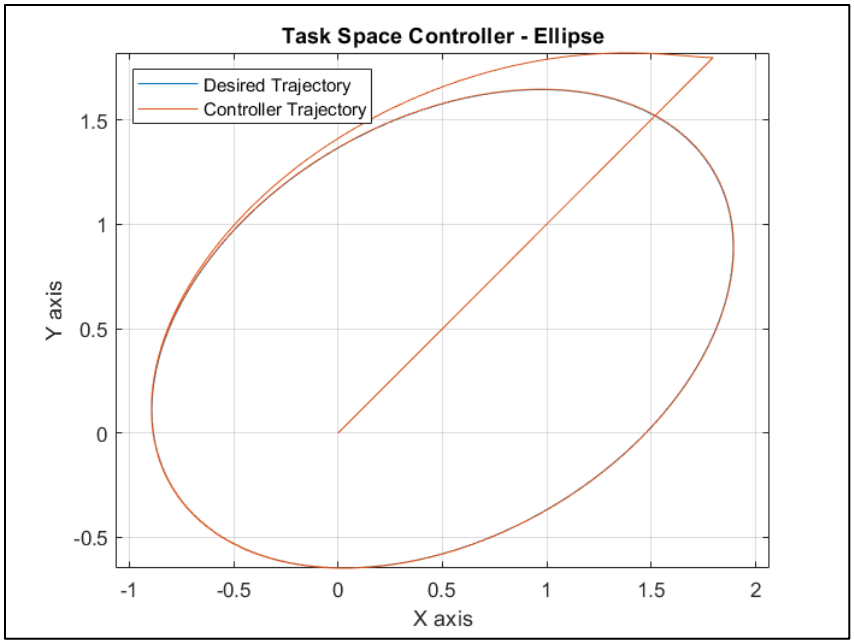


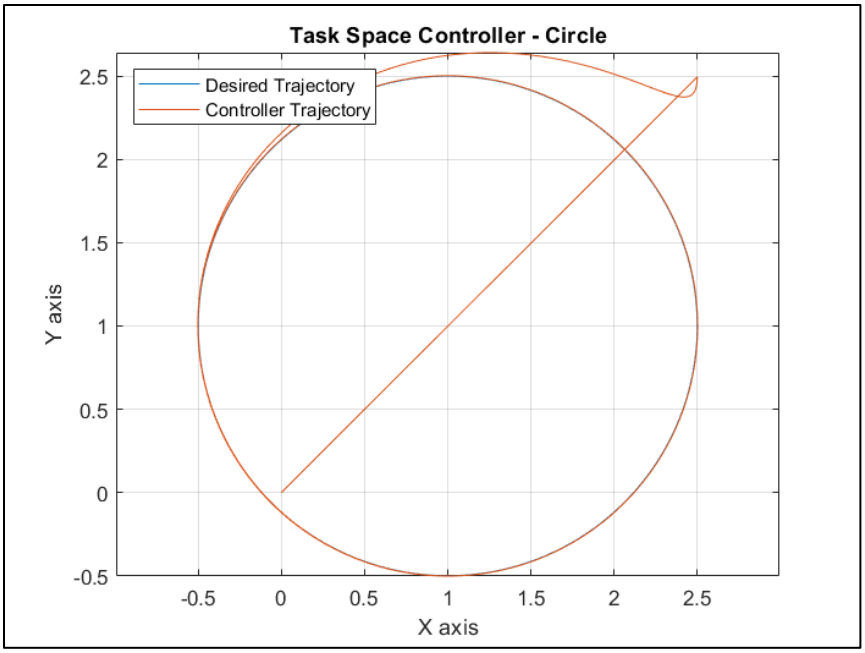
To reduce the error between the measured end effector and the desired end effector, we choose the desired coordinates of the end effector and adjust the joint variables in the task space controller. The controller receives its initial input thanks to the forward kinematics technique.

Below shown is the code:

```
%% Problem 2.2
% % Circle
% clc;
% clear;
% close all;
L1=4;L2=4;
w = 2*pi/10;
t = 0:.01:10;
len=length(t);
% r = 1.5;
% xc = r*cos(w*t);
% yc = r*sin(w*t);
% A = [xc;yc] + [1;1];
% xdot = [-r*sin(w*t)*w;r*cos(w*t)*w];
% plot(A(1,:),A(2,:))
% axis equal;
% Ellipse
clc;
w = 2*pi/10;
t = 0:.01:10;
```

```
p=1.5;q=1;
phi=deg2rad(30);
xc=p*cos(w*t);
yc=q*sin(w*t);
R=[cos(phi) -sin(phi);sin(phi) cos(phi)];
A=R*[xc;yc]+[.5;.5];
xdot=[-p*w*cos(phi)*sin(w*t)-q*w*sin(phi)*cos(w*t);-
p*w*sin(phi)*sin(w*t)+q*w*cos(phi)*cos(w*t)];
xc_dot=xdot(1,:);
yc_dot=xdot(2,:);
plot(A(1,:),A(2,:))
axis equal;
% Closed Loop Task Space
a = [4,4];
xe = A(1,1);
ye = A(2,1);
[t1,t2]=RR_InvPosKin([L1,L2],xe,ye,1);
K = [-1 0;0 -1];
th1 = t1; th2 = t2;
X=RR_ForwardPosKin([4 4],[th1,th2]);
Xe = zeros(3000,1);
Ye = zeros(3000,1);
for i=1:3*len
    XError = X - A(:,mod(i,1000)+1);
    XDot = K*XError + xdot(:,mod(i,1000)+1);
    X = X + XDot.*.01;
    J = [-a(1)*sin(th1)-a(2)*sin(th1+th2) -a(2)*sin(th1+th2);
        a(1)*cos(th1)+a(2)*cos(th1+th2) a(2)*cos(th1+th2)];
    ThDot = J\XDot;
    th1 = th1 + ThDot(1).*.01;
    th2 = th2 + ThDot(2).*.01;
    Xe(i+1) = X(1);
    Ye(i+1)=X(2);
end
hold on
plot(Xe,Ye);
grid on
xlabel('X axis');
ylabel('Y axis');
title('Task Space Controller - Ellipse');
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest');
```





Problem 3:

In this problem we will consider the following three-degree-of-freedom manipulator (considering only the positions and not the orientations). The corresponding link lengths of the 3-link manipulator are assumed to be: L_1 =2, L_2 =3, L_3 =1.5. The **end-effector position** described by $\underset{\sim}{\mathbf{x}}^d = (x, y)^T$ will be the manipulation variables of interest.

This end-effector traverses the same TWO desired trajectories as Problem 2 (1 full cycle in 10 seconds): (A) an ellipse with semi-major axis=1.5 oriented at an angle 30^0 w.r.t the horizontal and a semi-minor axis = 1, center at (0.5,0.5); and (B) a circle of radius 1.5 centered at (1,1).

We will now examine the effects of different methods of redundancy resolution to this problem at hand.

(i) Using the traditional pseudo-inverse solution.

(ii) Adding auxiliary constraints (one at a time) on the joint space variables of the form to resolve redundancy:

a. $\dot{\theta}_2 + \dot{\theta}_3 = 0$

b. $\dot{\theta}_1 = 0$

(iii) Using the minimization of an artificial potential described on the joint space as a secondary manipulation criterion to the traditional pseudoinverse solution (where $V = \theta_1^2 + 0.25\theta_2^2 + 0.66\theta_3^2$)

Simulate and plot the results of these various schemes for 3 full traversals of the ellipse of interest

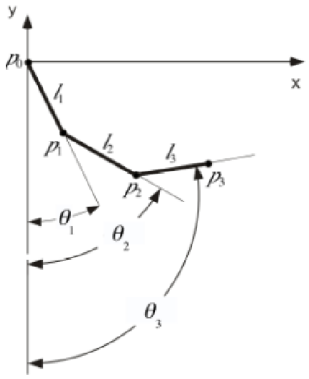


Figure 3: A 3-d.o.f. planar manipulator with absolute joint angles shown.

Solution:

It can be difficult to identify the controllable inputs in a manipulator with more than two links since there are more joint space variables than there are task space variables. This gives rise to kinematic redundancy. The pseudo-inverse is one of the solutions for resolving redundancy.

$$\theta = \dot{J}^{\#}\dot{x}$$

If A is full column rank, meaning $\mathbf{rank}(A) = n \leq m$, that is, $A^T A$ is not singular, then A† is a left inverse of A, in the sense that $A^\dagger A = I_n$. We have the closed-form expression

$$A^\dagger = (A^T A)^{-1} A^T.$$

If A is full row rank, meaning $\mathbf{rank}(A) = m \leq n$, that is, AA^T is not singular, then A† is a right inverse of A, in the sense that $AA^\dagger = I_m$. We have the closed-form expression

$$A^\dagger = A^T (AA^T)^{-1}.$$

In Matlab, it can be done using pinv function.

Below shown is the code:

```
%% Problem 3
% Circle
clc;
clear;
t=0:0.01:10;
len=length(t);
w=2*pi/10;
r=1.5;
xc=[];yc=[];xc_dot=[];yc_dot=[];
for i=1:len
    xc=[xc,1+r*cos(w*t(i))];
    xc_dot=[xc_dot,-r*sin(w*t(i))*w];
    yc=[yc,1+r*sin(w*t(i))];
    yc_dot=[yc_dot,r*cos(w*t(i))*w];
end
figure
plot(xc,yc);
```

```
grid on;
axis equal;
% Pseudo Inverse Solution
L1=2;L2=3;L3=1.5;
xe=xc(1);ye=yc(1);

t1 = -30*pi/180;
t2 = 97.0879*pi/180;
t3 = -40.6459*pi/180;
Xe=zeros(1,3*len);
Ye=zeros(1,3*len);
q=[t1;t2;t3];
for i=1:3*len
    Xe(i)=L1*cos(q(1,1))+L2*cos(q(2,1))+L3*cos(q(3,1));
    Ye(i)=L1*sin(q(1,1))+L2*sin(q(2,1))+L3*sin(q(3,1));
    t1=q(1,1);t2=q(2,1);t3=q(3,1);
    J=[-L1*sin(t1) -L2*sin(t2) -L3*sin(t3);...
        L1*cos(t1) L2*cos(t2) L3*cos(t3)];
    xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
    tDot=(pinv(J))*xDot;
    q=mod(q + tDot.*.01,2*pi);
end
hold on
plot(Xe,Ye)
grid on
xlabel('X axis')
ylabel('Y axis')
title('Pseudo Inverse Controller - Circle')
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest')
%Ellipse
w = 2*pi/10;
t = 0:.01:10;
len=length(t);
a = 1.5;b = 1;
phi = deg2rad(30);
xc = a*cos(w*t);
yc = b*sin(w*t);
R = [cos(phi) -sin(phi);sin(phi) cos(phi)];
A = R*[xc;yc] + [.5;.5];
xdot=[-a*w*cos(phi)*sin(w*t)-b*w*sin(phi)*cos(w*t);-
a*w*sin(phi)*sin(w*t)+b*w*cos(phi)*cos(w*t)];
xc_dot=xdot(1,:);
yc_dot=xdot(2,:);
figure
plot(A(1,:),A(2,:));
grid on;
axis equal;
% Pseudo Inverse Solution
L1=2;L2=3;L3=1.5;
xe=xc(1);ye=yc(1);

t1 = -30*pi/180;
t2 = 97.0879*pi/180;
t3 = -40.6459*pi/180;
Xe=zeros(1,3*len);
```

```
Ye=zeros(1,3*len);
q=[t1;t2;t3];
for i=1:3*len
    Xe(i)=L1*cos(q(1,1))+L2*cos(q(2,1))+L3*cos(q(3,1));
    Ye(i)=L1*sin(q(1,1))+L2*sin(q(2,1))+L3*sin(q(3,1));
    t1=q(1,1);t2=q(2,1);t3=q(3,1);
    J=[-L1*sin(t1) -L2*sin(t2) -L3*sin(t3);...
        L1*cos(t1) L2*cos(t2) L3*cos(t3)];
    xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
    tDot=(pinv(J))*xDot;
    q=mod(q + tDot.*.01,2*pi);
end
hold on
plot(Xe,Ye);
grid on
xlabel('X axis');
ylabel('Y axis');
title('Pseudo Inverse Controller - Ellipse');
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest')
```

(ii)

To add in the auxiliary constraints, they can be included in the Jacobian itself as shown below:

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ 0\end{bmatrix} = \begin{bmatrix}r11 & r12 & r13\\ r21 & r22 & r23\\ 0 & 1 & 1\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{bmatrix}$$

The above equation would include both the auxiliary constraints of part a and b (1 0 0 in the last row) respectively. It can modified as per the requirement. As observed, Jacobian is now a square matrix and is invertible.

Following is the code for both the methods:

```
%% problem 3.2 2a
% Circle
clc;
clear;
close;
t=0:0.01:10;
len=length(t);
w=2*pi/10;
r=1.5;
xc=[];yc=[];xc_dot=[];yc_dot=[];
for i=1:len
xc=[xc,1+r*cos(w*t(i))];
xc_dot=[xc_dot,-r*sin(w*t(i))*w];
yc=[yc,1+r*sin(w*t(i))];
yc_dot=[yc_dot,r*cos(w*t(i))*w];
end
figure
plot(xc,yc);
```

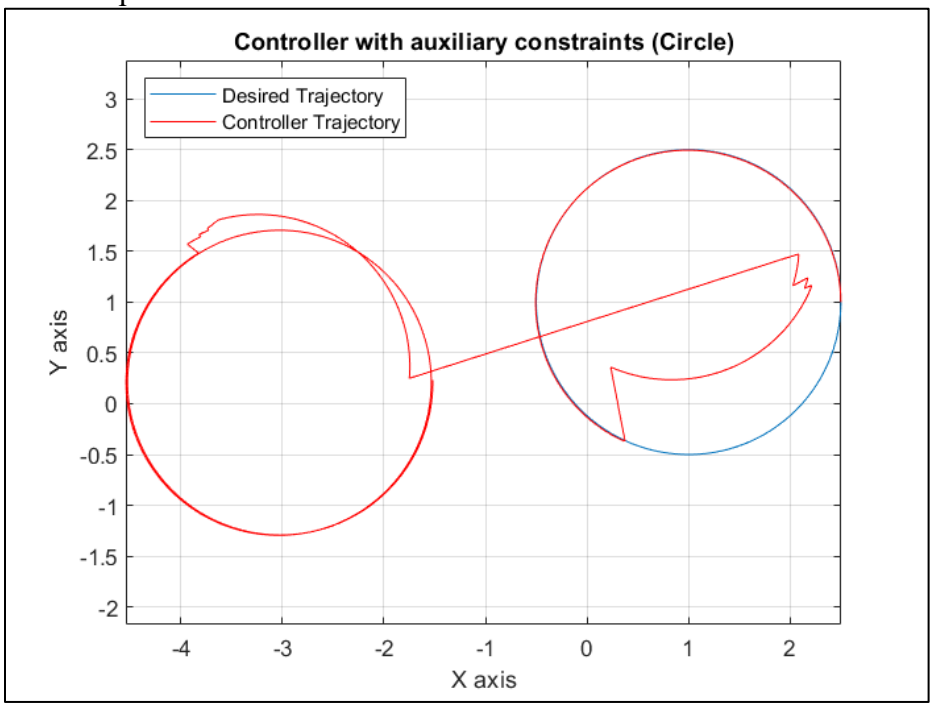
```
grid on;
axis equal;

L1=2; L2=3; L3=1.5;
xe=xc(1);ye=yc(1);
t1=-30*pi/180;
t2=97.0879*pi/180;
t3=-40.6459*pi/180;
Xe=zeros(1,3*len);
Ye=zeros(1,3*len);
q=[t1;t2;t3];
for i=1:3*len
Xe(i)=L1*cos(q(1,1))+L2*cos(q(2,1))+L3*cos(q(3,1));
Ye(i)=L1*sin(q(1,1))+L2*sin(q(2,1))+L3*sin(q(3,1));
t1=q(1,1);t2=q(2,1);t3=q(3,1);
J=[-L1*sin(t1) -L2*sin(t2) -L3*sin(t3);...
L1*cos(t1) L2*cos(t2) L3*cos(t3)];
Ja=[J;0 1 1];
xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
yDot=[xDot;0];
Jh=inv(Ja);
tDot=Jh*yDot;
q=mod(q + tDot.*0.01,2*pi);
end
hold on
plot(Xe,Ye,'r')
grid on
xlabel('X axis')
ylabel('Y axis')
title('Controller with auxiliary constraints (Circle)')
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest')
% Ellipse

w = 2*pi/10;
t = 0:.01:10;
len=length(t);
a = 1.5;b = 1;
phi = deg2rad(30);
xc = a*cos(w*t);
yc = b*sin(w*t);
R = [cos(phi) -sin(phi);sin(phi) cos(phi)];
A = R*[xc;yc] + [.5;.5];
figure
%plot(A(1,:),A(2,:))
xdot0 = [-a*sin(w*t)*w;b*cos(w*t)*w];
xdot=[-a*w*cos(phi)*sin(w*t)-b*w*sin(phi)*cos(w*t);-
a*w*sin(phi)*sin(w*t)+b*w*cos(phi)*cos(w*t)];
xc_dot=xdot(1,:);
yc_dot=xdot(2,:);

plot(A(1,:),A(2,:));
grid on;
axis equal;
% Pseudo Inverse Solution
L1=2;L2=3;L3=1.5;
```

```
xe=xc(1);ye=yc(1);

t1=-30*pi/180;
t2=97.0879*pi/180;
t3=-40.6459*pi/180;
Xe=zeros(1,3*len);
Ye=zeros(1,3*len);
q=[t1;t2;t3];
C=t2+t3;
for i=1:3*len
    Xe(i)=L1*cos(q(1,1))+L2*cos(q(2,1))+L3*cos(q(3,1));
    Ye(i)=L1*sin(q(1,1))+L2*sin(q(2,1))+L3*sin(q(3,1));
    t1=q(1,1);t2=q(2,1);t3=q(3,1);
    J=[-L1*sin(t1) -L2*sin(t2) -L3*sin(t3);...
        L1*cos(t1) L2*cos(t2) L3*cos(t3)];
    Ja=[J;0 1 1];
    xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
    yDot=[xDot;0];
    Jh=inv(Ja);
    tDot=Jh*yDot;
    q=mod(q + tDot.*0.01,2*pi);
end
hold on
plot(Xe,Ye,'r')
grid on
xlabel('X axis')
ylabel('Y axis')
title('Controller with auxiliary constraints (Ellipse)')
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest')

%% Problem 3.2b
%% problem 3.2
% Circle
clc;
clear;
close;
t=0:0.01:10;
len=length(t);
w=2*pi/10;
r=1.5;
xc=[];yc=[];xc_dot=[];yc_dot=[];
for i=1:len
xc=[xc,1+r*cos(w*t(i))];
xc_dot=[xc_dot,-r*sin(w*t(i))*w];
yc=[yc,1+r*sin(w*t(i))];
yc_dot=[yc_dot,r*cos(w*t(i))*w];
end
figure
plot(xc,yc);
grid on;
axis equal;

L1=2; L2=3; L3=1.5;
xe=xc(1);ye=yc(1);
t1=-30*pi/180;
```

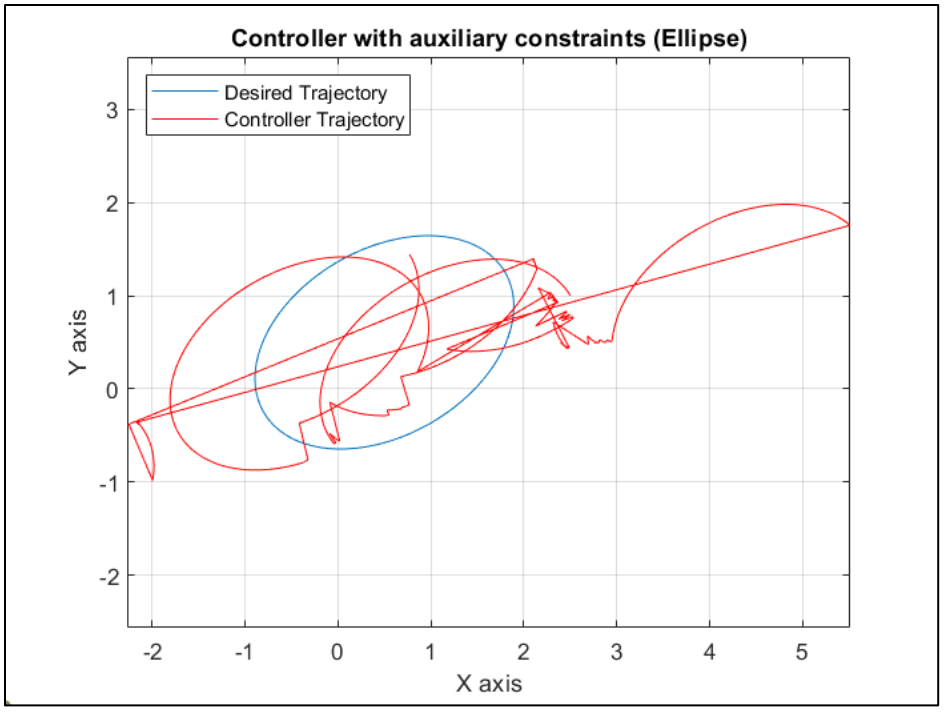
```
t2=97.0879*pi/180;
t3=-40.6459*pi/180;
Xe=zeros(1,3*len);
Ye=zeros(1,3*len);
q=[t1;t2;t3];
for i=1:3*len
Xe(i)=L1*cos(q(1,1))+L2*cos(q(2,1))+L3*cos(q(3,1));
Ye(i)=L1*sin(q(1,1))+L2*sin(q(2,1))+L3*sin(q(3,1));
t1=q(1,1);t2=q(2,1);t3=q(3,1);
J=[-L1*sin(t1) -L2*sin(t2) -L3*sin(t3);...
L1*cos(t1) L2*cos(t2) L3*cos(t3)];
Ja=[J;1 0 0];
xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
yDot=[xDot;0];
Jh=inv(Ja);
tDot=Jh*yDot;
q=mod(q + tDot.*0.01,2*pi);
end
hold on
plot(Xe,Ye,'r')
grid on
xlabel('X axis')
ylabel('Y axis')
title('Controller with auxiliary constraints (Circle)')
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest')
% Ellipse

w = 2*pi/10;
t = 0:.01:10;
len=length(t);
a = 1.5;b = 1;
phi = deg2rad(30);
xc = a*cos(w*t);
yc = b*sin(w*t);
R = [cos(phi) -sin(phi);sin(phi) cos(phi)];
A = R*[xc;yc] + [.5;.5];
figure
%plot(A(1,:),A(2,:))
xdot0 = [-a*sin(w*t)*w;b*cos(w*t)*w];
xdot=[-a*w*cos(phi)*sin(w*t)-b*w*sin(phi)*cos(w*t);-
a*w*sin(phi)*sin(w*t)+b*w*cos(phi)*cos(w*t)];
xc_dot=xdot(1,:);
yc_dot=xdot(2,:);

plot(A(1,:),A(2,:));
grid on;
axis equal;
% Pseudo Inverse Solution
L1=2;L2=3;L3=1.5;
xe=xc(1);ye=yc(1);

t1=-30*pi/180;
t2=97.0879*pi/180;
t3=-40.6459*pi/180;
Xe=zeros(1,3*len);
```

```
Ye=zeros(1,3*len);
q=[t1;t2;t3];
C=t2+t3;
for i=1:3*len
    Xe(i)=L1*cos(q(1,1))+L2*cos(q(2,1))+L3*cos(q(3,1));
    Ye(i)=L1*sin(q(1,1))+L2*sin(q(2,1))+L3*sin(q(3,1));
    t1=q(1,1);t2=q(2,1);t3=q(3,1);
    J=[-L1*sin(t1) -L2*sin(t2) -L3*sin(t3);...
        L1*cos(t1) L2*cos(t2) L3*cos(t3)];
    Ja=[J;1 0 0];
    xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
    yDot=[xDot;0];
    Jh=inv(Ja);
    tDot=Jh*yDot;
    q=mod(q + tDot.*0.01,2*pi);
end
hold on
plot(Xe,Ye,'r')
grid on
xlabel('X axis')
ylabel('Y axis')
title('Controller with auxiliary constraints (Ellipse)')
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest')
```

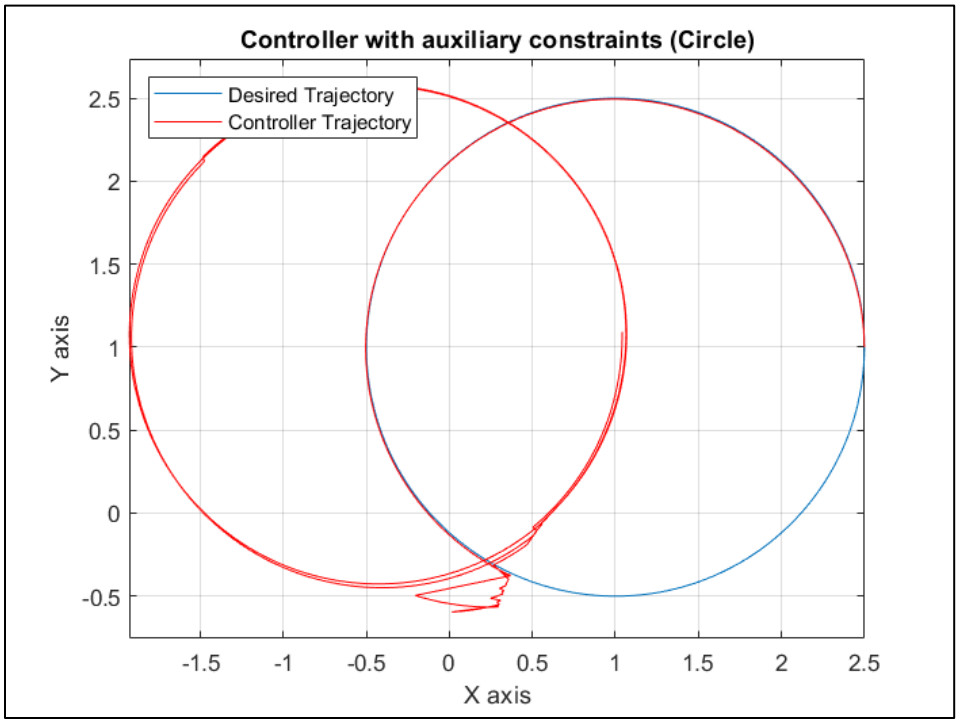
Below shown are the plots:



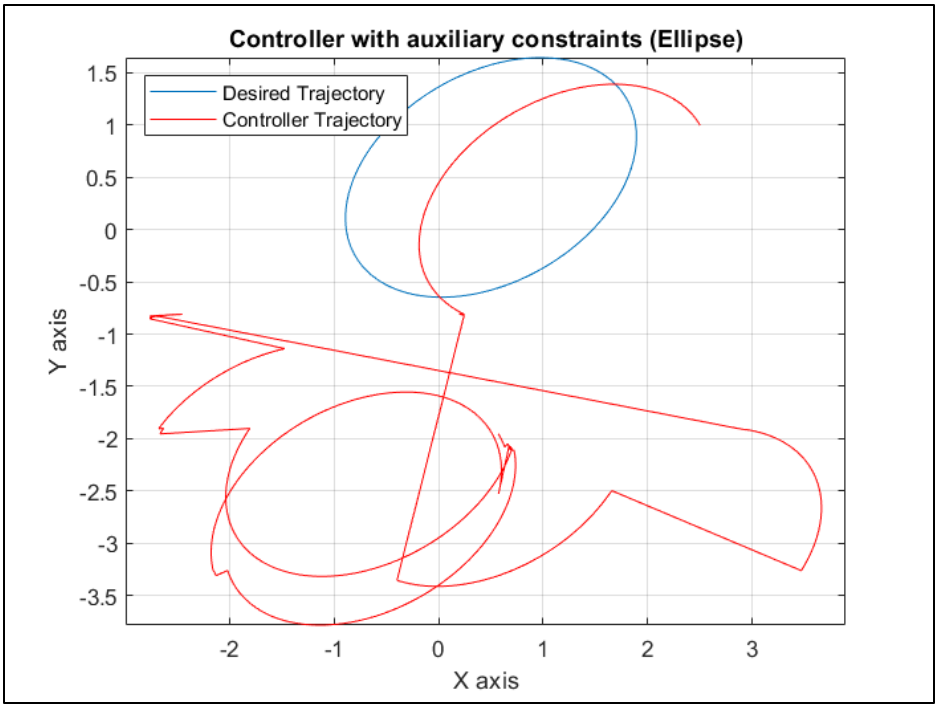
2a


Controller with auxiliary constraints (Ellipse)
Desired Trajectory
Controller Trajectory
Y axis
X axis
3
2
1
0
-1
-2
-2
-1
0
1
2
3
4
5

2a


Controller with auxiliary constraints (Circle)
Desired Trajectory
Controller Trajectory
Y axis
X axis
2.5
2
1.5
1
0.5
0
-0.5
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5

2b



2b

As observed, the redundancy resolution does not produce the desired trajectory. A lot depends on the initial conditions, time step and the method itself. There are better proven methods one of them which is using the minimization of artificial potential function.

Please refer the following link:

https://www.google.com/search?q=minimization+of+artificial+potential+function+redundancy&rlz=1C1VDKB_enUS1010US1010&source=lnms&tbm=vid&sa=X&ved=2ahUKEwjmgoKntM37AhUsRjABHVGFCHAQ_AUoA3oECAEQBQ&biw=1536&bih=792&dpr=1.25#fpstate=ive&vld=cid:717abb63,vid:Ls8EBoG_SEQ

(iii)

$$V = \theta_1^2 + 0.25\theta_2^2 + 0.66\theta_3^2$$

Redundancy resolution using the above method is:

$$\dot{\theta}=J^{\#}\dot{x}+[I-J^{\#}J](-\nabla V)^T$$

$J^{\#}x$ is the primary solution, and $[IJ^{\#}J]$ is the heuristic solution, where (V) is the Jacobian of V with respect to Θ. The questions above provide a description of how to obtain both Jacobians. is the control input that can be combined to determine the joint angles at each time step.

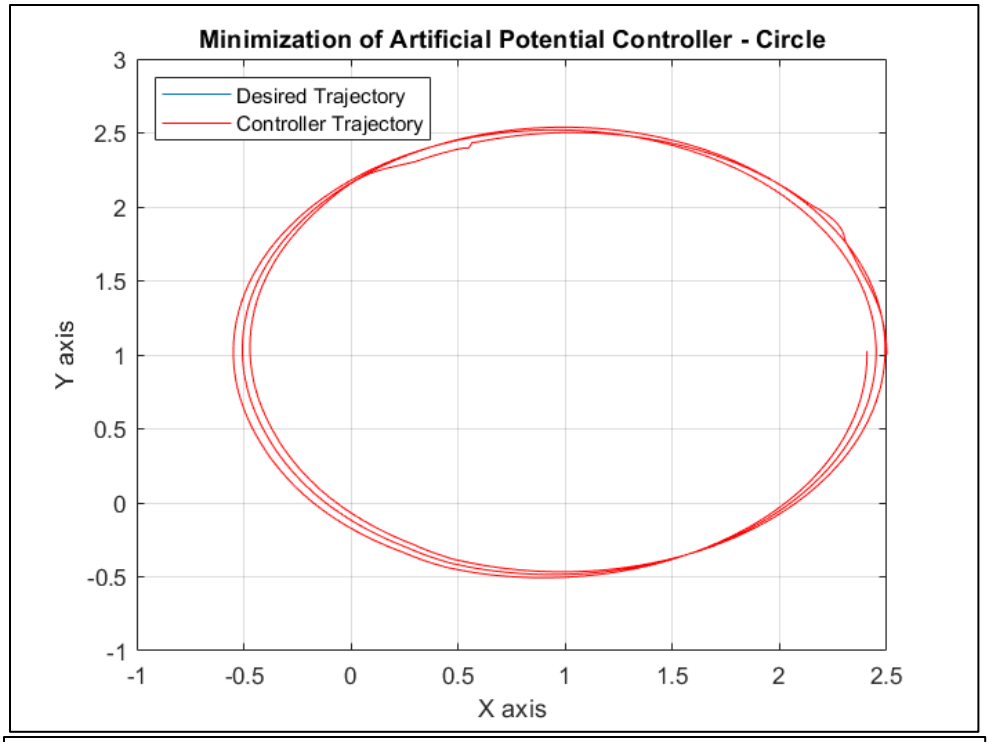
Following is the code:

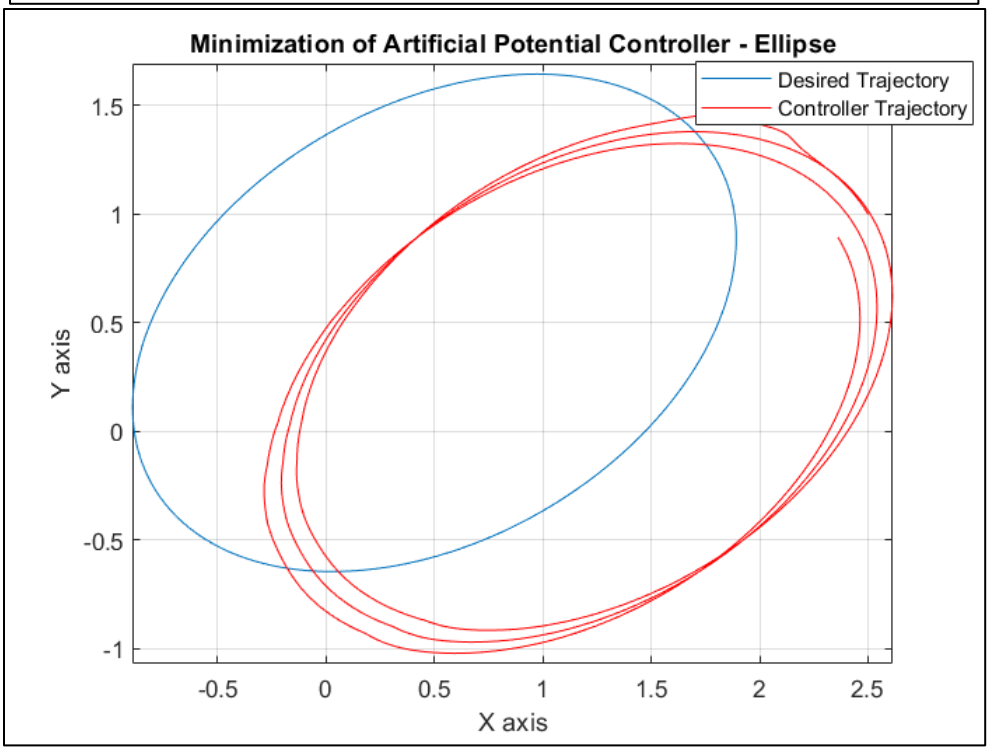
```
%% Problem 3.3
% Circle
clc;
```

```
clear;
clc;

t=0:0.01:10;
len=length(t);
w=2*pi/10;
r=1.5;
xc=[];yc=[];xc_dot=[];yc_dot=[];
for i=1:len
    xc=[xc,1+r*cos(w*t(i))];
    xc_dot=[xc_dot,-r*sin(w*t(i))*w];
    yc=[yc,1+r*sin(w*t(i))];
    yc_dot=[yc_dot,r*cos(w*t(i))*w];
end
plot(xc,yc,'k');
grid on;
axis equal;

L1=2; L2=3; L3=1.5;
xe=xc(1);ye=yc(1);
t1=-30*pi/180;
t2=97.0879*pi/180;
t3=-40.6459*pi/180;
Xe=zeros(1,3*len);
Ye=zeros(1,3*len);
plot(Xe,Ye);
q=[t1;t2;t3];
for i=1:3*len
    Xe(i)=L1*cos(q(1,1))+L2*cos(q(2,1))+L3*cos(q(3,1));
    Ye(i)=L1*sin(q(1,1))+L2*sin(q(2,1))+L3*sin(q(3,1));
    t1=q(1,1);t2=q(2,1);t3=q(3,1);

    J=[-L1*sin(t1) -L2*sin(t2) -L3*sin(t3);...
        L1*cos(t1) L2*cos(t2) L3*cos(t3)];
    xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
    Jh=pinv(J);
    tDotp=Jh*xDot;
    v=-[2*t1;0.5*t2;1.32*t3];
    tDotn=(eye(3)-(Jh*J))*v;
    qDot=tDotp+tDotn;
    q=mod(q + qDot.*.01,2*pi);
end
hold on
plot(Xe,Ye,'r');
grid on
xlabel('X axis');
ylabel('Y axis');
title('Minimization of Artificial Potential Controller - Circle');
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest');

%% Ellipse
clc;
clear;

w = 2*pi/10;
```

```
t = 0:.01:10;
len=length(t);
a = 1.5;b = 1;
phi = deg2rad(30);
xc = a*cos(w*t);
yc = b*sin(w*t);
R = [cos(phi) -sin(phi);sin(phi) cos(phi)];
A = R*[xc;yc] + [.5;.5];
plot(A(1,:),A(2,:))
xdot0 = [-a*sin(w*t)*w;b*cos(w*t)*w];
xdot=[-a*w*cos(phi)*sin(w*t)-b*w*sin(phi)*cos(w*t);-
a*w*sin(phi)*sin(w*t)+b*w*cos(phi)*cos(w*t)];
xc_dot=xdot(1,:);
yc_dot=xdot(2,:);
plot(A(1,:),A(2,:));
grid on;
axis equal;

L1=2;L2=3;L3=1.5;
xe=xc(1);ye=yc(1);

t1=-30*pi/180;
t2=97.0879*pi/180;
t3=-40.6459*pi/180;
Xe=zeros(1,3*len);
Ye=zeros(1,3*len);
q=[t1;t2;t3];
for i=1:3*len
    Xe(i)=L1*cos(q(1,1))+L2*cos(q(2,1))+L3*cos(q(3,1));
    Ye(i)=L1*sin(q(1,1))+L2*sin(q(2,1))+L3*sin(q(3,1));
    t1=q(1,1);t2=q(2,1);t3=q(3,1);
    J=[-L1*sin(t1) -L2*sin(t2) -L3*sin(t3);...
        L1*cos(t1) L2*cos(t2) L3*cos(t3)];
    xDot=[xc_dot(:,mod(i,1000)+1);yc_dot(:,mod(i,1000)+1)];
    Jh=pinv(J);
    tDotp=Jh*xDot;
    v=-[2*t1;0.5*t2;1.32*t3];
    tDotn=(eye(3)-(Jh*J))*v;
    qDot=tDotp+tDotn;
    q=mod(q + qDot.*.01,2*pi);
end
hold on
plot(Xe,Ye,'r')
grid on
xlabel('X axis')
ylabel('Y axis')
title('Minimization of Artificial Potential Controller - Ellipse')
legend('Desired Trajectory','Controller Trajectory','Location','NorthWest')
```

Below shown are the plots:





As observed, minimization of artificial potential function tracks the trajectories much better than the pseudo inverse and the auxiliary constraints method. With suitable initial conditions using more accurate numerical methods, better tracking can be achieved.