Mohammad Shahab, 227598

EE 656, **HW#2**

Question 2

Here we are generating joints commands to make the end-effector reach the target. So, using Artificial Force approach, joints commands are computed automatically. Here, **two artificial forces are tested**:

1) $$\begin{bmatrix} F_{ax} \\ F_{ay} \end{bmatrix} = \frac{K}{\sqrt{(x_f - x)^2 + (y_f - y)^2}} \begin{bmatrix} x_f - x \\ y_f - y \end{bmatrix}$$

2) $$\begin{bmatrix} F_{ax} \\ F_{ay} \end{bmatrix} = K \begin{bmatrix} x_f - x \\ y_f - y \end{bmatrix}$$

- As $(\boldsymbol{x}, \boldsymbol{y})$ is the end-effector point and with final target point of $(\boldsymbol{x_f}, \boldsymbol{y_f})$.
- $\boldsymbol{K}$ is a tuning parameter to be chosen.
- In this exercise, $(x_f, y_f) = (3, 0)$

We have end-effector point,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L3 \cdot \sin(\theta_3 + \theta_2 + \theta_1) + L2 \cdot \sin(\theta_2 + \theta_1) + L1 \cdot \sin(\theta_1) \\ L3 \cdot \cos(\theta_3 + \theta_2 + \theta_1) + L2 \cdot \cos(\theta_2 + \theta_1) + L1 \cdot \cos(\theta_1) \end{bmatrix}$$

So, we can have **torques** effect on joints as

$$\mathrm{T}(\theta) = \begin{bmatrix} \mathrm{T}_1(\theta_1) \\ \mathrm{T}_2(\theta_2) \\ \mathrm{T}_3(\theta_3) \end{bmatrix} = \mathcal{J}_\theta(\theta)^T \begin{bmatrix} F_{ax} \\ F_{ay} \end{bmatrix}$$

With the **Jacobian** (transposed) defined as

$$\boldsymbol{\mathcal{J}_\theta(\theta)^T} = \begin{bmatrix} L3 \cdot \cos(\theta_3 + \theta_2 + \theta_1) + L2 \cdot \cos(\theta_2 + \theta_1) + L1 \cdot \cos(\theta_1) & -(L3 \cdot \sin(\theta_3 + \theta_2 + \theta_1) + L2 \cdot \sin(\theta_2 + \theta_1) + L1 \cdot \sin(\theta_1)) \\ L3 \cdot \cos(\theta_3 + \theta_2 + \theta_1) + L2 \cdot \cos(\theta_2 + \theta_1) & -(L3 \cdot \sin(\theta_3 + \theta_2 + \theta_1) + L2 \cdot \sin(\theta_2 + \theta_1)) \\ L3 \cdot \cos(\theta_3 + \theta_2 + \theta_1) & -L3 \cdot \sin(\theta_3 + \theta_2 + \theta_1) \end{bmatrix}$$

So, we can construct joints commands as the torques reflected,

$$\dot{\theta} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} \mathrm{T}_1(\theta_1) \\ \mathrm{T}_2(\theta_2) \\ \mathrm{T}_3(\theta_3) \end{bmatrix}$$

So, numerically, we can have joints commands be

$$\theta(k) = \theta(k-1) + \Delta \cdot \mathrm{T}(\theta(k-1))$$

With $\boldsymbol{k}$ as iteration step.

Δ is approximation step (tuning parameter)

In our exercise,

$$\theta(0) = \begin{bmatrix} -\frac{\pi}{2} \\ \frac{\pi}{2} \\ -\frac{\pi}{2} \end{bmatrix}$$

The 3DOF robotic arm has links of length: $\boldsymbol{L3 = L2 = L1 = 1}$

In MATLAB 2007b, throughout the program, $\Delta = \mathbf{0.01}$ is used. Also, algorithm stopping criteria is selected to be:

$$\left| dist(\begin{bmatrix} x_f \\ y_f \end{bmatrix}, \begin{bmatrix} x \\ y \end{bmatrix}) \right| \leq \epsilon$$

As $dist\left(\begin{bmatrix} x_f \\ y_f \end{bmatrix}, \begin{bmatrix} x \\ y \end{bmatrix}\right) = \sqrt{(x_f - x)^2 + (y_f - y)^2}$.

Stopping condition $\boldsymbol{\epsilon = 0.015}$

NOTE: Animation is generated to show the motion of the arm.

- **First approach**

    - We use artificial force #1
    - K = 0.25

Variables trajectories are shown below.

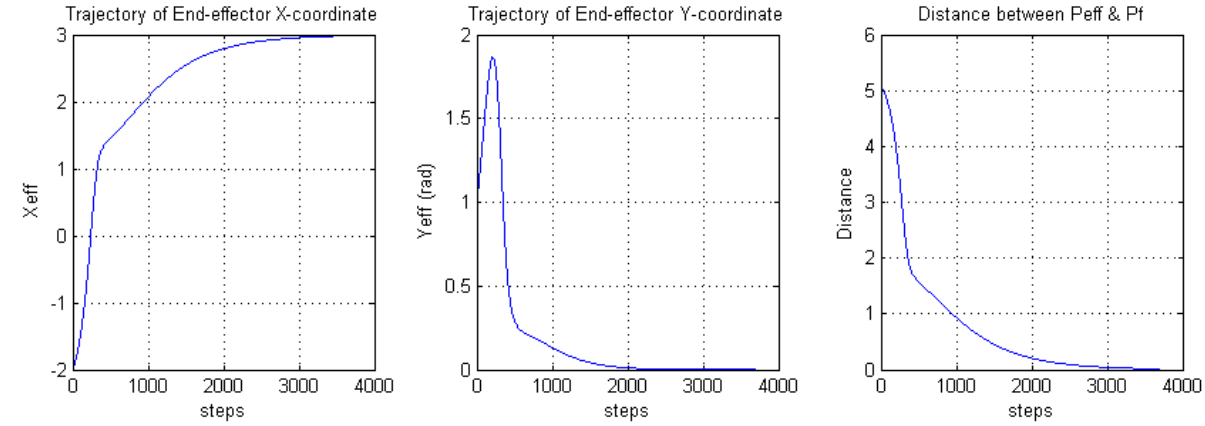

Figure 1: x-, y-coordinates & distance between final point and end-effector trajectories (with artificial force 1)

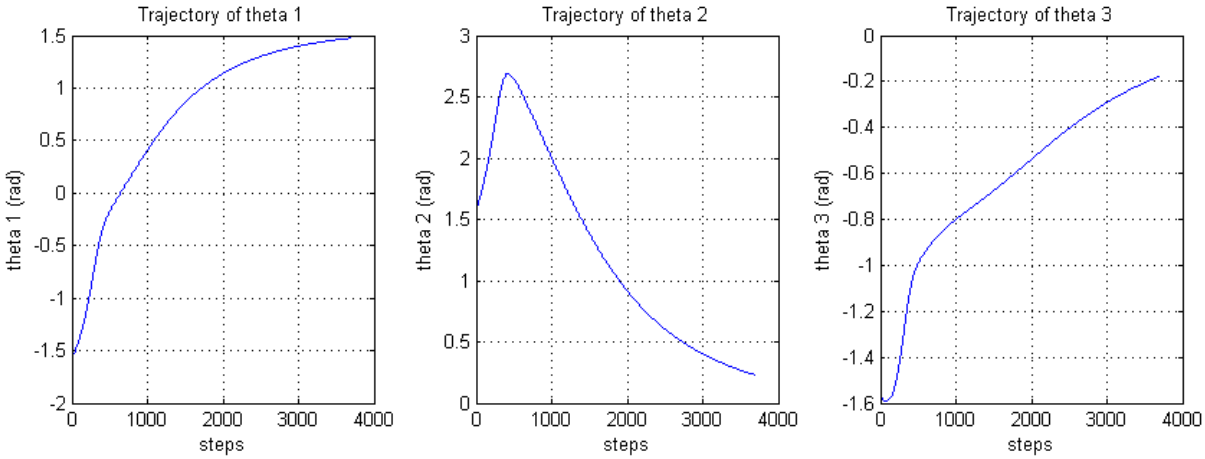

Figure 2: theta 1, theta 2 & theta 3 trajectories (radian)

Comments:

- You can see from above trajectories that end-effector approaches the target point.
- Y-coordinate trajectory experienced an overshoot.
- Overshoot also appears in joint 2 rotation.

- **2nd approach**

    - We use artificial force #2
    - K = 5 (note different structure of the force)

Variables trajectories are shown below.

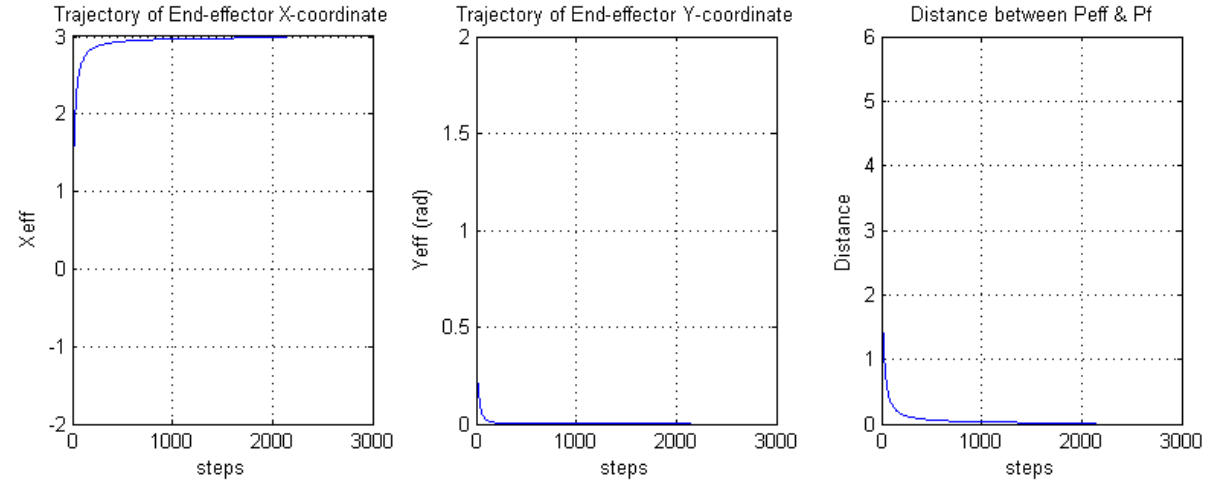

Figure 3: x-, y-coordinates & distance between final point and end-effector trajectories (with artificial force 2)

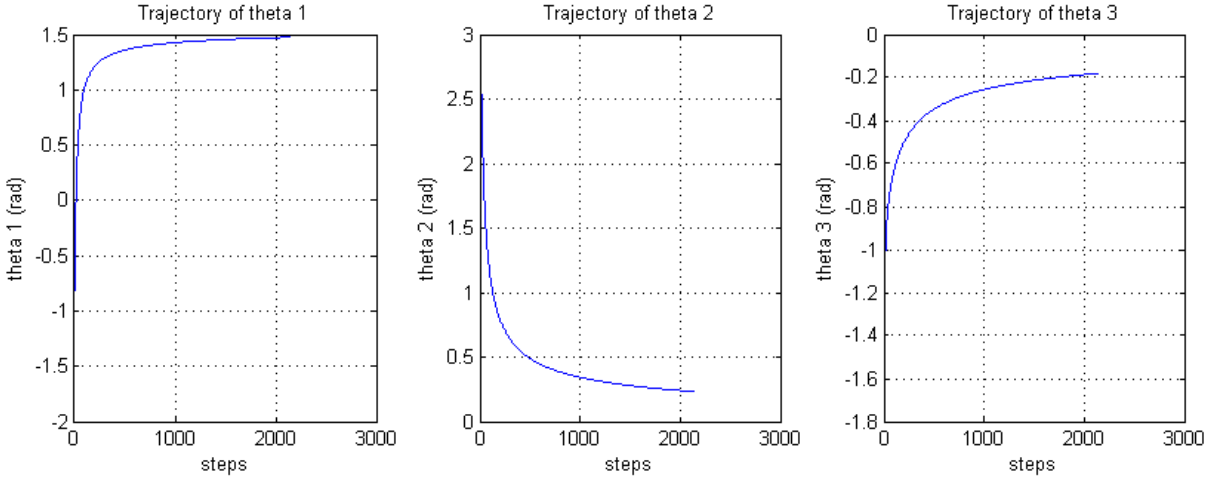

Figure 4: theta 1, theta 2 & theta 3 trajectories (radian) using Artificial Force 2

Comments:

- You can see from above trajectories that end-effector approaches the target point.
- This mechanism shows faster response
- Joints and end-effector evolution is smooth (no overshoot or oscillations)

MATLAB Code

```
clear all
close all
clc

%% Links Information
L1=1;
L2=1;
L3=1;

%% Target Point
xf=L1+L2+L3;
yf=0;

%% Algorithm Parameters

K=0.25; %force parameter
dt=0.01; %approximation step
eps=0.015; %stopping condition

dummy=1;

%% Initial Configuration
th1=-pi/2;
th2=pi/2;
th3=-pi/2;

%% Trajectory Generation
k=1;

while (dummy==1)

    %End-Effector
    x(k)=L3*sin(th1(k)+th2(k)+th3(k))+L2*sin(th1(k)+th2(k))+L1*sin(th1(k));
    y(k)=L3*cos(th1(k)+th2(k)+th3(k))+L2*cos(th1(k)+th2(k))+L1*cos(th1(k));

    P(:,k)=[x(k);y(k)];

    %Distance between end-effector & target
    distP(k)=sqrt(((xf-x(k))^2)+((yf-y(k))^2));

%     if (k>1)
%     if (abs(P(k)-P(k-1))<=[eps;eps])
%         dummy=0;
%         break
%     end
%     end

    %Stopping criteria
    if (distP(k)<=eps)
        dummy=0;
        break
    end

    %Artificial Force
    KF=K/distP(k);
    KF=K;
    Fax(k)=KF*(xf-x(k));
    Fay(k)=KF*(yf-y(k));
```

```
    %Torques Computation
    T1(k)=(L3*cos(th1(k)+th2(k)+th3(k))+L2*cos(th1(k)+th2(k))+L1*cos(th1(k)))*Fax(k)-
(L3*sin(th1(k)+th2(k)+th3(k))+L2*sin(th1(k)+th2(k))+L1*sin(th1(k)))*Fay(k);
    T2(k)=(L3*cos(th1(k)+th2(k)+th3(k))+L2*cos(th1(k)+th2(k)))*Fax(k)-
(L3*sin(th1(k)+th2(k)+th3(k))+L2*sin(th1(k)+th2(k)))*Fay(k);
    T3(k)=(L3*cos(th1(k)+th2(k)+th3(k)))*Fax(k)-(L3*sin(th1(k)+th2(k)+th3(k)))*Fay(k);

    %Joints commands
    th1(k+1)=th1(k)+dt*T1(k);
    th2(k+1)=th2(k)+dt*T2(k);
    th3(k+1)=th3(k)+dt*T3(k);

    k=k+1;
end

%% Plots
,figure
subplot(131)
plot(th1)
title('Trajectory of theta 1')
xlabel('steps')
ylabel('theta 1 (rad)')
grid
subplot(132)
plot(th2)
title('Trajectory of theta 2')
xlabel('steps')
ylabel('theta 2 (rad)')
grid
subplot(133)
plot(th3)
title('Trajectory of theta 3')
xlabel('steps')
ylabel('theta 3 (rad)')
grid

,figure
subplot(131)
plot(x)
title('Trajectory of End-effector X-coordinate')
xlabel('steps')
ylabel('Xeff')
grid
subplot(132)
plot(y)
title('Trajectory of End-effector Y-coordinate')
xlabel('steps')
ylabel('Yeff (rad)')
grid
subplot(133)
plot(distP)
title('Distance between Peff & Pf')
xlabel('steps')
ylabel('Distance')
grid

%% Animation Generation
% ,figure;
% % hold off;
% xL=[];
% yL=[];
```

```
% jj=10;
% kk=1:jj:length(th1);
% for kkk=1:length(kk)-1;
%
%     x1(kkk)=L1*sin(th1(kk(kkk)));
%     y1(kkk)=L1*cos(th1(kk(kkk)));
%
%     x2(kkk)=x1(kkk)+L2*sin(th1(kk(kkk))+th2(kk(kkk)));
%     y2(kkk)=y1(kkk)+L2*cos(th1(kk(kkk))+th2(kk(kkk)));
%
%     x3(kkk)=L3*sin(th1(kk(kkk))+th2(kk(kkk))+th3(kk(kkk)))+x2(kkk);
%     y3(kkk)=L3*cos(th1(kk(kkk))+th2(kk(kkk))+th3(kk(kkk)))+y2(kkk);
%
%     xp1=0:(sign(x1(kkk))*dt):x1(kkk);
%     xp2=x1(kkk):(sign(x2(kkk)-x1(kkk))*dt):x2(kkk);
%     xp3=x2(kkk):(sign(x3(kkk)-x2(kkk))*dt):x3(kkk);
%
%     yp1=(y1(kkk)./x1(kkk)).*xp1;
%     yp2=((y2(kkk)-y1(kkk))./(x2(kkk)-x1(kkk))).*(xp2-x1(kkk))+y1(kkk);
%     yp3=((y3(kkk)-y2(kkk))./(x3(kkk)-x2(kkk))).*(xp3-x2(kkk))+y2(kkk);
%
%   xL=[xp1 xp2 xp3];
%   yL=[yp1 yp2 yp3];
%
%
% axis_p=[-3.,3.5,-3.5,3];
% % ik=82;
% plot([0 x1(kkk) x2(kkk) x3(kkk)],[0 y1(kkk) y2(kkk)
y3(kkk)],'ko',xL,yL,'r',xf,yf,'rx');
% axis(axis_p)
% % hold on;
%
% M(kkk)=getframe;
% end
```