

**3-DoF Delta Manipulator Application for ping-pong ball package assembly**

|  |  |
| --- | --- |
| **Prepared by: Group 9**  Minh Hoang Nguyen (301243493)  Yuxiang Xie (301233084)  Dickson Ng (301187328) | **Prof:** Dr. Flavio Firmani  **TA:** Mr. Saeed Arasteh |



Table of Contents

[List of Figures 3](#_Toc499904992)

[Abstract/Executive Summary 4](#_Toc499904993)

[I. Trajectory Generation: 5](#_Toc499904994)

[1. Brief Introduction & Explanation: 5](#_Toc499904995)

[2. Result Plot: 6](#_Toc499904996)

[a. Joint Displacement: 6](#_Toc499904997)

[b. Joint Velocity: 7](#_Toc499904998)

[c. Joint Acceleration: 9](#_Toc499904999)

[d. End-Effector Motion in 3D: 10](#_Toc499905000)

[II. Jacobians: 10](#_Toc499905001)

[1. Jx and Jq: 10](#_Toc499905002)

[2. Singularities 12](#_Toc499905003)

[a. Singularity type I: 12](#_Toc499905004)

[b. Singularity type II: 13](#_Toc499905005)

[c. Singularity type III 13](#_Toc499905006)

[3. Forward Velocity Problem: 13](#_Toc499905007)

[4. Forward Static Problem: 14](#_Toc499905008)

[III. Conclusion: 14](#_Toc499905009)

[Appendix A 16](#_Toc499905010)

[Appendix B 16](#_Toc499905011)

[Main Script 16](#_Toc499905012)

[Inverse Kinematics 23](#_Toc499905013)

[Forward Kinematics 26](#_Toc499905014)

[Fixing Position Matrix 27](#_Toc499905015)

# List of Figures

[Figure 1.The manipulator's Trajectory from project I (Minh Nguyen, 2017)[1] 5](#_Toc499877440)

[Figure 2. Joint Displacement for ARM 1 6](#_Toc499877441)

[Figure 3. Joint Displacement for ARM 2 6](#_Toc499877442)

[Figure 4.Joint Displacement for ARM 3 7](#_Toc499877443)

[Figure 5. Joint Velocity for ARM 1 7](#_Toc499877444)

[Figure 6. Joint Velocity for ARM 2 8](#_Toc499877445)

[Figure 7. Joint Velocity for ARM 3 8](#_Toc499877446)

[Figure 8. Joint Acceleration for ARM 1 9](#_Toc499877447)

[Figure 9. Joint Acceleration for ARM 2 9](#_Toc499877448)

[Figure 10. Joint Acceleration for ARM 3 10](#_Toc499877449)

[Figure 11.End-Effector Motion 10](#_Toc499877450)

# **Abstract/Executive Summary**

The three-DoFs parallel delta manipulator is designed to complete repetitive pick-and-place packaging process for ping pong balls. After designing a manipulator with appropriate dimensions for our application and specifying trajectory points of the end-effector through path generation method in stage one of the project, the trajectory of the end-effector has been generated and plotted in our analysis using via-point matching-velocity-and-acceleration method. 3D motion of the manipulator is also emulated in MATLAB for visual aid.The Jacobians (*Jx*and *Jq*) of the Delta Manipulator and its singularity configurations are determined as well in order to solve both forward velocity and forward static problems of the manipulator. Furthermore, the aforementioned techniques and methods in each section serve as a continuation study of parallel delta manipulators commonly used in industry today.

# Trajectory Generation:

## Brief Introduction & Explanation:

As continued from project 1, we have determined the following trajectory:



Figure 1.The manipulator's Trajectory from project I (Minh Nguyen, 2017)[1]

To make an animation out of this trajectory, we first need to solve the inverse kinematics of each point. The output will be stored in a 9x20 matrix called *joints.* The other important step that need to be done is defining the time it takes to travel from one point to another. For this project, we can simply set all of the travelling time between two adjacent points to 0.5 sec.

After doing these two tasks, we can take advantage of the function **“via\_points\_match\_VA.m”** to “smooth out” the pathway between 2 adjacent points. To roughly explain how this function works, it essentially takes the *joints* matrix and define more set of joint angles in between any two adjacent set of joint angles in an attempt to sketch out the motion of the three arms. Furthermore, it assumes several requirements for the velocity and acceleration at each points using cubic scheme, e.g the joints start at rest, the final velocity at one point equals the initial velocity of the next point, etc.The final animation video link is included in the appendix.

## Result Plot:

Below are the plots of positional displacements, velocities and acceleration of each arm. Note that each arm consists of 3 joints, thus the 3 graphs in each figure.

### Joint Displacement:

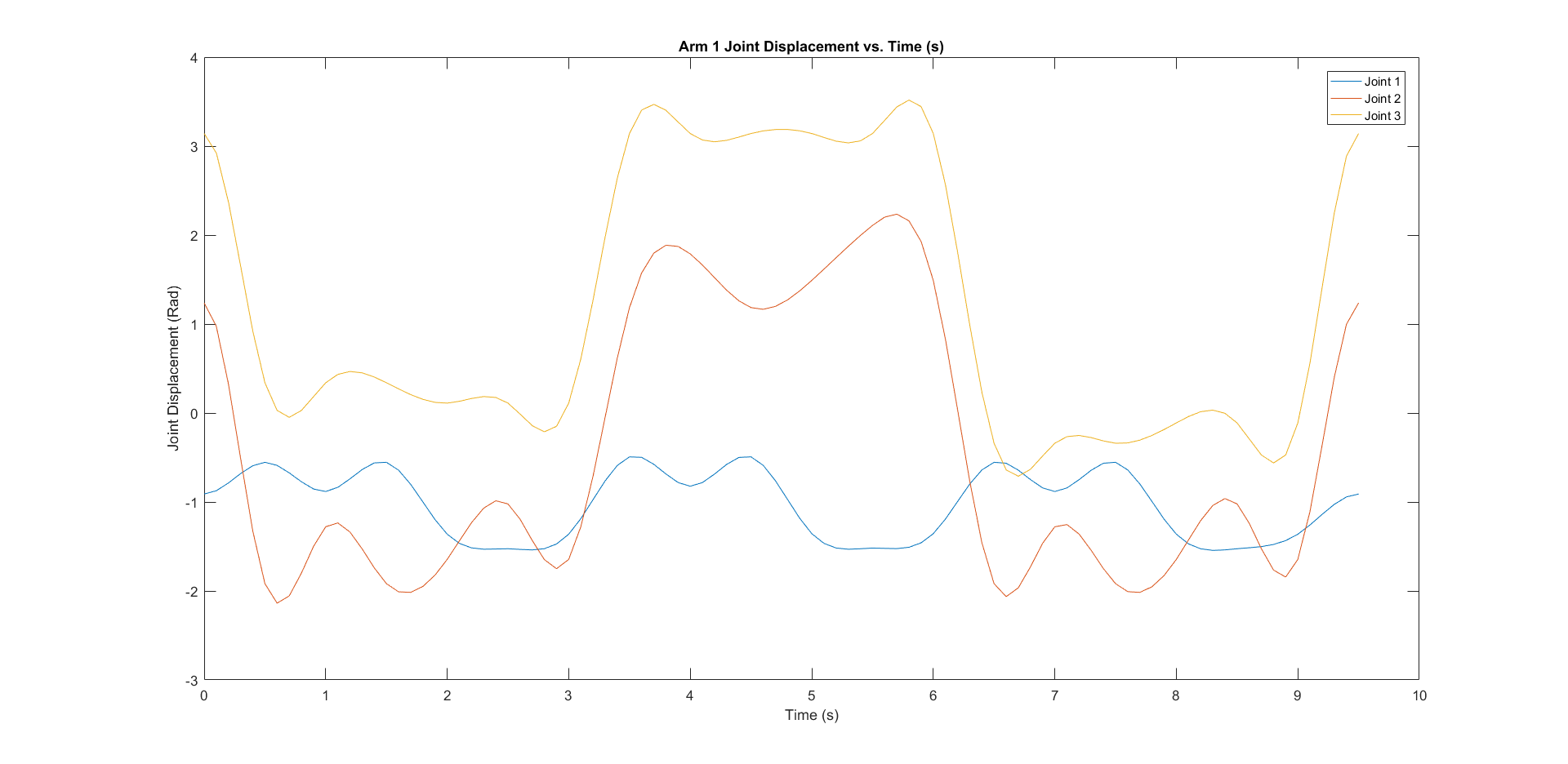


Figure 2. Joint Displacement for ARM 1

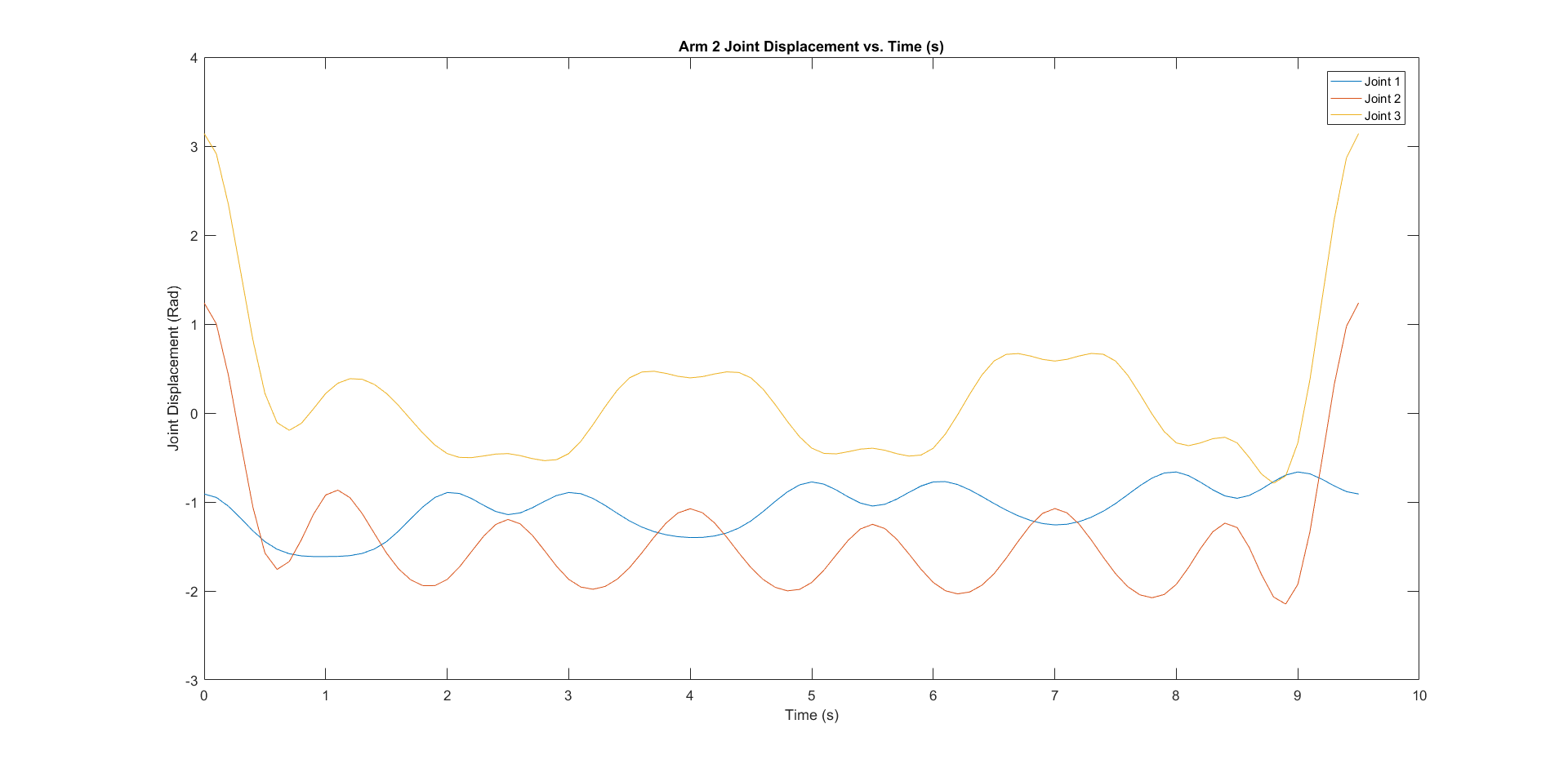


Figure 3. Joint Displacement for ARM 2

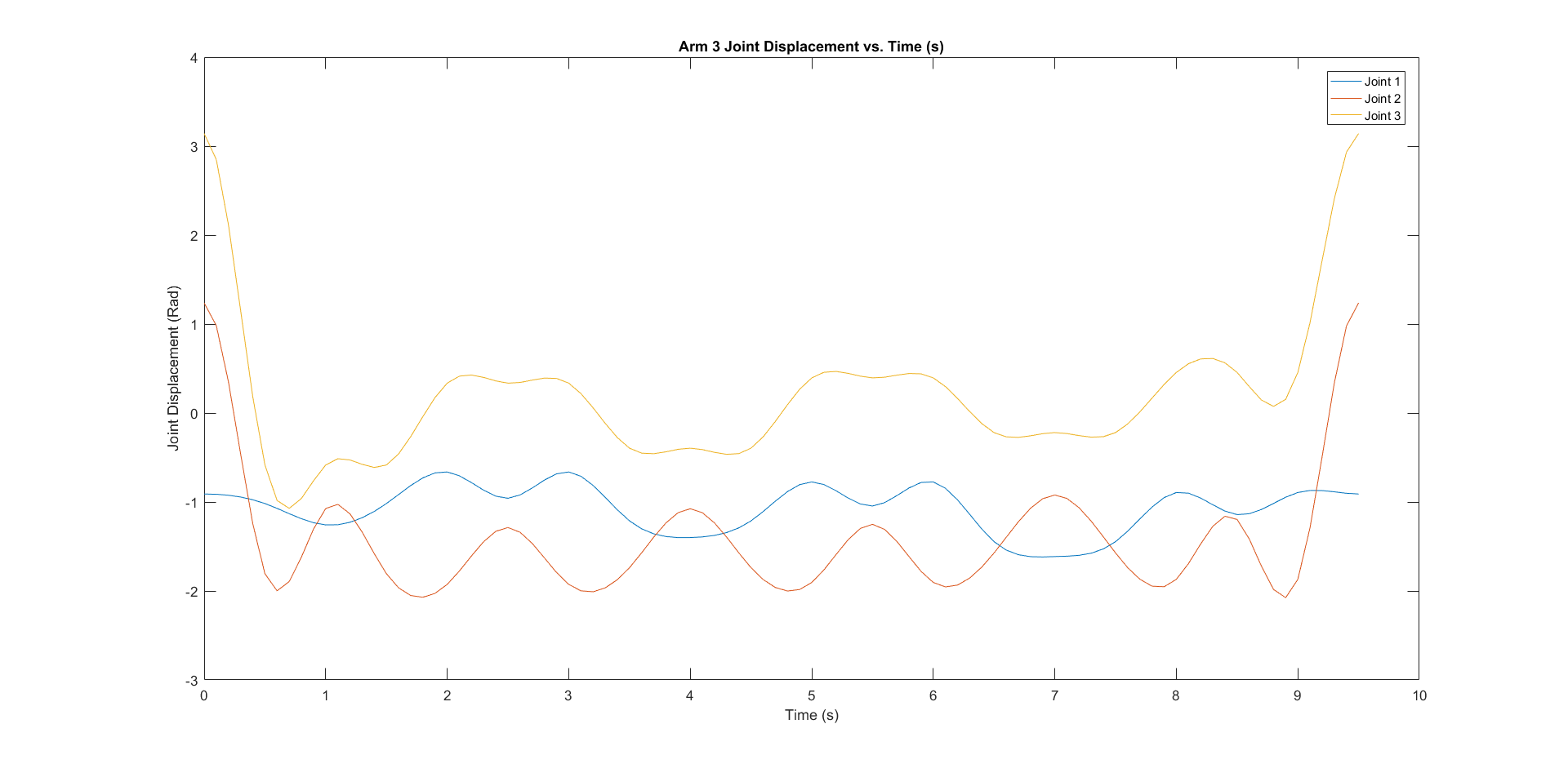


Figure 4.Joint Displacement for ARM 3

### Joint Velocity:

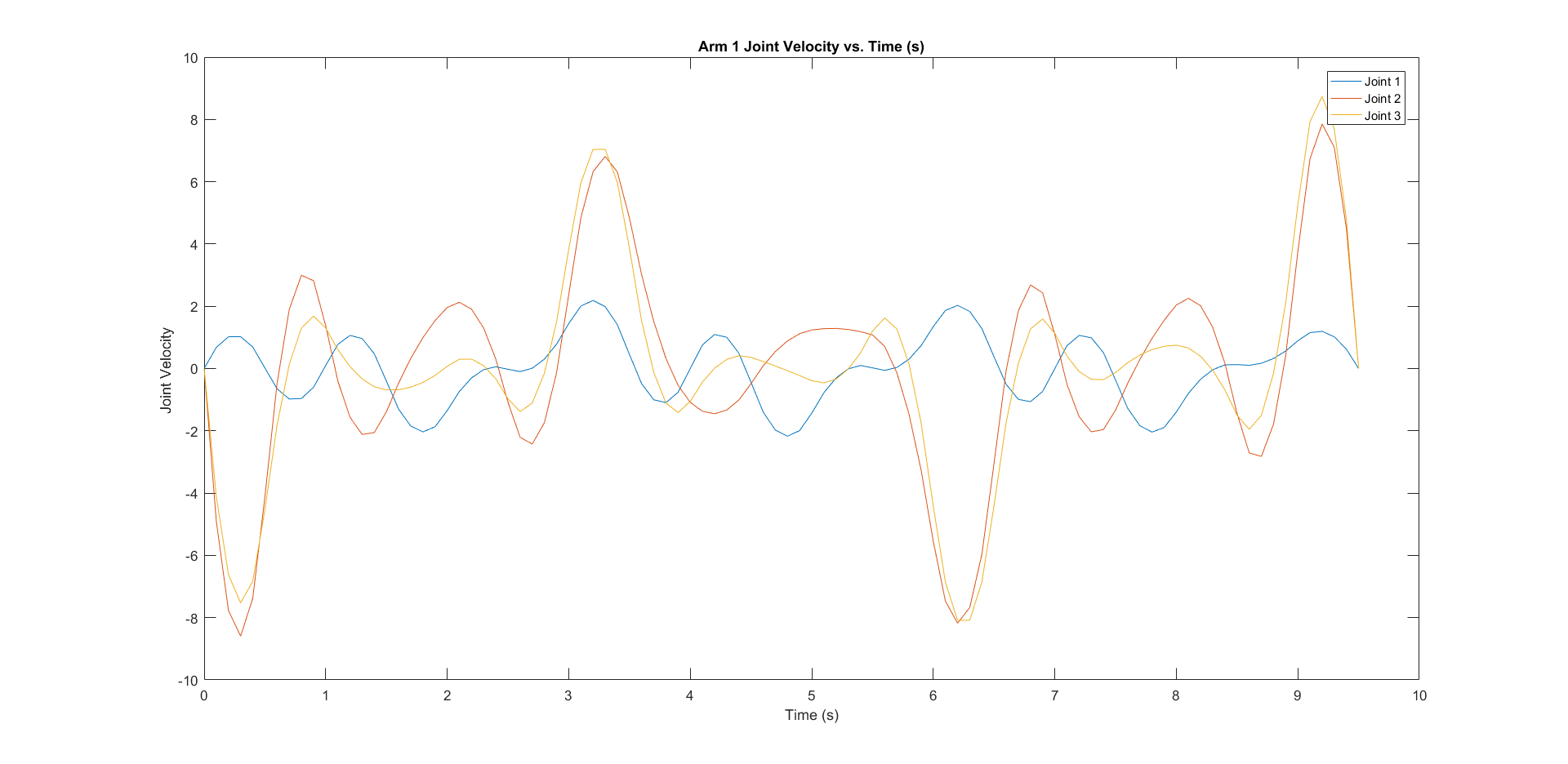


Figure 5. Joint Velocity for ARM 1

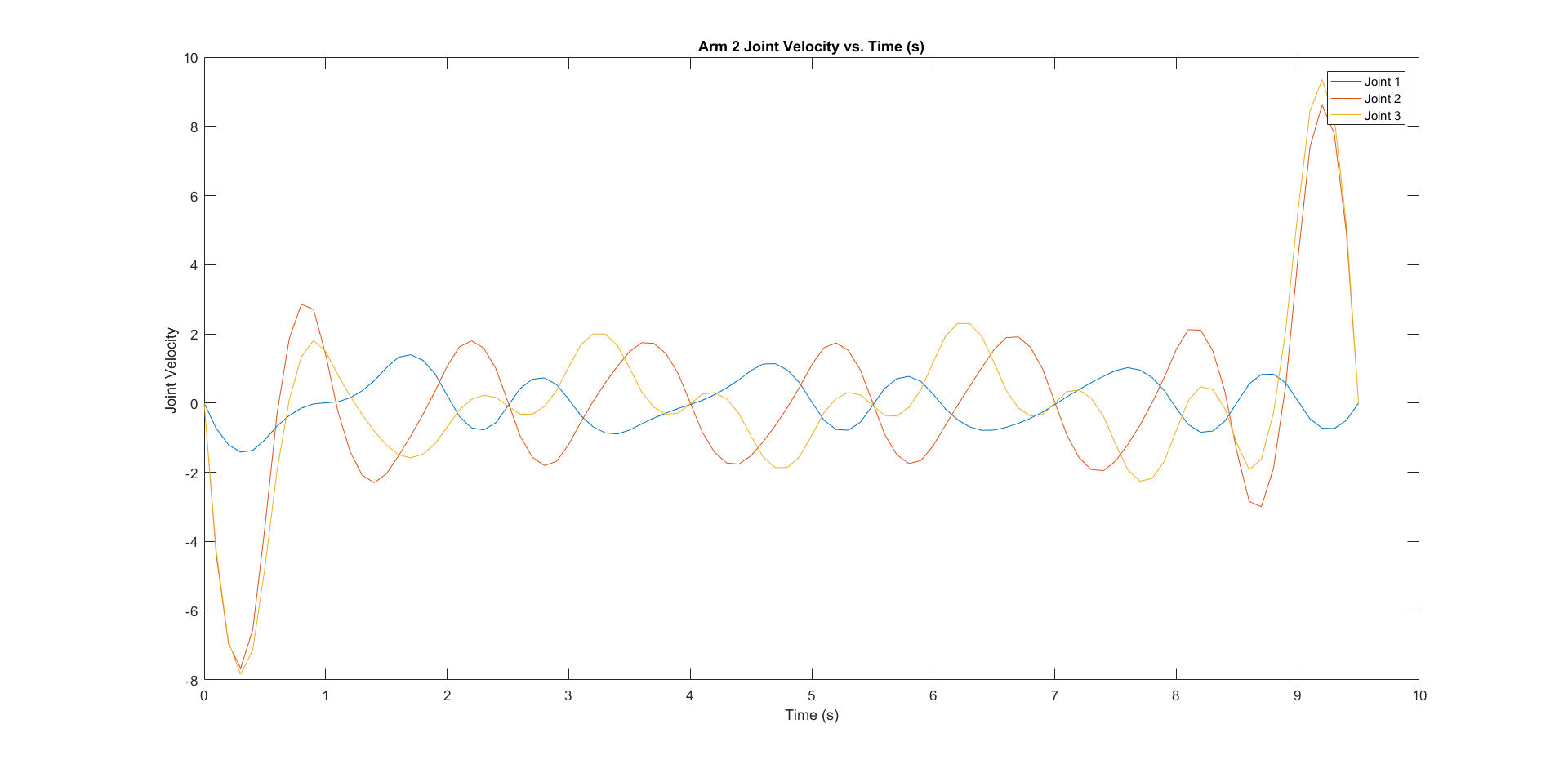


Figure 6. Joint Velocity for ARM 2

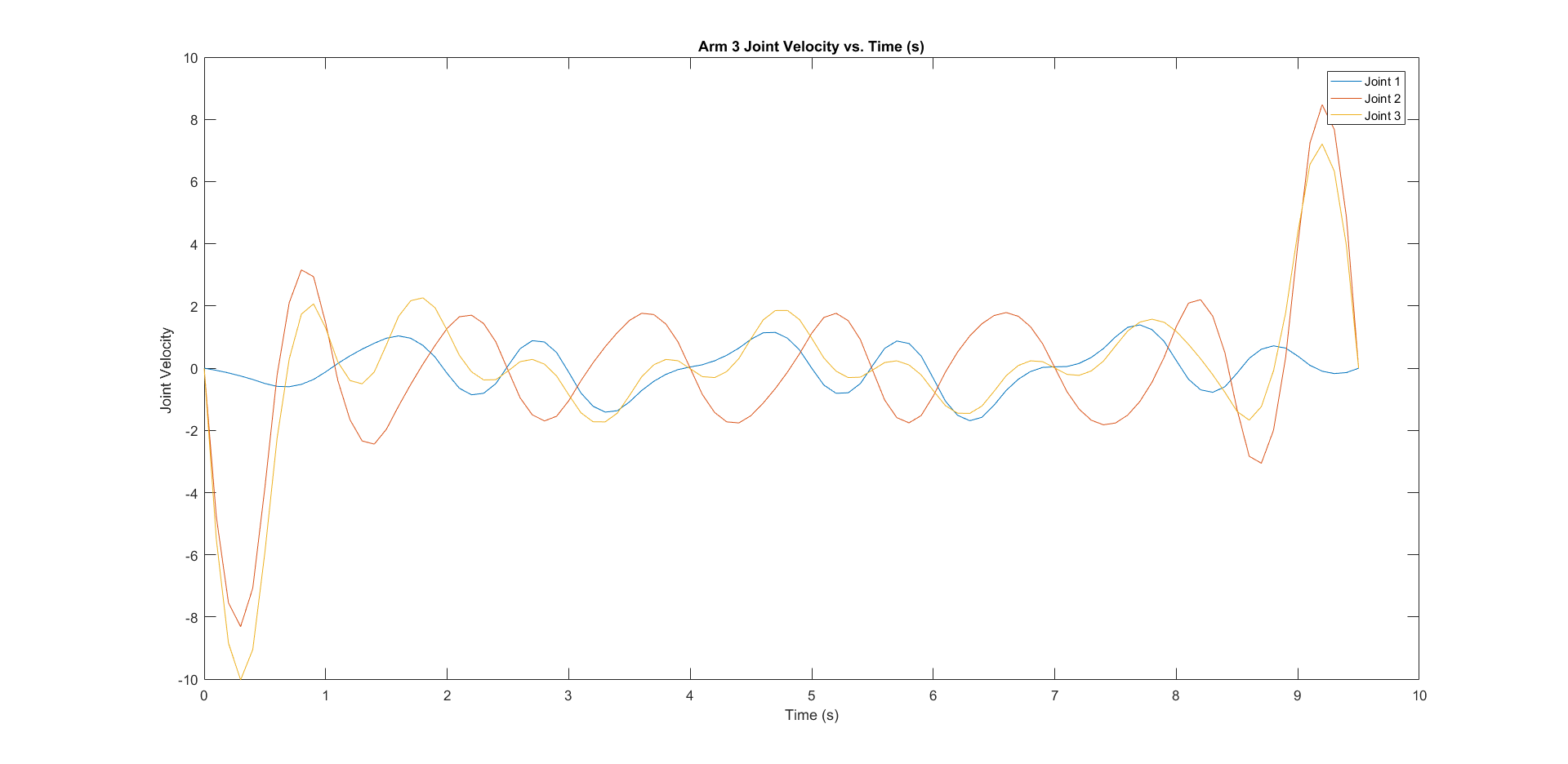


Figure 7. Joint Velocity for ARM 3

### Joint Acceleration:

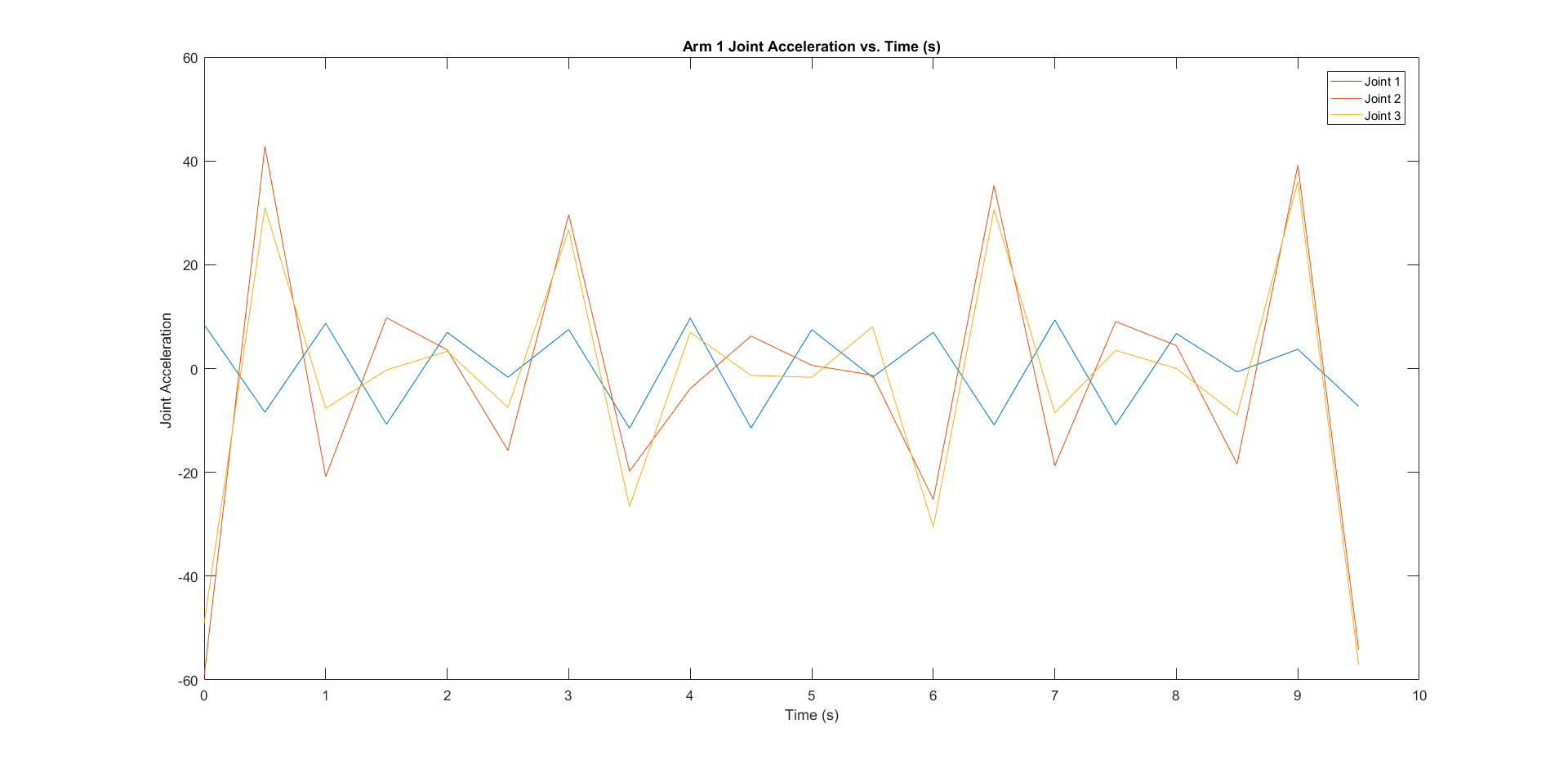


Figure 8. Joint Acceleration for ARM 1

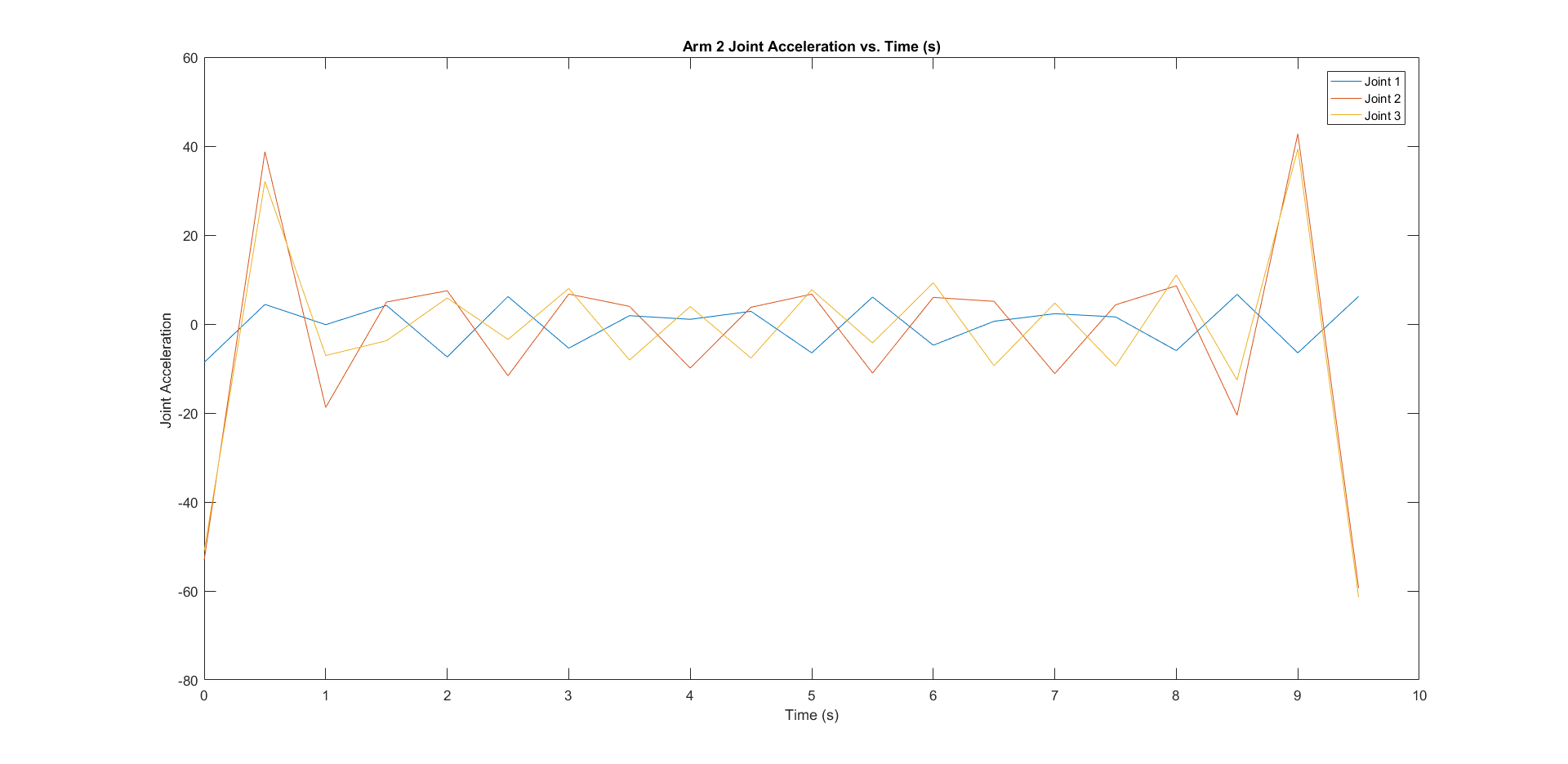


Figure 9. Joint Acceleration for ARM 2

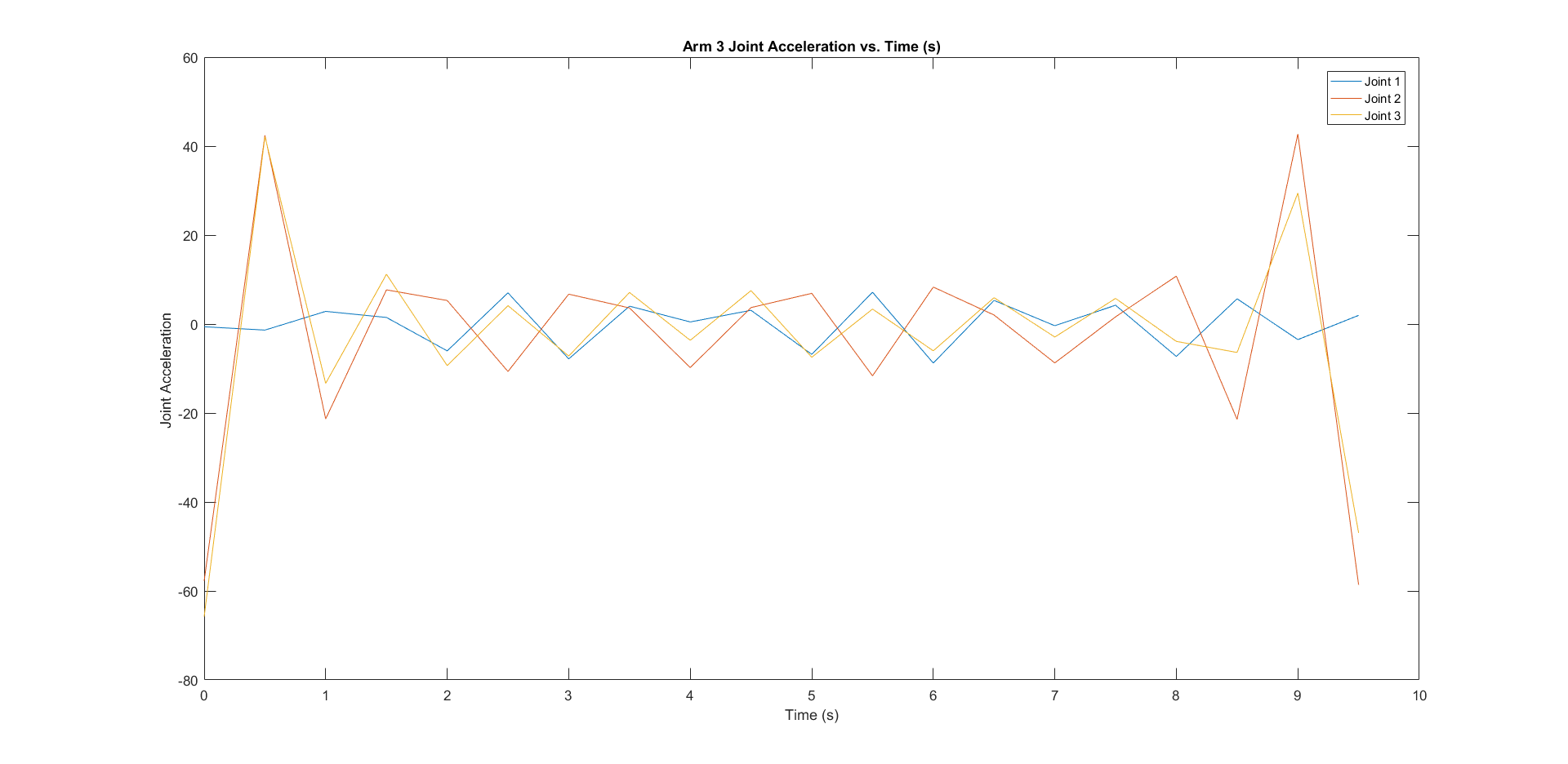


Figure 10. Joint Acceleration for ARM 3

### End-Effector Motion in 3D:

End-effector tracing in 3D using *comet3(x,y,z) function.*

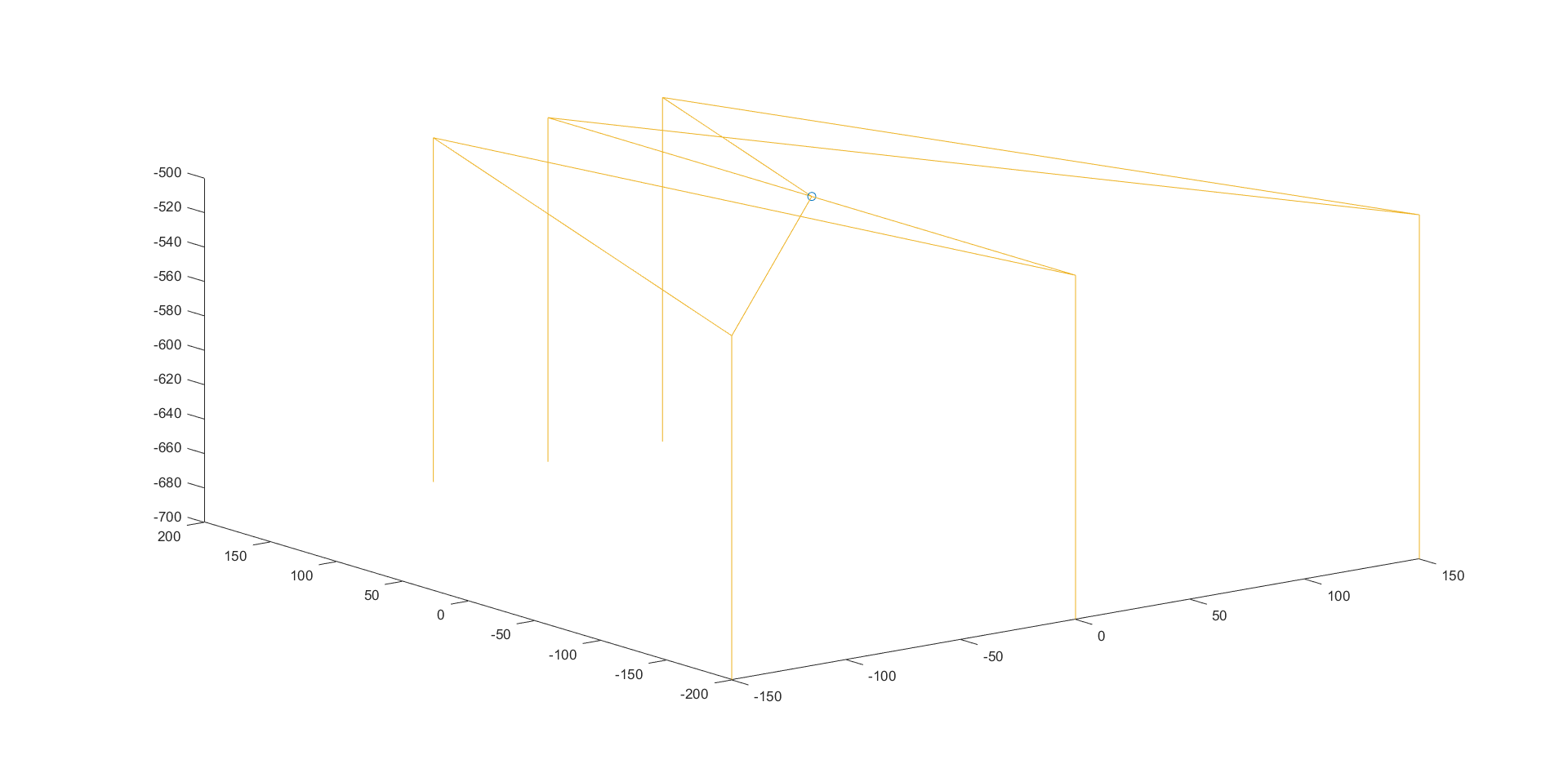


Figure 11.End-Effector Motion

# Jacobians:

## Jx and Jq:

The input of the problem is the 3 actuated joints providing 3 input angles (one per joint):

The coordinates of the moving platform, we simply denote them as:

Following the forward kinematic formulas we determined in project 1:

* Coordinates of points J1, J2, J3:

|  |  |
| --- | --- |
|  | (1) |

* Vectors

|  |  |
| --- | --- |
|  | (2) |

* From these, we can extrapolate point J1’, J2’, J3’:

|  |  |
| --- | --- |
|  | (3) |

We can now write the equations, note again that P, PP, L, LL are constants.

|  |  |
| --- | --- |
|  | (4) |
|  | (5) |
|  | (6) |

* From these 3 equations, we can calculate Jx and Jq matrices:

|  |  |
| --- | --- |
|  | (7) |

|  |  |
| --- | --- |
|  | (8) |

* Thus, we have the equation of:

With

## Singularities

### Singularity type I:

This is equivalent to:

With

Thus,

|  |  |
| --- | --- |
|  | (9) |

(2 solutions for each angles)

### Singularity type II:

If we denote the coordinate of J1’, J2’, J3’ in their most simplified form from equations (3):

Then the determinant of matrix Jx becomes:

Thus,

|  |  |
| --- | --- |
|  | (10) |

### Singularity type III

This happens when (9) and (10) happen at the same time.

## Forward Velocity Problem:

By setting

|  |  |
| --- | --- |
|  | (11) |

We can write the forward velocity formula:

|  |  |
| --- | --- |
|  | (12) |

To calculate matrix J, we have:

|  |  |
| --- | --- |
|  | (13) |

Multiply this with Jx, we have:

|  |  |
| --- | --- |
|  | (14) |

Then, we can calculate the inverse of matrix J.

## Forward Static Problem:

This is a less complicated problem for that we only have to calculate the transpose of matrix J. Indeed, from (14):

|  |  |
| --- | --- |
|  | (15) |

And the forward static formula can be written as follow:

|  |  |
| --- | --- |
|  | (16) |

With F being external forces acting on the end effector and being torques produced by them.

# Conclusion:

Delta Manipulator has been used in lot of industrial applications such as products packaging, CNC machine and even SMT machine. Have a chance to study how its work and making animation for the desired application is fun, and the knowledge and thought process obtained will benefit to the us in Robotic and Manufacturing career.

In the first part of the project we first defined the application which is suitable to the delta manipulator, and then we define the way point that the manipulator required to follow. With the help of CAD software and MATLAB we visualize the manipulator configuration in virtual space. Then in the second part of the project, we applied prescribed-velocity-via-point method between the waypoints that the end-effector has to pass through, and then taking snapshots in every time step jointed altogether to create a complete animation to our application. Furthermore, we also analyzed the Jacobian of the Delta Manipulator and solved for the singularity configurations. These singularies conditions were very challenging, especially for singularity type II where we have to solve for det(Jx)=0. Also later on, we have to calculate J-1 which is -1. This is far from trivial so we stop the calculation at J.

For the next step, this project can be further developed through choosing suitable materials for the robot structure, complete mechanical design, prototyping, test and adjustment to an usable product for packaging application.

# Appendix A

**Link to the animation**: https://imgur.com/a/iH9ed

# Appendix B

## Main Script

%1.1. Define size of figure and create figure handle (DO NOT MODIFY)

set(0,'Units','pixels');

dim = get(0,'ScreenSize');

fig\_handle = figure('doublebuffer','on','Position',[0,35,dim(3),dim(4)-100],...

'Name','3D Object','NumberTitle','off');

set(gcf,'color', [1 1 1]) %Background Colour

%1.2 Define the light in the figure (CHANGE POSITION VECTOR IF FIGUPP IS TOO BRIGHT/DARK)

set(fig\_handle,'renderer','zbuffer','doublebuffer','off')

light('color',[.5,.5,.5],'position',[0,1,3],'Style','infinite')

lighting gouraud

daspect([1 1 1]);

axis off

%Arrows (CHANGE PARAMETERS IF THEY ARE TOO SMALL OR TOO BIG)

% You need to have the file arrow3 in the same directory

arrow\_length=1; hold on

line([0,0],[0,0], [0,arrow\_length]); text(0,0,arrow\_length\*1.5,'z\_0','FontSize',14);

line([0,0],[0,arrow\_length],[0,0]); text(0,arrow\_length\*1.5, 0,'y\_0','FontSize',14);

line([0,arrow\_length],[0,0],[0,0]); text(arrow\_length\*1.5, 0, 0,'x\_0','FontSize',14);

axis ([-1000,1000,-1000,1000,-1000,1000]);

view(30,20)

% Convert figure into Object (LOAD YOUR PARTS)

load('Base\_Platform.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{1}=object;

load('Moving\_Platform.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{2}=object;

%%%%%%% ARM 1

load('Upper\_Link\_1.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{3}=object;

load('Lower\_Link\_1.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{4}=object;

load('Lower\_Link\_2.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{5}=object;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%% ARM 2

load('Upper\_Link\_2.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{6}=object;

load('Lower\_Link\_3.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{7}=object;

load('Lower\_Link\_4.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{8}=object;

%%%%%%% ARM 2

load('Upper\_Link\_3.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{9}=object;

load('Lower\_Link\_5.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{10}=object;

load('Lower\_Link\_6.mat');

setappdata(0,'object\_data',object);

object = getappdata(0,'object\_data');

obj{11}=object;

q=zeros(11);

for i=1:11%(CHANGE if you have 10 parts, change it to i=1:10)

q(i) = patch('faces', obj{i}.F, 'vertices', obj{i}.V);

set(q(i),'EdgeColor','none');

end

%Set colour to the componenets (CHANGE colours of new parts)

set(q(1),'FaceColor', [1,0.242,0.293]);

set(q(2),'FaceColor', [1,0.8,0.5]);

set(q(3),'FaceColor', [.3,0.5,0.3]);

set(q(4),'FaceColor', [.5,0.7,0.5]);

set(q(5),'FaceColor', [.5,0.7,0.5]);

set(q(6),'FaceColor', [.3,0.5,0.3]);

set(q(7),'FaceColor', [.5,0.7,0.5]);

set(q(8),'FaceColor', [.5,0.7,0.5]);

set(q(9),'FaceColor', [.3,0.5,0.3]);

set(q(10),'FaceColor', [.5,0.7,0.5]);

set(q(11),'FaceColor', [.5,0.7,0.5]);

%NEW

%ANIMATION

RGB=256; %Resolution

fm = getframe;

[img,map] = rgb2ind(fm.cdata,RGB,'nodither');

load('joints');

load('position');

%KINEMATICS

%Characteristics of the manipulator

% Definition of base platform

L=0.25\*1000; % base length (centPP to joint)

LL=0.1\*1000; % mobile length (centPP to joint)

P=0.45\*1000; % Link first length

PP =0.45\*1000; % Length second Length

syms x y z

%%%%%%%%%%%%%%%%%%%%

%INVERSE KINEMATICS%

%%%%%%%%%%%%%%%%%%%%

%%x0=0.05\*1000; y0=0.2\*1000; z0=-0.5\*1000; %Position of Mobile Platform

P\_ee=[0 -150 -150 -150 -50 -50 -50 0 0 0 0 0 0 150 150 150 50 50 50 0;

0 -200 -200 -200 200 200 200 -200 -200 -200 200 200 200 -200 -200 -200 200 200 200 0;

-500 -500 -700 -500 -500 -700 -500 -500 -700 -500 -500 -700 -500 -500 -700 -500 -500 -700 -500 -500];

tf=ones(1,length(P\_ee(1,:))-1)\*0.5;

% joints=zeros(9,length(P\_ee(1,:)));

% for i=1:length(P\_ee(1,:))

%

% joints(1:9,i)=inverse\_kin(P\_ee(1,i),P\_ee(2,i),P\_ee(3,i));

%

% end

% TRAJECTORY GENERATION. Trajectory generation of all the joints

dt=0.1; %stepsize

[position,velocity,acceleration,time]=via\_points\_match\_VA(joints, tf, dt, 'prescribed',[0,0]);

%ANIMATION (DO NOT CHANGE)

n=length(position(1,:));

mov(1:length(n)) = struct('cdata', [],'colormap', []);

[a,b]=size(img); gifim=zeros(a,b,1,n-1,'uint8');

% FORWARD KINEMATICS

% NEW LOOP

for k=1:n %Make sure you don’t use k

theta11=position(1,k);

theta12=position(2,k);

theta13=position(3,k);

theta21=position(4,k);

theta22=position(5,k);

theta23=position(6,k);

theta31=position(7,k);

theta32=position(8,k);

theta33=position(9,k);

[XYZ]=forward\_kin(theta11,theta12,theta13);

offset=50;

R1Z=[0 1 0; -1 0 0;0 0 1]; %rotate -90 degree

R1Y=[0 0 -1;0 1 0;1 0 0]; % rotate -90

R1Z\_2=[cosd(120) -sind(120) 0; sind(120) cosd(120) 0; 0 0 1]; % Rotate 120 degree in Z

J1=[0,-L-P\*cos(abs(theta11)),-P\*sin(abs(theta11))]; %Point J1, J2, J3

J2=[(L+P\*cos(abs(theta12)))\*cos(pi/6),(L+P\*cos(abs(theta12)))\*sin(pi/6),-P\*sin(abs(theta12))];

J3=[-(L+P\*cos(abs(theta13)))\*cos(pi/6),(L+P\*cos(abs(theta13)))\*sin(pi/6),-P\*sin(abs(theta13))];

%%%%% ARM 1 2 3 T matrices

T1\_1=[cos(theta11) -sin(theta11) 0 0; sin(theta11) cos(theta11) 0 0; 0 0 1 0; 0 0 0 1];

T2\_1=[cos(theta21) -sin(theta21) 0 P; sin(theta21) cos(theta21) 0 0; 0 0 1 0; 0 0 0 1];

T3\_1=[cos(theta31) sin(theta31) 0 0; 0 0 -1 0; sin(theta31) cos(theta31) 0 0; 0 0 0 1];

T1\_2=[cos(theta12) -sin(theta12) 0 0; sin(theta12) cos(theta12) 0 0; 0 0 1 0; 0 0 0 1];

T2\_2=[cos(theta22) -sin(theta22) 0 P; sin(theta22) cos(theta22) 0 0; 0 0 1 0; 0 0 0 1];

T3\_2=[cos(theta32) sin(theta32) 0 0; 0 0 -1 0; sin(theta32) cos(theta32) 0 0; 0 0 0 1];

T1\_3=[cos(theta13) -sin(theta13) 0 0; sin(theta13) cos(theta13) 0 0; 0 0 1 0; 0 0 0 1];

T2\_3=[cos(theta23) -sin(theta23) 0 P; sin(theta23) cos(theta23) 0 0; 0 0 1 0; 0 0 0 1];

T3\_3=[cos(theta33) sin(theta33) 0 0; 0 0 -1 0; sin(theta33) cos(theta33) 0 0; 0 0 0 1];

for i=1:11 %(CHANGE n=2 for the number of parts that you have)

V{i} = obj{i}.V';

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Position of End Effector

newV{1} = V{1} + repmat([0,0,0]',[1 length(V{1}(1,:))]); %The position of the base

newV{2} = V{2} + repmat([XYZ(1),XYZ(2),XYZ(3)]',[1 length(V{2}(1,:))]); %The position of the mobile platform

newV{3}= R1Y\*R1Z\*T1\_1(1:3,1:3)\*V{3}; % ARM 1 Upper LINK

newV{3}=newV{3}+ repmat([0;-L;0],[1 length(newV{3}(1,:))]);

newV{4}= (R1Y\*R1Z\*T1\_1(1:3,1:3))\*T2\_1(1:3,1:3)\*T3\_1(1:3,1:3)\*V{4}; % ARM 1 Lower LINK

newV{4}=newV{4}+ repmat([J1(1)+offset;J1(2);J1(3)],[1 length(newV{4}(1,:))]);

newV{5}= (R1Y\*R1Z\*T1\_1(1:3,1:3))\*T2\_1(1:3,1:3)\*T3\_1(1:3,1:3)\*V{5}; % ARM 1 Lower LINK

newV{5}=newV{5}+ repmat([J1(1)-offset;J1(2);J1(3)],[1 length(newV{5}(1,:))]);

newV{6}=R1Z\_2\*R1Y\*R1Z\*T1\_2(1:3,1:3)\*V{6}; % ARM 2 Upper LINK

newV{6}=newV{6}+ repmat([L\*cosd(30);L\*sind(30);0],[1 length(newV{6}(1,:))]);

newV{7}= R1Z\_2\*(R1Y\*R1Z\*T1\_2(1:3,1:3))\*T2\_2(1:3,1:3)\*T3\_2(1:3,1:3)\*V{7}; % ARM 2 Lower LINK

newV{7}=newV{7}+ repmat([J2(1)+offset\*cosd(60);J2(2)-offset\*sind(60);J2(3)],[1 length(newV{7}(1,:))]);

newV{8}= R1Z\_2\*(R1Y\*R1Z\*T1\_2(1:3,1:3))\*T2\_2(1:3,1:3)\*T3\_2(1:3,1:3)\*V{8}; % ARM 2 Lower LINK

newV{8}=newV{8}+ repmat([J2(1)-offset\*cosd(60);J2(2)+offset\*sind(60);J2(3)],[1 length(newV{8}(1,:))]);

newV{9}=R1Z\_2\*R1Z\_2\*R1Y\*R1Z\*T1\_3(1:3,1:3)\*V{9}; % ARM 3 Upper LINK

newV{9}=newV{9}+ repmat([-L\*cosd(30);L\*sind(30);0],[1 length(newV{9}(1,:))]);

newV{10}= R1Z\_2\*R1Z\_2\*(R1Y\*R1Z\*T1\_3(1:3,1:3))\*T2\_3(1:3,1:3)\*T3\_3(1:3,1:3)\*V{10}; % ARM 3 Lower LINK

newV{10}=newV{10}+ repmat([J3(1)-offset\*cosd(60);J3(2)-offset\*sind(60);J3(3)],[1 length(newV{10}(1,:))]);

newV{11}= R1Z\_2\*R1Z\_2\*(R1Y\*R1Z\*T1\_3(1:3,1:3))\*T2\_3(1:3,1:3)\*T3\_3(1:3,1:3)\*V{11}; % ARM 3 Lower LINK

newV{11}=newV{11}+ repmat([J3(1)+offset\*cosd(60);J3(2)+offset\*sind(60);J3(3)],[1 length(newV{11}(1,:))]);

for ii=1:11 %(CHANGE n=2 to the number of parts that you have)

set(q(ii),'Vertices',newV{ii}(1:3,:)'); %Set the new position in the handle (graphical link)

end

drawnow

im= frame2im(getframe);

gifim(:,:,:,k) = rgb2ind(im, map);

mov(k)=getframe(gcf);

end

% Close loop k

%ANIMATION, creates animated gif (DO NOT MODIFY)

imwrite(gifim,map,'Project.gif','DelayTime',0)%,'LoopCount',inf)

## Inverse Kinematics

function [J]=inverse\_kin(x0,y0,z0)

%KINEMATICS

%Characteristics of the manipulator

L=0.25\*1000; % base length (centPP to joint)

LL=0.1\*1000; % mobile length (centPP to joint)

P=0.45\*1000; % Link first length

PP =0.45\*1000; % Length second Length

syms x y z

%%%%%%%%%%%%%% SOLVE FOR THETA 1 ANGLES %%%%%%%%%%%%%%%%%%%%

%%%%Branch 1

xyz\_o1=[x0,y0,z0]; %%%% LOCATION OF THE MOVING PLATFORM: INPUT HEPP

E1\_prime=[0,xyz\_o1(2)-LL,xyz\_o1(3)]; % Point E1'

F1=[0,-L,0]; %Point F1

%Solving for the intersection of 2 circles

[soly,solz]=solve((y-F1(2))^2+(z-F1(3))^2-P^2==0,(y-E1\_prime(2))^2+(z-E1\_prime(3))^2-PP^2+xyz\_o1(1)^2==0);

%Choose the smaller set of solution

index=find(soly == min(soly(:)));

y1=soly(index);

z1=solz(index);

theta11=double(atan2(z1,F1(2)-y1)); %Solve for theta11

%%%%Branch 2

xyz\_o2=[xyz\_o1(1)\*cos(2\*pi/3)+xyz\_o1(2)\*sin(2\*pi/3),-xyz\_o1(1)\*sin(2\*pi/3)+xyz\_o1(2)\*cos(2\*pi/3),xyz\_o1(3)];

E2\_prime=[0,xyz\_o2(2)-LL,xyz\_o2(3)]; % Point E2'

F2=[0,-L,0]; % Point F2

%Solve for intersection of 2 circles

[soly,solz]=solve((y-F2(2))^2+(z-F2(3))^2-P^2==0,(y-E2\_prime(2))^2+(z-E2\_prime(3))^2-PP^2+xyz\_o2(1)^2==0);

%Choose the smaller set of solution

index=find(soly == min(soly(:)));

y2=soly(index);

z2=solz(index);

theta12=double(atan2(z2,F2(2)-y2)); %Solve for theta12

%%%%Branch 3

xyz\_o3=[xyz\_o1(1)\*cos(-2\*pi/3)+xyz\_o1(2)\*sin(-2\*pi/3),-xyz\_o1(1)\*sin(-2\*pi/3)+xyz\_o1(2)\*cos(-2\*pi/3),xyz\_o1(3)];

E3\_prime=[0,xyz\_o3(2)-LL,xyz\_o3(3)];% Point E3'

F3=[0,-L,0]; % Point F3

%Solve for intersection of 2 circles

[soly,solz]=solve((y-F3(2))^2+(z-F3(3))^2-P^2==0,(y-E3\_prime(2))^2+(z-E3\_prime(3))^2-PP^2+xyz\_o3(1)^2==0);

%Choose the smaller set of solution

index=find(soly == min(soly(:)));

y3=soly(index);

z3=solz(index);

theta13=double(atan2(z3,F3(2)-y3)); %Solve for theta13

R=[0 -1 0; 0 0 1; -1 0 0]; %Rotation Matrix

syms theta1 theta2 theta3

%% Branch 1

OE=[(xyz\_o1(1)-0);(xyz\_o1(2)-LL--L);(xyz\_o1(3)-0)];

OE\_prime=R\*OE;

[q1,w1,e1]=solve((P\*cos(theta1) + PP\*cos(theta3)\*(cos(theta1)\*cos(theta2) - sin(theta1)\*sin(theta2)))== OE\_prime(1),...

P\*sin(theta1) + PP\*cos(theta3)\*(cos(theta1)\*sin(theta2) + cos(theta2)\*sin(theta1))==OE\_prime(2),...

(PP\*sin(theta3))==OE\_prime(3));

q1=double(q1);

w1=double(w1);

e1=double(e1);

for i=1:1:4

if (abs(q1(i)-theta11)<10^-3)

theta21=w1(i);

theta31=e1(i);

break

end

end

%% Branch 2

OE2=[xyz\_o1(1)\*cos(2\*pi/3)+xyz\_o1(2)\*sin(2\*pi/3);-xyz\_o1(1)\*sin(2\*pi/3)+xyz\_o1(2)\*cos(2\*pi/3)-LL--L;OE(3)];

OE2\_prime=R\*OE2;

[q2,w2,e2]=solve((P\*cos(theta1) + PP\*cos(theta3)\*(cos(theta1)\*cos(theta2) - sin(theta1)\*sin(theta2)))== OE2\_prime(1),...

P\*sin(theta1) + PP\*cos(theta3)\*(cos(theta1)\*sin(theta2) + cos(theta2)\*sin(theta1))==OE2\_prime(2),...

(PP\*sin(theta3))==OE2\_prime(3));

q2=double(q2);

w2=double(w2);

e2=double(e2);

for i=1:1:4

if (abs(q2(i)-theta12)<10^-3)

theta22=w2(i);

theta32=e2(i);

break

end

end

% Branch 3

OE3=[xyz\_o1(1)\*cos(-2\*pi/3)+xyz\_o1(2)\*sin(-2\*pi/3);-xyz\_o1(1)\*sin(-2\*pi/3)+xyz\_o1(2)\*cos(-2\*pi/3)-LL--L;OE(3)];

OE3\_prime=R\*OE3;

[q3,w3,e3]=solve((P\*cos(theta1) + PP\*cos(theta3)\*(cos(theta1)\*cos(theta2) - sin(theta1)\*sin(theta2)))== OE3\_prime(1),...

P\*sin(theta1) + PP\*cos(theta3)\*(cos(theta1)\*sin(theta2) + cos(theta2)\*sin(theta1))==OE3\_prime(2),...

(PP\*sin(theta3))==OE3\_prime(3));

q3=double(q3);

w3=double(w3);

e3=double(e3);

for i=1:1:4

if (abs(q3(i)-theta13)<10^-3)

theta23=w3(i);

theta33=e3(i);

break

end

end

J=[theta11;theta12;theta13;theta21;theta22;theta23;theta31;theta32;theta33];

end

## Forward Kinematics

function [XYZ] = forward\_kin(theta11,theta12,theta13)

%Characteristics of the manipulator

L=0.25\*1000; % base length (centPP to joint)

LL=0.1\*1000; % mobile length (centPP to joint)

P=0.45\*1000; % Link first length

PP =0.45\*1000; % Length second Length

% %FORWARD KINEMATICS%

%%%%%%%%%%%%%%%%%%%%

syms x y z

J1=[0,-L-P\*cos(abs(theta11)),-P\*sin(abs(theta11))]; %Point J1, J2, J3

J2=[(L+P\*cos(abs(theta12)))\*cos(pi/6),(L+P\*cos(abs(theta12)))\*sin(pi/6),-P\*sin(abs(theta12))];

J3=[-(L+P\*cos(abs(theta13)))\*cos(pi/6),(L+P\*cos(abs(theta13)))\*sin(pi/6),-P\*sin(abs(theta13))];

E1E0=[0,LL,0]; %Vector E1E0, E2E0, E3E0

E2E0=[-LL\*cos(pi/6),-LL\*sin(pi/6),0];

E3E0=[LL\*cos(pi/6),-LL\*sin(pi/6),0];

J1\_prime=E1E0+J1; %Point J1',J2',J3'

J2\_prime=E2E0+J2;

J3\_prime=E3E0+J3;

[solx,soly,solz]=solve(x^2+(y-J1\_prime(2))^2+(z-J1\_prime(3))^2==PP^2,...

(x-J2\_prime(1))^2+(y-J2\_prime(2))^2+(z-J2\_prime(3))^2==PP^2,...

(x-J3\_prime(1))^2+(y-J3\_prime(2))^2+(z-J3\_prime(3))^2==PP^2);

index=find(solz==min(solz(:)));

XYZ=[double(solx(index));double(soly(index));double(solz(index))];

end

## Fixing Position Matrix

% m=1;

% while (m< (n+4)/6)

% [X1]=forward\_kin(position(1,6\*m-4),position(2,6\*m-4),position(3,6\*m-4));

% [X2]=forward\_kin(position(1,6\*m-3),position(2,6\*m-3),position(3,6\*m-3));

% [X3]=forward\_kin(position(1,6\*m-2),position(2,6\*m-2),position(3,6\*m-2));

% [X4]=forward\_kin(position(1,6\*m-1),position(2,6\*m-1),position(3,6\*m-1));

% [angle1]=inverse\_kin(X1(1),X1(2),X1(3));

% [angle2]=inverse\_kin(X2(1),X2(2),X2(3));

% [angle3]=inverse\_kin(X3(1),X3(2),X3(3));

% [angle4]=inverse\_kin(X4(1),X4(2),X4(3));

% for i=4:9

% position(i,6\*m-4)=angle1(i);

% position(i,6\*m-3)=angle2(i);

% position(i,6\*m-2)=angle3(i);

% position(i,6\*m-1)=angle4(i);

% end

% m=m+1;

% end