

**PROJECT REPORT # 4**

***Assembly Line Spray Painter***

***6 Dof – Serial Manipulator***

Student Name: - Syed Imad Azeem Rizvi

Student Id: - 301297227

Report due date: 07th Dec’ 20

MSE – 429 Advanced Kinematics For Robotics Systems

SCHOOL OF ENGINEERING SCIENCE

SIMON FRASER UNIVERSITY

Table of Contents

[Abstract 4](#_Toc58324119)

[Introduction. 5](#_Toc58324120)

[Design Specification 5](#_Toc58324121)

[Inverse Kinematics 7](#_Toc58324122)

[Kinematic Reconstruction Path Generation 13](#_Toc58324123)

[Workspace Analysis 16](#_Toc58324124)

[Path Generation 16](#_Toc58324125)

[Trajectory Generation 17](#_Toc58324126)

[Jacobian 21](#_Toc58324127)

[Singularity analysis 22](#_Toc58324128)

[Link Modeling 23](#_Toc58324129)

[Base 23](#_Toc58324130)

[Prismatic Joint 24](#_Toc58324131)

[Revolute Joint 2 24](#_Toc58324132)

[Revolute Joint 3 24](#_Toc58324133)

[Revolute Joint 4 25](#_Toc58324134)

[Revolute Joint 5 25](#_Toc58324135)

[Revolute Joint 6 25](#_Toc58324136)

[End Effector 26](#_Toc58324137)

[Dynamics 26](#_Toc58324138)

[Simulation 28](#_Toc58324139)

[Conclusion 32](#_Toc58324140)

[References 33](#_Toc58324141)

[Appendix 33](#_Toc58324142)

[1. my\_path 33](#_Toc58324143)

[2. P\_xyz\_abg 35](#_Toc58324144)

[3. Newton-Euler Recursive formulation 36](#_Toc58324145)

[4. My\_jacobian\_symbolic 37](#_Toc58324146)

[5. Inertia\_tensor 40](#_Toc58324147)

Table of Figures

[Figure 1 SolidWorks Layout 5](file:///C:\Users\Imad\Desktop\MSE%20429\Projects\Project%20III\Project%203.docx#_Toc55893857)

[Figure 2 MATLAB Layout 5](#_Toc55893858)

Table of Tables

[Table 1 Joints Information 5](#_Toc55893862)

[Table 2 Denavit-Hartenberg Parameters 6](#_Toc55893863)

# Abstract

This report is the combination of all 4 projects. It constitutes design specification and joints/links constraints of serial robot manipulators, Inverse Kinematics calculations with focus on kinematic reconstruction of the model, as well as Dynamic analysis of the manipulator. Based on rigid body conventions of the manipulator a hand calculation is performed to compute the inverse kinematics of the model. The kinematic reconstruction is performed with respect to the global origin of the model. The inverse kinematics is solved with specified path coordinates. A trajectory is generated for the spatial manipulator. The dynamic analysis is also conducted to finalize the report. At the end, conclusions will address the future improvement and recommendations in manipulator design and its application.

# Introduction.

This report is the continuation of the project that requires a kinematic reconstruction and inverse kinematics calculation as well as determining the workspace of the manipulator with path generation.

The fig 1 and fig 2 shows the MATLAB layout and solid works design of the manipulator.

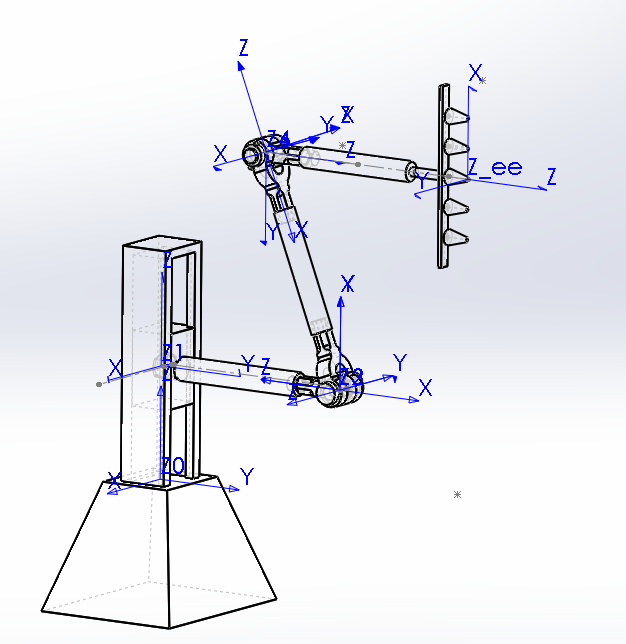
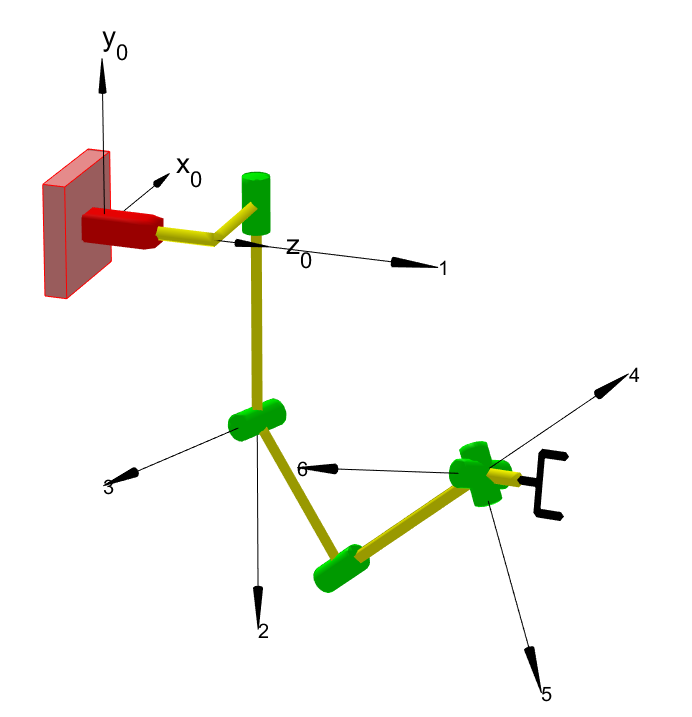


Figure 1 SolidWorks Layout

Figure 2 MATLAB Layout

# Design Specification

|  |  |  |
| --- | --- | --- |
| Joint | Type | Constraints |
| 1 | Prismatic |  |
| 2 | Revolute | 90270 |
| 3 | Revolute | 225 |
| 4 | Revolute | 0 |
| 5 | Revolute |  |
| 6 | Revolute |  |

Table 1 Joints Information

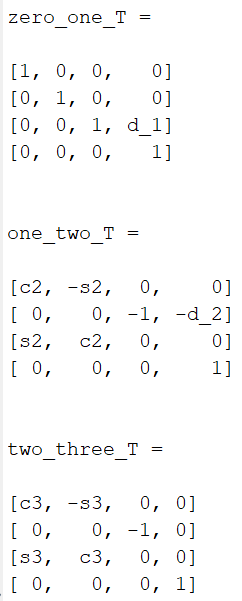
D-H Parameter

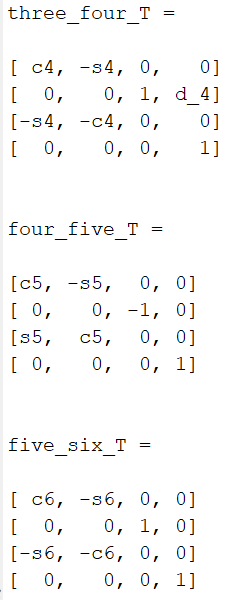
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 0 | 0 | 0 |  | 0 | 1 |
| 1 | +90 | 0 | 838.5mm |  | 2 |
| 2 | +90 | 0 | 0 |  | 3 |
| 3 | -90 | 0 | 972mm |  | 4 |
| 4 | -90 | 0 | 0 |  | 5 |
| 5 | +90 | 0 | 0 |  | 6 |
| 6 | 0 | 0 | 945 mm | 0 | ee |

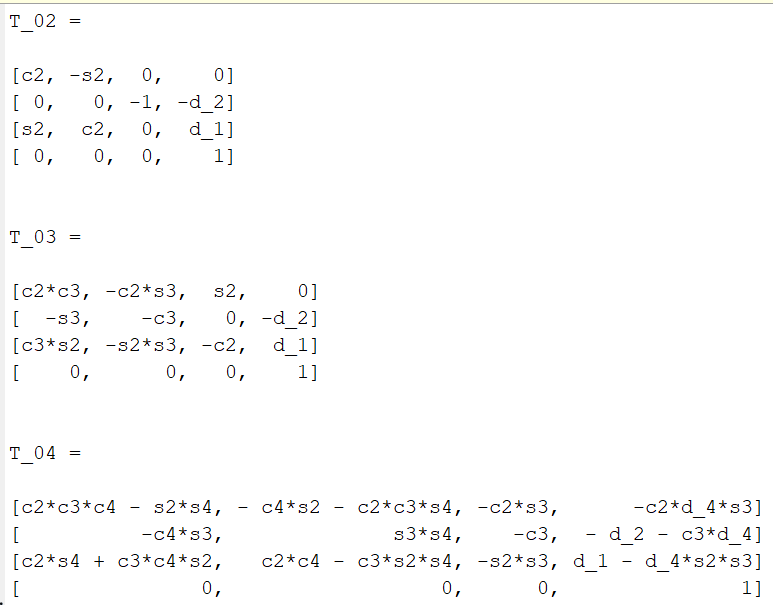
Table 2 Denavit-Hartenberg Parameters

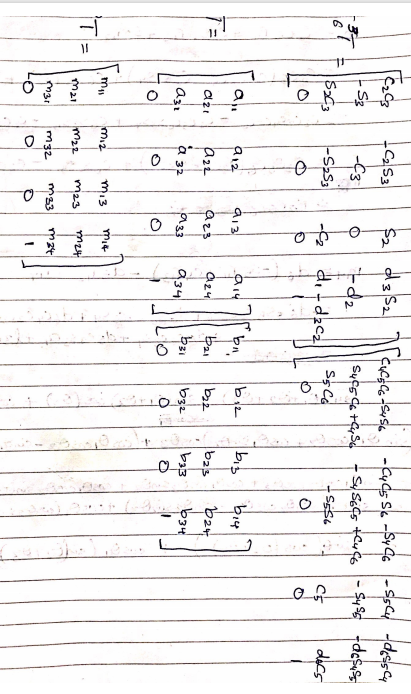
# Inverse Kinematics

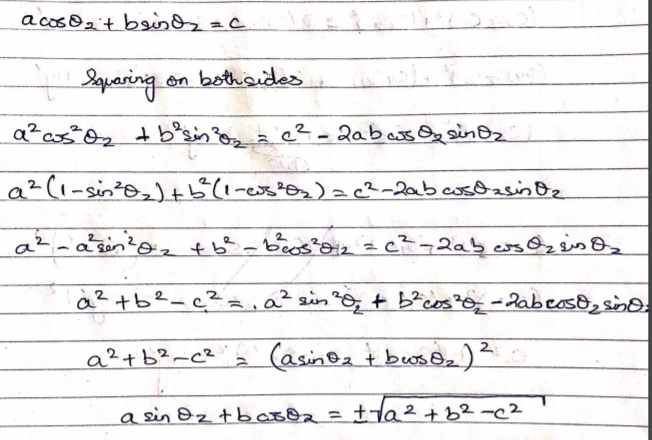
The following images show a detail hand calculation of the inverse kinematics for the manipulator.

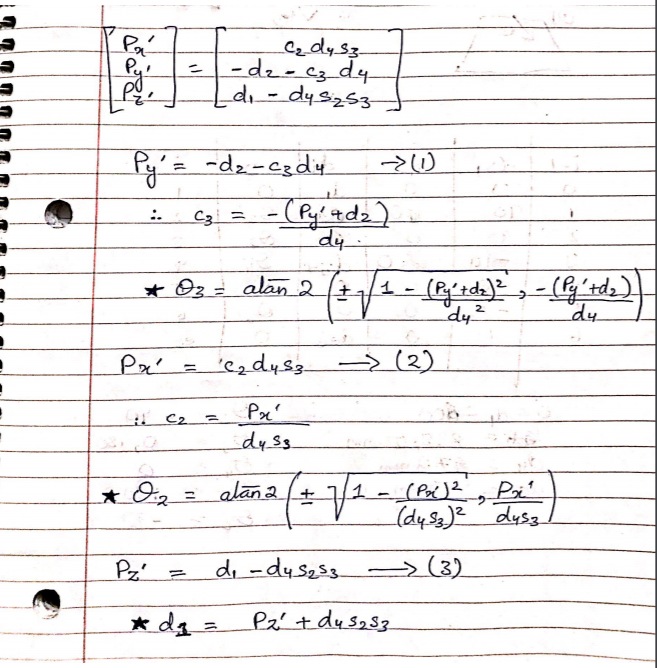


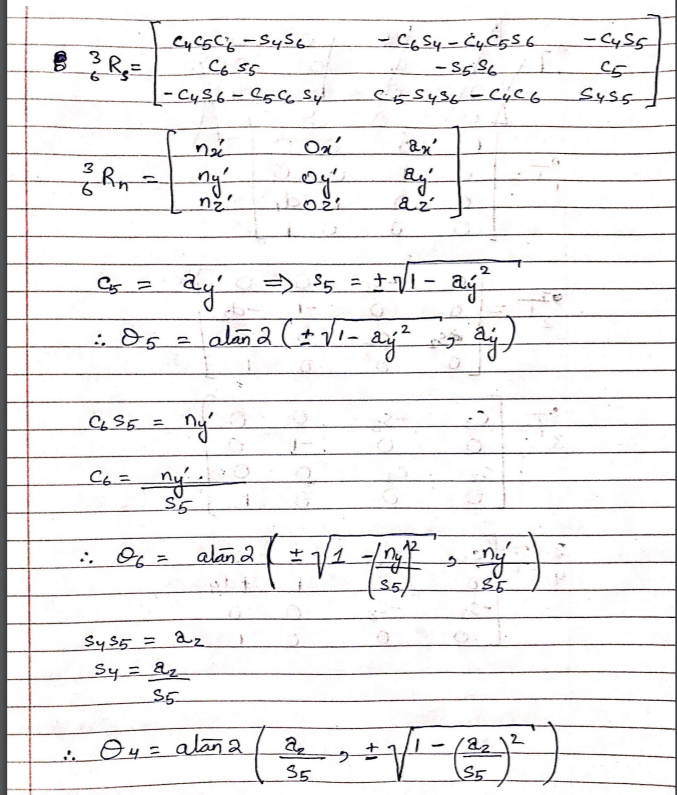






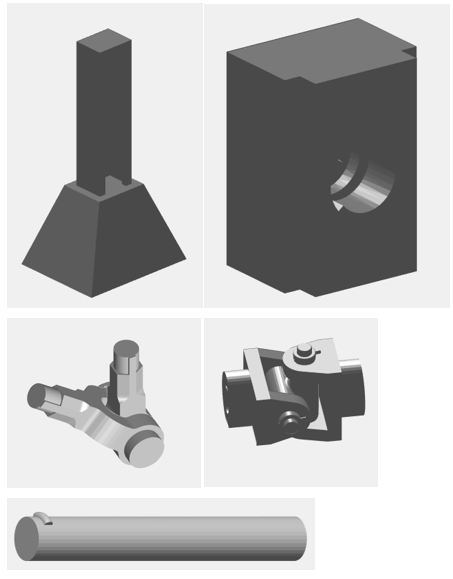


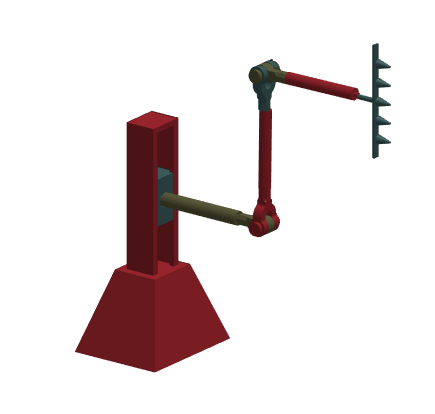




# Kinematic Reconstruction Path Generation

For this section of the report, the links and joints were aligned to be settled on the global origin of the reference frame on the MATLAB, to be animated when the get shape of the manipulator. Each part is to be associated to the reference frame that exerts or enable it to be in motion.

Below mention are the STL files for the project.

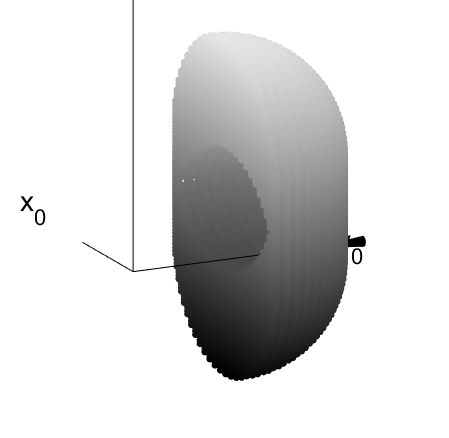


The picture above shows the kinematic reconstruction of the manipulator.

In the code, the path generation have 240 points in it and it moves on a continuous path.

# Workspace Analysis

For workspace model, kindly run the code attached in the zip file.



The hole in the middle of the workspace is due the length of fourth joint and translation of prismatic joint.

# Path Generation

The path generation for the spray manipulator is paint 5 at 4 locations each. Therefore 20 circles in total.

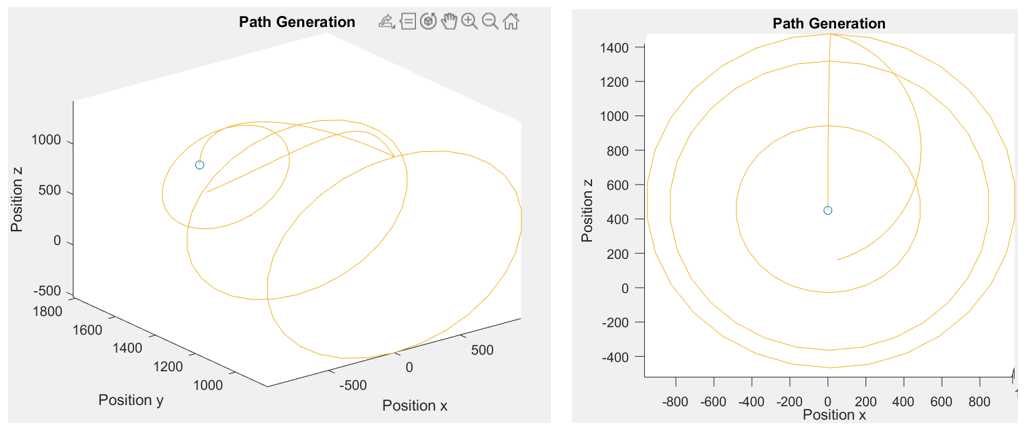
* The 1st set of 5 circles are plot at n=1, Plot = 838.5+972\*cos (90) =838.5 mm
* The 2nd set of 5 circles are plot at n=2, Plot = 838.5+972\*cos (60) =1324.5 mm
* The 3rd set of 5 circles are plot at n=3, Plot = 838.5+972\*cos (90) =1680.3 mm
* The 4th set of 5 circles are plot at n=4, Plot = 838.5+972\*cos (90) =1810.5 mm

By using the function *my\_path* & *P\_xyz\_abg*, the values of x, y, z, and alpha, beta, and gamma.

The *my\_path* function was used to compute the homogenous transforms for the matrices on the desired set of the values for joints. *P\_xyz\_abg* function was used to compute their respective x, y, z, and alpha, beta, and gamma for the end effector. The functions are provided in the appendix section below.

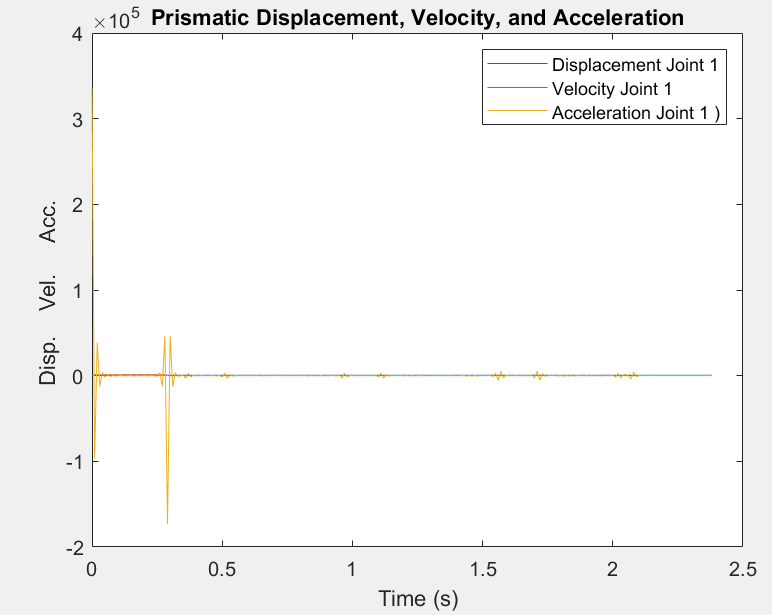
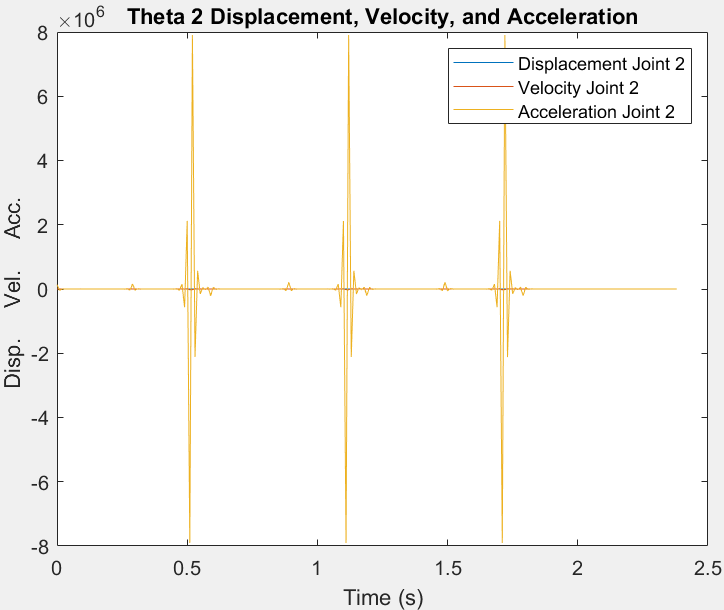
# Trajectory Generation

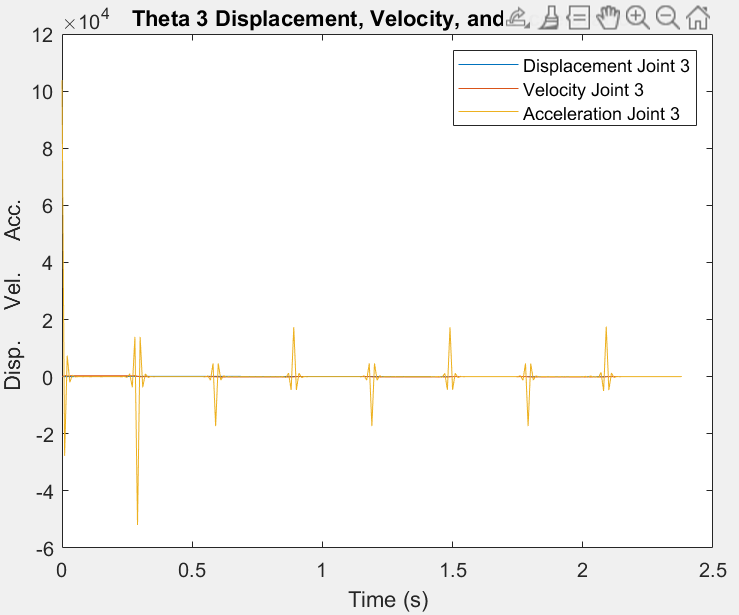
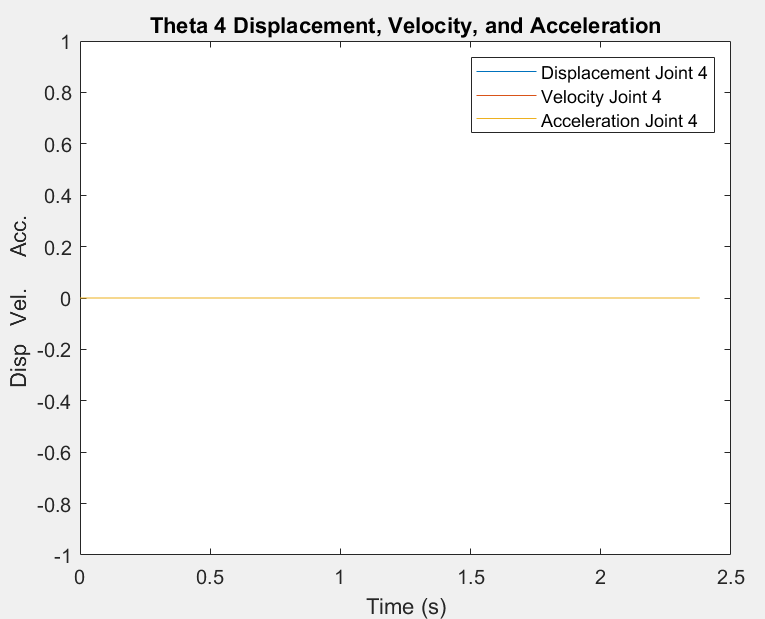
The following figure show the trajectory generation of the manipulator.

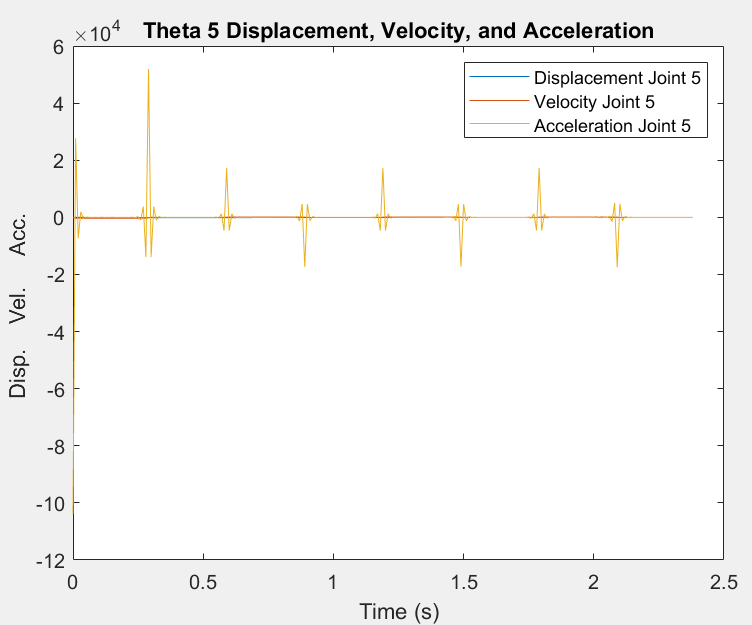
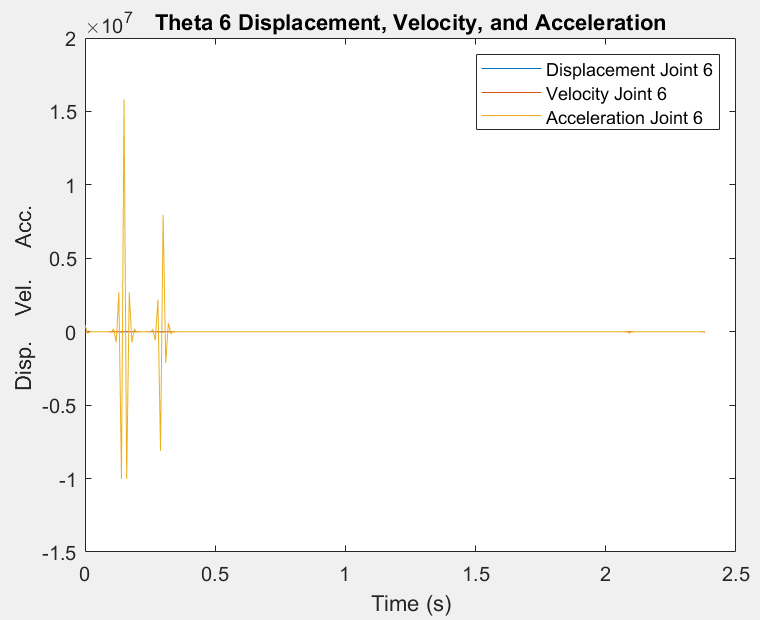


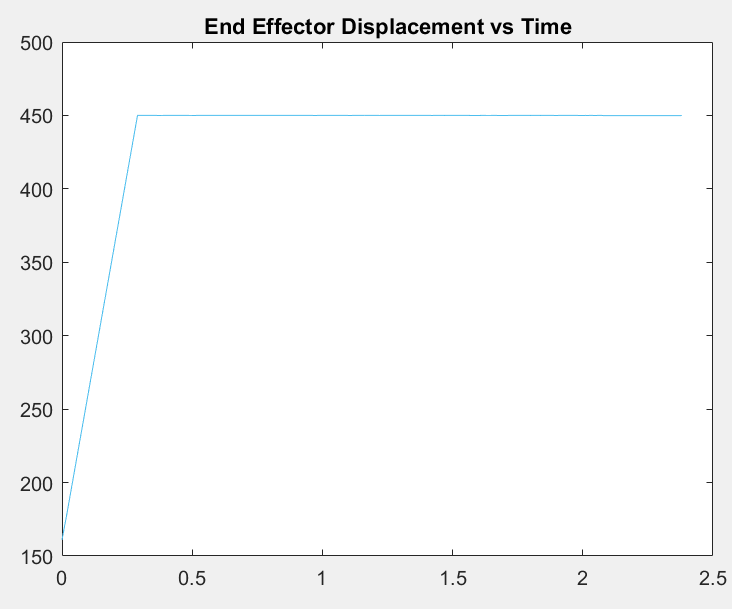
The above figure shows 4 circles (5 circles in each circle) in 3d. The XZ plane shows the 4 circles with the blue dot being the 4th one.

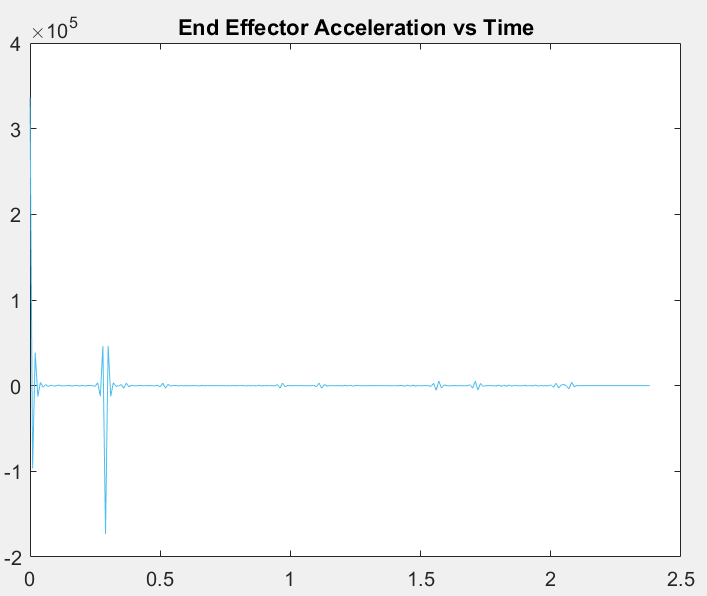
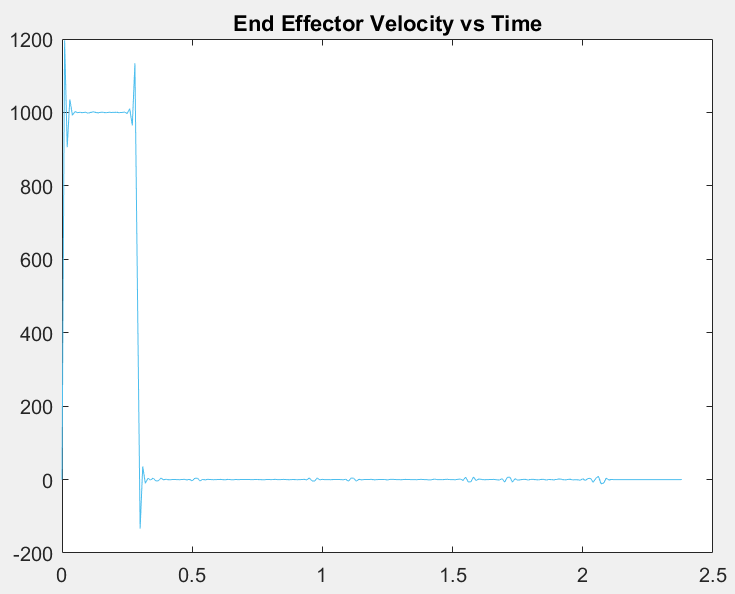
Below are the plots for each joints displacement, velocity and acceleration.

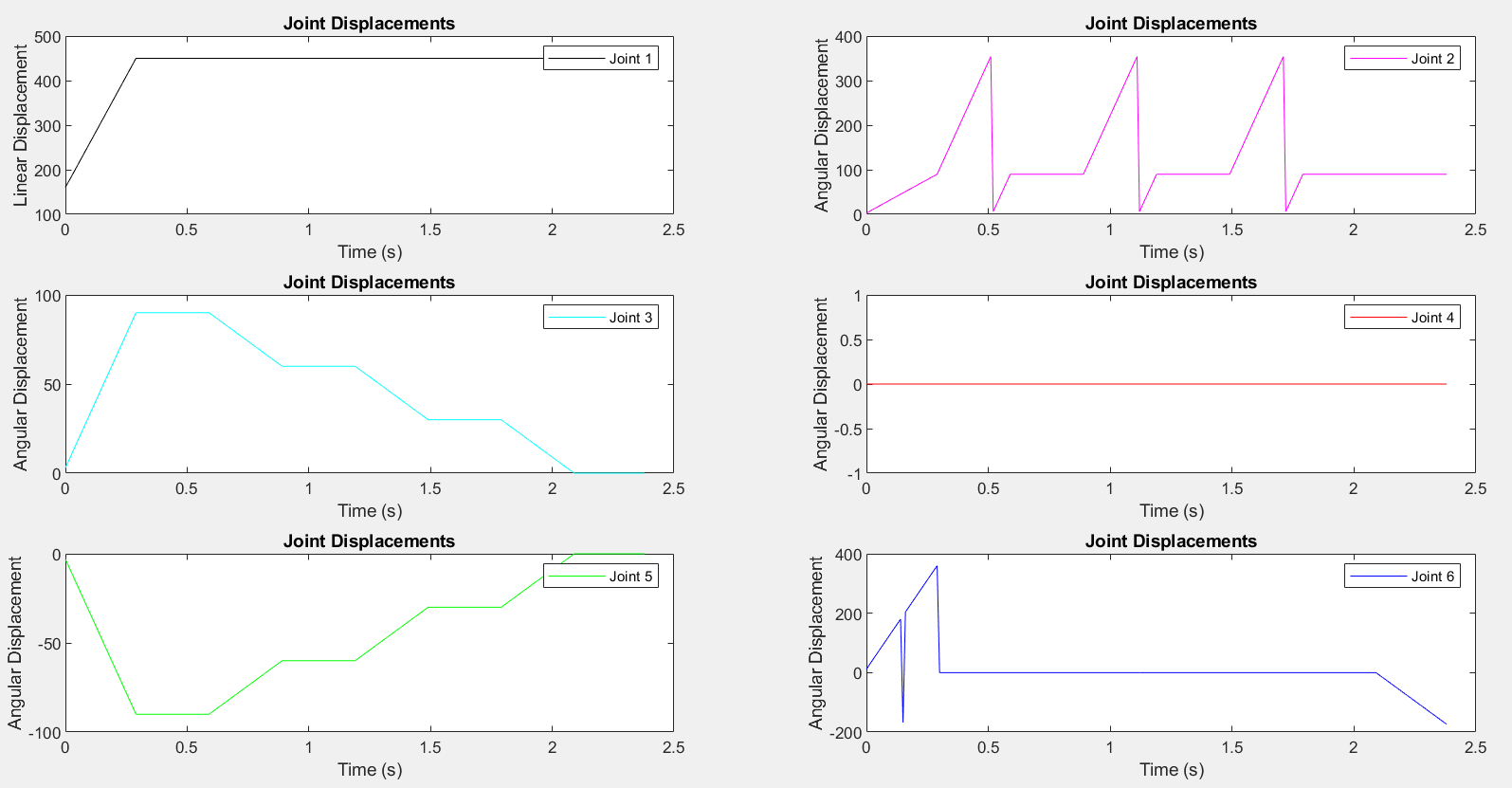
 

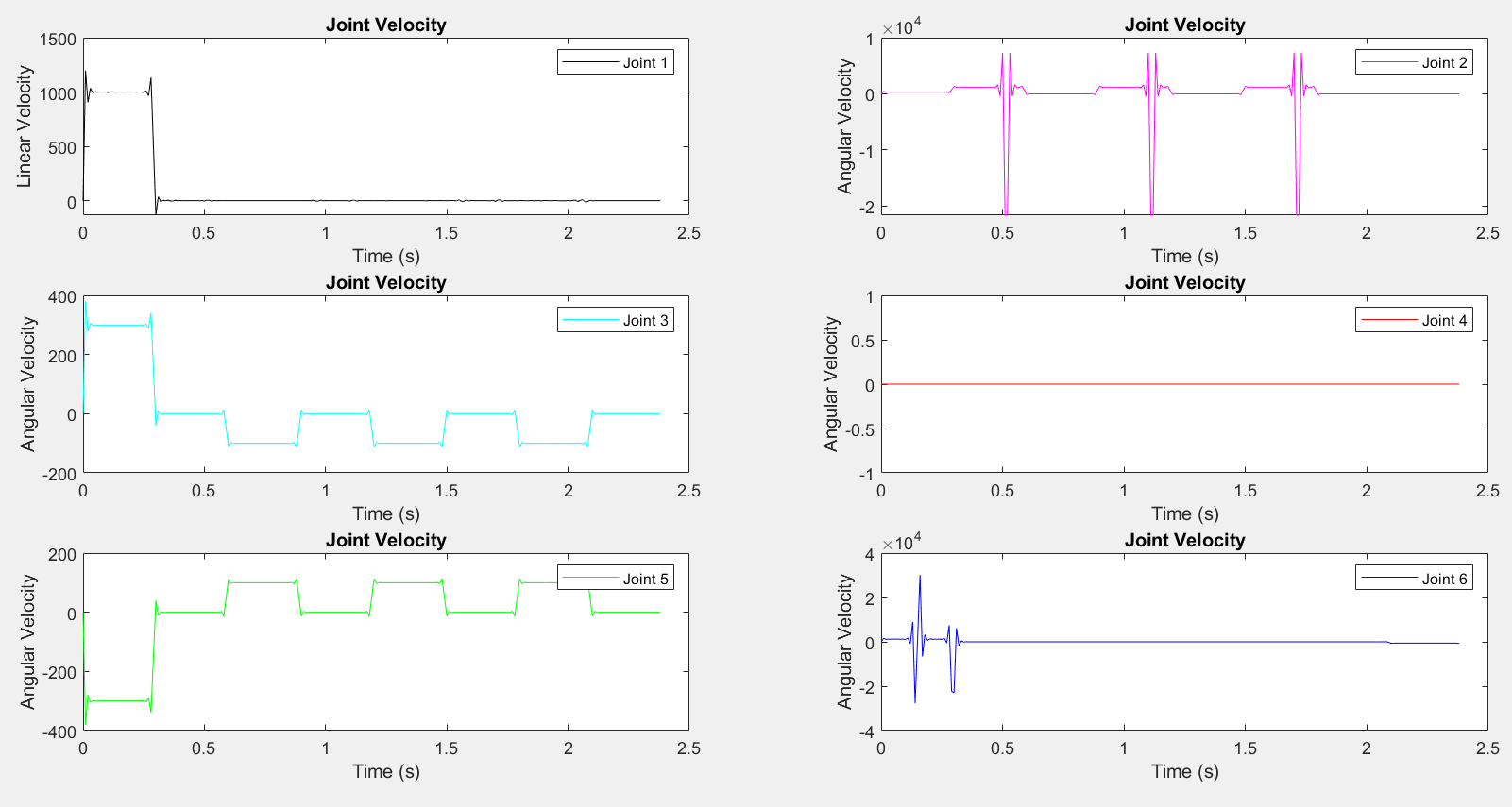




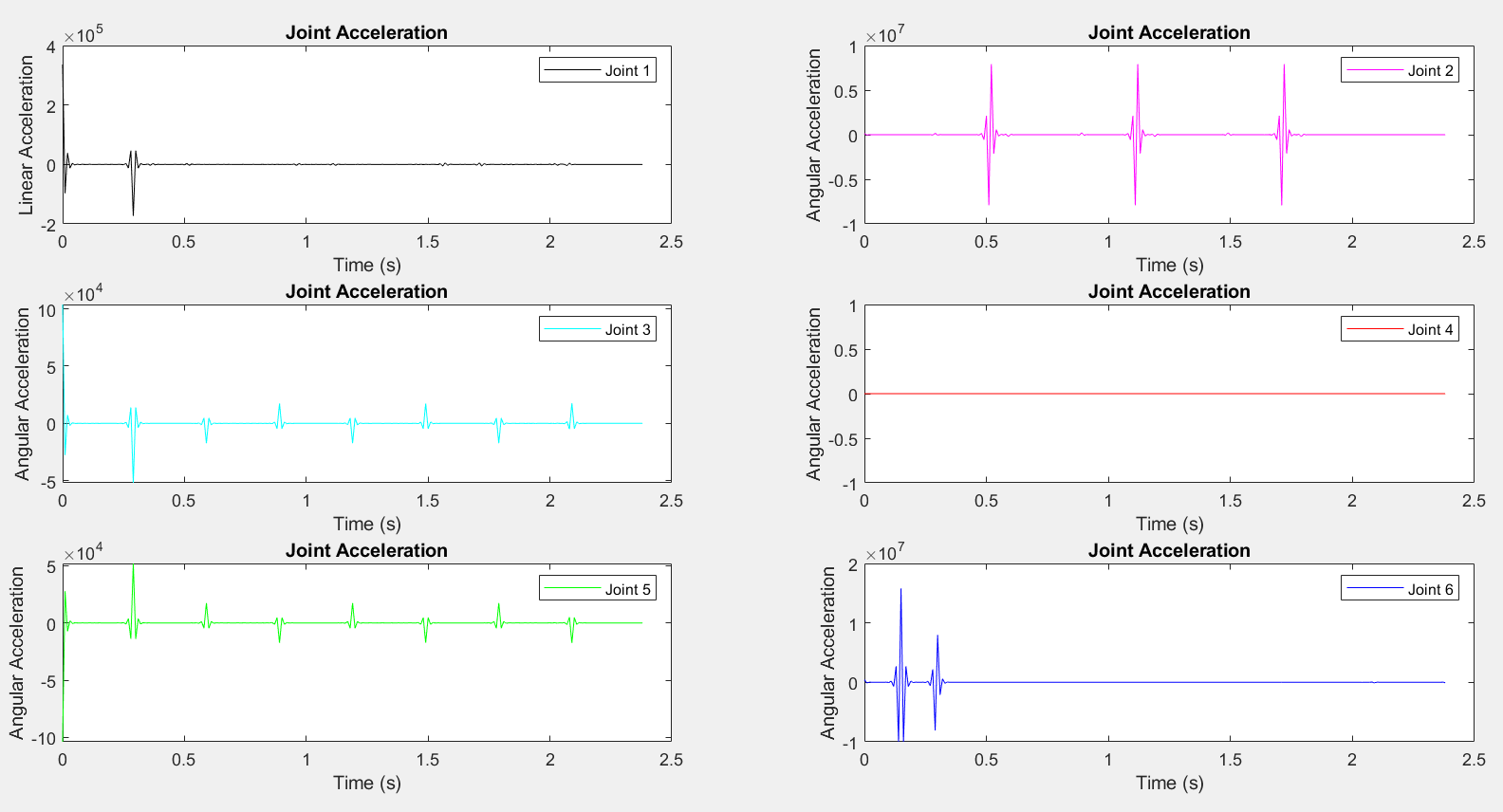
The below plot shows all the joint displacement all together.



The below plot shows all the joints velocities.

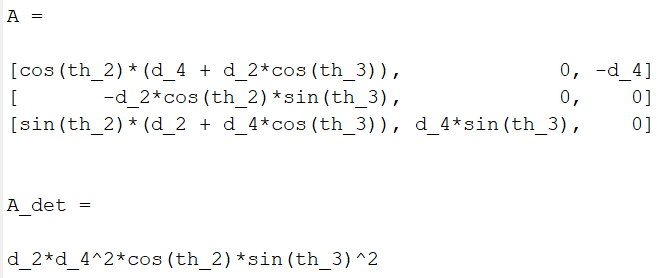


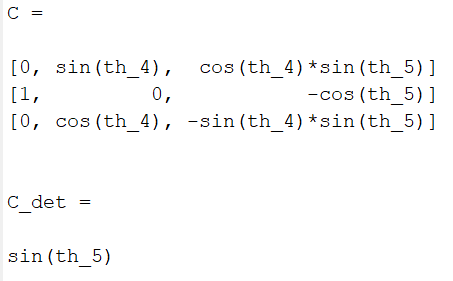
The below plot shows all the joints acceleration.

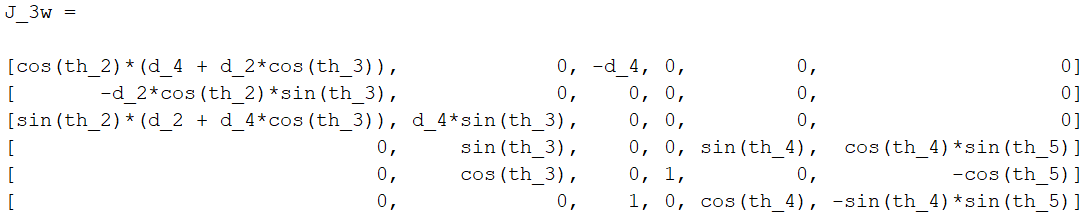


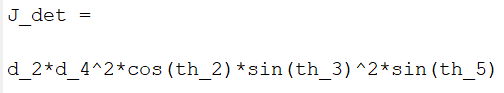
# Jacobian

The jacobian was computed with the symbolic computation on matlab. The function in the appendix my\_jacobian\_symbolic computes all the symbolic variables.









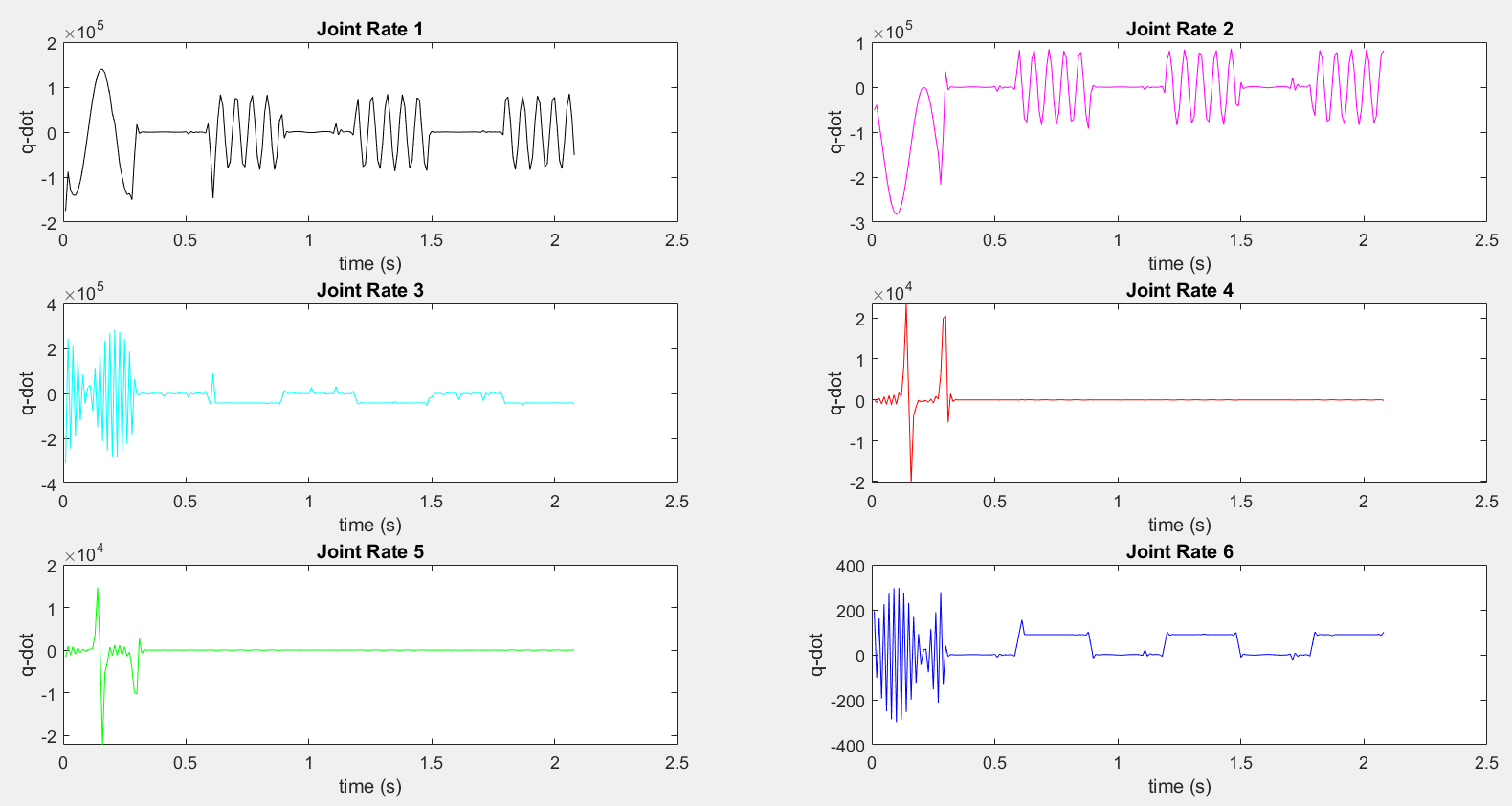
## Singularity analysis

Therefore , ,

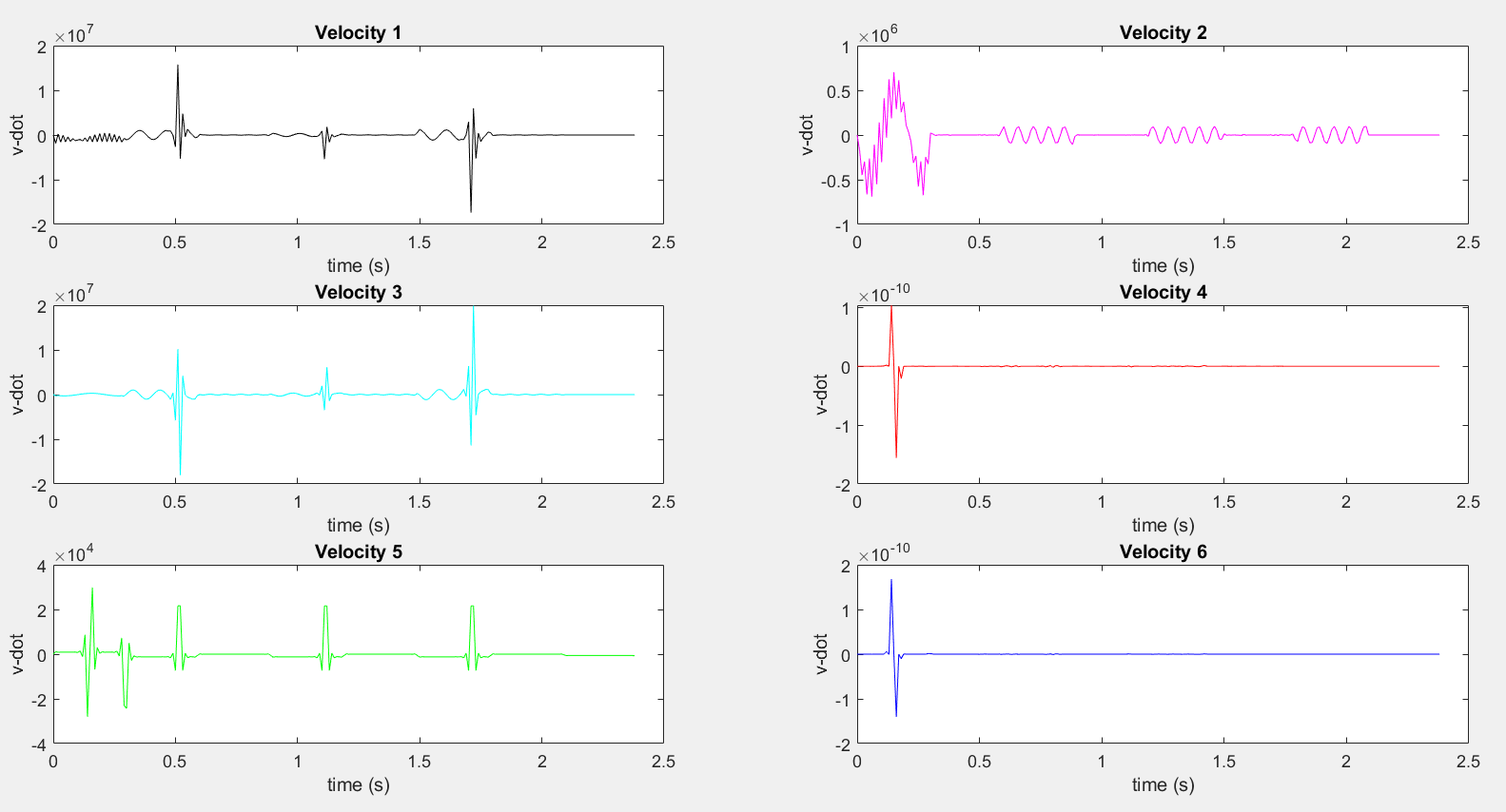
Therefore ,

Hence , , ,

The below plot is for inverse velocity.



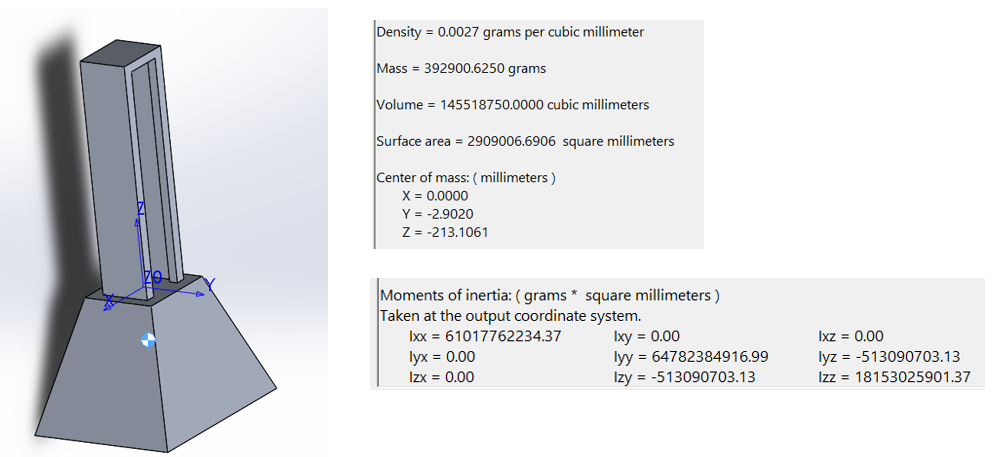
The below plot is for forward velocity.



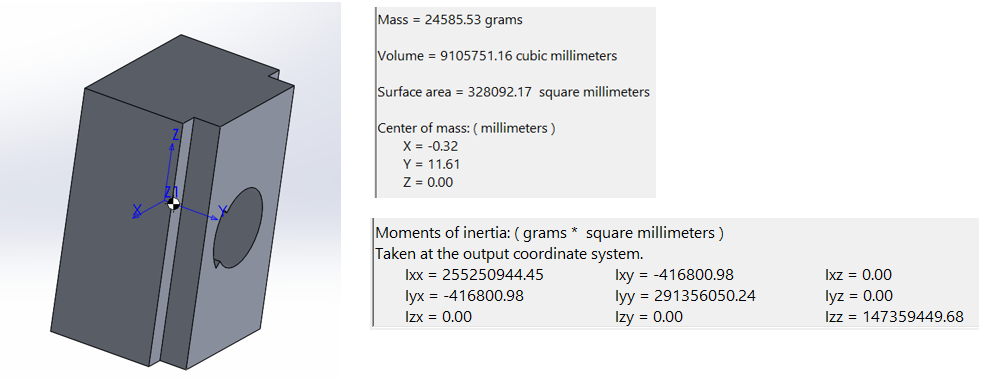
# Link Modeling

The following figure shows the inertia tensor for the manipulator parts. The function for the inertia tensor is in the appendix.

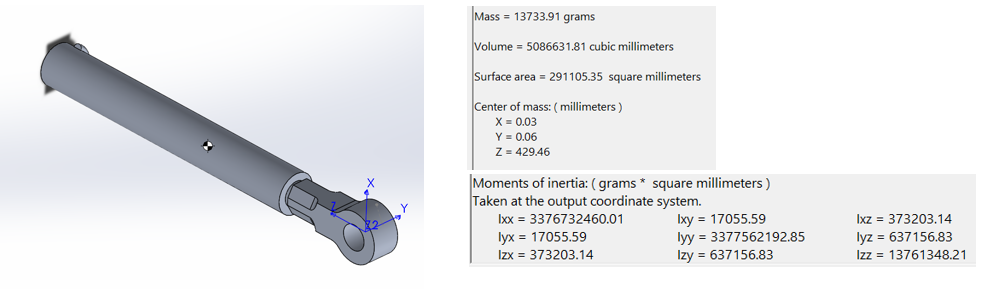
## Base



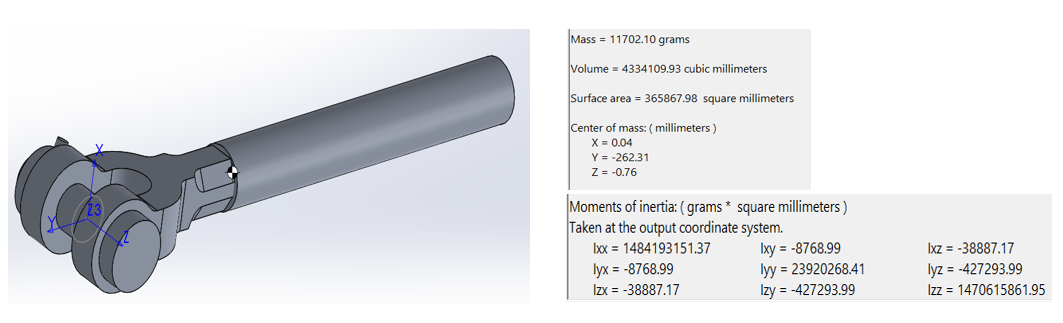
## Prismatic Joint



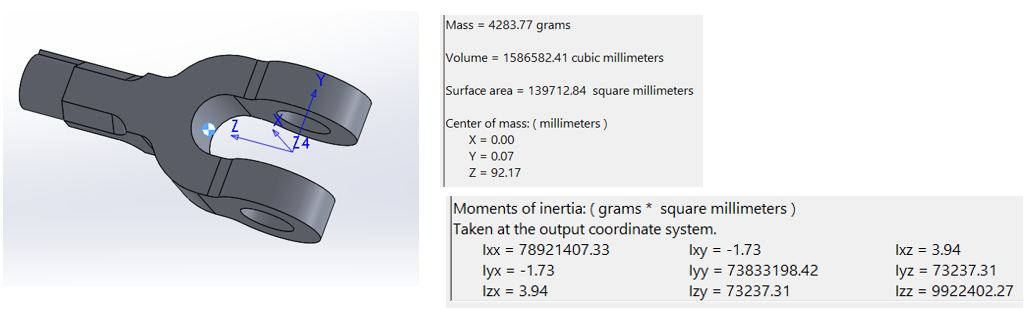
## Revolute Joint 2



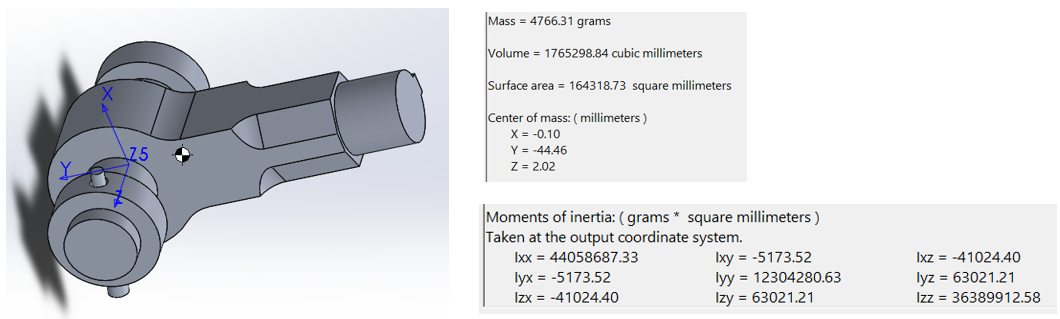
## Revolute Joint 3



## Revolute Joint 4



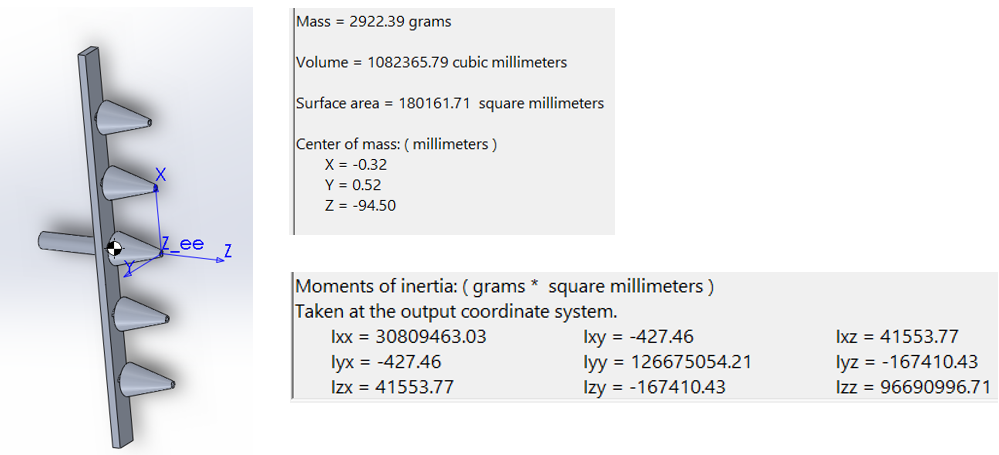
## Revolute Joint 5



## Revolute Joint 6



## End Effector



# Dynamics

Dynamics analysis was conducted using Newton-Euler Recursive formulation. The outward iteration was computed first and was followed by the Inward Iteration. The code for computation is provided in section3 of the appendix.

Execute the matlab my\_jacobian\_symbolic function in the appendix for the inverse velocity and forward velocity analysis as well as force and moment analysis and homogenous and velocity transformation matrix. The symbolic computation is a long matrix even after being simplified as well.

%% Velocity Transformation Matrix

%Position Vector P ref\_w->ee

R\_06 = T\_06(1:3,1:3);

P\_6ee = T\_6ee(1:3,4);

P\_0eew = -R\_06\*P\_6ee;

skew1 = [0 -P\_0eew(3,1) P\_0eew(2,1)

P\_0eew(3,1) 0 -P\_0eew(1,1)

-P\_0eew(2,1) P\_0eew(1,1) 0];

Tv = simplify([R\_03 skew1\*R\_03; zeros(3) R\_03])

%% Forward/Inverse Velocity Equations

%Forward Velocity

syms q1 q2 q3 q4 q5 q6

q = [q1 q2 q3 q4 q5 q6];

vel\_0ee = simplify(Tv\*J\_3w\*q.')

%Inverse Velocity

syms v1 v2 v3 v4 v5 v6

vel = [v1 v2 v3 v4 v5 v6];

q\_dot = simplify(inv(Tv)\*inv\_J\_3w\*vel.')

%% Force Transformation Matrix

R\_30 = R\_03.';

P\_3eew = R\_36\*P\_6ee;

skew2 = [ 0 -P\_3eew(3,1) P\_3eew(2,1)

P\_3eew(3,1) 0 -P\_3eew(1,1)

-P\_3eew(2,1) P\_3eew(1,1) 0];

Fv = simplify([R\_30 zeros(3); skew2\*R\_30 R\_30])

%% Inverse Static Force

syms f1 f2 f3 m1 m2 m3

f = [f1 f2 f3 m1 m2 m3];

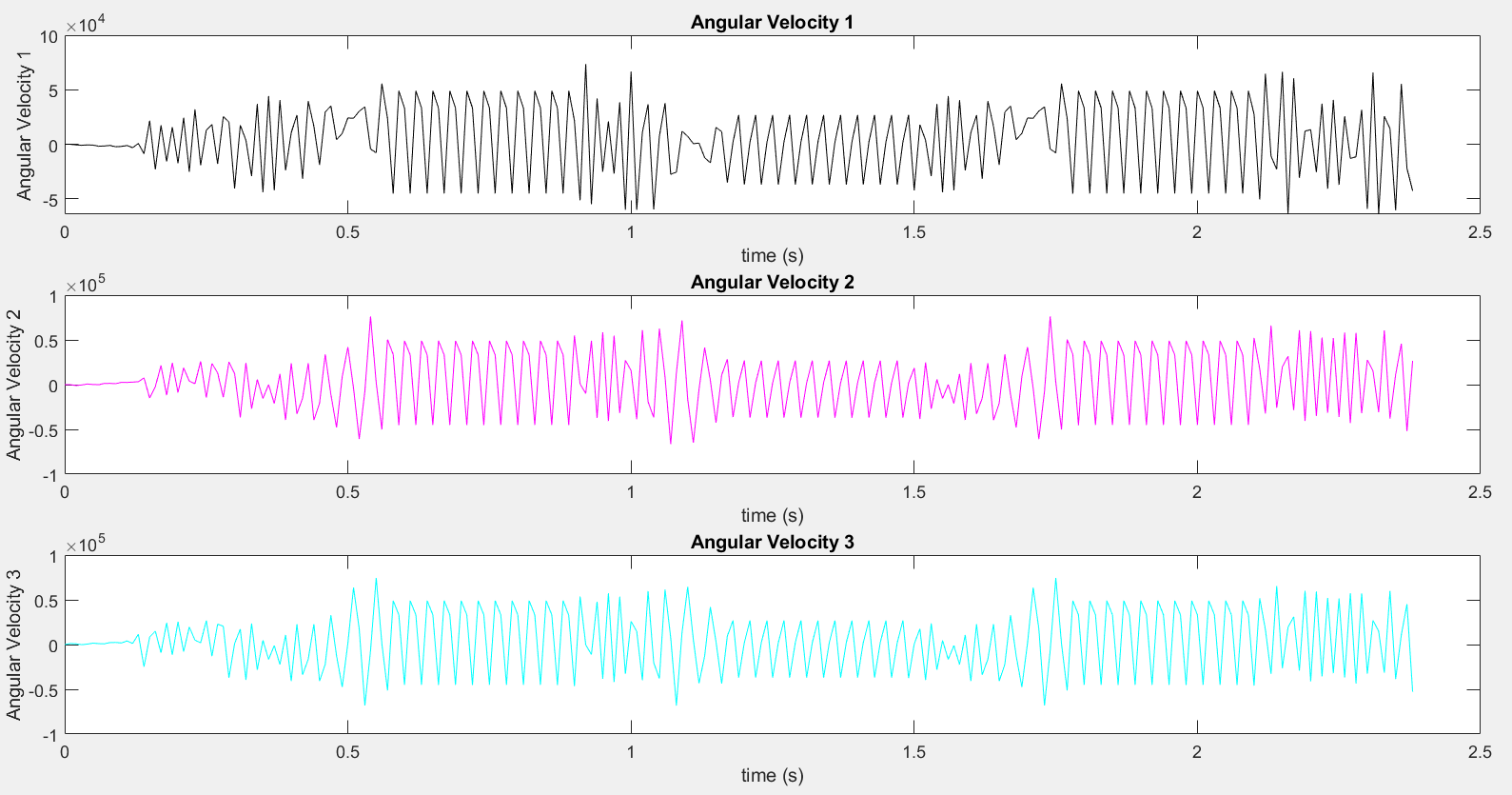
J\_3wt = transpose(J\_3w);

torque = simplify(J\_3wt\*Fv\*f.')

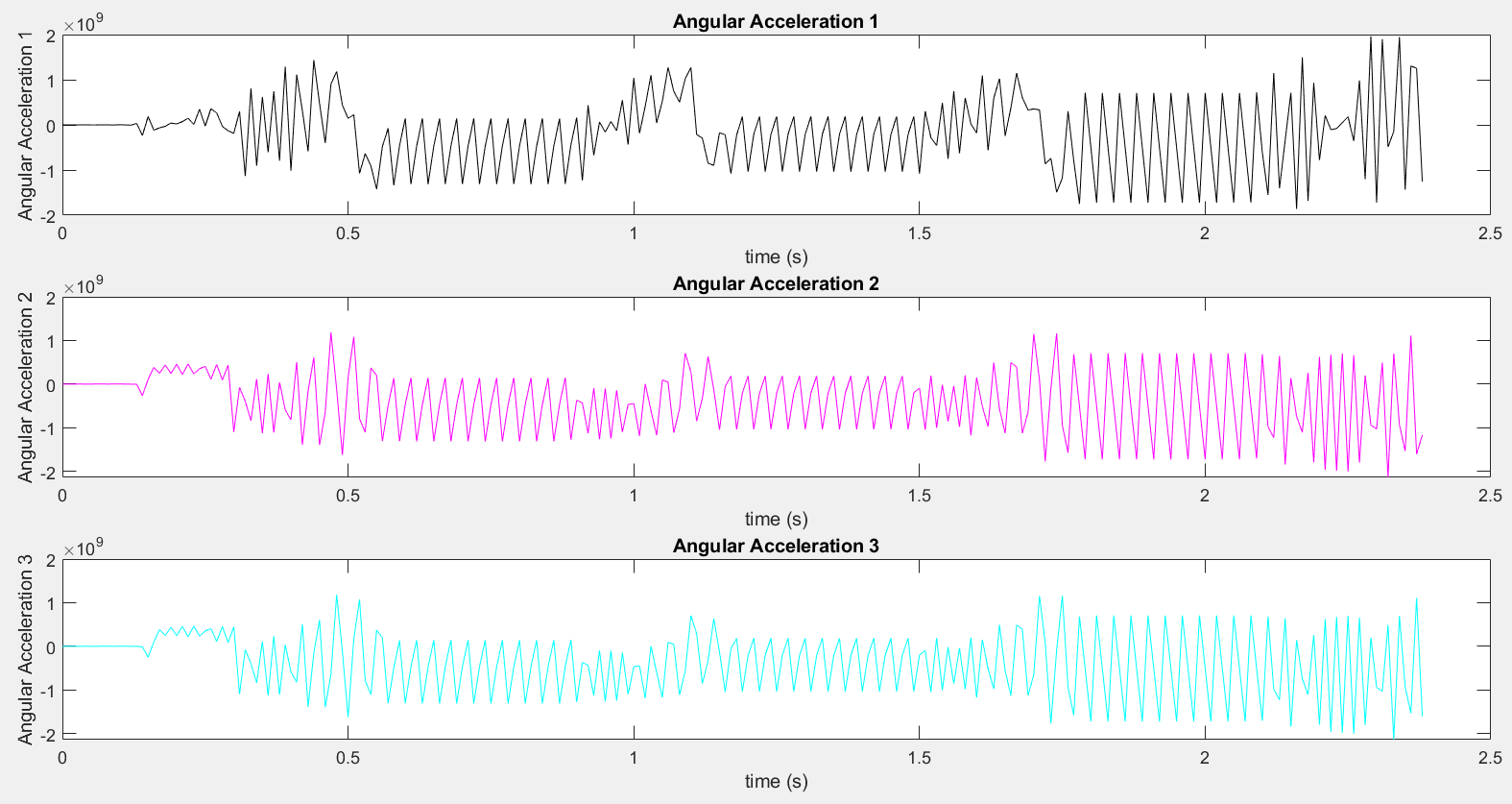
# Simulation

The following plots shows the simulations of the Dynamic analysis of the manipulator.

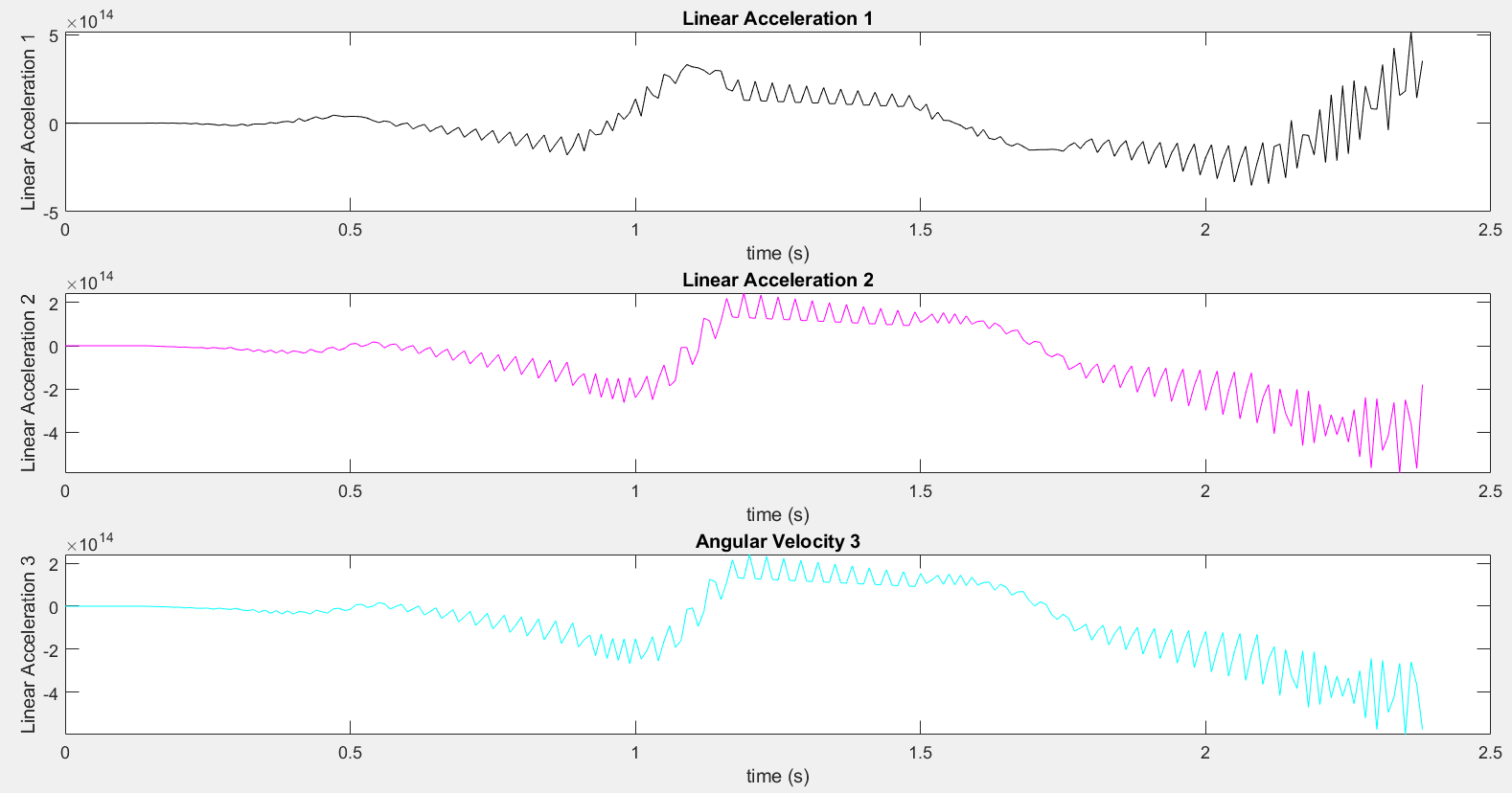
The below figure shows the Angular velocities plot against time.



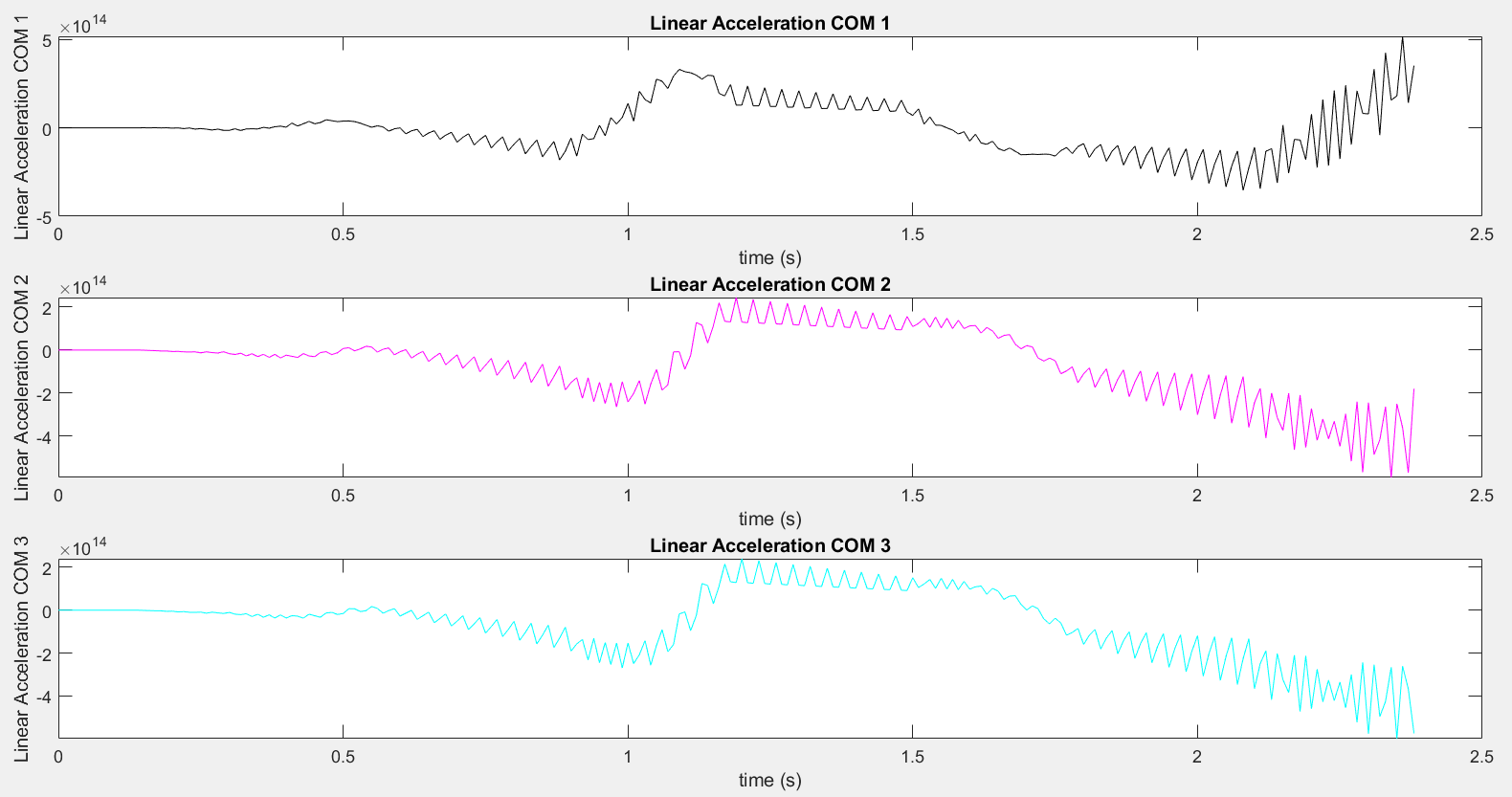
The below figure shows the Angular acceleration plot against time.



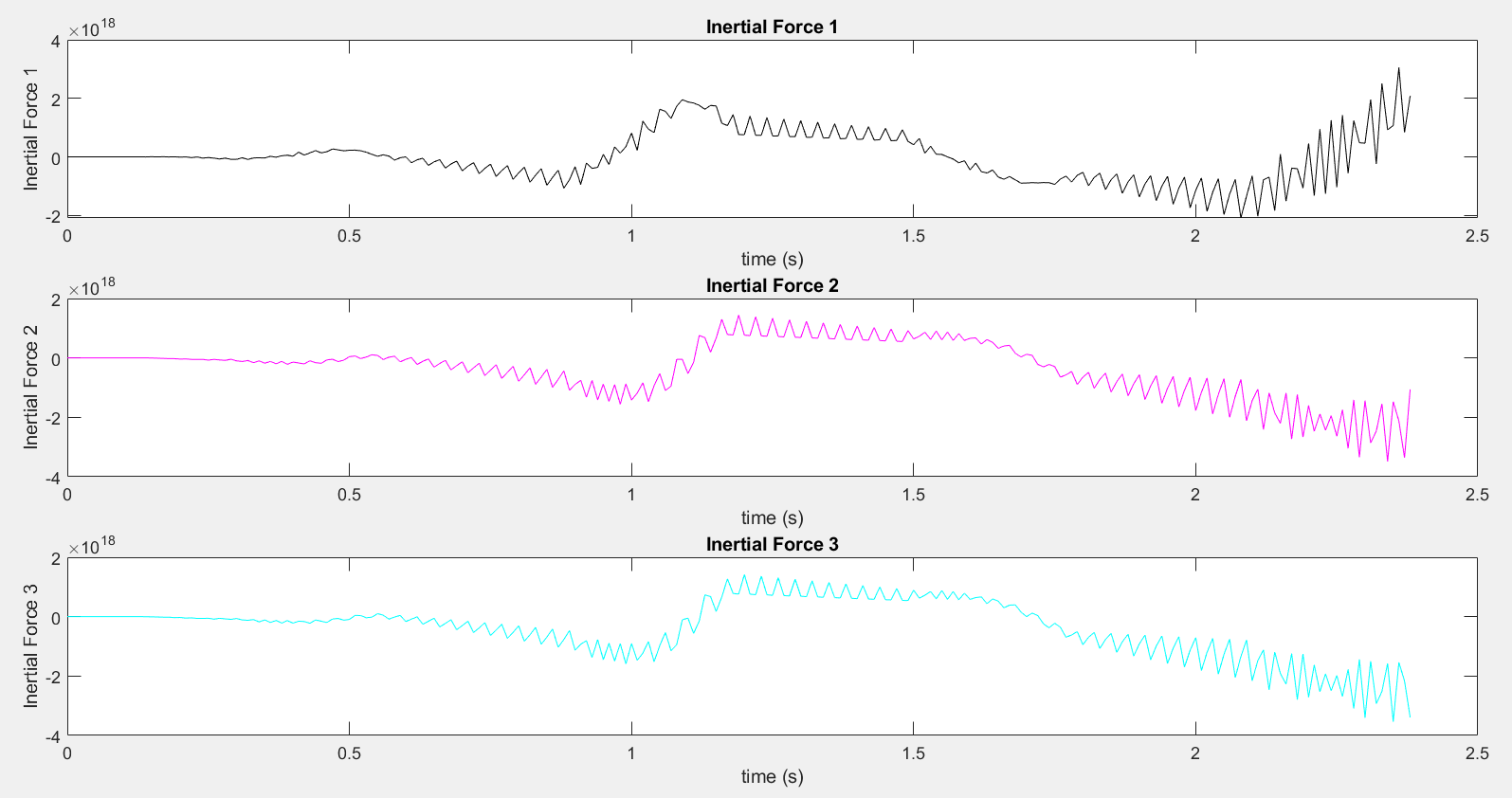
The below figure shows the Linear acceleration plot against time.



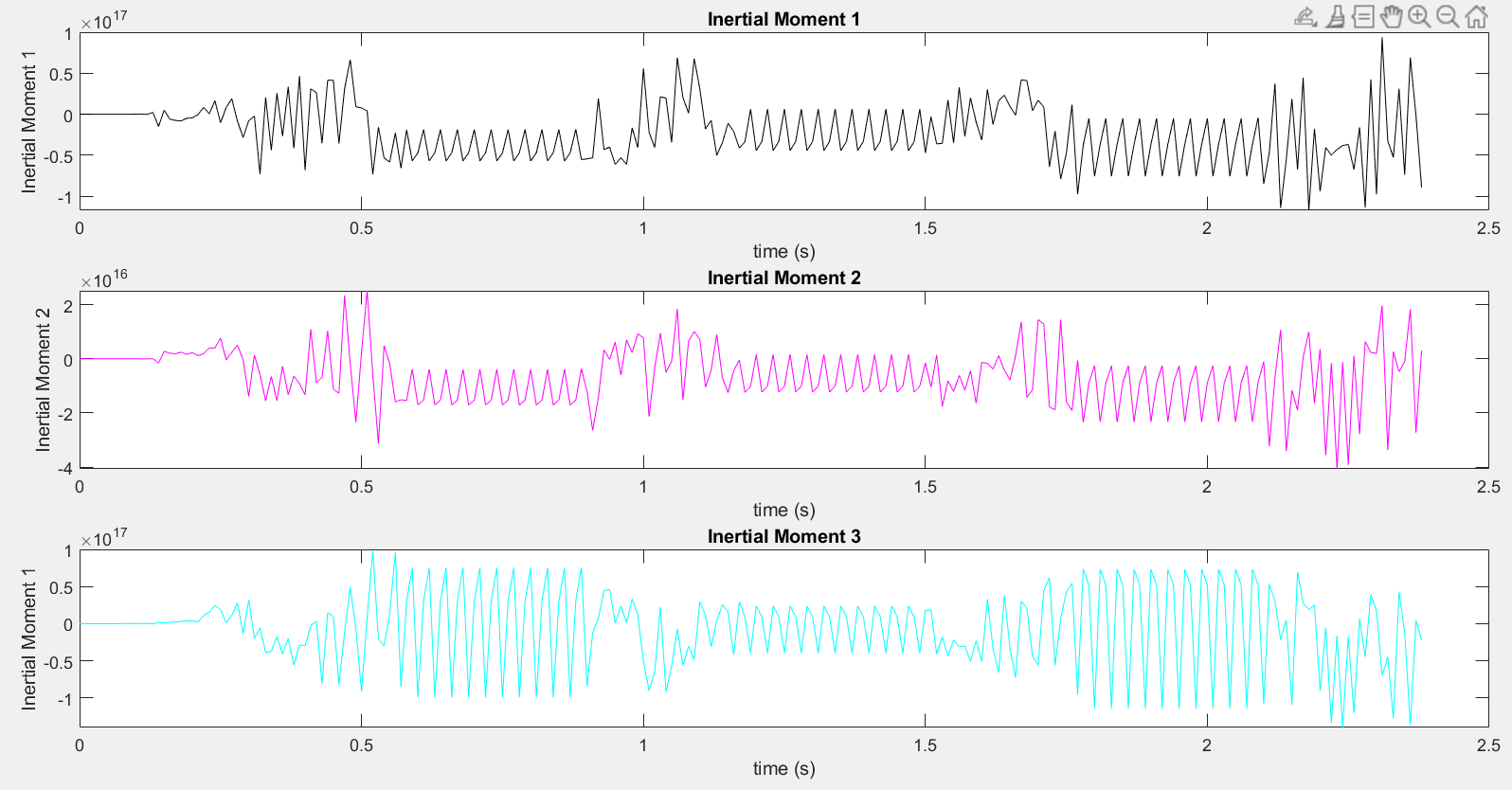
The below figure shows the Linear acceleration COM plot against time.



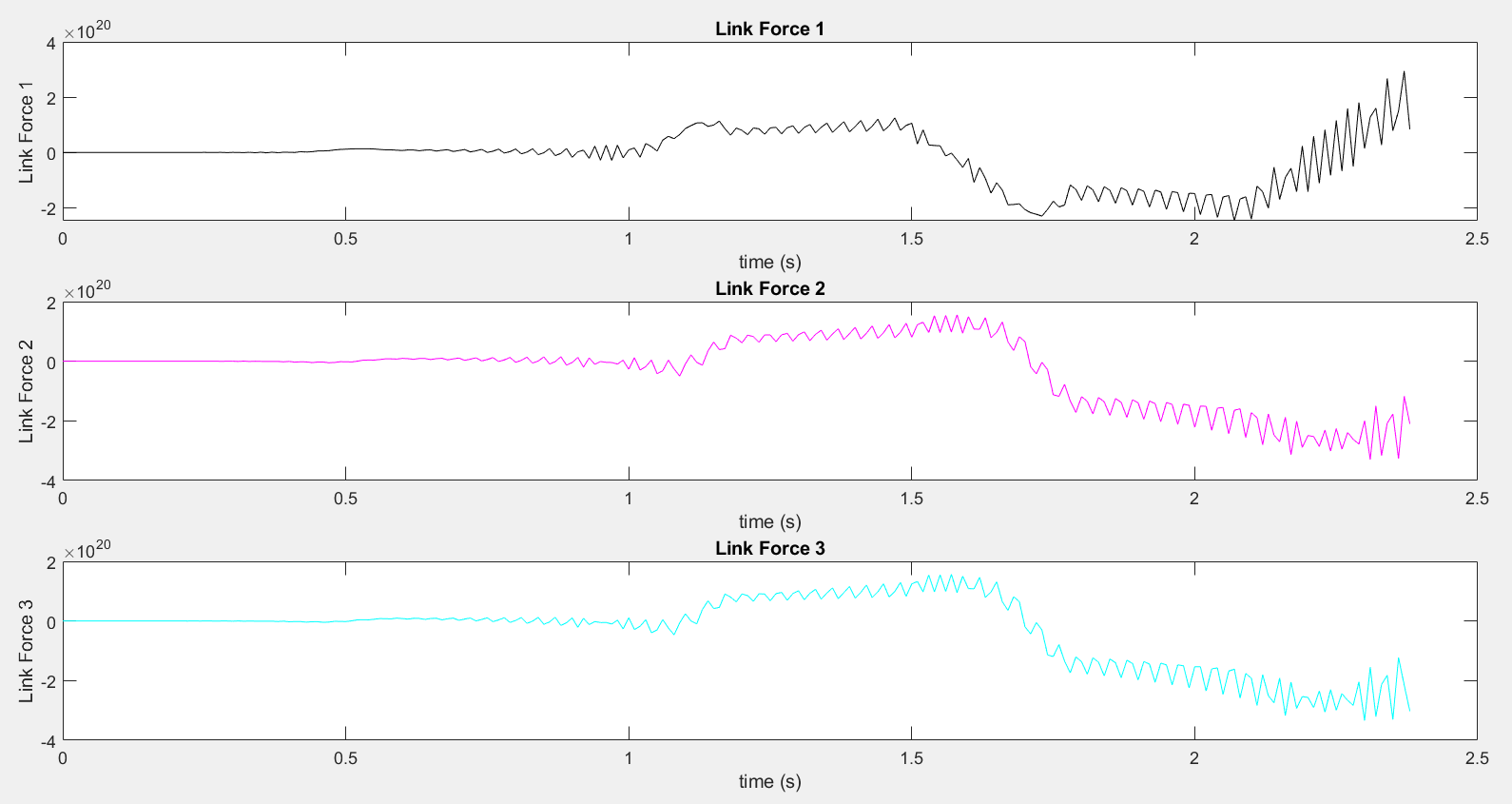
The below figure shows the Inertial Force plot against time.



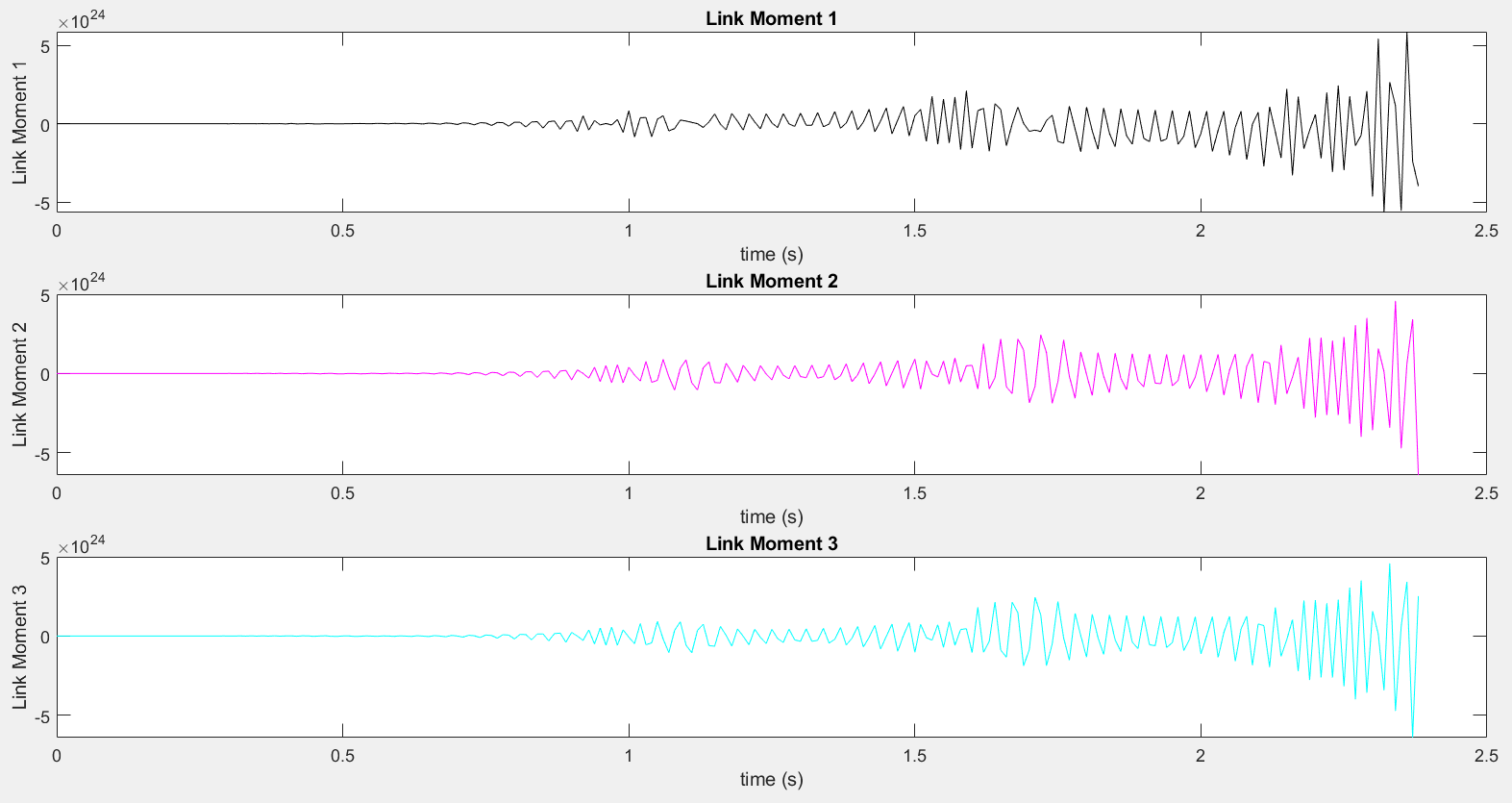
The below figure shows the Inertial Moment plot against time.



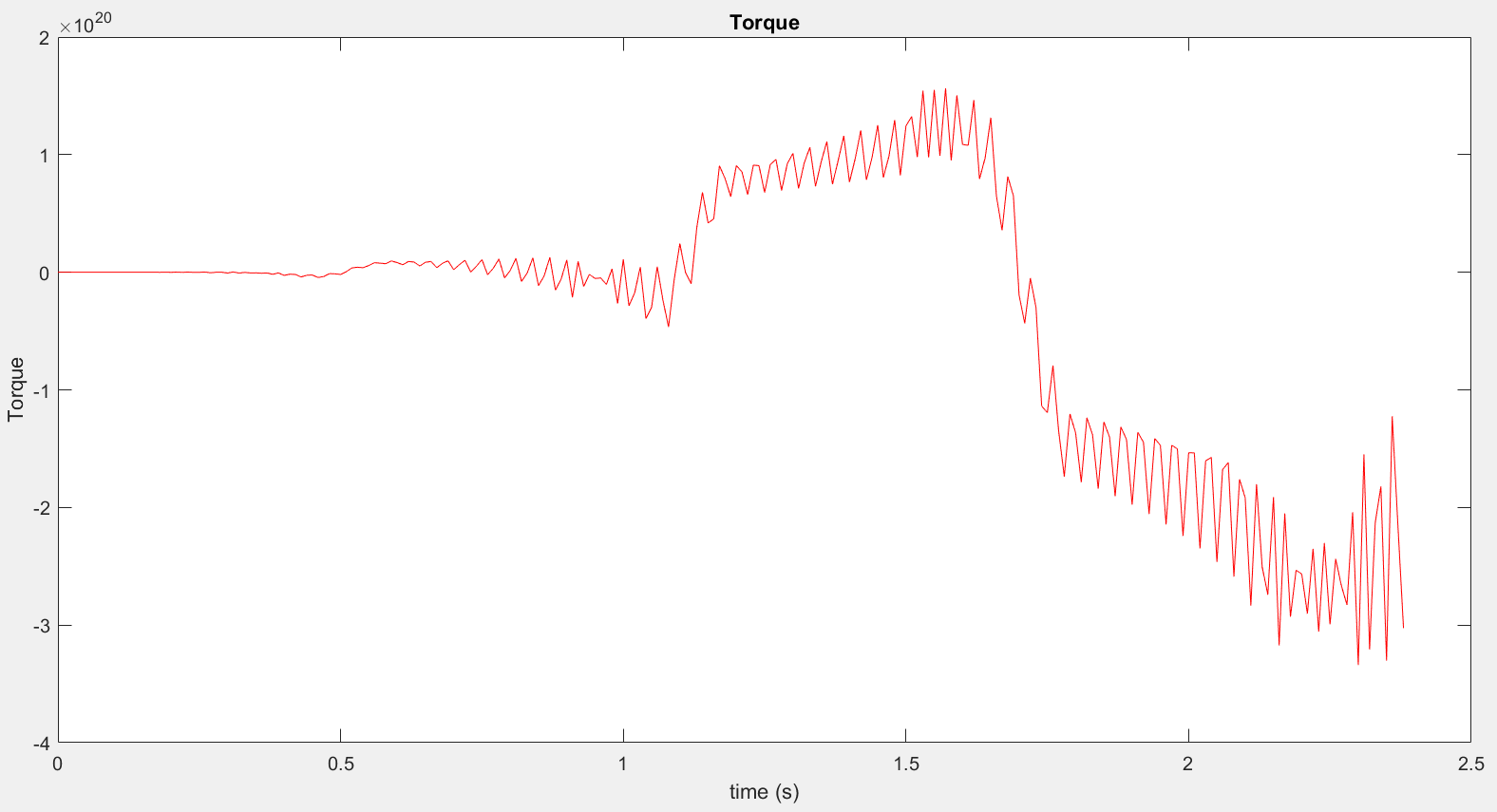
The below figure shows the Link Force plot against time.



The below figure shows the Link Moment plot against time.



The below figure shows the Torque plot against time.



# Conclusion

Working on this whole project report also help in giving out an insight for inverse kinematics, and dynamic analysis that one may need to consider in precise motion control and a better understanding of various kinds of joints and their useability. In conclusion, I would like to say that this design is opened to change as when new parameters are available, for example when implying the trajectory path for the end effector each of the link will move according to it and therefore new design constraint will introduce, as there will be new workspace, that will result in change of SolidWorks design and enhance workspace ability of the manipulator. The re-construction part was a bit time consuming as it requires individual parts in SolidWorks to settle on the global origin of the reference plane, and then maps in such a manner that it joins its link connection or the joint connection that allows it moves in the free space. The inverse kinematics, dynamic analysis coding was challenging, and it gives an insight of how to apply the iterative approach for the end effector.

# References

1. ABB Robotics - Manufacturer & Supplier of Industrial Robots. (2020). Retrieved 5 October 2020, from <https://new.abb.com/products/robotics>
2. MSE Lecture Notes Flavio Firmani. September 2020.
3. What is a Robotic Manipulator?. (2020). Retrieved 6 October 2020, from <https://www.azorobotics.com/Article.aspx?ArticleID=138>

# Appendix

These are the provided functions to code and compute the forward kinematics, inverse kinematics, trajectory, jacobian analysis and dynamic analysis of the manipulator.

## my\_path

function path\_mat = my\_path()

path\_mat=zeros(6,240);

for j=1:length(path\_mat(1,:))

if (j>=1 && j<=30)

path\_mat(1,j)=150+10\*j;

path\_mat(2,j)=3\*j;

path\_mat(3,j)=3\*j;

path\_mat(4,j)=0;

path\_mat(5,j)=-3\*j;

path\_mat(6,j)=12\*j;

if (path\_mat(6,j)>180)

path\_mat(6,j)=12\*j-360;

end

elseif (j>=31 && j<=60)

path\_mat(1,j)=450;

path\_mat(2,j)=90+12\*(j-30);

if (path\_mat(2,j)>180)

path\_mat(2,j)=90+12\*(j-30)-360;

end

path\_mat(3,j)=90;

path\_mat(4,j)=0;

path\_mat(5,j)=-90;

path\_mat(6,j)=0;%12\*(j-30);

elseif (j>=61 && j<=90)

path\_mat(1,j)=450;

path\_mat(2,j)=90;

path\_mat(3,j)=90-(j-60);

path\_mat(4,j)=0;

path\_mat(5,j)=-90+(j-60);

path\_mat(6,j)=0;

elseif (j>=91 && j<=120)

path\_mat(1,j)=450;

path\_mat(2,j)=90+12\*(j-90);

if (path\_mat(2,j)>180)

path\_mat(2,j)=90+12\*(j-90)-360;

end

path\_mat(3,j)=60;

path\_mat(4,j)=0;

path\_mat(5,j)=-60;

path\_mat(6,j)=0;%12\*(j-90);

elseif (j>=121 && j<=150)

path\_mat(1,j)=450;

path\_mat(2,j)=90;

path\_mat(3,j)=60-(j-120);

path\_mat(4,j)=0;

path\_mat(5,j)=-60+(j-120);

path\_mat(6,j)=0;

elseif (j>=151 && j<=180)

path\_mat(1,j)=450;

path\_mat(2,j)=90+12\*(j-150);

if (path\_mat(2,j)>180)

path\_mat(2,j)=90+12\*(j-150)-360;

end

path\_mat(3,j)=30;

path\_mat(4,j)=0;

path\_mat(5,j)=-30;

path\_mat(6,j)=0;%12\*(j-150);

elseif (j>=181 && j<=210)

path\_mat(1,j)=450;

path\_mat(2,j)=90;

path\_mat(3,j)=30-(j-180);

path\_mat(4,j)=0;

path\_mat(5,j)=-30+(j-180);

path\_mat(6,j)=0;

else

path\_mat(1,j)=450;

path\_mat(2,j)=90;

path\_mat(3,j)=0;

path\_mat(4,j)=0;

path\_mat(5,j)=0;

path\_mat(6,j)=6\*(210-j);

end

end

## P\_xyz\_abg

D=my\_path;

for i=1:length(D(1,:))

%DH parameters (CHANGE BASED ON THE JOINT VARIABLE)

T\_01 = tmat(alpha0, a0, D(1,i), theta1);

T\_12 = tmat(alpha1, a1, d2, D(2,i));

T\_23 = tmat(alpha2, a2, d3, D(3,i));

T\_34 = tmat(alpha3, a3, d4, D(4,i));

T\_45 = tmat(alpha4, a4, d5, D(5,i));

T\_56 = tmat(alpha5, a5, d6, D(6,i));

T\_6ee = tmat(alpha6, a6, dee, thetaee);

%Forward Kinematics

T\_02 = T\_01\*T\_12;

T\_03 = T\_02\*T\_23;

T\_04 = T\_03\*T\_34;

T\_05 = T\_04\*T\_45;

T\_06 = T\_05\*T\_56;

T\_0ee = T\_06\*T\_6ee; %Homogeneous Tranforms

%Position and Rotation matrices of frames

R\_01 = T\_01(1:3,1:3); P\_01 = T\_01(1:3,4);

R\_02 = T\_02(1:3,1:3); P\_02 = T\_02(1:3,4);

R\_03 = T\_03(1:3,1:3); P\_03 = T\_03(1:3,4);

R\_04 = T\_04(1:3,1:3); P\_04 = T\_04(1:3,4);

R\_05 = T\_05(1:3,1:3); P\_05 = T\_05(1:3,4);

R\_06 = T\_06(1:3,1:3); P\_06 = T\_06(1:3,4);

R\_0ee = T\_0ee(1:3,1:3); P\_0ee = T\_0ee(1:3,4);

% final position

Px(1,i)=P\_06(1,1);

Py(1,i)=P\_06(2,1);

Pz(1,i)=P\_06(3,1);

r11=R\_06(1,1);

r21=R\_06(2,1);

r31=R\_06(3,1);

r32=R\_06(3,2);

r33=R\_06(3,3);

%beta

sb = -r31; cb = sqrt(1.000000000000001-sb^2);

beta(:,i) =[atan2d(sb,cb) atan2d(sb,-cb)];

%alpha

sa = r21; ca = r11;

sa1 = r21/cosd(beta(1,i)); ca1 = r11/cosd(beta(1,i));

sa2 = r21/cosd(beta(2,i)); ca2 = r11/cosd(beta(2,i));

alpha(:,i)=[atan2d(sa,ca) atan2d(sa,-ca) atan2d(sa1,ca1) atan2d(sa2,ca2)];

%gamma

sg = r32; cg = r33;

sg1 = r32/cosd(beta(1,i)); cg1 = r33/cosd(beta(1,i));

sg2 = r32/cosd(beta(2,i)); cg2 = r33/cosd(beta(2,i));

gamma(:,i)=[atan2d(sg,cg) atan2d(-sg,-cg) atan2d(sg1,cg1) atan2d(sg2,cg2)];

end

Position=[Px; Py; Pz; alpha; beta; gamma];

## Newton-Euler Recursive formulation

%Outward iteration

for j=1:6

if (j==1) % Prismatic

ang\_v = Rot\_t(:,k:3\*j)\*omega;

ang\_a = Rot\_t(:,k:3\*j)\*omega\_dot;

lin\_a = Rot\_t(:,k:3\*j)\*(cross(omega\_dot,Pos(:,j))+cross(omega,cross(omega,Pos(:,j)))+v\_dot)+cross(2\*omega,Velocity(j,i)\*Z)+Acceleration(j,i)\*Z;

else % Revolute

ang\_v = Rot\_t(:,k:3\*j)\*omega+Velocity(j,i)\*Z;

ang\_a = Rot\_t(:,k:3\*j)\*omega\_dot+cross(Rot\_t(:,k:3\*j)\*omega,Velocity(j,i)\*Z)+(Acceleration(j,i)\*Z);

lin\_a = Rot\_t(:,k:3\*j)\*(cross(omega\_dot,Pos(:,j))+cross(omega,cross(omega,Pos(:,j)))+v\_dot);

end

lin\_a\_COM = cross(ang\_a,P\_G(:,j+1))+cross(ang\_v,cross(ang\_v,P\_G(:,j+1)))+lin\_a;

F\_inertial = mass(j+1)\*lin\_a\_COM;

N\_inertial = Inertia(:,k:3\*j)\*ang\_a+cross(ang\_v,Inertia(:,k:3\*j)\*ang\_v);

omega = ang\_v;

omega\_dot = ang\_a;

v\_dot = lin\_a;

k=k+3;

end

mat\_ang\_v(:,i) = ang\_v;

mat\_ang\_a(:,i) = ang\_a;

mat\_lin\_a(:,i) = lin\_a;

mat\_lin\_a\_COM(:,i) = lin\_a\_COM;

mat\_F\_inertial(:,i) = F\_inertial;

mat\_N\_inertial(:,i) = N\_inertial;

k=1;

% Inward Iteration

for m = 6:-1:1

link\_f = mat\_F\_inertial(:,i)+Rot(:,(3\*m)+1:kk)\*f\_i;

link\_n = mat\_N\_inertial(:,i)+Rot(:,(3\*m)+1:kk)\*m\_i+cross(P\_G(:,m+1),mat\_F\_inertial(:,i))+cross(Pos(:,m+1),Rot(:,(3\*m)+1:kk)\*f\_i);

if (m==1) % Prismatic

tau = link\_f.'\*Z;

else % Revolute

tau = link\_n.'\*Z;

end

f\_i = link\_f;

m\_i = link\_n;

kk = kk - 3;

end

mat\_link\_f(:,i) = link\_f;

mat\_link\_n(:,i) = link\_n;

mat\_tau(:,i) = tau;

kk=21;

## My\_jacobian\_symbolic

%% Parameters

% 1) Link Lengths (mm)

a\_0 = 0; a\_3 = 0;

a\_1 = 0; a\_4 = 0;

a\_2 = 0; a\_5 = 0;

a\_ee = 0;

% 2) Link Twists (deg)

alpha\_0 = 0; alpha\_3 = -90;

alpha\_1 = +90; alpha\_4 = +90;

alpha\_2 = +90; alpha\_5 = +90;

alpha\_ee = 0;

% 3) Link Offsets (mm)

syms d\_1 d\_2 d\_4 d\_ee

d\_3 = 0;

d\_5 = 0;

d\_6 = 0;

% 4) Joint Angles (deg)

syms th\_2 th\_3 th\_4 th\_5 th\_6

th\_1 = 0;

th\_ee = 0;

%% MATRICES

T\_01 = [cos(th\_1) sin(th\_1)\*(-1) 0 a\_0

sin(th\_1)\*cosd(alpha\_0) cos(th\_1)\*cosd(alpha\_0) sind(alpha\_0)\*(-1) d\_1\*sind(alpha\_0)\*(-1)

sin(th\_1)\*sind(alpha\_0) cos(th\_1)\*sind(alpha\_0) cosd(alpha\_0) d\_1\*cosd(alpha\_0)

0 0 0 1];

T\_12 = [cos(th\_2) sin(th\_2)\*(-1) 0 a\_1

sin(th\_2)\*cosd(alpha\_1) cos(th\_2)\*cosd(alpha\_1) sind(alpha\_1)\*(-1) d\_2\*sind(alpha\_1)\*(-1)

sin(th\_2)\*sind(alpha\_1) cos(th\_2)\*sind(alpha\_1) cosd(alpha\_1) d\_2\*cosd(alpha\_1)

0 0 0 1];

T\_23 = [cos(th\_3) sin(th\_3)\*(-1) 0 a\_2

sin(th\_3)\*cosd(alpha\_2) cos(th\_3)\*cosd(alpha\_2) sind(alpha\_2)\*(-1) d\_3\*sind(alpha\_2)\*(-1)

sin(th\_3)\*sind(alpha\_2) cos(th\_3)\*sind(alpha\_2) cosd(alpha\_2) d\_3\*cosd(alpha\_2)

0 0 0 1];

T\_34 = [cos(th\_4) sin(th\_4)\*(-1) 0 a\_3

sin(th\_4)\*cosd(alpha\_3) cos(th\_4)\*cosd(alpha\_3) sind(alpha\_3)\*(-1) d\_4\*sind(alpha\_3)\*(-1)

sin(th\_4)\*sind(alpha\_3) cos(th\_4)\*sind(alpha\_3) cosd(alpha\_3) d\_4\*cosd(alpha\_3)

0 0 0 1];

T\_45 = [cos(th\_5) sin(th\_5)\*(-1) 0 a\_4

sin(th\_5)\*cosd(alpha\_4) cos(th\_5)\*cosd(alpha\_4) sind(alpha\_4)\*(-1) d\_5\*sind(alpha\_4)\*(-1)

sin(th\_5)\*sind(alpha\_4) cos(th\_5)\*sind(alpha\_4) cosd(alpha\_4) d\_5\*cosd(alpha\_4)

0 0 0 1];

T\_56 = [cos(th\_6) sin(th\_6)\*(-1) 0 a\_5

sin(th\_6)\*cosd(alpha\_5) cos(th\_6)\*cosd(alpha\_5) sind(alpha\_5)\*(-1) d\_6\*sind(alpha\_5)\*(-1)

sin(th\_6)\*sind(alpha\_5) cos(th\_6)\*sind(alpha\_5) cosd(alpha\_5) d\_6\*cosd(alpha\_5)

0 0 0 1];

T\_6ee = [cos(th\_ee) sin(th\_ee)\*(-1) 0 a\_ee

sin(th\_ee)\*cosd(alpha\_ee) cos(th\_ee)\*cosd(alpha\_ee) sind(alpha\_ee)\*(-1) d\_ee\*sind(alpha\_ee)\*(-1)

sin(th\_ee)\*sind(alpha\_ee) cos(th\_ee)\*sind(alpha\_ee) cosd(alpha\_ee) d\_ee\*cosd(alpha\_ee)

0 0 0 1];

%% Forward Kinematics

%Position and Rotation matrices of matrices.

R\_01 = T\_01(1:3,1:3); P\_01 = T\_01(1:3,4);

R\_12 = T\_12(1:3,1:3); P\_12 = T\_12(1:3,4);

R\_23 = T\_23(1:3,1:3); P\_23 = T\_23(1:3,4);

R\_34 = T\_34(1:3,1:3); P\_34 = T\_34(1:3,4);

R\_45 = T\_45(1:3,1:3); P\_45 = T\_45(1:3,4);

R\_56 = T\_56(1:3,1:3); P\_56 = T\_56(1:3,4);

R\_6ee = T\_6ee(1:3,1:3); P\_6ee = T\_6ee(1:3,4);

%Homogeneous Tranforms, Position vectors and Rotation matrices of frames.

T\_02 = T\_01\*T\_12; R\_02 = T\_02(1:3,1:3); P\_02 = T\_02(1:3,4);

T\_03 = T\_02\*T\_23; R\_03 = T\_03(1:3,1:3); P\_03 = T\_03(1:3,4);

T\_04 = T\_03\*T\_34; R\_04 = T\_04(1:3,1:3); P\_04 = T\_04(1:3,4);

T\_05 = T\_04\*T\_45; R\_05 = T\_05(1:3,1:3); P\_05 = T\_05(1:3,4);

T\_06 = T\_05\*T\_56; R\_06 = T\_06(1:3,1:3); P\_06 = T\_06(1:3,4);

T\_0ee = T\_06\*T\_6ee; R\_0ee = T\_0ee(1:3,1:3); P\_0ee = T\_0ee(1:3,4);

T\_36 = T\_34\*T\_45\*T\_56; R\_36 = T\_36(1:3,1:3); P\_36 = T\_36(1:3,4);

%% Jacobian (ref\_J\_w) ref = 3, w = 4

% Joint Direction

% Main Arm

R\_33 = eye(3);

Z\_33 = R\_33(:,3);

R\_32 = R\_23.';

Z\_32 = R\_32(:,3);

R\_31 = (R\_12\*R\_23).';

Z\_31 = R\_31(:,3);

%Wrist

Z\_34 = R\_34(:,3);

R\_35 = R\_34\*R\_45;

Z\_35 = R\_35(:,3);

R\_36 = R\_34\*R\_45\*R\_56;

Z\_36 = R\_36(:,3);

% Positions vectors

T\_14 = T\_12\*T\_23\*T\_34;

p\_1w = T\_14(1:3,4);

P\_31w = R\_31\*p\_1w;

T\_24 = T\_23\*T\_34;

p\_2w = T\_24(1:3,4);

P\_32w = R\_32\*p\_2w;

p\_3w = T\_34(1:3,4);

P\_33w = R\_33\*p\_3w;

P\_34w = [0 0 0]';

P\_35w = [0 0 0]';

P\_36w = [0 0 0]';

% Cross Products

e1 = cross(Z\_31,P\_31w);

e2 = cross(Z\_32,P\_32w);

e3 = cross(Z\_33,P\_33w);

e4 = cross(Z\_34,P\_34w);

e5 = cross(Z\_35,P\_35w);

e6 = cross(Z\_36,P\_36w);

% Matrices & Singularities

lin\_vel = [e1 e2 e3 e4 e5 e6];

ang\_vel = [zeros(3,1) Z\_32 Z\_33 Z\_34 Z\_35 Z\_36];

J\_3w = simplify([lin\_vel; ang\_vel]);

B = simplify(J\_3w(4:6,1:3));

B\_det = simplify(det(B));

zero\_J = simplify(J\_3w(1:3,4:6));

zero\_J\_det = det(zero\_J);

A = simplify(J\_3w(1:3,1:3));

A\_det = simplify(det(A));

C = simplify(J\_3w(4:6,4:6));

C\_det = simplify(det(C));

J\_det = simplify(A\_det\*C\_det);

inv\_J\_3w = simplify([inv(A) zeros(3); -inv(C)\*B\*inv(A) inv(C)]);

%% Velocity Transformation Matrix

%Position Vector P ref\_w->ee

R\_06 = T\_06(1:3,1:3);

P\_6ee = T\_6ee(1:3,4);

P\_0eew = -R\_06\*P\_6ee;

skew1 = [0 -P\_0eew(3,1) P\_0eew(2,1)

P\_0eew(3,1) 0 -P\_0eew(1,1)

-P\_0eew(2,1) P\_0eew(1,1) 0];

Tv = simplify([R\_03 skew1\*R\_03; zeros(3) R\_03])

%% Forward/Inverse Velocity Equations

%Forward Velocity

syms q1 q2 q3 q4 q5 q6

q = [q1 q2 q3 q4 q5 q6];

vel\_0ee = simplify(Tv\*J\_3w\*q.')

%Inverse Velocity

syms v1 v2 v3 v4 v5 v6

vel = [v1 v2 v3 v4 v5 v6];

q\_dot = simplify(inv(Tv)\*inv\_J\_3w\*vel.')

%% Force Transformation Matrix

R\_30 = R\_03.';

P\_3eew = R\_36\*P\_6ee;

skew2 = [ 0 -P\_3eew(3,1) P\_3eew(2,1)

P\_3eew(3,1) 0 -P\_3eew(1,1)

-P\_3eew(2,1) P\_3eew(1,1) 0];

Fv = simplify([R\_30 zeros(3); skew2\*R\_30 R\_30])

%% Inverse Static Force

syms f1 f2 f3 m1 m2 m3

f = [f1 f2 f3 m1 m2 m3];

J\_3wt = transpose(J\_3w);

torque = simplify(J\_3wt\*Fv\*f.')

## Inertia\_tensor

function [mass,P\_G,Inertia] = inertia\_tensor

%% Parameters (units grams and milimeter)

material\_density = 0.0027;

u1 = 0.001;

u2 = 1.0\*10^-6;

u3 = 1.0\*10^-9;

%Base

mass\_b = 392900.625;

P\_gb = [0.000 -2.9020 -213.1061]';

I\_bxx = 61017762234.37; I\_bxy = 0.00; I\_bxz = 0.00;

I\_byy = 64782384916.99; I\_byz = -513090703.13;

I\_bzz = 18153025901.37;

I\_base = [I\_bxx -I\_bxy -I\_bxz

-I\_bxy I\_byy -I\_byz

-I\_bxz -I\_byz I\_bzz];

% Joint 1

mass\_1 = 24585.53;

P\_g1 = [-0.32 11.61 0.00]';

I\_1xx = 255250944.45; I\_1xy = -416800.98; I\_1xz = 0.00;

I\_1yy = 291346050.24; I\_1yz = 0.00;

I\_1zz = 147359449.68;

I\_1 = [I\_1xx -I\_1xy -I\_1xz

-I\_1xy I\_1yy -I\_1yz

-I\_1xz -I\_1yz I\_1zz];

% Joint 2

mass\_2 = 13733.91;

P\_g2 = [0.03 0.06 429.46]';

I\_2xx = 3376732460.01; I\_2xy = 17055.59; I\_2xz = 373203.14;

I\_2yy = 33775621925.85; I\_2yz = 637156.83;

I\_2zz = 13761348.21;

I\_2 = [I\_2xx -I\_2xy -I\_2xz

-I\_2xy I\_2yy -I\_2yz

-I\_2xz -I\_2yz I\_2zz];

% Joint 3

mass\_3 = 11702.10;

P\_g3 = [0.04 -262.31 -0.76]';

I\_3xx = 1484193151.37; I\_3xy = -8768.99; I\_3xz = -38887.17;

I\_3yy = 23920268.41; I\_3yz = -427293.99;

I\_3zz = 1470615861.95;

I\_3 = [I\_3xx -I\_3xy -I\_3xz

-I\_3xy I\_3yy -I\_3yz

-I\_3xz -I\_3yz I\_3zz];

% Joint 4

mass\_4 = 4283.77;

P\_g4 = [0.00 0.07 92.17]';

I\_4xx = 78921407.33; I\_4xy = -1.73; I\_4xz = 3.94;

I\_4yy = 73833198.42; I\_4yz = 73237.31;

I\_4zz = 9922402.27;

I\_4 = [I\_4xx -I\_4xy -I\_4xz

-I\_4xy I\_4yy -I\_4yz

-I\_4xz -I\_4yz I\_4zz];

% Joint 5

mass\_5 = 4766.31;

P\_g5 = [-0.10 -44.46 2.02]';

I\_5xx = 44058687.33; I\_5xy = -5173.52; I\_5xz = -41024.40;

I\_5yy = 12304280.63; I\_5yz = 63021.21;

I\_5zz = 36389912.58;

I\_5 = [I\_5xx -I\_5xy -I\_5xz

-I\_5xy I\_5yy -I\_5yz

-I\_5xz -I\_5yz I\_5zz];

% Joint 6

mass\_6 = 5901.84;

P\_g6 = [0.00 0.06 445.50]';

I\_6xx = 1286844631.76; I\_6xy = 0.00; I\_6xz = 0.00;

I\_6yy = 1286855956.51; I\_6yz = 78000.83;

I\_6zz = 4397544.68;

I\_6 = [I\_6xx -I\_6xy -I\_6xz

-I\_6xy I\_6yy -I\_6yz

-I\_6xz -I\_6yz I\_6zz];

% End-Effector

mass\_ee = 2922.39;

P\_gee = [-0.32 0.52 -94.50]';

I\_eexx = 30809463.03; I\_eexy = -427.46; I\_eexz = 41553.77;

I\_eeyy = 126675054.21; I\_eeyz = -167410.43;

I\_eezz = 96690996.71;

I\_ee = [I\_eexx -I\_eexy -I\_eexz

-I\_eexy I\_eeyy -I\_eeyz

-I\_eexz -I\_eeyz I\_eezz];

% Matrix

mass = [mass\_b, mass\_1, mass\_2, mass\_3, mass\_4, mass\_5, mass\_6, mass\_ee];%\*u1;

P\_G = [P\_gb, P\_g1, P\_g2, P\_g3, P\_g4, P\_g5, P\_g6, P\_gee];%\*u2;

Inertia = [I\_base, I\_1, I\_2, I\_3, I\_4, I\_5, I\_6, I\_ee];%\*u3;

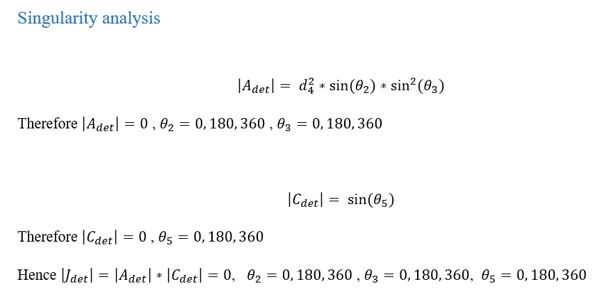
I have just noticed a typo in my *my\_jacobian  & my\_jacobian\_symbolic functions.*

My joint 1 is prismatic so my e1 should be Z\_31, instead of cross(Z\_31,P\_31w). It my mistake that I was being careless whe I was copying all of it, I still have consider zeros(3,1) in ang\_vel for my prismatic joint.  
Below are the attached snapshots



And for my singularity analysis, it still depend on th2,th3 and th5, however th2 is a function of sine instead of cosine. Therefore its limit should be 0,180, and 360.

I have add the fixed snapshot of that below. You can compare it with my report and you will see th2 a function of cosine.



When you execute the computation of *my\_jacobian* & *my\_jacobian\_symbolic*, you may please take e1=Z\_31 , **instead** of e1=cross(Z\_31,P\_31) and it will fix it all.

I haven’t include the force transformation and velocity transformation matrix in my report since they are very large to copy even after being simplified.

Its a humble request to kindly consider it without applying a penalty.

If you want I can change that line in both the functions in the code and resend it to you?

The outward and inward iteration is independent of jacobian analysis.

Otherwise the rest is hopefully ok. Thanks

​Best Regards,

**Syed Rizvi**  
Student, Mechatronic Systems Engineering | Bachelors of Applied Science  
Simon Fraser University | Surrey Campus  
250-13450 102 Ave, Surrey, BC V3T 0A3  
M: 778.968.4522

[](http://trk.cp20.com/click/d9xp-1izw33-ldatnj-8tl984x9/)