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

Table of Figures

Table of Tables

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.

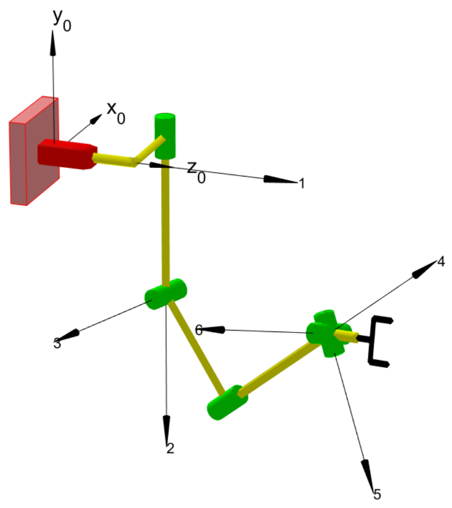


Figure 2 MATLAB Layout

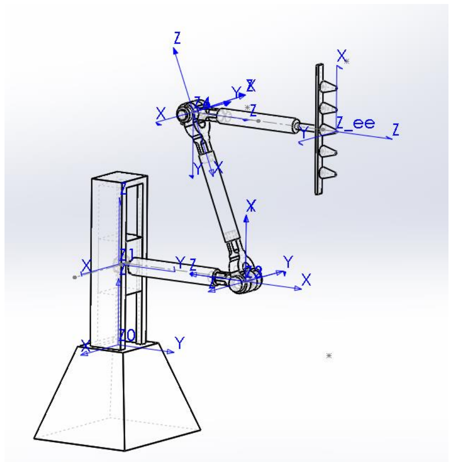


Figure 1 SolidWorks Layout

Design Specification

Joint	Type	Constraints
1	Prismatic	$125\ mm < P_1 < 775\ mm$
2	Revolute	$90 < \theta_2 < 270$
3	Revolute	$45 < \theta_3 < 225$
4	Revolute	$0 < \theta_4 < 120$
5	Revolute	$0 < \theta_5 < 360$
6	Revolute	$0 < \theta_6 < 360$

Table 1 Joints Information

D-H Parameter

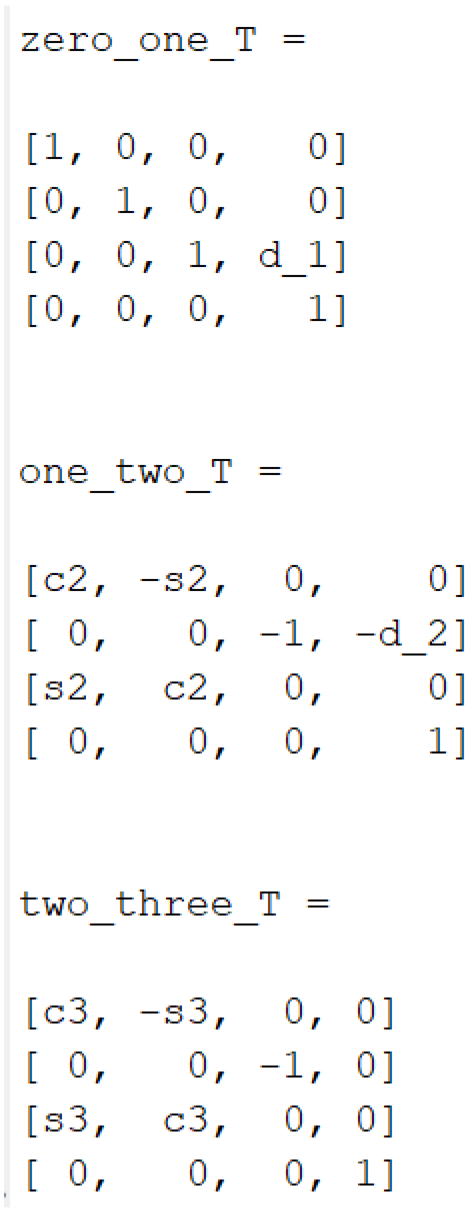
$i-1$	α_{i-1}	a_{i-1}	d_i	θ_i	i
0	0	0	d_1	0	1

1	+90	0	838.5mm	θ_2	2
2	+90	0	0	θ_3	3
3	-90	0	972mm	θ_4	4
4	-90	0	0	θ_5	5
5	+90	0	0	θ_6	6
6	0	0	945 mm	0	ee

Table 2 Denavit-Hartenberg Parameters

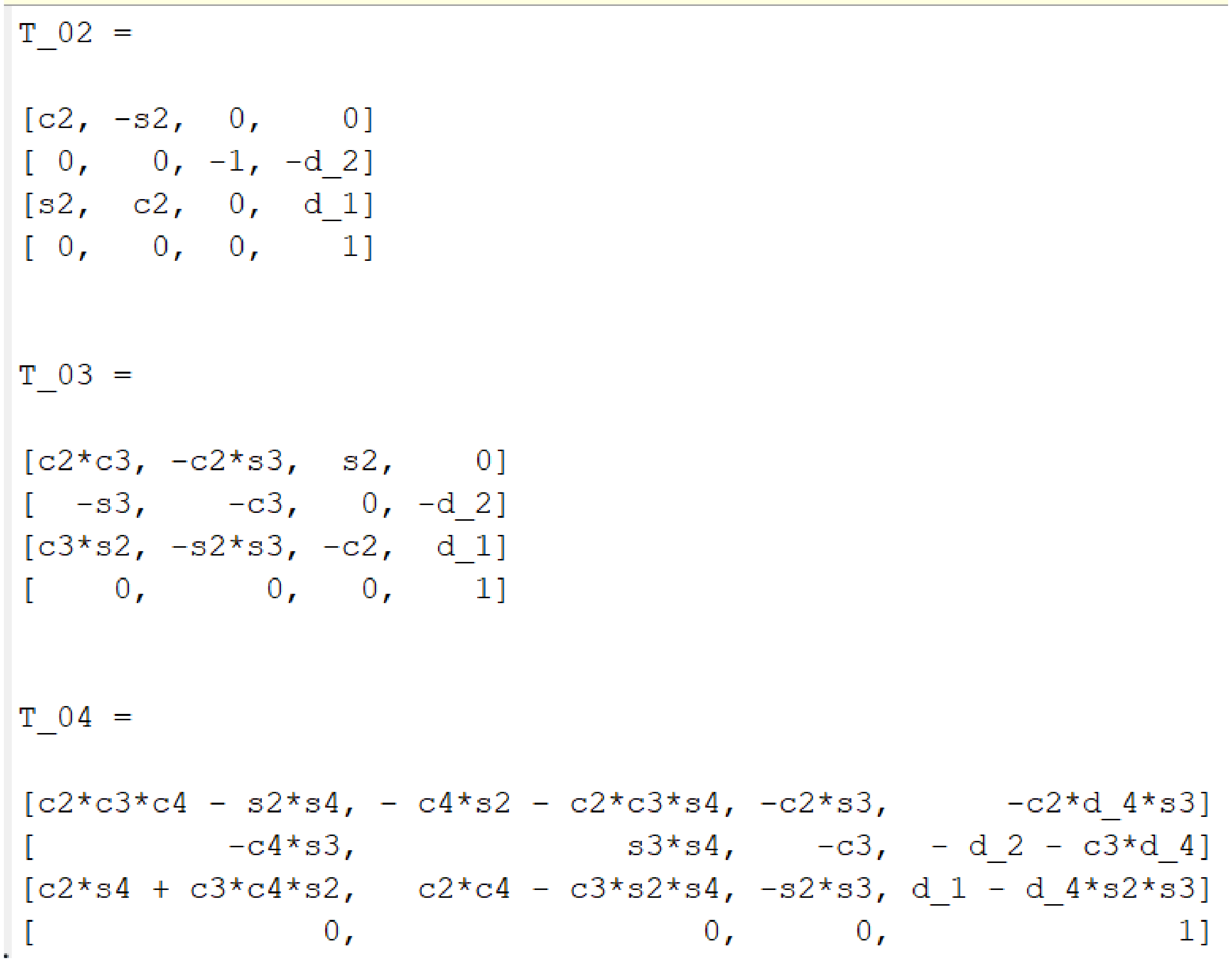
Inverse Kinematics

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



```
zero_one_T =

[1, 0, 0,   0]
[0, 1, 0,   0]
[0, 0, 1, d_1]
[0, 0, 0,   1]


one_two_T =

[c2, -s2,  0,    0]
[ 0,   0, -1, -d_2]
[s2,  c2,  0,    0]
[ 0,   0,  0,    1]


two_three_T =

[c3, -s3,  0, 0]
[ 0,   0, -1, 0]
[s3,  c3,  0, 0]
[ 0,   0,  0, 1]
```

```
three_four_T =

[ c4, -s4, 0,   0]
[  0,   0, 1, d_4]
[-s4, -c4, 0,   0]
[  0,   0, 0,   1]


four_five_T =

[c5, -s5,  0, 0]
[ 0,   0, -1, 0]
[s5,  c5,  0, 0]
[ 0,   0,  0, 1]


five_six_T =

[ c6, -s6, 0, 0]
[  0,   0, 1, 0]
[-s6, -c6, 0, 0]
[  0,   0, 0, 1]
```

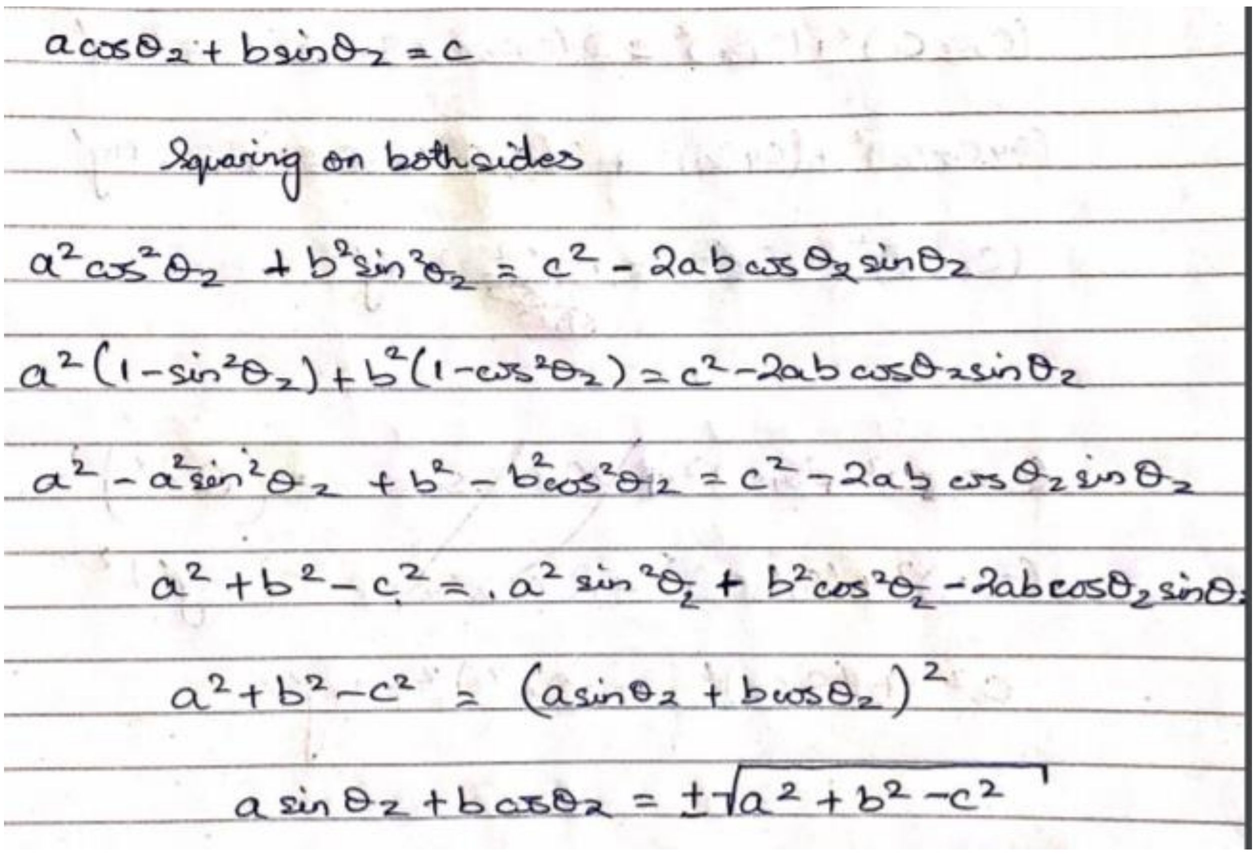


```
T_02 =

[c2, -s2,  0,    0]
[ 0,   0, -1, -d_2]
[s2,  c2,  0,  d_1]
[ 0,   0,  0,    1]


T_03 =

[c2*c3, -c2*s3,  s2,    0]
[  -s3,    -c3,   0, -d_2]
[c3*s2, -s2*s3, -c2,  d_1]
[    0,      0,   0,    1]


T_04 =

[c2*c3*c4 - s2*s4, - c4*s2 - c2*c3*s4, -c2*s3,       -c2*d_4*s3]
[          -c4*s3,              s3*s4,    -c3,   - d_2 - c3*d_4]
[c2*s4 + c3*c4*s2,   c2*c4 - c3*s2*s4, -s2*s3, d_1 - d_4*s2*s3]
[               0,                  0,      0,                1]
```

$$ {}^{3}_{6}T = \begin{bmatrix} C_2C_3 & -C_2S_3 & S_2 & d_3S_2 \\ -S_3 & -C_3 & 0 & -d_2 \\ S_2C_3 & -S_2S_3 & -C_2 & d_1-d_2C_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_4C_5C_6-S_4S_6 & -C_4C_5S_6-S_4C_6 & -S_5C_4 & -d_6S_5C_4 \\ S_4C_5C_6+C_4S_6 & -S_4S_6C_5+C_4C_6 & -S_4S_5 & -d_6S_4S_5 \\ S_5C_6 & -S_5S_6 & C_5 & d_6C_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ T = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$



$a\cos\theta_2 + b\sin\theta_2 = c$

Squaring on bothsides

$a^2\cos^2\theta_2 + b^2\sin^2\theta_2 = c^2 - 2ab\cos\theta_2\sin\theta_2$

$a^2(1-\sin^2\theta_2) + b^2(1-\cos^2\theta_2) = c^2 - 2ab\cos\theta_2\sin\theta_2$

$a^2 - a^2\sin^2\theta_2 + b^2 - b^2\cos^2\theta_2 = c^2 - 2ab\cos\theta_2\sin\theta_2$

$a^2 + b^2 - c^2 = a^2\sin^2\theta_2 + b^2\cos^2\theta_2 - 2ab\cos\theta_2\sin\theta$

$a^2 + b^2 - c^2 = (a\sin\theta_2 + b\cos\theta_2)^2$

$a\sin\theta_2 + b\cos\theta_2 = \pm\sqrt{a^2 + b^2 - c^2}$

$$\begin{bmatrix} P_x' \\ P_y' \\ P_z' \end{bmatrix} = \begin{bmatrix} c_2 d_4 s_3 \\ -d_2 - c_3 d_4 \\ d_1 - d_4 s_2 s_3 \end{bmatrix}$$

$$P_y' = -d_2 - c_3 d_4 \quad \rightarrow (1)$$

$$\therefore \quad c_3 = \frac{-(P_y' + d_2)}{d_4}$$

$$\star\ \theta_3 = \operatorname{atan2}\left(\pm\sqrt{1 - \frac{(P_y'+d_2)^2}{d_4^2}},\ \frac{-(P_y'+d_2)}{d_4}\right)$$

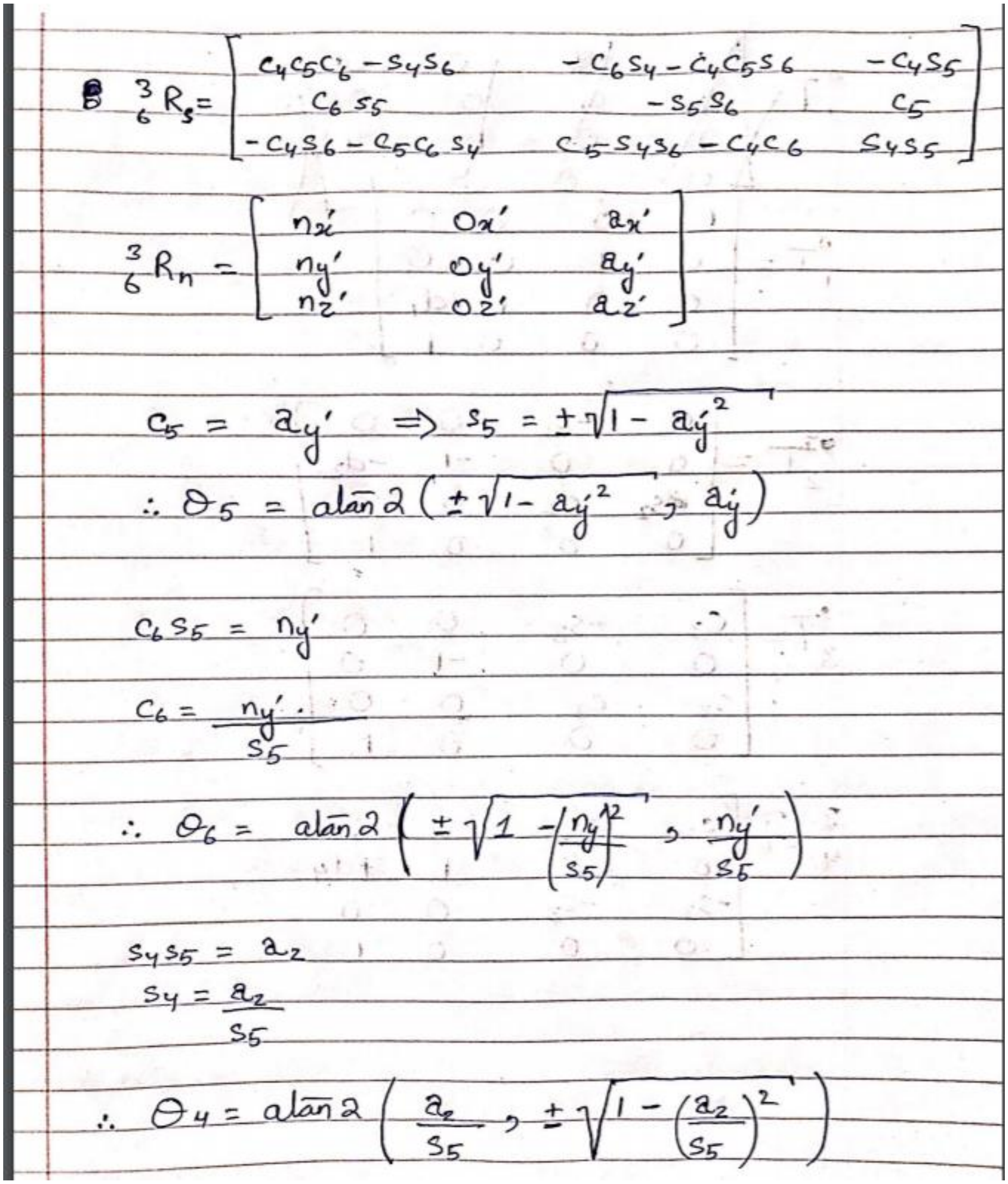
$$P_x' = c_2 d_4 s_3 \longrightarrow (2)$$

$$\therefore \quad c_2 = \frac{P_x'}{d_4 s_3}$$

$$\star\ \theta_2 = \operatorname{atan2}\left(\pm\sqrt{1 - \frac{(P_x')^2}{(d_4 s_3)^2}},\ \frac{P_x'}{d_4 s_3}\right)$$

$$P_z' = d_1 - d_4 s_2 s_3 \longrightarrow (3)$$

$$\star\ d_1 = P_z' + d_4 s_2 s_3$$



$${}^{3}_{6}R_{5} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_6s_4 - c_4c_5s_6 & -c_4s_5 \\ c_6s_5 & -s_5s_6 & c_5 \\ -c_4s_6 - c_5c_6s_4 & c_5s_4s_6 - c_4c_6 & s_4s_5 \end{bmatrix}$$

$${}^{3}_{6}R_{n} = \begin{bmatrix} n_x' & o_x' & a_x' \\ n_y' & o_y' & a_y' \\ n_z' & o_z' & a_z' \end{bmatrix}$$

$$c_5 = a_y' \Rightarrow s_5 = \pm\sqrt{1 - a_y'^2}$$

$$\therefore \theta_5 = \operatorname{atan2}\left(\pm\sqrt{1 - a_y'^2}, a_y'\right)$$

$$c_6 s_5 = n_y'$$

$$c_6 = \frac{n_y'}{s_5}$$

$$\therefore \theta_6 = \operatorname{atan2}\left(\pm\sqrt{1 - \left(\frac{n_y'}{s_5}\right)^2}, \frac{n_y'}{s_5}\right)$$

$$s_4 s_5 = a_z$$

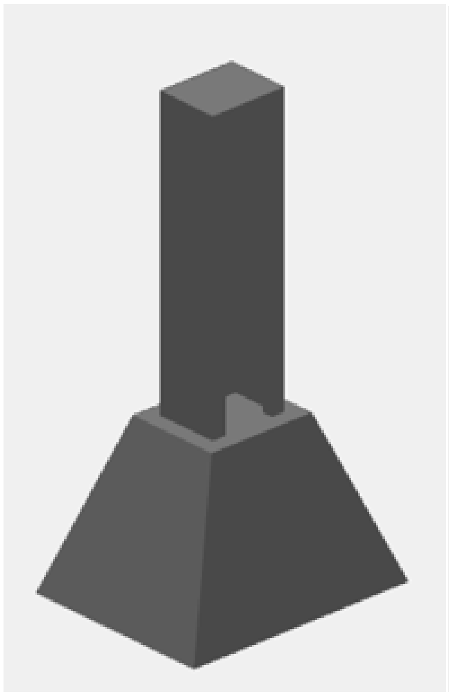
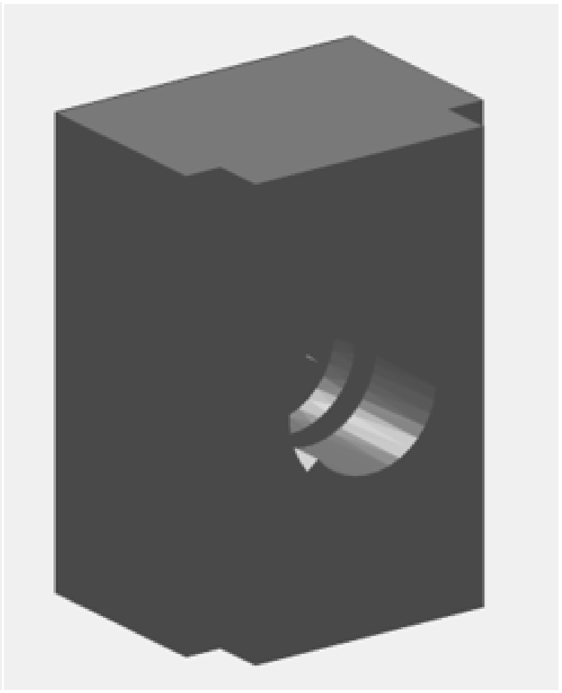
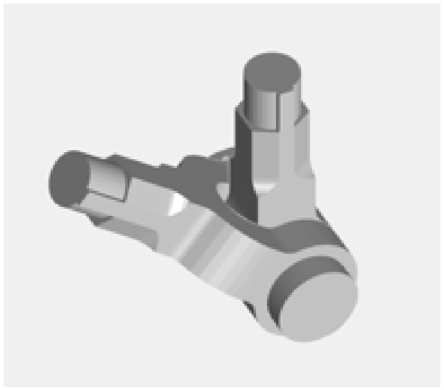
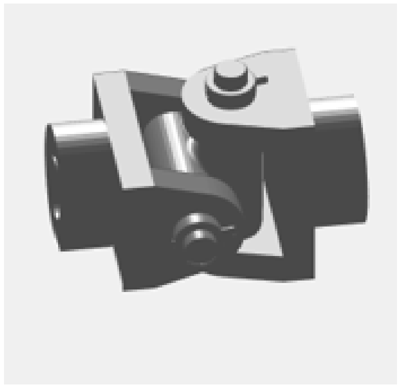
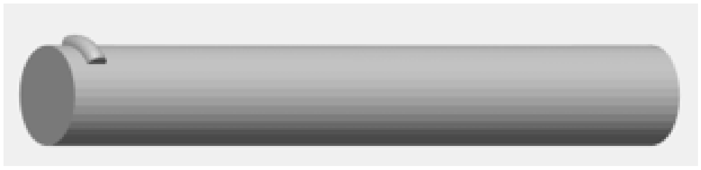
$$s_4 = \frac{a_z}{s_5}$$

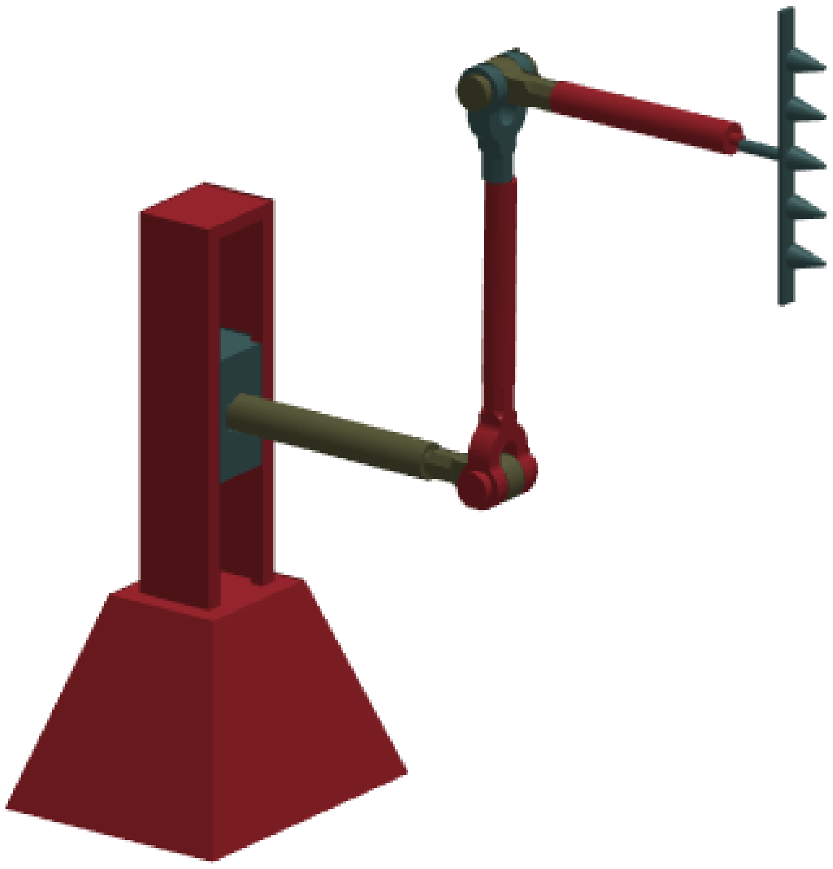
$$\therefore \theta_4 = \operatorname{atan2}\left(\frac{a_z}{s_5}, \pm\sqrt{1 - \left(\frac{a_z}{s_5}\right)^2}\right)$$

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.

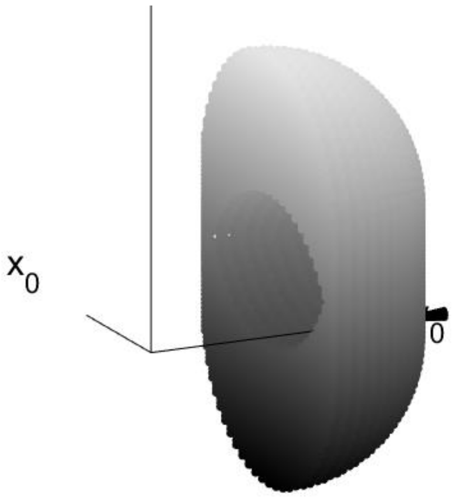


The picture above shows the kinematic reconstruction of the manipulator.

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.

$$plot_n = d_2 + d_4 * \cos(\theta_n)$$

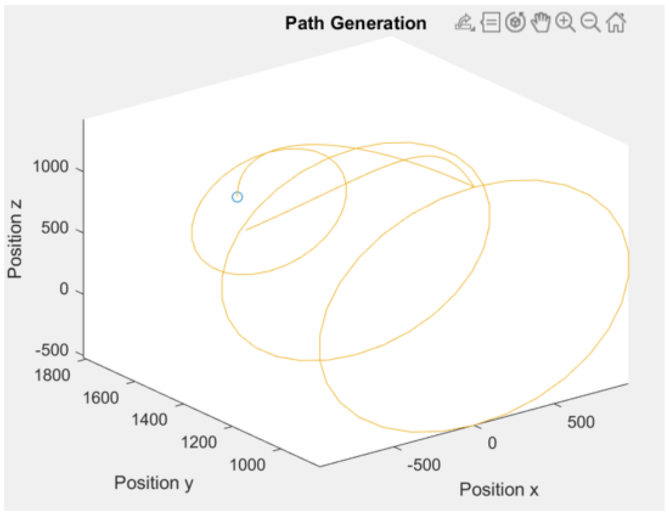
- The 1st set of 5 circles are plot at n=1, $\theta_1 = 90$ Plot = 838.5+972*cos (90) =838.5 mm
- The 2nd set of 5 circles are plot at n=2, $\theta_2 = 60$ Plot = 838.5+972*cos (60) =1324.5 mm
- The 3rd set of 5 circles are plot at n=3, $\theta_3 = 30$ Plot = 838.5+972*cos (90) =1680.3 mm
- The 4th set of 5 circles are plot at n=4, $\theta_4 = 0$ Plot = 838.5+972*cos (90) =1810.5 mm

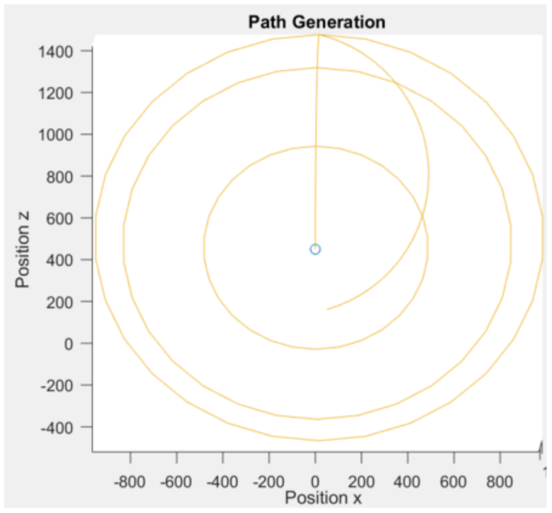
By using the function *my_path* & *P_xyz_abg*, the values of x, y, z, and alpha, beta, and gamma.

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

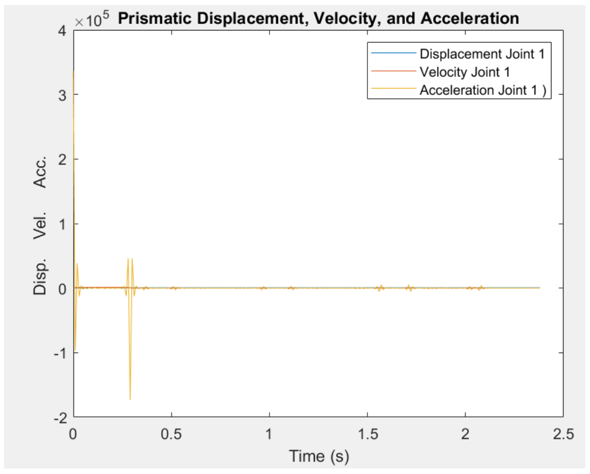
The following figure show the trajectory generation of the manipulator.

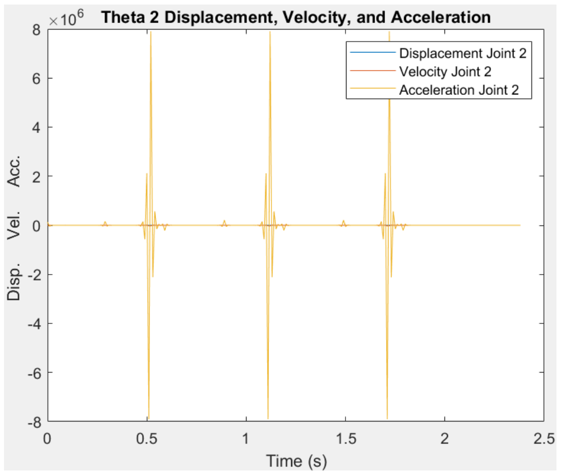


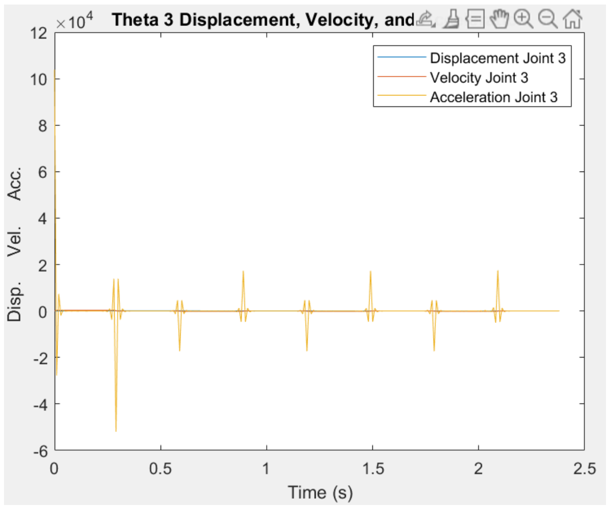


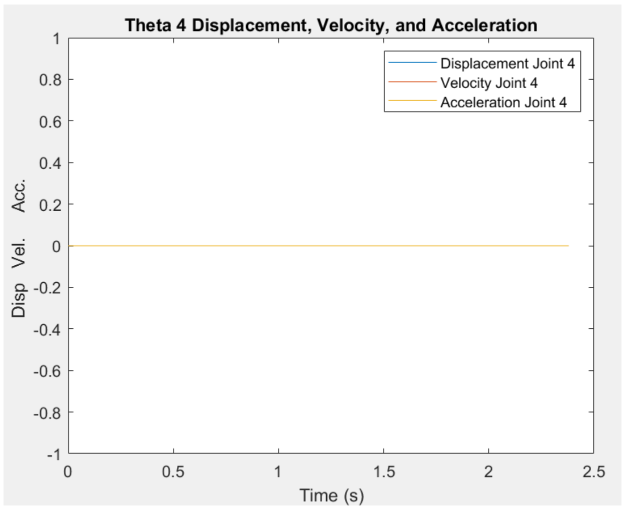
The above figure shows 4 circles (5 circles in each circle) in 3d. The XZ plane shows the 4 circles with the blue dot being the 4th one.

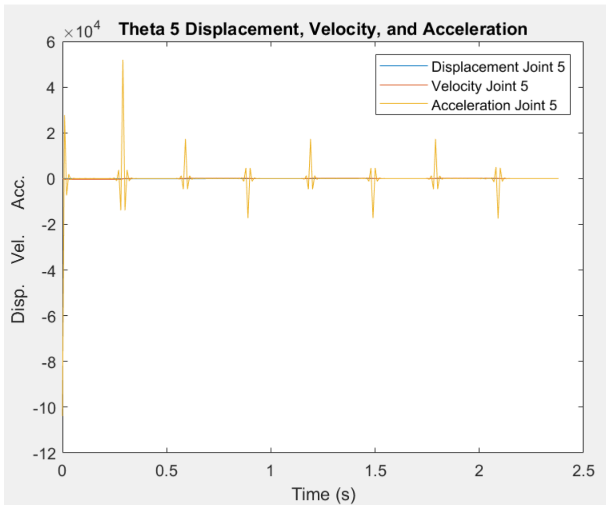
Below are the plots for each joints displacement, velocity and acceleration.

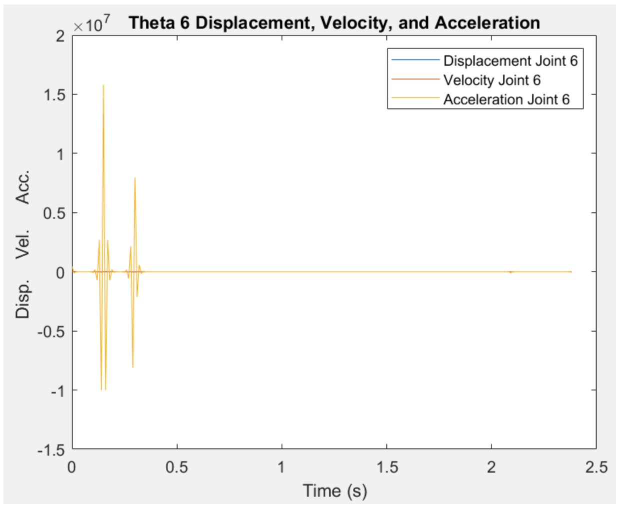


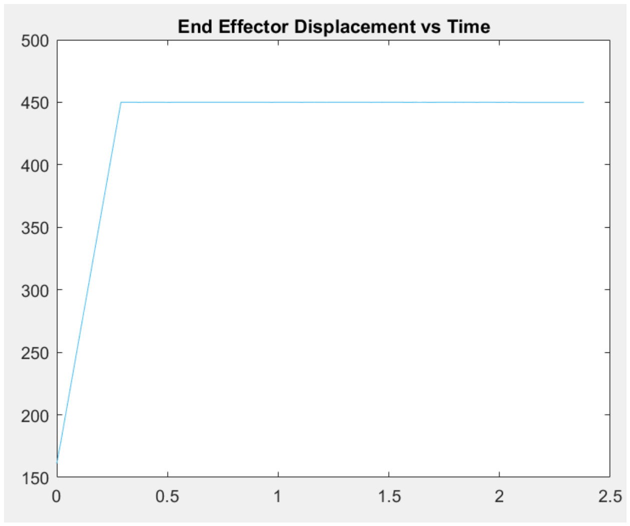


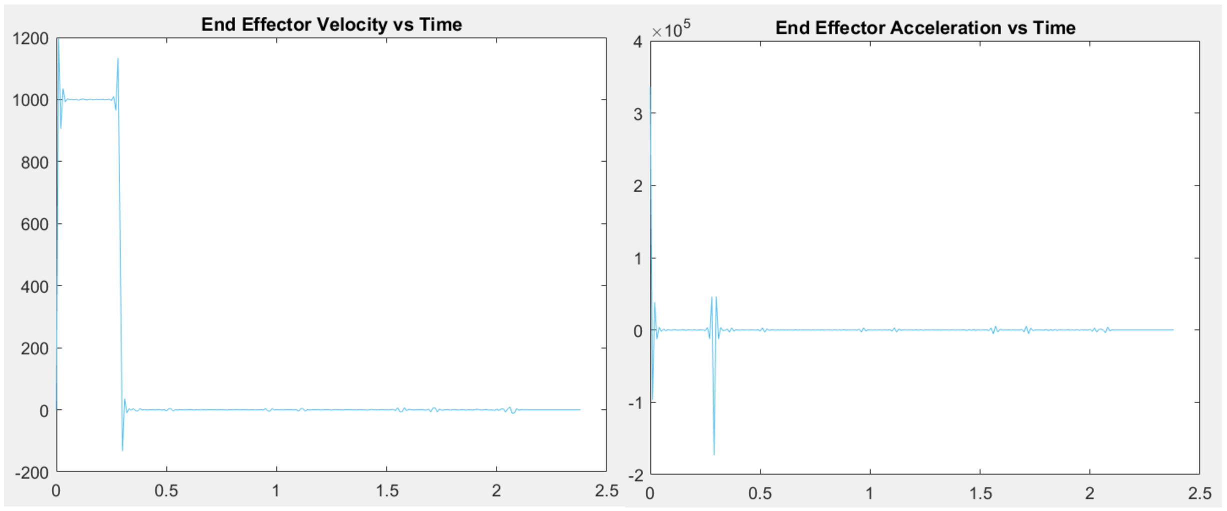

Theta 3 Displacement, Velocity, and
×10^4
Displacement Joint 3
Velocity Joint 3
Acceleration Joint 3
Disp. Vel. Acc.
Time (s)


Theta 4 Displacement, Velocity, and Acceleration
Displacement Joint 4
Velocity Joint 4
Acceleration Joint 4
Disp Vel. Acc.
Time (s)

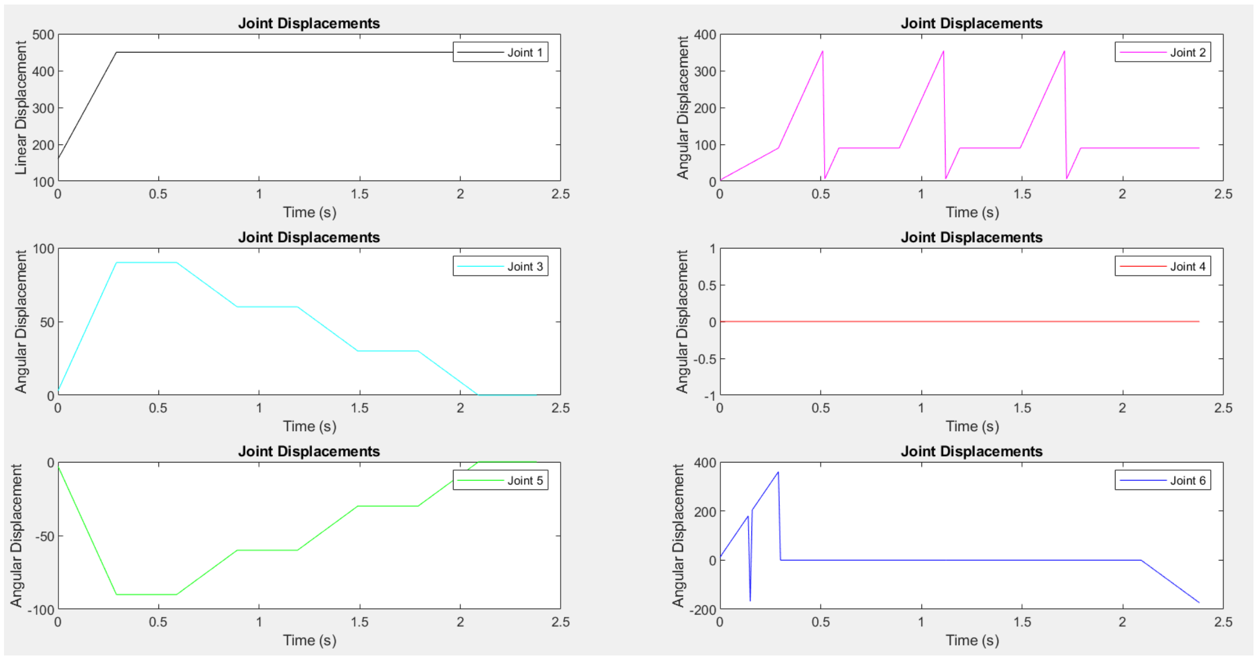

Theta 5 Displacement, Velocity, and Acceleration
×10^4
Displacement Joint 5
Velocity Joint 5
Acceleration Joint 5
Disp. Vel. Acc.
Time (s)


Theta 6 Displacement, Velocity, and Acceleration
×10^7
Displacement Joint 6
Velocity Joint 6
Acceleration Joint 6
Disp. Vel. Acc.
Time (s)

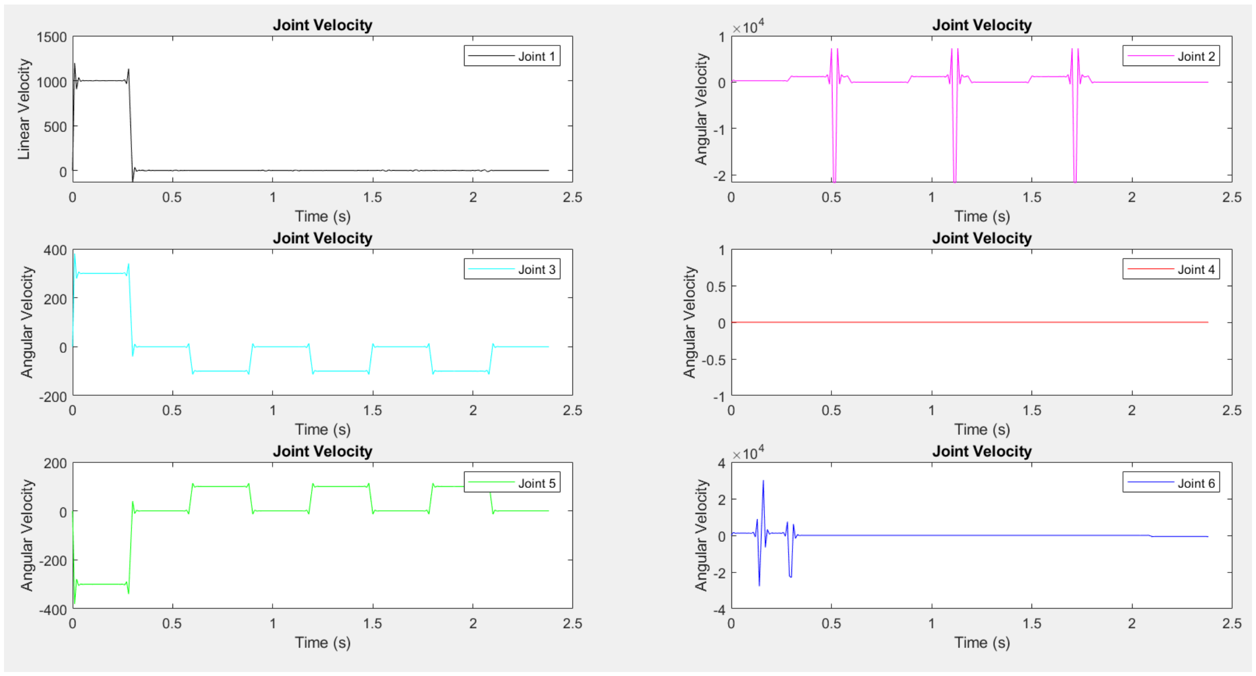

End Effector Displacement vs Time



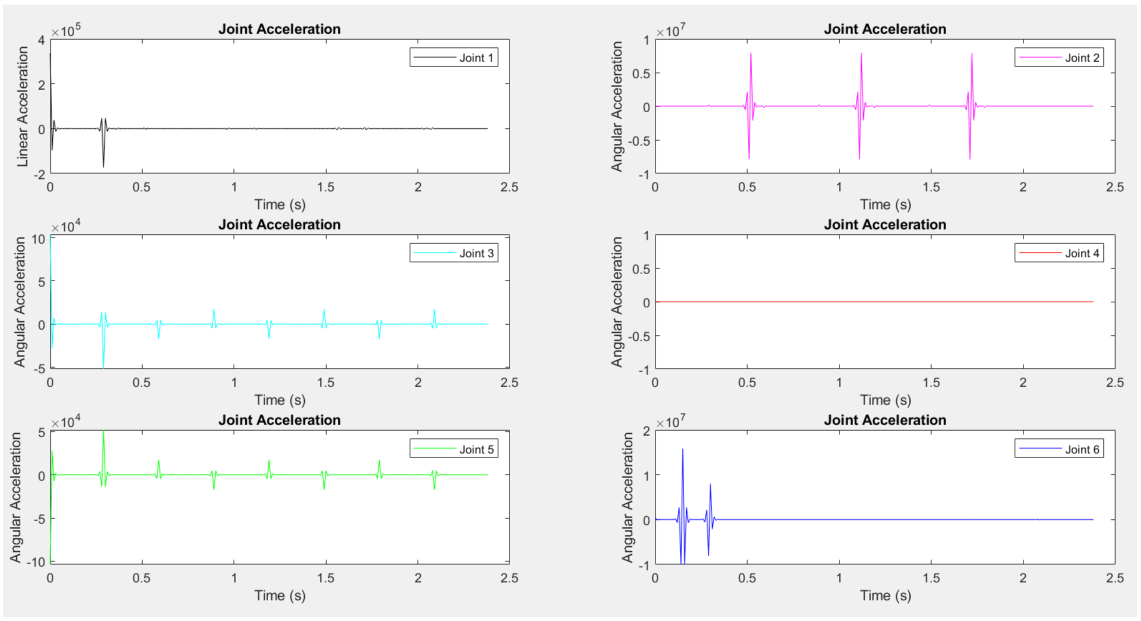
The below plot shows all the joint displacement all together.



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.

```
A =

[cos(th_2)*(d_4 + d_2*cos(th_3)),             0, -d_4]
[       -d_2*cos(th_2)*sin(th_3),             0,    0]
[sin(th_2)*(d_2 + d_4*cos(th_3)), d_4*sin(th_3),    0]


A_det =

d_2*d_4^2*cos(th_2)*sin(th_3)^2
```

```
C =

[0, sin(th_4),  cos(th_4)*sin(th_5)]
[1,         0,          -cos(th_5)]
[0, cos(th_4), -sin(th_4)*sin(th_5)]


C_det =

sin(th_5)
```

```
J_3w =

[cos(th_2)*(d_4 + d_2*cos(th_3)),             0, -d_4, 0,         0,                    0]
[       -d_2*cos(th_2)*sin(th_3),             0,    0, 0,         0,                    0]
[sin(th_2)*(d_2 + d_4*cos(th_3)), d_4*sin(th_3),    0, 0,         0,                    0]
[                              0,     sin(th_3),    0, 0, sin(th_4),  cos(th_4)*sin(th_5)]
[                              0,     cos(th_3),    0, 1,         0,           -cos(th_5)]
[                              0,             0,    1, 0, cos(th_4), -sin(th_4)*sin(th_5)]
```

```
J_det =

d_2*d_4^2*cos(th_2)*sin(th_3)^2*sin(th_5)
```

Singularity analysis

$$|A_{det}| = d_2 * d_4^2 * \cos(\theta_2) * \sin^2(\theta_3)$$

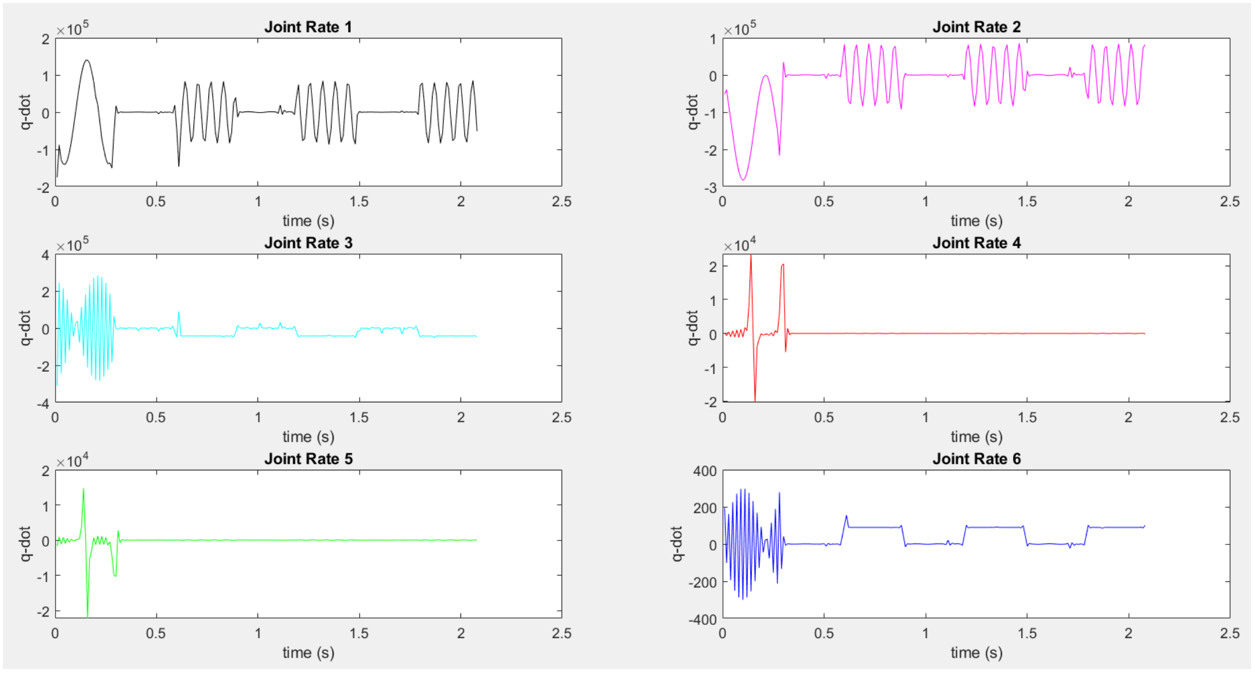
Therefore $|A_{det}| = 0$, $\theta_2 = 90 , 270$, $\theta_3 = 0, 180, 360$

$$|C_{det}| = \sin(\theta_5)$$

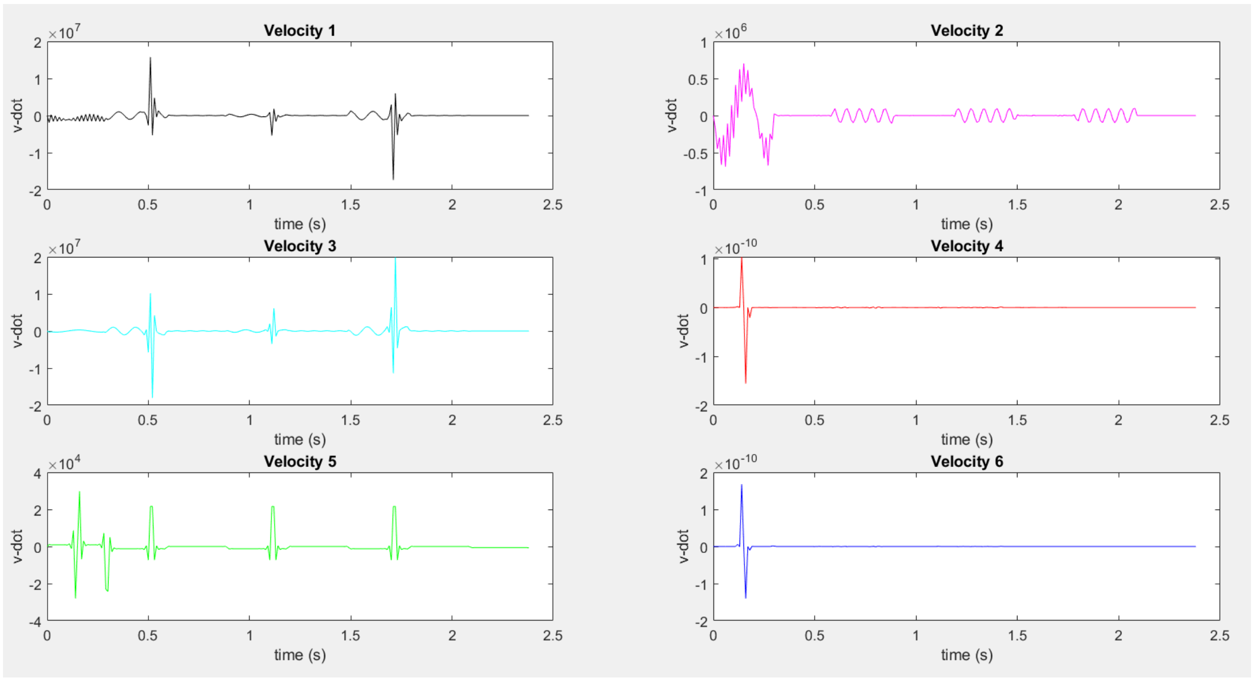
Therefore $|C_{det}| = 0$, $\theta_5 = 0, 180, 360$

Hence $|J_{det}| = |A_{det}| * |C_{det}| = 0, \quad \theta_2 = 90 , 270 , \theta_3 = 0, 180, 360, \;\; \theta_5 = 0, 180, 360$

The below plot is for inverse velocity.



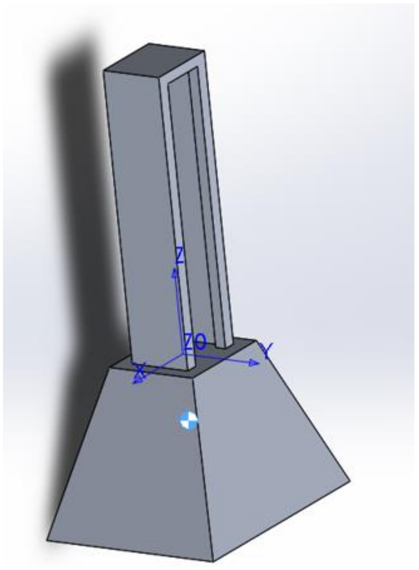
The below plot is for forward velocity.

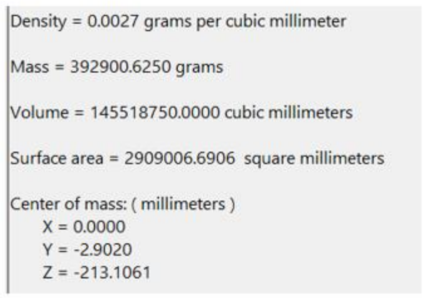


Link Modeling

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

Base





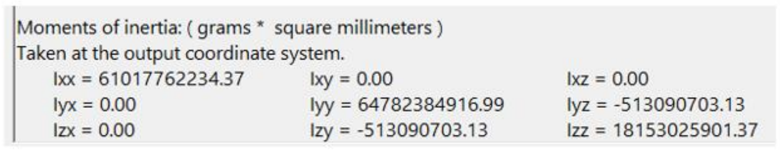
Density = 0.0027 grams per cubic millimeter

Mass = 392900.6250 grams

Volume = 145518750.0000 cubic millimeters

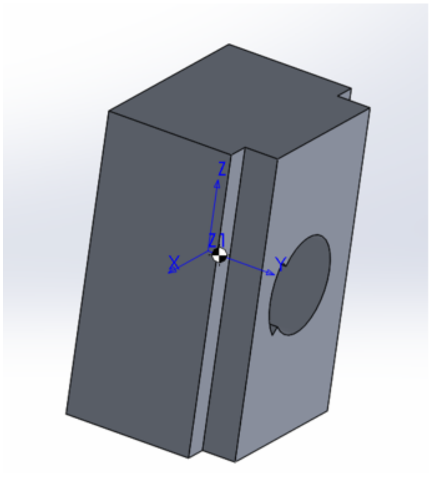
Surface area = 2909006.6906 square millimeters

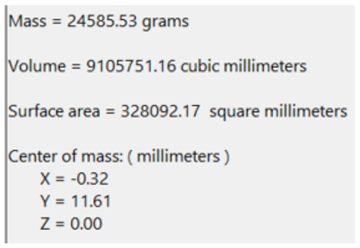
Center of mass: (millimeters)
X = 0.0000
Y = -2.9020
Z = -213.1061

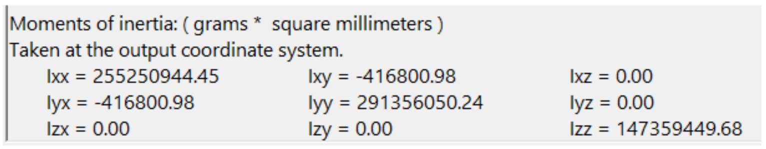


```
Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system.
    Ixx = 61017762234.37        Ixy = 0.00                Ixz = 0.00
    Iyx = 0.00                  Iyy = 64782384916.99      Iyz = -513090703.13
    Izx = 0.00                  Izy = -513090703.13       Izz = 18153025901.37
```

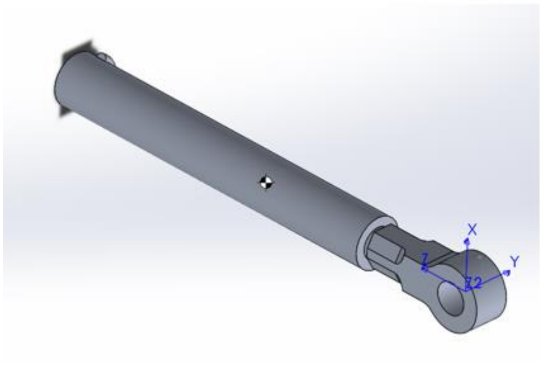
Prismatic Joint

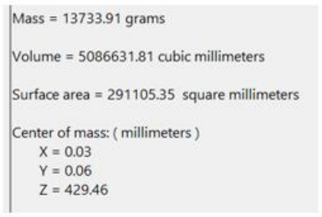


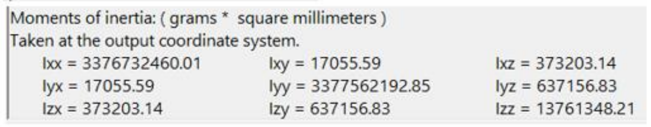

```
Mass = 24585.53 grams

Volume = 9105751.16 cubic millimeters

Surface area = 328092.17  square millimeters

Center of mass: ( millimeters )
     X = -0.32
     Y = 11.61
     Z = 0.00
```


```
Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system.
     Ixx = 255250944.45          Ixy = -416800.98           Ixz = 0.00
     Iyx = -416800.98            Iyy = 291356050.24         Iyz = 0.00
     Izx = 0.00                  Izy = 0.00                 Izz = 147359449.68
```

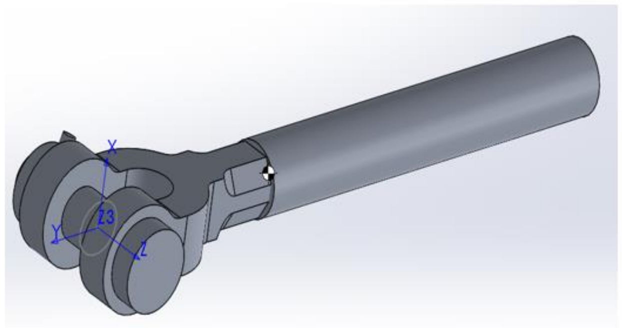
Revolute Joint 2

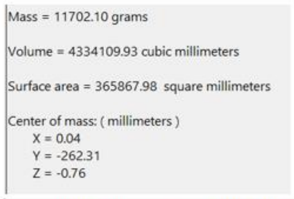


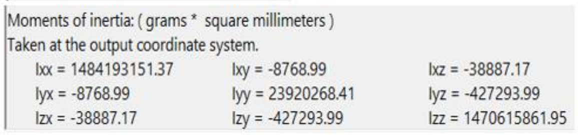

```
Mass = 13733.91 grams

Volume = 5086631.81 cubic millimeters

Surface area = 291105.35  square millimeters

Center of mass: ( millimeters )
     X = 0.03
     Y = 0.06
     Z = 429.46
```


```
Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system.
     Ixx = 3376732460.01         Ixy = 17055.59             Ixz = 373203.14
     Iyx = 17055.59              Iyy = 3377562192.85        Iyz = 637156.83
     Izx = 373203.14             Izy = 637156.83            Izz = 13761348.21
```

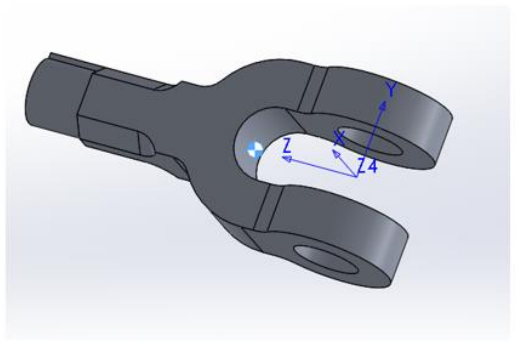
Revolute Joint 3

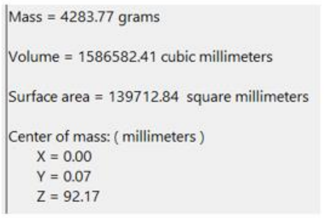



```
Mass = 11702.10 grams

Volume = 4334109.93 cubic millimeters

Surface area = 365867.98  square millimeters

Center of mass: ( millimeters )
     X = 0.04
     Y = -262.31
     Z = -0.76
```

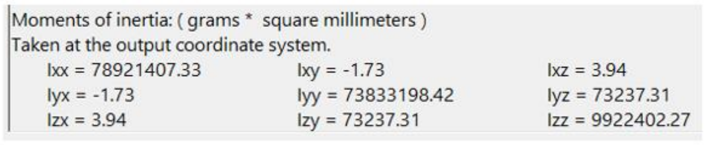

```
Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system.
     Ixx = 1484193151.37         Ixy = -8768.99             Ixz = -38887.17
     Iyx = -8768.99              Iyy = 23920268.41          Iyz = -427293.99
     Izx = -38887.17             Izy = -427293.99           Izz = 1470615861.95
```

Revolute Joint 4



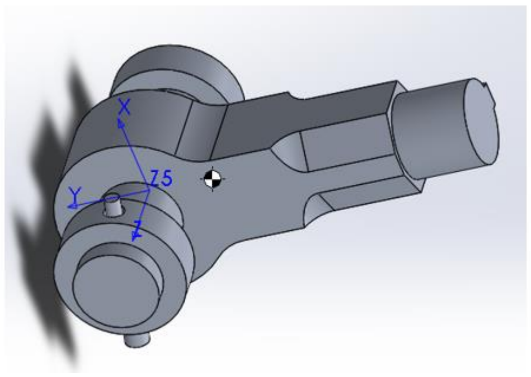


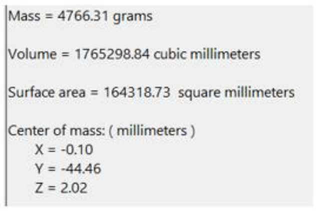
```
Mass = 4283.77 grams

Volume = 1586582.41 cubic millimeters

Surface area = 139712.84  square millimeters

Center of mass: ( millimeters )
     X = 0.00
     Y = 0.07
     Z = 92.17
```



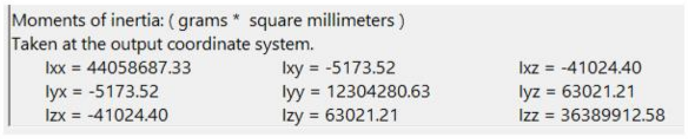
```
Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system.
      Ixx = 78921407.33          Ixy = -1.73               Ixz = 3.94
      Iyx = -1.73                Iyy = 73833198.42         Iyz = 73237.31
      Izx = 3.94                 Izy = 73237.31            Izz = 9922402.27
```

Revolute Joint 5



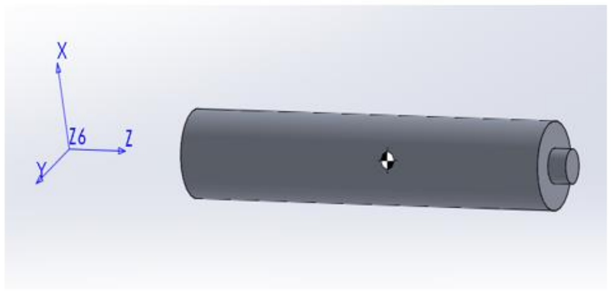


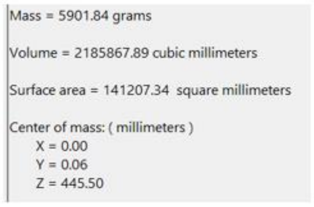
```
Mass = 4766.31 grams

Volume = 1765298.84 cubic millimeters

Surface area = 164318.73  square millimeters

Center of mass: ( millimeters )
     X = -0.10
     Y = -44.46
     Z = 2.02
```



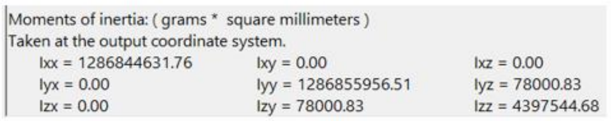
```
Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system.
      Ixx = 44058687.33          Ixy = -5173.52            Ixz = -41024.40
      Iyx = -5173.52             Iyy = 12304280.63         Iyz = 63021.21
      Izx = -41024.40            Izy = 63021.21            Izz = 36389912.58
```

Revolute Joint 6



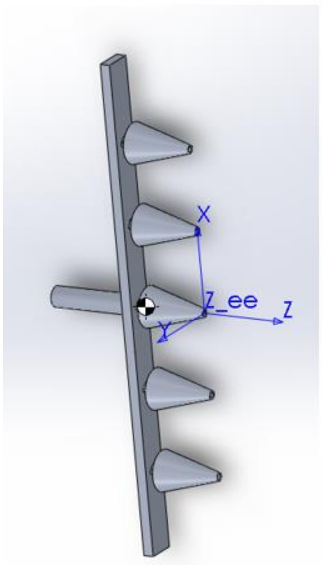


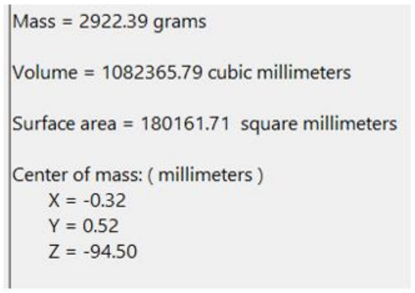
```
Mass = 5901.84 grams

Volume = 2185867.89 cubic millimeters

Surface area = 141207.34  square millimeters

Center of mass: ( millimeters )
     X = 0.00
     Y = 0.06
     Z = 445.50
```



```
Moments of inertia: ( grams *  square millimeters )
Taken at the output coordinate system.
      Ixx = 1286844631.76        Ixy = 0.00                Ixz = 0.00
      Iyx = 0.00                 Iyy = 1286855956.51       Iyz = 78000.83
      Izx = 0.00                 Izy = 78000.83            Izz = 4397544.68
```

End Effector

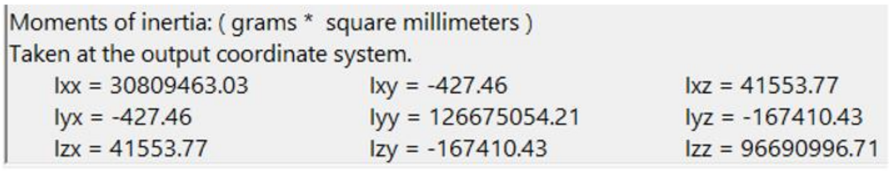



Mass = 2922.39 grams

Volume = 1082365.79 cubic millimeters

Surface area = 180161.71 square millimeters

Center of mass: (millimeters)
X = -0.32
Y = 0.52
Z = -94.50


Moments of inertia: (grams * square millimeters)
Taken at the output coordinate system.

```
Ixx = 30809463.03      Ixy = -427.46          Ixz = 41553.77
Iyx = -427.46          Iyy = 126675054.21     Iyz = -167410.43
Izx = 41553.77         Izy = -167410.43       Izz = 96690996.71
```

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.

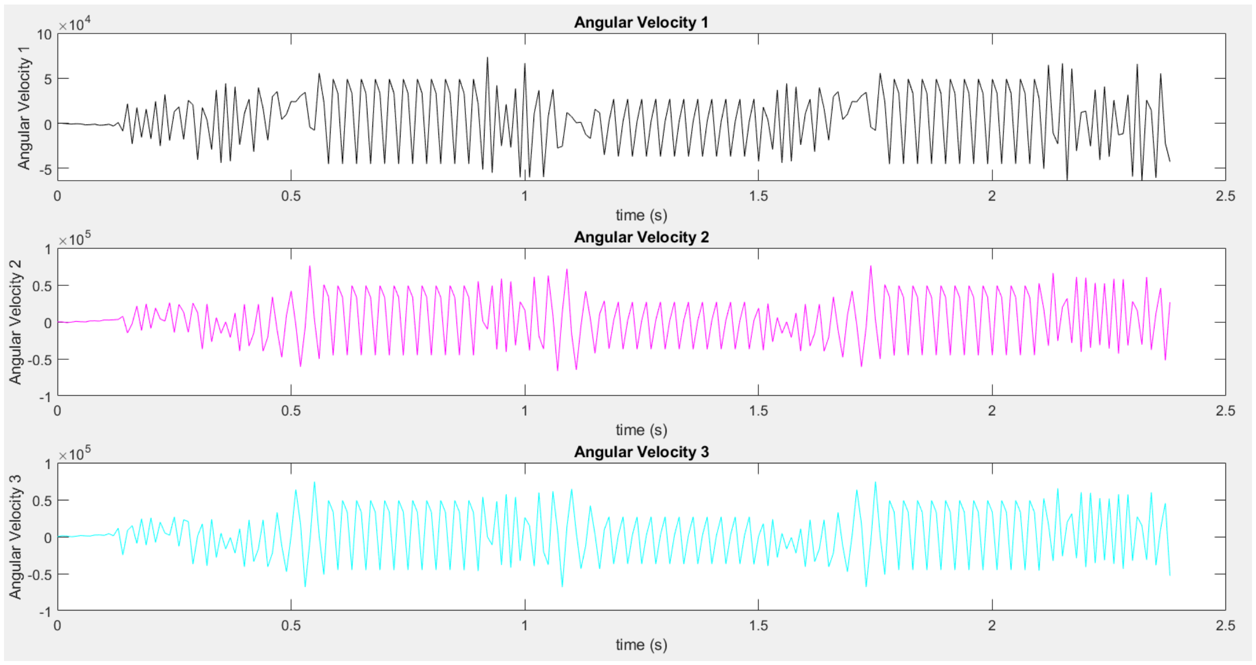
```
%% Velocity Transformation Matrix
%Position Vector P ref_w->ee
R_06 = T_06(1:3,1:3);
P_6ee = T_6ee(1:3,4);

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.')
```

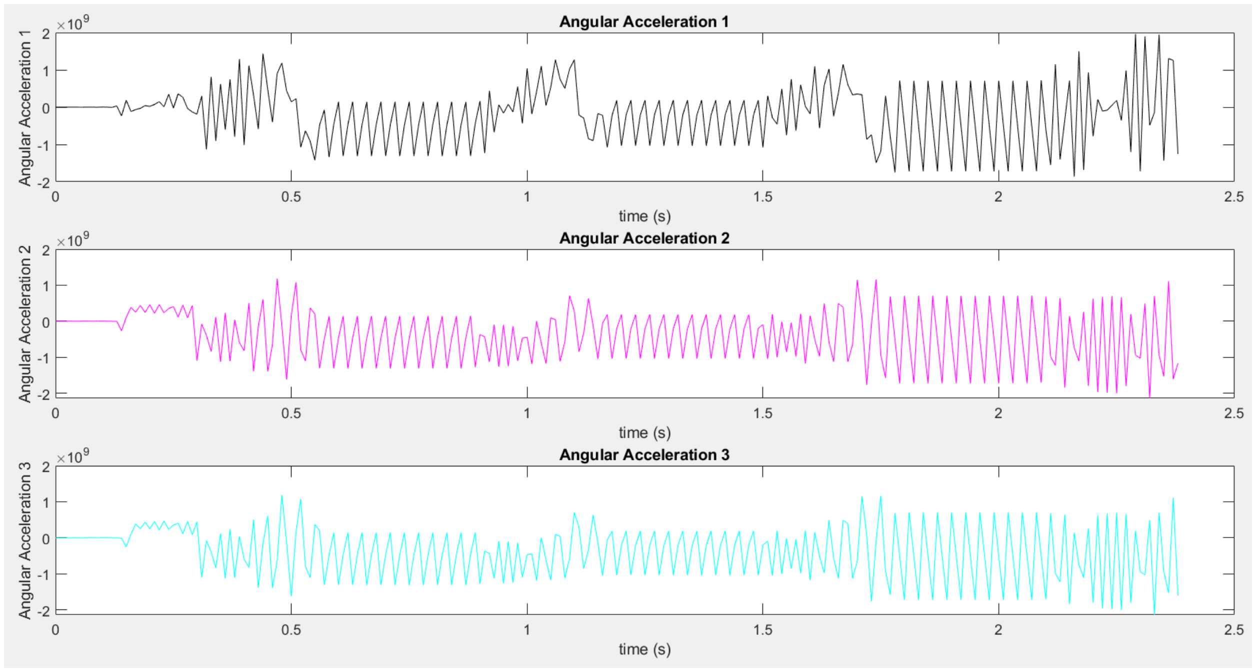
Simulation

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

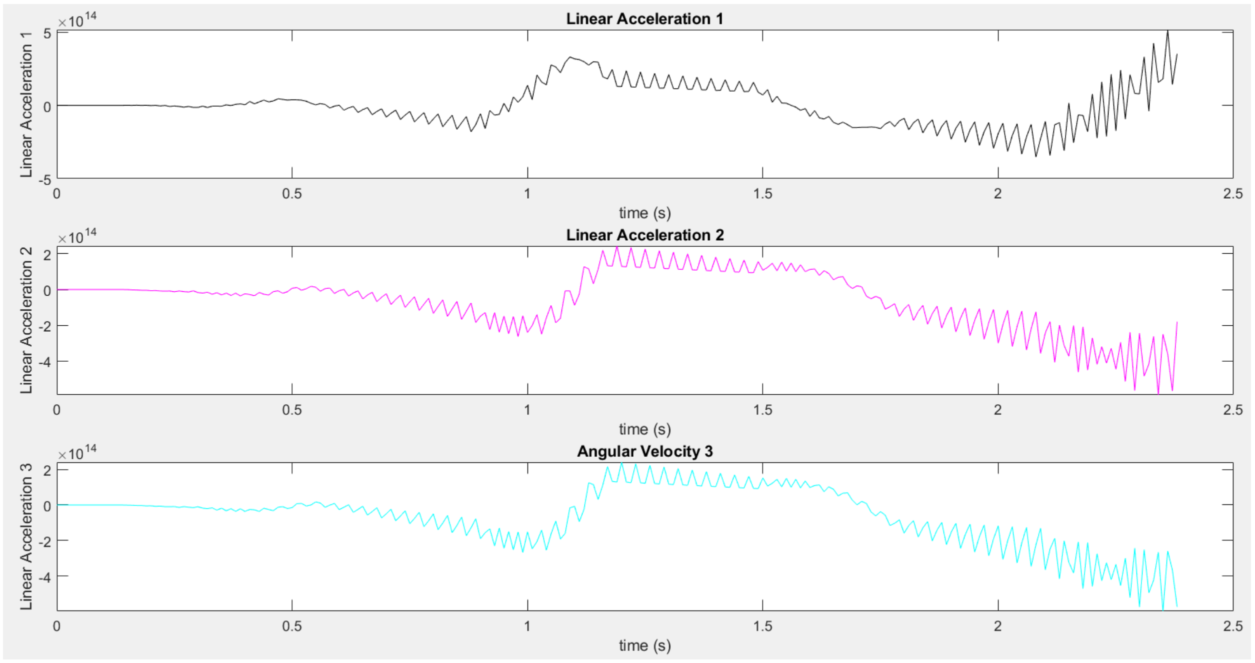
The below figure shows the Angular velocities plot against time.



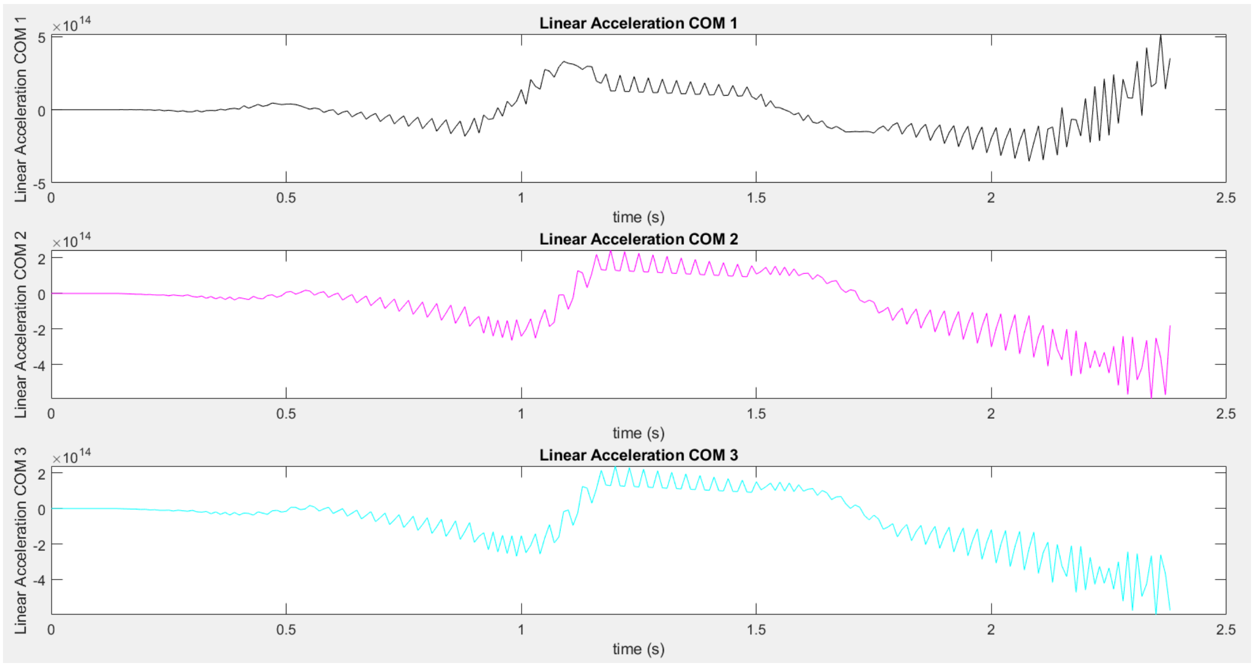
The below figure shows the Angular acceleration plot against time.



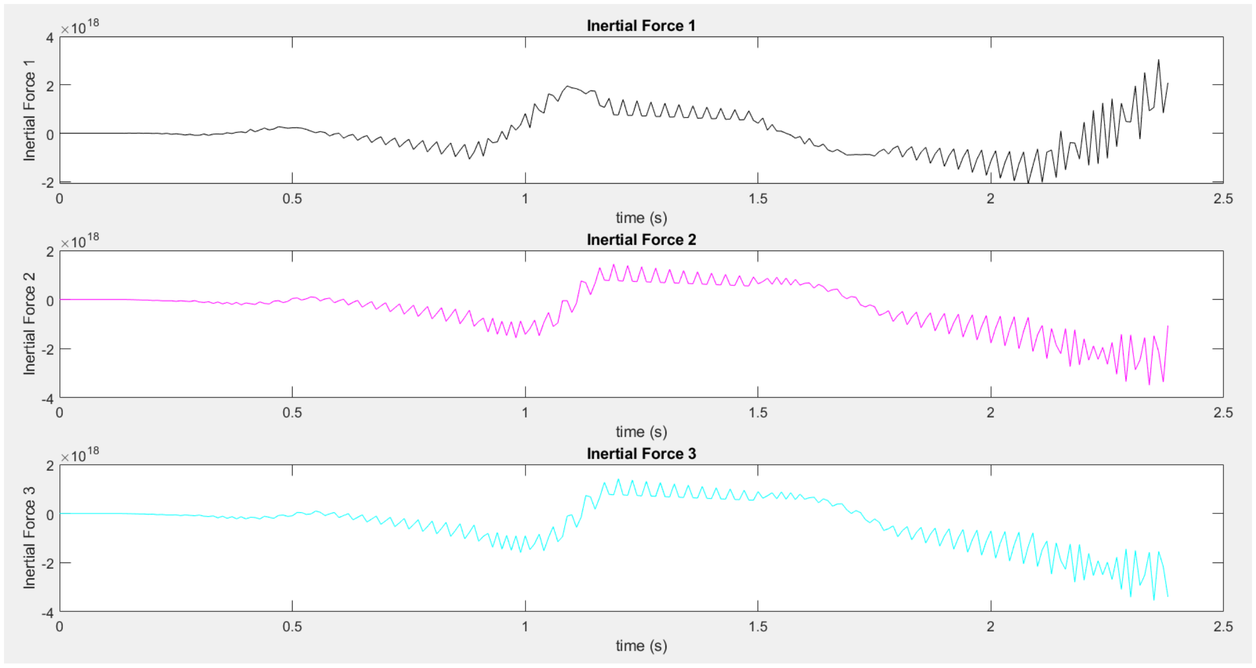
The below figure shows the Linear acceleration plot against time.



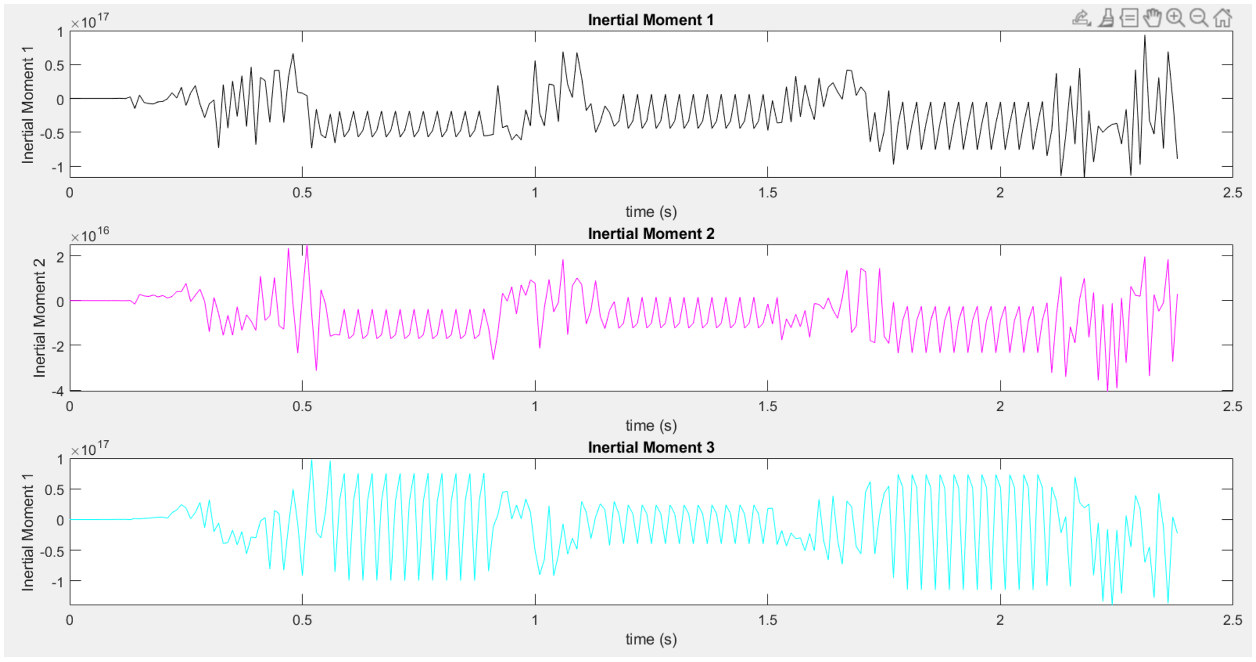
The below figure shows the Linear acceleration COM plot against time.



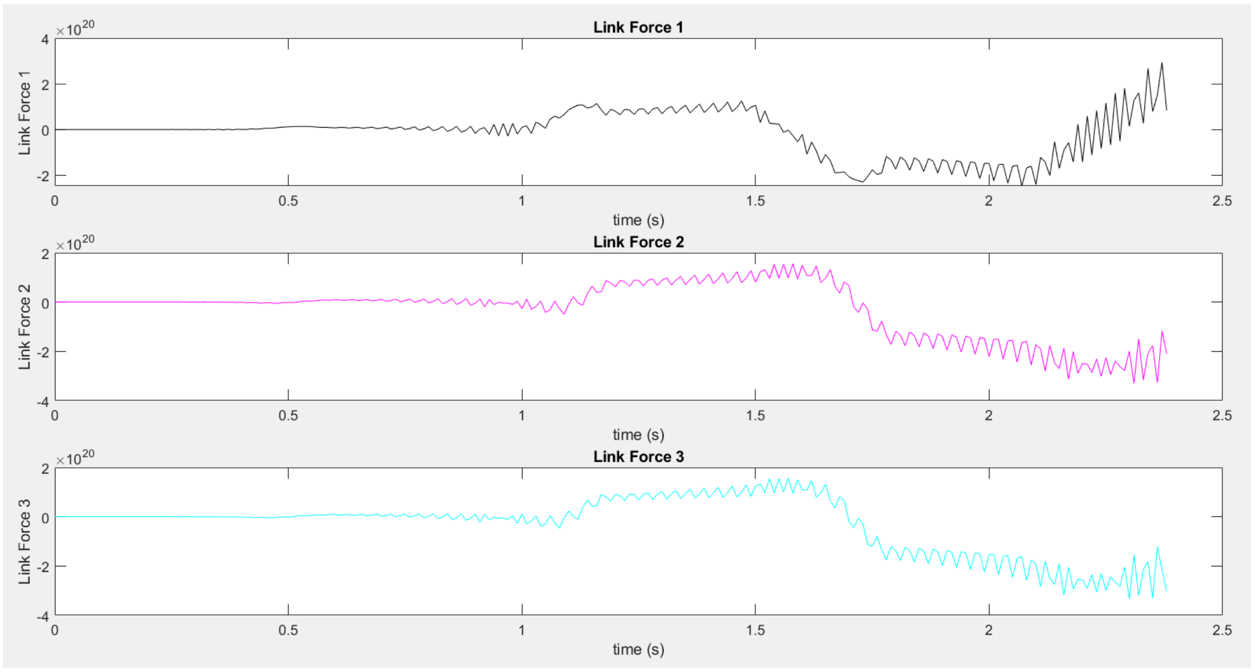
The below figure shows the Inertial Force plot against time.



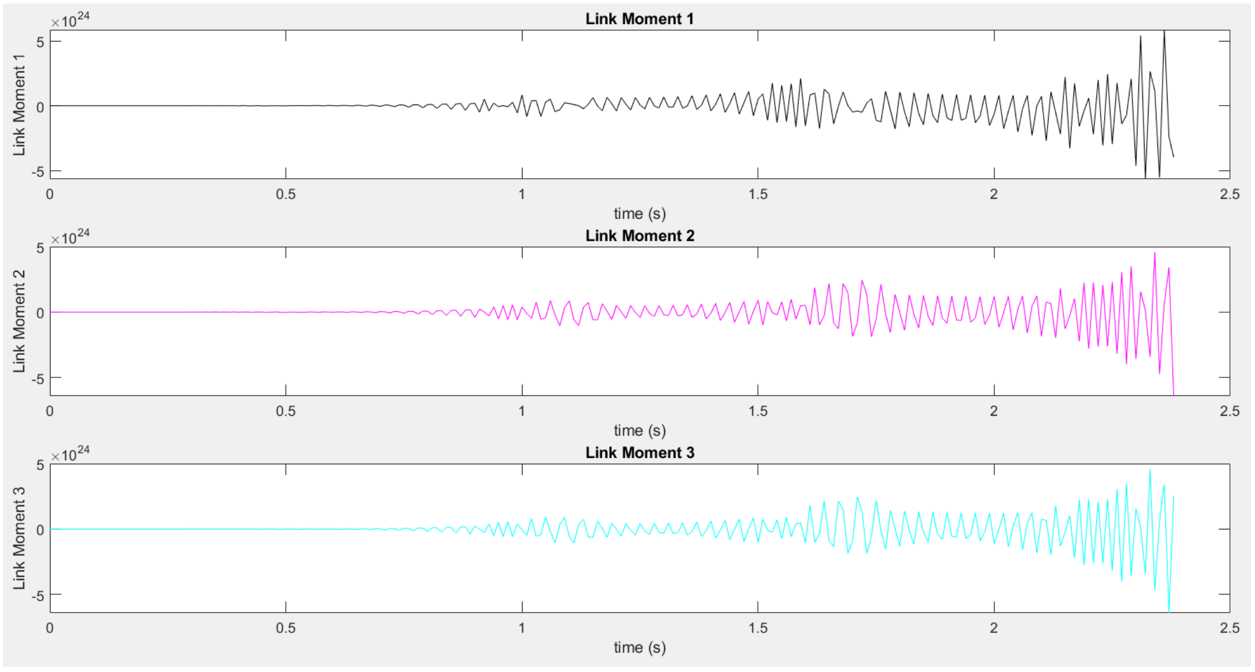
The below figure shows the Inertial Moment plot against time.



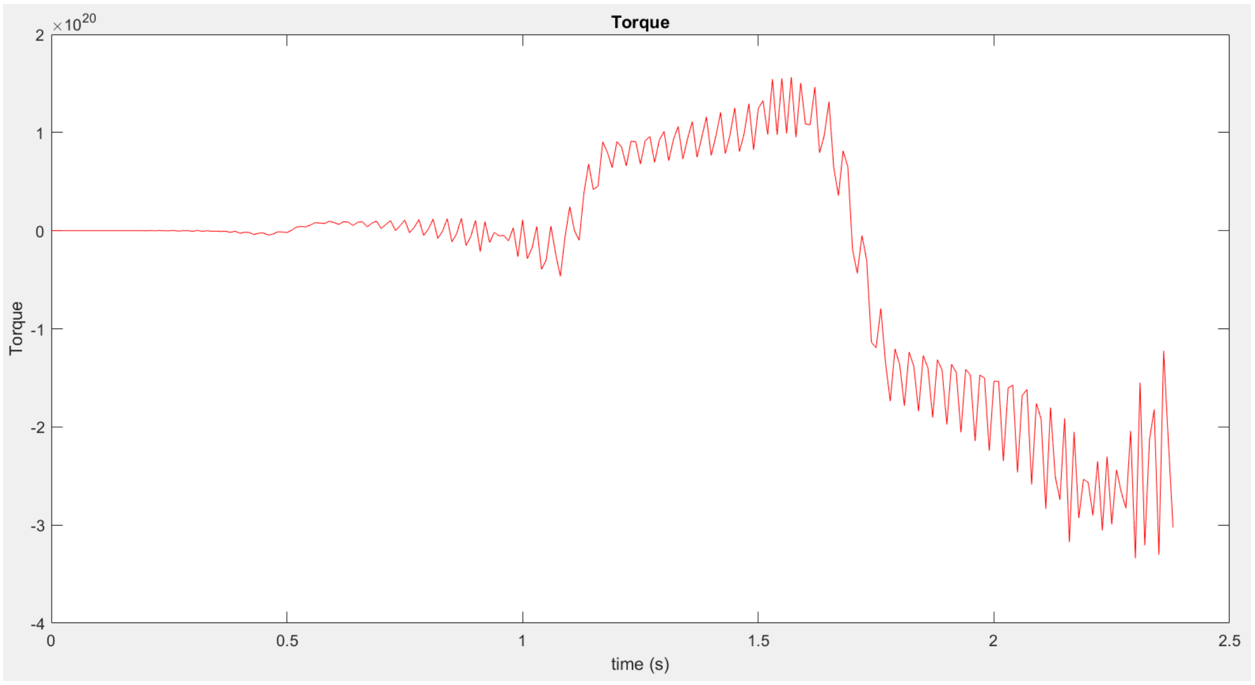
The below figure shows the Link Force plot against time.



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.

1. 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
```

2. 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];
```

3. 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;
```

4. 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.')
```

5. 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;
```