



Course Code: AE/ME8138		
Course Title: Computational Dynamics		
Semester: W2020		
Instructor: Dr. Puren Ouyang		
<b>Submission:</b>	Select from the following...	Report
Due Date: April 28, 2020		

## Title: Final Project Report

Section Number: N/A

Submission Date: April 22, 2020

Submission by:		
Name	Student Number (XXXX99999)	Signature
Kevin Lin	33260	
Tianhao William Jiang	38719	

*By signing the above you attest that you have contributed to this submission and confirm that all work you contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, and "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at [www.ryerson.ca/senate/current/pol60.pdf](http://www.ryerson.ca/senate/current/pol60.pdf).*

## **Table of Contents**

<b>Manipulator Model and DH Parameters</b>	<b>2</b>
<b>Forward Kinematics</b>	<b>3</b>
<b>Inverse Kinematics</b>	<b>4</b>
<b>Workspace Analysis</b>	<b>5</b>
<b>Dynamics Model</b>	<b>6</b>
<b>Trajectory Planning</b>	<b>10</b>
<b>Appendix: Matlab Code</b>	<b>17</b>

## Manipulator Model and DH Parameters

The image shown in Fig. # is the CAD model of the RRP robot manipulator, along with the DH convention coordinate system, chosen for this project.

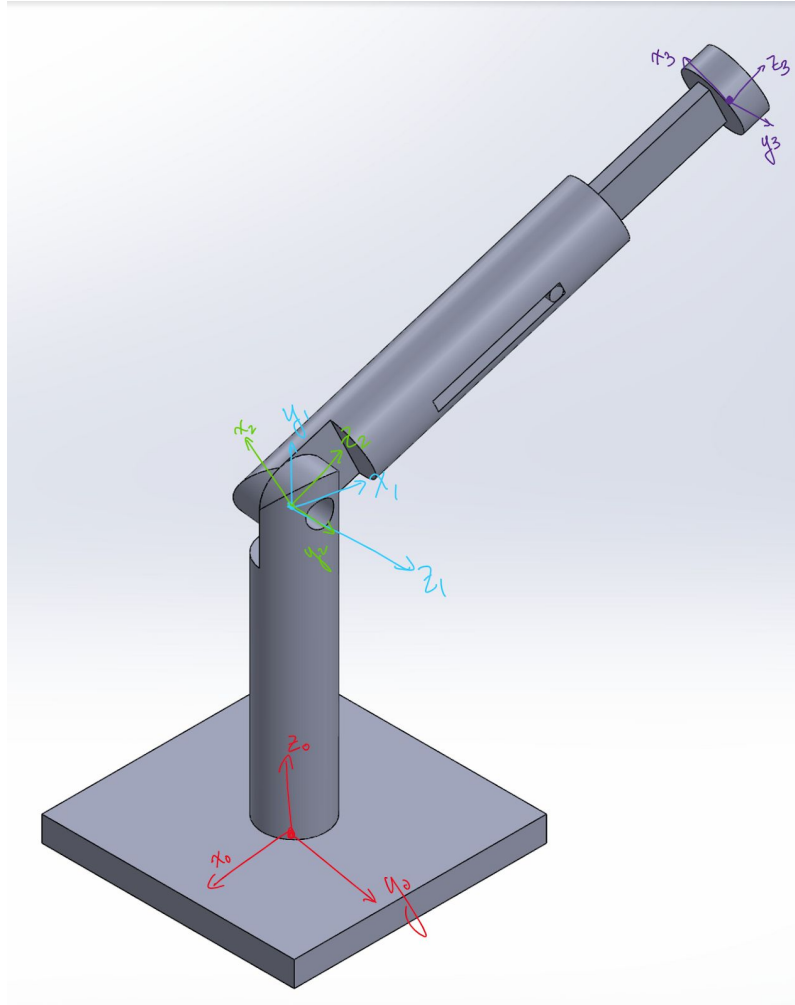


Fig. 1. Model of RRP manipulator with DH convention coordinate systems

Table 1. DH parameters of RRP manipulator

<b>i</b>	<b><math>\theta</math></b>	<b>d</b>	<b><math>\alpha</math></b>	<b>a</b>
1	$q_1$	0.120 m	$90^\circ$	0
2	$q_2$	0	$90^\circ$	0
3	0	$q_3$	0	0

NOTE:  $q_3$  is constrained between 0.120 m to 0.170 m

## Forward Kinematics

The transformation matrix from the base frame  $F^0$  to the end manipular frame  $F^3$ , was found using the DH parameters:

Table 2. Transformation matrix  $T_0^3$

$\cos(q_1) \cdot \cos(q_2)$	$\sin(q_1)$	$\cos(q_1) \cdot \sin(q_2)$	$q_3 \cdot \cos(q_1) \cdot \sin(q_2)$
$\cos(q_2) \cdot \sin(q_1)$	$-\cos(q_1)$	$\sin(q_1) \cdot \sin(q_2)$	$q_3 \cdot \sin(q_1) \cdot \sin(q_2)$
$\sin(q_2)$	0	$-\cos(q_2)$	$0.12 - q_3 \cdot \cos(q_2)$
0	0	0	1

The forward kinematics are as follows, where x,y, and z are in metres.

$$x = q_3 \cdot \cos(q_1) \cdot \sin(q_2)$$

$$y = q_3 \cdot \sin(q_1) \cdot \sin(q_2)$$

$$z = 0.12 - q_3 \cdot \cos(q_2)$$

## Inverse Kinematics

The inverse kinematics were solved algebraically from the forward kinematic equations from the previous section. A benefit of the system is that there the system does not have arm up/down configurations.

Algebraically:

$$A = x^2 + y^2 = q_3^2 \cdot \sin^2(q_2)$$

$$B = (z - 0.12)^2 = q_3^2 \cdot \cos^2(q_2)$$

$$\frac{y}{x} = \frac{q_3 \cdot \sin(q_1) \cdot \sin(q_2)}{q_3 \cdot \cos(q_1) \cdot \sin(q_2)} = \tan(q_1)$$

$$q_1 = \tan^{-1}\left(\frac{y}{x}\right)$$

$$\frac{A}{B} = \frac{x^2 + y^2}{(z - 0.12)^2} = \frac{q_3^2 \sin^2(q_2)}{q_3^2 \cos^2(q_2)} = \tan^2(q_2)$$

$$q_2 = \tan^{-1}\left(\frac{\sqrt{x^2 + y^2}}{z - 0.12}\right)$$

$$A + B = x^2 + y^2 + (z - 0.12)^2 = q_3^2$$

$$q_3 = \sqrt{x^2 + y^2 + (z - 0.12)^2}$$

## Workspace Analysis

This robot arm works within two spherical volumes of radius 0.170 m and 0.120 m about joint 2. A singular configuration exists when joint variable  $q_2 = 90^\circ$  or  $270^\circ$  or when  $q_3 = 0$  mm. However,  $q_3$  cannot result in a singular configuration as a limit of its range has been set. Note that the physical limitation such as link interference was not explored but is possible in joint 2.

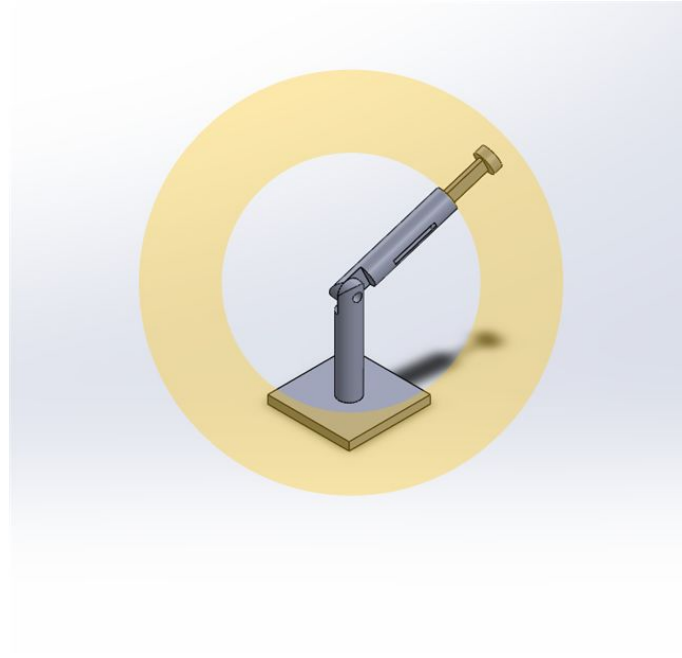


Fig. 2. Workspace of robot

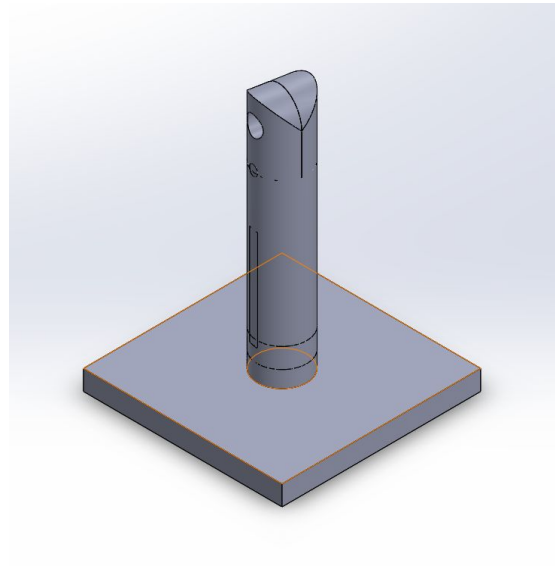


Fig. 3. One of the existing singular configuration

## Dynamics Model

For the dynamics model, Solidworks was used to determine the moments of inertia of the links. The values shown below were then substituted into the Matlab code shown in the Appendix to retrieve the dynamics model of the robot manipulator.

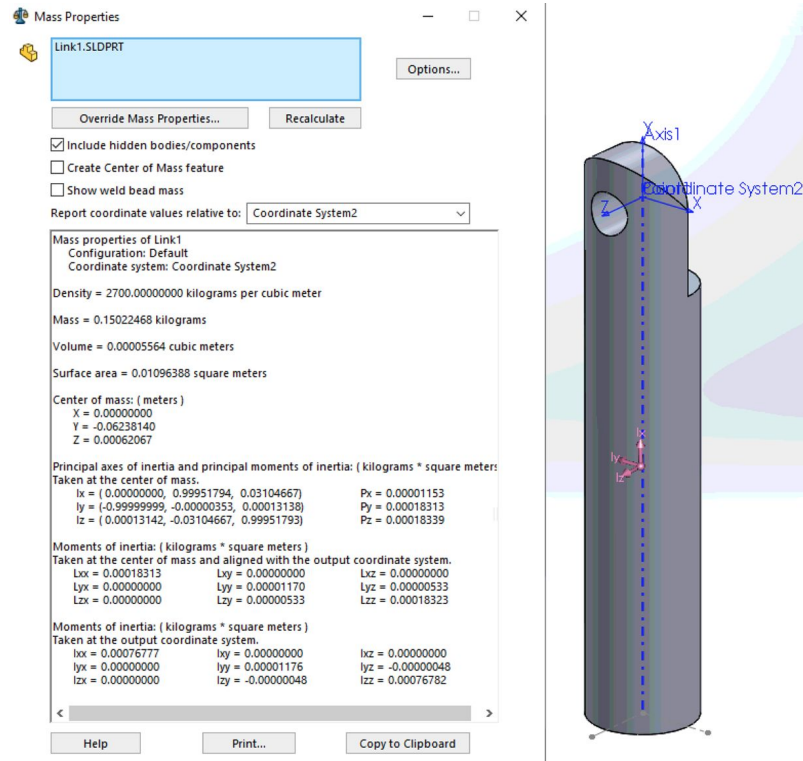


Fig. 4. Moment of inertia values from Solidworks for link 1

Table 3. Inertial matrix  $I_1$

5.9E-06	0	0	0
0	0.00017733	0.00000533	-0.009357
0	0.00000533	5.8E-06	0.000093
0	-0.009357	0.000093	0.15

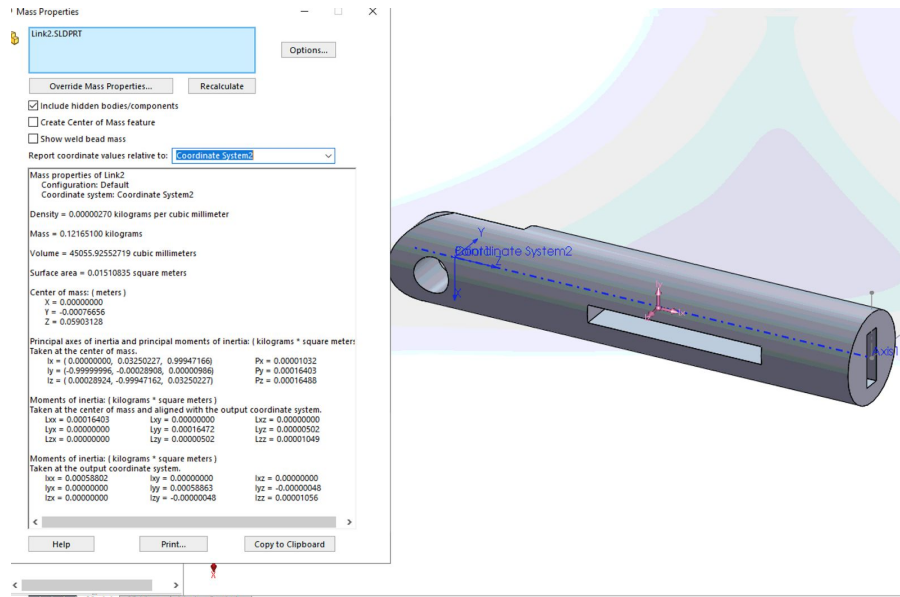


Fig. 5. Moment of inertia values from Solidworks for link 2

Table 4. Inertial matrix  $I_2$

0.00000559	0	0	0
0	0.0000049	0.00000502	-9.357E-05
0	0.00000502	0.00015913	0.007198
0	-9.357E-05	0.007198	0.122

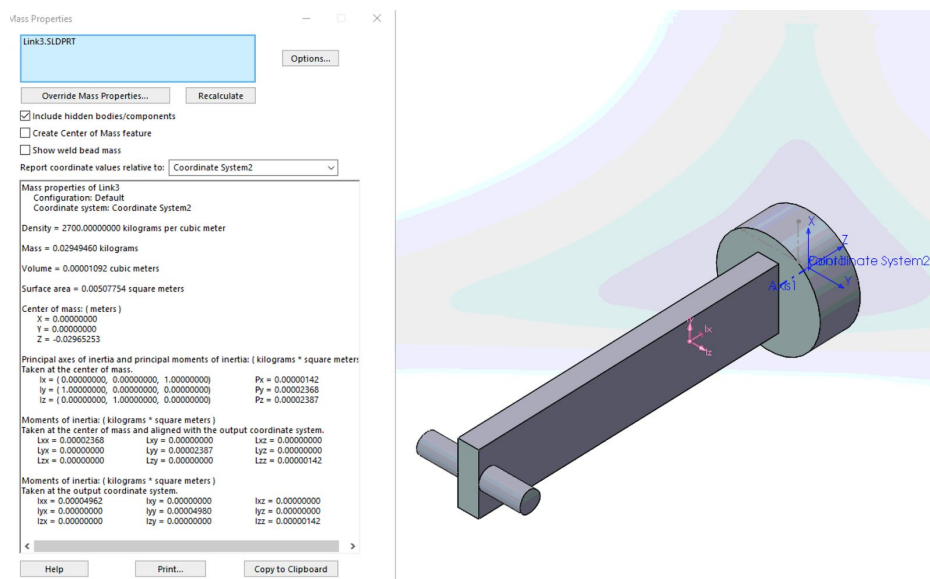


Fig. 6. Moment of inertia values from Solidworks for link 3



Table 5. Inertial matrix  $I_3$

8.05E-07	0	0	0
0	6.15E-07	0	0
0	0	2.3065E-05	-0.0008762
0	0	-0.0008762	0.0295

Material = Al 6061-T6

Density = 2700 kg/m<sup>3</sup>

$m_1 = 0.15$  kg

$m_2 = 0.122$  kg

$m_3 = 0.0295$  kg

Rbar:

rbar1 = [0; -0.06238; 0.00062; 1]

rbar2 = [0; -0.000767; 0.059; 1]

rbar3 = [0; 0; -0.0297; 1]

Gravity = [0, 0, -9.81, 0] (wrt to base frame  $F^0$ )

Using the parameters from above, the dynamic model of the manipulator was determined: Where  $F_1$ ,  $F_2$ , and  $F_3$  are the forces/torques acting on joints 1,2, and 3 respectively. Additionally,  $\dot{q}_i = \frac{q_i}{t}$  and  $\ddot{q}_i = \frac{\dot{q}_i}{t}$  for joints  $i = 1,2$ , and 3.

$$\begin{aligned}
 F_1 = & 0.0001115*\ddot{q}_1 - 5.02e-6*\dot{q}_2^2*\sin(q_2) - 0.0004381*\dot{q}_1*\dot{q}_3 - 0.0008761*\ddot{q}_1*q_3 - \\
 & 8.79e-5*\ddot{q}_1*\cos(2.0*q_2) + 0.01475*\ddot{q}_1*q_3^2 + 5.02e-6*\ddot{q}_2*\cos(q_2) - \\
 & 0.01475*\ddot{q}_1*q_3^2*\cos(2.0*q_2) + 2.51e-6*\dot{q}_1*\dot{q}_2*\sin(q_2) + 2.51e-6*\dot{q}_1*\dot{q}_2*\sin(2.0*q_1 - \\
 & 1.0*q_2) - 4.395e-5*\dot{q}_1*\dot{q}_2*\sin(2.0*q_1 - 2.0*q_2) + 0.01475*\dot{q}_1*\dot{q}_3*q_3 + \\
 & 0.0004381*\dot{q}_1*\dot{q}_3*\cos(2.0*q_2) + 0.0008762*\ddot{q}_1*q_3*\cos(2.0*q_2) + \\
 & 4.715e-5*\dot{q}_1*\dot{q}_2*\sin(2.0*q_1) + 0.0001319*\dot{q}_1*\dot{q}_2*\sin(2.0*q_2) - \\
 & 0.01475*\dot{q}_1*\dot{q}_3*q_3*\cos(2.0*q_2) - 0.0004381*\dot{q}_1*\dot{q}_2*q_3*\sin(2.0*q_1) - \\
 & 0.001314*\dot{q}_1*\dot{q}_2*q_3*\sin(2.0*q_2) - 0.0008761*\dot{q}_1*\dot{q}_3*\cos(q_1)*\sin(q_2)^2 + \\
 & 0.007375*\dot{q}_1*\dot{q}_2*q_3^2*\sin(2.0*q_1) + 0.02213*\dot{q}_1*\dot{q}_2*q_3^2*\sin(2.0*q_2) + \\
 & 0.0004381*\dot{q}_1*\dot{q}_2*q_3*\sin(2.0*q_1 - 2.0*q_2) - 0.007375*\dot{q}_1*\dot{q}_2*q_3^2*\sin(2.0*q_1 - 2.0*q_2) + \\
 & 0.0295*\dot{q}_1*\dot{q}_3*q_3*\cos(q_1)*\sin(q_2)^2
 \end{aligned}$$

$$\begin{aligned}
F_2 = & \sin(q_2) * (0.2894 * q_3 + 0.06202) + ddq_2 * (0.0295 * q_3^2 - 0.001752 * q_3 + 0.0001886) + \\
& dq_2 * (0.0295 * dq_3 * q_3 - 0.0008761 * dq_3 + 4.715e-5 * dq_1 * \sin(2.0 * q_1) + \\
& 4.395e-5 * dq_1 * \sin(2.0 * q_2) + 4.395e-5 * dq_1 * \sin(2.0 * q_1 - 2.0 * q_2) + \\
& 0.007375 * dq_1 * q_3^2 * \sin(2.0 * q_1) + 0.007375 * dq_1 * q_3^2 * \sin(2.0 * q_2) - \\
& 0.0004381 * dq_1 * q_3 * \sin(2.0 * q_1 - 2.0 * q_2) + 0.007375 * dq_1 * q_3^2 * \sin(2.0 * q_1 - 2.0 * q_2) - \\
& 0.0004381 * dq_1 * q_3 * \sin(2.0 * q_1) - 0.0004381 * dq_1 * q_3 * \sin(2.0 * q_2)) + 5.02e-6 * ddq_1 * \cos(q_2) - \\
& 4.235e-25 * dq_1^2 * \sin(2.0 * q_2) * (3.483e+22 * q_3^2 - 2.069e+21 * q_3 + 2.075e+20) - \\
& 1.0 * dq_3 * \cos(q_2) * (0.0295 * q_3 - 0.0008762) * (dq_2 - 1.0 * dq_1 * \sin(q_1) * \sin(q_2))
\end{aligned}$$

$$\begin{aligned}
F_3 = & 0.0295 * ddq_3 - 0.2894 * \cos(q_2) - 0.0295 * dq_3 * \sin(q_2) * (dq_2 - 1.0 * dq_1 * \sin(q_1) * \sin(q_2)) - \\
& 6.939e-21 * dq_1^2 * \sin(q_2)^2 * (4.251e+18 * q_3 - 1.263e+17) - 1.0 * dq_2 * (0.007375 * q_3 - \\
& 0.000219) * (4.0 * dq_2 - 1.0 * dq_1 + dq_1 * \cos(2.0 * q_1) + dq_1 * \cos(2.0 * q_2) - 1.0 * dq_1 * \cos(2.0 * q_1 - \\
& 2.0 * q_2))
\end{aligned}$$

## Trajectory Planning

NOTE:  $q_3$  is constrained between 0.120 m to 0.170 m

The trajectory for this robot arm will use a predetermined movement similar to a pick and place. Below, the table shows the joint variables and their values with the respective time.

Table 6. Trajectory Values for One Cycle

Time Period	Time (s)	q1 (rad)	q2 (rad)	q3 (m)
1 (0:0.5)	0	0	$\pi/3$	0.160
2 (0.5:2)	0.5	0	$\pi/3$	0.130
3 (2:2.2)	2	$\pi/2$	0	0.130
4 (2.2:2.5)	2.2	$\pi/2$	0	0.160
5 (2.5:4)	2.5	$\pi/2$	0	0.130
6 (4:4.5)	4	0	$\pi/3$	0.130
7	4.5	0	$\pi/3$	0.160

A 5-3-5 Trajectory pattern will be used for the movement. This is because the movement in joints 1 and 2 during periods 3, 4, 5 are zero and thus, the '3' component of the trajectory will be constant. The '5' components will ensure a continuous acceleration so no jerky motion will affect the movement.

The notation for the trajectories are defined below:

$$q_{ij} = a_{ij0} + a_{ij1}t + a_{ij2}t^2 + a_{ij3}t^3 + a_{ij4}t^4 + a_{ij5}t^5$$

Where i is the joint number and j is the motion defined by the Time Period.

For all joints, the velocities/accelerations at the beginning and end of the Time Periods are 0 to ensure continuous movement. With these constraints, the equations of motion for joints 1 and 2 were solved;

### Joint 1

$$q_{11} = q_{16} = 0$$

$$q_{13} = q_{14} = \pi/2$$

$$q_{12} = -0.9114 + 6.2056t - 15.514t^2 + 17.0654t^3 - 7.757t^4 + 1.2411t^5$$

$$q_{15} = 377.3014 - 620.5615t + 403.365t^2 - 128.7665t^3 + 20.1682t^4 - 1.2411t^5$$

### Joint 2

$$q_{21} = q_{26} = \pi/3$$

$$q_{23} = q_{24} = 0$$

$$q_{22} = 1.6548 - 4.1371t + 10.3427t^2 - 11.377t^3 + 5.1713t^4 - 0.8274t^5$$

$$q_{25} = -250.4871 + 413.7077t - 268.91t^2 + 85.8443t^3 - 13.4455t^4 + 0.8274t^5$$

### Joint 3

$$q_{32} = q_{35} = 0.130$$

$$q_{31} = 0.16 - 2.4t^3 + 7.2t^4 - 5.76t^5$$

$$q_{33} = -22799.86999879618 + 54449.9999971293t - 51974.99999726391t^2$$

$$+ 24787.49999869711t^3 - 5906.24999969003t^4 + 562.4999997052t^5$$

$$q_{34} = 5237.39851860875 - 11203.70370389647t + 9574.07407423861t^2$$

$$- 4085.1851852553t^3 + 870.37037038529t^4 - 74.07407407534t^5$$

$$q_{36} = -7894.91 + 9331.2t - 4406.4t^2 + 1039.2t^3 - 122.4t^4 + 5.76t^5$$

The velocity and accelerations were computed using the derivative and double derivative respectively. The graphs of the joint paths, velocities, accelerations, forces/torques, and end-effector trajectory are shown below. Note the continuity in the velocity and acceleration which would mean a non-jerky motion.

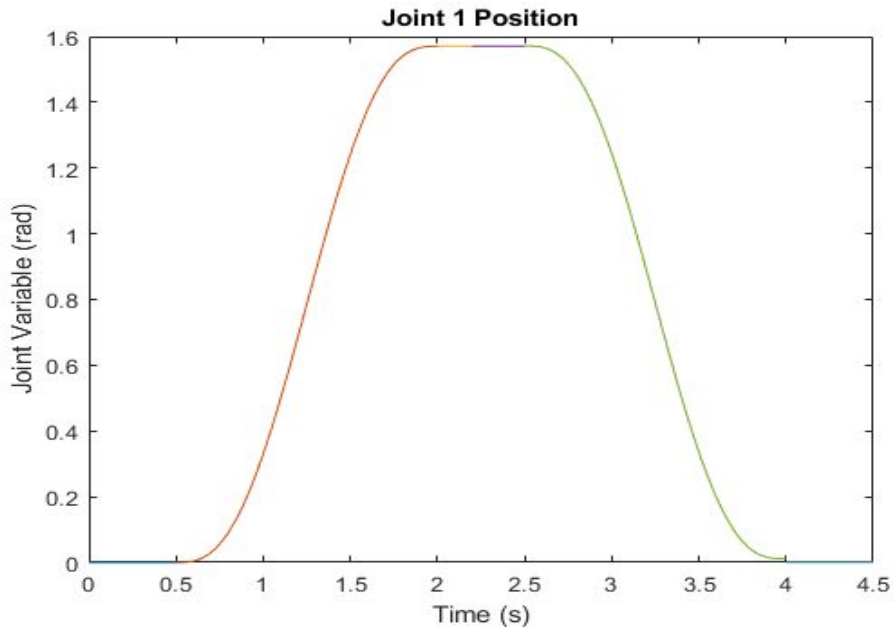


Fig. 7. Joint 1 Position

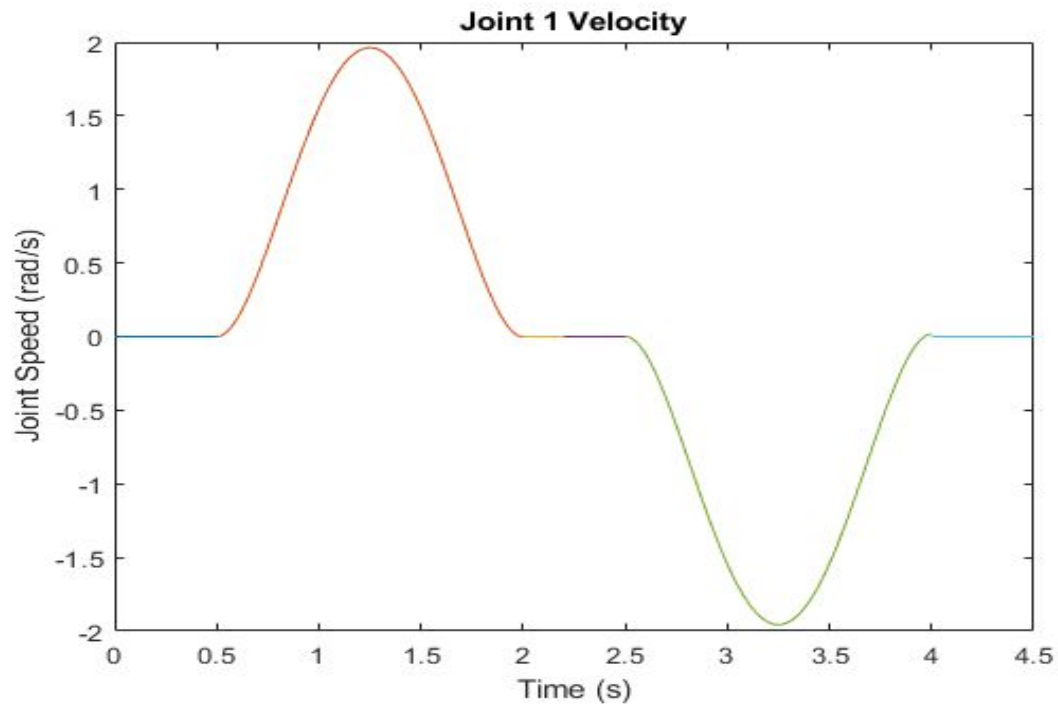


Fig. 8. Joint 1 Velocity

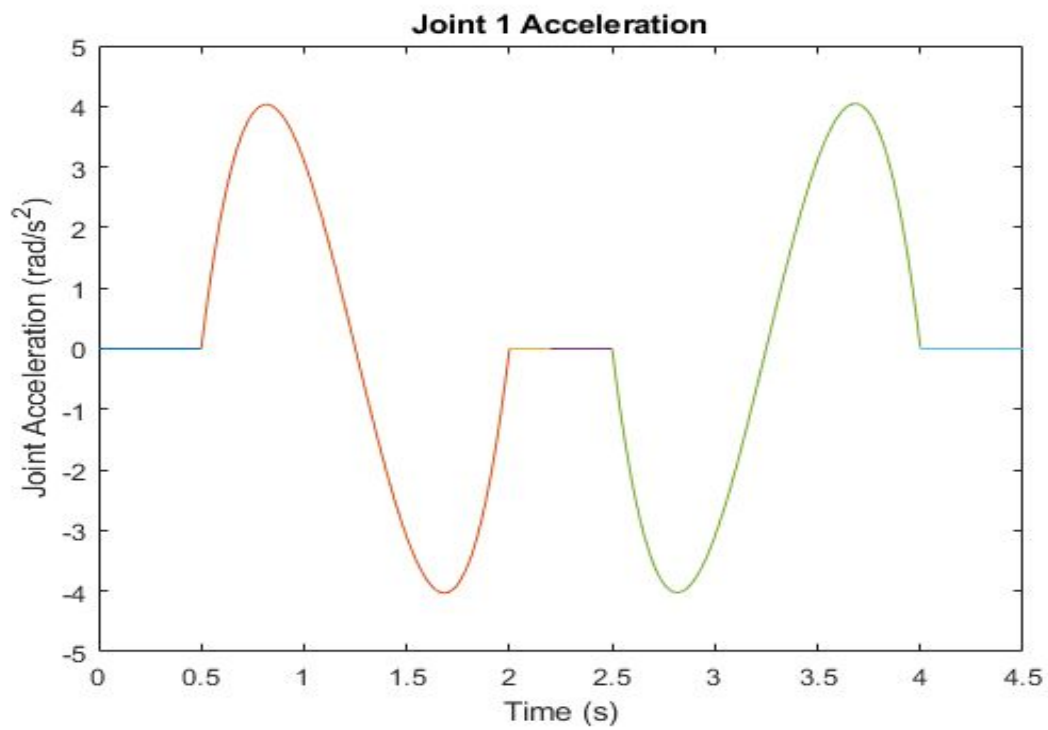


Fig. 9. Joint 1 Acceleration

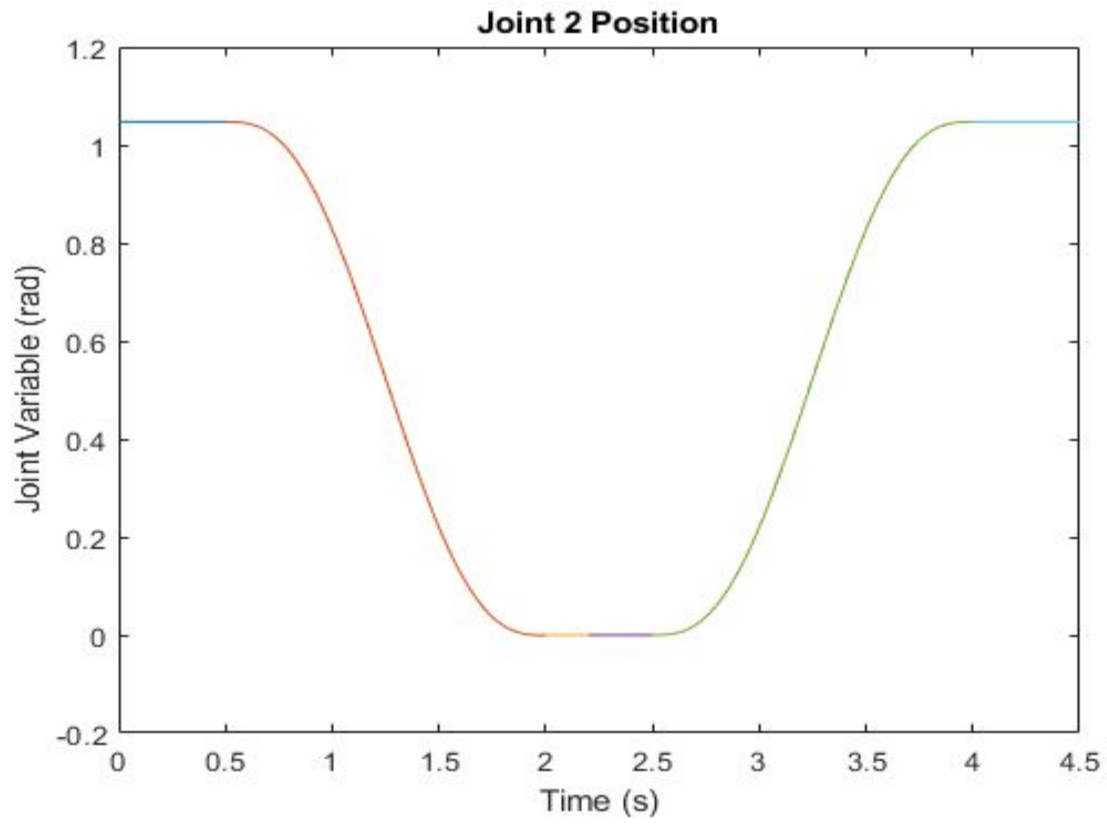


Fig. 10. Joint 2 Position

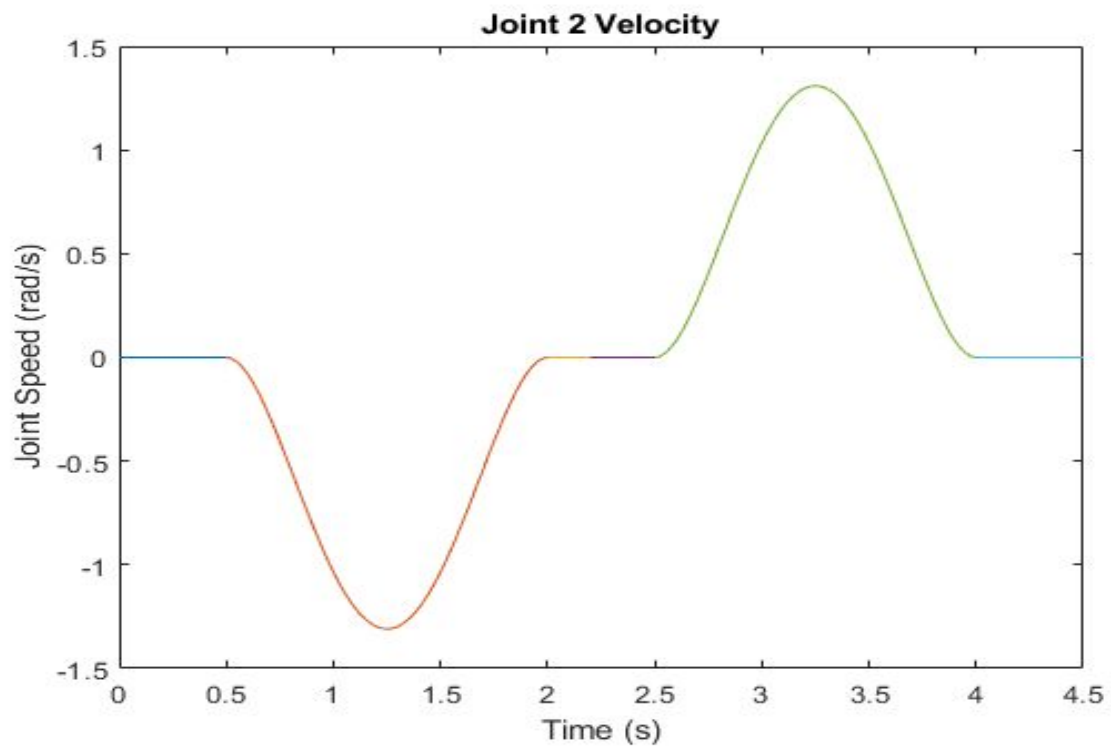


Fig.11. Joint 2 Velocity

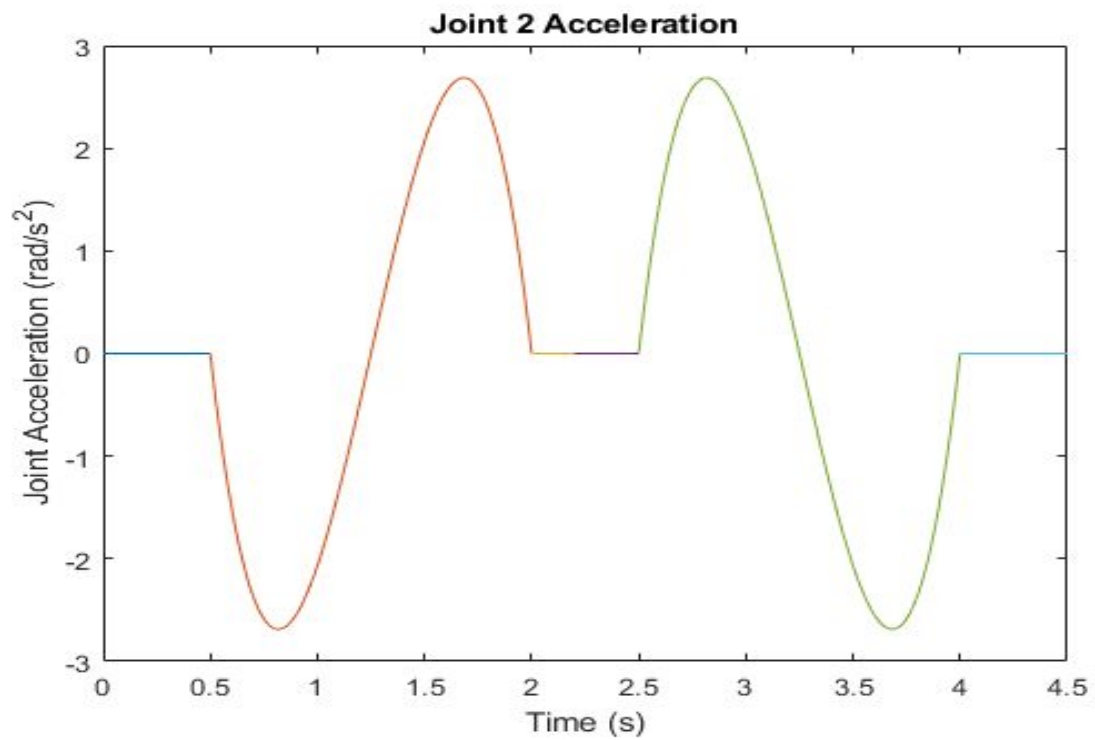


Fig. 12. Joint 2 Acceleration

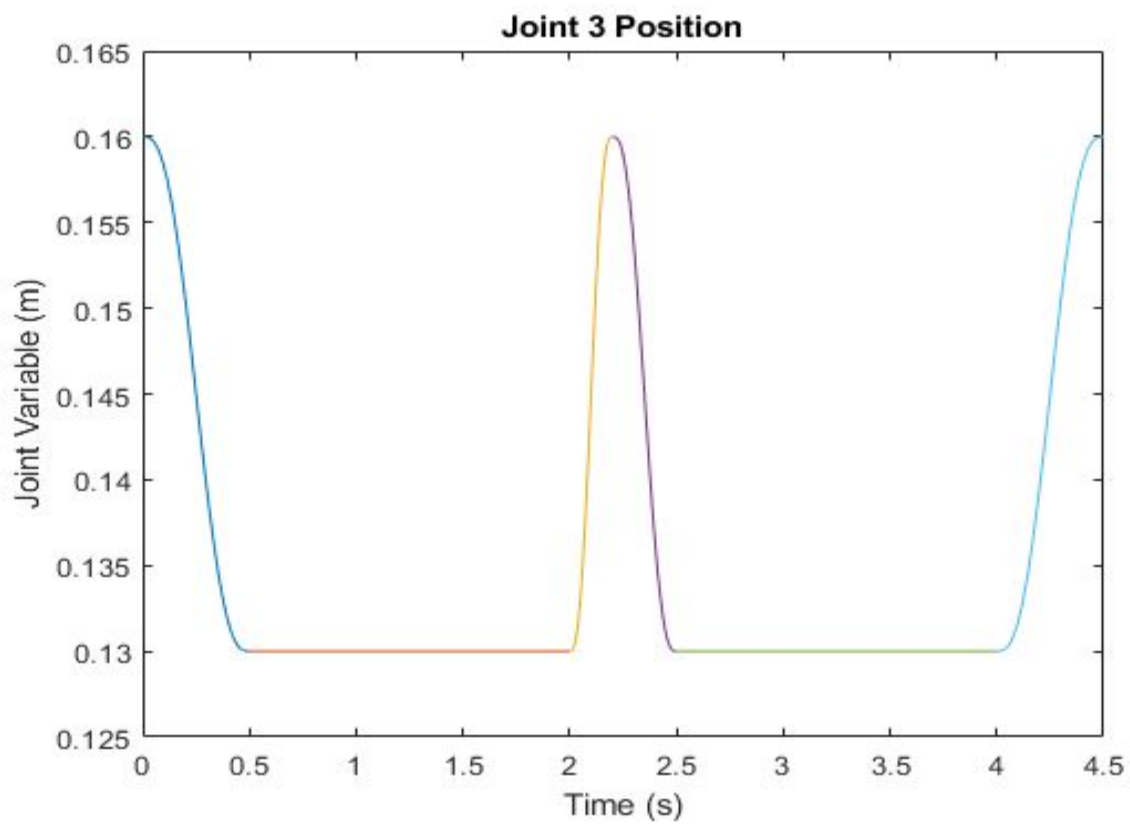


Fig. 13. Joint 3 Position

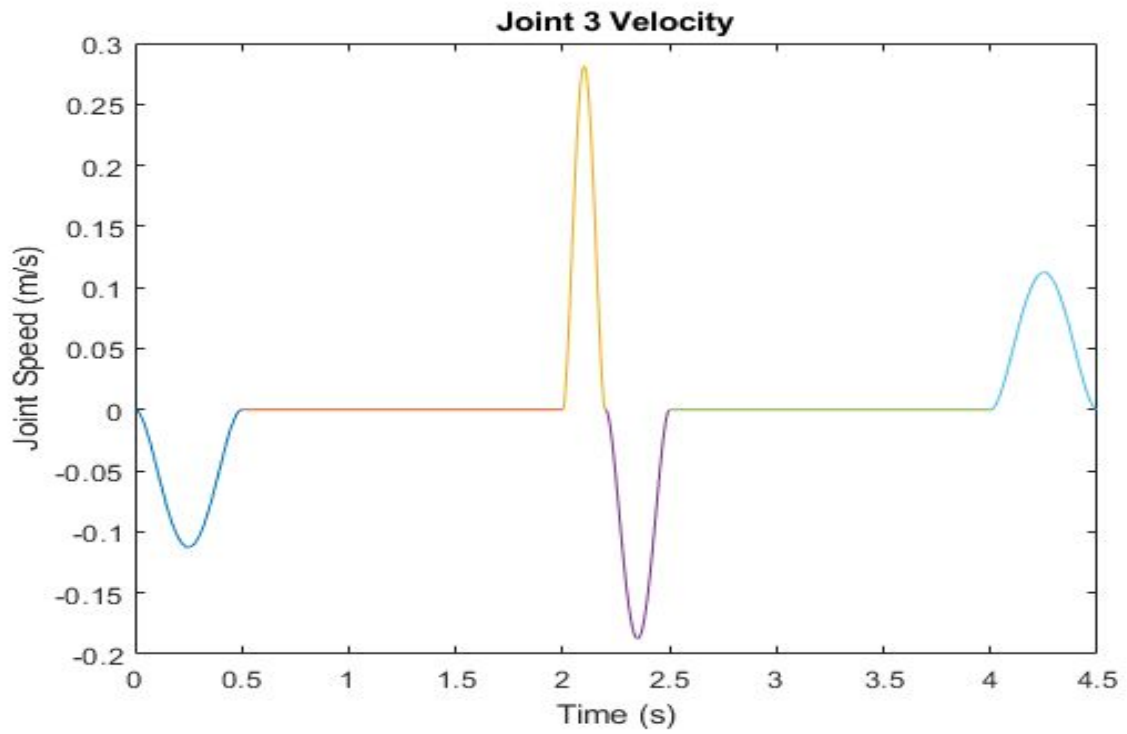


Fig. 14. Joint 3 Velocity

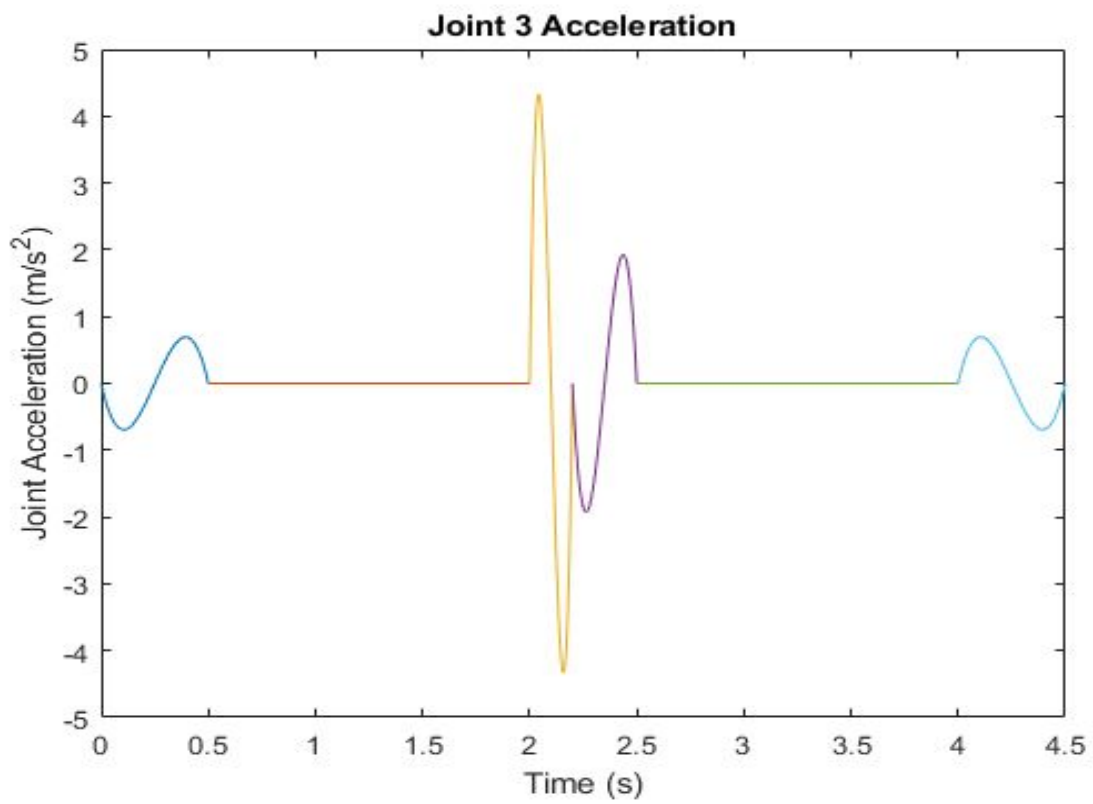


Fig. 15. Joint 3 Acceleration



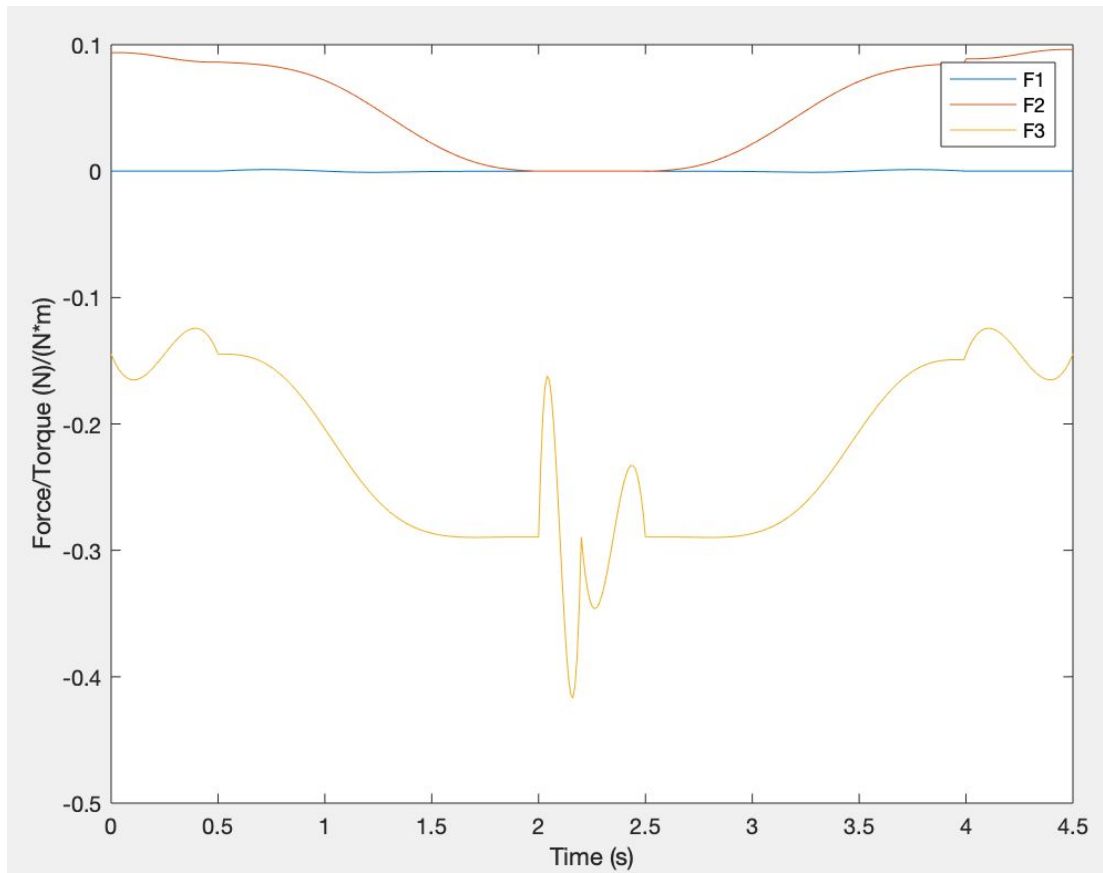


Fig. 16. Force/Moment graph of joints 1,2, and 3

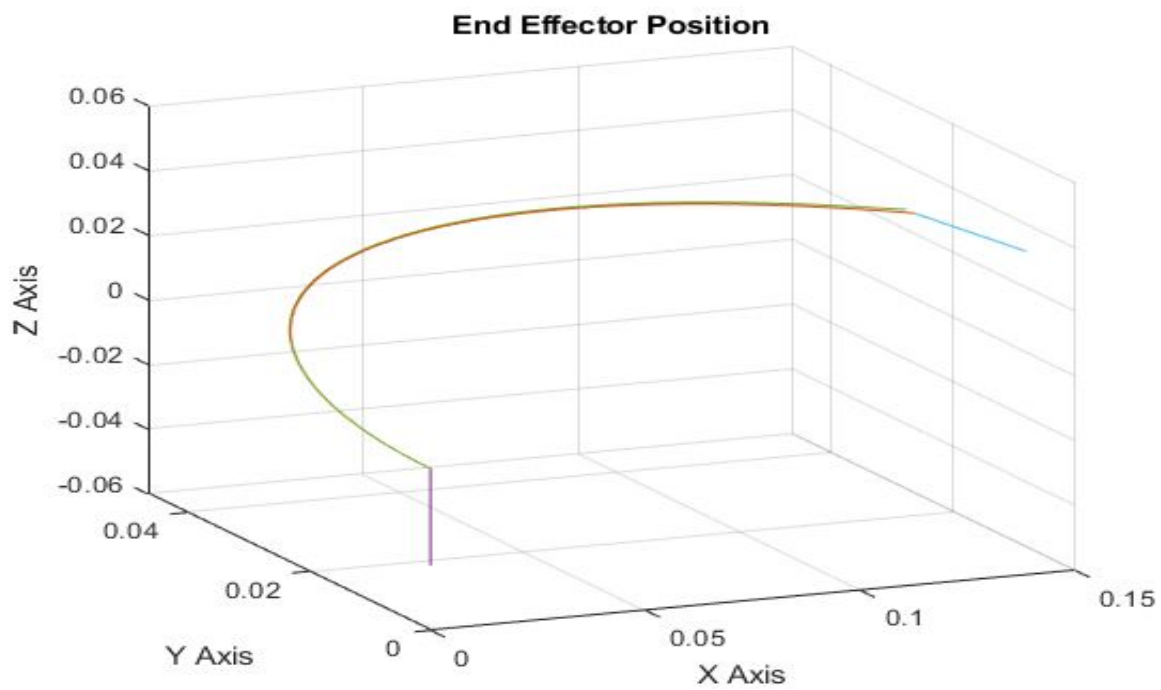


Fig. 17. Path of the End Effector in 3D Space

## Appendix: Matlab Code

The code below was used to determine the dynamics model of the RRP manipulator and the force/moment graph for the trajectory chosen in the previous section:

```
%% Input DH param, create DH tables, and transformation matrices

clear
clc

joints = input('Enter the type of serial robot to determine Jacobian(RRR, PRP, etc.): ','s');
n = strlength(joints);

DHt = cell(n,4); % initialize DH table
for i=1:n % create DH table row by row

    fprintf('\nFor row %d of the DH table (can be numeric or symbolic):\n\n',i)
    d_t = input('Enter the value for d: ','s');
    th_t = input('Enter the value for theta in degrees: ','s');
    a_t = input('Enter the value for a: ','s');
    alf_t = input('Enter the value for alpha in degrees: ','s');
    DHt(i,:) = {d_t th_t a_t alf_t};
end

DHt = string(DHt);

%Initialization
sz = size(DHt);
d = sym('d%d',[1 n]); % lists syms d1 to dn
th = sym('th%d',[1 n]); % lists syms th1 to thn
a = sym('a%d',[1 n]); % lists syms a1 to an
alf = sym('alf%d',[1 n]); % lists syms alf1 to alfn
q=sym('q',[n,1],'real'); % joint variables

DHtd = str2double(DHt); % converts string to double, strings become NaN
d_d = DHtd(1:n,1);
th_d = deg2rad(DHtd(1:n,2)); % convert deg to rad for the DH matrix function
a_d = DHtd(1:n,3);
alf_d = deg2rad(DHtd(1:n,4));
T=sym('T',[4,4],'real');
Ti = sym(zeros(4,4)); % initialize T (contains T matrices between frames)
T0n = sym(eye(4,4)); % initialize T0n (contains T matrices of frames WRT 0 frame)

for i=1:n % runs loop for DH matrix and jacobian components
    % checks if values in DH are numbers to know what values to sub
    if (joints(i)=='P' || joints(i)=='p')
        d(i) = q(i);
        fprintf('hey');
    elseif (isnan(d_d(i)) == 0) % if input is a number then store in corresponding variable
        d(i) = d_d(i);
    end

    if (joints(i)=='R' || joints(i)=='r')
        th(i)=q(i);
    elseif (isnan(th_d(i)) == 0)
        th(i) = th_d(i);
    end
end
```

```

end

if isnan(a_d(i)) == 0
    a(i) = a_d(i);
end

if isnan(alf_d(i)) == 0
    alf(i) = alf_d(i);
end

% creates transformation matrices
Ti(:,i) = DHmatrix(d(i),th(i),a(i),alf(i)); % T matrix
T0n(:,i) = simplify(T0n(:,i)*Ti(:,i)); % multiplies T0 to Tn saving between matrices
T0n(:,i+1) = T0n(:,i); %To be used in next iteration

end

T(:,1) = eye(4,4); % First element is T00
for i=1:n
    T(:,i+1)=T0n(:,i);
end

%%% Variables and constants

I=sym('I',[4,4],'real');
U=sym('U',[4,4],'real');
UU = sym('UU',[4,4],'real');
M=sym('M',[1,1],'real');
Gam=sym('Gam',[1,1],'real');
C = sym('C',[1,1],'real');
G = sym('G',[1,1],'real');
F = sym('F',[n,1],'real');

Qr=[0 -1 0 0; 1 0 0 0; 0 0 0 0; 0 0 0 0];
Qp=[0 0 0 0; 0 0 0 0; 0 0 0 1; 0 0 0 0];

mass=sym('m',[n,1],'real');
mass(1) = 0.15;
mass(2) = 0.122;
mass(3) = 0.0295;

dq=sym('dq',[n,1],'real');
ddq=sym('ddq',[n,1],'real');

% gravity stuff
gx = 0;
gy = 0;
gz = -9.81; % m/s^2 wrt global frame F0
g = [gx,gy,gz,0];

Q = zeros (4,4,n);

% Creates Q arrays depending on the joint
for i=1:n
    % joints char array
    if joints(i)=='R' || joints(i)=='r'
        Q(:,i)=Qr;
    else
        Q(:,i)=Qp;
    end
end

```

```

end

% local centre of mass vectors (** change/add depending on robot)
rbar(:,1) = [0.06238;0;-0.00062;1]; %vector from local sys to mass of the 1st link
rbar(:,2) = [0.000767;0;0.059;1];
rbar(:,3) = [0;0;-0.0297;1];

% Inertial matrices (** Change/add depending on robot)
I(:,1)=[0.000177, 0, 0.00000533, 0.06238*mass(1);
        0, 0.0000059, 0, 0;
        0.00000533, 0, 0.0000058, -0.00062*mass(1);
        0.06238*mass(1), 0, -0.00062*mass(1), mass(1)];

I(:,2)=[0.000004745, 0, -0.00000502, 0.000767*mass(2);
        0, 0.000005745, 0, 0;
        -0.00000502, 0, 0.000156, 0.059*mass(2);
        0.000767*mass(2), 0, 0.059*mass(2), mass(2)];

I(:,3) = [0.000000615, 0, 0, 0;
          0, 0.000000805, 0, 0;
          0, 0, 0.00002307, -0.0297*mass(3);
          0, 0, -0.0297*mass(3), mass(3)];

% --- Begin calculation of required matrices for dynamics ---

% U matrix, i = rows, j = columns 4 x 4 matrices placed within an i x j
% matrix
for i=1:n
    for j=1:n

        % when n = 1, only executes j = 1, not j = 2, see condition for U matrix in sheet
        % j <= i
        if j<=i
            % starts at (j-1), aka 0. j here = j-1. for i, it is i+1
            % (same with j) for transformation matrix only!
            % start at T01 (T 0 to j-1)
            % Q1 = Q(:,1), T01 = T(:,2)
            U(:,i,j)=simplify(T(:,j)*Q(:,j)*inv(T(:,j))*T(:,i+1));
        % j > i
        else
            U(:,i,j)=zeros(4,4);
        end
    end
end

% M matrix i = row, k = column note that this is just a ixk matrix with
% scalar terms M is n x n
for i=1:n
    for k=1:n
        M(:,i,k)=0;

        for j=max(i,k):n
            % prev plus new trace for summation
            M(:,i,k)=M(:,i,k)+trace(U(:,j,k)*I(:,j)*U(:,j,i));
        end
    end
end
end

```

```

M=simplify(M);

% UU matrix
% 3 dimensions: 4 x 4 matrices within each cell of ixjxk array
for i=1:n
    for j=1:n
        for k=1:n

            % i>=k>=j
            if (i>=k && k>=j)
                % starts at (j-1), aka 0. j here = j-1. for i, it is i+1
                % (same with j) for transformation matrix only!
                % Q1 = Q(:,j), T01 = T(:,2)
                UU(:,i,j,k)=simplify(T(:,j)*Q(:,j)*inv(T(:,j))*T(:,k)*Q(:,k)*inv(T(:,k))*T(:,i+1));

            % i>=j>=k
            elseif (i>=j && j>=k)
                UU(:,i,j,k)=simplify(T(:,k)*Q(:,k)*inv(T(:,j))*T(:,k)*Q(:,j)*inv(T(:,j))*T(:,i+1));

            % i<j or i<k
            else
                UU(:,i,j,k)=zeros(4,4);
            end
        end
    end
end

% Gamma matrix

for i=1:n
    for k=1:n
        for m=1:n

            Gam(:,i,k,m)=0;

            for j=max(i,max(k,m)):n
                Gam(:,i,k,m)=Gam(:,i,k,m)+trace(UU(:,j,k,m)*I(:,j)*U(:,j,i));
            end
        end
    end
end

% C matrix, C is n x n
for i=1:n
    for j=1:n
        C(:,i,j)=0;
        for k=1:n
            C(:,i,j)=C(:,i,j)+Gam(:,i,j,k)*dq(k);
        end
    end
end

C=simplify(C);

% G matrix, should be n x 1
for i=1:n
    G(:,i)=0;
    for j=1:n
        G(:,i)=G(:,i)+mass(j)*g*U(:,j,i)*rbar(:,j);
    end
end

```

```

end
G(:,i)=-1*G(:,i);
end

G=simplify(G);

% Final dynamics model: Forces and torques (this vec is n x 1)

for i=1:n

    F(i) = 0;

    for k=1:n
        F(i)=F(i)+ddq(k)*M(:,i,k)+dq(k)*C(:,i,k);
    end

    F(i) = F(i)+G(:,i);
end

F= vpa(simplify(F,4))

%% Plug in trajectory into F to find force at that time
clear
clc

syms q1 q2 q3 dq1 dq2 dq3 ddq1 ddq2 ddq3 t

F1 = 0.0001115*ddq1 - 5.02e-6*dq2^2*sin(q2) - 0.0004381*dq1*dq3 - 0.0008761*ddq1*q3 - 8.79e-5*ddq1*cos(2.0*q2) + 0.01475*ddq1*q3^2 + 5.02e-6*ddq2*cos(q2) - 0.01475*ddq1*q3^2*cos(2.0*q2) + 2.51e-6*dq1*dq2*sin(q2) + 2.51e-6*dq1*dq2*sin(2.0*q1 - 1.0*q2) - 4.395e-5*dq1*dq2*sin(2.0*q1 - 2.0*q2) + 0.01475*dq1*dq3*q3 + 0.0004381*dq1*dq3*cos(2.0*q2) + 0.0008762*ddq1*q3*cos(2.0*q2) + 4.715e-5*dq1*dq2*sin(2.0*q1) + 0.0001319*dq1*dq2*sin(2.0*q2) - 0.01475*dq1*dq3*q3*cos(2.0*q2) - 0.0004381*dq1*dq2*q3*sin(2.0*q1) - 0.001314*dq1*dq2*q3*sin(2.0*q2) - 0.0008761*dq1*dq3*cos(q1)*sin(q2)^2 + 0.007375*dq1*dq2*q3^2*sin(2.0*q1) + 0.02213*dq1*dq2*q3^2*sin(2.0*q2) + 0.0004381*dq1*dq2*q3*sin(2.0*q1 - 2.0*q2) - 0.007375*dq1*dq2*q3^2*sin(2.0*q1 - 2.0*q2) + 0.0295*dq1*dq3*q3*cos(q1)*sin(q2)^2;
F2 = sin(q2)*(0.2894*q3 + 0.06202) + ddq2*(0.0295*q3^2 - 0.001752*q3 + 0.0001886) + dq2*(0.0295*dq3*q3 - 0.0008761*dq3 + 4.715e-5*dq1*sin(2.0*q1) + 4.395e-5*dq1*sin(2.0*q2) + 4.395e-5*dq1*sin(2.0*q1 - 2.0*q2) + 0.007375*dq1*q3^2*sin(2.0*q1) + 0.007375*dq1*q3^2*sin(2.0*q2) - 0.0004381*dq1*q3*sin(2.0*q1 - 2.0*q2) + 0.007375*dq1*q3^2*sin(2.0*q1 - 2.0*q2) - 0.0004381*dq1*q3*sin(2.0*q1) - 0.0004381*dq1*q3*sin(2.0*q2)) + 5.02e-6*ddq1*cos(q2) - 4.235e-25*dq1^2*sin(2.0*q2)*(3.483e+22*q3^2 - 2.069e+21*q3 + 2.075e+20) - 1.0*dq3*cos(q2)*(0.0295*q3 - 0.0008762)*(dq2 - 1.0*dq1*sin(q1)*sin(q2));
F3 = 0.0295*ddq3 - 0.2894*cos(q2) - 0.0295*dq3*sin(q2)*(dq2 - 1.0*dq1*sin(q1)*sin(q2)) - 6.939e-21*dq1^2*sin(q2)^2*(4.251e+18*q3 - 1.263e+17) - 1.0*dq2*(0.007375*q3 - 0.000219)*(4.0*dq2 - 1.0*dq1 + dq1*cos(2.0*q1) + dq1*cos(2.0*q2) - 1.0*dq1*cos(2.0*q1 - 2.0*q2));

n = 1;

% For loop for 0 to 4.5 seconds
for i=0:0.01:4.5

    % Trajectory for interval 1
    if (i >= 0 && i < 0.5)
        q1t = 0;
        q2t = pi/3;
        q3temp = 0.16-2.4*t^3+7.2*t^4-5.76*t^5;
        q3t = subs(q3temp,t,i);
        dq1t = 0;
        dq2t = 0;
        dq3temp = diff(q3temp,t);
        dq3t = subs(dq3temp,t,i);
    end
end

```

```

ddq1t = 0;
ddq2t = 0;
ddq3temp = diff(dq3temp,t);
ddq3t = subs(ddq3temp,t,i);
end

% Trajectory for interval 2
if (i >=0.5 && i<2.0)
    q1temp = -0.9114+6.2056*t-15.514*t^2+17.0654*t^3-7.757*t^4+1.2411*t^5;
    q1t = subs(q1temp,t,i);
    q2temp = 1.6548-4.1371*t+10.3427*t^2-11.377*t^3+5.1713*t^4-0.8274*t^5;
    q2t = subs(q2temp,t,i);
    q3t = 0.13;
    dq1temp = diff(q1temp,t);
    dq1t = subs(dq1temp,t,i);
    dq2temp = diff(q2temp,t);
    dq2t = subs(dq2temp,t,i);
    dq3t = 0;
    ddq1temp = diff(dq1temp,t);
    ddq1t = subs(ddq1temp,t,i);
    ddq2temp = diff(dq2temp,t);
    ddq2t = subs(ddq2temp,t,i);
    ddq3t = 0;
end

% Trajectory for interval 3
if (i >=2.0 && i<2.2)
    q1t = pi/2;
    q2t = 0;
    q3temp =
-22799.86999879618+54449.99999712930*t-51974.99999726391*t^2+24787.49999869711*t^3-5906.24999969003*t^4+562.49999997052*t^5;
    % q3temp = -22800+54450*t-51975*t^2+24787*t^3-5906.2*t^4+562.5*t^5;
    q3t = subs(q3temp,t,i);
    dq1t = 0;
    dq2t = 0;
    dq3temp = diff(q3temp,t);
    dq3t = subs(dq3temp,t,i);
    ddq1t = 0;
    ddq2t = 0;
    ddq3temp = diff(dq3temp);
    ddq3t = subs(ddq3temp, t,i);
end

% Trajectory for interval 4
if (i >=2.2 && i<2.5)
    q1t = pi/2;
    q2t = 0;
    q3temp = 5237.39851860875-11203.70370389647*t+9574.07407423861*t^2-4085.1851852553*t^3+870.37037038529*t^4-74.07407407534*t^5;
    q3t = subs(q3temp,t,i);
    dq1t = 0;
    dq2t = 0;
    dq3temp = diff(q3temp,t);
    dq3t = subs(dq3temp,t,i);
    ddq1t = 0;
    ddq2t = 0;
    ddq3temp = diff(dq3temp,t);
    ddq3t = subs(ddq3temp,t,i);
end

% Trajectory for interval 5
if (i >=2.5 && i<4.0)

```

```

q1temp = 377.3014-620.5615*t+403.365*t^2-128.7665*t^3+20.1682*t^4-1.2411*t^5;
q1t = subs(q1temp,t,i);
q2temp = -250.4871+413.7077*t-268.91*t^2+85.8443*t^3-13.4455*t^4+0.8274*t^5;
q2t = subs(q2temp,t,i);
q3t = 0.13;
dq1temp = diff(q1temp,t);
dq1t = subs(dq1temp,t,i);
dq2temp = diff(q2temp,t);
dq2t = subs(dq2temp,t,i);
dq3t = 0;
ddq1temp = diff(dq1temp,t);
ddq1t = subs(ddq1temp,t,i);
ddq2temp = diff(dq2temp,t);
ddq2t = subs(ddq2temp,t,i);
ddq3t = 0;
end

% Trajectory for interval 6
if (i >=4.0 && i <=4.5)
    q1t = 0;
    q2t = pi/3;
    q3temp = -7894.9+9331.2*t-4406.4*t^2+1039.2*t^3-122.4*t^4+5.76*t^5;
    q3t = subs(q3temp,t,i);
    dq1t = 0;
    dq2t = 0;
    dq3temp = diff(q3temp,t);
    dq3t = subs(dq3temp,t,i);
    ddq1t = 0;
    ddq2t = 0;
    ddq3temp = diff(dq3temp,t);
    ddq3t = subs(ddq3temp,t,i);
end

force1(n) = subs(F1,[q1,q2,q3,dq1,dq2,dq3,ddq1,ddq2,ddq3],[q1t,q2t,q3t,dq1t,dq2t,dq3t,ddq1t,ddq2t,ddq3t]);
force2(n) = subs(F2,[q1,q2,q3,dq1,dq2,dq3,ddq1,ddq2,ddq3],[q1t,q2t,q3t,dq1t,dq2t,dq3t,ddq1t,ddq2t,ddq3t]);
force3(n) = subs(F3,[q1,q2,q3,dq1,dq2,dq3,ddq1,ddq2,ddq3],[q1t,q2t,q3t,dq1t,dq2t,dq3t,ddq1t,ddq2t,ddq3t]);

time(n) = i;
n = n + 1;
end

figure
plot (time,force1);
hold on
plot (time,force2);
hold on
plot (time,force3);
xlabel('Time (s)')
ylabel('Force/Torque (N)/(N*m)')
legend('F1','F2','F3')

%% Test Plug in trajectory and graph force of joints required for trajectory
clear
clc

syms x y z a b c

F = x + y
x = a + b

```



```

F = subs (F,x)

for i=1:100

    time(i) = i+100;
    GraphF(i) = subs(F,[ a,b,y],[i,10,15]);

end

plot(time,GraphF);

%% See the M matrix and test
Mmatrix=sym('Mmatrix',[3,3],'real');

for i=1:3
    for j=1:3
        Mmatrix(i,j)= M(:,i,j);
    end
end

vpa(simplify(Mmatrix),4)

%%
m_dot=0;
for i=1:n

    m_dot = m_dot+diff(M(:,i,2),q(i))*dq(i)

end

x = [1 2];
check = simplify(x*(m_dot-2*C(:,1,2))*x')

%% Symbolic DH matrix function
function Mdh = DHmatrix(d,theta,a,alpha)
% Note: uses trig(sym(angle)) to produce 0 when required as cases involving
% cos(pi/2) produce a non-zero answer when it should be 0
Mdh = [cos(sym(theta)) -sin(sym(theta))*cos(sym(alpha)) sin(sym(theta))*sin(sym(alpha)) a*cos(sym(theta));
        sin(sym(theta)) cos(sym(theta))*cos(sym(alpha)) -cos(sym(theta))*sin(sym(alpha)) a*sin(sym(theta));
        0 sin(sym(alpha)) cos(sym(alpha)) d;
        0 0 0 1];
end

```