WESTERN UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

ECE 9053B

Robot manipulators

ECE 9053B Project Report

KUMARAJEEVA ELAVARASAN

251124570

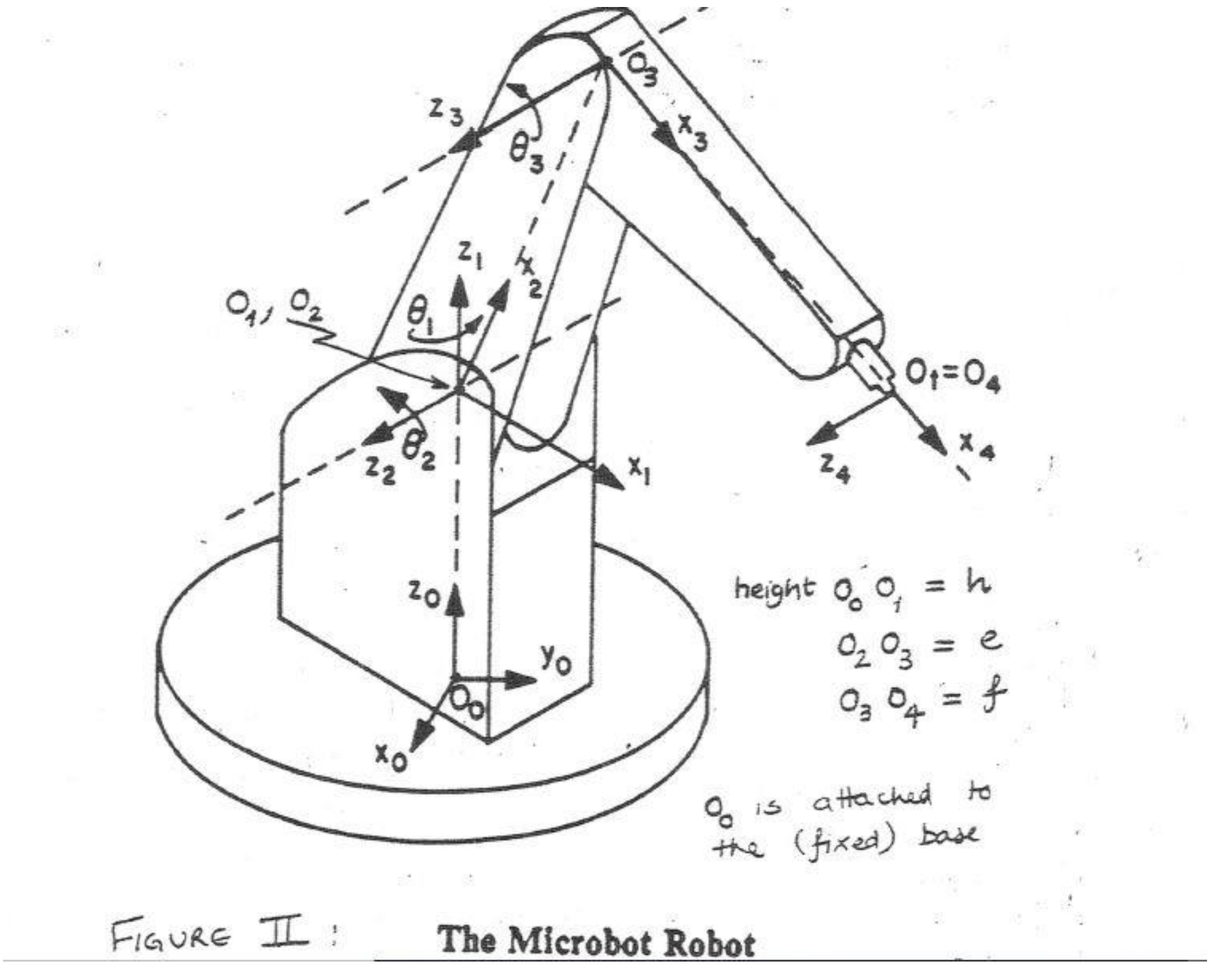

Fig 1: The Microbot Robot

PART 1

D-H PARAMETERS:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	h	θ_1
2	90^o	0	0	θ_2
3	0	e	0	θ_3
4	0	f	0	0

whereas h = 2, e = 3 and f = 4

FORWARD KINEMATICS:

Forward kinematics is the method of obtaining the position and orientation of the manipulators from Denavit-Hartenberg parameters. The D-H parameters are given. Our objective is to compute the homogeneous transformation which relates wrist frame to base frame. The wrist frame is frame {4} in our case.

The generalized transformation matrix is given by:

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

OUTPUT:

Enter the values of i : 4
Enter the values of alphaminusone: [0 90 0 0]
Enter the values of aminusone: [0 0 3 4]
Enter the values of d: [2 0 0 0]

$$ {}^{0}_{1}T = \begin{bmatrix} -0.9487 & -0.31615 & 0 & 0 \\ 0.31615 & -0.9487 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{2}T = \begin{bmatrix} -0.2589 & 0.9127 & 0.3161 & 0 \\ 0.0863 & -0.3041 & 0.9487 & 0 \\ 0.9620 & 0.2730 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{3}T = \begin{bmatrix} 0.9443 & -0.0916 & 0.3161 & -0.7769 \\ -0.3147 & -0.0305 & 0.9487 & 0.2589 \\ -0.0965 & -0.9953 & 0 & 4.8861 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{0}_{4}T = \begin{bmatrix} 0.9443 & -0.0916 & 0.3161 & 3.0003 \\ -0.3147 & 0.0305 & 0.9487 & -0.9999 \\ -0.0965 & -0.9953 & 0 & 4.4999 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

We are getting $P_x = 3$, $P_y = -1$ and $P_z = 4.5$

PART - 2

INVERSE KINEMATICS:

It is the method of finding joint angles from the given position values and in some cases with the given transformation matrix relating the wrist frame to the base frame. Our task is to get the expression of position of the origin of frame{4} relative to frame {0} and to calculate the joint angles for different solutions. We verify the solutions using forward kinematics.

Expression of position of the origin of frame{4} relative to frame {0} in Microbot shown in Fig.1

$$0_{P_{4ORG}} = \begin{bmatrix} c_1(fc_{23} + ec_2) \\ s_1(fc_{23} + ec_2) \\ fs_{23} + es_2 + h \end{bmatrix}$$

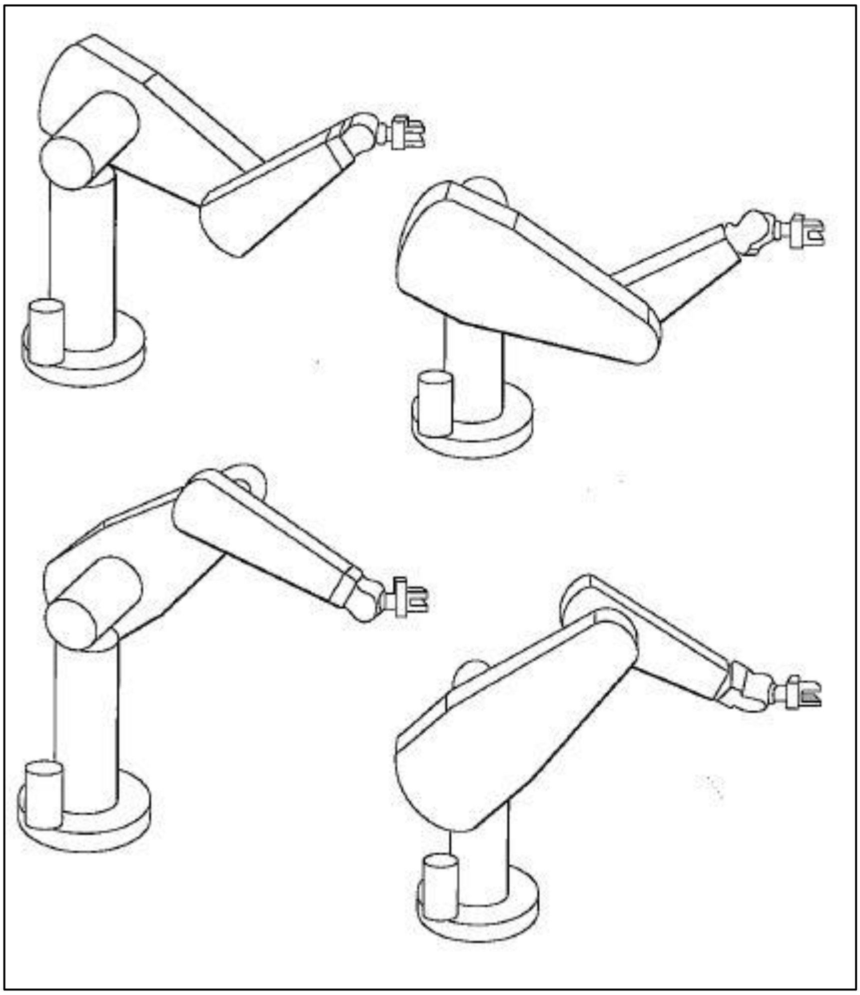

Fig.2 Four different possible solutions

$\theta_1 = arctan2\ (Pos\_y, Pos\_x)$

whereas Pos_x, Pos_y and Pos_z represents the positions of the end effector. The other possible value of theta1 is obtained by rotating the given joint in the opposite direction and is given by,

$\theta_1 = arctan2(-Pos\_y\ , -Pos\_x)$

The links two and three acts as a mere two link manipulator and works in “elbow up” and “elbow down” configuration.

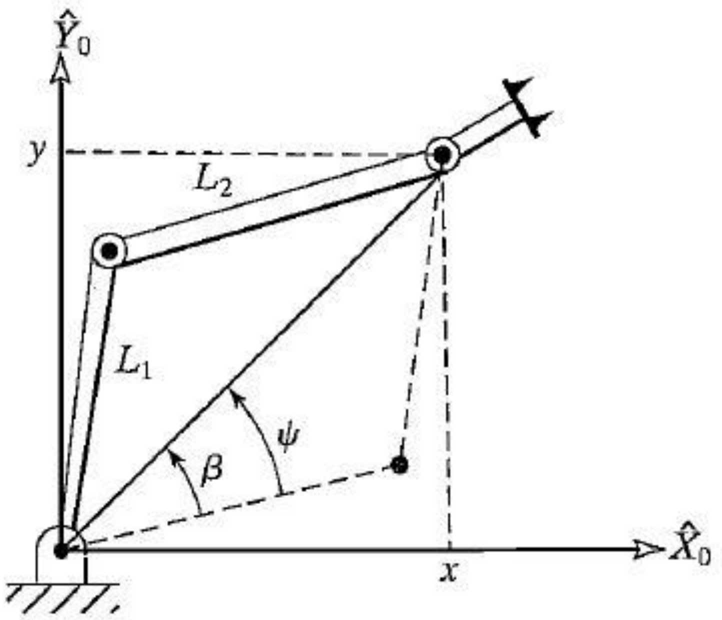

Fig.3 “Elbow up” and “Elbow down” configurations

By Cosine law, we get

$$q_1 = \sqrt{Pos\_x^2 + Pos\_y^2}$$

$$q_2 = Pos\_z - h$$

$$\cos(\beta) = \frac{f^2 - e^2 - q_1^2 - q_2^2}{-2\, e\sqrt{q_1^2 + q_2^2}}$$

$$\sin(\beta) = \sqrt{1 - (\cos(\beta)^2}$$

$$\psi_1 = \mathrm{arctan2}\ (q_2\ , q_1)$$

$$\psi_2 = \mathrm{arctan2}\ (\sin(\beta)\ , \cos(\beta))$$

$$\theta_2 = \psi_1 + \psi_2$$

$$\cos(\varphi) = \frac{q_1^2 + q_2^2 - e^2 - f^2}{-2ef}$$

$$\sin(\varphi) = \sqrt{1 - (\cos\,(\varphi)^2}$$

$$\varphi = arctan2(\sin(\varphi)\,, \cos(\varphi))$$

$$\theta_3 = \,\pm\,(180 - \varphi)$$

We are solving them using geometric method analysing the front and top view considering four cases namely elbow up right, elbow down right, elbow up left and elbow down left. We get four different solutions which are verified by the forward kinematics to give the same desired positions.

OUTPUT:

possiblesolutions:

```
[  -18.4349    105.8439   -111.3819               % solution 1

   -18.4349    -29.1863    111.3819               % solution 2

   161.5651     74.1561    111.3819               % solution 3

   161.5651   -150.8137   -111.3819 ]             % solution 4
```

VERIFICATION IN FORWARD KINEMATICS:

For Solution 1:

$$ {}^0_4T = \begin{bmatrix} 0.9443 & 0.0916 & -0.3162 & 3.0000 \\ -0.3148 & -0.0305 & -0.9487 & -1.0000 \\ -0.0965 & 0.9953 & 0 & 4.5000 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For Solution 2:

$$ {}^0_4T = \begin{bmatrix} 0.1288 & -0.9399 & -0.3162 & 3.0000 \\ -0.0429 & 0.3133 & -0.9487 & -1.0000 \\ 0.9907 & 0.1358 & 0 & 4.5000 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For Solution 3:

$$ {}^{0}_{4}T = \begin{bmatrix} 0.9443 & -0.0916 & 0.3162 & 3.0000 \\ -0.3148 & 0.0305 & 0.9487 & -1.0000 \\ -0.0965 & -0.9953 & 0 & 4.5000 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

For Solution 4:

$$ {}^{0}_{4}T = \begin{bmatrix} 0.1288 & 0.9399 & 0.3162 & 3.0000 \\ -0.0429 & -0.3133 & 0.9487 & -1.0000 \\ 0.9907 & -0.1358 & 0 & 4.5000 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

It is interesting to note that all the possible solutions gives the value of $P_x = 3$, $P_y = -1$ and $P_z = 4.5$ when fed as an input to the forward kinematics code. All possible solutions are obtained by applying cosine law to the manipulators.

PART-3

Angular velocity $\boldsymbol{\omega}$ of link i+1 with respect to frame {i+1} is given by

$${}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\ {}^{i}\omega_{i} + \dot{\theta}_{i+1}\ {}^{i+1}\hat{Z}_{i+1}$$

$$\dot{\theta}_{i+1}\ {}^{i+1}\hat{Z}_{i+1} = {}^{i+1}\begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{i+1} \end{bmatrix}$$

Linear velocity $\boldsymbol{v}$ is given by

$${}^{i+1}v_{i+1} = {}^{i+1}_{i}R\ ({}^{i}v_{i} + {}^{i}\omega_{i} \times {}^{i}P_{i+1})$$

Formulas:

$${}^{1}\omega_{1} = {}^{1}_{0}R\ {}^{0}\omega_{0} + \dot{\theta}_{1}\ {}^{1}\hat{Z}_{1}$$

$${}^{1}v_{1} = {}^{1}_{0}R\ ({}^{0}v_{0} + {}^{0}\omega_{0} \times {}^{0}P_{1})$$

$${}^{2}\omega_{2} = {}^{2}_{1}R\ {}^{1}\omega_{1} + \dot{\theta}_{2}\ {}^{2}\hat{Z}_{2}$$

$${}^{2}v_{2} = {}^{2}_{1}R\ ({}^{1}v_{1} + {}^{1}\omega_{1} \times {}^{1}P_{2})$$

$${}^{3}\omega_{3} = {}^{3}_{2}R\ {}^{2}\omega_{2} + \dot{\theta}_{3}\ {}^{3}\hat{Z}_{3}$$

$${}^{3}v_{3} = {}^{3}_{2}R\ ({}^{2}v_{2} + {}^{2}\omega_{2} \times {}^{2}P_{3})$$

Cartesian velocity = $\begin{bmatrix} linear\ velocity \\ angular\ velocity \end{bmatrix}_{6x1}$

PART- 3) 1:

Our task is to compute the Cartesian velocity of the end-effector given the set of joint velocities and D-H parameters of the manipulator. Our program is checked with the manipulator from example 5.3 from the text book.

D-H PARAMETERS:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L1	0	θ_2
3	0	L2	0	θ_3

L1 = L2 = 0.5m

OUTPUT:

omega1wrt1 = [0 0 theta1dot] ^{T}

V1wrt1 = [0 0 0] ^{T}

omega2wrt2 = [0 0 theta1dot + theta2dot] ^{T}

V2wrt2 = [L1*sin(theta2)* theta1dot L1*cos(theta2)*theta1dot 0] ^{T}

omega3wrt3 = [0 0 theta1dot + theta2dot + theta3dot] ^{T}

$$\text{V3wrt3} = \begin{bmatrix} \text{L1} * \sin(\text{theta2}) * \text{theta1dot} \\ \text{L1} * \cos(\text{theta2}) * \text{theta1dot} + \text{L2} * (\text{theta1dot} + \text{theta2dot}) \\ 0 \end{bmatrix}$$

$$\text{Cartesian velocity} = \begin{bmatrix} \text{V3wrt3} \\ \text{omega3wrt3} \end{bmatrix}$$

The values we get for ${}^3\omega_3$ and ${}^3v_3$ matches exactly with the example 5.3 given in the text book.

PART- 3) 2:

Our main objective is to calculate the Cartesian linear velocity of origin of frame{4} of the MICROBOT given in fig.1. Also, $\theta_1 = -18.43°, \theta_2 = -29.18° \, and \, \theta_3 = -111.38°$ and $\dot{\theta}_1 = 35°/s, \dot{\theta}_2 = 20°/s \, and \, \dot{\theta}_3 = 45°/s$

D-H PARAMETERS:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	h	θ_1
2	90^o	0	0	θ_2
3	0	e	0	θ_3
4	0	f	0	0

$$ {}^0_4T = \begin{bmatrix} 0.1288 & 0.9399 & 0.3162 & 3.0000 \\ -0.0429 & -0.3133 & 0.9487 & -1.0000 \\ 0.9907 & -0.1358 & 0 & 4.5000 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

VELOCITIES:

$$ {}^1\omega_1 = {}^1_0R \, {}^0\omega_0 + \dot{\theta}_1 \, {}^1\hat{Z}_1 $$

$$ {}^1v_1 = {}^1_0R \, ({}^0v_0 + {}^0\omega_0 \times {}^0P_1) $$

$$ {}^2\omega_2 = {}^2_1R \, {}^1\omega_1 + \dot{\theta}_2 \, {}^2\hat{Z}_2 $$

$$ {}^2v_2 = {}^2_1R \, ({}^1v_1 + {}^1\omega_1 \times {}^1P_2) $$

$$ {}^3\omega_3 = {}^3_2R \, {}^2\omega_2 + \dot{\theta}_3 \, {}^3\hat{Z}_3 $$

$$ {}^3v_3 = {}^3_2R \, ({}^2v_2 + {}^2\omega_2 \times {}^2P_3) $$

$$ {}^4\omega_4 = {}^4_3R \, {}^3\omega_3 + 0 $$

$$ {}^4v_4 = {}^4_3R \, ({}^3v_3 + {}^3\omega_3 \times {}^2P_3) $$

OUTPUT:

W1_1 = $\begin{bmatrix} 0 \\ 0 \\ 0.6109 \end{bmatrix}$

W2_2 = $\begin{bmatrix} -0.2978 \\ 0.5333 \\ 0.3491 \end{bmatrix}$

W3_3 = $\begin{bmatrix} -0.3881 \\ -0.4718 \\ 1.1345 \end{bmatrix}$

W4_4 = $\begin{bmatrix} -0.3881 \\ -0.4718 \\ 1.1345 \end{bmatrix}$

V1_1 = $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

V2_2 = $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

V3_3 = $\begin{bmatrix} -0.9751 \\ -0.3818 \\ 1.6000 \end{bmatrix}$

V4_4 = $\begin{bmatrix} -0.9751 \\ 4.1561 \\ 0.2870 \end{bmatrix}$

Cartesian_Velocity = $\begin{bmatrix} -0.9751 \\ 4.1561 \\ 0.2870 \\ -0.3881 \\ -0.4718 \\ 1.1345 \end{bmatrix}$

PART – 4

PART 4a

INVERSE DYNAMICS:

The main objective of this part is to calculate the torques from the given positions using Newton-Euler algorithm. The velocities and acceleration is calculated from the given joint position. D-H parameters is taken from the microbot. Velocity is obtained by differentiating the given position with respect to time. Acceleration is obtained by differentiating the given velocity with respect to time.

Recursive Newton-Euler Algorithm:

Outward recursions: (0 to n-1)

${}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_i + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}$ - Angular velocity

${}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_i + {}^{i+1}_{i}R\,{}^{i}\omega_i \times \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} + \ddot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1}$ - Angular acceleration

${}^{i+1}\dot{v}_{i+1} = {}^{i+1}_{i}R\,({}^{i}\dot{\omega}_i \times {}^{i}P_{i+1} + {}^{i}\omega_i \times ({}^{i}\omega_i \times {}^{i}P_{i+1}) + {}^{i}\dot{v}_i$ - Linear acceleration

${}^{i+1}\dot{v}_{C_{i+1}} = {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{C_{i+1}} + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{C_{i+1}}) + {}^{i+1}\dot{v}_{i+1}$ -
(Linear acceleration of centre of mass)

${}^{i+1}F_{i+1} = m_{i+1}\,{}^{i+1}\dot{v}_{C_{i+1}}$

${}^{i+1}N_{i+1} = {}^{C}i+1_{I_{i+1}}\,{}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{C}i+1_{I_{i+1}}\,{}^{i+1}\omega_{i+1}$

Outward recursion are from link1 to link3

Inward recursions: (n to 0)

${}^{i}f_i = {}^{i}_{i+1}R\,{}^{i+1}f_{i+1} + {}^{i}F_i$

${}^{i}f_i = {}^{i}N_i + {}^{i}_{i+1}R\,{}^{i+1}n_{i+1} + {}^{i}P_{C_i} \times {}^{i}F_i + {}^{i}P_{i+1}\,{}^{i}_{i+1}R\,{}^{i+1}f_{i+1}$

$\tau_i = {}^{i}n_i^T\,{}^{i}\hat{Z}_i$

OUTPUT GRAPHS:

Torques:

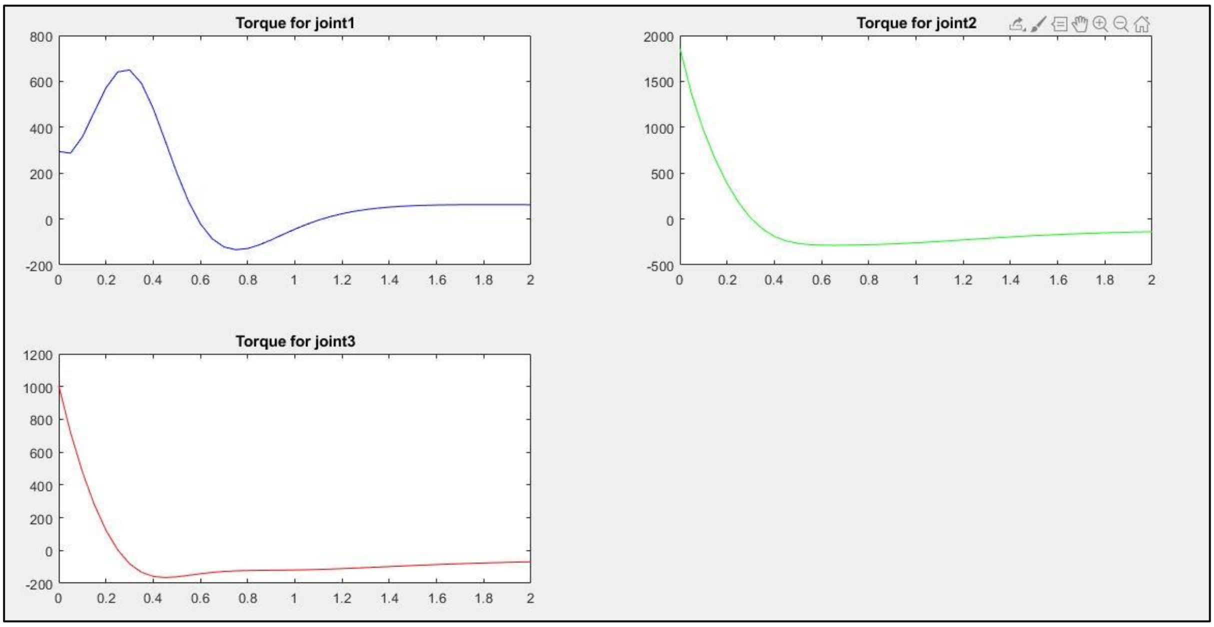

Fig 4. Torques against time

Acceleration for all joints:

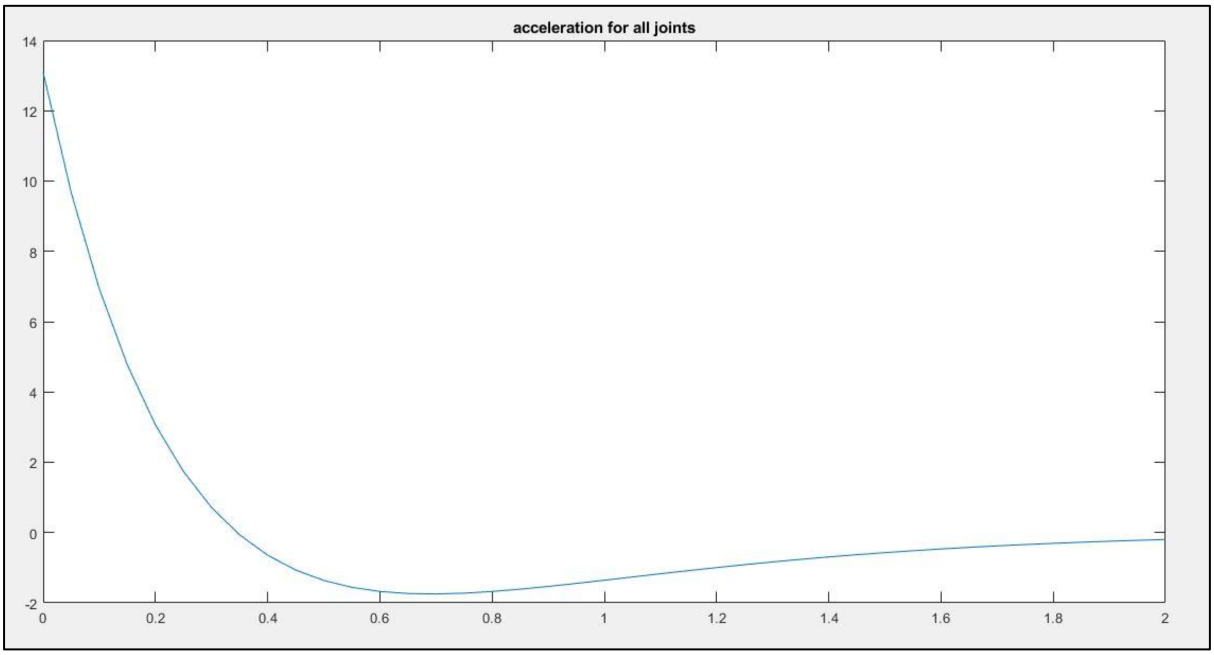

Fig 5. Acceleration of all joints

Velocity for all joints:

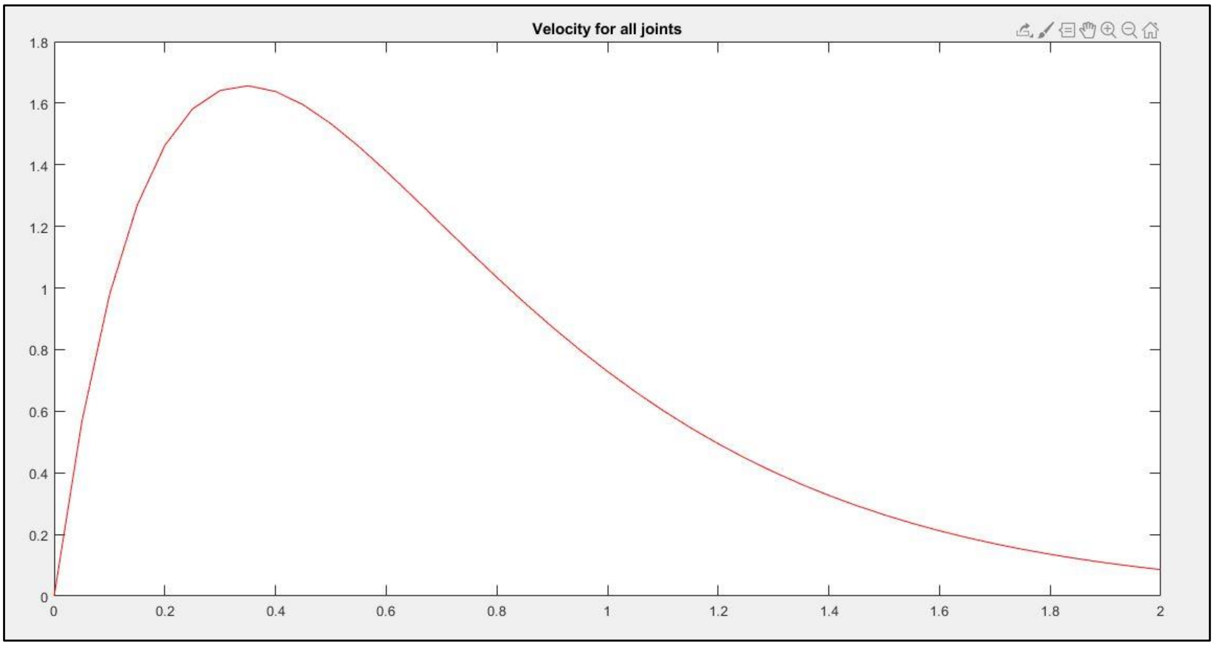

Fig 6. Velocity for all joints

Positions for all joints:

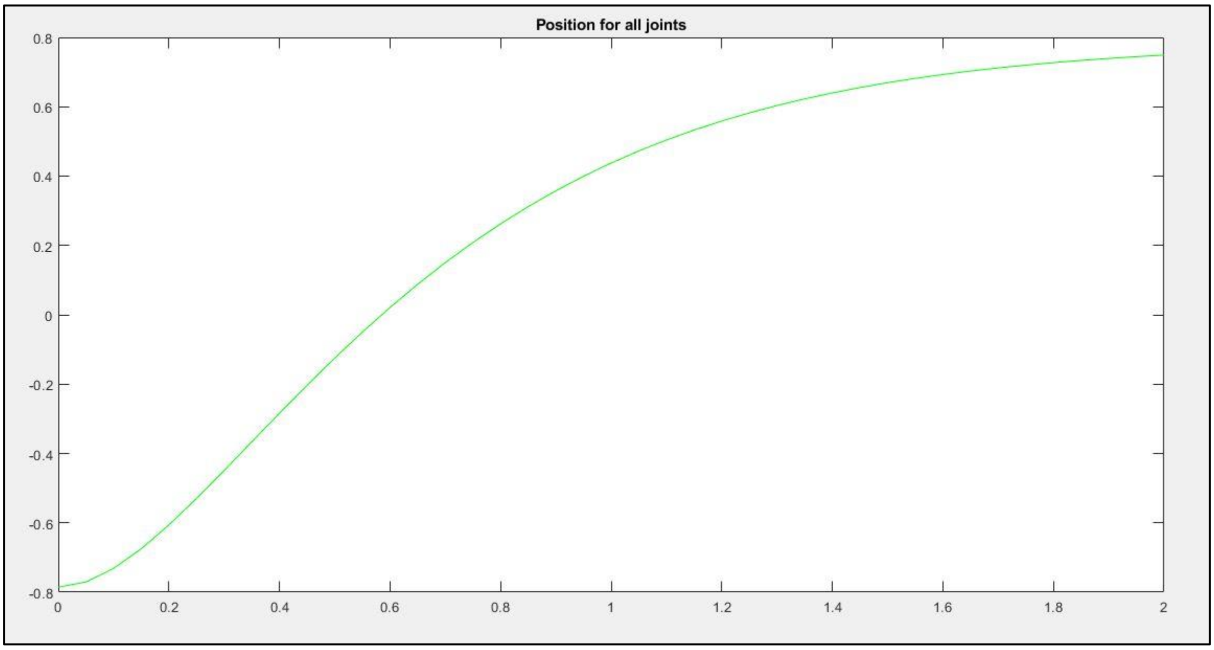

Fig 7. Position for all joints

PART 4b:

FORWARD DYNAMICS:

The main objective of this part is to calculate position, velocity and acceleration from the torques obtained from the inverse dynamics.

$$\boldsymbol{\tau} = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta)$$

First, we are getting the mass matrix. And then we are substituting the values of thetadoubledots as zero so as to get the remaining equation. Lastly, the inverse of torque is multiplied with equation to get the acceleration. Integrating acceleration once gives velocity whereas integrating it twice gives position.

$$\ddot{\boldsymbol{\theta}} = M^{-1}(\theta)\left[\tau - V(\theta, \dot{\theta}) - G(\theta) - F(\theta, \dot{\theta})\right]$$

The acceleration, velocity, position and torques are plotted against time below.

OUTPUT GRAPHS:

ACCELERATION:

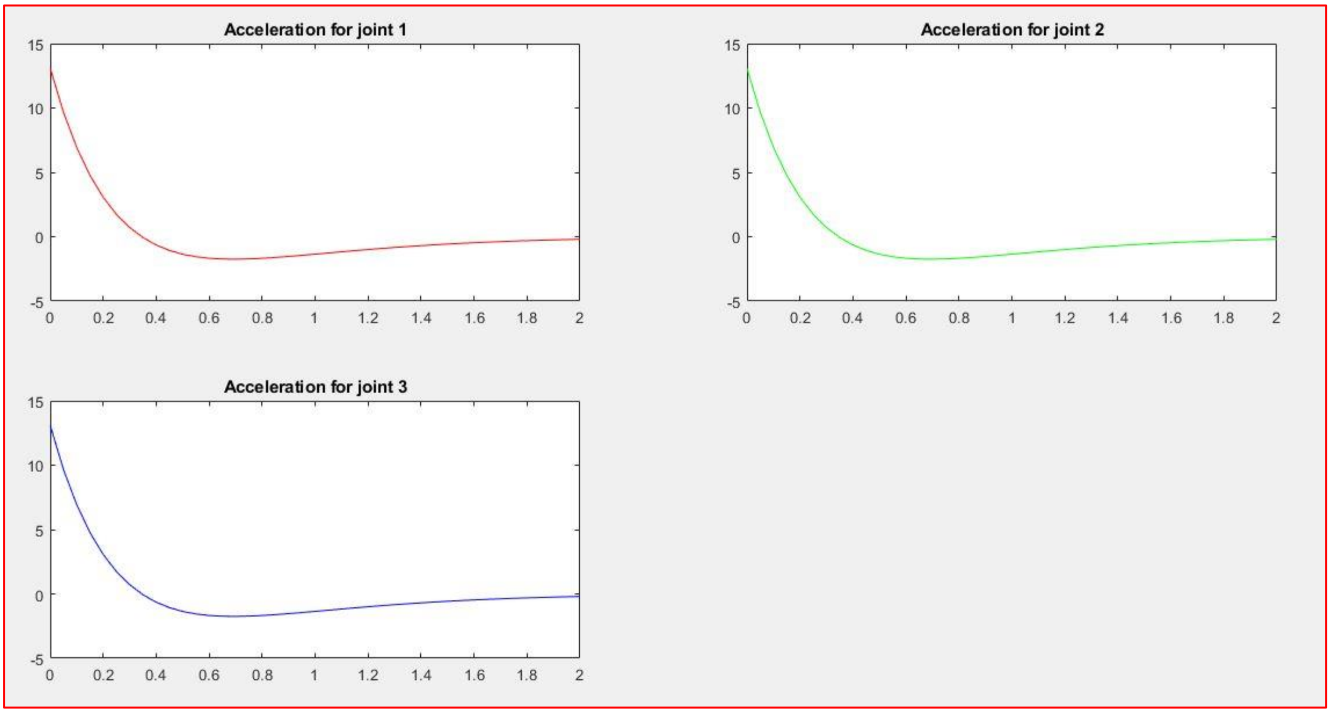

Fig 8. Acceleration for joints 1,2 and 3

VELOCITY:

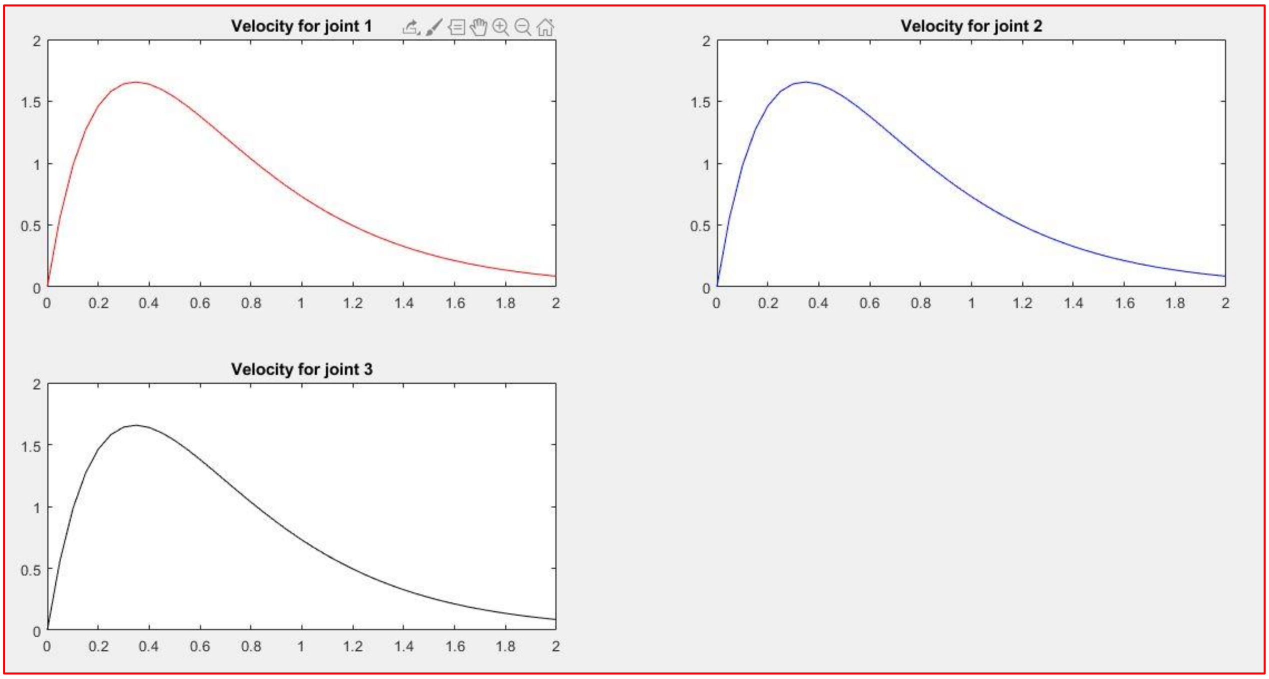

Fig 9. Velocity for joints 1,2 and 3

POSITION:

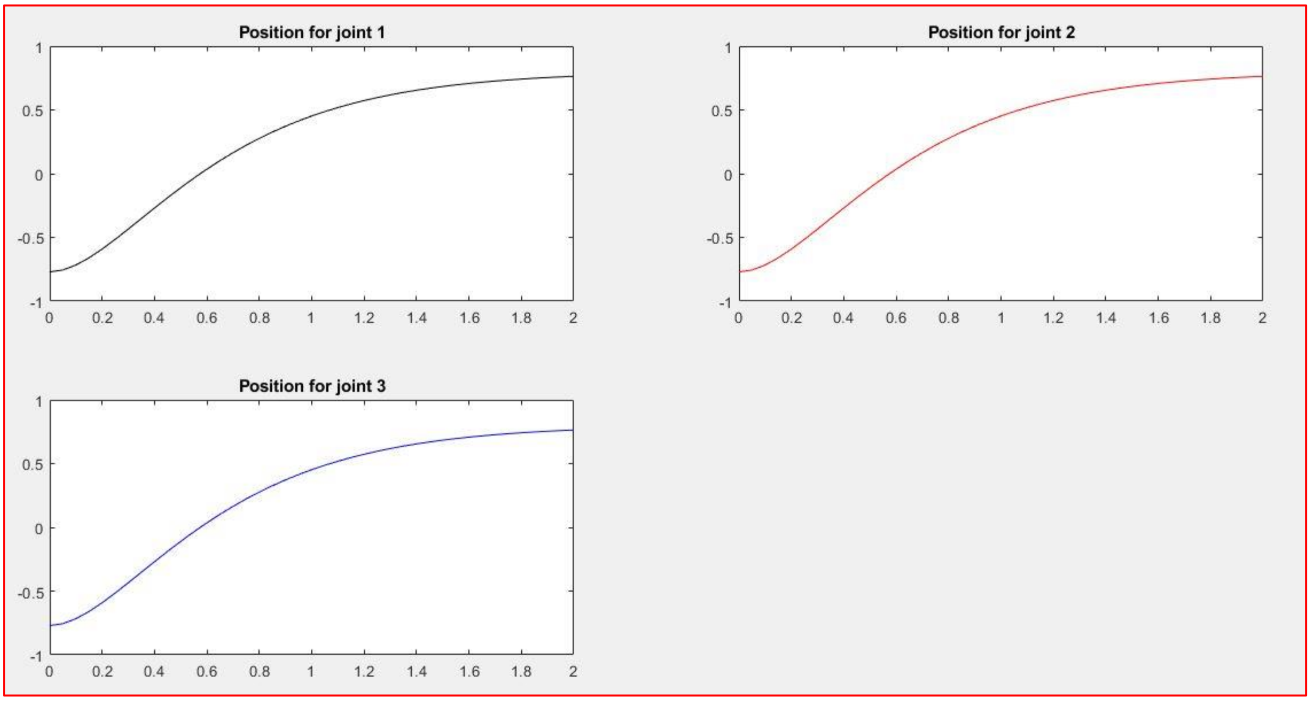

Fig 10. Positions for joints 1,2 and 3

TORQUES:

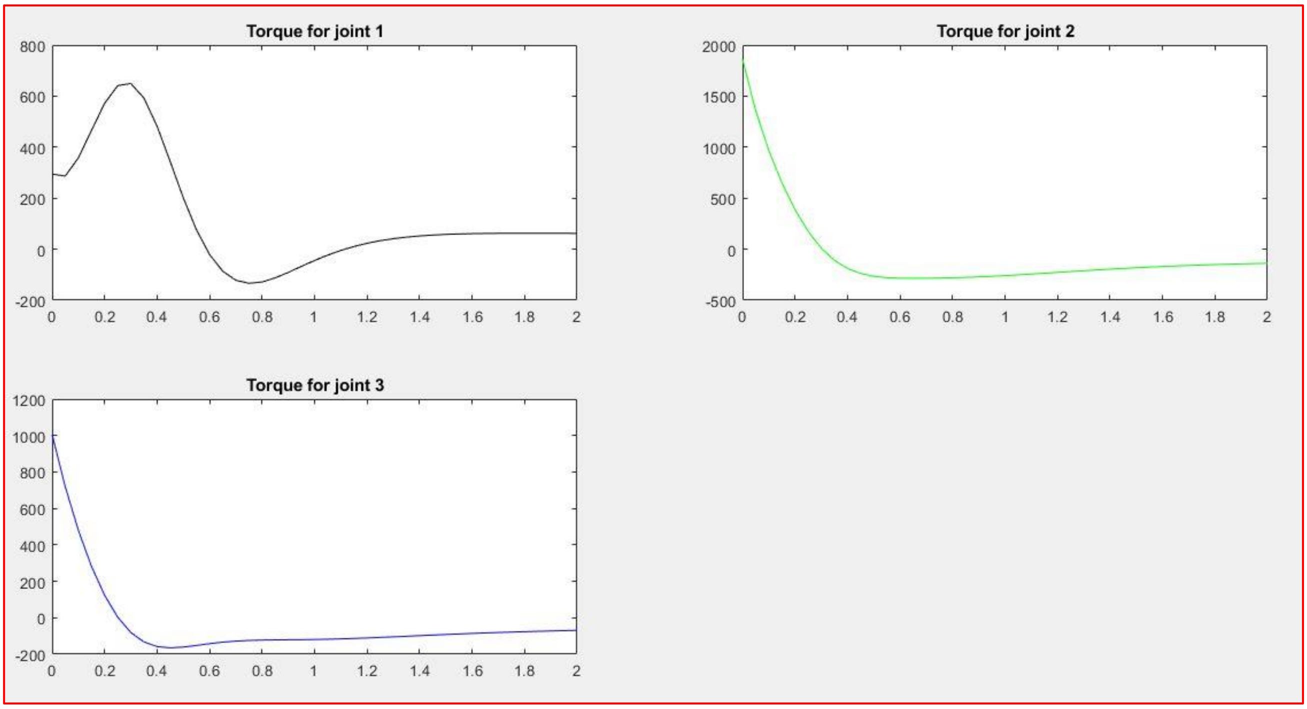

Fig 11. Torques for joints 1,2 and 3

PART 5

Part 5 i):

Cubic Polynomial Segments trajectories :

$a_0 = \theta_0$, $a_1 = \dot{\theta}_0$

$$a_2 = \frac{3(\theta_f - \theta_0) - (2\dot{\theta}_0 + \dot{\theta}_f)(t_f - t_0)}{(t_f - t_0)^2}$$

$$a_3 = \frac{2(\theta_0 - \theta_f) - (\dot{\theta}_0 + \dot{\theta}_f)(t_f - t_0)}{(t_f - t_0)^3}$$

$\boldsymbol{\theta(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3}$ (Position)

$\boldsymbol{\dot{\theta}(t) = a_1 + 2a_2(t - t_0) + 3a_3(t - t_0)^2}$ (Velocity)

$\boldsymbol{\ddot{\theta}(t) = 2a_2 + 6a_3(t - t_0)}$ (Acceleration)

All CPS trajectories are calculated from the above given equations.

OUTPUT:

CUBIC POLYNOMIAL SEGMENTS:

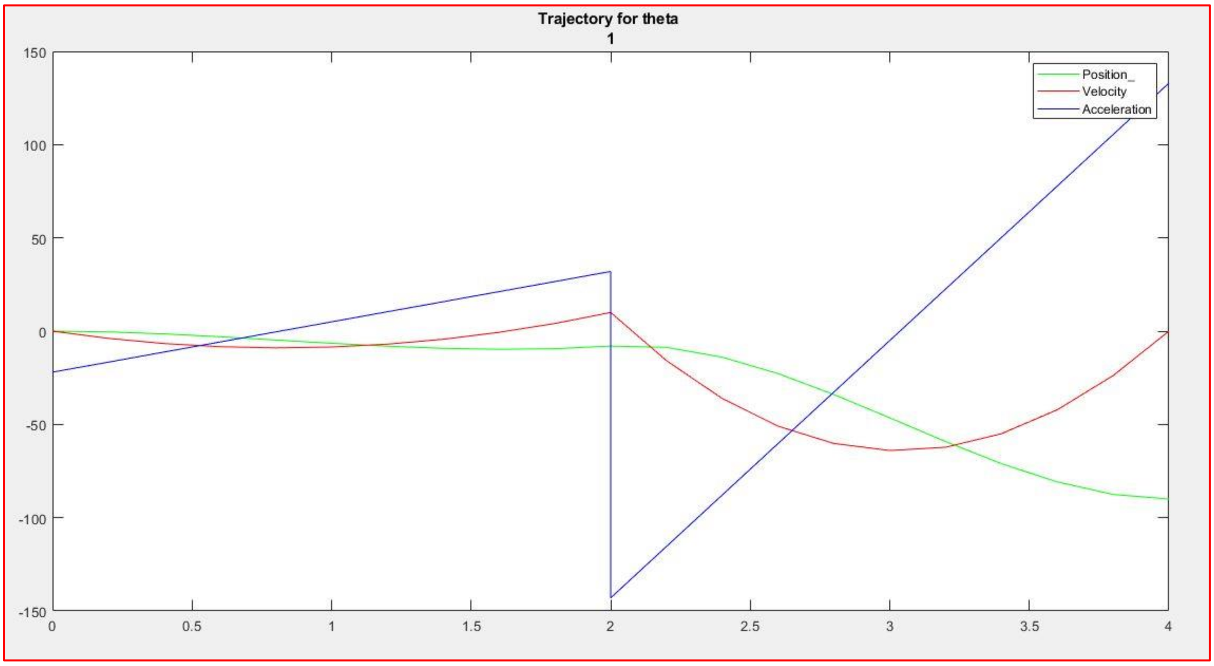

Fig 12. Position, Velocity and acceleration for theta1 (CPS)

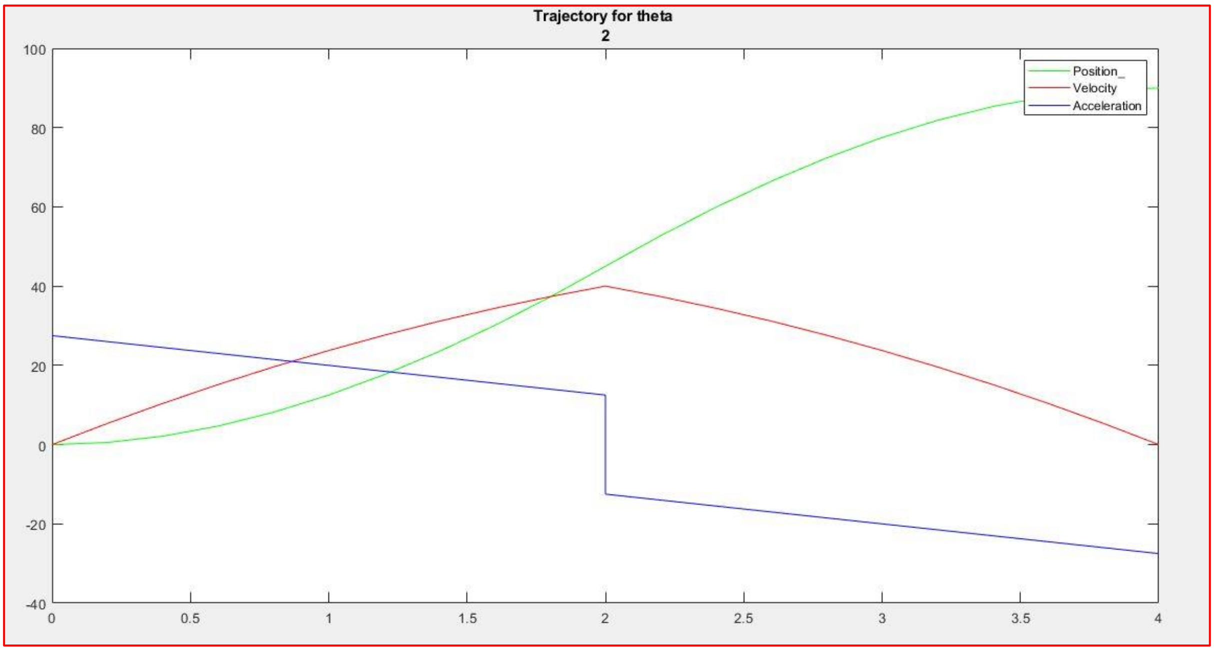

Fig 13. Position, Velocity and acceleration for theta2 (CPS)

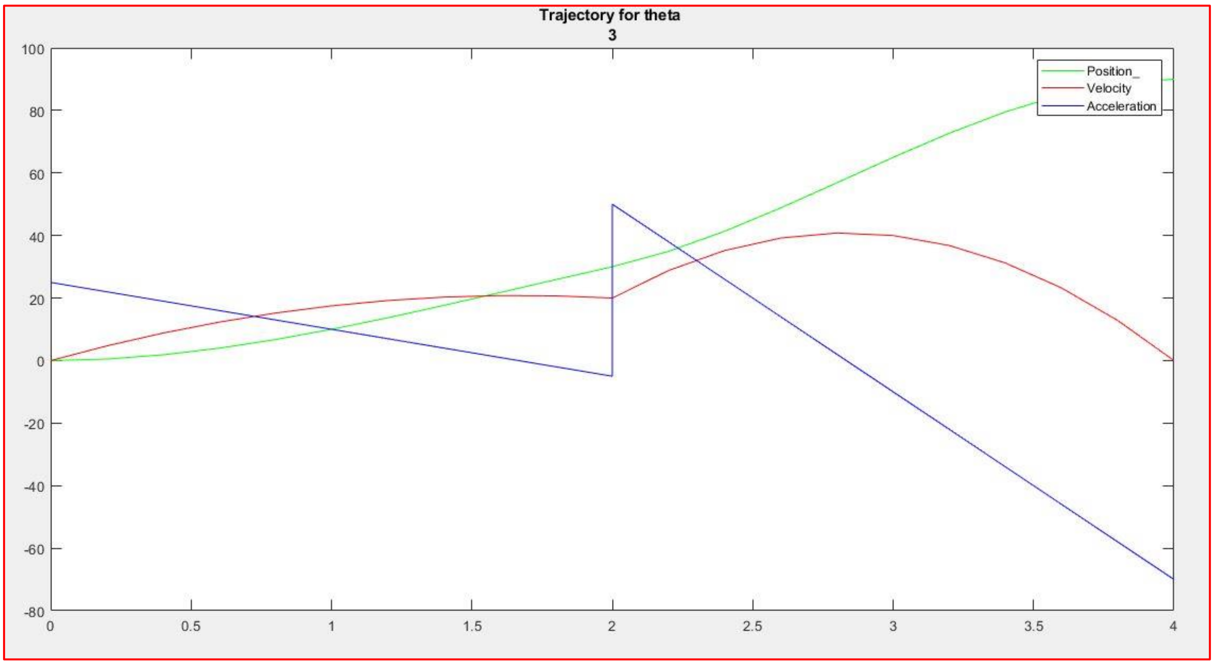

Fig 14. Position, Velocity and acceleration for theta3 (CPS)

Part 5 ii):

Linear Segments with Parabolic Blends trajectory:

$$V = \theta_f - \theta_0 - \frac{1}{2}\frac{t_b(\dot{\theta}_0 - \dot{\theta}_f)}{t_f - (t_b + t_0)}$$

For $0 \le t \le t_b$,

$$\boldsymbol{\theta(t) = \theta_0 + \dot{\theta}_0(t - t_0) + 0.5 * \frac{(V - \dot{\theta}_0)}{t_b}\ (t - t_0)^2}$$

For $t_b \le t \le t_f - t_b$,

$$\boldsymbol{\theta(t) = Vt - t_0 + (\theta_0 + \tfrac{1}{2}\dot{\theta}_0 t_b - \tfrac{1}{2}Vt_b)}$$

For $t_f - t_b \le t \le t_f$,

$$\boldsymbol{\theta(t) = \left[\theta_f - \dot{\theta}_f t_f + \frac{(\dot{\theta}_f - V)t_f^2}{2t_b}\right] + \left[\dot{\theta}_f - \frac{(\dot{\theta}_f - V)t_f}{t_b}\right]t + \frac{(\dot{\theta}_f - V)}{2t_b}t^2}$$

All LSPB trajectories are calculated from the above equations

OUTPUT GRAPHS:

LINEAR SEGMENTS WITH PARABOLIC BLENDS:

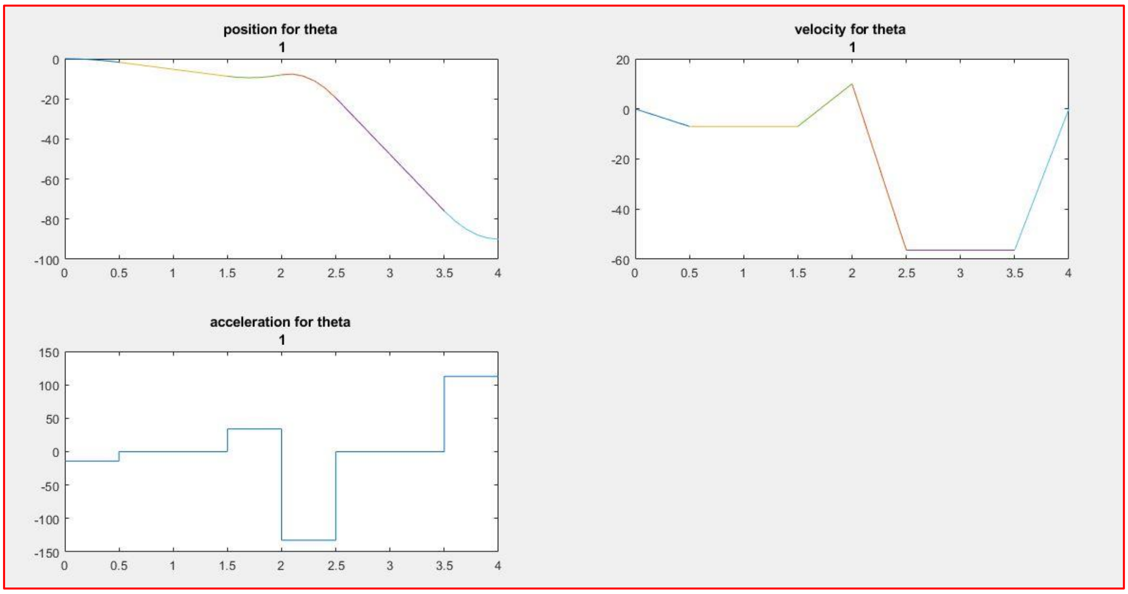

Fig 15. Position, Velocity and acceleration for theta1 (LSPB)

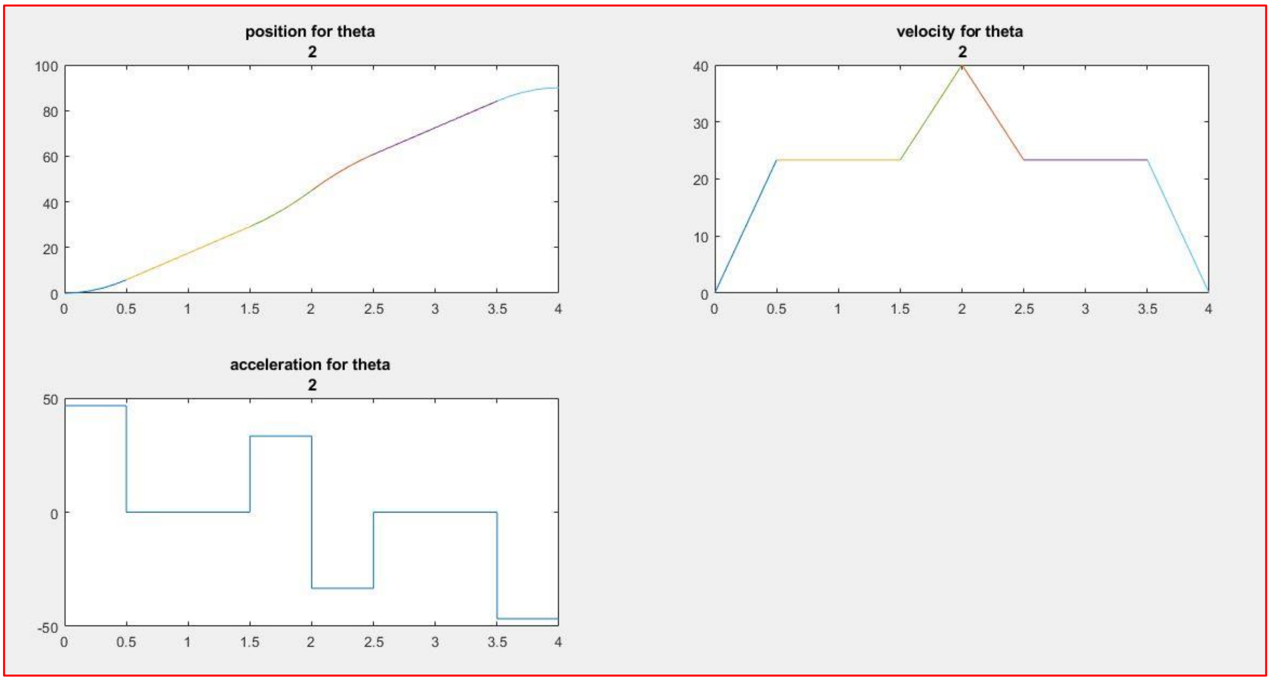

Fig 16. Position, Velocity and acceleration for theta2 (LSPB)

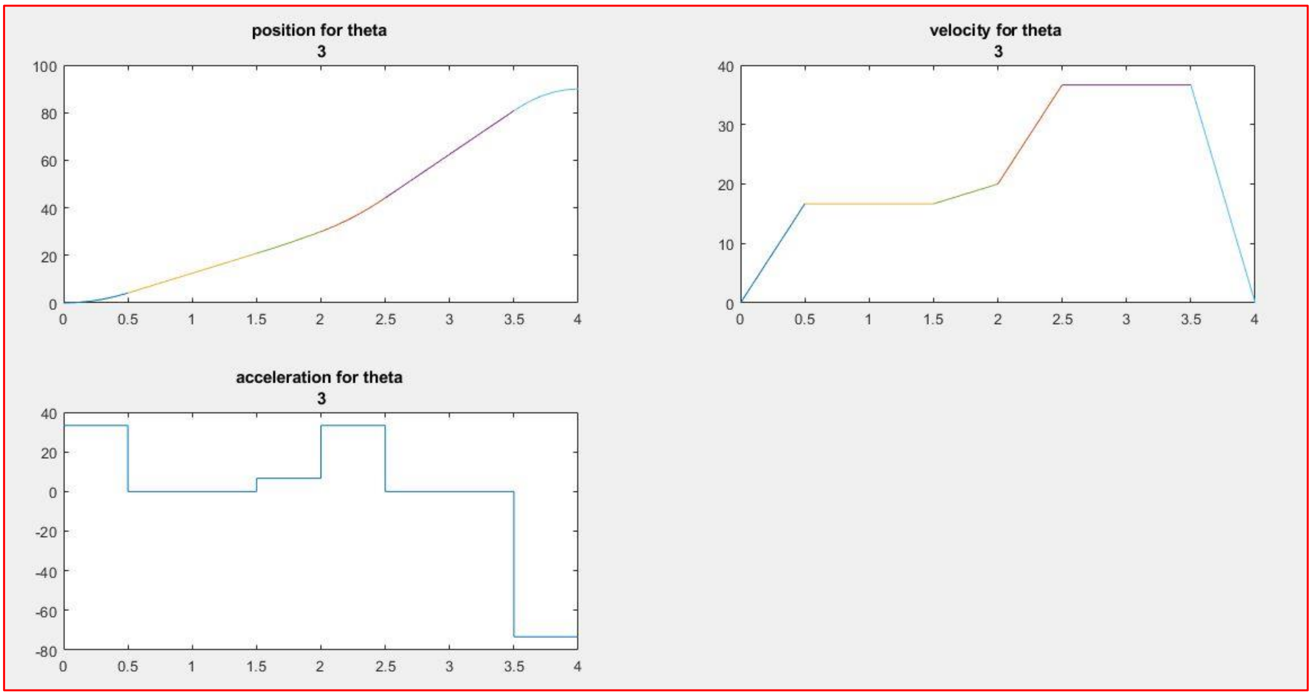

Fig 17. Position, Velocity and acceleration for theta3 (LSPB)

PART-6

Our objective is to implement a partitioned control law for the Microbot given in fig1 which will track the trajectory. The block diagram of the system is given below.

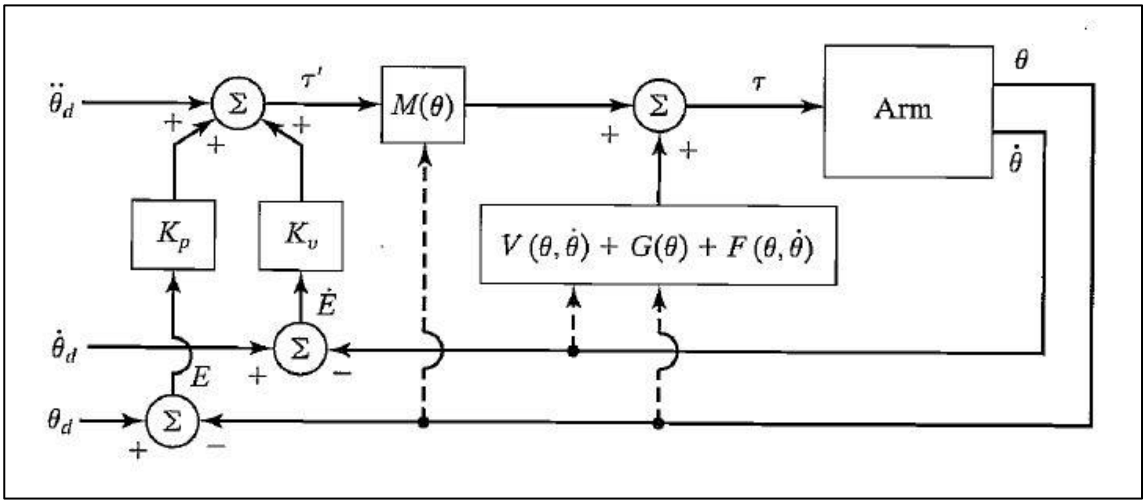

Fig 18. Manipulator control system

The figure shows that the output after each cycle is fed again. The feedback loop is used in this model. The inputs are position, velocity and acceleration. Two stages of error correction takes place. K_p and K_v are the gains and K_v is given by $2\sqrt{k_p}$ and these constitutes the partitioned control law. Summers are used at various places. Torque is yielded after correction. The arm acts as a block of forward dynamics and yield position and velocity for the given time 0 to 2s.

OUTPUT GRAPHS:

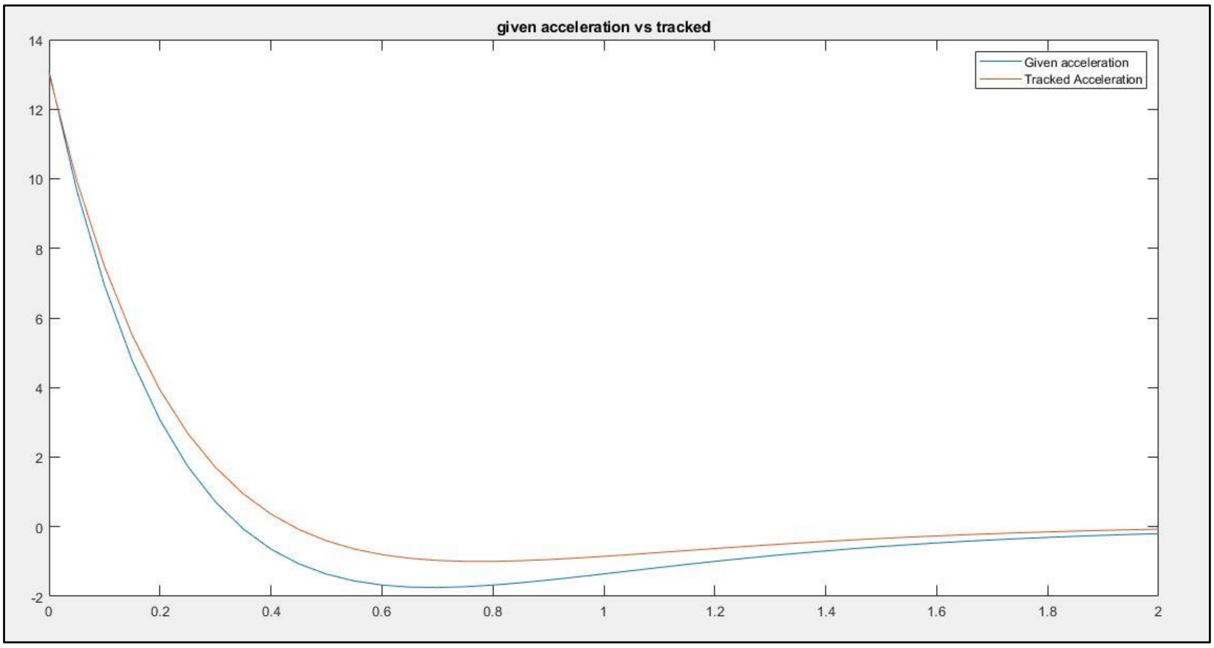

Fig 18. Given acceleration and tracked acceleration for all joints

Velocity:

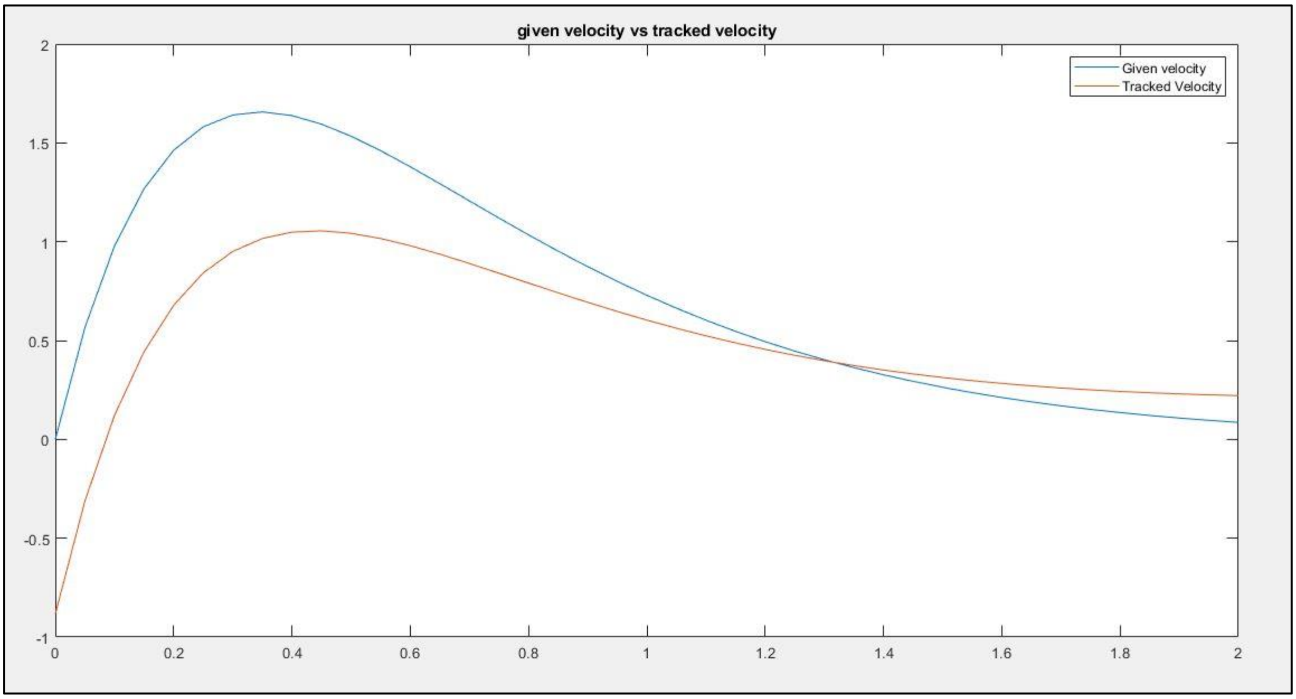

Fig 19. Given velocity and tracked velocity for all joints

Position:

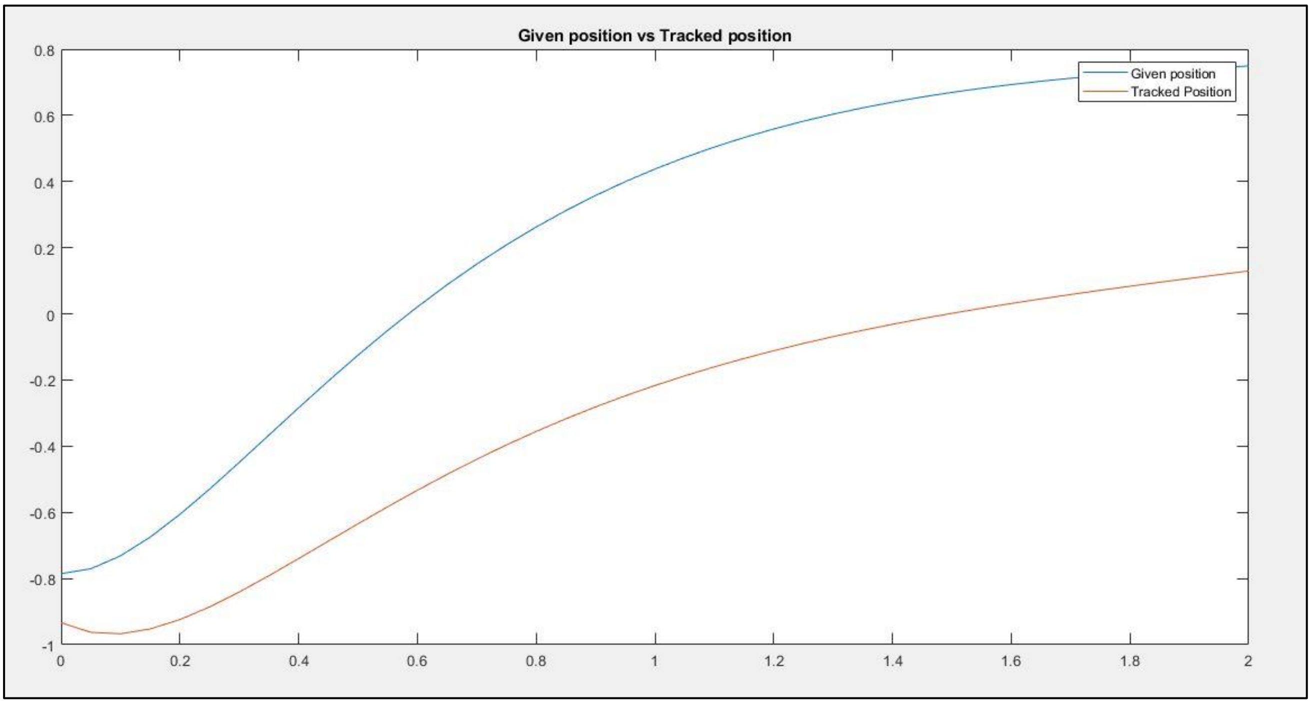

Fig 20. Given position and tracked position for all joints

Torque1:

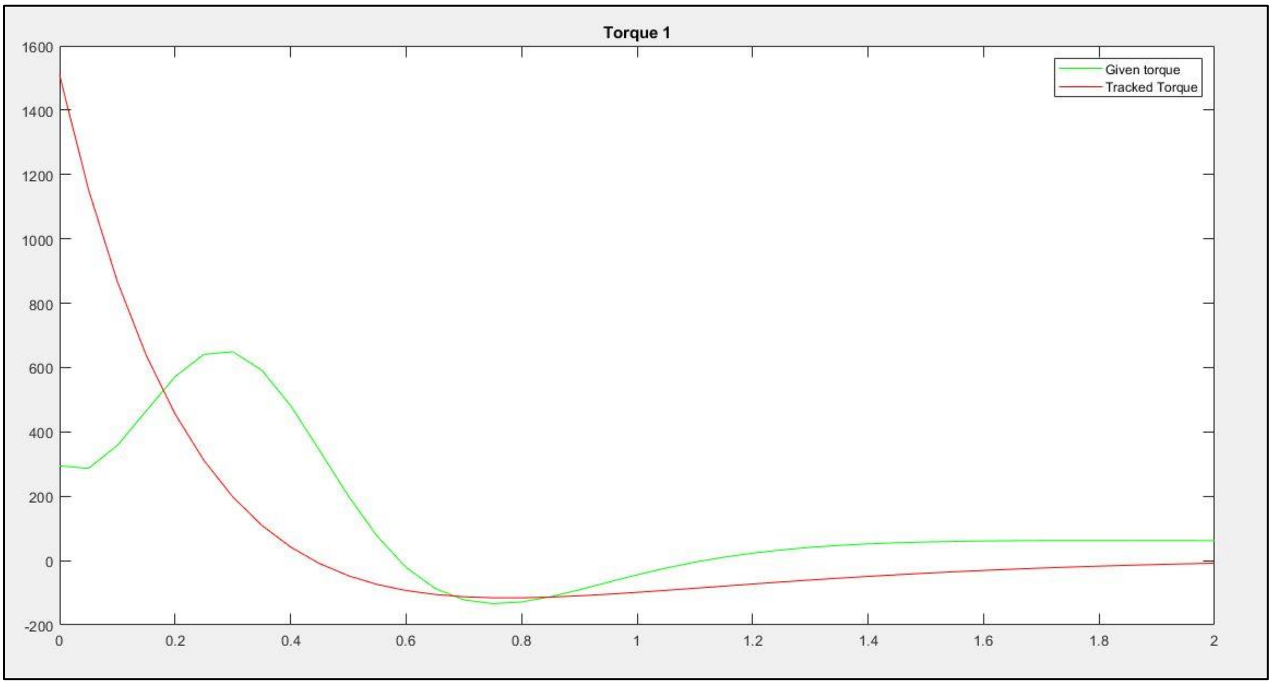

Fig 21. Given torque and tracked torque for joint 1

Torque2:

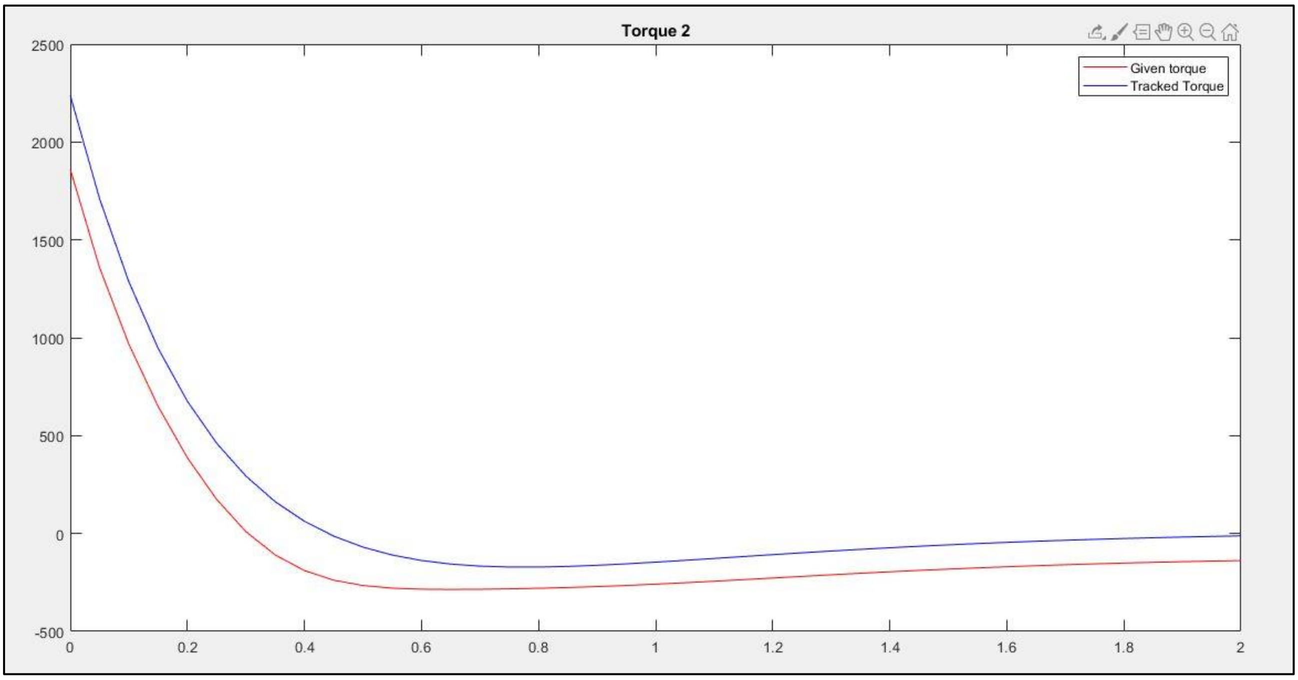

Fig 22. Given torque and tracked torque for joint 2

Torque3:

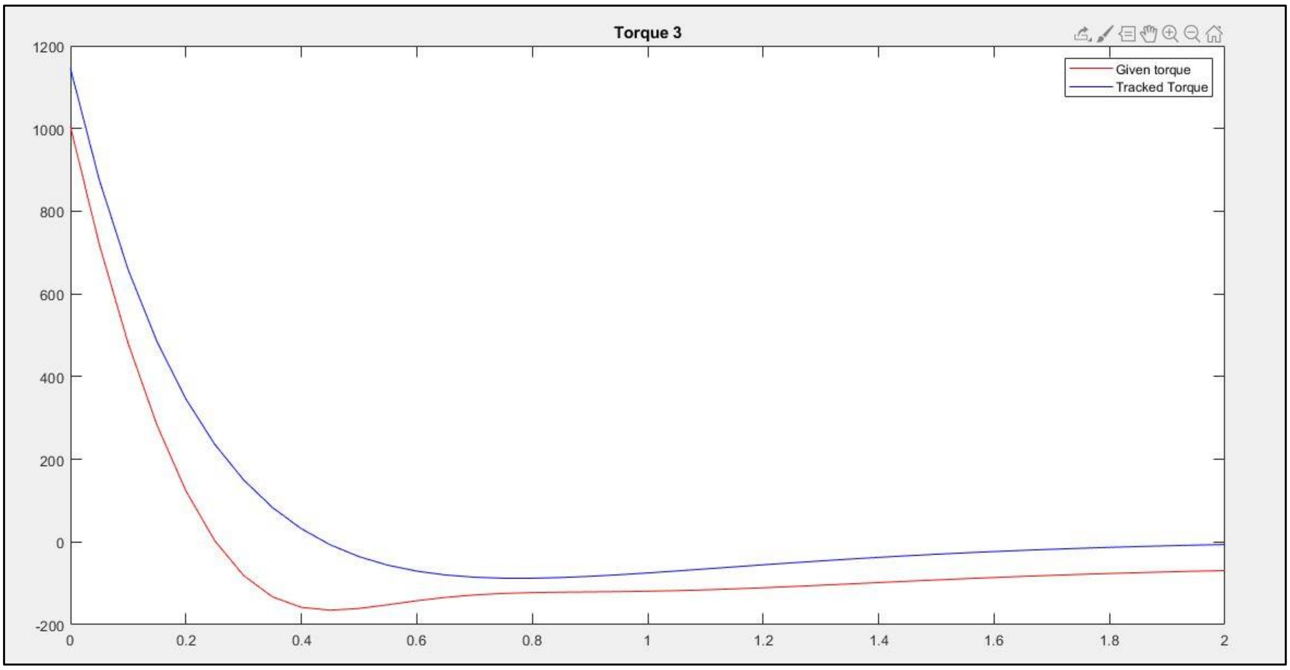

Fig 23. Given torque and tracked torque for joint 3

APPENDIX a:

MATLAB CODES:

PART-1 CODE:

```
clear all
close all
clc
prompt='Enter values of i:'
n=input(prompt)
% Getting D-H parameters from the user
prompt='Enter values of alphaminusone:'
alpha_iminusone=input(prompt)
prompt='Enter the values of aminusone:'
a_iminusone=input(prompt)
prompt='Enter the values of d:'
d=input(prompt)
e=3;f=4;h=2;
theta=[-86.41 22.886 -3.4150 0]
F=1;

for i=1:n
    A=[cosd(theta(i))  -sind(theta(i))  0  a_iminusone(i)];
    B=[(sind(theta(i)))*(cosd(alpha_iminusone(i)))
(cosd(theta(i)))*(cosd(alpha_iminusone(i))) (-sind(alpha_iminusone(i))) (-
sind(alpha_iminusone(i)))*(d(i))];
    C=[(sind(theta(i)))*(sind(alpha_iminusone(i)))
(cosd(theta(i)))*(sind(alpha_iminusone(i))) (cosd(alpha_iminusone(i)))
(cosd(alpha_iminusone(i)))*(d(i))];
    D=[0 0 0 1];
    T=[A;B;C;D];
    F=F*T  %Transformation matrix
end
```

PART-2 CODE:

```
clear all
close all
clc
Pos_x=3;Pos_y=-1;Pos_z=4.5;%given
positions
h=2;e=3;f=4;

% theta1 anticlockwise and elbow
up
th_1=atan2d(Pos_y,Pos_x); %θ_1
q_1=sqrt(Pos_x^2+Pos_y^2); %q1
q_2=Pos_z-h;    %q2
Psi_1=atan2d(q_2,q_1);  %ψ_1
cbeta_num=f^2-e^2-q_2^2-q_1^2;
cbeta_den=-2*e*sqrt(q_1^2+q_2^2);
Cbeta=cbeta_num/cbeta_den; %cos(β)
Sbeta=sqrt(1-Cbeta^2);     %sin(β)
Psi_2=atan2d(Sbeta,Cbeta);  %ψ_2
th_2=Psi_1+Psi_2;        %θ_2
```

```
cphi_num=q_1^2+q_2^2-f^2-e^2;
cphi_den=-2*e*f;
CPhi=cphi_num/cphi_den;   %cos(φ)
SPhi=sqrt(1-CPhi^2);      %sin(φ)
Phi=atan2d(SPhi,CPhi);      %φ
th_3= -(180-Phi);           %θ_3
ans1=[th_1,th_2,th_3];

%theta1 anticlockwise and elbow
down
th_1=atan2d(Pos_y,Pos_x);   %θ_1
q_1=sqrt(Pos_x^2+Pos_y^2);  %q1
q_2=Pos_z-h;          %q2
Psi_1=atan2d(q_2,q_1);       %ψ_1
cbeta_num=f^2-e^2-q_2^2-q_1^2;
cbeta_den=-2*e*sqrt(q_1^2+q_2^2);
Cbeta=cbeta_num/cbeta_den; %cos(β)
Sbeta=sqrt(1-Cbeta^2);     %sin(β)
Psi_2=atan2d(Sbeta,Cbeta);    %ψ_2
th_2=Psi_1-Psi_2;           %θ_2
```

```
cphi_num=q_1^2+q_2^2-f^2-e^2;
cphi_den=-2*e*f;
CPhi=cphi_num/cphi_den;    %cos(φ)
SPhi=sqrt(1-CPhi^2);       %sin(φ)
Phi=atan2d(SPhi,CPhi);      %φ
th_3=(180-Phi);              %θ_3
ans2=[th_1,th_2,th_3];

%theta 1 clockwise and elbow up
th_1=atan2d(-Pos_y,-Pos_x);   %θ_1
q_1=sqrt(Pos_x^2+Pos_y^2);  %q1
q_2=Pos_z-h;          %q2
Psi_1=atan2d(q_2,q_1);      %ψ_1
cbeta_num=f^2-e^2-q_2^2-q_1^2;
cbeta_den=-2*e*sqrt(q_1^2+q_2^2);
Cbeta=cbeta_num/cbeta_den; %cos(β)
Sbeta=sqrt(1-Cbeta^2);      %sin(β)
Psi_2=atan2d(Sbeta,Cbeta);   %ψ_2
psi=(Psi_1+Psi_2);
th_2=(180-psi);              %θ_2
cphi_num=q_1^2+q_2^2-f^2-e^2;
cphi_den=-2*e*f;
CPhi=cphi_num/cphi_den;     %cos(φ)
SPhi=sqrt(1-CPhi^2);        %sin(φ)
Phi=atan2d(SPhi,CPhi);         %φ
th_3=(180-Phi);              %θ_3
ans3=[th_1,th_2,th_3];

%theta 1 clockwise and elbow down
th_1=atan2d(-Pos_y,-Pos_x);  %θ_1
q_1=sqrt(Pos_x^2+Pos_y^2);   %q1
q_2=Pos_z-h;         %q2
Psi_1=atan2d(q_2,q_1);      %ψ_1
cbeta_num=f^2-e^2-q_2^2-q_1^2;
cbeta_den=-2*e*sqrt(q_1^2+q_2^2);
Cbeta=cbeta_num/cbeta_den; %cos(β)
Sbeta=sqrt(1-Cbeta^2);      %sin(β)
Psi_2=atan2d(Sbeta,Cbeta);  %ψ_2
psi=(Psi_2-Psi_1);
th_2=-(180-psi);             %θ_2
cphi_num=q_1^2+q_2^2-f^2-e^2;
cphi_den=-2*e*f;
CPhi=cphi_num/cphi_den;    %cos(φ)
SPhi=sqrt(1-CPhi^2);       %sin(φ)
Phi=atan2d(SPhi,CPhi);          %φ
th_3=-(180-Phi);             %θ_3
ans4=[th_1,th_2,th_3];

possiblesolutions=[ans1;ans2;ans3;ans4]
```

PART-3i CODE:

```
clear all
close all
clc
syms theta1 theta2 theta3 theta1dot theta2dot theta3dot L1 L2 ;
alpha_iminusone=[0 0 0];
a_iminusone=[0;L1;L2];
d=[0 0 0];
theta=[theta1 theta2 theta3];

T0to1=[cos(theta1) (-sin(theta1)) 0 0;sin(theta1) cos(theta1) 0 0;0 0 1 0;0 0 0 1]
R0to1=T0to1(1:3,1:3)
P0to1=T0to1(1:3,4)

T1to2=[cos(theta2) (-sin(theta2)) 0 L1;(sin(theta2)) (cos(theta2)) 0 0;0 0 1 0;0 0 0 1]
R1to2=T1to2(1:3,1:3)
P1to2=T1to2(1:3,4)

T2to3=[1 0 0 L2;0 1 0 0;0 0 1 0;0 0 0 1]
R2to3=T2to3(1:3,1:3)
P2to3=T2to3(1:3,4)

omega0wrt0=[0;0;0]
V0wrt0=[0;0;0]

omega1wrt1=(((R0to1))'*(omega0wrt0)+[0;0;theta1dot])
V1wrt1=((R0to1)*((V0wrt0)+(cross(omega0wrt0,P0to1))))
```

```
omega2wrt2=(((R1to2)')*(omega1wrt1)+[0;0;theta2dot])
V2wrt2=(((R1to2)')*((V1wrt1)+(cross(omega1wrt1,P1to2))))

omega3wrt3=(((R2to3)')*(omega2wrt2)+[0;0;theta3dot])   %Angular velocity
V3wrt3=(((R2to3)')*((V2wrt2)+(cross(omega2wrt2,P2to3))))  %Linear velocity

Cartesianvelocity=[V3wrt3;omega3wrt3]
```

PART-3ii CODE:

```
clc
clear all
close all

%D-H parameters
a0 = 0;a1 = 0;a2 = 3;a3 = 4;
d1 = 2;d2 = 0;d3 = 0;d4 = 0;
alph0 = 0;alph1 = pi/2;alph2 = 0;alph3 = 0;
th1 = deg2rad(-18.43);th2 = deg2rad(-29.18);th3 = deg2rad(-111.38);th4 =
deg2rad(0);
L1 = 0;L2 = 3;L3 = 4;

%Theta dots
td1 = deg2rad(35);
td2 = deg2rad(20);
td3 = deg2rad(45);
td4 = 0;

%Transformation matrix
T0_1 = [cos(th1) -sin(th1) 0 a0; sin(th1)*cos(alph0) cos(th1)*cos(alph0) -
sin(alph0) -sin(alph0)*d1; sin(th1)*sin(alph0) cos(th1)*sin(alph0)
cos(alph0) cos(alph0)*d1;0 0 0 1];
T1_2 = [cos(th2) -sin(th2) 0 a1; sin(th2)*cos(alph1) cos(th2)*cos(alph1) -
sin(alph1) -sin(alph1)*d2; sin(th2)*sin(alph1) cos(th2)*sin(alph1)
cos(alph1) cos(alph1)*d2;0 0 0 1];
T2_3 = [cos(th3) -sin(th3) 0 a2; sin(th3)*cos(alph2) cos(th3)*cos(alph2) -
sin(alph2) -sin(alph2)*d3; sin(th3)*sin(alph2) cos(th3)*sin(alph2)
cos(alph2) cos(alph2)*d3;0 0 0 1];
T3_4 = [cos(th4) -sin(th4) 0 a3; sin(th4)*cos(alph3) cos(th4)*cos(alph3) -
sin(alph3) -sin(alph3)*d4; sin(th4)*sin(alph3) cos(th4)*sin(alph3)
cos(alph3) cos(alph3)*d4;0 0 0 1];
Tb_w = T0_1*T1_2*T2_3*T3_4; %Final transformation matrix

%Position
P0_1 = [0;0;2];
P1_2 = [0;0;0];
P2_3 = [3;0;0];
P3_4 = [4;0;0];
%Transpose of rotation matrix
R1_0 = transpose(T0_1(1:3,1:3));
R2_1 = transpose(T1_2(1:3,1:3));
R3_2 = transpose(T2_3(1:3,1:3));
R4_3 = transpose(T3_4(1:3,1:3));

%Angular velocity
W0_0 = zeros(3,1);
```

```
W1_1 = R1_0*W0_0 + [0;0;td1];
W2_2 = R2_1*W1_1 + [0;0;td2];
W3_3 = R3_2*W2_2 + [0;0;td3];
W4_4 = R4_3*W3_3 + [0;0;td4];

%Linear velocity
V0_0 = zeros(3,1);
V1_1 = R1_0 * (V0_0 + cross(W0_0,P0_1));
V2_2 = R2_1 * (V1_1 + cross(W1_1,P1_2));
V3_3 = R3_2 * (V2_2 + cross(W2_2,P2_3));
V4_4 = R4_3 * (V3_3 + cross(W3_3,P3_4));

Cartesian_Velocity = [V4_4;W4_4] %Required Cartesian velocity
```

PART 4aCODE:

```
clc
clear all
close all
syms t

%D-H parametes
a0 = 0;a1 = 0;a2 = 3;a3 = 4;
d1 = 2;d2 = 0;d3 = 0;d4 = 0;
alpha0 = 0;alpha1 = pi/2;alpha2 = 0;alpha3 = 0;
L1 = 0;L2 = 3;L3 = 4;

%given position
th1 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
th2 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
th3 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
th4 = deg2rad(0);
%calculating velocity from given position
td1 = diff(th1);
td2 = diff(th2);
td3 = diff(th3);
td4 = 0;
%calculating acceleration from given position
tdd1 =  diff(td1);
tdd2 =  diff(td3);
tdd3 =  diff(td3);
tdd4 =  0;

M1 = 4;M2 = 2;M3 = 2;       %masses
g = 9.8;  %gravitational constant

T0_1 = [cos(th1) -sin(th1) 0 a0; sin(th1)*cos(alpha0) cos(th1)*cos(alpha0)
-sin(alpha0) -sin(alpha0)*d1; sin(th1)*sin(alpha0) cos(th1)*sin(alpha0)
cos(alpha0) cos(alpha0)*d1;0 0 0 1];
T1_2 = [cos(th2) -sin(th2) 0 a1; sin(th2)*cos(alpha1) cos(th2)*cos(alpha1)
-sin(alpha1) -sin(alpha1)*d2; sin(th2)*sin(alpha1) cos(th2)*sin(alpha1)
cos(alpha1) cos(alpha1)*d2;0 0 0 1];
T2_3 = [cos(th3) -sin(th3) 0 a2; sin(th3)*cos(alpha2) cos(th3)*cos(alpha2)
-sin(alpha2) -sin(alpha2)*d3; sin(th3)*sin(alpha2) cos(th3)*sin(alpha2)
cos(alpha2) cos(alpha2)*d3;0 0 0 1];
T3_4 = [cos(th4) -sin(th4) 0 a3; sin(th4)*cos(alpha3) cos(th4)*cos(alpha3)
-sin(alpha3) -sin(alpha3)*d4; sin(th4)*sin(alpha3) cos(th4)*sin(alpha3)
cos(alpha3) cos(alpha3)*d4;0 0 0 1];
```

```
Tb_w = T0_1*T1_2*T2_3*T3_4; %Homogenous transformation matrix

%for inward iterations
R1_0 = transpose(T0_1(1:3,1:3));
R2_1 = transpose(T1_2(1:3,1:3));
R3_2 = transpose(T2_3(1:3,1:3));
R4_3 = transpose(T3_4(1:3,1:3));
%for outward iterations
r0_1 = T0_1(1:3,1:3);
r1_2 = T1_2(1:3,1:3);
r2_3 = T2_3(1:3,1:3);
r3_4 = T3_4(1:3,1:3);

%Positions from the T matrix and centre of mass
p0_1 = zeros(3,1);
p1_2 = [L1;0;0];
p2_3 = [L2;0;0];
p3_4 = [L3;0;0];
pc1_1 = [L1;0;0];
pc2_2 = [L2;0;0];
pc3_3 = [L3;0;0];
%Assumptions for inertia
IC11 = zeros(3,3);
IC22 = zeros(3,3);
IC33 = zeros(3,3);
%Initial assumptions
VD0_0 = [0;g;0];
f4_4 = zeros(3,1);
n4_4 = zeros(3,1);
w0_0 = zeros(3,1);
wd0_0 = zeros(3,1);

%OUTWARD RECURSIONS

%link 1
w1_1 = R1_0*w0_0 + [0;0;td1]; %angular velocity of link1 w.r.t frame1
wd1_1 = R1_0*wd0_0 + cross(R1_0*w0_0,[0;0;td1]) + [0;0;tdd1];  %angular
acceleration of link1 w.r.t frame1
vd1_1 = R1_0*(cross(wd0_0,p0_1) + cross(w0_0 , cross(w0_0,p0_1)) + VD0_0);
%linear acceleration of link1 w.r.t frame1
vdc1_1 = cross(wd1_1,pc1_1) + cross(w1_1, cross(w1_1,pc1_1)) + vd1_1;
%linear acceleration of centre of mass of link1
F1_1 = M1*vdc1_1; %Force acting on link1
N1_1 = IC11*wd1_1 + cross(w1_1,IC11*w1_1);  %Moment acting on link1

 %link 2
w2_2 = R2_1*w1_1 + [0;0;td2];
wd2_2 = R2_1*wd1_1 + cross(R2_1*w1_1,[0;0;td2]) + [0;0;tdd2];
vd2_2 = R2_1*(cross(wd1_1,p1_2) + cross(w1_1 , cross(w1_1,p1_2)) + vd1_1);
vdc2_2 = cross(wd2_2,pc2_2) + cross(w2_2, cross(w2_2,pc2_2)) + vd2_2;
F2_2 = M2*vdc2_2;
N2_2 = IC22*wd2_2 + cross(w1_1,IC11*w1_1);
%link 3
w3_3 = R3_2*w2_2 + [0;0;td3];
wd3_3 = R3_2*wd2_2 + cross(R3_2*w2_2,[0;0;td3]) + [0;0;tdd3];
vd3_3 = R3_2*(cross(wd2_2,p2_3) + cross(w2_2 , cross(w2_2,p2_3)) + vd2_2);
vdc3_3 = cross(wd3_3,pc3_3) + cross(w3_3, cross(w3_3,pc3_3)) + vd3_3;
F3_3 = M3*vdc3_3;
N3_3 = IC33*wd3_3 + cross(w2_2,IC22*w2_2);
```

```
%INWARD RECURSIONS
%link 3
f3_3 = r3_4*f4_4 + F3_3;
n3_3 = N3_3 + r3_4*n4_4 + cross(pc3_3,F3_3) + cross(p3_4,r3_4*f4_4);

%link 2
f2_2 = r2_3*f3_3 + F2_2;
n2_2 = N2_2 + r2_3*n3_3 + cross(pc2_2,F2_2) + cross(p2_3,r2_3*f3_3);

%link 1
f1_1 = r1_2*f2_2 + F1_1;
n1_1 = N1_1 + r1_2*n2_2 + cross(pc1_1,F1_1) + cross(p1_2,r1_2*f2_2);

%Joint torques
time = 0:0.05:2;
Torque1 = transpose(n1_1)*[0;0;1]
Torque1_plot=vpa(subs(Torque1,t,time)); %Torque of joint1
Torque2 = transpose(n2_2)*[0;0;1]
Torque2_plot=vpa(subs(Torque2,t,time));  %Torque of joint2
Torque3 = transpose(n3_3)*[0;0;1]
Torque3_plot=vpa(subs(Torque3,t,time));  %Torque of joint3

%Plotting torques of the joints against time
figure(1)
subplot(2,2,1)
plot(time,Torque1_plot,'b')
title('Torque for joint1')
subplot(2,2,2)
plot(time,Torque2_plot,'g')
title('Torque for joint2')
subplot(2,2,3)
plot(time,Torque3_plot,'r')
title('Torque for joint3')
hold off

%Plotting position
th1_plot=vpa(subs(th1,t,time));
figure(2)
plot(time,th1_plot,’g’)
title('Position for all joints')

%Plotting velocity
td1_plot=vpa(subs(td1,t,time));
figure(3)
plot(time,td1_plot,’r’)
title('Velocity for all joints')

%Plotting acceleration
tdd1_plot=vpa(subs(tdd1,t,time));
figure(4)
plot(time,tdd1_plot)
title('acceleration for all joints')
```

PART 4b CODE:

```
clear all
close all
clc
syms a0 a1 a2 a3 alpha0 alpha1 alpha2 alpha3 d1 d1 d2 d4 theta1 theta2
theta3 theta4 tdot1 tdot2 tdot3 tdot4 tddot1 tddot2 tddot3 L1 L2 L3 M1 M2
M3 t

%D-H parameters
a0 = 0;a1 = 0;a2 = 3;a3 = 4;
d1 = 2;d2 = 0;d3 = 0;d4 = 0;
alpha0 = 0;alpha1 = pi/2;alpha2 = 0;alpha3 = 0;
L1 = 0;L2 = 3;L3 = 4;
```

%NOTE: Since the torque equations are very long (10 pages long) , I am not adding the equations in the code. Please look appendix b for these equations

```
tou1=sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)^2*(32*cos((pi*(6*exp(-(10*t)/3)+........

tou2=8*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*
+.......

tou3=8*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4+.....

%mass and gravity
M1 = 4;M2 = 2;M3 = 2;
g = 0;

t0_1 = [cos(theta1) -sin(theta1) 0 a0; sin(theta1)*cos(alpha0)
cos(theta1)*cos(alpha0) -sin(alpha0) -sin(alpha0)*d1;
sin(theta1)*sin(alpha0) cos(theta1)*sin(alpha0) cos(alpha0)
cos(alpha0)*d1;0 0 0 1];
t1_2 = [cos(theta2) -sin(theta2) 0 a1; sin(theta2)*cos(alpha1)
cos(theta2)*cos(alpha1) -sin(alpha1) -sin(alpha1)*d2;
sin(theta2)*sin(alpha1) cos(theta2)*sin(alpha1) cos(alpha1)
cos(alpha1)*d2;0 0 0 1];
t2_3 = [cos(theta3) -sin(theta3) 0 a2; sin(theta3)*cos(alpha2)
cos(theta3)*cos(alpha2) -sin(alpha2) -sin(alpha2)*d3;
sin(theta3)*sin(alpha2) cos(theta3)*sin(alpha2) cos(alpha2)
cos(alpha2)*d3;0 0 0 1];
t3_4 = [cos(theta4) -sin(theta4) 0 a3; sin(theta4)*cos(alpha3)
cos(theta4)*cos(alpha3) -sin(alpha3) -sin(alpha3)*d4;
sin(theta4)*sin(alpha3) cos(theta4)*sin(alpha3) cos(alpha3)
cos(alpha3)*d4;0 0 0 1];
Tb_w = t0_1*t1_2*t2_3*t3_4;   %Transformation matrix

%for outward iterations
r1_0 = transpose(t0_1(1:3,1:3));
r2_1 = transpose(t1_2(1:3,1:3));
r3_2 = transpose(t2_3(1:3,1:3));
r4_3 = transpose(t3_4(1:3,1:3));
%for inward iterations
r0_1 = t0_1(1:3,1:3);
r1_2 = t1_2(1:3,1:3);
r2_3 = t2_3(1:3,1:3);
```

```
r3_4 = t3_4(1:3,1:3);

%position and centre of mass position
p0_1 = [0;0;2];
p1_2 = [0;0;0];
p2_3 = [3;0;0];
p3_4 = [4;0;0];
pc1_1 = [0;0;0];
pc2_2 = [3;0;0];
pc3_3 = [4;0;0];
%Assumptions
IC11 = zeros(3,3);
IC22 = zeros(3,3);
IC33 = zeros(3,3);
vd0_0 = [0;g;0];
f4_4 = zeros(3,1);
n4_4 = zeros(3,1);
w0_0 = zeros(3,1);
wd0_0 = zeros(3,1);

%OUTWARD ITERATIONS

%link 1
w1_1 = r1_0*w0_0 + [0;0;tdot1];  %angular velocity
wd1_1 = r1_0*wd0_0 + cross(r1_0*w0_0,[0;0;tdot1]) + [0;0;tddot1];
%angular acceleration
vd1_1 = r1_0*(cross(wd0_0,p0_1) + cross(w0_0 , cross(w0_0,p0_1)) + vd0_0);
%linear acceleration
vdc1_1 = cross(wd1_1,pc1_1) + cross(w1_1, cross(w1_1,pc1_1)) + vd1_1;
%linear acceleration of centre of mass
F1_1 = M1*vdc1_1;  %force exerted
N1_1 = IC11*wd1_1 + cross(w1_1,IC11*w1_1);  %moment exerted

%link 2
w2_2 = r2_1*w1_1 + [0;0;tdot2];
wd2_2 = r2_1*wd1_1 + cross(r2_1*w1_1,[0;0;tdot2]) + [0;0;tddot2];
vd2_2 = r2_1*(cross(wd1_1,p1_2) + cross(w1_1 , cross(w1_1,p1_2)) + vd1_1);
vdc2_2 = cross(wd2_2,pc2_2) + cross(w2_2, cross(w2_2,pc2_2)) + vd2_2;
F2_2 = M2*vdc2_2;
N2_2 = IC22*wd2_2 + cross(w1_1,IC11*w1_1);

%link 3
w3_3 = r3_2*w2_2 + [0;0;tdot3];
wd3_3 = r3_2*wd2_2 + cross(r3_2*w2_2,[0;0;tdot3]) + [0;0;tddot3];
vd3_3 = r3_2*(cross(wd2_2,p2_3) + cross(w2_2 , cross(w2_2,p2_3)) + vd2_2);
vdc3_3 = cross(wd3_3,pc3_3) + cross(w3_3, cross(w3_3,pc3_3)) + vd3_3;
F3_3 = M3*vdc3_3;
N3_3 = IC33*wd3_3 + cross(w2_2,IC22*w2_2);

%INWARD ITERATIONS
%link 3
f3_3 = r3_4*f4_4 + F3_3;
n3_3 = N3_3 + r3_4*n4_4 + cross(pc3_3,F3_3) + cross(p3_4,r3_4*f4_4);

%link 2
f2_2 = r2_3*f3_3 + F2_2;
n2_2 = N2_2 + r2_3*n3_3 + cross(pc2_2,F2_2) + cross(p2_3,r2_3*f3_3);

%link 1
```

```
f1_1 = r1_2*f2_2 + F1_1;
n1_1 = N1_1 + r1_2*n2_2 + cross(pc1_1,F1_1) + cross(p1_2,r1_2*f2_2);

%Joint torques
Torque1 = transpose(n1_1)*[0;0;1];
Torque2 = transpose(n2_2)*[0;0;1];
Torque3 = transpose(n3_3)*[0;0;1]; %Obtaining torques in terms of symbol

%for calculating M
variables = [tddot1,tddot2,tddot3];
Torque = [Torque1,Torque2,Torque3];
[M1,xx] = equationsToMatrix(Torque,variables); %Obtaining M matrix

Th_1 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
Th_2 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
Th_3 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
TH=[Th_1;Th_2;Th_3];

Thd_1 = gradient(Th_1);
Thd_2 = gradient(Th_2);
Thd_3 = gradient(Th_3);

Thdd_1 = gradient(Thd_1);
Thdd_2 = gradient(Thd_2);
Thdd_3 = gradient(Thd_3);

%Forward dynamics
M2 = subs(M1,theta3,Th_3);
Minv = inv(M2);
VplusG = subs(Torque,[tddot1,tddot2,tddot3],[0,0,0]);
VplusG_sub =
subs(VplusG,[theta1,theta2,theta3,tdot1,tdot2,tdot3],[Th_1,Th_2,Th_3,Thd_1,
Thd_2,Thd_3]);
eqn = Torque - VplusG_sub;

acceleration = eqn*Minv;
time = 0:0.05:2;
%Acceleration
accelaration_numerical =
subs(acceleration,[theta1,theta2,theta3,tdot1,tdot2,tdot3,tddot1,tddot2,tdd
ot3],[Th_1,Th_2,Th_3,Thd_1,Thd_2,Thd_3,Thdd_1,Thdd_2,Thdd_3]);
accel_time=subs(accelaration_numerical,t,time);
accel_plot=vpa(subs(accelaration_numerical,t,time));
ac_1=accel_plot(1:41);ac_2=accel_plot(42:82);ac_3=accel_plot(83:123);
AC=[ac_1;ac_2;ac_3]
figure(1)
subplot(2,2,1)
plot(time,ac_1,'r')
title('Acceleration for joint 1')
subplot(2,2,2)
plot(time,ac_2,'g')
title('Acceleration for joint 2')
subplot(2,2,3)
plot(time,ac_3,'b')
title('Acceleration for joint 3')

%Integration
th=deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
```

```
th_d=diff(th);
th_dd=diff(th_d);
th_init=subs(th,t,0);   %Initial condition for theta
thd_init=subs(th_d,t,0);   %Initial condition for thetadot
thdd_init=subs(th_dd,t,0);  %Initial condition for thetadoubledot

syms t;
T(1)=th_init;
T_dot(1)=thd_init;
time=linspace(0,2,41);

%Euler's integration method
for i=1:length(time)-1
        T(i+1)=T(i)+T_dot(i)*time(i)+0.5*AC(1,i)*(time(i)^2);
        T_dot(i+1)=T_dot(i)+AC(1,i)*time(i);
end
z=linspace(0,2,41);

%Plotting Velocity
vel=int(accelaration_numerical,t);
vel_ans=vpa(subs(vel,t,z))
pos_z=0.8;
figure(2)
subplot(2,2,1)
plot(z,vel_ans(1:41),'r')
title('Velocity for joint 1')
hold on
subplot(2,2,2)
plot(z,vel_ans(42:82),'b')
title('Velocity for joint 2')
subplot(2,2,3)
plot(z,vel_ans(83:123),'k')
title('Velocity for joint 3')
hold off

%Plotting Position
pos=int(vel,t);
pos_ans=vpa(subs(pos,t,z))
pos_final=pos_z+pos_ans;
figure(3)
subplot(2,2,1)
plot(z,pos_final(1:41),'k')
title('Position for joint 1')
hold on
subplot(2,2,2)
plot(z,pos_final(42:82),'r')
title('Position for joint 2')
subplot(2,2,3)
plot(z,pos_final(83:123),'b')
title('Position for joint 3')
hold off

%Plotting torques
figure(4)
subplot(2,2,1)
tou1_plot=vpa(subs(tou1,t,z));
plot(z,tou1_plot,'k')
title('Torque for joint 1')
hold on
subplot(2,2,2)
```

```
tou2_plot=vpa(subs(tou2,t,z));
plot(z,tou2_plot,'g')
title('Torque for joint 2')
subplot(2,2,3)
tou3_plot=vpa(subs(tou3,t,z));
plot(z,tou3_plot,'b')
title('Torque for joint 3')
hold off
```

PART 5i CODE (CPS):

```
clc
clear all
close all
syms t
Theta=[0,-8,-90;0,45,90;0,30,90];%given theta values
Thetadot=[0,10,0;0,40,0;0,20,0];%given thetadot values
T=[0,2,4];%given time
Th_eq=[];
Thd_eq=[];
Thdd_eq=[];
%calculation
for i=1:2
    th_i=Theta(:,i);
    th_idot=Thetadot(:,i);
    th_f=Theta(:,i+1);
    th_fdot=Thetadot(:,i+1);
    T_0=T(:,i);T_f=T(:,i+1);
    A0=th_i; %calculating a0
    A1=th_idot;  %calculating a1
    A2=(3*(th_f-th_i)-(2*th_idot+th_fdot)*(T_f-T_0))/(T_f-T_0)^2;
%calculating a2
    A3=(2*(th_i-th_f)+(th_idot+th_fdot)*(T_f-T_0))/(T_f-T_0)^3;
%%calculating a3
    th_eq=A0+A1*(t-T_0)+A2*(t-T_0)^2+A3*(t-T_0)^3;   %calculating θ(t)
    Th_eq=[Th_eq,th_eq];
    thdot_eq=A1+2*A2*(t-T_0)+3*A3*(t-T_0)^2;    %calculating θ̇(t)
    Thd_eq=[Thd_eq,thdot_eq];
    thddot_eq=2*A2+6*A3*(t-T_0);          %calculating θ̈(t)
    Thdd_eq=[Thdd_eq,thddot_eq];
end


% plotting CPS
for j=1:3
    q1=0:0.2:2
    Jx=vpa(subs(Th_eq(j,1),{t},{q1}));%substituting time in θ(t) for 0 to
2s
    Jx1=vpa(subs(Thd_eq(j,1),{t},{q1})); %substituting time in θ̇(t) for 0
to 2s
    Jx2=vpa(subs(Thdd_eq(j,1),{t},{q1})); %substituting time in θ̈(t) for 0
to 2s
    q2=2:0.2:4
    Nx_1=vpa(subs(Th_eq(j,2),{t},{q2})); }));%substituting time in θ(t) for
2 to 4s
    Nx1_2=vpa(subs(Thd_eq(j,2),{t},{q2}));%substituting time in θ̇(t) for 2
to 4s
```

```
    Nx2_2=vpa(subs(Thdd_eq(j,2),{t},{q2}));%substituting time in $\ddot{\theta}(t)$ for 2 to 4s
    figure(j)
    a=plot(q1,Jx,'-g');
    hold on
    b=plot(q2,Nx_1,'-g');
    c=plot(q1,Jx1,'-r');
    d=plot(q2,Nx1_2,'-r');
   Jxx=[Jx2,Nx2_2];
    Nxx=[q1,q2];
    e=plot(Nxx,Jxx,'-b');
    title( {'Trajectory for theta ';num2str(j)});
    legend([a,c,e],"Position_","Velocity","Acceleration")
    hold off
end
```

PART 5ii CODE:

```
clc
clear all
close all
syms t;
Theta=[0,-8,-90;0,45,90;0,30,90];%given theta values
Thetadot=[0,10,0;0,40,0;0,20,0];%given thetadot values
T=[0,2,4]; %given time
Trajectory_1=[]; %null matrix
Trajectory_2=[];
Trajectory_3=[];
t_b=0.5;
Velocity_ans=[];
for i=1:2
    th_i=Theta(:,i);
    th_idot=Thetadot(:,i);
    th_f=Theta(:,i+1);
    th_fdot=Thetadot(:,i+1);
    T_0=T(:,i);
    T_f=T(:,i+1);
    Velocity=(th_f-th_i-0.5*t_b*(th_idot+th_fdot))/(T_f-(t_b+T_0)); %calculating velocity
    Velocity_ans=[Velocity_ans Velocity];
    %calculating $\theta(t)$ for $0 \le t \le t_b, t_b \le t \le t_f - t_b, t_f - t_b \le t \le t_f$
    traj_1=th_i+th_idot*(t-T_0)+0.5*(Velocity-th_idot)*(t-T_0)^2/t_b; %$0 \le t \le t_b$
    Trajectory_1=[Trajectory_1,traj_1];
    traj_2=Velocity*(t-T_0)+(th_i+0.5*th_idot*t_b-0.5*Velocity*t_b); %$t_b \le t \le t_f - t_b$
    Trajectory_2=[Trajectory_2,traj_2];
    traj_3=(th_f-th_fdot*T_f+((th_fdot-Velocity)*T_f^2)/(2*t_b))+(th_fdot-(th_fdot-Velocity)*T_f/t_b)*(t)+((th_fdot-Velocity)*((t)^2)/(2*t_b)); %$t_f - t_b \le t \le t_f$
    Trajectory_3=[Trajectory_3,traj_3];
end
%Calculating $\dot{\theta}$ and $\ddot{\theta}$
Trajdot1=diff(Trajectory_1,t);
Trajdot2=diff(Trajectory_2,t);
Trajdot3=diff(Trajectory_3,t);
Trajddot1=diff(Trajdot1,t);
Trajddot2=diff(Trajdot2,t);
```

```
Trajddot3=diff(Trajdot3,t);

%Plotting LSPB
b=4;
%For three segments
for j=1:3
T_0=0;T_f=2;
q1=T_0:0.1:t_b+T_0;
%substituting time
Trx_1=vpa(subs(Trajectory_1(j,1),{t},{q1}));
Trx_d1=vpa(subs(Trajdot1(j,1),{t},{q1}));
Trx_dd1=vpa(subs(Trajddot1(j,1),{t},{q1}));
T_0=T_0+2;T_f=T_f+2;
k11=T_0:0.1:t_b+T_0;
Trxx_1=vpa(subs(Trajectory_1(j,2),{t},{k11}));
Trxx_d1=vpa(subs(Trajdot1(j,2),{t},{k11}));
Trxx_dd1=vpa(subs(Trajddot1(j,2),{t},{k11}));
T_0=0;T_f=2;
q2=t_b+T_0:0.1:T_f-t_b;
Trx_2=vpa(subs(Trajectory_2(j,1),{t},{q2}));
Trx_d2=vpa(subs(Trajdot2(j,1),{t},{q2}));
Trx_dd2=vpa(subs(Trajddot2(j,1),{t},{q2}));
T_0=T_0+2;T_f=T_f+2;
k21=t_b+T_0:0.1:T_f-t_b;
Trxx_2=vpa(subs(Trajectory_2(j,2),{t},{k21}));
Trxx_d2=vpa(subs(Trajdot2(j,2),{t},{k21}));
Trxx_dd2=vpa(subs(Trajddot2(j,2),{t},{k21}));
T_0=0;T_f=2;
k3=T_f-t_b:0.1:T_f;
Trx_3=vpa(subs(Trajectory_3(j,1),{t},{k3}));
Trx_d3=vpa(subs(Trajdot3(j,1),{t},{k3}));
Trx_dd3=vpa(subs(Trajddot3(j,1),{t},{k3}));
T_0=T_0+2;T_f=T_f+2;
k31=T_f-t_b:0.1:T_f;
Trx_3x=vpa(subs(Trajectory_3(j,2),{t},{k31}));
Trxx_d3=vpa(subs(Trajdot3(j,2),{t},{k31}));
Trxx_dd3=vpa(subs(Trajddot3(j,2),{t},{k31}));

figure(b)
    subplot(2,2,1);
    plot(q1,Trx_1,k11,Trxx_1);
    hold on
    plot(q2,Trx_2,k21,Trxx_2);
    plot(k3,Trx_3,k31,Trx_3x);
    title( {'Position for theta ';num2str(j)});
    hold off
    subplot(2,2,2);
    plot(q1,Trx_d1,k11,Trxx_d1);
    hold on
    plot(q2,Trx_d2,k21,Trxx_d2);
    plot(k3,Trx_d3,k31,Trxx_d3);
    title( {'Velocity for theta ';num2str(j)});
    hold off
    subplot(2,2,3);
    ac_p=[Trx_dd1,Trx_dd2,Trx_dd3,Trxx_dd1,Trxx_dd2,Trxx_dd3];
    ac_pt=[q1,q2,k3,k11,k21,k31];
    plot(ac_pt,ac_p);
    title( {'Acceleration for theta ';num2str(j)});
    %hold off
    b=b+1;
end
```

PART 6 CODE:

```
clc
clear all
close all

syms  atdot atheta theta1 theta2 theta3 tdot1 tdot2 tdot3 tddot1 tddot2
tddot3 t
atd_1 = 0;atd_2 = 0;atd_3 = 0;at_1 = 0;at_2 = 0;at_3 = 0; z = 0:0.05:2;
%Obtained from inverse dynamics
%Truncated equations are added in appendix b
Torque1 = sin((pi*(6*exp(-(10*t)/3)+……………
Torque1_ans=vpa(subs(Torque1,t,z));
Torque2 = 8*sin((pi*(6*exp(-(10*conj(t))/3) - ……………………………….
Torque2_ans=vpa(subs(Torque2,t,z));
Torque3 = 8*sin((pi*(6*exp(-(10*conj(t))/3) - ………………………….….
Torque3_ans=vpa(subs(Torque3,t,z));

%Given position
Th_1 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
Th_2 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
Th_3 = deg2rad(45)*(1+6*exp(-t/0.3)-8*exp(-t/0.4));
TH = [Th_1,Th_2,Th_3];
%Given velocity
Thd_1 = gradient(Th_1);
Thd_2 = gradient(Th_2);
Thd_3 = gradient(Th_3);
THD = [Thd_1,Thd_2,Thd_3];
%Given acceleration
Thdd_1 = gradient(Thd_1);
Thdd_2 = gradient(Thd_2);
Thdd_3 = gradient(Thd_3);
THDD = [Thdd_1,Thdd_2,Thdd_3];

Atheta_output = [at_1,at_2,at_3];
Atdot_output = [atd_1,atd_2,atd_3];
%Gain
kp = 0.1;
kv = 2*(kp^(0.5));
%Taken from forward dynamics
V_block=[
(14903272800064533*cos(conj(theta3))*(2*sin(conj(theta3))+…….%truncated
Vblock_t=subs(V_block,[theta1,theta2,theta3,tdot1,tdot2,tdot3],[at_1,at_2,a
t_3,atd_1,atd_2,atd_3]);%substituting all values
M_block=7403584671771444860240298816936 3*cos(conj(theta3)))/822752278660603
0210774845912786752524913679328+…………………. %truncated
Mblock_t = subs(M_block,[theta1,theta2,theta3],[at_1,at_2,at_3]);
%Error correction
Edot1 = THD - Atdot_output;
Etheta1 = TH - Atheta_output;
Torque_dash = kv*Edot1 + kp*Etheta1 + THDD;     %Torque dash
torque = Torque_dash*Mblock_t + Vblock_t        %Torque

%Forward dynamics
Minv = inv(Mblock_t);       %M_inverse
eq_part=torque-Vblock_t;
accelaration=eq_part*Minv     %Acceleration
ac1=accelaration(1);
acc_1_ans=vpa(subs(ac1,t,z));   %Acceleration for joint 1
ac2=accelaration(2);
```

```
acc_2_ans=vpa(subs(ac2,t,z));    %Acceleration for joint 2
ac3=accelaration(2);
acc_3_ans=vpa(subs(ac3,t,z));    %Acceleration for joint 3
%Velocity and position
vel=int(accelaration,t);
vel_ans=vpa(subs(vel(1,1),t,z))
pos=int(vel,t);
pos_ans=vpa(subs(pos(1,1),t,z))
torqueans1=vpa(subs(torque(1,1),t,z)); %Torques
torqueans2=vpa(subs(torque(1,2),t,z));
torqueans3=vpa(subs(torque(1,3),t,z));
%Plotting graphs
%Plotting Acceleration
figure(1)
Thdd1_final=vpa(subs(Thdd_1,t,z));
plot(z,Thdd1_final)
hold on
plot(z,acc_1_ans)
title('given acceleration vs tracked')
legend('Given acceleration','Tracked Acceleration')
hold off
%Velocity plotting
figure(2)
Thd1_final=vpa(subs(Thd_1,t,z));
plot(z,Thd1_final)
hold on
plot(z,vel_ans)
title('given velocity vs tracked velocity')
legend('Given velocity','Tracked Velocity')
hold off
%Position plotting
figure(3)
Th1_final=vpa(subs(Th_1,t,z));
plot(z,Th1_final)
hold on
plot(z,pos_ans)
title('Given position vs Tracked position')
legend('Given position','Tracked Position')
hold off
%Plotting torques
figure(4);
plot(z,Torque1_ans,'g');
title('Torque 1')
hold on
plot(z,torqueans1,'r');
legend('Given torque', 'Tracked Torque')
hold off
figure(5);
plot(z,Torque2_ans,'r');
title('Torque 2')
hold on
plot(z,torqueans2,'b');
legend('Given torque', 'Tracked Torque')
hold off
figure(6);
plot(z,Torque3_ans,'r');
title('Torque 3')
hold on
plot(z,torqueans3,'b');
legend('Given torque', 'Tracked Torque')
hold off
```

APPENDIX b (Truncated equations used in part 4b,6):

```
Torque1 = sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)^2*(32*cos((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*((pi^2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/16 +
(pi*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/4) - (392*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4))/5 +
32*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*((pi^2*cos((pi*(6*exp(-(10*conj(t))/3)
- 8*exp(-(5*conj(t))/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/16 -
(pi*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/4) + (3*pi^2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/2 + 6*pi*cos((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-(10*t)/3))/3) +
(2433889152438200599906398154967 25*pi^2*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/162259276829213363391578010288128 + 4*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) -
8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2 +
(162259276829213368359335610309639*pi^2*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/202824096036516704239472512860 16) -
(16524327490667539036901295542150738761909780466889*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/13164036458569648337239753460458804039861886925068638906788872192 -
cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(24*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-
(5*t)/2) + 1))/4)*((pi^2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/16 + (pi*cos((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-(10*t)/3))/3))/4) -
(588*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4))/5 + 24*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*((pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/16 -
(pi*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/4) + cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(32*cos((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*((pi^2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/16 +
(pi*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/4) - (392*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4))/5 +
32*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*((pi^2*cos((pi*(6*exp(-(10*conj(t))/3)
- 8*exp(-(5*conj(t))/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/16 -
(pi*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/4) + (3*pi^2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/2 + 6*pi*cos((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-(10*t)/3))/3) +
(243388915243820059990639815496725*pi^2*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/162259276829213363391578010288128 + 4*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) -
8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2 +
(162259276829213368359335610309639*pi^2*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/202824096036516704239472512860 16) +
(9*pi^2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2)/4 + 9*pi*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-
(5*t)/2) + 1))/4)*(50*exp(-(5*t)/2) - (200*exp(-(10*t)/3))/3) +
(730166745731460179971919446490175*pi^2*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/324518553658426726783156020576256 + 3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) -
8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))^2 +
(486777830487640105078006830928917*pi^2*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/811296384146066816957890051440 64) -
(4967757600021511*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*((243388915243820059990639815496725*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/324518553658426726783156020576256 - (243420122401054039*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/40564819207303340847894502572 0320 +
(49*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/5 -
(3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16))/10141204801825835211973625643008
+ (4967757600021511*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*((1974605468785447492405389711764512203747948388697844981949524 1875*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/105312291668557186697918027683670432318895095400549111254310977536 -
(795070456463145529302756730622635*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
```

```
1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4))/81129638414606681695789005144064 + (3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/16))/10141204801825835211973625643008 - (14903272800064533*sin((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*(2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) +
1))/4)*((19746054687854474924053897117645122037479483886978449819495241875*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/10531229166855718669791802768367043231889509540054911254310977536 -
(795070456463145529302756730622635*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4))/81129638414606681695789005144064 + (3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) +
2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*((243388915243820059990639815496725*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/324518553658426726783156020576256 - (243420122401054039*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/405648192073033408478945025720320 +
(49*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/5 -
(3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) + 2*(pi*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3)) - pi*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)))*((pi*cos((pi*(6*exp(-(10*t)/3) -
8*exp(-(5*t)/2) + 1))/4)^2*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/4 - (pi*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)^2*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/4) +
(26328072917139298286609018205555527386232636944904737985456310321*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/13164036458569648337239753460458804039861886925068638906788872192))/81129638414606
681695789005144064 + (3627756489168007981517746420298787*cos((pi*(6*exp(-(10*conj(t))/3) -
8*exp(-(5*conj(t))/2) +
1))/4)^2)/16455045573212060421549691825573505049827358656335798633486090240 -
(14903272800064533*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*(2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*((243388915243820059990639815496725*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/324518553658426726783156020576256 - (243420122401054039*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/405648192073033408478945025720320 +
(49*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/5 -
(3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) +
(162259276829213368359335610309639*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/40564819207303340847894502572032 - 2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) +
1))/4)*((19746054687854474924053897117645122037479483886978449819495241875*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/105312291668557186697918027683670432318895095400549111254310977536 -
(795070456463145529302756730622635*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4))/81129638414606681695789005144064 + (3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) -
pi*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-
(5*t)/2) + 1))/4)*(pi*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)) - pi*sin((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3)))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3))))/81129638414606681695789005144064 -
(730260367203162117*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)^2)/202824096036516704239472512860160 + (44709818400193599*pi^2*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/649037107316853453566312041152512 + (4967757600021511*pi*cos((pi*(6*exp(-(10*t)/3)
- 8*exp(-(5*t)/2) + 1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*(pi*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)) - pi*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)))*(20*exp(-
(5*t)/2) - 20*exp(-(10*t)/3)))/20282409603651670423947251286016;
```

```
Torque2 = 8*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*((19746054687854474924053897117645122037479483886978449819495241875*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/105312291668557186697918027683670432318895095400549111254310977536 -
(795070456463145529302756730622635*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
```

```
1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4))/81129638414606681695789005144064 + (3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) -
(3326315174998874073721289211444399*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/162259276829213363391578010288128 - 8*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*((243388915243820059990639815496725*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/324518553658426726783156020576256 - (243420122401054039*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/405648192073033408478945025720320 +
(49*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/5 -
(3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) - 3*sin((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*(2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) +
1))/4)*((19746054687854474924053897117645122037479483886978449819495241875*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/105312291668557186697918027683670432318895095400549111254310977536 -
(795070456463145529302756730622635*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4))/81129638414606681695789005144064 + (3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) +
2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*((243388915243820059990639815496725*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/324518553658426726783156020576256 - (243420122401054039*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/405648192073033408478945025720320 +
(49*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/5 -
(3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) + 2*(pi*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3)) - pi*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)))*((pi*cos((pi*(6*exp(-(10*t)/3) -
8*exp(-(5*t)/2) + 1))/4)^2*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/4 - (pi*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)^2*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/4) +
(26328072917139298286609018205555527386232636944904737985456310321*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/13164036458569648337239753460458804039861886925068638906788872192) +
(730260367203162117*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)^2)/202824096036516704239472512860160 - 3*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*(2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*((243388915243820059990639815496725*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/324518553658426726783156020576256 - (243420122401054039*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/405648192073033408478945025720320 +
(49*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/5 -
(3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) +
(162259276829213368359335610309639*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/40564819207303340847894502572032 - 2*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) +
1))/4)*((19746054687854474924053897117645122037479483886978449819495241875*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/105312291668557186697918027683670432318895095400549111254310977536 -
(795070456463145529302756730622635*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4))/81129638414606681695789005144064 + (3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) -
pi*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-
(5*t)/2) + 1))/4)*(pi*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)) - pi*sin((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3)))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3))) - (294*sin((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/5 + (9*pi^2*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/8 + 4*pi*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(pi*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)) -
pi*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2*(20*exp(-(5*conj(t))/2)
- 20*exp(-(10*conj(t))/3)))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3));
```

```
Torque3 = 8*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*((197460546878544749240538971176451220374794838869784498194952418 75*pi^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-
(10*t)/3)))/105312291668557186697918027683670432318895095400549111254310977536 -
(795070456463145529302756730622635*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4)*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) +
1))/4))/81129638414606681695789005144064 + (3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*cos((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) -
(1622592768292133683593356103096 39*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/10141204801825835211973625643008 - 8*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)*((2433889152438200599906398154967 25*pi*(50*exp(-(5*t)/2) - (200*exp(-
(10*t)/3))/3))/324518553658426726783156020576256 - (243420122401054039*cos((pi*(6*exp(-
(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/405648192073033408478945025720320 +
(49*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2)/5 -
(3*pi^2*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)*sin((pi*(6*exp(-
(10*t)/3) - 8*exp(-(5*t)/2) + 1))/4)*(20*exp(-(5*conj(t))/2) - 20*exp(-
(10*conj(t))/3))*(20*exp(-(5*t)/2) - 20*exp(-(10*t)/3)))/16) + 4*pi*cos((pi*(6*exp(-(10*t)/3)
- 8*exp(-(5*t)/2) + 1))/4)*sin((pi*(6*exp(-(10*t)/3) - 8*exp(-(5*t)/2) +
1))/4)*(pi*cos((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-(5*conj(t))/2) + 1))/4)^2*(20*exp(-
(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)) - pi*sin((pi*(6*exp(-(10*conj(t))/3) - 8*exp(-
(5*conj(t))/2) + 1))/4)^2*(20*exp(-(5*conj(t))/2) - 20*exp(-(10*conj(t))/3)))*(20*exp(-
(5*t)/2) - 20*exp(-(10*t)/3));


V_block = [
(14903272800064533*cos(conj(theta3))*(2*sin(conj(theta3))*(((14903272800064533*conj(tdot1))/81
129638414606681695789005144064 +
3*conj(tdot2))*((4967757600021511*tdot1)/81129638414606681695789005144064 + tdot2) +
3*tdot1*cos(conj(theta2))*conj(tdot1)*cos(theta2)) + 2*(tdot1*cos(theta2)*sin(theta3) +
tdot1*cos(theta3)*sin(theta2))*(4*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1) -
4*sin(conj(theta2))*sin(conj(theta3))*conj(tdot1)) +
6*tdot1*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1)*sin(theta2)))/811296384146066816957890
05144064 -
(14903272800064533*sin(conj(theta3))*(2*cos(conj(theta3))*(((14903272800064533*conj(tdot1))/81
129638414606681695789005144064 +
3*conj(tdot2))*((4967757600021511*tdot1)/81129638414606681695789005144064 + tdot2) +
3*tdot1*cos(conj(theta2))*conj(tdot1)*cos(theta2)) + 2*(tdot1*cos(theta2)*cos(theta3) -
tdot1*sin(theta2)*sin(theta3))*(4*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1) -
4*sin(conj(theta2))*sin(conj(theta3))*conj(tdot1)) +
2*((4967757600021511*conj(tdot1))/20282409603651670423947251286016 + 4*conj(tdot2) +
4*conj(tdot3))*((4967757600021511*tdot1)/81129638414606681695789005144064 + tdot2 + tdot3) -
6*tdot1*cos(conj(theta2))*sin(conj(theta3))*conj(tdot1)*sin(theta2)))/811296384146066816957890
05144064 - cos(theta2)*(24*conj(tdot3)*(cos(conj(theta2))*sin(conj(theta3))*conj(tdot1) +
cos(conj(theta3))*sin(conj(theta2))*conj(tdot1)) + 6*(tdot1*cos(theta2)*sin(theta3) +
tdot1*cos(theta3)*sin(theta2))*((4967757600021511*conj(tdot1))/2028240960365167042394725128601
6 + 4*conj(tdot2) + 4*conj(tdot3)) +
cos(theta3)*(32*conj(tdot3)*(cos(conj(theta2))*sin(conj(theta3))*conj(tdot1) +
cos(conj(theta3))*sin(conj(theta2))*conj(tdot1)) + 8*(tdot1*cos(theta2)*sin(theta3) +
tdot1*cos(theta3)*sin(theta2))*((4967757600021511*conj(tdot1))/2028240960365167042394725128601
6 + 4*conj(tdot2) + 4*conj(tdot3)) +
8*tdot1*sin(theta2)*((14903272800064533*conj(tdot1))/81129638414606681695789005144064 +
3*conj(tdot2)) + 24*sin(conj(theta2))*conj(tdot1)*conj(tdot2) +
32*cos(conj(theta2))*conj(tdot1)*conj(tdot2)*sin(theta3) +
32*sin(conj(theta2))*conj(tdot1)*conj(tdot2)*cos(theta3)) +
12*tdot1*sin(theta2)*((14903272800064533*conj(tdot1))/81129638414606681695789005144064 +
3*conj(tdot2)) + 36*sin(conj(theta2))*conj(tdot1)*conj(tdot2) +
24*cos(conj(theta2))*conj(tdot1)*conj(tdot2)*sin(theta3) +
24*sin(conj(theta2))*conj(tdot1)*conj(tdot2)*cos(theta3)) +
(4967757600021511*sin(conj(theta3))*(((14903272800064533*conj(tdot1))/811296384146066816957890
05144064 + 3*conj(tdot2))*((4967757600021511*tdot1)/81129638414606681695789005144064 + tdot2)
+ 3*tdot1*cos(conj(theta2))*conj(tdot1)*cos(theta2)))/10141204801825835211973625643008 +
(4967757600021511*(tdot1*cos(theta2)*sin(theta3) +
tdot1*cos(theta3)*sin(theta2))*(4*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1) -
4*sin(conj(theta2))*sin(conj(theta3))*conj(tdot1)))/10141204801825835211973625643008 +
sin(theta2)*sin(theta3)*(32*conj(tdot3)*(cos(conj(theta2))*sin(conj(theta3))*conj(tdot1) +
cos(conj(theta3))*sin(conj(theta2))*conj(tdot1)) + 8*(tdot1*cos(theta2)*sin(theta3) +
tdot1*cos(theta3)*sin(theta2))*((4967757600021511*conj(tdot1))/2028240960365167042394725128601
6 + 4*conj(tdot2) + 4*conj(tdot3)) +
8*tdot1*sin(theta2)*((14903272800064533*conj(tdot1))/81129638414606681695789005144064 +
3*conj(tdot2)) + 24*sin(conj(theta2))*conj(tdot1)*conj(tdot2) +
32*cos(conj(theta2))*conj(tdot1)*conj(tdot2)*sin(theta3) +
32*sin(conj(theta2))*conj(tdot1)*conj(tdot2)*cos(theta3)) +
(44709818400193599*tdot1*cos(conj(theta2))*conj(tdot1)*sin(theta2))/405648192073033408478945025
72032 +
(14903272800064533*tdot1*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1)*sin(theta2))/10141204
801825835211973625643008,
```

```
3*cos(conj(theta3))*(2*sin(conj(theta3))*(((14903272800064533*conj(tdot1))/8112963841460668169
5789005144064 + 3*conj(tdot2))*((4967757600021511*tdot1)/81129638414606681695789005144064 +
tdot2) + 3*tdot1*cos(conj(theta2))*conj(tdot1)*cos(theta2)) + 2*(tdot1*cos(theta2)*sin(theta3)
+ tdot1*cos(theta3)*sin(theta2))*(4*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1) -
4*sin(conj(theta2))*sin(conj(theta3))*conj(tdot1)) +
6*tdot1*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1)*sin(theta2)) -
3*sin(conj(theta3))*(2*cos(conj(theta3))*(((14903272800064533*conj(tdot1))/8112963841460668169
5789005144064 + 3*conj(tdot2))*((4967757600021511*tdot1)/81129638414606681695789005144064 +
tdot2) + 3*tdot1*cos(conj(theta2))*conj(tdot1)*cos(theta2)) + 2*(tdot1*cos(theta2)*cos(theta3)
- tdot1*sin(theta2)*sin(theta3))*(4*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1) -
4*sin(conj(theta2))*sin(conj(theta3))*conj(tdot1)) +
2*((4967757600021511*conj(tdot1))/20282409603651670423947251286016 + 4*conj(tdot2) +
4*conj(tdot3))*((4967757600021511*tdot1)/81129638414606681695789005144064 + tdot2 + tdot3) -
6*tdot1*cos(conj(theta2))*sin(conj(theta3))*conj(tdot1)*sin(theta2)) +
8*sin(conj(theta3))*(((14903272800064533*conj(tdot1))/81129638414606681695789005144064 +
3*conj(tdot2))*((4967757600021511*tdot1)/81129638414606681695789005144064 + tdot2) +
3*tdot1*cos(conj(theta2))*conj(tdot1)*cos(theta2)) + 8*(tdot1*cos(theta2)*sin(theta3) +
tdot1*cos(theta3)*sin(theta2))*(4*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1) -
4*sin(conj(theta2))*sin(conj(theta3))*conj(tdot1)) +
18*tdot1*cos(conj(theta2))*conj(tdot1)*sin(theta2) +
24*tdot1*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1)*sin(theta2),
8*sin(conj(theta3))*(((14903272800064533*conj(tdot1))/81129638414606681695789005144064 +
3*conj(tdot2))*((4967757600021511*tdot1)/81129638414606681695789005144064 + tdot2) +
3*tdot1*cos(conj(theta2))*conj(tdot1)*cos(theta2)) + 8*(tdot1*cos(theta2)*sin(theta3) +
tdot1*cos(theta3)*sin(theta2))*(4*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1) -
4*sin(conj(theta2))*sin(conj(theta3))*conj(tdot1)) +
24*tdot1*cos(conj(theta2))*cos(conj(theta3))*conj(tdot1)*sin(theta2)];
```

M_block=

```
[(740358467177144486024029881693 63*cos(conj(theta3)))/822752278660603021077484591278675252491 3
679328167899316743 04512 + cos(theta2)*(36*cos(theta2) + cos(theta3)*(24*cos(theta2) +
32*cos(theta2)*cos(theta3) - 32*sin(theta2)*sin(theta3)) + 24*cos(theta2)*cos(theta3) -
24*sin(theta2)*sin(theta3)) +
(14903272800064533*cos(conj(theta3))*((14903272800064533*cos(conj(theta3)))/405648192073033408
47894502572032 +
4967757600021511/10141204801825835211973625643008))/81129638414606681695789005144064 +
(2221075401531433458072089645 08089*sin(conj(theta3))^2)/329100911464241208430993836511470100 99
654717312671597266972 18048 - sin(theta2)*sin(theta3)*(24*cos(theta2) +
32*cos(theta2)*cos(theta3) - 32*sin(theta2)*sin(theta3)) +
6169653893142870716866915680780 25/3291009114642412084309938365114701009965471731267159726697 21
8048, (14903272800064533*cos(conj(theta3)))/10141204801825835211973625643008 +
(14903272800064533*cos(conj(theta3))*(6*cos(conj(theta3)) +
8))/81129638414606681695789005144064 +
(44709818400193599*sin(conj(theta3))^2)/40564819207303340847894502572032 +
124193940000537775/40564819207303340847894502572032,
(14903272800064533*cos(conj(theta3)))/10141204801825835211973625643008 +
4967757600021511/2535301200456458802993406410752;(14903272800064533*cos(conj(theta3)))/1014120
4801825835211973625643008 +
3*cos(conj(theta3))*((14903272800064533*cos(conj(theta3)))/40564819207303340847894502572032 +
4967757600021511/10141204801825835211973625643008) +
(44709818400193599*sin(conj(theta3))^2)/40564819207303340847894502572032 +
124193940000537775/40564819207303340847894502572032,24*cos(conj(theta3)) +
3*cos(conj(theta3))*(6*cos(conj(theta3)) + 8) + 18*sin(conj(theta3))^2 +
50,24*cos(conj(theta3)) +
32;(14903272800064533*cos(conj(theta3)))/10141204801825835211973625643008 +
4967757600021511/2535301200456458802993406410752,24*cos(conj(theta3)) + 32,32];
```