# Dynamics and Control of a SCARA Manipulator

Manthan Pawar
MAE Department
NYU Tandon School of Engineering
Brooklyn, New York
mvp321@nyu.edu

Wenli Shen
MAE Department
NYU Tandon School of Engineering
Brooklyn, New York
ws1577@nyu.edu

**Abstract— Second order inversion kinematic algorithm for a SCARA robot is investigated. The performances of the system is tested in Simulink. The report presents an inverse dynamic control approach for the SCARA robot using an example when the end effector has 5kg od load. The setpoints for each joint are generated considering a second order inversion kinematic algorithm. For the same system, a robust controller is generated. Trajectory generation prototype in the robot operational space is shown with an example of 4 s with trapezoidal velocity profile for each segment passing through the waypoints p0 =[0 −0.80 0] at time t0 = 0.0, p1 = [0 −0.80 0.5] at time t1 = 0.6, p2 =[0.5 −0.6 0.5] at time t2 = 2.0, p3 =[0.8 0.0 0.5] at time t3 = 3.4, p4 = [0.8 0.0 0.0}at time t4 = 4.0. The trajectory is generated such that the robot will not stop at each waypoint so that the waypoints are via points. The anticipation time for each segment should be 0.2 s.**

**Index Terms—SCARA, Kinematic Algorithm, inverse dynamic, inversion kinematic algorithm, trajectory, robust controller, etc**

## 1. INTRODUCTION

A selective compliance assembly/articulated robot arm (SCARA) (Selective Compliance Assembly Robot Arm) is a standard manipulator used commonly in the industry. Comprised of three revolute and one prismatic joints, the robot has 4 degrees of freedom (DOF). SCARA is an open looptype manipulator that is used for assembly in production industries successfully. It is composed of three revolute joints allowing it to move and orient in the x-y plane, and one prismatic joint allows the movement of the end effector in the vertical (z-direction) direction.

The aim of this report is to demonstrate the implementation of Dynamics and control of a SCARA arm with particular parameters, given a trajectory. Simulink and MATLAB are employed to implement inverse dynamic control considering a second order inversion kinematic algorithm, where the Cartesian values are given. The first part of this report outlines the steps of second order inversion kinematic algorithm. The second part focuses on the dynamics of the manipulator. A control system for the SCARA robot is designed for the implementation of a dynamic control algorithm. To demonstrate the performance and efficiency of the system, a robust dynamic algorithm is implemented for a faster and more precise control.

## 2. INVERSION KINEMATIC ALGORITHM

### A) Algorithm for inversion kinematic:

For the simulation purpose the manipulator parameters are assumed to be:

$d0 = 1$ m, $a1 = a2 = 0.5$ m, $l1 = l2 = 0.25$ m $\theta 1min = -\pi/2 rad$,

$1max = \pi/2$ rad, $\theta 2min = -\pi/2$ rad, $\theta 2max = \pi/4$ rad $ml1 = ml2 = 25$ kg, $ml3 = 10$ kg, $Il1 = Il2 = 5$ kgm2, $Il4 = 1kgm2$

$kr1 = kr2 = 1$, $kr3 = 50$ rad/m, $kr4 = 20$,

$Im1 = Im2 = 0.0001$ kgm2, $Im3 = 0.01$ kgm2, $Im4 = 0.005kgm2$

$Fm1 = Fm2 = 0.0001$ Nms/rad, $Fm3 = 0.01$ Nms/rad, $Fm4 = 0.005$ Nms/rad

$d3min = 0.25$ m, $d3max = 1$ m, $\theta 4min = -2\pi$ rad, $\theta 4max = 2\pi$ rad

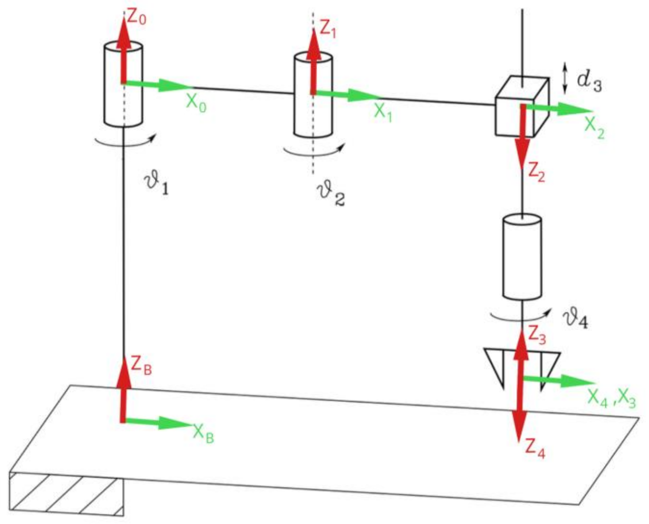

*Figure 1 Schematic figure of SCARA robot*

The Denavit-Hartenberg Parameter Table for the given manipulator is as follows:

| Link | $\Theta_i$ | $d_i$ | $a_i$ | $\alpha_i$ |
|---|---|---|---|---|
| 1 | $\Theta_1$ | 0 | 0.5 | 0 |
| 2 | $\Theta_2$ | 0 | 0.5 | 0 |
| 3 | 0 | $d_3$ | 0 | 0 |
| 4 | $\Theta_4$ | 0 | 0 | 0 |

*Table 1*

Now to calculate the direct kinematic equation, individual transformation matrices were calculated to be as follows:

$$^0T_1 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^1T_2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^2T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^3T_4 = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The final transformation matrix for the entire manipulator will be given as the multiplication of all the individual transformation matrices. The direct kinematics equation with minimum parametrization is given by

$$^0T_4 = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4$$

$$^0T_4 = \begin{bmatrix} c_1 & -s_1 & 0 & a_1c_1 \\ s_1 & c_1 & 0 & a_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^0T_4 = \begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The representation of the Differential Kinematics Equation.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{d}_3 \\ \dot{\theta}_4 \end{bmatrix}$$

**Let $a_1$=$a_2$=0.5**

$$\dot{x} = \frac{-s_{12}\dot{\theta}_2}{2} - \frac{(s_1+s_{12})\dot{\theta}_1}{2}$$

$$\dot{y} = \frac{C_{12}\dot{\theta}_2}{2} + \frac{(C_1 + C_{12})\dot{\theta}_1}{2}$$

$$\dot{z} = \dot{d}_3$$

$$\dot{\omega}_z = \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_4$$

In the geometric Jacobian there is no possibility of rotation around the x and y axes hence we can conclude that the 6x4 Matrix can be reduced to a 4x4 Matrix. This is the Analytical Jacobian. So, for this case the Geometrical and Analytical Jacobian is the same. Advantage of using this Analytical Jacobian is that we avoid redundancy. The Geometric Jacobian is represented as follows:

$$J = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

$$J_A = \begin{bmatrix} -a_1s_1 - a_2s_{12} & -a_2s_{12} & 0 & 0 \\ a_1c_1 + a_2c_{12} & a_2c_{12} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

$$\ddot{xe} = JA(q) * \ddot{q} + \dot{J}A(q,\dot{q}) * \dot{q}$$

The numerical integration to reconstruct the joint velocities and positions would unavoidably lead to a drift of the solution; therefore, similarly to the inverse kinematics algorithm with the Jacobian inverse, it is worth considering the error along with its derivative:

$$\ddot{e} = \ddot{xd} - \ddot{xe}$$

In this way, the joint acceleration vector can be simply defined as:

$$\ddot{q} = JA1(q) * (\ddot{xd} + KD * \dot{e} + Kp * e - \dot{J}A(q,\dot{q}) * \dot{q})$$

where $K$D and $K$P are positive definite (typically diagonal) matrices. which is asymptotically stable: the error tends to zero along the trajectory with a convergence speed depending on the choice of the matrices $K$D and $K$P.

$$\dot{J}(q) : \mathrm{J}(q,\dot{q}) =$$

$$\begin{bmatrix} -(a_1c_1 + a_2c_{12}) * \dot{\theta}_1 & -a_2s_{12} * \dot{\theta}_2 & 0 & 0 \\ -(a_1s_1 + a_2s_{12}) * \dot{\theta}_1 & a_2c_{12} * \dot{\theta}_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 1 & 0 & 1 \\ & & & \end{bmatrix}$$

B) Changes with respect to 1st order:

Kinematic control of a robot manipulator consists of solving the control problem in two stages, i.e., the desired task trajectory is transformed via the inverse kinematics into corresponding joint trajectories, which then constitute the reference inputs to some joint space control scheme. This approach differs from operational space control in the sense that manipulator kinematics is handled outside the control loop, thus allowing the problem of kinematic singularities and/or redundancy to be solved separately from the motion control problem. Moreover, most built-in controllers of industrial robots are based on joint servos and, thus, they can be easily embedded into a kinematic control strategy. The key point of kinematic control is the solution to the inverse kinematics problem. If the manipulator has simple geometry and is nonredundant with respect to the given task, closed-form solutions can be found for the joint variables, but special care has to be paid for the occurrence of singular configurations. On the other hand, for general manipulator structures, the inverse kinematics problem can be tackled by resorting to differential kinematics and, thus, inverting the end-effector motion into an equivalent joint motion. This is based on the manipulator Jacobian, which is the fundamental tool for analyzing kinematic singularities and/or redundancy.

In the first order kinematic algorithm, we used Jacobian and Jacobian transpose to generate the trajectory. However, in the second order inversion kinematic algorithm, along with Jacobian and Jacobian Transpose, we are using the derivative of Jacobian as a feedback control.

First-order algorithms allow the inversion of a motion trajectory, specified at the end-effector in terms of position and orientation, into the equivalent joint positions and velocities. For control purposes it may be necessary to invert a motion trajectory specified in terms of position, velocity and acceleration using the 2nd order algorithm. In this algorithm, we use $J(q)$ function to compute $\dot{x}$, $\dot{J}(q,\dot{q})$ function to compute the $\ddot{x}$, the rest structure is very similar to project 1. The joint space $q$ now is still calculated by accumulated $q$, the initial $q$ is $q0$, however, $q$ is no longer directly calculated by Jacobian inverse, $q$ is calculated by accumulated $q$ which is: $\ddot{q} = J(q) * (\ddot{x} + K * \dot{e} + K * e - \dot{J}(q,\dot{q}) * \dot{q})$, the initial $q$ is $q$ , the input now into the Jacobian inverse function consist of 4 parts, the $\ddot{x}$ ; $K * \dot{e}$; $K * e$; and $\dot{J}(q,\dot{q}) * \dot{q}$, which is illustrated in previous part.

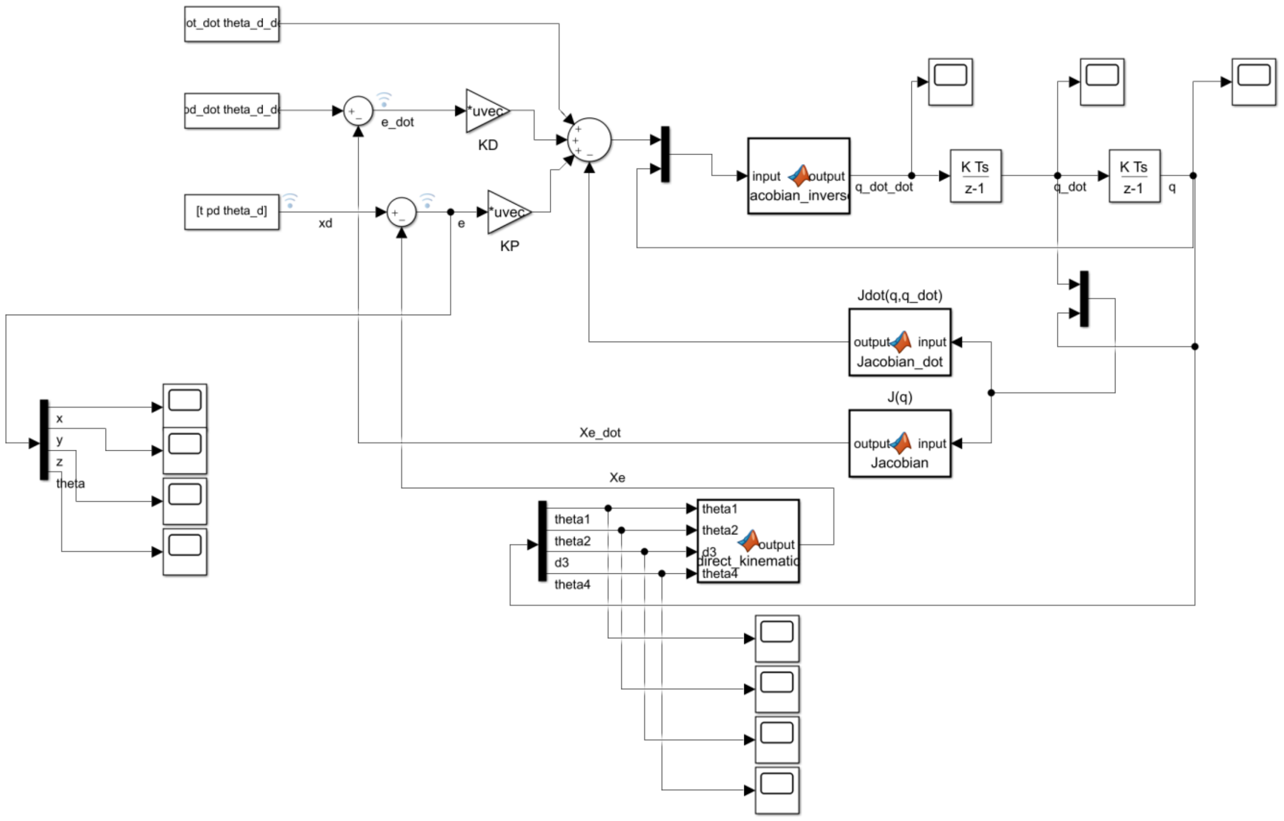

*Table 2:  Second order inversion kinematic algorithm*

C) Plots of joint variables and the errors in the operational space:

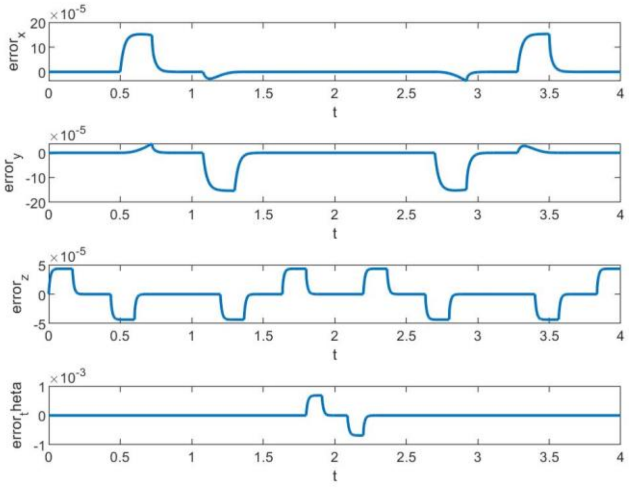

*Figure 2 Errors in the operational space*

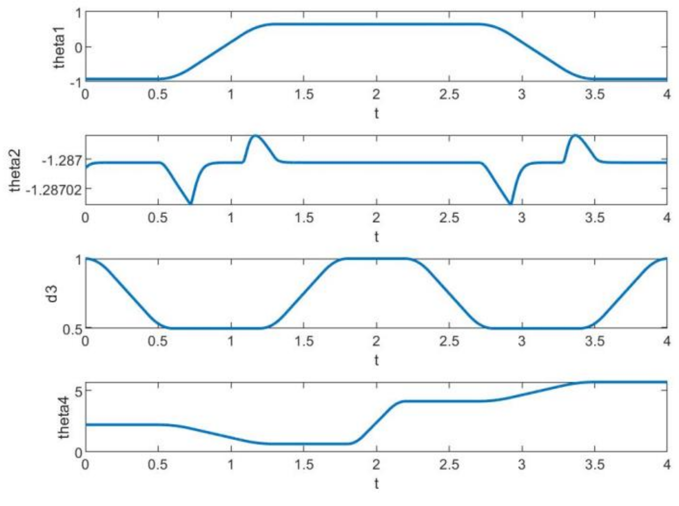

*Figure 3 Plots of joint variables*

## 3. INVERSE DYNAMICS CONTROL

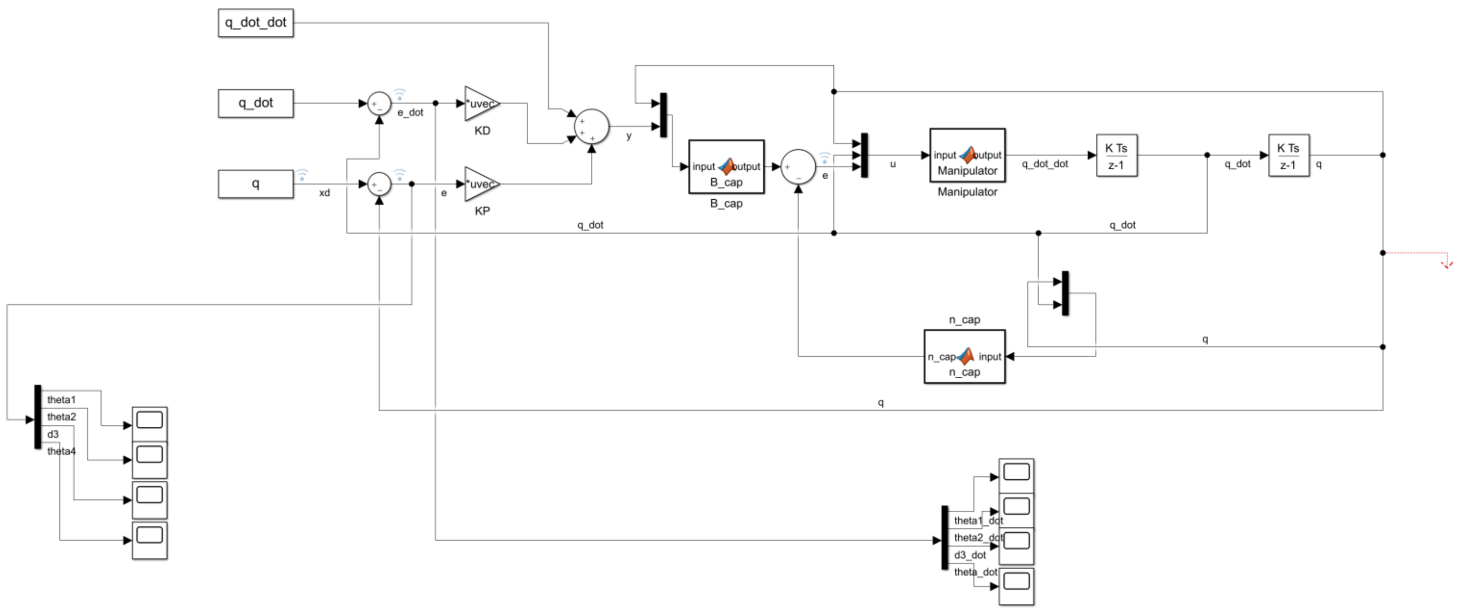

*Table 3: Inverse dynamic control approach*

A) Dynamic Model:

$$y = -K_P q - K_D \dot{q} + r$$

$$r = \ddot{q}_d + K_D \dot{q}_d + K_P q_d$$

$$\ddot{\tilde{q}} + K_D \dot{\tilde{q}} + K_P \tilde{q} = 0$$

The approach that follows is founded on the idea to find a control vector u, as a function of the system state, which is capable of realizing an input/output relationship of linear type; in other words, it is desired to perform not an approximate linearization but an exact linearization of system dynamics obtained by means of a nonlinear state feedback. The possibility of finding such a linearizing controller is guaranteed by the form of system dynamics.

B) B function:

The dynamic model of a manipulator provides a description of the relationship between the joint actuator torques and the motion of the structure. With Lagrange formulation, the equations of motion can be derived in a systematic way independently of the reference coordinate frame. the generalized. Once a set of variables $qi,i = 1,...,n$ , termed generalized coordinates, are chosen which effectively describe the link positions of this 4-DOF manipulator, the Lagrangian of the mechanical system can be defined as a function of the generalized coordinates:

$\boldsymbol{L} = \boldsymbol{\tau} - \boldsymbol{u}$

By summing the translational and rotational contributions and express the kinetic energy as a function of the generalized coordinates of the system, that are the joint variables, the geometric method for Jacobian computation can be applied to the intermediate link other than the end effector:

$$\tau = \frac{1}{2} * \sum_{i=1}^{4}\sum_{j=1}^{4} b_{ij}(q) * \dot{q}_i\dot{q}_j = \frac{1}{2}\dot{q}^T B(q)\dot{q}$$

Where,

$$B(q) = \sum_{i=1}^{n}\left(m_{\ell_i} J_P^{(\ell_i)T} J_P^{(\ell_i)} + J_O^{(\ell_i)T} R_i I_{\ell_i}^{i} R_i^T J_O^{(\ell_i)} + m_{m_i} J_P^{(m_i)T} J_P^{(m_i)} + J_O^{(m_i)T} R_{m_i} I_{m_i}^{m_i} R_{m_i}^T J_O^{(m_i)}\right)$$

In our case,

$$\mathbf{B(q)} = \begin{bmatrix} b11 & b11 & b13 & b14 \\ b21 & b22 & b23 & b24 \\ b31 & b32 & b33 & b34 \\ b41 & b42 & b43 & b44 \\ & & & \end{bmatrix}$$

$b_{11} = l_1^2 * m_{l1} + (0.25 + l_2^2 + l_2 * c_2) * m_{l2} + (0.5 + 0.5c_2) * (m_{l3} + m_{l4}) + I_{l1} + I_{l2} + I_{l4} + I_{m1} * k_{r1}^2 + I_{m2} + I_{m3} + I_{m4};$

$b_{12} = b_{21} = (0.5 * l_2 * c_2 + l_2^2) * m_{l2} + (0.25 + 0.25 * c_2) * (m_{l3} + m_{l4}) + I_{l2} + I_{l4} + I_{m2} * k_{r2} + I_{m3} + I_{m4};$

$b_{13} = b_{31} = -k_{r3} * I_{m3};$

$b_{14} = b_{41} = I_{l4} + k_{r4} * I_{m4};$

$b_{22} = l_2^2 * m_{l2} + 0.25 * (m_{l3} + m_{l4}) + I_{l2} + I_{l4} + I_{m2} * k_{r2}^2 + I_{m3} + I_{m4};$

$b_{23} = b_{32} = -k_{r3} * I_{m3};$

$b_{24} = b_{42} = I_{l4} + k_{r4} * I_{m4};$

$b_{33} = m_{l3} + I_{m3} * k_{r3}^2;$

$b_{34} = b_{43} = 0;$

$b_{44} = I_{l4} + k_{r4}^2 * I_{m4};$

C) N function:

Besides kinetic energy $\tau$, dynamic model also contains potential energy $u$ , the represent dynamic model is:

$$\boldsymbol{B}(\boldsymbol{q}) * \ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{\xi}$$

For $n(q,\dot{q})$, in this question,

$$\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) * \boldsymbol{q} + \boldsymbol{F}\boldsymbol{v} * \dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$$

It contains the configuration-dependent terms $g$, which represents the moment generated at Joint $i$ axis of the manipulator, in the current configuration, by the presence of gravity; viscous friction torques $F * \dot{q}$ , $F$ denotes the $(n \times n)$ diagonal matrix of viscous friction coefficients. Where $C$ is a suitable $(n \times n)$ matrix such that its elements $c$ satisfy this condition that the generic element of C is:

$$c_{ij} = \sum_{k=1}^{n} c_{ijk} * \dot{q}_k$$

$$c_{ijk} = \frac{1}{2} * \left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right)$$

We get,

$c_{11} = \frac{1}{2} * \frac{\partial b_{11}}{\partial q_2} * \dot{\theta}_2;$

$c_{11} = \frac{-l_2 * m_{l2} * s_2 - 0.5 * (m_{l3} + m_{l4}) * s_2}{2} * \dot{\theta}_2;$

$c_{12} = \frac{\partial b_{12}}{\partial q_2} * \dot{\theta}_2 + \frac{1}{2} * \frac{\partial b_{11}}{\partial q_2} * \dot{\theta}_1;$

$c_{12} = \frac{-l_2 * m_{l2} * s_2 - 0.5 * (m_{l3} + m_{l4}) * s_2}{2} * (\dot{\theta}_1 + \dot{\theta}_2);$

$c_{21} = -\frac{1}{2} * \frac{\partial b_{11}}{\partial q_2} * \dot{\theta}_1;$

$c_{21} = \frac{l_2 * m_{l2} * s_2 + 0.5 * (m_{l3} + m_{l4}) * s_2}{2} * \dot{\theta}_1;$

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \begin{bmatrix} c_{11} & c_{12} & 0 & 0 \\ c_{21} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{G} = \begin{bmatrix} 0 \\ 0 \\ -(m_{l3} + m_{l4}) * g \\ 0 \end{bmatrix}$$

Output of $n(q,\dot{q})$ is:

$$\boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) * \boldsymbol{q} + \boldsymbol{F}\boldsymbol{v} * \dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$$

D) Error plots in the joint space for position and velocity:

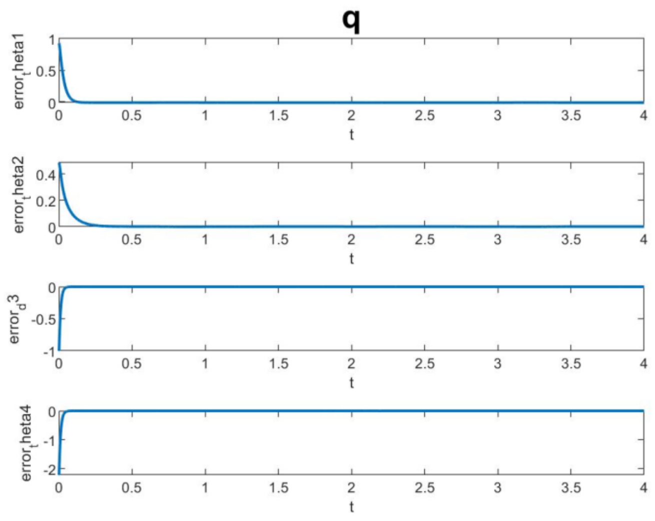

*Figure 4: Error plots for position*

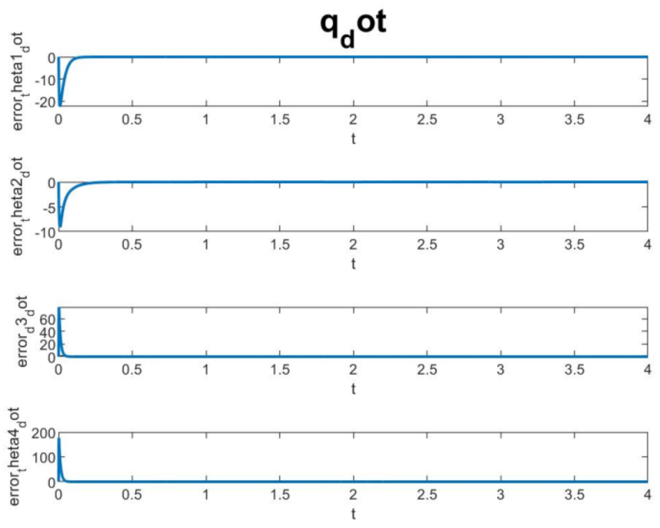

*Figure 5: Error plots for position*

## 4. TRAJECTORY GENERATION

### A) Trajectory Planning:

The acceleration we defined in this trajectory is:

$$\mathbf{a} = 5 * (sf - si)/(tf - ti)^2 - 4 * ti * tf$$

Based on the character of the method, the length travelled can be represented as:

$$q(t) = \begin{cases} q_i + \frac{1}{2} * \ddot{q}_c * t^2 & 0 < t < tc \\ q_i + \ddot{q}_c * t_c\left(t - \frac{t_c}{2}\right) & t_c < t < t_f - t_c \\ q_f - \frac{1}{2} * \ddot{q}_c * (t_f - t_c)^2 & t_f - t_c < t < t_f \end{cases}$$

Where,

$$t_c = \frac{(t_f + t_i)}{2} - \frac{1}{2} * \sqrt{\frac{t_f^2 * q_c - 4 * (\ddot{q_f - q_\iota})}{\ddot{q_c}}}$$

To compute path on operational space,

$$s_j(t) = \begin{cases} 0 \\ s_j'(t + \Delta t_j) \\ \|p_j - p_{j-1}\| \end{cases} \left| \begin{array}{c} 0 \le t \le t_{j-1} - \Delta t_j \\ t_{j-1} - \Delta t_j < t < \ t_j - \Delta t \\ t_j - \Delta t \le t \le t_f - \Delta t_N \end{array} \right.$$

For position of each point,

$$p_e = p_0 + \sum_{j=1}^{N} \frac{s_j}{\|p_j - p_{j-1}\|} * (p_j - p_{j-1})$$

for $j = 1,\ldots,N$

### B) 3D generated trajectory:

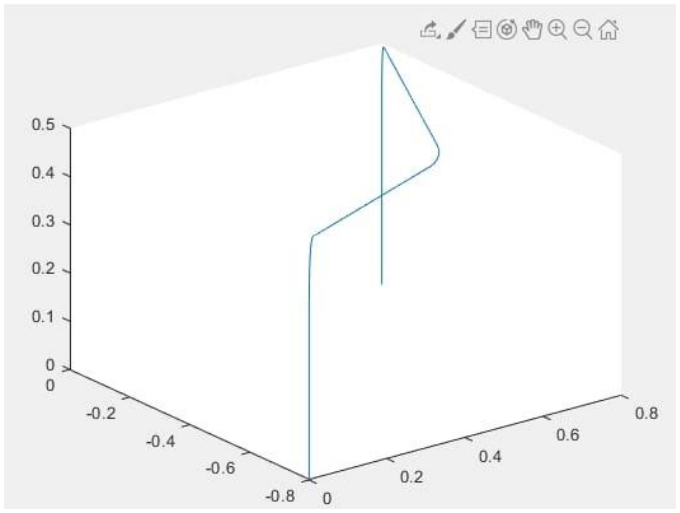

*Figure 6: 3D generated trajectory*

### C) Position and velocities along axis:

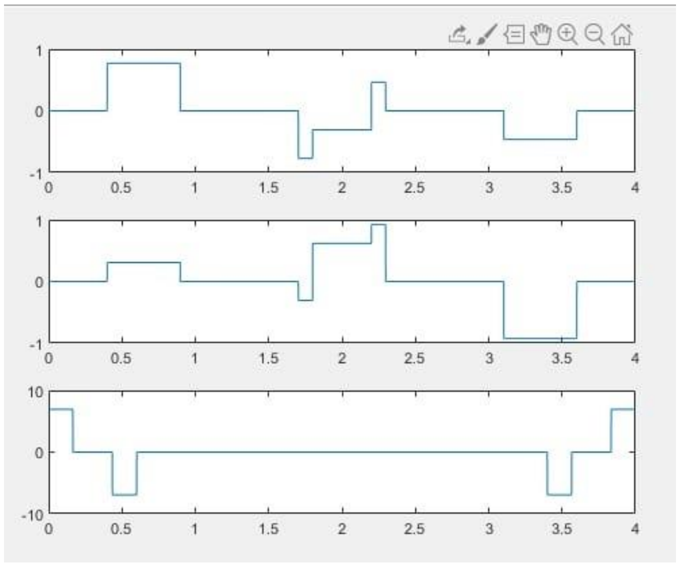

*Figure 7: Position along axis*

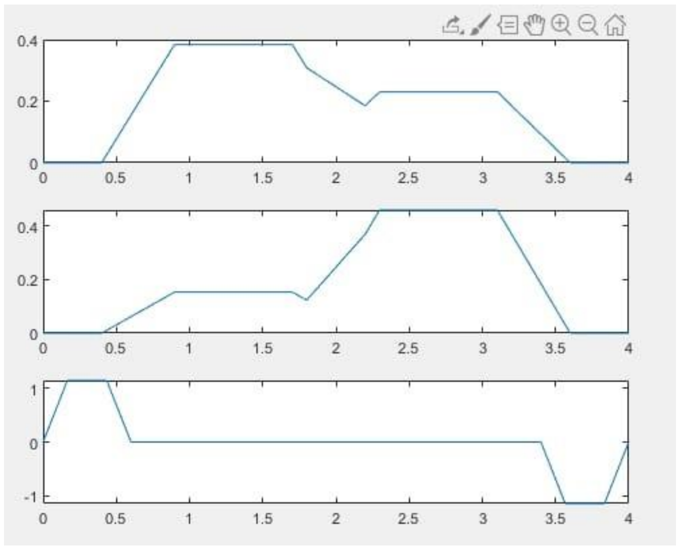

*Figure 8: Velocities along axis*

## 5. ROBUST CONTROLLER

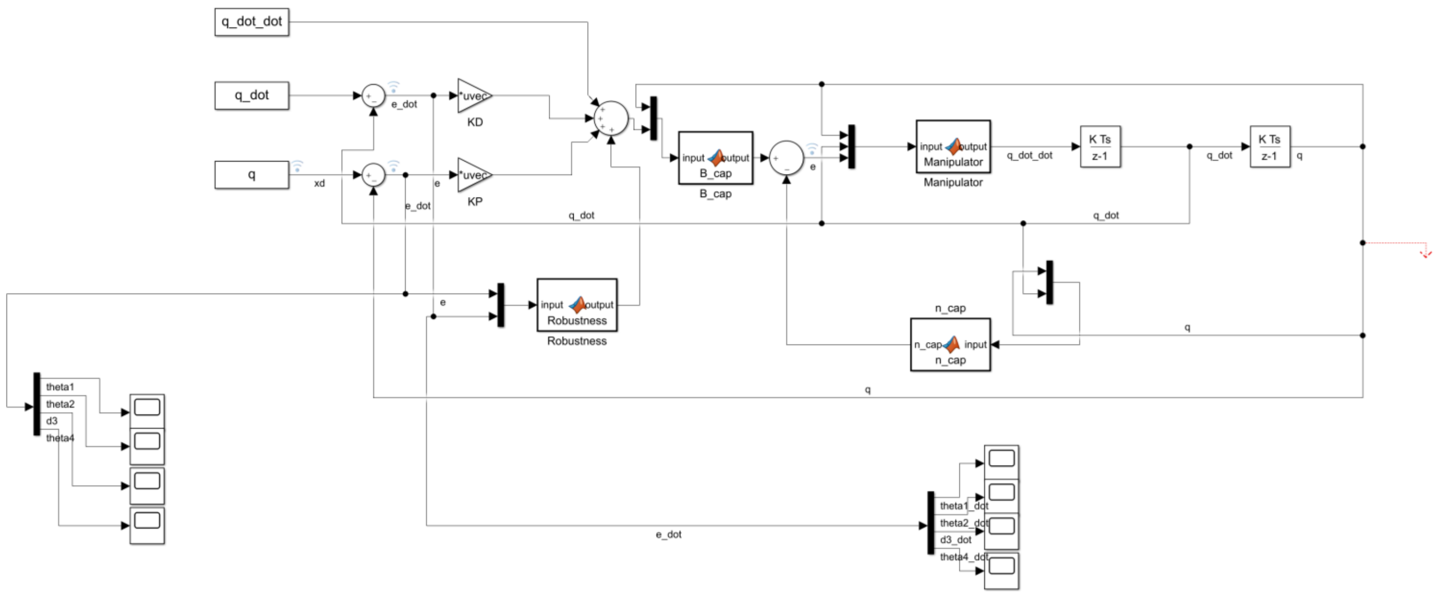

*Table 4: Robust Controller*

A) $\boldsymbol{B\ cap}(\boldsymbol{q})$ function:

$$b_{11} = l_1^2 * m_{l1} + (0.25 + l_2^2 + l_2 * c_2) * m_{l2} + (0.5 + 0.5c_2) * (m_{l3} + m_{l4}) + I_{l1} + I_{l2} + I_{l4} + I_{m1} * k_{r1}^2 + I_{m2} + I_{m3} + I_{m4};$$

$$b_{12} = b_{21} = (0.5 * l_2 * c_2 + l_2^2) * m_{l2} + (0.25 + 0.25 * c_2) * (m_{l3} + m_{l4}) + I_{l2} + I_{l4} + I_{m2} * k_{r2} + I_{m3} + I_{m4};$$

$$b_{13} = b_{31} = -k_{r3} * I_{m3};$$

$$b_{14} = b_{41} = I_{l4} + k_{r4} * I_{m4};$$

$$b_{22} = l_2^2 * m_{l2} + 0.25 * (m_{l3} + m_{l4}) + I_{l2} + I_{l4} + I_{m2} * k_{r2}^2 + I_{m3} + I_{m4};$$

$$b_{23} = b_{32} = -k_{r3} * I_{m3};$$

$$b_{24} = b_{42} = I_{l4} + k_{r4} * I_{m4};$$

$$b_{33} = m_{l3} + I_{m3} * k_{r3}^2;$$

$$b_{34} = b_{43} = 0;$$

$$b_{44} = I_{l4} + k_{r4}^2 * I_{m4};$$

$$\boldsymbol{B\ cap}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{b11} & \boldsymbol{b11} & \boldsymbol{b13} & \boldsymbol{b14} \\ \boldsymbol{b21} & \boldsymbol{b22} & \boldsymbol{b23} & \boldsymbol{b24} \\ \boldsymbol{b31} & \boldsymbol{b32} & \boldsymbol{b33} & \boldsymbol{b34} \\ \boldsymbol{b41} & \boldsymbol{b42} & \boldsymbol{b43} & \boldsymbol{b44} \\ & & & \end{bmatrix}$$

$$\boldsymbol{y} = \ddot{\boldsymbol{q}}\boldsymbol{d} + \boldsymbol{KD} * \boldsymbol{q} + \boldsymbol{KP} * \boldsymbol{q} + \dot{\boldsymbol{\omega}}$$

Where $\omega$ is the output of function, the output is:

$$\boldsymbol{B\ cap}(\boldsymbol{q}) * \text{y} = \begin{bmatrix} \boldsymbol{b11} & \boldsymbol{b11} & \boldsymbol{b13} & \boldsymbol{b14} \\ \boldsymbol{b21} & \boldsymbol{b22} & \boldsymbol{b23} & \boldsymbol{b24} \\ \boldsymbol{b31} & \boldsymbol{b32} & \boldsymbol{b33} & \boldsymbol{b34} \\ \boldsymbol{b41} & \boldsymbol{b42} & \boldsymbol{b43} & \boldsymbol{b44} \\ & & & \end{bmatrix} * \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

B) **N cap($\boldsymbol{q},\dot{\boldsymbol{q}}$)** function:

$$\boldsymbol{n}\ (\boldsymbol{q},\dot{\boldsymbol{q}}) \sim \boldsymbol{n}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) * \dot{\boldsymbol{q}} + \boldsymbol{Fv} * \dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$$

$$c_{11} = \frac{1}{2} * \frac{\partial b_{11}}{\partial q_2} * \dot{\theta}_2;$$

$$c_{11} = \frac{-l_2 * m_{l2} * s_2 - 0.5 * (m_{l3} + m_{l4}) * s_2}{2} * \dot{\theta}_2;$$

$$c_{12} = \frac{\partial b_{12}}{\partial q_2} * \dot{\theta}_2 + \frac{1}{2} * \frac{\partial b_{11}}{\partial q_2} * \dot{\theta}_1;$$

$$c_{12} = \frac{-l_2 * m_{l2} * s_2 - 0.5 * (m_{l3} + m_{l4}) * s_2}{2} * (\dot{\theta}_1 + \dot{\theta}_2);$$

$$c_{21} = -\frac{1}{2} * \frac{\partial b_{11}}{\partial q_2} * \dot{\theta}_1;$$

$$c_{21} = \frac{l_2 * m_{l2} * s_2 + 0.5 * (m_{l3} + m_{l4}) * s_2}{2} * \dot{\theta}_1;$$

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \begin{bmatrix} c_{11} & c_{12} & 0 & 0 \\ c_{21} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{G} = \begin{bmatrix} 0 \\ 0 \\ -(m_{l3} + m_{l4}) * g \\ 0 \end{bmatrix}$$

C) Robustness function:

Input $\boldsymbol{\xi} = \begin{bmatrix} \tilde{\boldsymbol{q}} \\ \dot{\tilde{\boldsymbol{q}}} \end{bmatrix}$

$$\boldsymbol{\omega} = \begin{cases} \boldsymbol{\rho} * \boldsymbol{z} / \|\boldsymbol{z}\| \\ \boldsymbol{per}\ \|\boldsymbol{z}\| \geq \boldsymbol{\epsilon} \\ \boldsymbol{\rho}\ \ * \boldsymbol{z}\ \boldsymbol{\epsilon} \\ \boldsymbol{per}\ \|\boldsymbol{z}\| < \boldsymbol{\epsilon} \end{cases}$$

$\boldsymbol{where}\ \boldsymbol{\rho} > \boldsymbol{0}$
$\boldsymbol{z} = \boldsymbol{DT} * \boldsymbol{Q} * \boldsymbol{\xi}$
Then set $\boldsymbol{\epsilon} = \boldsymbol{0.06}, \boldsymbol{\rho} = \boldsymbol{70}$.

D) Result of Robust Control

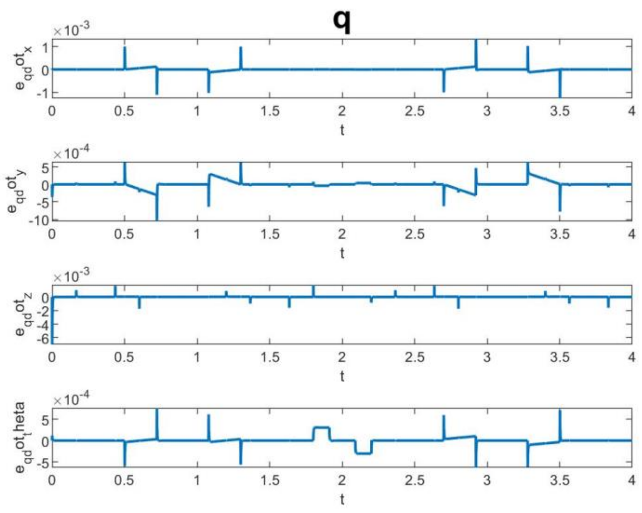

*Figure 9: Error in q*

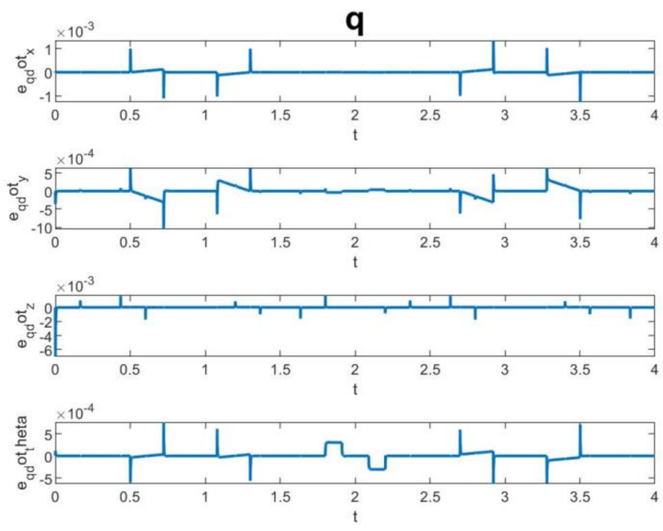

*Figure 10: Error in q_dot*

E) Disadvantages of this inverse dynamic control approach:

This technique requires perfect knowledge of the dynamic, and then sensitivity and robustness problems cannot be avoidable due to unavoidably imperfect compensation. The inverse dynamics control scheme will not lead to a homogeneous error equation. Online computation of all terms is difficult and drive the choice of control architectures it requires that parameters of the system dynamic model are accurately known, and the complete equations of motion are computed in real time, which are quite impossible in practice. Inverse dynamics computation is to be performed at sampling times of the order of a millisecond to ensure that the assumption of operating in the continuous time domain is realistic. Perfect cancellation might not be possible cause of uncertainty of the model and payload or for approximations due to real time constraints This may pose severe constraints on the hardware/software architecture of the control system. In such cases, it may be advisable to lighten the computation of inverse dynamics and compute only the dominant terms. With imperfect compensation, other better schemes should be applied to improve this framework.

## 6. CONCLUSION

**The kinematics and dynamics of the SCARA assembly robot are studied, and the dynamic equations of motions are derived. The setpoints for each joint are generated considering a second order inversion kinematic algorithm, where the Cartesian values are given by the trajectory provided. Furthermore, The trajectory generated is such that the robot does not stop at each waypoint making the waypoints, via points.**