Name: Megha Veerendra

NYU ID: N11967918

PROJECT REPORT

Part 1

Second order inverse kinematics algorithm

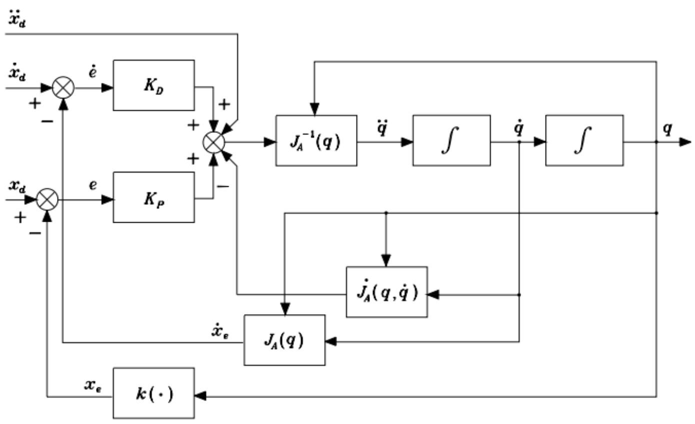

Fig 1: Block scheme of the second-order inverse kinematics algorithm with Jacobian inverse

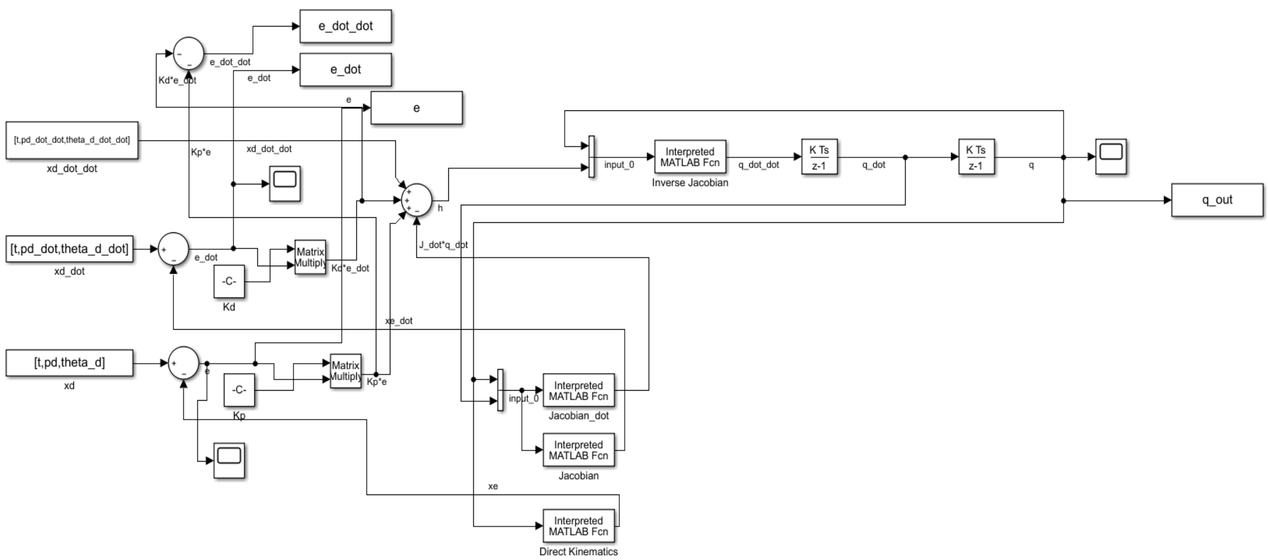

Fig 2: Second order inverse kinematics algorithm Simulink model

Explanation

The aim of part 1 project 2 is to implement a second order inverse kinematic algorithm for a SCARA manipulator. Fig 2 above shows the Simulink model of the implemented algorithm. It follows the schematic shown in Fig 1. I have computed $\dot{x}_e$ = J_A q inside the MATLAB interpreted function *Jacobian*. Similarly, $\dot{J}_A$* $\dot{q}$ is computed inside the MATLAB interpreted function *Jacobian_dot.* The gains Kd and Kp are chose as I*1000 where I is a 4x4 matrix. The results of the simulation are summarized as below:

Joint plots

The Joint space variables Θ1, Θ2, d3 and Θ4 are plotted as below

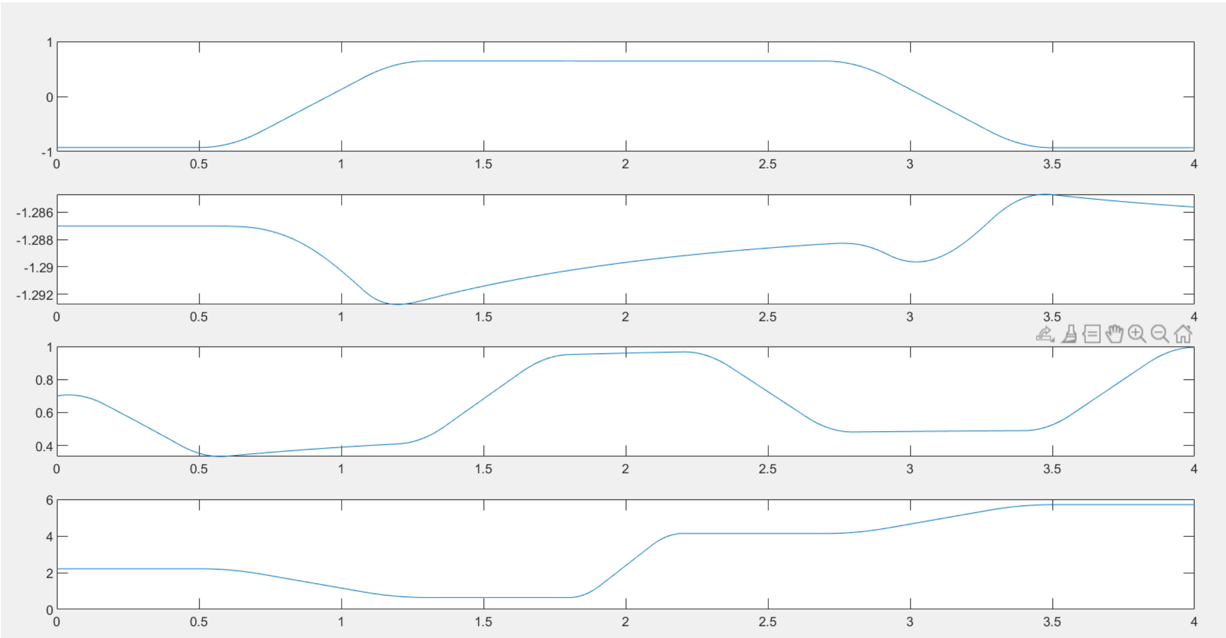

Fig 3: Plots representing joint space variables v/s time

Error Plots

Acceleration error plot

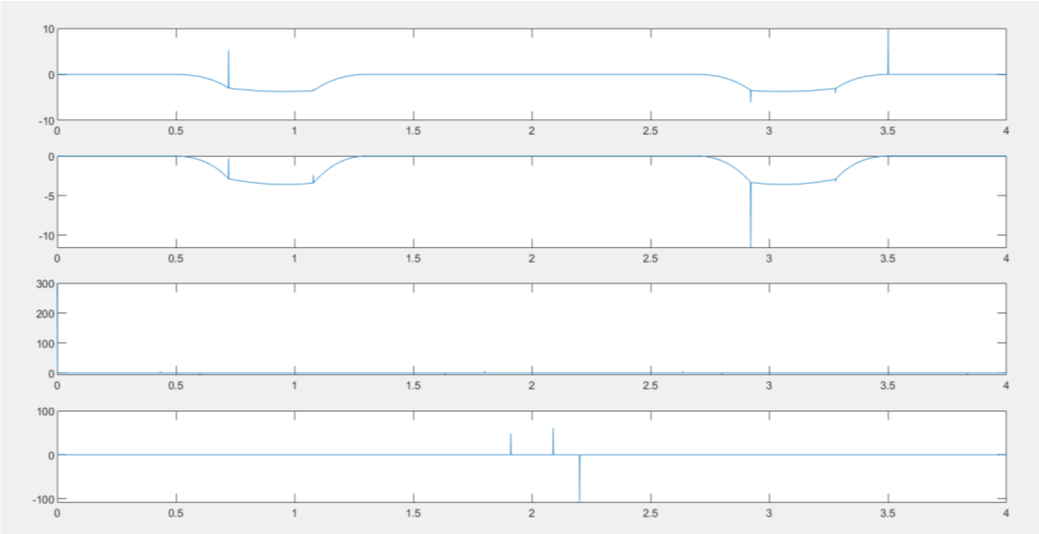

Fig 4: Plots representing acceleration error v/s time

The acceleration error is given by:

$$\ddot{e} + \boldsymbol{K}_D \dot{e} + \boldsymbol{K}_P e = \mathbf{0}$$

From the plots it can be observed that the errors are are almost zero. This shows that acceleration was efficiently controlled.

Velocity error plots

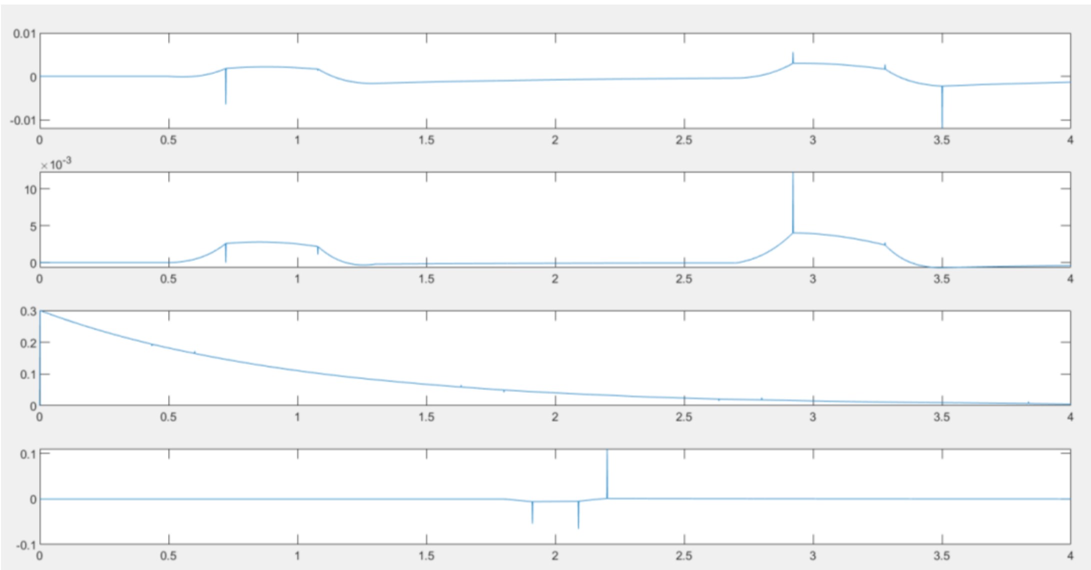

Fig 5: Plots representing velocity error v/s time

The velocity error is given by $\dot{e} = \dot{x}_d - \dot{x}_e$. Again, the plots show that the error is nearly zero. Hence desired velocity is maintained.

Position error plots

The position error is given by $e = x_d - x_e$. The plots show that the desired position is followed in the operational space as the error 'e' is nearly zero.

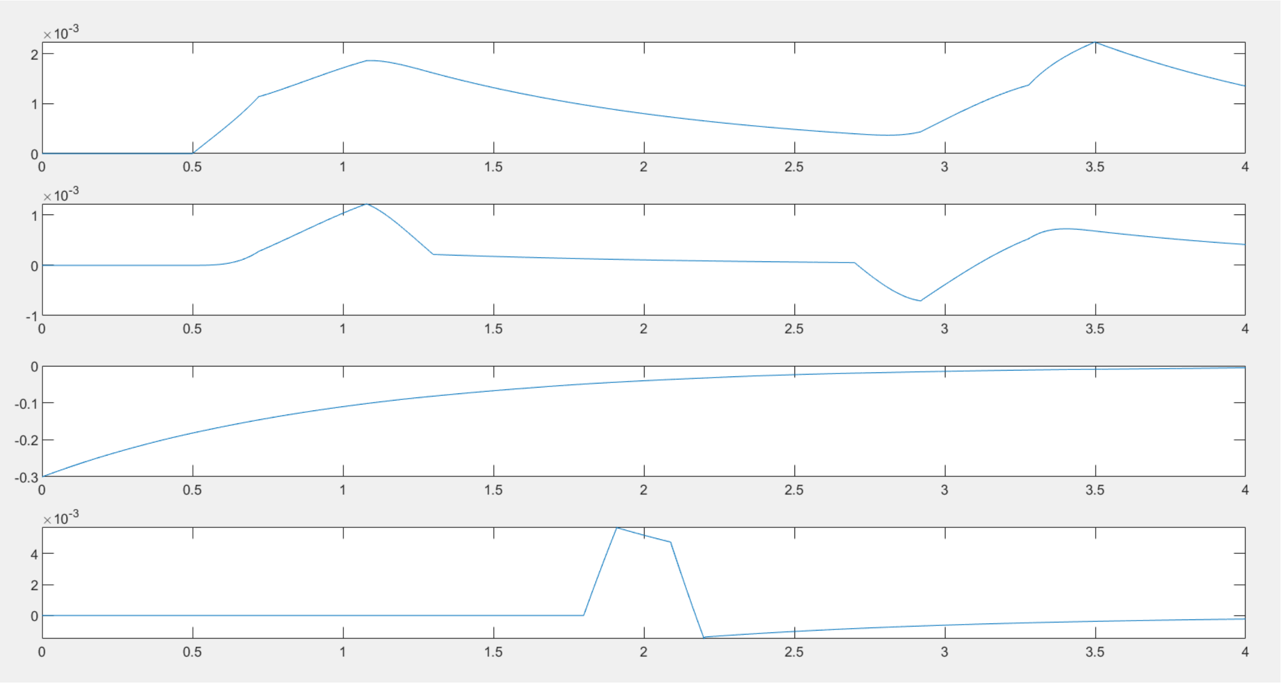

Fig 6: Plots representing position error v/s time

Part 2

Redundancy second order Simulink model

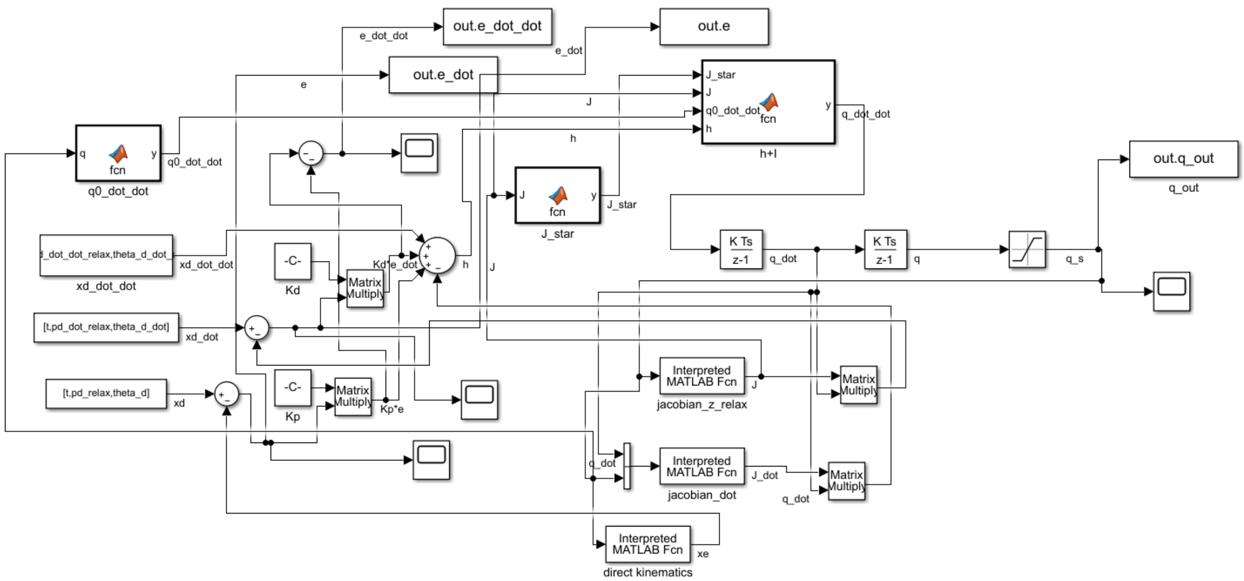

Fig 7: Second order inverse kinematics Simulink model for a redundant SCARA manipulator

Explanation

The aim of the second part of the project is to implement for a redundant manipulator, a second order algorithm and for maximize the distance from the given obstacle. The Simulink model follows the same schematic shown in Fig 2. However, due to relaxation of ‘z’ in the operation space, we drop ‘z’ from the jacobian. The jacobian J_A of the redundant manipulator is 3x4 in dimension. Just as for the redundant manipulator in a first order algorithm, we calculate $\ddot{q}_0$ by differentiating w(q) twice. Since the secondary task here is to maximize the distance from the obstacle, w(q) = ||p - o|| where, ‘p’ is the position vector of the end effector and ‘o’ is the centre of the obstacle which in this case is a sphere. The position vector of the end effector is obtained from the direct kinematics equations.

Joint outputs

The trajectory followed by the redundant manipulator is shown below:

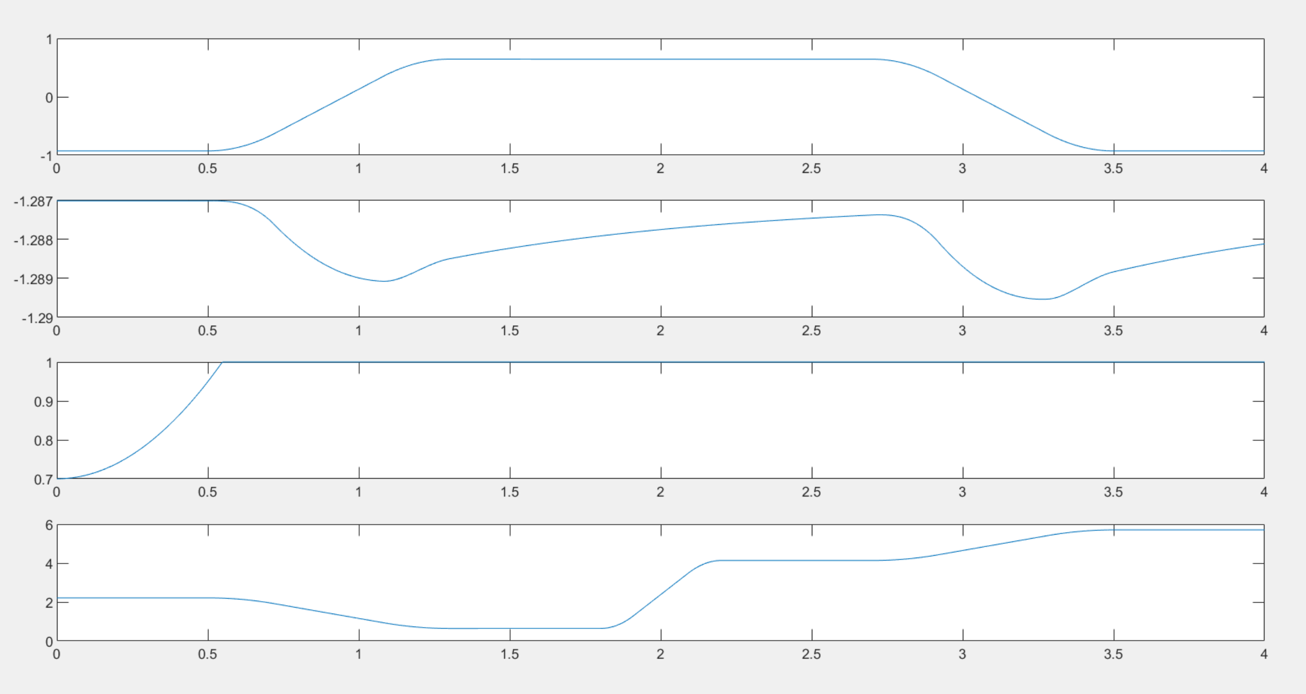

Fig 8: Plots representing joint space variables v/s time

Acceleration Error

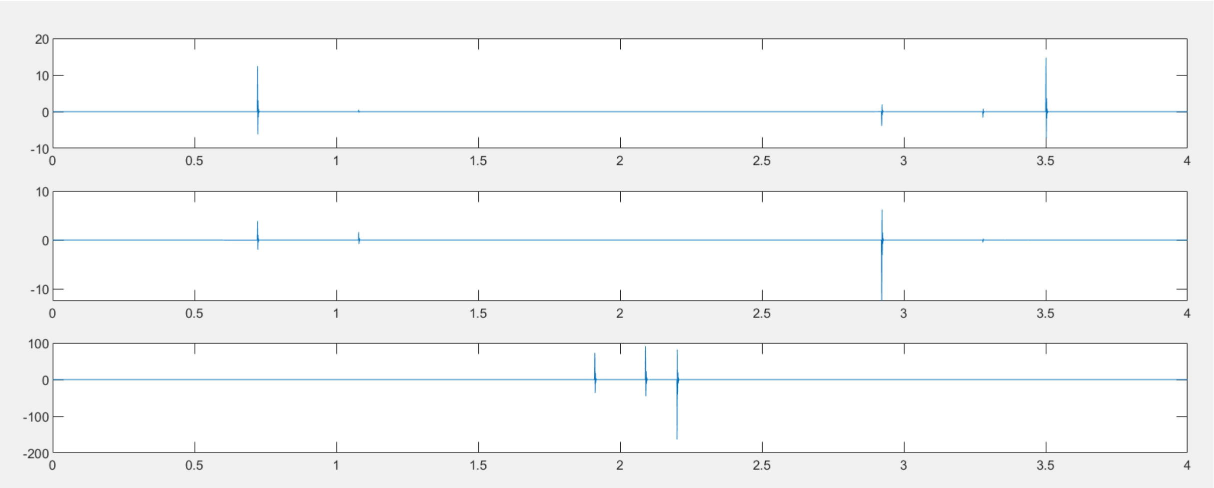

Fig 9: Plots representing acceleration error v/s time

Velocity Errors

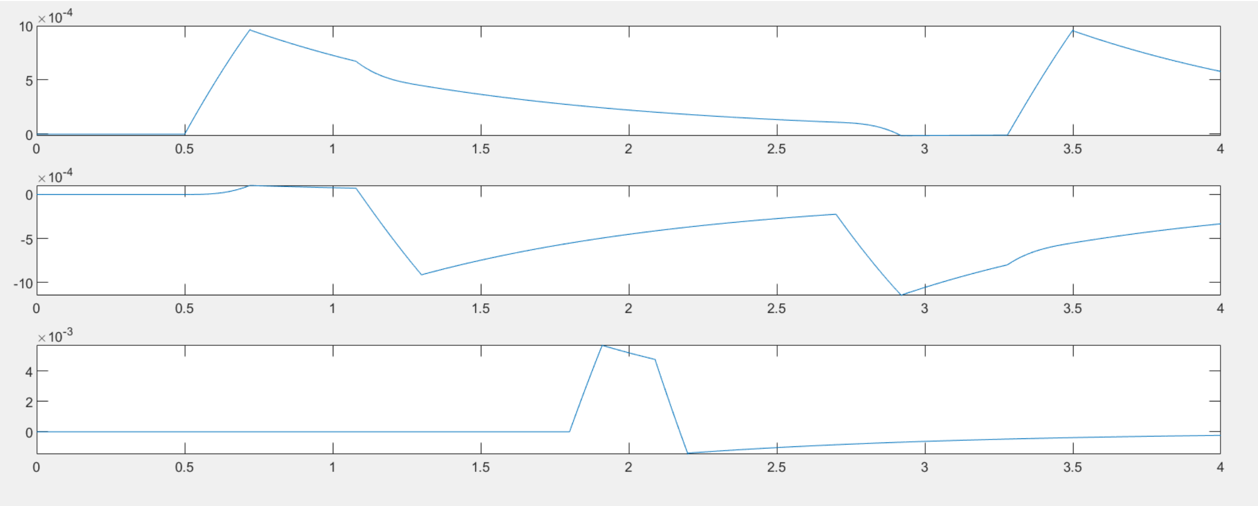

Fig 10: Plots representing velocity error v/s time

Position Errors

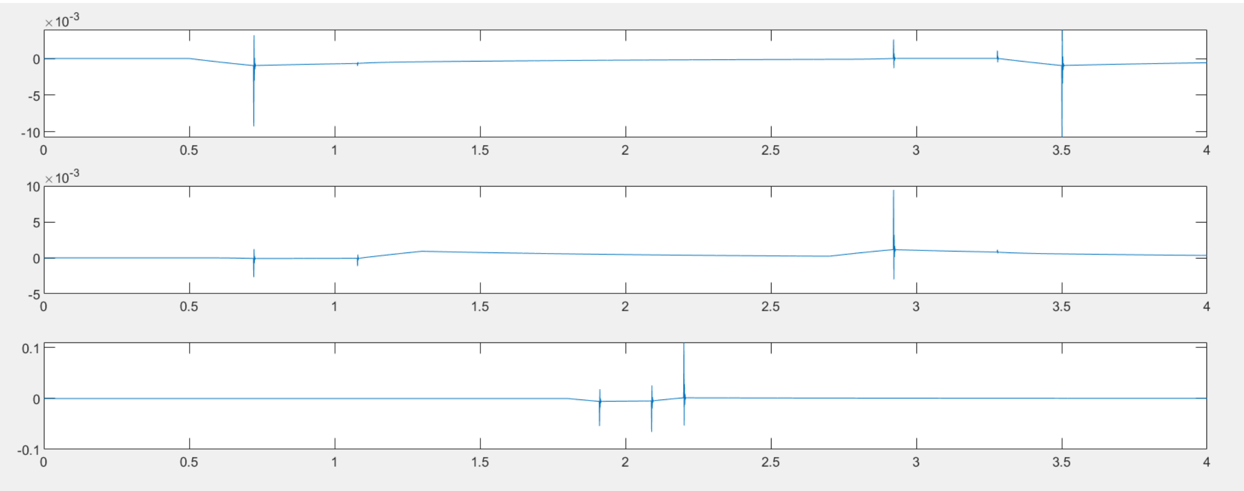

Fig 11: Plots representing position error v/s time

The acceleration, velocity and position errors are computed in the operation space. In the case of the redundant manipulator the equations for these errors remain the same as described in the non-redundant case. However, it can be observed that due to the relaxation of ‘z’, there are only three errors in each operational error instead of the typical four for the non-redundant case. The errors are nearly zero which shows that the desired acceleration, velocity, and position was achieved.