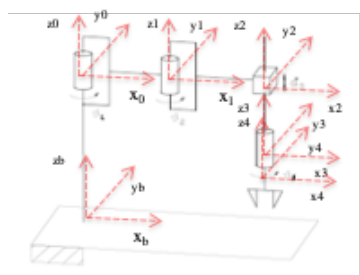
# Foundations of Robotics – Project 1 Report

Chinmay Prashanth

cp3873



Text

Description automatically generated with medium confidence

Table

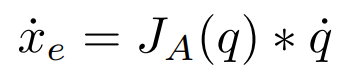
Description automatically generated

Part 1:-

Methodology:

Here, in this part we’ve used Matlab/Simulink algorithm to obtain Kinematic Inversion of Jacobian Inverse and Transpose.

Below is the Jacobian 4X4 matrix, used to track 4 DOF’s in operational space of an End Effector. The differential equation of Forward Euler Integration is used

Forward Euler Integration

Here we’ve obtained the analytical jacobian by differentiating the position and orientation of end effector. Joint velocity vector are used to compute joint components at each time instant for desired end effector velocity.

The jacobian\_inverse and jacobian\_transpose function have used the below 4 rank jacobian matrix. In Jacobian Inverse, the inverse of Jacobian matrix has been multiplied by Kin which is the desired velocity at the end effector. In Jacobian transpose, the transpose of Jacobian has been multiplied by the Kin. Both functions have been defined in their respective SimuLink Models.

Jacobian = Graphical user interface, application

Description automatically generated with medium confidence

The Position and Orientation expression for the end effector is defined in the direct\_kin file. The translation part of the homogeneous transformation from base to end effector is considered as 3X1 form then the angles are included in new matrix to obtain Position and Orientation.

HT(Base-End Effector) = A picture containing text

Description automatically generated

[Position, Orientation] = Graphical user interface, application

Description automatically generated with medium confidence

Finally, after defining all the function in their respective Simulink file, all the visualization and loading the trajectory from the predefined file is happening in init.m with the help of Kinematic\_traj and visualize\_results files.

Result:

Simulink Model:

Diagram

Description automatically generated

Jacobian Inverse

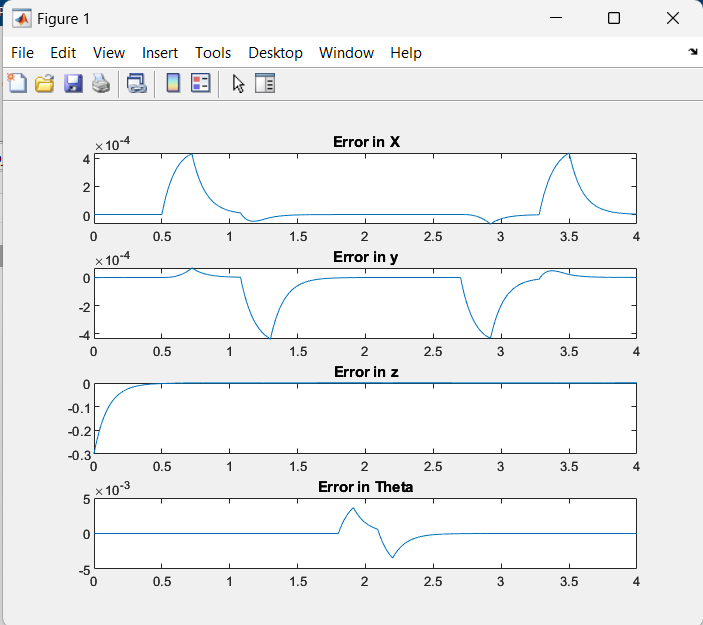
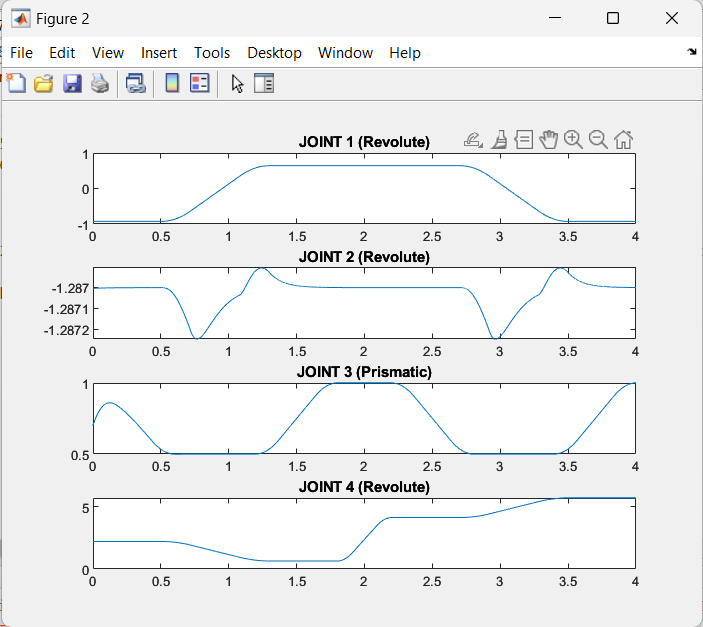
**Diagram

Description automatically generated**

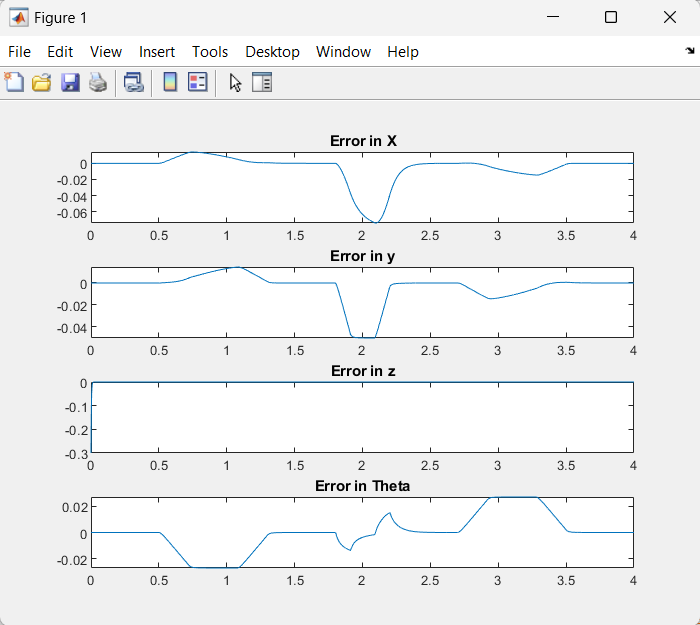
Jacobian Transpose

Here, the Simulink Model for both Jacobian Inverse and Transpose have been created in clik\_inverse and clik\_transpose Simulink files. All the input variables are being imported from the workspace by using ‘From Workspace’ blocks and the all the output variables are exported back to workspace by ‘To Workspace’ blocks which are being visualized.

Plots:

Jacobian Inverse

 Engineering drawing

Description automatically generated

Jacobian Transpose

Part 2:

Methodology:

The redundancy of a SCARA manipulator is used by relaxing the revolute joints (Z axis) in the workspace. The sum of separate joint angular velocity will be relaxed. Due to the relaxation, we obtain a non-square jacobian matrix (similar to the previous jacobian matrix but with last row removed).

Jacobian matrix = Graphical user interface, text, application

Description automatically generated with medium confidence

The above non square matrix is representing a system, with 4 variables for 3 equations, which means that will have infinite solutions for velocity. Velocity equation is considered as,

A picture containing clock, watch

Description automatically generated

A picture containing diagram

Description automatically generated

A close-up of a clock

Description automatically generated with low confidence

Result:

Simulink model:

Diagram

Description automatically generated

Jacobian\_phi\_relax

Here the jacobian\_phi\_relax is implemented into the model in the interpreted Matlab function and Position variables are defined in the direct\_kin.

Plots:

Diagram, engineering drawing

Description automatically generated

Diagram, engineering drawing

Description automatically generated