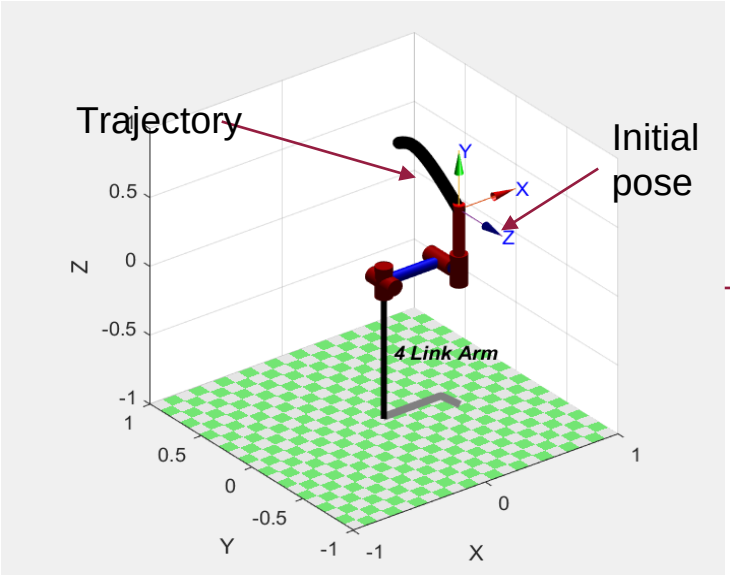
Planning & control of a 4 -DOF manipulator using inverse kinematics

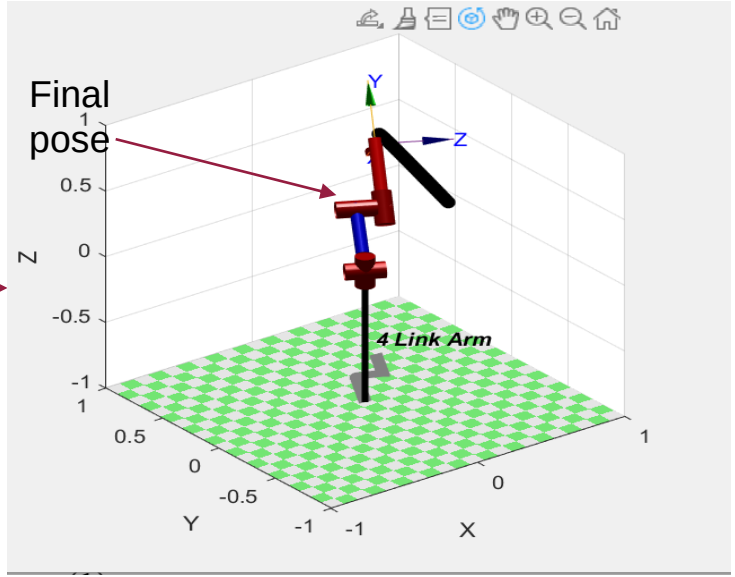
Names	Affan Bin Usman	Vishnu T. P	Vihari Gandrakota	Sidhant Das
Asurite ID	ausman4	vpishara	vgandrak	sdas116

PROBLEM STATEMENT

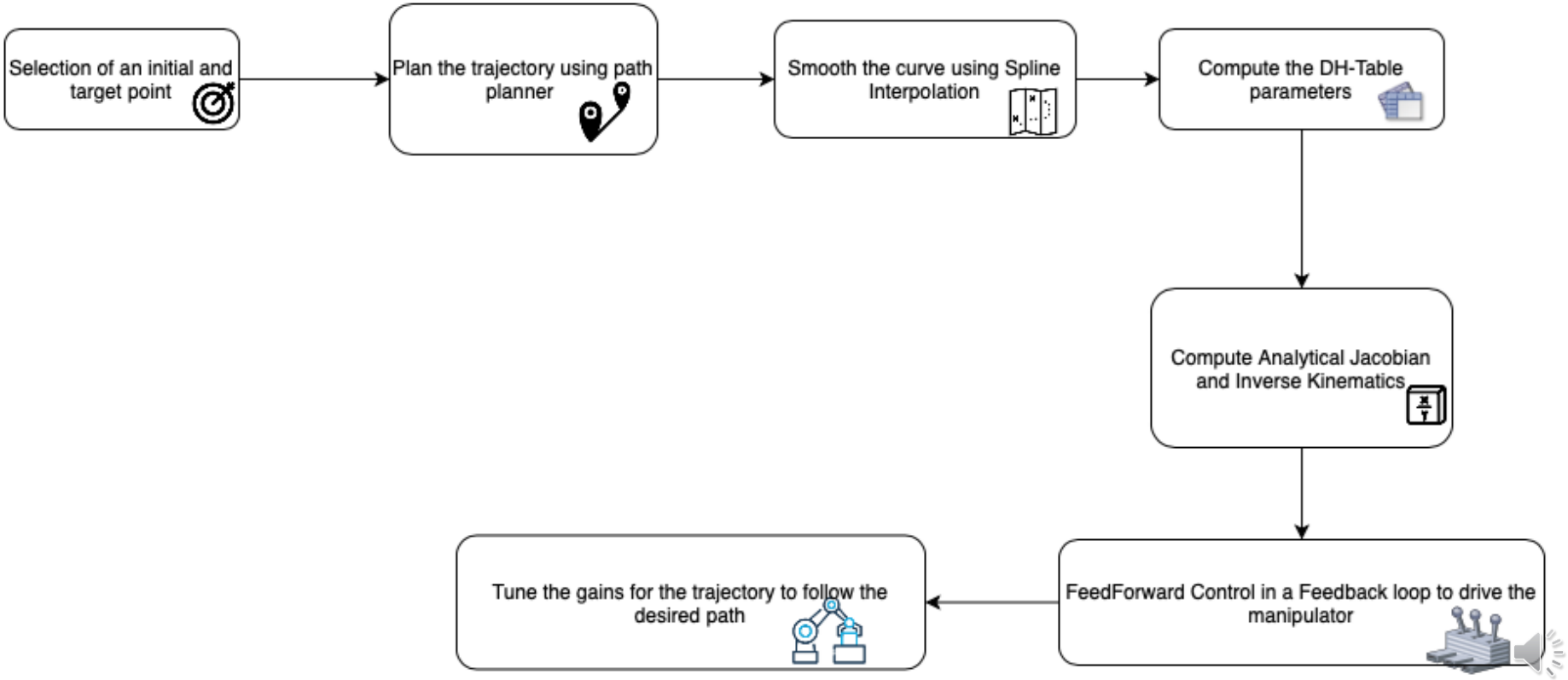
- Considering a 4-DOF manipulator with all the 4 joints as revolute and assuming certain DH parameters for the manipulator, we want the manipulator to navigate from an initial pose to a desired final pose.

$$\frac{2\pi}{3} < \theta_i < -\frac{2\pi}{3}$$

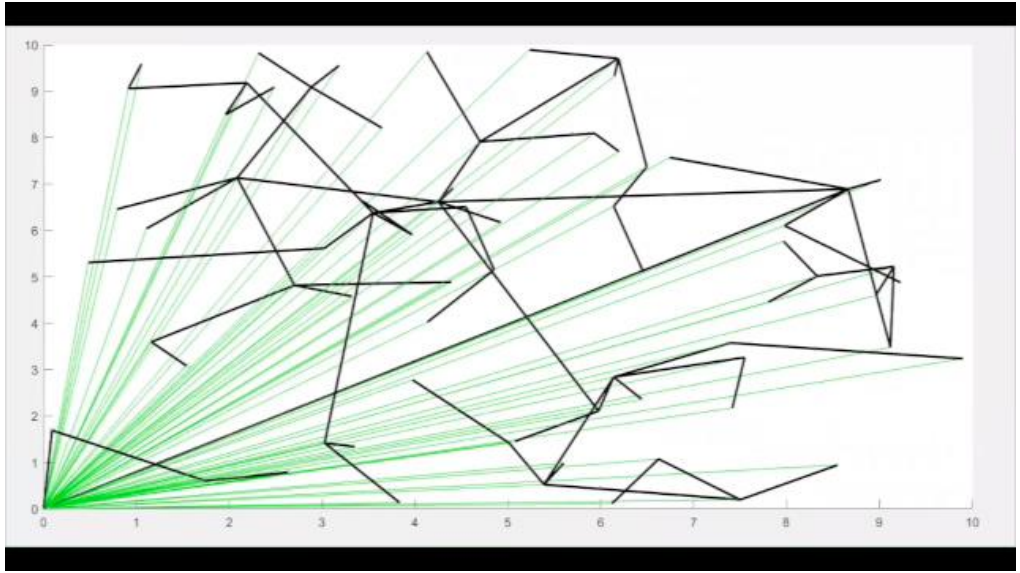




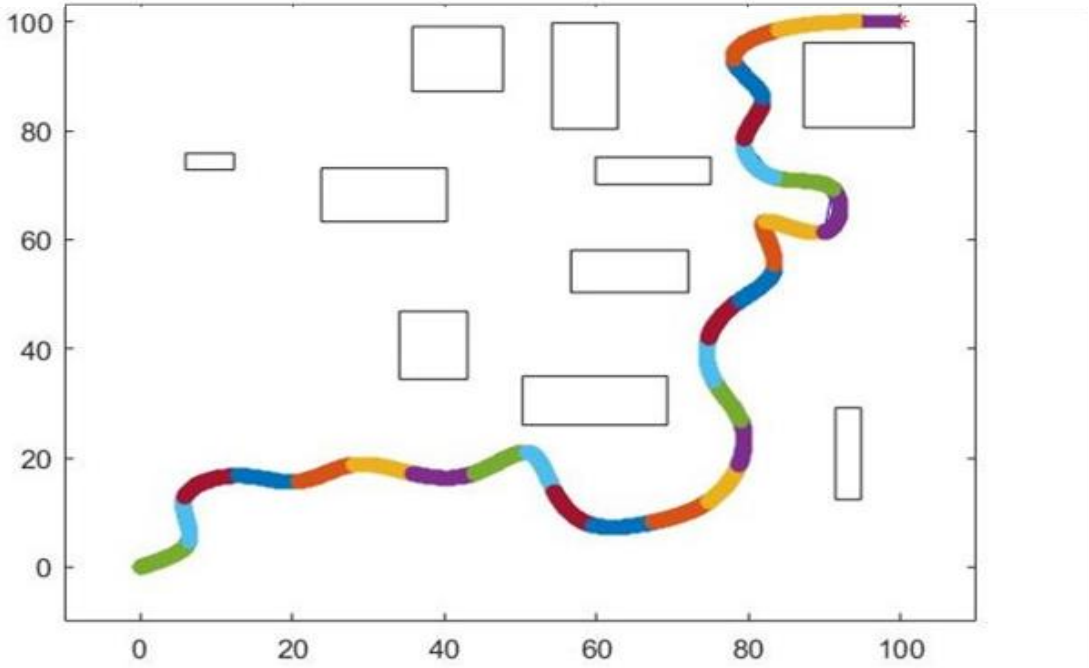
PROCESS FLOW



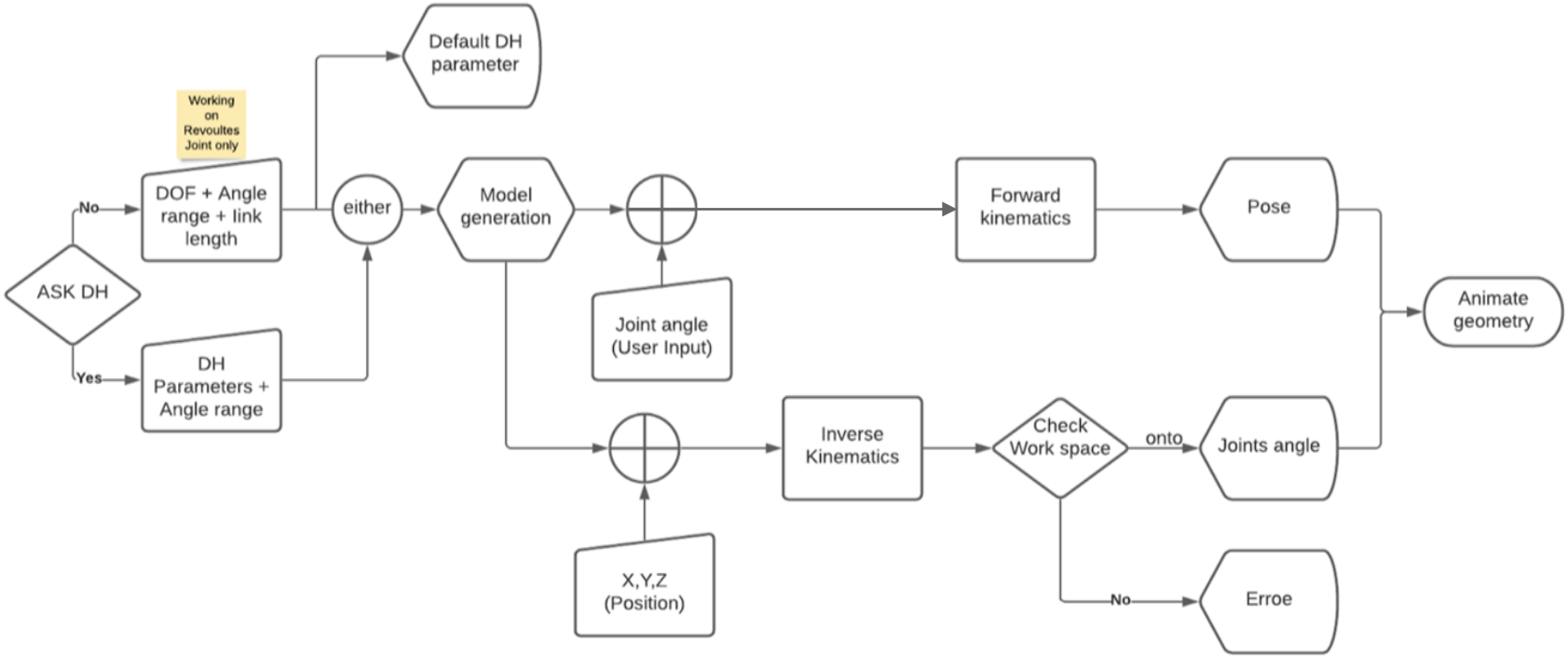
RRT*



Interpolation

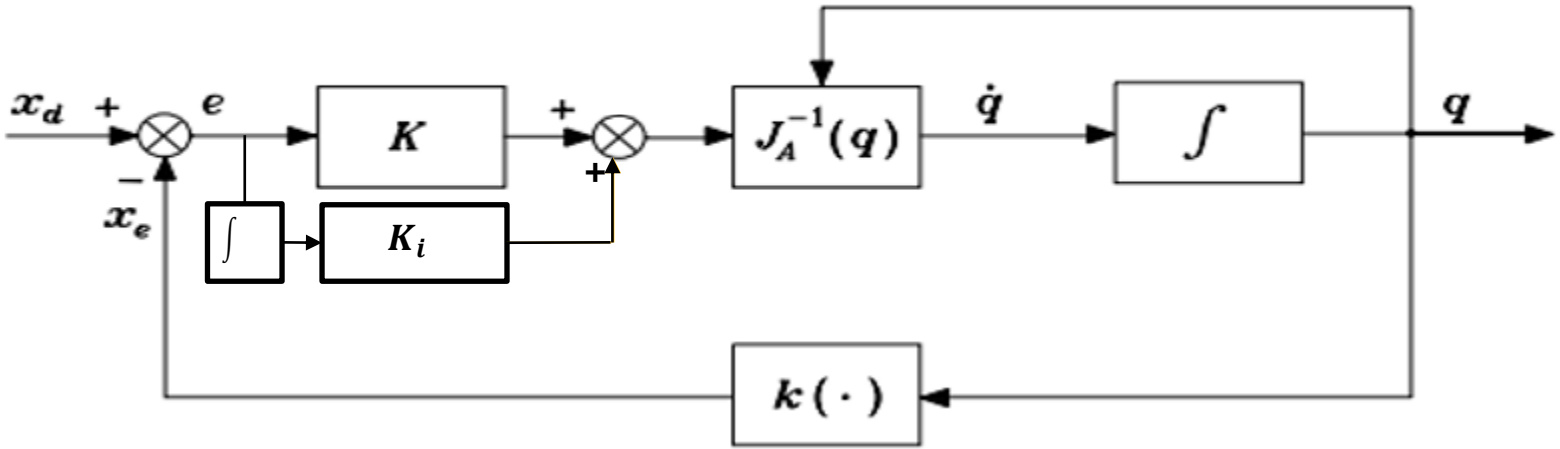


Working schematics



Controller

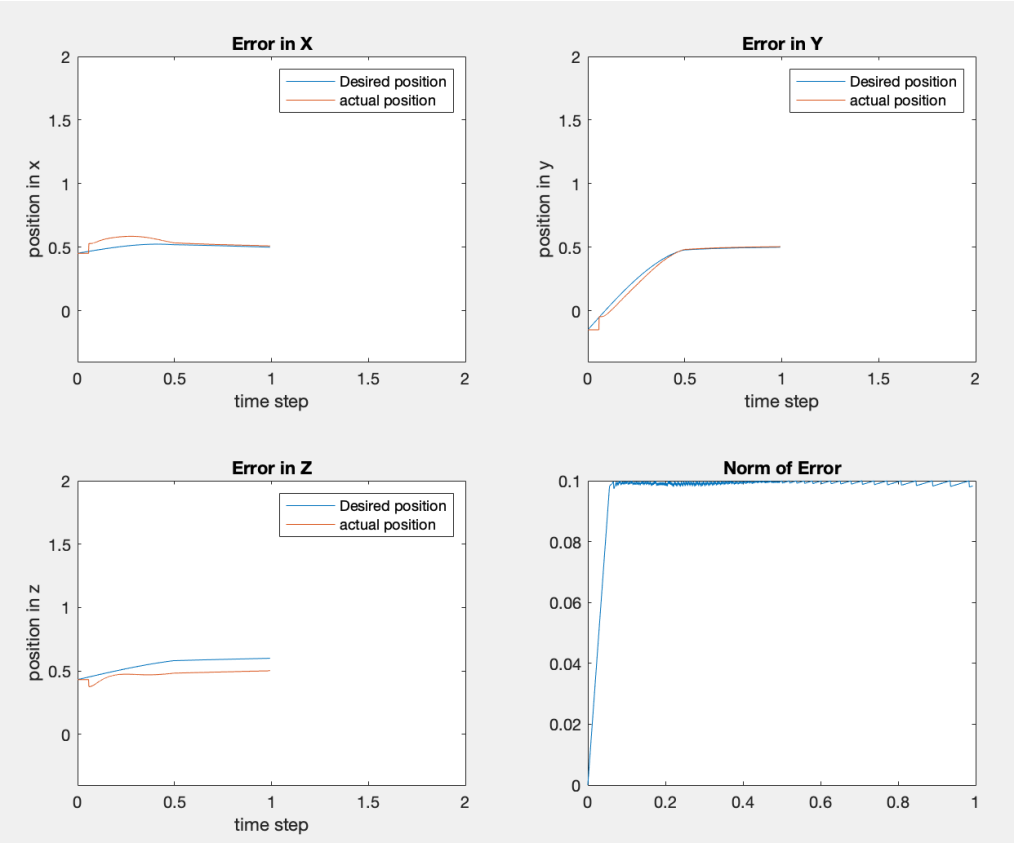
Proportional Integral Controller



DEMO

Results

- The end effector of the manipulator was able to reach the desired pose following the desired trajectory.
- Some initial deviation from trajectory is observed but after certain time the controller is able to make the manipulator follow the desired trajectory.
- Some steady state error does exist since its just a proportional controller



Code Description

The following functions have been defined and implemented in our code:

- **rrt_star_plan_obs**
  Plans the path from the start point to the goal coordinates.
- **forward_kinematics**
  Finds the transformation matrix from base frame to end effector frame.
- **analytical_jacobian_val**
  gives the analytical jacobian if the input joint positions are inputted.
- **workspaceforproj**
  Plots the workspace of the manipulator
- **ellipse_plot**
  To find and plot the velocity ellipsoid in 3D.
- Controller Parameters
  Proportional gain=30 & Integral gain=0.05

Steps for Execution

Software Used: MATLAB

MATLAB Packages To Be Installed

- RRT and RRT* for 3D --- https://www.mathworks.com/matlabcentral/fileexchange/105615-rrt-and-rrt-for-3d
- 2D/3D RRT* algorithm --- https://www.mathworks.com/matlabcentral/fileexchange/60993-2d-3d-rrt-algorithm
- Curve Fitting Toolbox version 3.6 by MathWorks
- Statistics and Machine Learning Toolbox version 12.2 by MathWorks
- Robotic Toolbox for MATLAB version 10.4 by PETER CORKE

The above files can be added using MATLAB add on manager either directly through the manager or going to the above link and installing it
to the MATLAB add on manager.
If the files are downloaded using the link not on MATLAB but somewhere in system, they must be first saved in the execution folder.

Steps To Run The Code:

- run file named finalcode.m (type finalcode in the MATLAB command window after making sure the file is added to the path)
- Provide the inputs as indicated.

Discussion

- Challenges of implementing inverse-kinematics from scratch.
- Drawbacks of existing controllers.
- How these drawbacks can be overcome through other approaches.
- How this work can be extended to arms with multiple joints and higher DOF.
- The Kinematic model can be incorporated with a dynamic controller, in an adaptive manner.
- Algorithms like particle swarm optimization can be used to compute inverse kinematics without using analytical Jacobian.

Team Contributions

Team Member Name	Contribution	Contribution Percentage
Affan Bin Usman	Velocity ellipsoid, App designer - GUI Design, Trying implementation of OpenMANIPULATOR-X on Ubuntu (18.04) ROS (melodic), Workspace, Code integration	25%
Vishnu T. P	RRT, Trajectory planning, PI Controller, Presentation, Inverse Kinematic algorithms, Code integration	25%
Vihari Gandrakota	Research on different inverse kinematic algorithms, Analytical Jacobian, Trying implementation of OpenMANIPULATOR-X on Ubuntu (18.04) ROS (melodic), Presentation, Code integration	25%
Sidhant Das	Transformation matrix calculation, Computation of DH parameters, N-DOF robot making interactive code, Presentation, Workspace, Code integration	25%

Our team met on a weekly basis to discuss theory behind our project & the execution of the code to that theory. Each member of the team has had full attendance & participation within each of these project meetings.

Thank You!