Comparison of various trajectory-tracking controllers using UR5 robot

Aditya Gupte, Gokul Narayanan, Joshua, Sinan Morcel,

I. Abstract

This project compares three different controllers in terms of trajectory tracking performance. Specifically, we went over three different four different controllers, Inverse Dynamics Controller, Passivity-based Controller, Robust Controller, Passivity-based Robust Controller. Passivity-based controller prooved to be superior.

II. Introduction

Manipulators are widely used in tasks such as painting, welding, pick and place tasks. The tasks like pick and place need a point to point motion where the trajectory followed by the robot is not a concern. Whereas for tasks such as painting and welding the robot needs to follow a desired trajectory. The task of tracking some desired trajectory effectively has been studied for some time, and efforts to find an effective controller which can track a desired trajectory accurately were put forth in the literature. Many different controllers have been suggested to solve the problem with varying performance.

The Inverse Dynamics controller is among the simpler non-linear controller techniques, according to [2]. However, despite its simplicity, it provides an example of ideal manipulator trajectory-tracking performance given that the complete dynamic model of the robot is available. In-case of model uncertainties, Robust control can be used if uncertainty is bounded ; Adaptive control is also used when there is uncertain parameters that need to be learned. The Robust Inverse Dynamics controller and the Passivity Based Robust Controller are examples of robust controllers. The latter type is used when establishing a constant bound over the uncertainty is desired. The authors in [3] conducted a survey of different controllers including Feedback Linearization (they call it computed torque controller (CTC)), Variable Structure Compensator, Passivity Based Controllers, Disturbance Observer Based Controller. Both the Feedback Linearization Controller and the Passivity Based Controller guarentee asymptotic stability (in the second case the body has to have a passive nature). The first can have global asymptotic stability and the non-linearities in the dynamics cancel out when included in the controller. The drawbacks of the CTC is that the exact dynamics need to be given and additional controllers need to be used to account for the non-linear Actuator Saturation and Friction parts of the dynamics. As for Passivity Based Controllers, on the other hand, tuning the parameters is more difficult.

III. Problem Statement

The aim of this work is to compare the performance of the following controllers in task space: an Inverse Dynamics Controller, a Passivity Based Controller, a Robust Feedback Linearization Controller and a Passivity Based Robust Controller following a set of desired trajectories.

IV. Methodology

The methodology of choice in this project is shown in the block diagram in Figure 1.

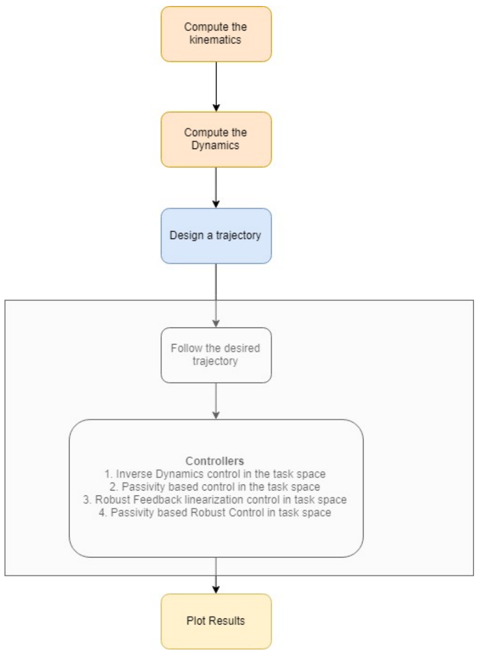
Compute the kinematics
Compute the Dynamics
Design a trajectory
Follow the desired trajectory
Controllers
1. Inverse Dynamics control in the task space
2. Passivity based control in the task space
3. Robust Feedback linearization control in task space
4. Passivity based Robust Control in task space
Plot Results

Fig. 1: Methodology

A. Simulating Hardware platform

The manipulator which we have chosen for simulating and testing our controller is UR5 shown in Figure 2. It is a 6 Degrees of freedom serial manipulator.

B. Kinematics of UR5 manipulator

The kinematics of the UR5 manipulator were calculated using the DH parameters [1] listed in Table 1.

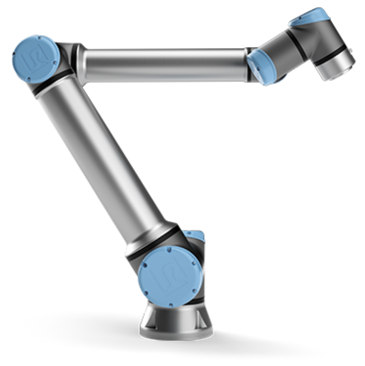

Fig. 2: UR5

link	$\alpha(^\circ)$	a **(mm)**	d**(mm)**	$\theta(^\circ)$
1	**90**	**0**	**89.2**	**90**
2	**90**	**425**	**0**	**90**
3	**0**	**392**	**0**	**0**
4	**90**	**0**	**109.3**	**90**
5	**-90**	**0**	**94.75**	**0**
6	**0**	**392**	**82.5**	**0**

TABLE I: DH Parameters

C. Dynamical modelling of UR5 manipulator

We will be using the Euler-Lagrangian method to derive the dynamics of the manipulator. The Lagrangian L is given by

$$L = K - P$$

where K is the kinetic energy and P is the potential energy. Using the Jacobian of the manipulator we find out the velocities of each link. The linear kinetic energy is given by $\frac{1}{2}mv^2$ and the rotational kinetic energy is given as $\frac{1}{2}I\omega^2$. The potential energy of each link is given by mgz where z is the distance from the origin to the center of each link along the z-axis (assuming gravity acts along the z-axis). Now, the torque to be generated at each link is given by

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i$$

This equation allows us to compute the following dynamic model of the manipulator.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$

In our work we are controlling only the positions in X, Y, Z frame. Since the first 3 joint angles are responsible for position of the end effector we set θ_4, θ_5 and θ_6 to be 0. When the last 3 joint angles are constant our mass matrix and coriolis matrix becomes a 3×3. The Jacobian also is now a 3×3 matrix. An additional simplification in the model is in the case of a robust feedback controller and passivity based robust controller we consider an error in the mass of the first 3 links only.

D. Controllers

The following controllers will be used for the purpose of carrying out the comparison.

1) Inverse Dynamics Controller: Inverse Dynamics controller is also known as computed torque control. The dynamics of the manipulator are given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$

We track the error and make it converge to zero. The error dynamics equation is given as

$$\ddot{e}(t) + K_1 e(t) + K_0 e(t) = 0$$

Using the equation

$$\dot{e} = Ae + Bv$$

Where $v = u - u_d$ we get the torque which is to be given as the input to the system. $u = \ddot{q}^d(t) + K_0 q^d(t) + K_1 \dot{q}^d(t)$ In our work we are doing the control in the task space. We first find the acceleration in the task space

$$a_X = \ddot{X}^d + K_P\left(X^d - X\right) + K_D\left(\dot{X}^{\dot{d}} - \dot{X}\right)$$

To convert this to joint space we use use

$$a_q = J^{-1}\left\{a_X - J\dot{q}\right\}$$

where J is the Jacobian and $J\dot{q}$ is the time derivative of the Jacobian. Using a_q we find the torque and supply it to the manipulator. Inverse dynamics relies on the exact cancellation of the non-linearities in the robot. Therefor one of the disadvantage of the inverse dynamics controller is the mass, Coriolis and gravity matrix needs to be accurate.

2) Passivity Based control in task space: In passivity based control we change the coordinate system to r, v and a. Where

$$v = \dot{q}^d - \Lambda\tilde{q}$$
$$a = \dot{v} = \ddot{q}^d - \Lambda\dot{\tilde{q}}$$
$$r = \dot{q}^d - v = \dot{\tilde{q}} + \Lambda\tilde{q}$$

$\Lambda$is a diagonal matrix consisting of the positive gains. The control input is given in terms of a and v as

$$u = \hat{M}(q)a + \hat{C}(q,\dot{q})v + \hat{g}(q) - Kr$$

For control in the task space the equations modify to

$$r_x = \dot{e}_x + \Lambda e_x$$
$$\mathrm{v}_x = x_d - \Lambda e_x$$
$$\mathrm{a}_x = \dot{x}_d - \Lambda e_x$$

a_x is then converted to joint space using

$$a_q = J^*\left(a_q - J(q)v_q\right)$$
$$\mathrm{v}_q = J^* v_x$$

Where J^* is the pseudo-inverse of the jacobian.
Passivity based controller does not seek to linearize or decouple the system. It is mainly concerned about the asymptotic convergence of the error to zero. In case of small uncertainties in the model the performance of the passivity based controller is better than inverse dynamics controller. The controllers we have discussed so far does not take into account the errors in mass, coriolis or gravity matrix.There also might be some noise in the system. In such cases the performance of the controller is affected. To overcome this we use the Robust

feedback Linearization controller and the Passivity based Robust controller. Both these controller take into consideration the errors in the model.

3) Robust Feedback Linearization: In the real world the equations of motion are nonlinear. Moreover an accurate model of the equation involves a lot of uncertainties.For a manipulator the equation of motion is given by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u$$

To differentiate between the real world values and the computed values let us denote the computed values using $(\hat{\cdot})$. The equation now becomes

$$u = \hat{M}(q)a_q + \hat{C}(q,\dot{q})\dot{q} + \hat{g}(q)$$

From these two equations we get

$$\ddot{q} = a_q + M^{-1}\left(\tilde{M}a_q + \tilde{C}\dot{q} + \tilde{g}\right)$$

The term $M^{-1}\left(\tilde{M}a_q + \tilde{C}\dot{q} + \tilde{g}\right)$ captures the effect of uncertainties. We represent this term as η. Our aim now is to come up with a control system that would handle this term so that we get a stable solution at the equilibrium. To our control input a_q we add a term δa that helps us do this. δa is the control input designed to overcome the effects of uncertainties in the model. We observe that equation for a_q is nonlinear. To solve this we use Lyapunov's second method. This system is stabilized be considering a PD control. In the task space the acceleration input is given as

$$a_X = \ddot{X}^d + K_P\left(X^d - X\right) + K_D\left(\dot{X}^d - \dot{X}\right)$$

The acceleration in the joint space is computed using

$$a_q = J^{-1}\left\{a_X - J\dot{q}\right\} + \delta a$$

To compute δa we find a scalar function $\rho(e,t) \geq 0$ that would guarantee that the actual trajectory does not deviate too much from the desired trajectory.

$$\|\eta\| \leq \rho(e,t)$$

We assume a bound in $\|\eta\|$ of the form $\|\eta\| \leq \alpha\|\delta a\| + \gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3$. Where γ_i are non-negative constants. If we assume $\|\delta a\| \leq \rho(e,t)$ then we get an expression for ρ as

$$\rho(e,t) = \tfrac{1}{1-\alpha}\left(\gamma_1\|e\| + \gamma_2\|e\|^2 + \gamma_3\right)$$

When we define the δa as

$$\delta a = -\rho(e,t)\tfrac{B^TPe}{\|B^TPe\|}$$

the Lyapunov function $V = e^TPe$ satisfies $\dot{V} \leq 0$.

4) Passivity Based Robust Control: Passivity based control is better than the previous controllers discussed. In this method we set the control input u as

$$u = \hat{M}(q)a_q + \hat{C}(q,\dot{q})v_q + \hat{g}(q) - Kr$$

Where a_q, v_q, r is calculated using the equations Let us take $Y(|q,\dot{q},a,v) = [a_q, v_q, 1]$ and $\hat{\theta} = [\hat{M},\hat{C},\hat{G}]^T$ Where $Y\hat{\theta} = u + Kr$ and

$$\hat{\theta} = \theta_0 + u$$

Our equation now becomes

$$u = Y(q,\dot{q},a,v)\hat{\theta} - Kr$$

Therefore we get,

$$\begin{gathered} M(q)\dot{r} + C(q,\dot{q})r + Kr = Y\left(\theta - \theta_{0,}\right) \\ M(q)\dot{r} + C(q,\dot{q})r + Kr = Y(a,v,q,\dot{q})(\tilde{\theta} + u) \end{gathered}$$

where $\tilde{\theta} = \theta_0 - \theta$. $\tilde{\theta}$ captures the uncertainty in the system. If there exists ρ such that $\|\tilde{\theta}\| = \|\theta_0 - \theta\| \leq \rho$ then the control input u is given as

$$u = \begin{cases} -\rho\frac{Y^Tr}{\|Y^Tr\|} & ; \quad \text{if} \quad \left\|Y^Tr\right\| > \epsilon \\ -\frac{\rho}{\epsilon}Y^Tr & ; \quad \text{if} \quad \left\|Y^Tr\right\| \leq \epsilon \end{cases}$$

E. Experiment

The bench mark trajectory-set that was used in the comparison, similar to [4] except for polynomial trajectories, is the following:

- a circular trajectory
- a straight line trajectory

Those trajectories were chosen in a way such that they did not violate the limitations of the arm in terms of singularities and dynamics. Only one instance of each type of trajectories was used.

Each controller's actual trajectory-tracking performance will be reported individually and compared to the desired one. Plots will articulate the results in the baseline deliverable, while a simulation showing the trajectory being tracked by the manipulator for additional insight will be shown as part of the optimistic deliverable-set.

F. Implementation: The Pipeline

1) Operating the Pipeline: To generate our results, we used a pipeline whose operation will be described in the following paragraph. The pipeline allows us to 1) simulate the dynamics and the controllers action on the robot, 2) visualize the manipulator as it tracks the trajectory that we provide and 3) validate that we are tracking the trajectory by showing plots with the desired and actual trajectories plotted. Using SimuLinks graphics engine and tools, we can thus generate videos from the data that our dynamics simulations give, albeit with some caveats that we will talk about later.

This pipeline is a set of Matlab scripts that we run in sequence. The first one of the scripts initializes the SimuLink model by adding to Matlabs path variable the locations of the files that describe the robots meshes and other properties. It also initializes some variables that SimuLink needs to set some properties of the robot (i.e. friction coefficient). After running this file, we can then open the file ur5_v5.slx in SimuLink. This version of the SimuLink project reads the sequences of theta values from the workspace of Matlab and then prescribes them to the first three joints (we managed to have the make the controllers converge by freezing the last three joints). Initially, there would be nothing to prescribe, so we run any of the Matlab scripts whose name is based on the controller used and the trajectory that is being tracked. For example, InverseDynamicsControllerCircle.m tracks a circle using an inverse dynamics controller while InverseDynamicsController-Line.m tracks a line with the same kind of controller (the gains might have been turned to improve the performance of tracking this specific trajectory). Once one of these files is run, we get a plot of the tracking performance and the workspace gets populated with variables among which theta1, theta2 and theta3 are of interest for us. Those variables are read by the SimuLink project. This is done automatically when we click the green Run button. SimuLink would then compile for some time and produce a sequence of prerecorded 3D frames in a Matlab tab. We can change the perspective and even record a video using the Record Video after right clicking one of the windows. As an artifact of the previous attempts to directly use the torque controllers we were developing in SimuLink (when we were prescribing torque instead of motion directly in SimuLink), we developed a code that reads the theta values from the workspace of Matlab that SimuLink would have written after running a simulation and then computes the forward kinematics and plots the position of the end-effector in a Matlab plot for us to validate whether we were actually tracking the trajectories — at that time, when we were directly specifying torque in SimuLink, we never got to see any satisfying result.

The following list summarizes the steps to run the pipeline:

1) Run data_file.m (note that you should be inside the directory of the file in Matlab)
2) Load the SimuLink model ur5_v5.slx
3) Run one of the trajectory tracking controllers having the following naming format: ControllerNameTrajectoryType.m
4) Run the SimuLink simulation.

In the following section, the implementation-details of each phase is explained.

2) Implementation Description: **The SimuLink Model:** The SimuLink model contains a deprecated element: the controller (a Matlab function block) which was previously used to control the arm using direct torque prescription. It also contains blocks that specify the simulation-environment properties, like gravity and the world-frame, how the model interacts with the world, like transformation blocks (i.e. homogeneous transformations) and how it is structured, like joints, transformation blocks and solid bodies (the links). The inertia and masses are expressed in the solid-bodies blocks (we had to set them to zero when we decided that we only needed SimuLink to generate the animations only and not the physics and simulations). Finally, the file also contains the blocks that load the thetas as signals from the Matlab workspace.

We generated the structure of the robot by loading a URDF file of the UR5 robot, acquired from the official website that provides information about this robot (data-sheets, CAD models, etc). We then built the surrounding infrastructure that can be currently seen in the last version of the SimuLink file (version 5). The controller is commented out. Figure 3 shows the described SimuLink model. The blocks in red are those that fetch the thetas from the Matlab workspace; the red color indicates that those variables were not instantiated and that at least one controller file needs to be run.

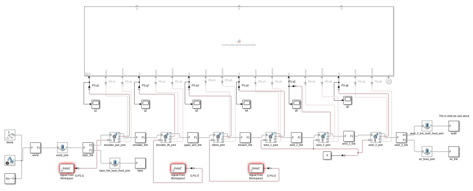
Fig. 3: A snapshot of the SimuLink model.

The Controller Files: The controllers are files that we run to 1) acquire plots for performance assessment and 2) populate the workspace with successive theta values. Each consists of an ODE function-definition that simulates the dynamics of the system and the actions of the controller thereon for a specific time step, taking as input the state of the system so far (q, and possibly q) and return a state-update vector that gets added to the state vector. This definition, in each controller file, is fed to an integrator and used to generate the state evolution of the system. This evolution is stored in the form of two vectors, one for the different states across time and the other for the time stamps thereof. Those vectors are assigned to the theta1, theta2 and theta3 variables which are then read by SimuLink. They are also used to plot the trajectories of the end-effector, after a forward kinematics computation on each element of this state vector is performed.

V. Results and Discussion

A. Circular Trajectory testing

The desired trajectory is a circle of radius 0.2m as shown in the figure 4

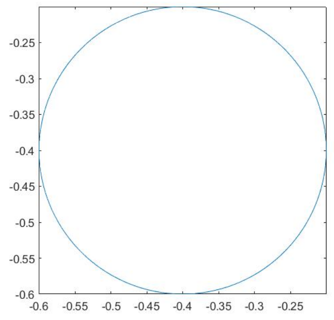

Fig. 4: Desired Circle trajectory

1) Test1: Comparison of controllers with perfect dynamics: In this test scenario, the controllers are tested with zero error in the estimated dynamics of the system. The K_p, K_d values are set to default 100 and 75 respectively. From the figures in 5 it's very clear that the endeffector is able to follow the desired trajectory perfectly with all the four controllers given the perfect dynamics of the system.

2) Test2: Comparison of controllers with imperfect dynamics: In this test scenario, the controllers are tested with error in the estimated dynamics of the system. For the Inverse Dynamics controller and the passivity based controller, a Gaussian noise of mean=0 and standard deviation= 0.00058 was added to one of the parameters of the inertia matrix. For the inverse dynamics controller, the K_p and K_d values are

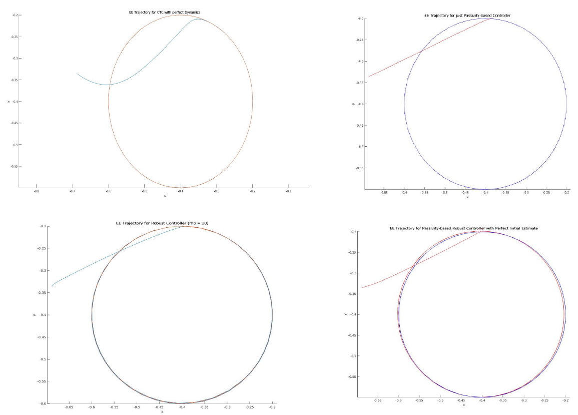

Fig. 5: Tracking Circular Trajectory with different controllers considering perfect

set to 500 and 75 respectively. But both of these controllers are not able to follow the desired trajectory as the estimated system dynamics are not perfect. The trajectory tracked by the end effector for these controllers are shown in the figure 6

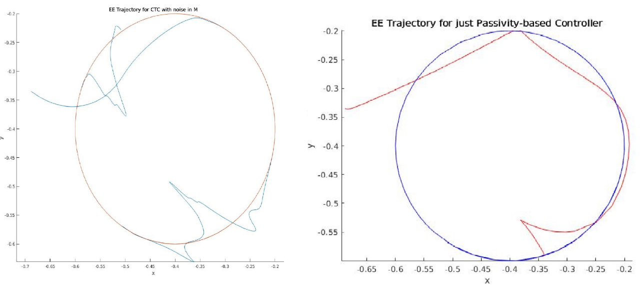

Fig. 6: Inverse Dynamics Controller and Passivity Based Controller with error in Dynamics

For the robust controller and the passivity based robust controller, the initial error in the dynamics of the system was set high. But the controllers were able to track the desired trajectory perfectly even though the initial errors in the estimate were set high, The values set for the robust controller are $K_p = 300, K_d = 250, \rho = 10$ And the values set for the passivity based robust controller are $K_p = 250, K_d = 90, \lambda = 1$. The figures in 7 shows the end-effector trajectories.

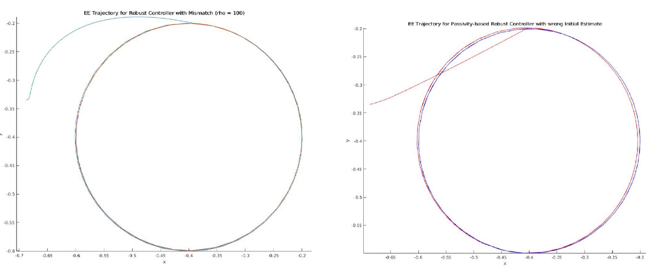

Fig. 7: Tracking circular trajectory with Robust feedback linearization and Passivity based Robust controller

B. Straight line Trajectory testing

A straight line of length $0.4m$ was set as desired trajectory for the system.

1) Test3: Comparison of controllers with perfect dynamics: In this test scenario, the controllers are tested with zero error in the estimated dynamics of the system. The K_p, K_d values are set to default $500, 75$ respectively. From the figures in 8 it's very clear that the endeffector is able to follow the desired trajectory perfectly with all the four controllers given the perfect dynamics of the system.

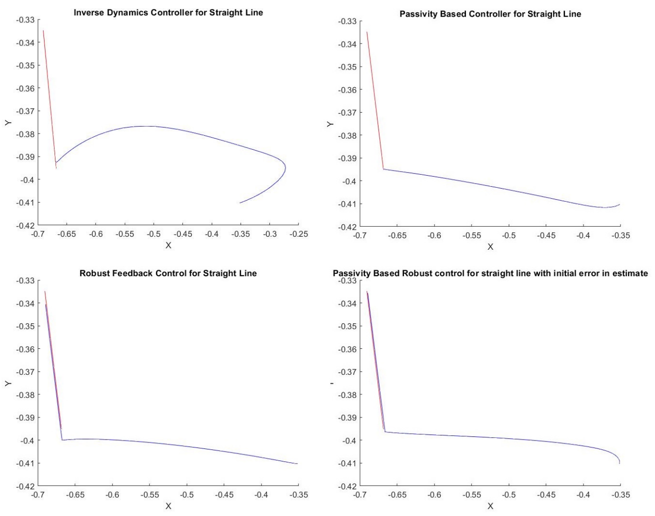

Fig. 8: Tracking Straight line trajectory with different controllers

2) Test2: Comparison of controllers with imperfect dynamics: In this test scenario, the controllers are tested with error in the estimated dynamics of the system. For the Inverse Dynamics controller and the passivity based controller, a Gaussian noise of mean $= 0$ and standard deviation$= 0.00058$ was added to one of the parameters of the inertia matrix. For the inverse dynamics controller, the K_p and K_d values are set to 500 and 75 respectively. But both of these controllers are not able to follow the desired trajectory as the estimated system dynamics are not perfect. The trajectory tracked by the end effectors for these controllers are shown in the figure 9

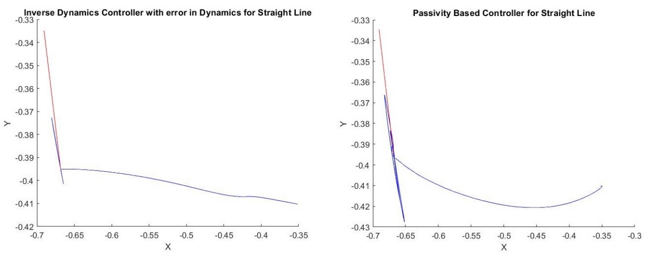

Fig. 9: Tracking Straight Line trajectory with Inverse dynamics and Passivity Based Controller considering error in the dynamic model

3) Test2: Comparison of controllers with imperfect dynamics: In this test for the robust controller and the passivity based robust controller, the initial error in the dynamics of the system was set high. But the controllers were able to track the desired trajectory perfectly even though the initial errors in the estimate were set high, The values set for the robust controller are $K_p = 100, K_d = 75, \rho = 10$. And the values set for the passivity based robust controller are $K_p = 250, K_d = 50\lambda = 0.5$. The figure 10 shows the end effector trajectories.

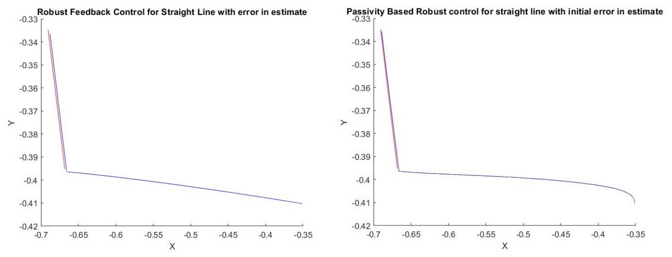

Fig. 10: Tracking Straight Line trajectory with Inverse dynamics and Passivity Based Controller considering error in the dynamic model

C. Discussion

We were able to succesfully implement inverse dynamics, passivity based, robust feedback linearization and passivity based robust control in the task space. We were able to see the difference between the controllers in different trajectories. We also considered uncertainties in the mass of each link and observed the difference. Passivity based robust control works best in case of uncertainties. We also found that K_p, K_d plays an important role in the inverse dynamics control and robust feedback control. The value of ρ also plays an important role in the passivity based controllers.

References

[1] P. M. Kebria, S. Al-wais, H. Abdi and S. Nahavandi, ”Kinematic and dynamic modelling of UR5 manipulator,” 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, 2016, pp. 004229-004234. doi: 10.1109/SMC.2016.7844896

[2] Spong, Mark W., Seth Hutchinson, and Mathukumalli Vidyasagar. Robot modeling and control. 2006.

[3] Khan, Muhammad Fayaz, Raza ul Islam, and Jamshed Iqbal. ”Control strategies for robotic manipulators.” 2012 International Conference of Robotics and Artificial Intelligence. IEEE, 2012.

[4] Gasparetto, A., et al. ”Experimental validation and comparative analysis of optimal time-jerk algorithms for trajectory planning.” Robotics and Computer-Integrated Manufacturing 28.2 (2012): 164-181.