UNIVERSITÀ
di VERONA

Revolute Prismatic Revolute Robotic Manipulator
Dynamics and Controls

Advance Control System
(A.Y. 2022/2023)
Department of Computer Science

Table of Contents

Abstract

This literature presents the modeling and control of a 3 degree of freedom (DoF) robot with a fixed base and two revolute joints and one prismatic joint. The robot is modeled using the Denavit-Hartenberg convention and its forward and inverse kinematics are derived using transformation matrices and symbolic equations. The differential kinematics of the robot are obtained using geometrical and analytical Jacobians and used to study the velocities of the end-effector. The dynamics of the robot are derived using both Lagrange and Newton-Euler formulations and compared for accuracy and efficiency. Different controllers for the robot are designed and tested in both joint space and operational space using MATLAB Simulink. The controllers include proportional-derivative (PD) control, inverse dynamics control, adaptive control, direct force control, and indirect force control. The performance of each controller is evaluated for various tasks such as tracking a desired trajectory or applying a desired force.

1. Introduction

Modern manufacturing relies heavily on industrial robots as critical automation tools, combining advanced technologies from various fields such as machinery, electronics, control, sensors, and artificial intelligence. Industrial robots such as electric welding, distribution, assembly, and handling have become commonplace in industrial production activities. The development of industrial robots is currently focused on improving positioning accuracy, a crucial technology for the practical application in advanced robotic manufacturing systems [1].

To make the robot work with the task efficiently and effectively, the dynamical characteristics of the robot should be determined. With the help of the dynamical characteristics, the analysis of the robot can be taken place from which the controls for the robot can be defined [2]. These controls help the robot to achieve the certain position and orientation to do some tasks assign to it. To determine the dynamics of the robots, the kinematics of the robot is needed to be calculated with the help of the transformation matrices and Denavit–Hartenberg parameters. Moreover, the relationship between the velocities and end-effectors needed to determine with the help of geometrical and analytical Jacobians [3]. It is important to note that the geometrical and analytical Jacobians are divided into two parts, the first part is the linear velocity part which maps the linear velocity of the end-effector. However the second part of the Jacobian matrix is rotational part which is different for both Jacobians matrix. In the geometrical Jacobian, the matrix referred to the angular velocity of the end-effector while in the analytical Jacobian it referred to the rotational velocity of the end-effector. The dynamics of the robot manipulator includes the gravitational, Coriolis, and Inertia matrix derived from the Newton-Euler approach or from Lagrange approach [4]. After getting the dynamics of the particular robot, it is possible to determine the dynamical properties of the robot and its working in the space.

To make the robot to perform some different tasks, the controller is needed to control the robot actions. Firstly, it is very important to define the working space for the robot. There are two types of spaces that can be used in the controller. The joint space uses the values of the joints to

determine its position and orientation of the end-effector. However, the operational space can be used instead of joint space. The operational space is more natural as it is defined as the cartesian coordinates from the base frame [5]. The controller helps the robot to reach some desired positions as well to achieve some desired trajectories. As defined, the robots also need to interact the environment, so force controller is needed to control the force of the robot. The force controller helps the robot to not damage itself or the environment.

The report is structured into distinct sections. In the second section, the literature's robot is elucidated. Sections 4, 5, and 6 provide definitions for the kinematics, differential kinematics, and dynamics of the RPR robotic manipulator. Subsequently, sections 8 through 10 detail the development and assessment of motion controllers for both operational and joint space. Sections 11 and 12 delve into the indirect and direct methods of force control. Finally, section 13 serves as a conclusion to the entire report.

2. Robot

The subject of the literature is a 3 degree of freedom (DoF) robot with a fixed base. Figure 1 in the literature presents a URDF diagram of the 3 DoF manipulator, where the green cylinder represents the fixed base that cannot be moved. The robot has 2 revolute joints that can rotate in a specific direction, i.e., along the z-axis, and 1 prismatic joint that can move along a particular direction, specifically the z-axis. The base frame is located in the bottom left corner of the figure, with the z-axis pointing upwards. However, with reference to the same base frame, the end-effector (ee) frame is defined, with the z-axis pointing out of the end-effector. All rest frame visible in the figure is not according to the Denavit–Hartenberg (DH) convention.

Table 1: Denavit-Hartenberg parameters table for RPR 3 DoF robot manipulator.

i	$\boldsymbol{\theta}$	$\boldsymbol{\alpha}$	r	d
1	π/2	π/2	0	a_1
2	θ_1	-π/2	a_2	0
3	0	0	0	$a_3 + d_1$
4	θ_2	0	a_4	0

Table 1 provides the Denavit-Hartenberg (DH) parameters for the RPR 3 DoF robot manipulator, as defined in the frame displayed in Figure 2. The frame associated with each joint follows the DH convention, whereby the rotation for the revolute joints only occurs around the z-axis, while the prismatic joint displacement occurs only along the z-axis. It is important to note that the base frame and end-effector frame are the same as those depicted in the URDF diagram of Figure 1.

Figure 2 in the literature provides both 2D and 3D variants of the same manipulator to enable a more comprehensive understanding of its features. As previously mentioned, the figure displays the frames associated with each joint in the robotic manipulator in accordance with the DH convention. However, the base frame and end-effector (ee) frame are the same as those of the original robot, enabling the kinematics to be mapped to the kinematics of the robotics toolbox.

The figure also includes the symbols for the lengths of each joint, as well as the mechanical symbols for each joint. Specifically, a_1, a_2, a_3 and a_4 represent the lengths from the base frame to the end-effector, while θ_1, $d_1$and θ_2 represent the mechanical parameters for each RPR joint, respectively. It is important to note that the base frame is the fixed joint and does not have any mechanical parameters.

3. Environmental Setup

In this literature, the MATLAB is used to compute the kinematics, differential kinematics, and dynamics. The MATLAB symbolic framework is used for creating the complex mathematical equations. MATLAB Robotic Toolkit is used to cross check the results of the manual computation with the toolkit provided results. MATLAB Simulink is used to design and test controllers with ease. These experiments are performed on I7 10750H CPU with 16 GB of RAM. The GTX 1660 TI GPU is used along with 6 GB of VRAM. Microsoft Windows 11 is used to run all the tools mentioned above.

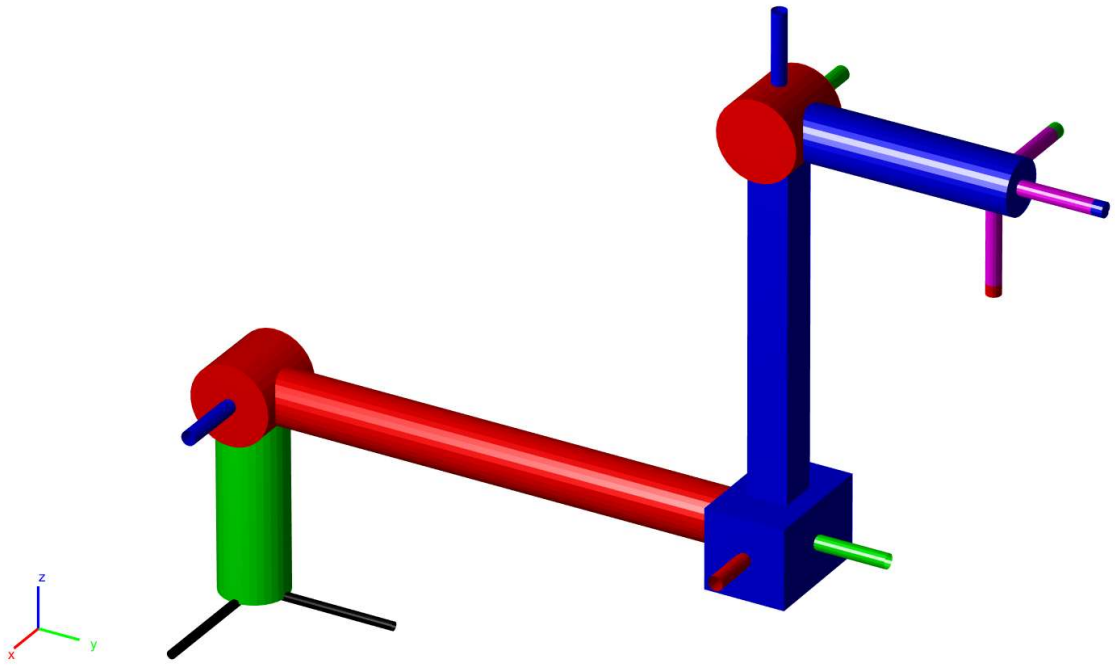


Figure 1: Revolute Prismatic Revolute (RPR) robotic arm manipulator URFD diagram

It should be noted that Figure 2 shows an additional rotation at the end of the manipulator, which is not included in the DH table. This extra rotation, denoted as R_1, is necessary to align the frame of the end-effector with the original URDF frame, thereby enabling the kinematics to be mapped to the MATLAB robot toolbox. However, adding this rotation via DH parameters is not feasible

due to the DH convention, which does not allow such explicit rotations. Consequently, the R_1 rotation has been added as an extra feature to the robot manipulator.

4. Kinematics

Robot manipulator kinematics refers to the study of the motion of robotic manipulators, including the relationships between the motion of the robot's joints and the resulting motion of the end-effector. The kinematics of a robotic manipulator can be represented by its forward kinematics and inverse kinematics [6], [7]. Forward kinematics describes the mapping between the joint angles of the robot and the position and orientation of the end-effector in the workspace. Inverse kinematics, on the other hand, describes the mapping between the desired position and orientation of the end-effector and the joint angles required to achieve that position and orientation. The Denavit-Hartenberg (DH) convention is used to model the kinematics of robotic manipulators. DH parameters define the location and orientation of each joint. By using the DH convention, the forward kinematics of a robotic manipulator can be calculated using transformation matrix. In order to perform kinematic analysis and simulation, software tools such as the MATLAB Robotics Toolbox is used to map the generated kinematics with the URDF robot.

4.1. Forward Kinematics

Forward kinematics is a method used in robotics to determine the position and orientation of the end-effector of a robot manipulator with respect to a given reference frame, based on the robot's joint angles and the geometric parameters of the robot. It involves finding the relationship between joint angles and the pose of the end-effector. The forward kinematics solution is a key step in controlling a robot's motion, as it provides a way to move the robot's end-effector to a desired location in its workspace. In this literature DH table is used to form the homogeneous transformation matrix to solve the forward kinematics problem.

Homogeneous transformation matrix is a 4x4 transformation matrix that is used to represent the position and orientation of an object or a robot manipulator in 3D space. In forward kinematics, the homogeneous matrix is used to calculate the position and orientation of the robot's end-effector based on the joint angles or displacements of the robot's joints. The homogeneous matrix consists of a rotation matrix and a translation matrix as shown in the Equation 1. The rotation matrix represents the orientation of the robot's end-effector relative to the robot's base frame, while the translation matrix represents the position of the end-effector relative to the base frame.

However, the Equation 2 elaborate the generalize homogeneous translation matrix according to the DH convention. It can be used to determine any translation and rotation from desired frame to target frame by multiplying it in the sequences. Moreover, the n is referred to the n^{th} joint of the robotic manipulator, in the DH table it referred to the i^{th} row of the Table 1.

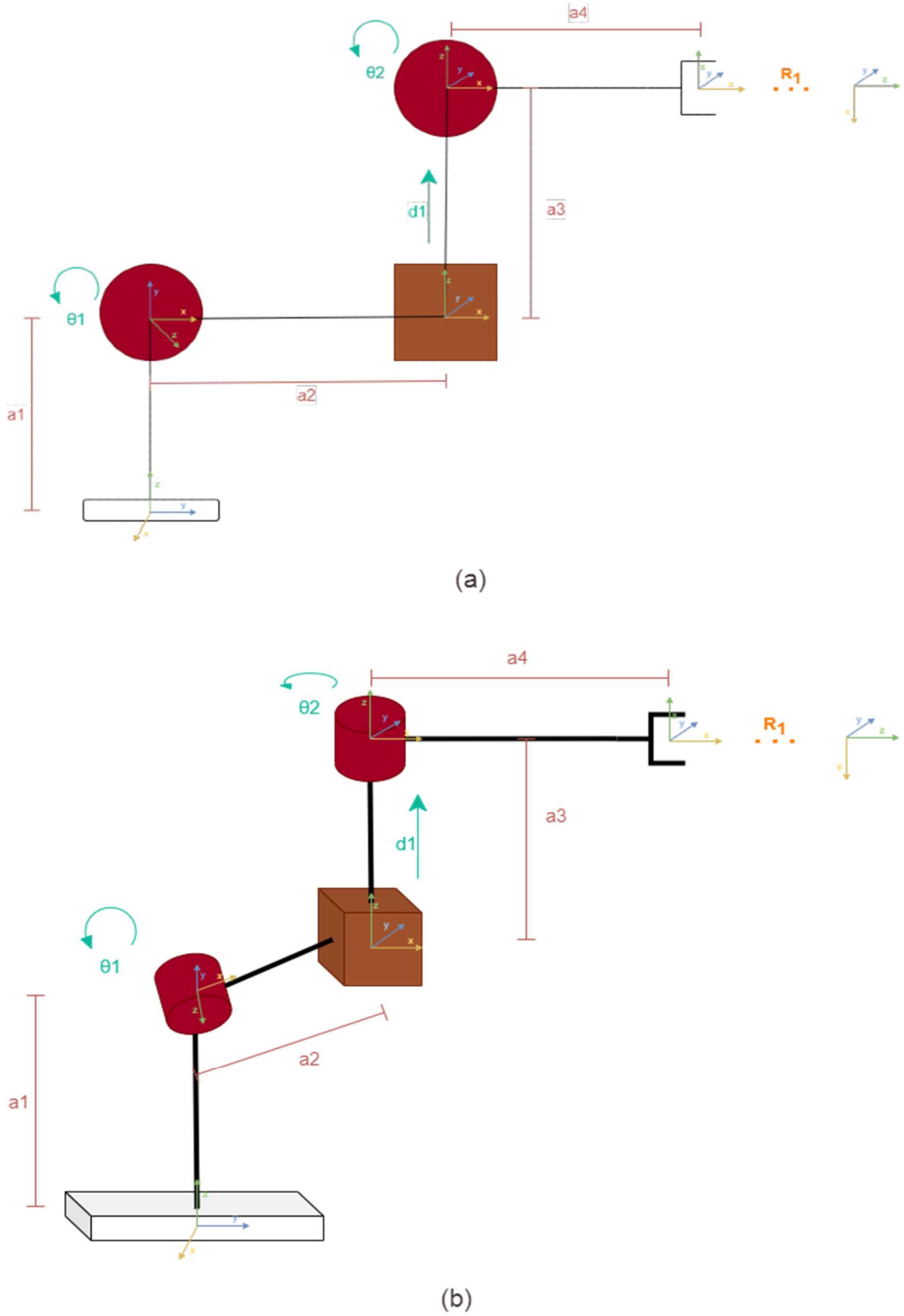


Figure 2: RPR robotic manipulator with 3 degrees of freedom. (a) 2D diagram of RPR manipulator. (b) 3D diagram of RPR manipulator.

$$H = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} & P_x \\ R_{yx} & R_{yy} & R_{yz} & P_y \\ R_{zx} & R_{zy} & R_{zz} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$H_n^{n-1} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & a_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & a_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

To determine the forward or direct kinematics of the robotic manipulator from base to end-effector following equation can be used.

$$H_4^0 = H_1^0 H_2^1 H_3^2 H_4^3 H_E^4 \quad (3)$$

where the H_1^0, H_2^1, H_3^2, H_4^3, and H_E^4 given as

$$H_1^0 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$H_2^1 = \begin{bmatrix} cos\theta_1 & 0 & -sin\theta_1 & a_2 cos\theta_1 \\ sin\theta_1 & 0 & cos\theta_1 & a_2 sin\theta_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$H_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & a_3 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$H_4^3 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & \mathrm{a}_4\cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & \mathrm{a}_4 sin\,\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

$$H_E^4 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

It is important to note that the homogeneous matrix H_E^4 in the Equation 3 is the empty homogeneous matrix with only rotation and empty translation given by Equation 8. Furthermore, following Equation demonstrate the homogeneous transformation from base to end-effector obtain by multiplying each homogeneous matrix to its successor. Where rotation part demonstrates the rotation from the base to the end-effector while the translation part demonstrate the translation from base to the end-effector.

$$H_4^0 = \begin{bmatrix} 0 & -cos\theta_2 & -sin\theta_2 & -a_4 sin\theta_2 \\ sin\theta_1 & -cos\theta_1 sin\theta_2 & cos\theta_1 cos\theta_2 & a_2 cos\theta_1 - sin\theta_1(a_3 + d_1) + a_4 cos\theta_1 cos\theta_2 \\ -cos\theta_1 & -sin\theta_1 sin\theta_2 & sin\theta_1 cos\theta_2 & a_1 + cos\theta_1(a_3 + d_1) + a_2 sin\theta_1 + a_4 sin\theta_1 cos\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

By substituting the values of θ_1, d_1, and θ_2 to Equation 9 the homogenous transformation matrix can be achieved for transformation from base to the end-effector. To check the correctness of the transformation matrix the results are compared with the MATLAB robot toolkit. Following are the results the result that obtained from substituting the values in the Equation 9 with $\theta_1 = 1.34\, rad$, $d_1 = -0.2$, and $\theta_2 = -0.66\, rad$.

$$H_4^0 = \begin{bmatrix} 0 & -0.7900 & 0.6131 & 0.0981 \\ 0.9735 & 0.1403 & 0.1807 & 0.0231 \\ -0.2288 & 0.5969 & 0.7690 & 0.6853 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

While the results obtained from the MATLAB robotic toolkit is presented below.

$$H_4^0 = \begin{bmatrix} -0.0001 & -0.7900 & 0.6131 & 0.0979 \\ 0.9736 & 0.1398 & 0.1803 & 0.0231 \\ -0.2281 & 0.5970 & 0.7691 & 0.6853 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As it can observed that the results are accurate with $Error \leq 0.0003$ which is acceptable in our scenario.

4.2. Inverse Kinematics

Inverse kinematics is the process of determining the joint angles or joint displacement of a robot manipulator to achieve a desired end-effector position and orientation. In other words, it is the reverse process of forward kinematics. Inverse kinematics is important because it allows us to specify the desired pose of the robot end-effector, and then solve for the joint angles or joint displacement that will achieve that pose. So, by using the P_x, P_y, and P_z from the Equation 9, we can analytically extract the θ_1, θ_2, and d_1.

$$P_x = -a_4 sin\theta_2 \quad (10)$$

$$P_y = a_2 cos\theta_1 - sin\theta_1(a_3 + d_1) + a_4 cos\theta_1 cos\theta_2 \quad (11)$$

$$P_z = a_1 + cos\theta_1(a_3 + d_1) + a_2 sin\theta_1 + a_4 sin\theta_1 cos\theta_2 \quad (12)$$

By using mathematical operations on Equation 10-12, we can analytically obtain following equations.

$$sin\theta_2 = -P_x / a_4 \quad (13)$$

$$cos\theta_2 = \sqrt{1 - (sin\theta_2)^2} \quad (14)$$

By using the Equation 13 and 14 it can get

$$\theta_2 = Atan2(sin\theta_2, cos\theta_2) \quad (15)$$

Similarly, the following equations can be used to determine the d_1

$$r = a_2 + a_4 cos\theta_2 \quad (16)$$

$$d_1 = \sqrt{P_y^2 + (P_z - a_1)^2 - r^2} - a_3 \quad (17)$$

Furthermore, by using the equations 10-12, 16, and 17, the θ_1 can be obtained as

$$cos\theta_1 = \frac{(P_z - a_1)(a_3 + d_1) + rP_y}{(a_3 + d_1)^2 + r^2} \quad (18)$$

$$sin\theta_1 = \frac{P_y a_3 d_1 + r a_1 - r P_z}{r^2 + (a_3 + d_1)} \quad (19)$$

$$\theta_1 = Atan2(sin\theta_1, cos\theta_1) \quad (20)$$

To check the correctness of the obtained equations, if the value of P_x, P_y, and P_z are same as the direct kinematic respective values then the values of the joint parameters are also same as the direct kinematics.

For P_x, P_y, and P_z

$$P_x = 0.0981, \qquad P_y = 0.0231, \qquad P_z = 0.6853$$

Substituting the values in Equation 15, 17, and 20 with respective joints lengths following joint parameters are obtained.

$$\theta_1 = 1.3400$$

$$d_1 = -0.2000$$

$$\theta_2 = -0.6600$$

As it can be seen that, the values are exactly same as the joint parameters for direct kinematics mentioned above.

5. Differential Kinematics

Differential kinematics plays a vital role in understanding the relationship between the joint velocities and the end-effector's linear and angular velocity. The Jacobian matrix, which is a matrix representation of this relationship, is dependent on the manipulator's configuration. This matrix is used to calculate the robot's end-effector velocity given the joint velocity inputs. There are two types of Jacobian matrices used in differential kinematics: analytical and geometrical Jacobians [8]. The analytical Jacobian can be obtained by the partial derivatives of the forward kinematics equations with respect to joint variables or by using the relationship with the geometrical Jacobian. Furthermore, the geometrical Jacobian can be obtained by the concatenation of the homogeneous matrices.

Both matrixes are similar in the sense of the translational velocities, but they are different as analytical Jacobian maps the end-effector rotational velocities while the geometrical Jacobians maps the angular velocities of end-effector.

5.1. Geometrical Jacobian

The geometrical Jacobian considers the orientation of the end-effector and the effect of the joint velocities on it. It uses the relationship between the joint velocity and the corresponding end-effector velocity in a coordinate frame fixed at the robot's base. The geometrical Jacobian matrix can be obtained by concatenating the translational and rotational Jacobian matrices. The geometrical Jacobian for the 3 DoF robot is 6x3 matrix which contains the translational and angular velocities of robotic manipulator. The generalize Jacobian matrix is given as

$$J(q) = \begin{bmatrix} \dot{P}_e \\ \omega_e \end{bmatrix} \tag{21}$$

Where q is the joint variable i.e. $\theta_1, \theta_2, \text{and } d_1$. Furthermore $\dot{P}_e$ is the translational/linear velocity and ω_e is the angular velocity. The generalize equation of $J(q)$ for 3 DoF robotic manipulator is given by.

$$J(q) = \begin{bmatrix} Jp_1 & Jp_2 & Jp_3 \\ Jo_1 & Jo_2 & Jo_3 \end{bmatrix} \tag{22}$$

Where Jp_1 is the linear velocity according to the first joint and Jo_1 is the angular velocity according to the first joint of the robotic manipulator. However to calculate these velocities following generalize formula is used.

$$\begin{bmatrix} Jp_i \\ Jo_i \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & \text{for prismatic joint} \\ \begin{bmatrix} z_{i-1} \times (P_e - P_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{for revolute joint} \end{cases} \tag{23}$$

Where P_e is the positional vector of the end-effector and z_{i-1} and P_{i-1} are the 3rd column and positional vector of H_{i-1}^{0} matrix. By using Equation 22 and 23, the geometrical Jacobian of RPR manipulator yields following matrix.

$$J(q) = \begin{bmatrix} z_1 \times (P_4 - P_1) & z_2 & z_3 \times (P_4 - P_3) \\ z_1 & 0 & z_3 \end{bmatrix} \quad (24)$$

By substituting the values from the direct kinematics Equations 3-8, the matrix from Equation 22 becomes

$$J(q) = \begin{bmatrix} 0 & 0 & -a_4 cos\theta_2 \\ -cos\theta_1(a_3 + d_1) - a_2 \sin\theta_1 - a_4 cos\theta_2 sin\theta_1 & -sin\theta_1 & -a_4 cos\theta_1 sin\theta_2 \\ a_2 cos\theta_1 - sin\theta_1(a_3 + d_1) - a_4 cos\theta_2 cos\theta_1 & \cos\theta_1 & -a_4 sin\theta_1 sin\theta_2 \\ 1 & 0 & 0 \\ 0 & 0 & -sin\theta_1 \\ 0 & 0 & cos\theta_1 \end{bmatrix} \quad (25)$$

To map the matrix from Equation 25 to the geometrical Jacobian of MATLAB robotics toolkit, the values of joint variables along with the joint lengths are substituted. Following numerical geometrical Jacobian matrix is obtained for the RPR given robotic manipulator.

$$J(q) = \begin{bmatrix} 0 & 0 & -0.1264 \\ -0.5353 & -0.9735 & 0.0224 \\ 0.0231 & 0.2288 & 0.0955 \\ 1 & 0 & 0 \\ 0 & 0 & -0.9735 \\ 0 & 0 & 0.2288 \end{bmatrix}$$

While the results obtained from the MATLAB robotic toolkit is presented below.

$$J(q) = \begin{bmatrix} 0 & 0 & -0.1264 \\ -0.5353 & -0.9735 & 0.0224 \\ 0.0231 & 0.2288 & 0.0955 \\ 1 & 0 & 0.0006 \\ 0 & 0 & -0.9735 \\ 0.0008 & 0 & 0.2288 \end{bmatrix}$$

As it can observed that the results are accurate with $Error \leq 0.0003$ which is acceptable in our scenario.

5.2. Analytical Jacobian

As stated earlier, the analytical Jacobian can be acquired through the partial derivatives of the forward kinematics equations with respect to the joint variables or by utilizing the connection with the geometrical Jacobian. In simpler terms, the analytical Jacobian can be derived by calculating the rate of change of end-effector position and orientation with respect to the joint velocities. Alternatively, the analytical Jacobian can be calculated by manipulating the geometrical Jacobian. However, in this literature we use the $J(q)$ relationship to compute the analytical Jacobian matrix.

$$J(q) = Ta(\phi)Ja(q) \quad (26)$$

Where $Ta(\phi)$ is the transformation matrix which depends on orientation of end-effector and $Ja(q)$ is the analytical Jacobian. The $Ta(\phi)$ according to ZYZ Euler angles can be defined as

$$Ta(\phi) = \begin{bmatrix} I & O \\ O & T \end{bmatrix} \quad (27)$$

However, the I is the 3x3 identity matrix and O is the 3x3 empty matrix. The matrix T is given by.

$$T = \begin{bmatrix} 0 & -sin\phi & cos\phi \sin\theta \\ 0 & cos\phi & sin\phi \sin\theta \\ 1 & 0 & cos\theta \end{bmatrix} \quad (28)$$

So, to calculate the analytical Jacobian, the Equation 26 can be used as follow.

$$Ja(q) = \left(Ta(\phi)\right)^{-1} J(q) \quad (29)$$

Additionally, it should be noted that the computation of the analytical Jacobian cannot be directly mapped to the MATLAB robotics toolbox as the toolbox lacks the capability to calculate the analytical Jacobian. However, since the linear velocity component of the analytical Jacobian is identical to the geometrical Jacobian, a partial correction for the analytical Jacobian can be determined using this relationship. By substituting the values to Equation 29, yields

$$Ja(q) = \begin{bmatrix} -\frac{2sin\theta_1}{5} - cos\theta_1\left(d_1 + \frac{3}{10}\right) - \frac{4cos\theta_2 sin\theta_1}{25} & -\sin\theta_1 & -\frac{4cos\theta_1 sin\theta_2}{25} \\ \frac{2cos\theta_1}{5} + \frac{4cos\theta_1 cos\theta_2}{25} - sin\theta_1(d_1 + \frac{3}{10}) & cos\theta_1 & -\frac{4sin\theta_1 sin\theta_2}{25} \\ 0 & 0 & -\frac{4cos\theta_2}{25} \\ 1 & 0 & -\frac{\sin(\phi - \theta_1)\, cos\theta}{sin\theta} \\ 0 & 0 & \cos(\phi - \theta_1) \\ 0 & 0 & \sin(\phi - \theta_1)/sin\theta \end{bmatrix} \quad (30)$$

To get the numerical values from above equation, the values are substituted in the Equation 30. The Euler angles ϕ and θ is obtained from the rotation matrix R_4^0 via direct kinematics. The above matrix became.

$$Ja(q) = \begin{bmatrix} 0 & 0 & -0.1264 \\ -0.5353 & -0.9735 & 0.0224 \\ 0.0231 & 0.2288 & 0.0955 \\ 1 & 0 & 0.5599 \\ 0 & 0 & 0.9338 \\ 0 & 0 & -0.4306 \end{bmatrix}$$

As it can be observed that the $\dot{P}_e$ is exactly same as the geometrical Jacobian linear velocity.

6. Dynamics

The dynamics of a robotic manipulator is the study of the forces and torques acting on the manipulator while it is in motion. It is essential for controlling and simulating the behavior of the robot in various tasks [9]. The dynamics of the manipulator can be divided into two categories, namely, the forward dynamics and the inverse dynamics. The forward dynamics problem involves determining the motion of the manipulator given the forces and torques applied to it. It involves solving the equations of motion that describe the behavior of the manipulator in terms of its position, velocity, and acceleration. On the other hand, the inverse dynamics problem involves determining the forces and torques required to achieve a desired motion of the manipulator i.e. velocity, acceleration, and its position. The dynamics of a robotic manipulator can be modeled using the Lagrange-Euler equations, which describe the relationship between the forces and torques applied to the manipulator and its motion by using kinetic energy and potential energy. However, the dynamics of robotics manipulator can also be determined by Newton-Euler formulation, in which the Newton law of motion and Euler law of rotation is used.

In this literature, the dynamics of a 3 DoF RPR robotic manipulator is evaluated using both the Lagrange formulation and the Newton-Euler formulation. To ensure the accuracy of the dynamic model, the results obtained from both formulations are compared. It is also particularly important to note that, the dynamics of the manipulator is calculated from the first joint referred as 'Frame 1' not from the base frame referred as 'Frame 0', as base frame is fixed and immovable, the dynamics of the base frame can be ignored.

6.1. Lagrange Formulation

To obtain the equations of motion for a system, the Lagrange formulation involves determining the kinetic and potential energies of the manipulator. This process includes determining the inertia matrix, Coriolis and centrifugal forces, and gravitational forces of the robotic manipulator. The Lagrange Formulation stated that.

$$L = T - U \qquad (31)$$

Where *T* and *U* are the kinetic and potential energy of the 3 DoF RPR robotic manipulator, respectively. To calculate the desired energies following methods are used.

6.1.1. Kinetic Energy

The kinetic energy is calculated by summing up the kinetic energy of each link in the system. This includes both the translational and rotational kinetic energies of each link. The translational kinetic energy is proportional to the mass of the link and its linear velocity, while the rotational kinetic energy is proportional to the moment of inertia of the link and its angular velocity. As stated before, the total kinetic energy stated as

$$T = \sum_{i=1}^{3} T_i \qquad (32)$$

Where T_i is the i^{th} link of the manipulator and T denotes the total kinetic energy of 3 DoF RPR robotic manipulator. Furthermore, the total kinematic energy also expressed as the inertia matrix as follows.

$$T(q, \dot{q}) = \dot{q}^T B(q)\, \dot{q} \qquad (33)$$

With $B(q)$ is the inertia matrix, expresses the inertia in x, y, and z direction for each link for the robotic manipulator. It is dependent on the parameters of the joint i.e. q. The q is the generalized parameter of the joint while in RPR robotic manipulator it is referred to θ_1, d_1, and θ_2. To calculate the $B(q)$ matrix following formulation can be used.

$$B(q) = \sum_{i=1}^{3} ((m_i {J_p^i}^T J_p^i) + ({J_o^i}^T R_i\, I^i R_i^T J_o^i)) \qquad (34)$$

Where m_i and R_i is the mass and rotation matrix of the i^{th} link of the manipulator with respect to frame 1. Furthermore, J_P^i and J_o^i are the partial Jacobian up to i^{th} joint targeted to the center of mass of the joint. The partial Jacobian J^i can be calculated with the following formula.

$$J^i = \begin{bmatrix} J_p^i \\ J_o^i \end{bmatrix} \qquad (35)$$

To elaborate more, the partial Jacobian for the 2nd joint will be.

$$J^2 = \begin{bmatrix} J_{p_1}^2 & J_{p_2}^2 & 0 \\ J_{o_1}^2 & J_{o_2}^2 & 0 \end{bmatrix} \qquad (36)$$

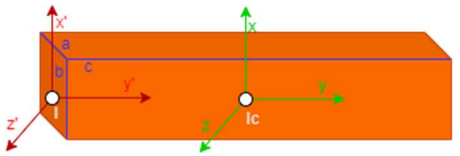
$$\begin{bmatrix} J_{p_j}^i \\ J_{o_j}^i \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{j-1} \\ 0 \end{bmatrix} & \text{for prismatic joint} \\ \begin{bmatrix} z_{j-1} \times (P_{l_i} - P_{j-1}) \\ z_{j-1} \end{bmatrix} & \text{for revolute joint} \end{cases} \qquad (37)$$

$$P_{l_i} = \mathrm{R_i^1(CoM)^T} + P_i \qquad (38)$$

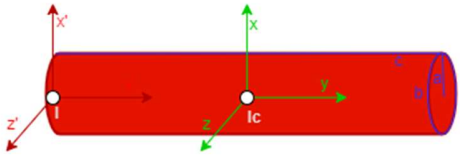
Where $R_i^1$is the rotation matrix up to i^{th} frame with respect to frame 1 and $P_i$is the position of the i^{th} frame. Central of Mass (CoM) matrix is given according to RPR robot expressed in Figure 2.

$$CoM = \begin{bmatrix} -\frac{a_2}{2} & 0 & 0 \\ 0 & 0 & -\frac{a_3}{2} \\ 0 & 0 & -\frac{a_4}{2} \end{bmatrix} \quad (39)$$

Back to Equation 34, the I^i is the i^{th} joint inertia vector, the inertia vector is express with respect of the start of the joint not with the CoM by using the parallel axis theorem (Steiner theorem). The I vector is of following form for revolute or for prismatic.



$$I = I_C + m\left(\begin{bmatrix} -\frac{c}{2} \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} -\frac{c}{2} \\ 0 \\ 0 \end{bmatrix} I_{3\times3} \begin{bmatrix} -\frac{c}{2} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} -\frac{c}{2} \\ 0 \\ 0 \end{bmatrix}^T\right) \quad (40)$$



$$I = I_C + m\left(\begin{bmatrix} -\frac{c}{2} \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} -\frac{c}{2} \\ 0 \\ 0 \end{bmatrix} I_{3\times3} \begin{bmatrix} -\frac{c}{2} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} -\frac{c}{2} \\ 0 \\ 0 \end{bmatrix}^T\right) \quad (41)$$

6.1.2. Potential Energy

Potential energy is typically related to the position of the manipulator and its end-effector relative to the gravity. The potential energy of the manipulator is directly proportional to its height above a reference point. The higher the manipulator are raised, the greater the potential energy is. The potential energy is easy to calculate and can be stated as

$$U = \sum_{i=1}^{3} m_i {g_1}^T P_{l_i} \quad (42)$$

Where the P_{l_i} is the position of the CoM as already stated in Equation 38, while the g_1 is the gravity vector expressed in the frame 1, for the specified robot the vector is of form

$$g_1 = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} \quad (43)$$

Where g is the gravity constant of which value is 9.806.

Back to the Lagrange formulation, after finding the $B(q)$ the inertia matrix, it is easy to use both the potential and kinetic energy to form the following equation of motion for the RPR robotic manipulator.

$$B(q)\ddot{q} + C(q,\ddot{q})\dot{q} + F_V\dot{q} + F_s sign(\dot{q}) + g(q) = \tau - J^T h_e \quad (44)$$

According to the Equation 44, the h_e is the external wrench of the end-effector interaction with the environment. The F_v is the viscous friction of the joints and F_s is the coulomb friction. However, the $C(q,\ddot{q})$ is Coriolis matrix obtained by the derivation of $B(q)$ elements with each respective joint, the following formulation can be form to determine the Coriolis matrix

$$C(q,\ddot{q}) = C_{ij} = \sum_{i=j}^{3} C_{ij}(q)\dot{q}_J = \sum_{i=j}^{3}\sum_{i=k}^{3} \frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right)\dot{q}_k \quad (45)$$

and the $g(q)$ can be achieved by the derivation of the potential energy with each joint variable or by gravity formulation. The following equation shows the gravity matrix obtained from the partial derivative method

$$g(q) = \begin{bmatrix} \frac{\partial U}{\partial q_1} \\ \frac{\partial U}{\partial q_2} \\ \frac{\partial U}{\partial q_3} \end{bmatrix} \quad (46)$$

the potential energy using the gravity formulation can be achieved by

$$g_j(q) = -\sum_{i=1}^{3} m_i g_1^T J_{P_j}^i(q) \quad (47)$$

It is important to note that, to check the correctness of Equation 44 the MATLAB robotics toolkit do not provide any built in functionality. However, to partially check the correctness of equation we can take few steps. Firstly, according to the theory the $B(q)$ matrix will be symmetric and positive definite. By computing the $B(q)$ matrix numerically we can ensure the correctness of $B(q)$ matrix.

$$B(q) = \begin{bmatrix} 0.3343 & 0.2356 & -0.0008 \\ 0.2356 & 0.5616 & 0 \\ -0.0008 & 0 & 0.0036 \end{bmatrix}$$

and eigen vector formed from the $B(q)$ matrix is

$$eigen\big(B(q)\big) = \begin{bmatrix} 0.0036 \\ 0.1864 \\ 0.7095 \end{bmatrix}$$

with the help of above results, it can be seen that the $B(q)$ inertia matrix is positive definite and symmetric matrix. Furthermore, we can also compare the Equations 46-47 to check the correctness of the gravity matrix. The results are displayed below

$$g(q) = \begin{bmatrix} 1.1579 \\ 1.2598 \\ 0.0809 \end{bmatrix} \quad \text{(Differential Method)}$$

$$g(q) = \begin{bmatrix} 1.1579 \\ 1.2598 \\ 0.0809 \end{bmatrix} \quad \text{(Gravity Formulation)}$$

according to above results the gravity vectors are both exactly same.

6.2. Newton–Euler Formulation

The Newton-Euler formulation is a method for computing the dynamics of a robotic manipulator by recursively calculating the forces and torques at each joint. It involves deriving the equations of motion for the system by calculating the linear and angular velocities and accelerations of the manipulator, as well as the forces and torques acting on it. The process starts from the desired frame of the manipulator and moves towards the end-effector, using the laws of Newton and Euler to compute the forces and torques at each joint. The Newton-Euler formation can be computed using recursive algorithm. However, the results are not very interpretable. The following pseudocode can be used to determine Newton-Euler formulation.

Newton Euler Formulation: Forward Equation

$$\omega_0 \leftarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \ddot{P} \leftarrow \begin{bmatrix} 0 \\ g \\ 0 \end{bmatrix}, z_0 \leftarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \text{and } \dot{\omega}_0 \leftarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

for i = 1 to n:

$R_i^{i-1} = R_i^{i-1}(q_i)$

$\omega_i^i = \left(R_i^{i-1}\right)^T \omega_{i-1}^{i-1}$

$\dot{\omega}_i^i = \left(R_i^{i-1}\right)^T \dot{\omega}_{i-1}^{i-1}$

if joint == REVOLUTE:

$\omega_i^i = \omega_i^i + \left(R_i^{i-1}\right)^T \dot{q}_i z_0$

$\dot{\omega}_i^i = \dot{\omega}_i^i + \left(R_i^{i-1}\right)^T (\ddot{q}_i z_0 + \dot{q}_i \omega_{i-1}^{i-1} \times z_0)$

$\ddot{P}_i = \left(R_i^{i-1}\right)^T \ddot{P}_{i-1}^{i-1} + \dot{\omega}_i^i \times r_{i-1,i}^i + \omega_i^i \times (\omega_i^i \times r_{i-1,i}^i)$

if joint == PRISMATIC:

$\ddot{P}_i = \ddot{P}_i + \left(R_i^{i-1}\right)^T \ddot{q}_i z_0 + 2\dot{q}_i \omega_i^i \times \left(\left(R_i^{i-1}\right)^T z_0\right)$

$\ddot{P}_{c_i}^i = \ddot{P}_i^i + \dot{\omega}_i^i \times r_{i,C_i}^i + \omega_i^i \times (\omega_i^i \times r_{i,C_i}^i)$

where the $\ddot{P}$ depends on the axis of the gravity from the first base frame. $r^i_{i-1,i}$ is the position of the frame i with respect of frame $i-1$, it value include $(R_i^{i-1})^T P_i$, where P_i is the position of the i^{th} link. Similarly, r^i_{i,C_i} is the position of the frame i^{th} central of mass which is same as Equation 39.

To get the torques from the forward equation, similar backward equation is needed to process the output from the forward equation to the respected torques.

Newton Euler Formulation: Backward Equation

1 for i = 1 to n:
2 $R^i_{i+1} = R^i_{i+1}(q_i)$
3 $f^i_i = R^i_{i+1} f^{i+1}_{i+1} + m_i \ddot{P}^i_{C_i}$
4 $\mu^i_i = -f^i_i \times \left(r^i_{i-1,i} + r^i_{i,C_i}\right) + R^i_{i+1}\mu^{i+1}_{i+1} + (R^i_{i+1} f^{i+1}_{i+1}) \times r^i_{i,C_i} + I^i_i \dot{\omega}^i_i + \omega^i_i \times (I^i_i \omega^i_i)$
5 if joint == PRISMATIC:
7 $\tau_i = \left(f^i_i\right)^T \left(R^{i-1}_i\right)^T z_0$
8 if joint == REVOLUTE:
9 $\tau_i = \left(\mu^i_i\right)^T \left(R^{i-1}_i\right)^T z_0$
10 $\tau_i = \tau_i + F_{V_i}\dot{q}_i + F_{S_i} sign(\dot{q}_i)$

where I_i is same as I in Equation 40 and 41. Furthermore, the f^i_i and μ^i_i are the linear and angular external wrench, initially it is equal to he i.e. $he = \begin{bmatrix} f_{n+1} \\ \mu_{n+1} \end{bmatrix}$.

After computing the Newton-Euler formulation and Lagrange formulation, we can now confidently check the correctness of our dynamic model. Following are the numeric results for the τ from both models.

$$\tau = \begin{bmatrix} 5.0834 \\ 6.3608 \\ 2.1520 \end{bmatrix} \quad \text{(Lagrange formulation)}$$

$$\tau = \begin{bmatrix} 5.0834 \\ 6.3608 \\ 2.1520 \end{bmatrix} \quad \text{(Newton} - \text{Euler formulation)}$$

According to above results, the dynamics of both methods provide exactly same results. It is important to note that, the results are friction less and without environment wrench.

7. Dynamic Model in Operational Space

The operational space of a robotic manipulator is a simplified representation of the end-effector's position and orientation. The position corresponds to the end-effector's real-world coordinates,

while orientation is represented using Euler angles. The operational space is not dependent on the joint variables of the manipulator. The operational space can be represented using the minimalistic notation x_e.

$$x_e = \begin{bmatrix} P_e \\ \phi_e \end{bmatrix} \quad (48)$$

where the P_e is the position of the end-effector and ϕ_e is the Euler angles, referred as the orientation of the end-effector. The x_e is the 6×1 vector, and it is worth noting that x_e is very interpretable and convenient then the joint space q especially when interacting with the environment.

However, to convert the joint space dynamic model to the operational space dynamic model, we can take the help of analytical Jacobian. The dynamic model in operational space can became

$$Ba(x)\ddot{x} + Ca(x,\dot{x})\dot{x} + ga(x) = u - u_e \quad (49)$$

where

$$Ba(x) = (Ja^{-1})^T B Ja^{-1} \quad (50)$$

$$Ca(x,\dot{x})\dot{x} = (Ja^{-1})^T C\dot{q} - Ba\dot{Ja}\dot{q} \quad (51)$$

$$ga(x) = (Ja^{-1})^T g \quad (52)$$

$$u = (Ta)^T h \quad (53)$$

$$ga(x) = (Ta)^T h_e \quad (54)$$

8. Joint Space Controllers

Joint space controllers are a type of control technique that enables the control of robotic manipulators by controlling the position and velocity of the joint variables [10]. The joint space controllers use feedback of joint position and velocity to adjust the torque applied to the joints. They can be implemented using various control techniques such as PD, Inverse Dynamic, and adaptive control. One of the advantages of joint space controllers is that they are computationally efficient and very easy to implement. However, they do not provide direct control over the end-effector position and orientation, which can be important when interacting to the environment.

8.1. PD controller with gravity compensation

A proportional-derivative (PD) control is the type of control which uses gains to control the position of the joints of each link. The PD controller calculates the error between the desired and actual positions of the joints, and then applies a proportional and derivative gain to this error.

With the help of this, new torque is generated to move the joint toward the desired goal position. However, in the presence of gravity, the weight of the robot's links can affect the torques generated by the PD controller, which leads to the error in the position control. To compensate for this, a gravity compensation term is added to the controller. The gravity compensation term accounts for the gravitational forces acting on the robot's links and ensures that the control signal accurately reflects the desired position of the joints. For the PD controller following control law is defined

$$\tau = g(q) + K_p(q_d - q) - K_d\ddot{q} \qquad (55)$$

Figure 3 shows the PD controller schema with gravity compensation. It is important to note that there are different scenarios that are controlled by manual switch. These scenarios include the neglecting gravity, adding fixed gravity matrix, and adding noise to the desired q.

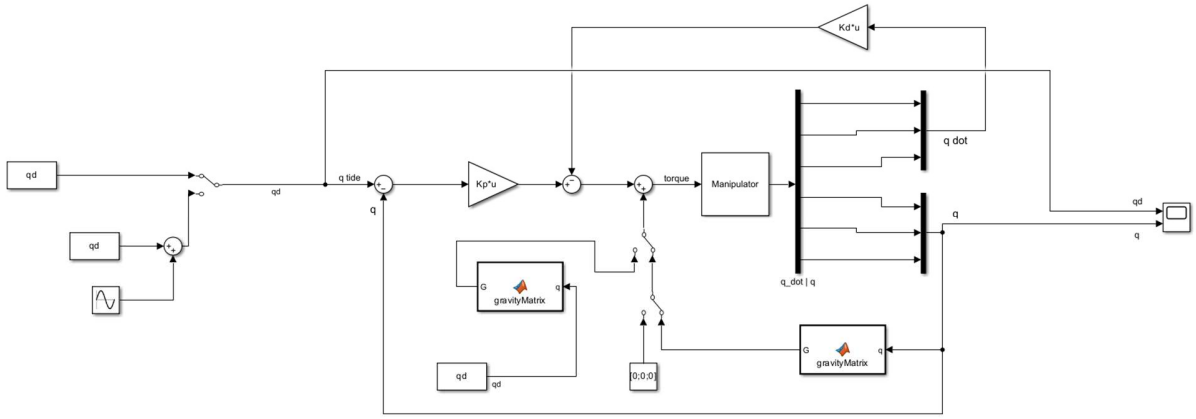


Figure 3 Simulink PD controller with gravity compensation schema

The gravity compensation matrix is the same matrix that is defined in the equation of motion which is the function of joint variable q. Manipulator in this literature is implemented using the MATLAB s-function which provides the q and $\dot{q}$ for each time step. Furthermore, the scope is used to plot the graphs between the desired and current joint variables. Following parameters are set on the controller during initializing.

$$q_d = [2.1, -0.2, 1]$$

$$K_p = \begin{bmatrix} 150 & 0 & 0 \\ 0 & 150 & 0 \\ 0 & 0 & 150 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}$$

Figure 4 shows the graph between q and q_d. It can clearly be seen that the q after some time converges to the desired joint variable q_d. It is worth noting that, for some set of K_p and K_d there is an optimal solution existed that converges faster than the shown image.

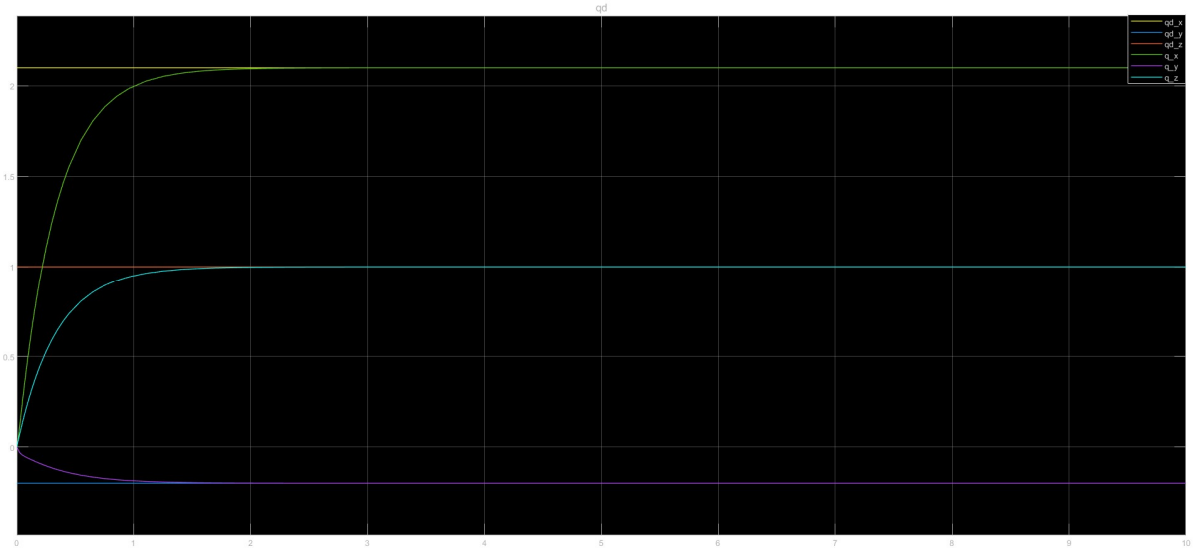


Figure 4 Joint variables with actual joint variables with gravity compensation.

The Figure 5 shows the graph of the q and q_d in which the gravity compensation is neglected, as it can be seen that for q_1 and q_2 there is a slight error, and they are not overlapping with q_{d1} and q_{d2} as the Figure 4. The reason behind it is the controller no longer consider the gravity of the manipulator and do not converge to q_d fully. According to motion of equation, the robot need compensation to gravity even if the robot is not moving.

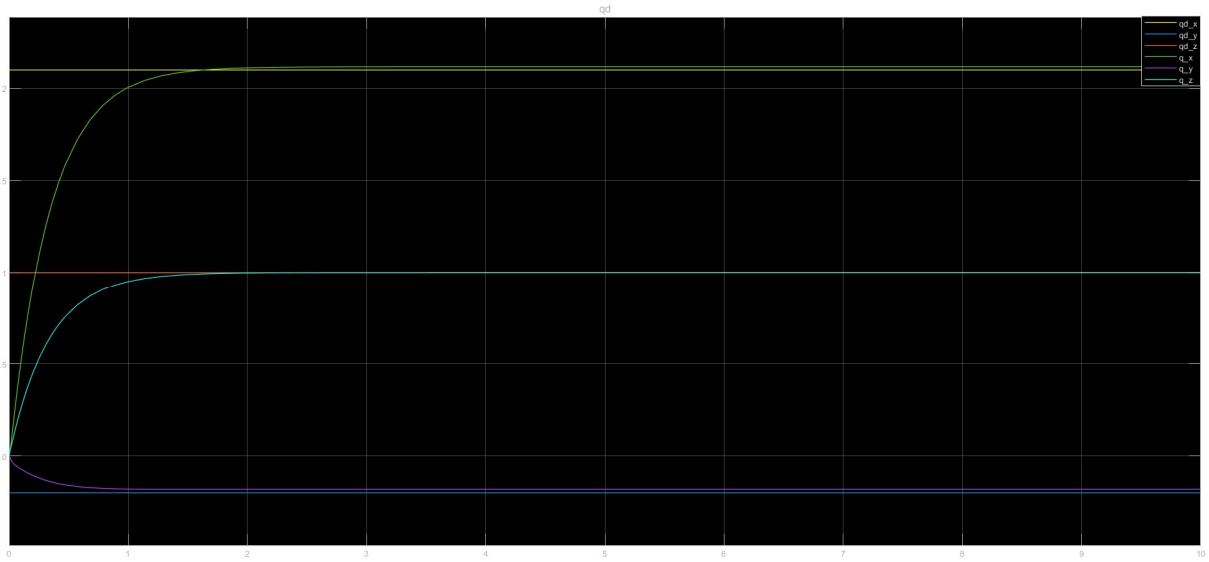


Figure 5 Joint variables with actual joint variables without gravity compensation.

The one of the critical disadvantages of using the PD controller with gravity compensation is prune to noise. The noise effect the q_d which makes the robot joints little moveable, according to the equation of motion we must compensate the inertia and Coriolis forces as well if the robot needs movement. For this reason, the controller did not accurately reach to the desired joint variables as shown in the Figure 6.

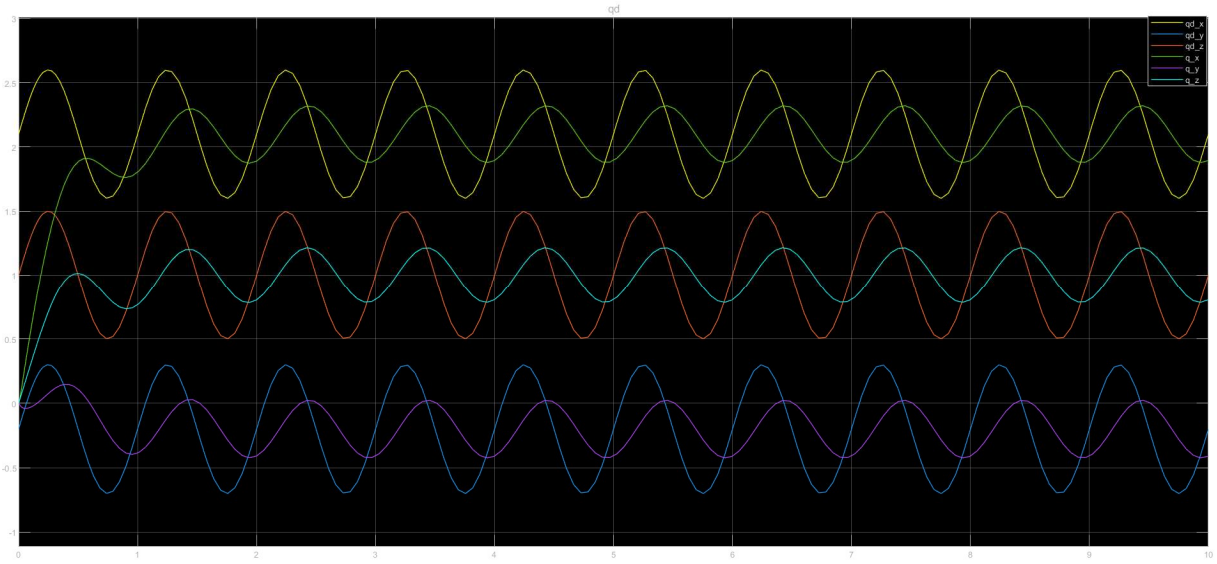


Figure 6 Joint variables with actual joint variables with gravity compensation and noise.

8.2. Inverse dynamics controller

As mentioned earlier, the drawback of PD control with gravity compensation is that the controller's performance suffers in the presence of noise in the desired joint variable q_d, as illustrated in Figure 6. To address this issue, the inverse dynamic controller can be an effective solution, as it performs well even in noisy conditions. This controller has been shown to be highly effective in tracking trajectories both in the joint space and in the operational space. The inverse dynamic controller uses elements from the equations of motion, such as inertia, Coriolis, and gravity terms, to compensate for all noise. It takes into account the dynamic nature of the joint variables, but not the constant term. It is important to note that the manipulator's dynamic model must be accurate for this controller to work effectively. Therefore, accurate modeling of the system is crucial to ensure the success of this type of controller. The control law for inverse dynamics stated that

$$\tau = B(q)y + n(q, \ddot{q}) \tag{56}$$

Back to Equation 44, if the friction forces are neglected the equation became

$$\tau = B(q)\ddot{q} + \underbrace{C(q,\dot{q})\dot{q} + g(q)}_{n(q,\ddot{q})} \tag{57}$$

Moreover the y known as the stabilizing linear control, which main purpose it to add the stability to the controller. The y includes

$$y = \ddot{q} + K_d(\dot{q}_d - \dot{q}) + K_p(q_d - q) \quad (58)$$

The control parameters that are used in this controller are

$$K_p = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 200 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 30 & 0 & 0 \\ 0 & 30 & 0 \\ 0 & 0 & 30 \end{bmatrix}$$

The Figure 7, shows the control schema for the inverse dynamics control. It is important to note that there are manual switches in the schema to switch between different scenarios, e.g. constant q_d and q_d trajectory. Furthermore, this is the compact form of the controller. The controller is made up with 2 different blocks that is discussed below.

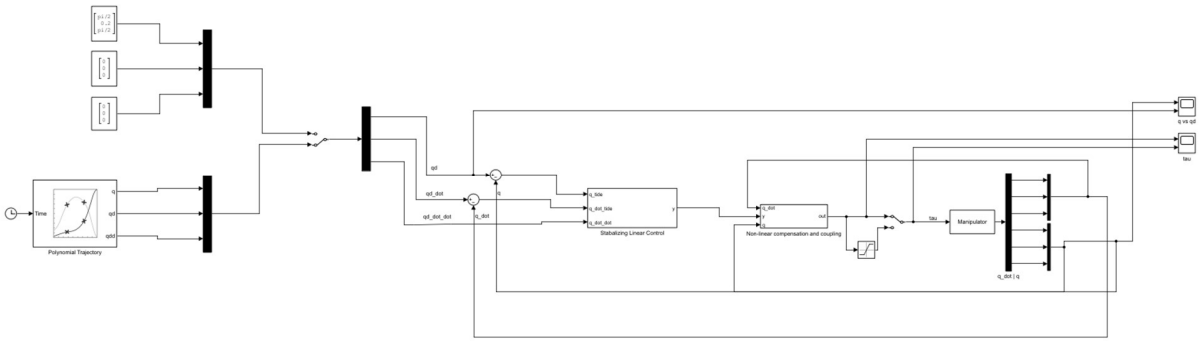


Figure 7 Inverse dynamic controller Simulink schema.

The stabilizing linear control assure that the controller is stabilize and coverage to the goal. According to Figure 8, the stabilizing linear control schema is proposed. The K_d and K_p are the gains which are sum up to the $\ddot{q_d}$.

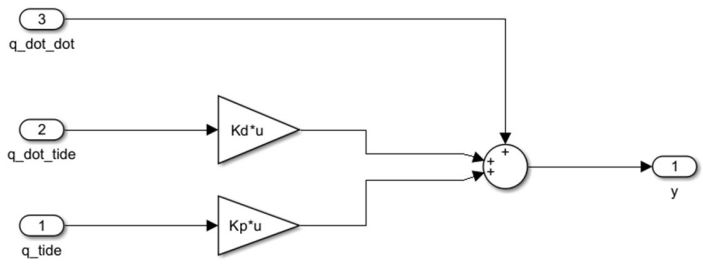


Figure 8 Stabilizing linear control.

It is very important to compensate the gravity, Coriolis, and Inertia when the robot is actually moving in the desired trajectory as stated earlier. Figure 9 shows the Non-linear compensation of $n(q, \dot{q})$ and $B(q)$. It is important to note that the schema contain different scenarios for testing the controller. These scenarios include the neglection of Inertia, Coriolis or gravity compensation. Furthermore, it is possible to add the noise to the inverse dynamics as well.

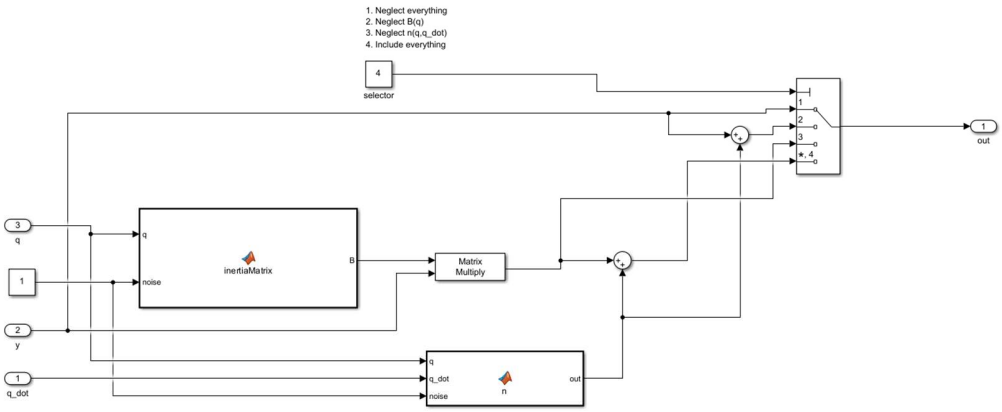


Figure 9 Non-linear compensation and coupling.

The results of the inverse dynamics controller are depicted in the Figure 10. According to figure, the q is converged to q_d accurately to the requested trajectory. It is important to note that the trajectory is very dynamics i.e. the graph is for just 3-time steps.

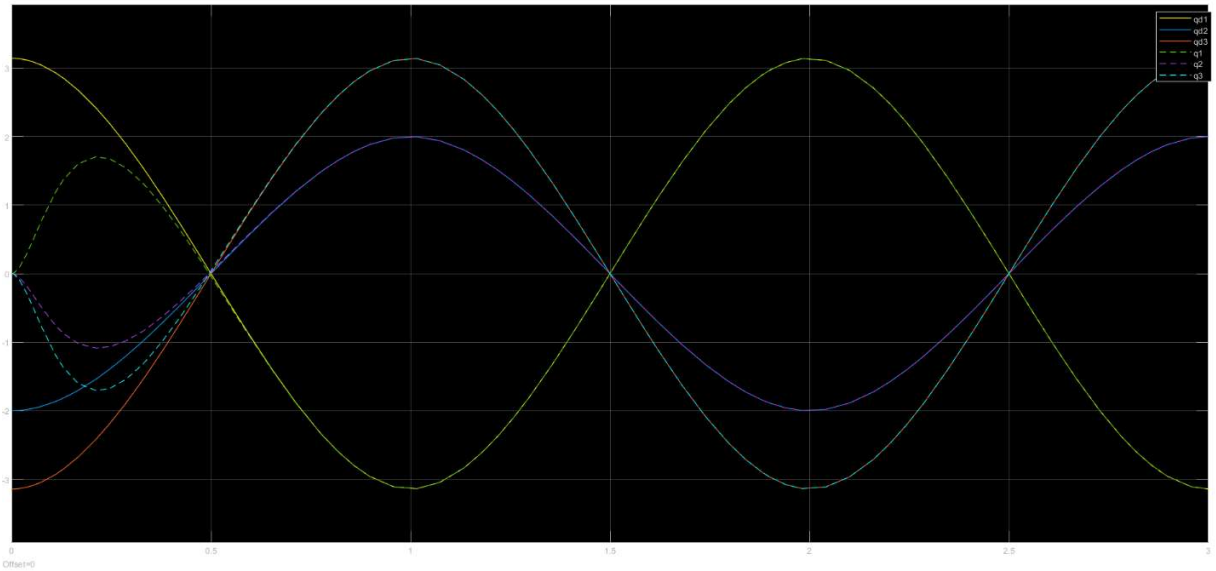


Figure 10 Inverse dynamics converges to q_d.

However, the inverse dynamics control alike PD controller with gravity compensation need accurate dynamic model to work. If the dynamic model is not accurate the controller might lead toward the error. Figure 11 shows the output of the controller with different inverse dynamic

model element. The masses of the links are different than the actual masses of manipulator to mimic the scenario. Figure shows that the graph does not converge toward the desired joint variable accurately as Figure 10.

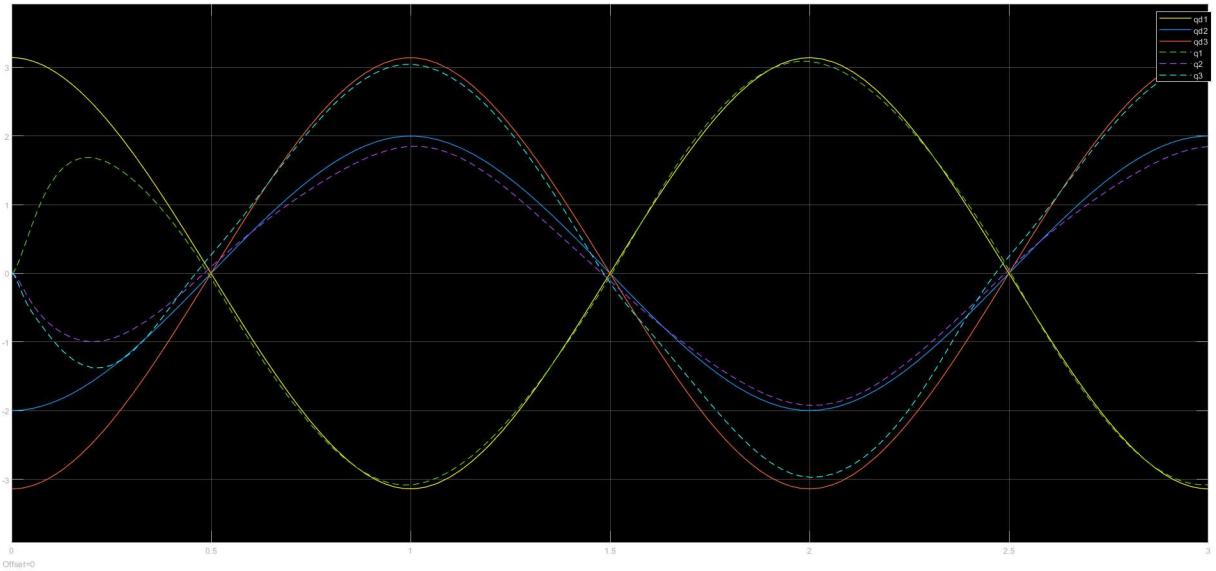


Figure 11 Inverse dynamics with slightly inaccurate model.

However, to find the importance of the compensation elements i.e. gravity, Coriolis, and inertia the graphs are made with the neglection of these elements from the controller. Figures 12-14 shows the graph of the inverse dynamic's controller without inertia, $n(q,\dot{q})$, and both respectively. Figure 12 shows the graph of the controller without inertia term in the controller. According to the figure, it is important to note that when the robot's joint variables change the

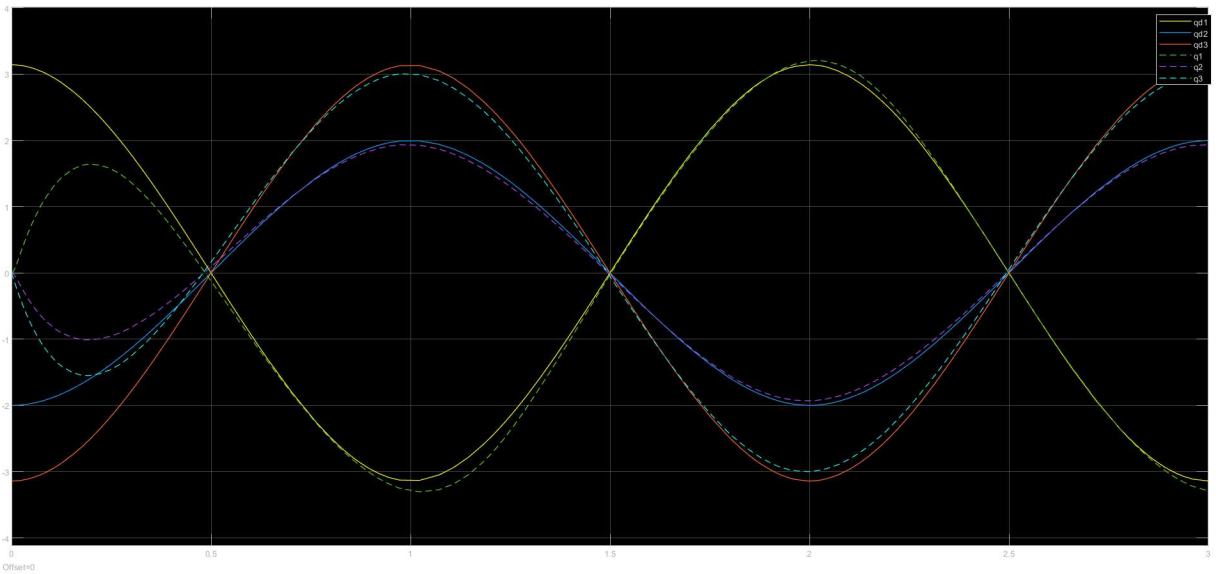


Figure 12 Inverse dynamic without inertia compensation

direction there is greater error then when the robot's joint variables are moving linearly. This is because to change the direction, inertia is needed to stop the robot and move to other direction. Similarly, Figure 13 demonstrate the graph of the controller without the $n(q, \dot{q})$ term in the controller. It can be seen that the controller has much greater error then the Figure 12. But still the controller for some time reaches the goal values. If all the compensation terms are removed as Figure 14 shows, the error is not much greater, but the controller never reach the desired goal values for joint values.

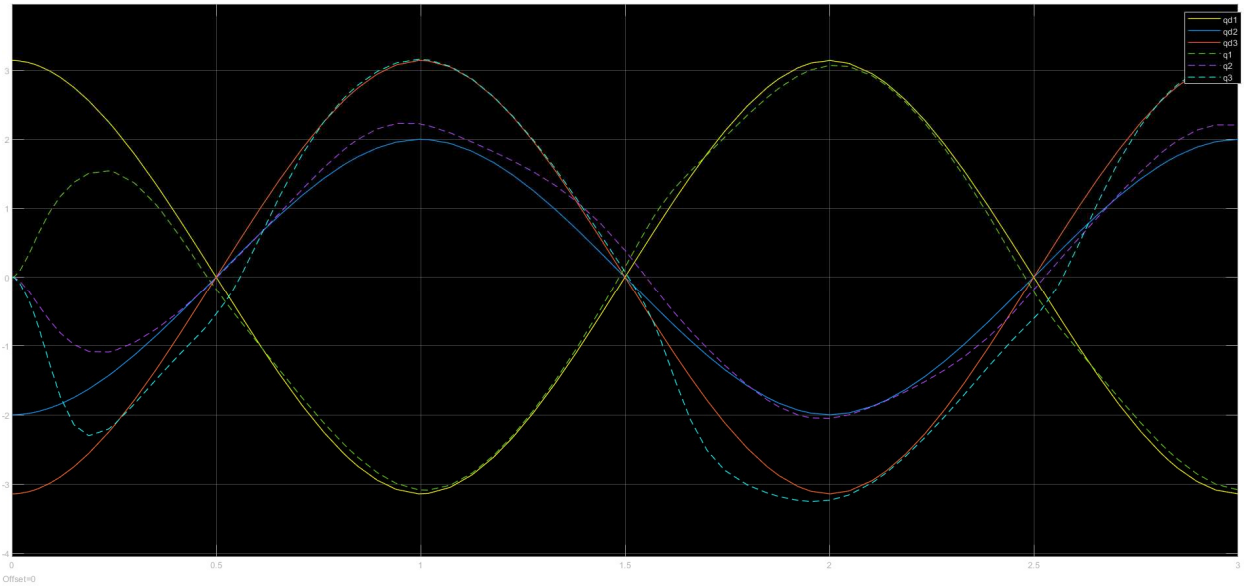


Figure 13 Inverse dynamic without gravity and Coriolis compensation

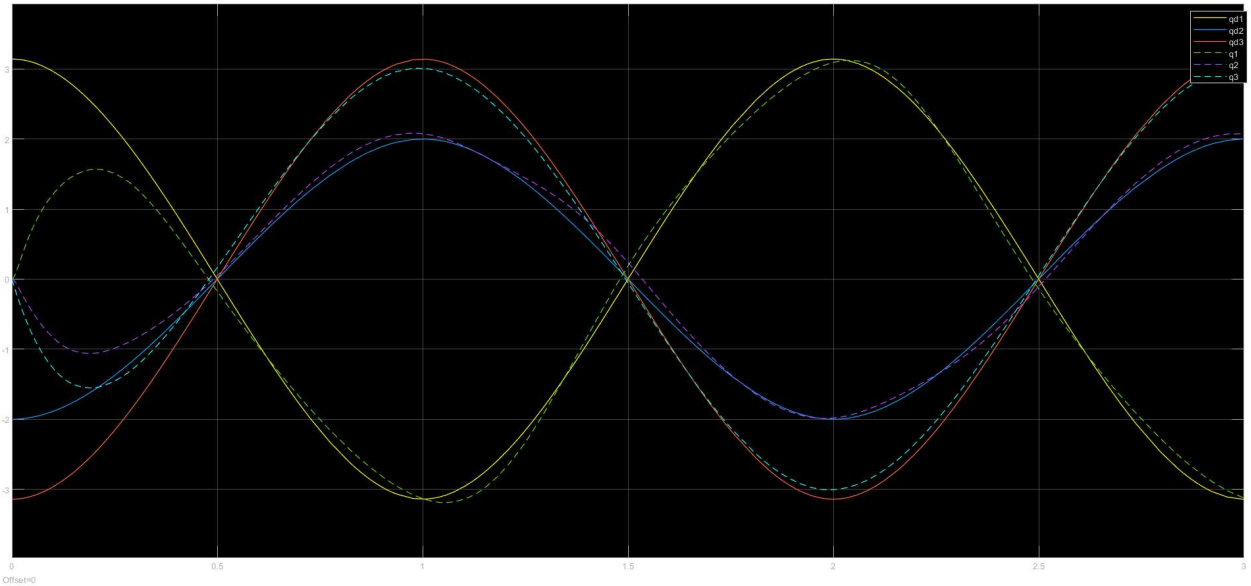


Figure 14 Inverse dynamics without any compensation

Increasing K_p values can help reduce settling time in a control system, but this also requires higher torque, which can affect the stability of the system and even damage its mechanical or electrical components. In real-world manipulators, there are often limits on the available current or voltage to ensure safety. To address this issue, a saturator block can be added to the Simulink model to detect when torque values exceed these limits. As it can be seen in Figure 15, very unrealistic large amount of torque is applied to reach the goal value due to the high value of K_p. However, in real world it can affect the accuracy of the controller as the mechanical error always have protections against the high current value. So if the controller can't have the particular torques, some part of the trajectory will not achievable. Unfortunately, in the current scenario, all saturated values are under the desired torques circumstances, so this effect is not visible.

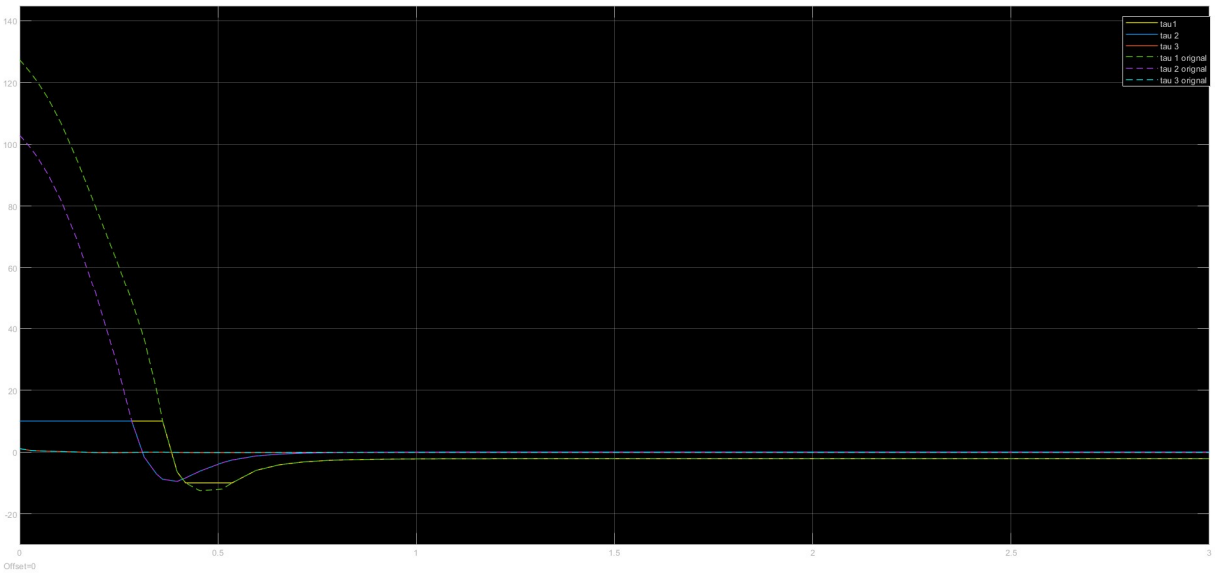


Figure 15 Torque pruning

9. Adaptive Controller

As previously mentioned, the performance of PD control with gravity compensation and inverse dynamics control highly depends on the accuracy of the manipulator's dynamic model. In case of inaccuracies, these controllers may not converge to the desired positions or values. To overcome this limitation, adaptive controllers can be utilized. Adaptive controllers can adapt to the changing dynamics of the manipulator by learning its dynamic parameters in real-time. These controllers use the dynamic parameters as a regressor problem and solve it for the particular manipulator. This way, the adaptive controller can estimate the unknown parameters and use them to adjust the controller's behavior accordingly.

However, implementing adaptive controllers is not a trivial task. To demonstrate the effectiveness of adaptive controllers, a 1 Degree of Freedom (DoF) manipulator is used. With an adaptive controller, the manipulator's dynamic parameters can be learned and updated in real-

time, allowing for better accuracy even if the dynamic model is inaccurate. The control law for the adaptive controller is as follow

$$\tau = Y(\ \ddot{q}_r, \dot{q}_r, q)\widehat{\Theta} + K_d\sigma \qquad (59)$$

where $Y(\ \ddot{q}_r, \dot{q}_r, q)\widehat{\Theta}$ for the 1 DoF manipulator is

$$\tau = \underbrace{\hat{B}(q)\ddot{q}_r + \hat{F}\dot{q}_r + \hat{g}(q)}_{Y(\ \ddot{q}_r, \dot{q}_r, q)\widehat{\Theta}} + K_d\sigma \qquad (60)$$

where $\hat{B}$, $\hat{F}$ and $\hat{g}$ is the estimated parameters for the adaptive control. Furthermore, $\ddot{q}_r$ and $\dot{q}_r$ is the reference joint variables which can be defined as

$$\ddot{q}_r = \ \ddot{q}_d + \lambda\dot{\tilde{q}} \qquad (61)$$

$$\dot{q}_r = \ \dot{q}_d + \lambda\tilde{q} \qquad (62)$$

Moreover, the σ is the error between the current joint variable and the reference joint variable velocity.

$$\sigma = \ \dot{q}_r - \dot{q} \qquad (63)$$

The variable $\widehat{\Theta}$ can be calculated using the integration of

$$\dot{\widehat{\Theta}} = K^{-1}\big(Y(\ \ddot{q}_r, \dot{q}_r, q)\big)^T\sigma \qquad (64)$$

It is important to note that the value of λ can be constant or can be used as ratio between the K_p and K_d given as K_p/K_d. Furthermore, Figure 16 shows the schema of adaptive control using the following manipulator

$$I\ddot{q} + F\dot{q} + mgdsin(q) = \tau \qquad (65)$$

where I is the inertia, F is the friction, m is the mass, g is gravity constant and d is the center of mass of the single link.

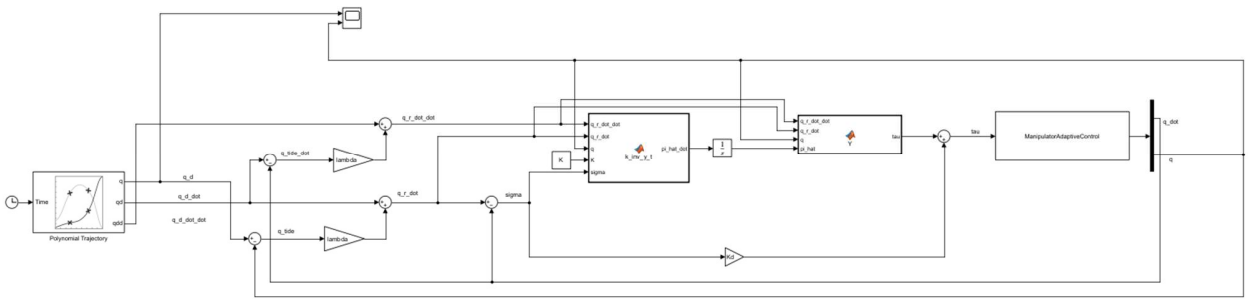


Figure 16 Control schema for the adaptive control in Simulink

The control parameters that are used in the adaptive controller

$$I = 0.1, F = 0.2, \text{g} = 9.806$$

$$k = 30.2, K_d = 500, \text{and } \lambda = 100$$

The mass of the object is calculated using

$$m = a * b * c * \rho$$

where $a = 1.1,\ b = 1.1$, and $c = 3.5$ are the width, height and length of a link with density $\rho = 2.7$. Moreover the center of mass d = $c/2$. By using the defined parameters the adaptive controller produces graph that is shown in the Figure 17. According to Figure 17, the q easily converges to q_d with no time.

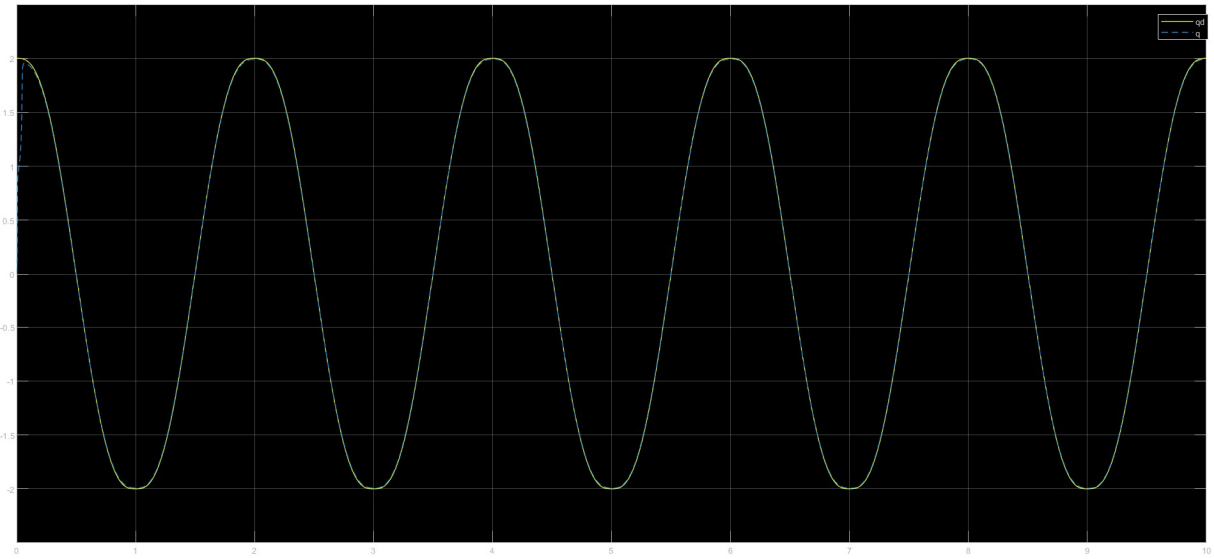


Figure 17 Adaptive control graph 1 DoF between desired and actual joint variable

10. Operational Space Controllers

As described earlier, operational space is a concept used in robotics to describe the desired motion of the end-effector of a robotic manipulator in the real world, typically represented by a Cartesian coordinate system [10]. It can also include the orientation of the end-effector, represented by Euler angles. Operational space is independent of the joint variables of the robot and provides a simplified way to describe and control the end-effector's motion. In operational space, the motion of the end-effector is specified as a set of desired positions and velocities, which are then translated into the appropriate joint commands to achieve the desired motion. This approach is useful in many applications, where the desired end-effector motion is specified in terms of the desired position and orientation of an object in the real world. By controlling the end-effector motion in operational space, the robot can perform the task more effectively and with greater accuracy.

By using the operational space controller, we can control the motion of the robot by controlling its position in real world cartesian coordinates. To convert from joint space to operational space, the accurate kinematic and differential kinematics i.e. analytical and geometrical Jacobian is needed. It is also important to note that, the problem of singularity is present on the operational space, where the robot loses 1 or more degree of freedom and cannot achieve the desired position. Furthermore, the operational space controllers are more complex and slower than the join space controllers.

10.1. PD controller with gravity compensation

As stated earlier, to control the positions of a robot, a proportional-derivative (PD) control with gains is used. In operational space, the PD controller is similar to the joint space PD controller, but it operates in a different space. The operational space PD controller controls the position in Cartesian coordinates, unlike the joint space PD controller that controls the position in joint variables. The controller calculates the difference between the desired and actual positions as well as orientations in terms of Euler angles and applies a proportional and derivative gain to this error to generate new torque, which moves the end-effector towards the desired position. However, the weight of the robot's links can affect the torques generated by the PD controller when the robot is subject to gravity, leading to errors in position control. To compensate for this, a gravity compensation term is added to the controller to account for the gravitational forces acting on the robot's links, ensuring that the control signal accurately reflects the desired joint positions. However, the operational space PD controller may face the problem of singularity, which may cause it to fail to converge to the desired positions or orientations.

It is important to note that, in this literature, the desired position and orientation is defined using the direct kinematics. The desired set of q_d is defined similar to joint space and homogeneous matrix H_4^0 is used to extract the position P_e. The orientation ϕ_e in term of ZYZ Euler angles are extracted from the R_4^0 rotation matrix. The desired position and orientation vector x_d is represented as 6×1 vector as follows

$$x_d = \begin{bmatrix} P_e \\ \phi_e \end{bmatrix} \tag{66}$$

However, the velocity of the end-effector in cartesian space can be calculated using the analytical Jacobian.

$$\dot{x}_d = J_a(q) x_d \tag{67}$$

The control law for the PD controller for operational space with gravity compensation can take a following form which is also reflected in Figure 18.

$$\tau = g(q) + (J_a(q))^T K_p \tilde{x} - K_d \dot{q} \tag{68}$$

It is important to note that, there is a manual switch which can be used to remove the gravity compensation from the controller. The PD controller parameters that are used in the controller are following.

$$K_p = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 250 & 0 & 0 & 0 & 0 & 0 \\ 0 & 250 & 0 & 0 & 0 & 0 \\ 0 & 0 & 250 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Whereas 2 set of q's are used to generate the desired position vector for this experiment.

$$q_{d1} = [2, -0.1, -3]$$

$$q_{d2} = [-2, 0.12, -3]$$

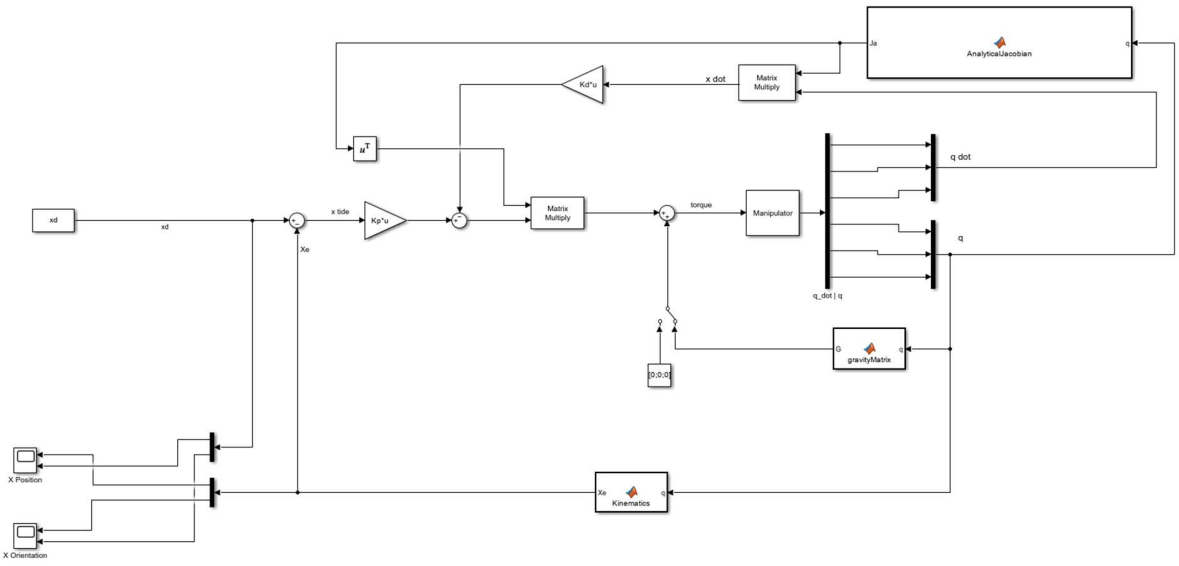


Figure 18 Operational space PD controller with gravity compensation schema in Simulink

The graphs in the Figures 19 and 20, shows the convergence of the position and orientation to desired position and orientation for the q_{d1}. In the orientation graph there is a step for some time step in which the orientation moves suddenly to the desired is due to unknown reason. The best estimate is to that value of orientation and position x, the singularity occurs which shows

such weird behavior; however the robot recovers from this situation and move toward the desired position and orientation.

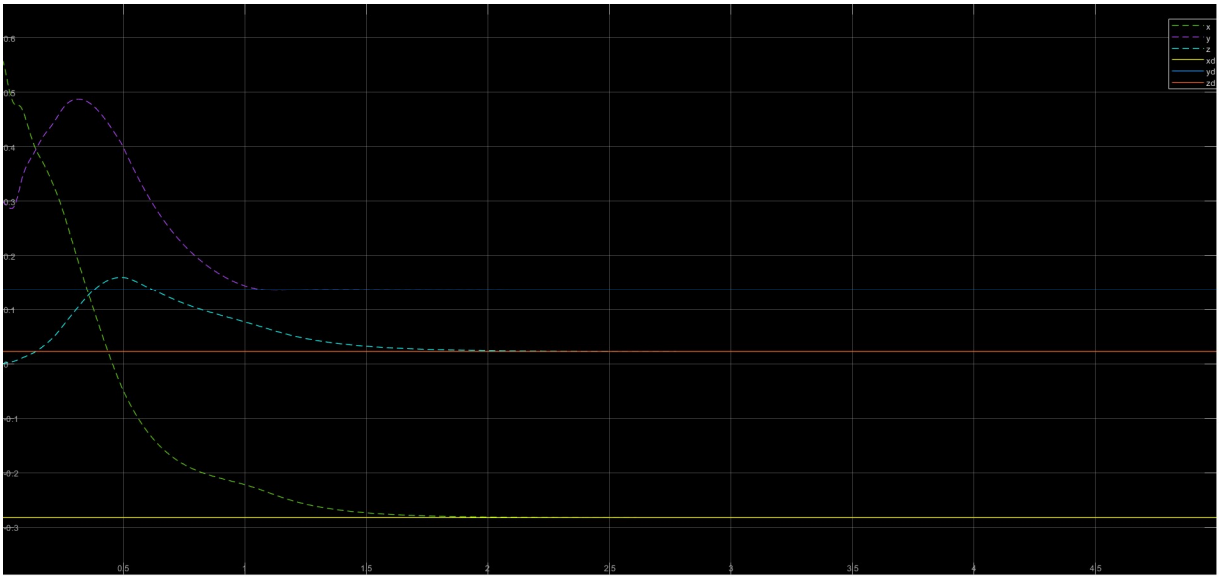


Figure 19 A graph for q_{d1} converges to desired position P_{de}.

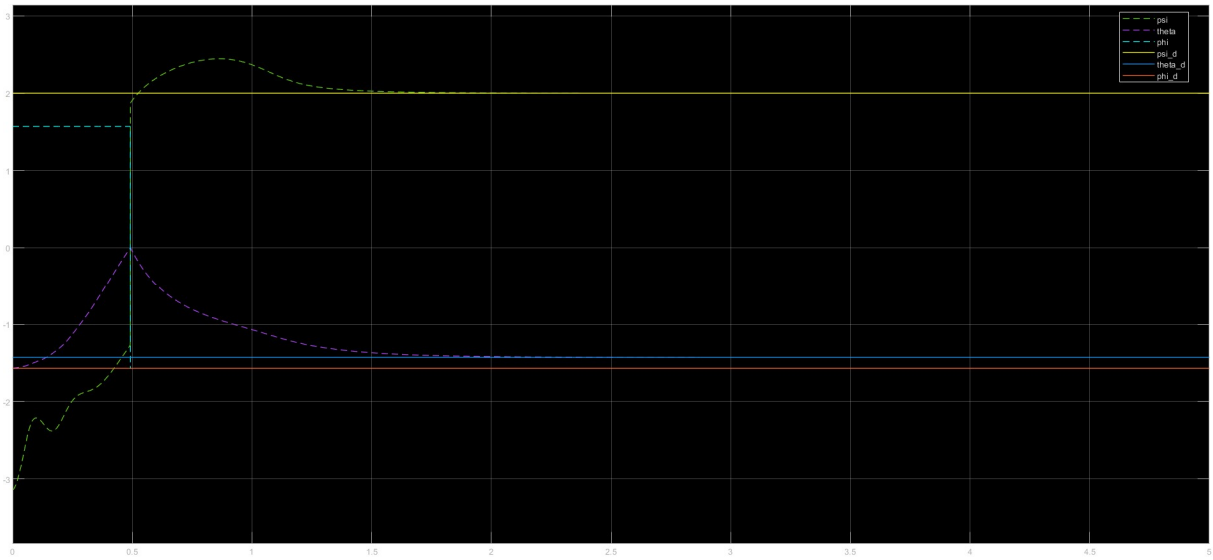


Figure 20 A graph for q_{d1} converges to desired orientation ϕ_{de}.

Figure 21 and 22 demonstrate the controller without the gravity compensation. When the gravity is not compensated in the controller, both position and orientation do not converge to the desired position and orientation P_{e1} and ϕ_{e1}, respectively. The reason why orientation, Euler angle θ (theta), and ψ (psi) do not converge to the desired orientation is, the link 1 and link 2 of the robot is highly dependent on the gravity. Due to lack of torques on the joints lead to not converge to desired orientation and of course desired position.

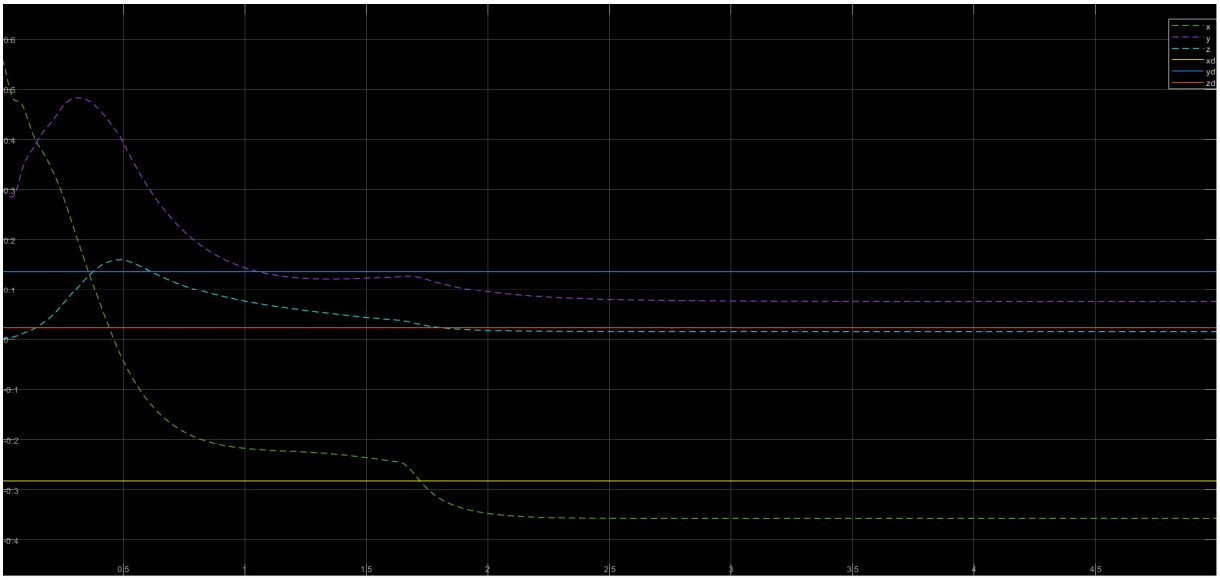


Figure 21 A graph without gravity compensation for q_{d1} does not converges to desired position P_{de}.

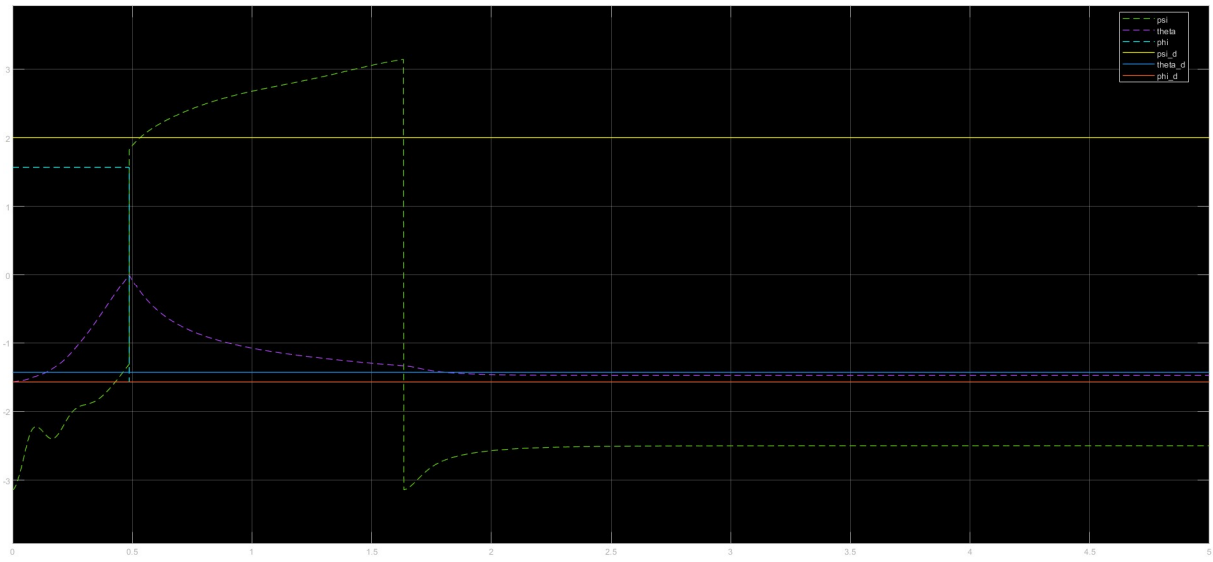


Figure 22 A graph for q_{d1} without gravity compensation does not converge to desired orientation ϕ_{de}.

The Figures 23,24 shows the graphs for q_{d2} orientation and position. The graphs clearly shows that the position and orientation of the end-effector did not reach to the desired position and orientation of the end-effector. The reason behind this situation is singularity. The robot for this orientation and position may lose some degree of freedom, which makes the robot reach to its desired position impossible. This is the one of the biggest problems which is already stated about the operational space controllers.

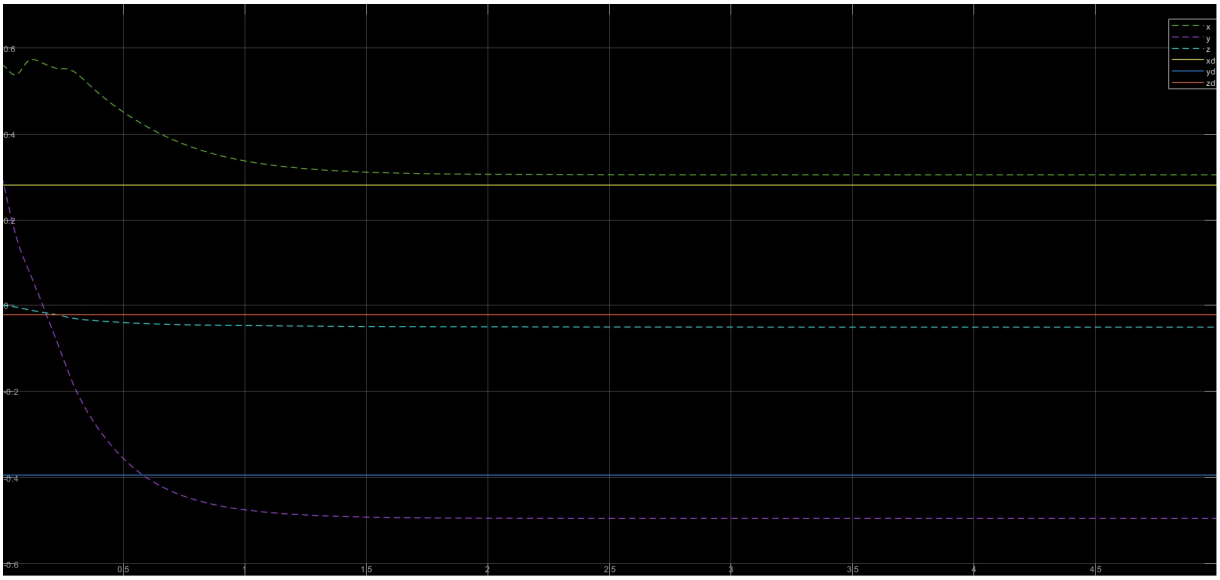


Figure 23 A graph for q_{d2} does not converges to desired position P_{de} due to singularity.

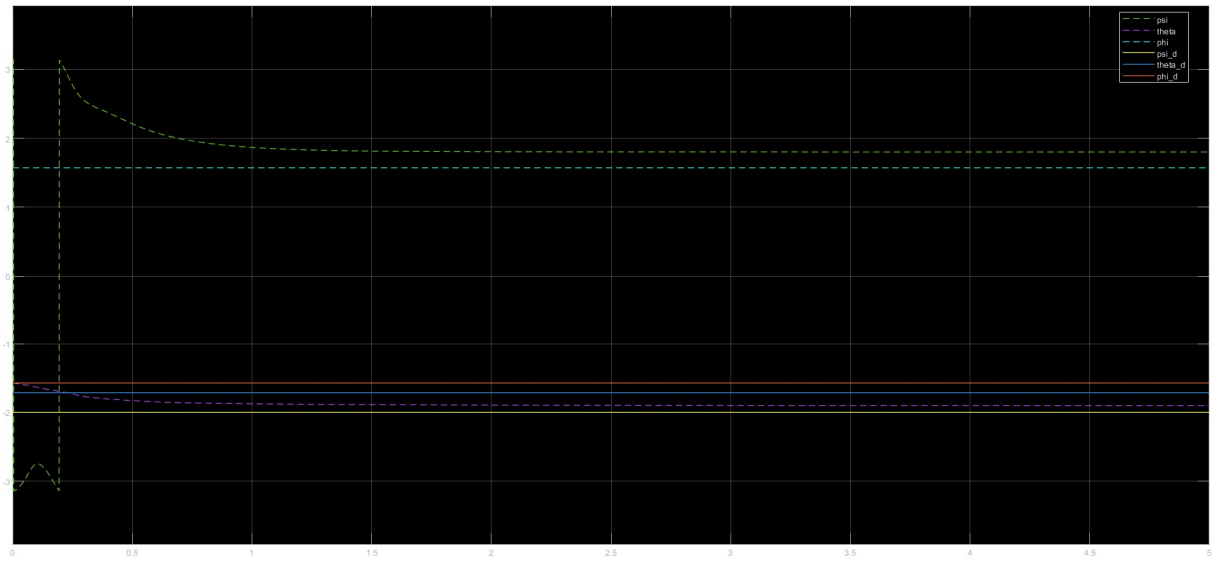


Figure 24 A graph for q_{d2} does not converge to desired orientation ϕ_{de} due to singularity.

10.2. Inverse dynamics controller

Same as joint space PD controller, the operational space PD controller with gravity compensation suffers from performance degradation in the presence of noise in the desired position and orientation. To overcome this issue, an effective solution is the use of the inverse dynamic controller in the operational space. The inverse dynamic controller has been shown to perform well even in the presence of noise and has been highly effective in tracking trajectories in the operational space. It compensates for all noise in the operational space by utilizing elements from

the equations of motion such as inertia, Coriolis, and gravity terms, while also accounting for the dynamic nature of the position and orientation.

It is important to note that the effectiveness of the inverse dynamic controller is highly dependent on the accuracy of the manipulator's dynamic model. Therefore, accurate modeling of the system is crucial to ensure the success of this type of controller. Additionally, like the PD controller in operational space, the problem of singularity may also occur in the inverse dynamic controller. This can result in the controller failing to converge to the desired positions and orientations. The control law for inverse dynamics stated that

$$\tau = B(q)y + n(q, \dot{q}) \quad (69)$$

Where y is a stabilizing linear control defined as

$$y = J_a^{-1}(q)(\ddot{x}_d + K_d\dot{\tilde{x}} + K_p\tilde{x} - \dot{J}_a(q, \dot{q})\dot{q}) \quad (70)$$

The PD controller parameters that are used in the controller are following.

$$K_p = \begin{bmatrix} 280 & 0 & 0 & 0 & 0 & 0 \\ 0 & 280 & 0 & 0 & 0 & 0 \\ 0 & 0 & 70 & 0 & 0 & 0 \\ 0 & 0 & 0 & 70 & 0 & 0 \\ 0 & 0 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 70 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 30 & 0 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 \end{bmatrix}$$

Figure 25 shows the schema designed in Simulink for the inverse dynamic controller in operational space. To set the x_d trajectory for the controller, the q_d is defined using polynomial trajectory block and then converted to x_d, $\dot{x}_d$, and $\ddot{x}_d$ using the direct kinematics, analytical Jacobians and chain rule of differential respectively. The two blocks are defined named as stabilizing linear control and non-linear compensation and coupling.

Figure 26 shows the PD gains used along with analytical Jacobian to add the stability to the controllers. It is worth mentioning that, the output y from the block is not actual y stated in Equation 69, it needs to be multiplied by J_a^{-1} to produce the y term.

It is worth mentioning that the framework comprises various test scenarios to evaluate the controller's performance as shown in Figure 27. These scenarios involve examining the controller's behavior when neglecting the Inertia, Coriolis, or gravity compensation. Additionally, it is feasible to introduce noise to the inverse dynamics as part of the testing process.

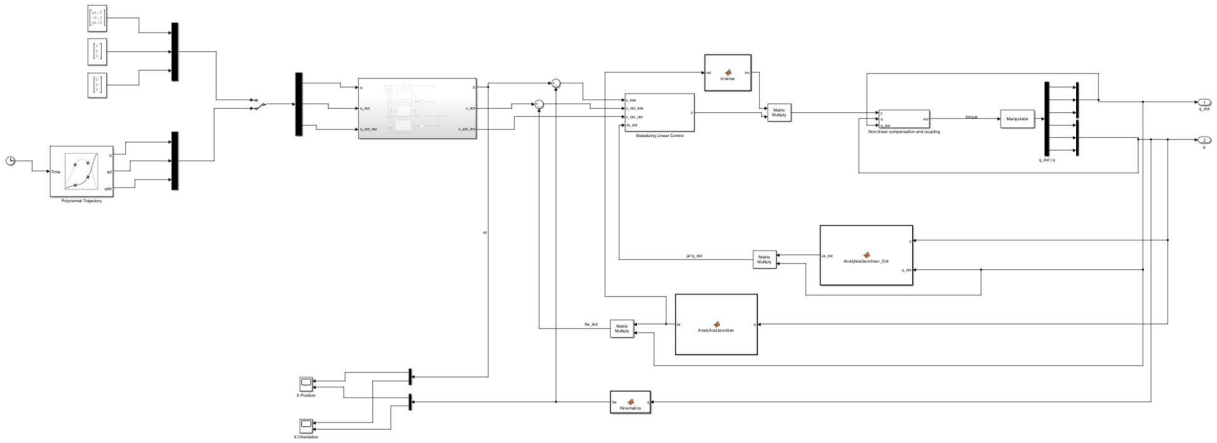


Figure 25 Operational space inverse dynamics schema in Simulink.

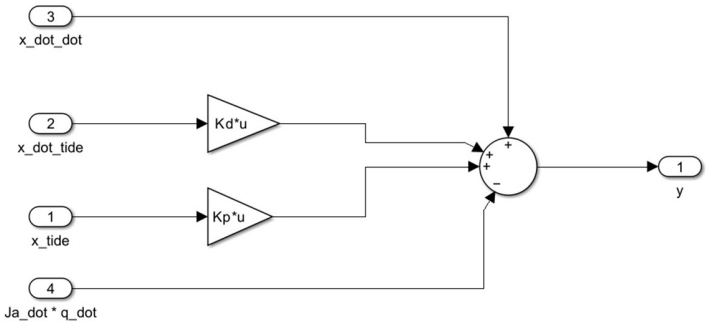


Figure 26 Stabilizing linear control.

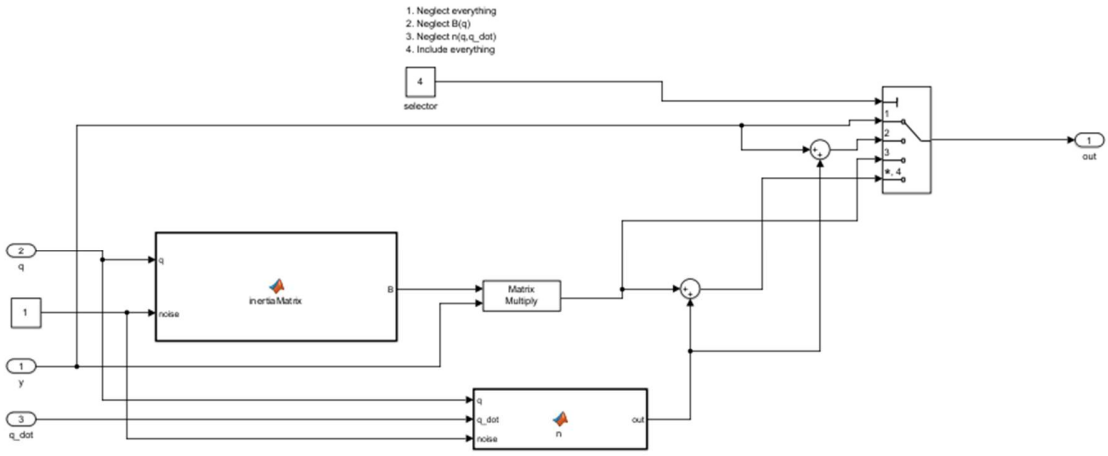


Figure 27 Non-linear compensation and coupling

As it can be clearly seen by the Figure 28 and 29, the position and orientation are tracked precisely by the controller. However, again the orientation shows the weird jump in the phi, the best guess that can be made for this behavior is singularity in Jacobian matrix at some particular value of q from which the robot recovered in other time step.

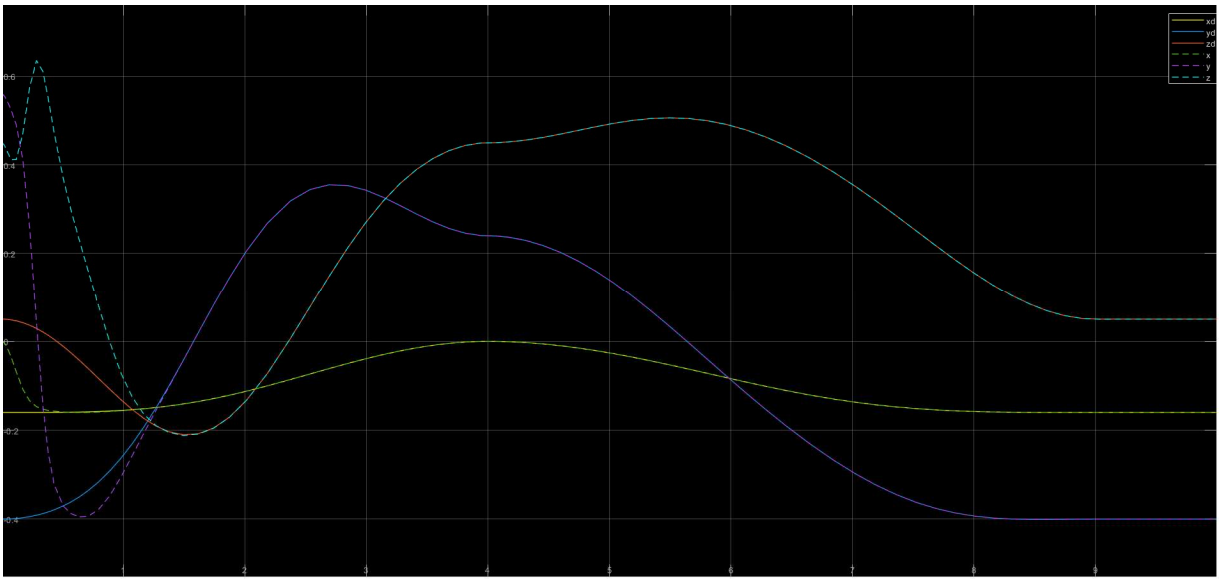


Figure 28 Inverse dynamics in operational space position P_e graph.

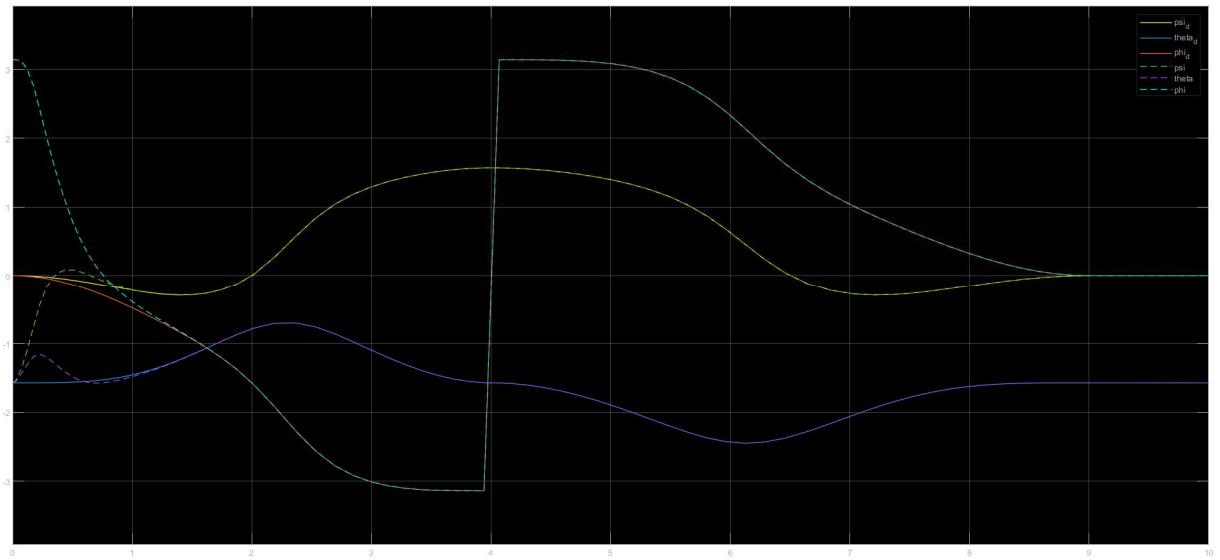


Figure 29 Inverse dynamics in operational space position ϕ_e graph.

11. Force Control

The robotic manipulator can have an interaction with the environment, and it is very important to control the force of the manipulator [11]. The contact force at the manipulator's end-effector is the key measure of this interaction. It is necessary to estimate the contact forces that

manipulator could have in the future. This section focuses on the operational space motion control schemes during manipulation tasks, and introduces concepts such as mechanical compliance, impedance and admittance. These force controls using motion controls are known as indirect force control. Direct force control schemes are also presented, which are obtained by modifying motion control schemes with an outer force feedback loop. However, the forces are naturally defined in operational space, so in this section the operational space models are used for better convenience.

11.1. Indirect Force Control

As stated earlier, indirect force control is a type of control strategy where the force applied by a robotic manipulator is indirectly controlled by adjusting its motion. Specifically, this is done by designing a motion control law that generates desired motion trajectories for the manipulator, which in turn results in the desired contact force between the end-effector of the manipulator and the environment. The force is not directly controlled, but rather is an outcome of the motion control. In the context of indirect force controllers, mechanical impedance, compliance, and admittance controllers are defined in the upcoming sections.

11.1.1. Compliance Controller

Compliance control refers to the ability of a robot to passively adapt to external forces, allowing it to deform or move in response to those forces. In other words, compliance control allows the robot to be compliant or "soft" in the face of external forces. The goal of compliance control is to reduce the effects of external forces on the robot and the objects it is manipulating, as well as to improve the accuracy and safety of the robot's movements. In the compliance control the virtual wall is used with the spring model, which allow the robot to interact with it.

It is important to consider that, for the sake of simplicity, the virtual wall is perpendicular only to one axis. In other words, the robot can only interact with the environment in only one direction. In this literature, the robot only interacts on z-axis. Furthermore, in this controller, the J_{a_d} is not working, so J_a is used in the controller. The control law for the compliance control is defined as

$$\boldsymbol{\tau} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J_a^T}(\boldsymbol{q})(\boldsymbol{K_p}\tilde{\boldsymbol{x}} - \boldsymbol{K_d}\boldsymbol{J_a}(\boldsymbol{q})\dot{\boldsymbol{q}}) \quad (71)$$

Where the outer wrench h_e can be defined as

$$\boldsymbol{h_e} = \boldsymbol{K_e}\boldsymbol{dx_{r,e}} \quad (72)$$

Where K_e is the spring constant, and $dx_{r,e}$ is the displacement between the reference frame and position of end-effector. The total control law with external wrench became

$$\boldsymbol{\tau} - \boldsymbol{J^T}(\boldsymbol{q})\boldsymbol{h_e} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J_a^T}(\boldsymbol{q})(\boldsymbol{K_p}\tilde{\boldsymbol{x}} - \boldsymbol{K_d}\boldsymbol{J_a}(\boldsymbol{q})\dot{\boldsymbol{q}}) \quad (73)$$

In Equation 72, the J referred to geometrical Jacobian. However, two extreme cases that is $K_e \gg K_p$ and $K_e \ll K_p$ used to study the compliance control. However, the K_p and K_d is not changed and kept constant.

$$K_p = \begin{bmatrix} 550 & 0 & 0 & 0 & 0 & 0 \\ 0 & 550 & 0 & 0 & 0 & 0 \\ 0 & 0 & 550 & 0 & 0 & 0 \\ 0 & 0 & 0 & 30 & 0 & 0 \\ 0 & 0 & 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 0 & 0 & 30 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 70 & 0 & 0 & 0 & 0 & 0 \\ 0 & 70 & 0 & 0 & 0 & 0 \\ 0 & 0 & 70 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

and Ke two scenarios are used

$$K_e = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 150 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (K_p \gg K_e)$$

$$K_e = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 750 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (K_p \ll K_e)$$

Note, K_e only needed in z-axis because the environment is presented on the z-axis only.

The Figures 31, 32 shows compliance control output for position P_e and orientation ϕ_e. As it can be seen that the position and orientation are not converged because of the environment stopping them to do so. However, there is little elasticity due to the spring-based model of the environment. However, the figures show the extreme scenario where $K_p \gg K_e$. As it can be observed that the error in $\tilde{x}$ is smaller as $\tilde{x} = x_d - x$. The greater effect is shown in the z-axis of the position, but very small error can also be observed in y-axis as well. This is due to the effect of one link on the other link of the manipulator. In orientation, the error is very small but it can be observed that only theta is effected and cannot achieve its desired orientation.

The Figure 33 shows the external wrench of the end-effector on the environment. On the contact to the environment, the manipulator starts exserting force upon the environment. As it can be seen that the force starts getting higher and environment get deformed until there is a point where it became constant due to elasticity property. As $K_p \gg K_e$, the h_e force is greater than the equilibrium force exserted by the environment to the manipulator.

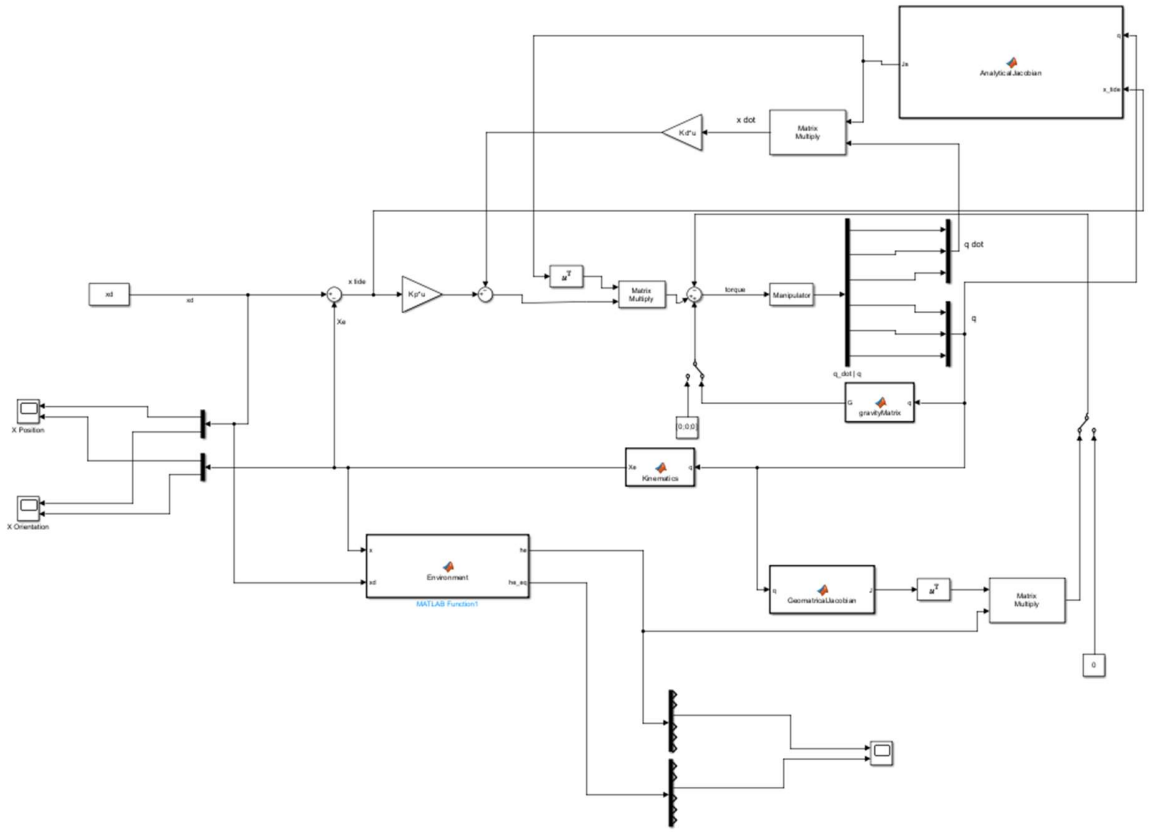


Figure 30 Compliance controller

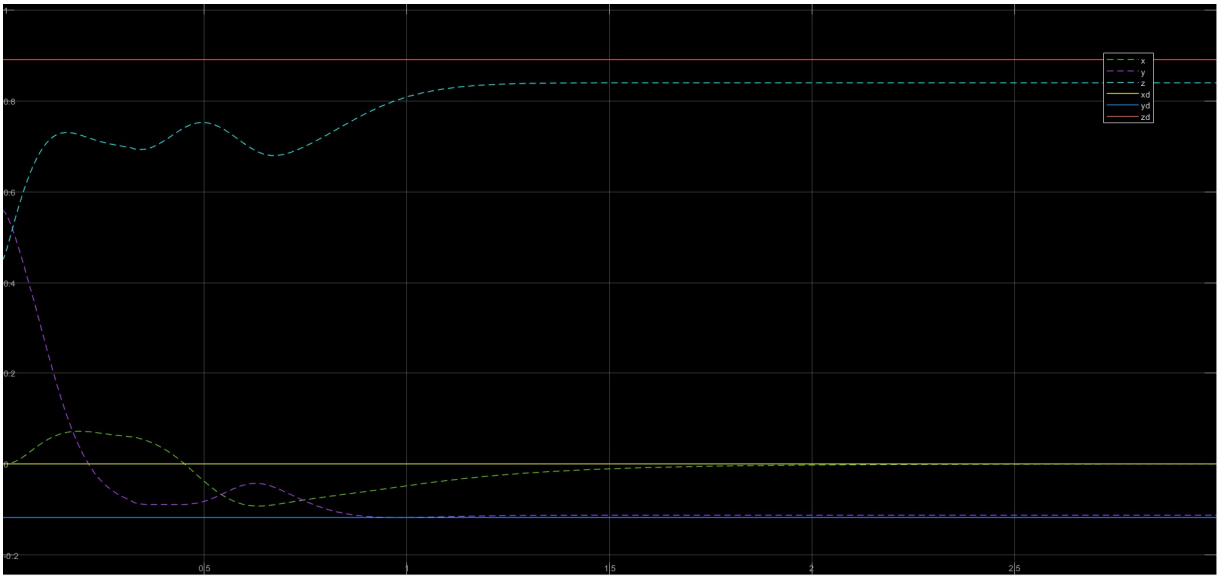


Figure 31 Compliance controller with $K_p \gg K_e$ position graph.

Figure 34,35 shows the position and orientation of the end-effector for the case $K_p \ll K_e$. As the environment is stiffer than the manipulator. The manipulator has more error in the position as

well as in the orientation as compared to the above case. The environment shows very less elasticity as showed in the figures.

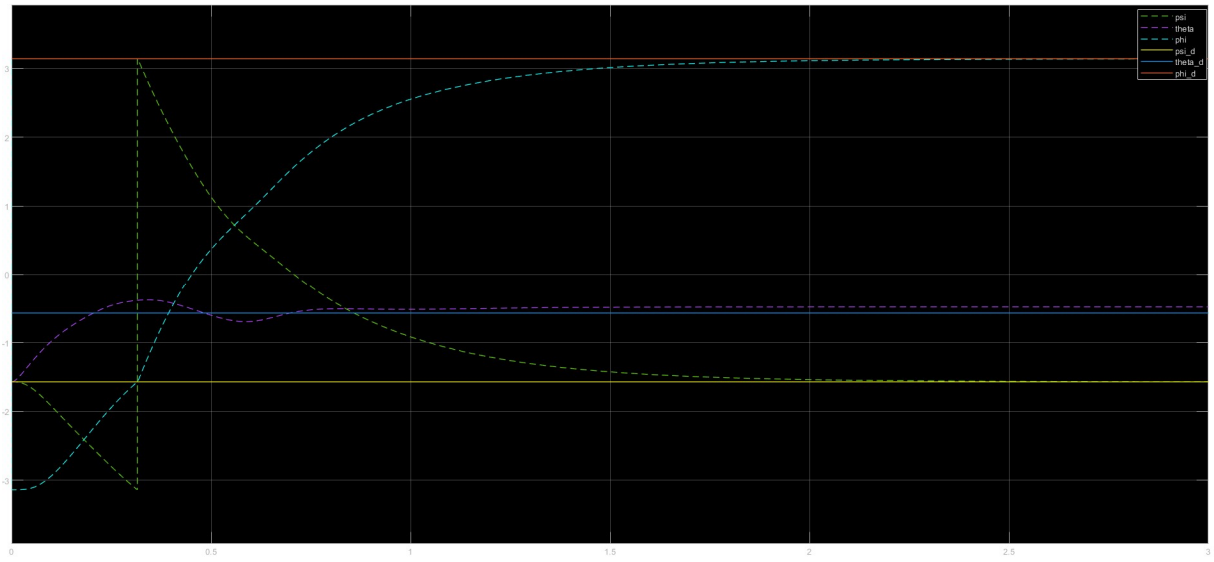


Figure 32 Compliance controller with $K_p \gg K_e$ orientation graph.

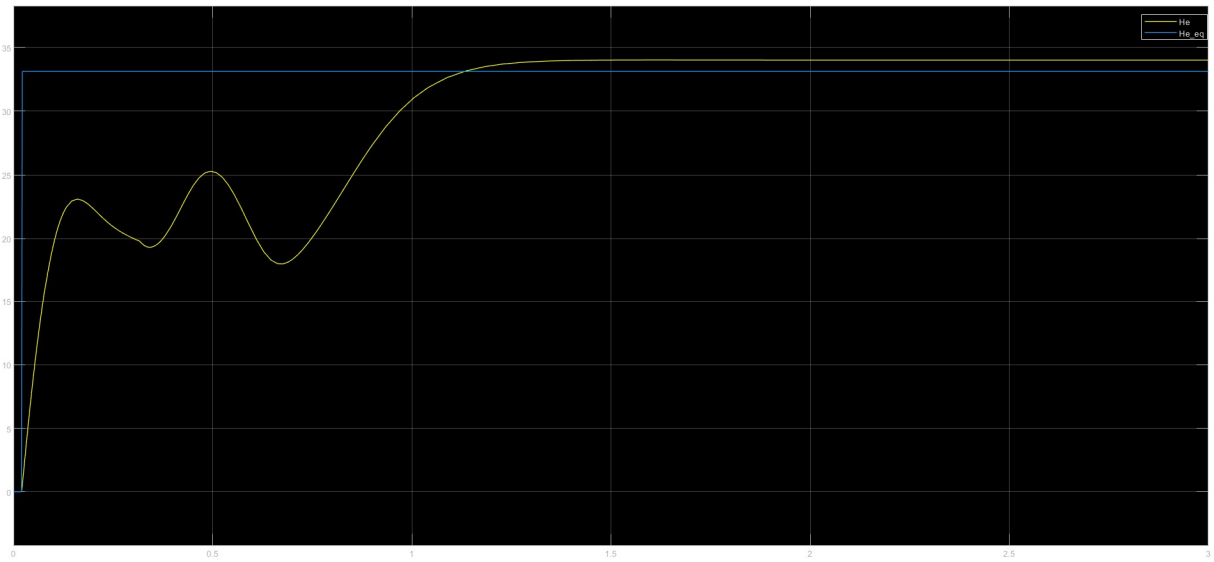


Figure 33 Compliance controller with $K_p \gg K_e$ external wrench h_e graph.

However, the Figure 36 shows that the he force is much greater than the equilibrium force because the manipulator require more force to reach the desired position and orientation as the environment is very stiff and it is not easily deformable.

If the robot is set to the free motion i.e. it do not interact with the environment, the controller act as a PD controller in operational space and perfectly converges to the desired position and

orientation as shown in the Figure 37 and 38. Furthermore, the equilibrium force and the external wrench is 0 as well as it can imagined.

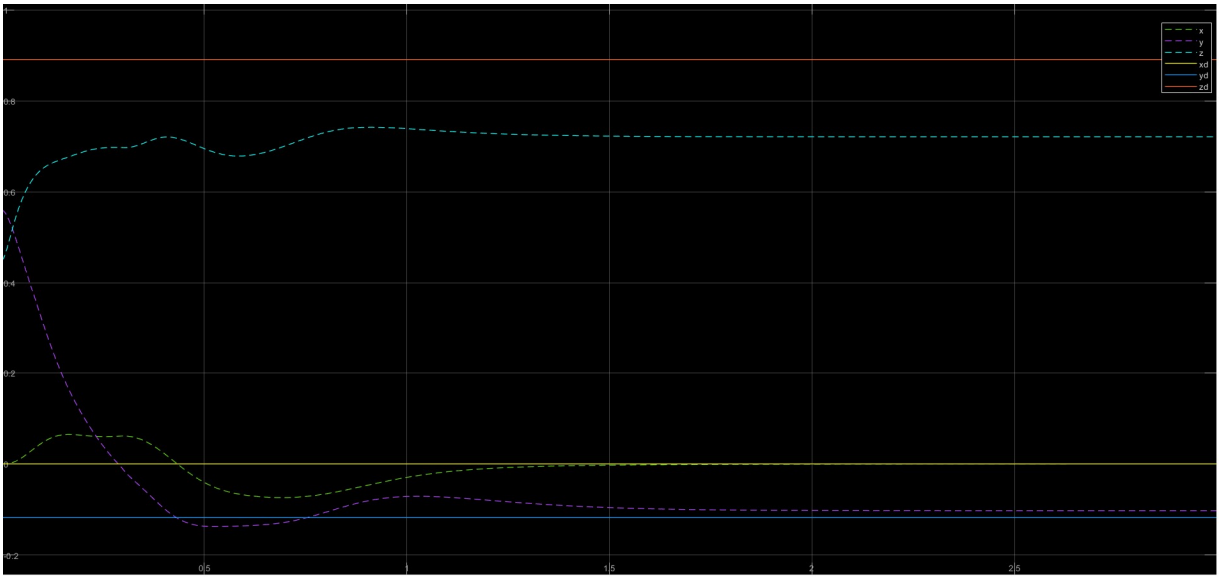


Figure 34 Compliance controller with $K_p \ll K_e$ position graph.

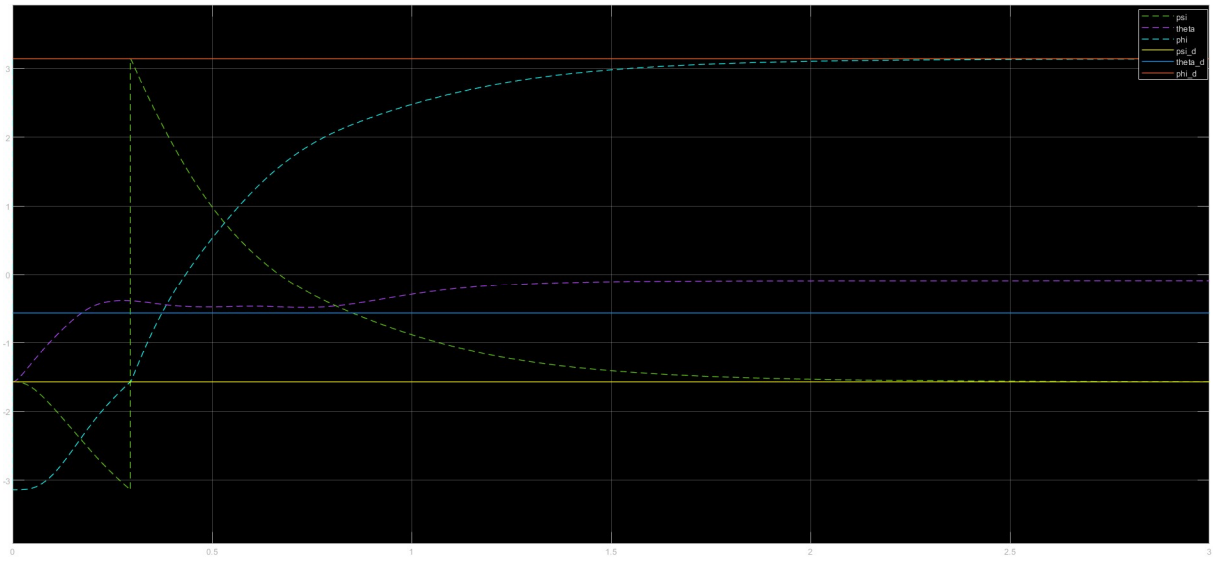


Figure 35 Compliance controller with $K_p \ll K_e$ orientation graph.

11.1.2. Impedance Controller

In impedance control, the control law for inverse dynamics controls in operational space defined in Equation 70 is modified to form the impedance control law. According to impedance control law

$$\tau = B(q)y + n(q,\dot{q}) - J^T(q)h_e \quad (74)$$

where

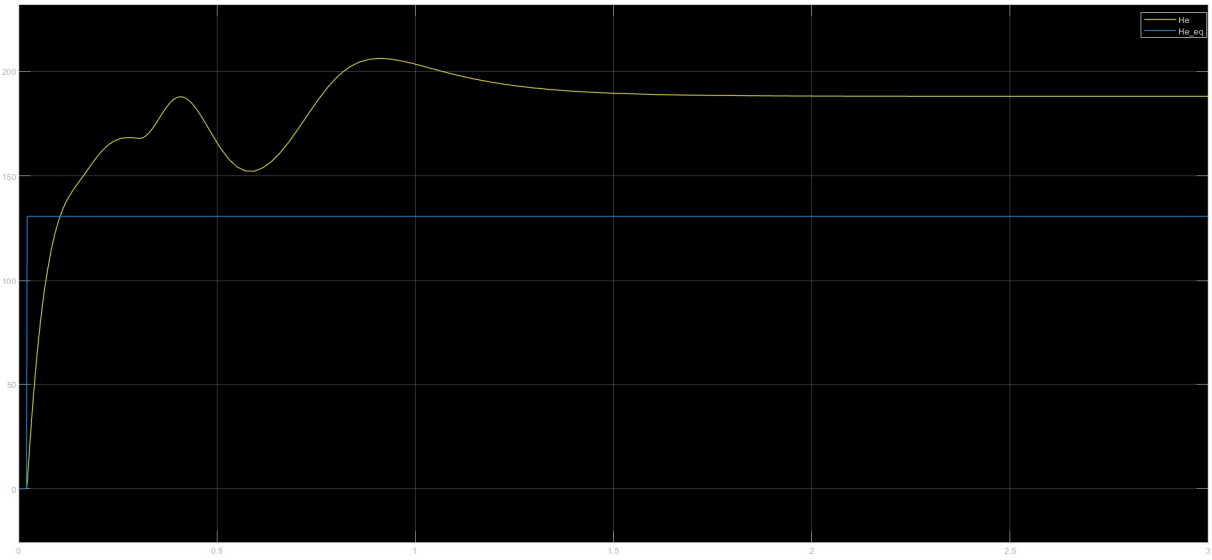


Figure 36 Compliance controller with $K_p \ll K_e$ external wrench h_e graph.

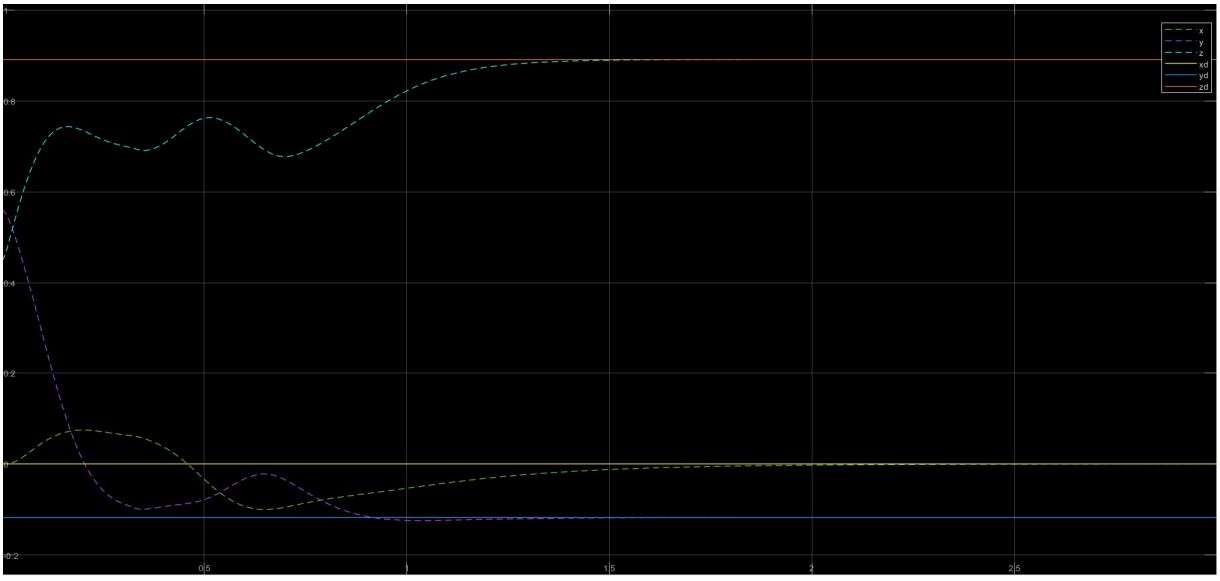


Figure 37 Compliance control without external wrench h_e position graph.

$$y = J_a^{-1}(q)M_d^{-1}(M_d\ddot{x}_d + K_d\dot{\tilde{x}} + K_p\tilde{x} - M_d\dot{J_a}(q,\dot{q})\dot{q}) \quad (75)$$

In the above equation the M_d referred to the mass matrix of the system. The overall schema of the Simulink is displayed in Figure 40, 41, and 42. However, in this experiment, following parameters are used in the control law

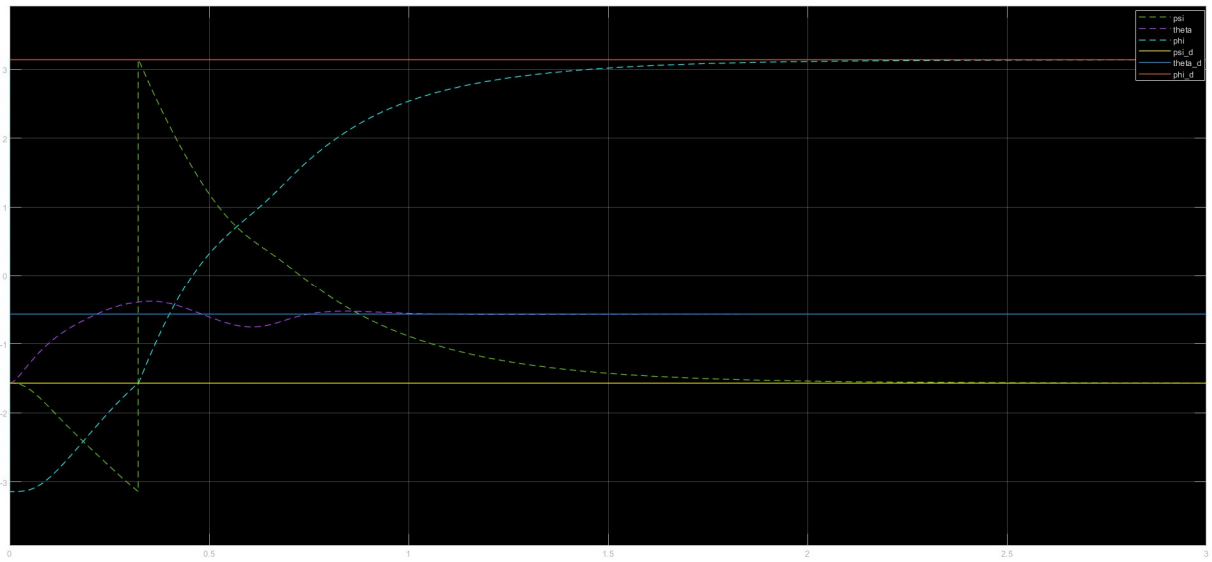


Figure 38 Compliance control without external wrench h_e orientation graph.

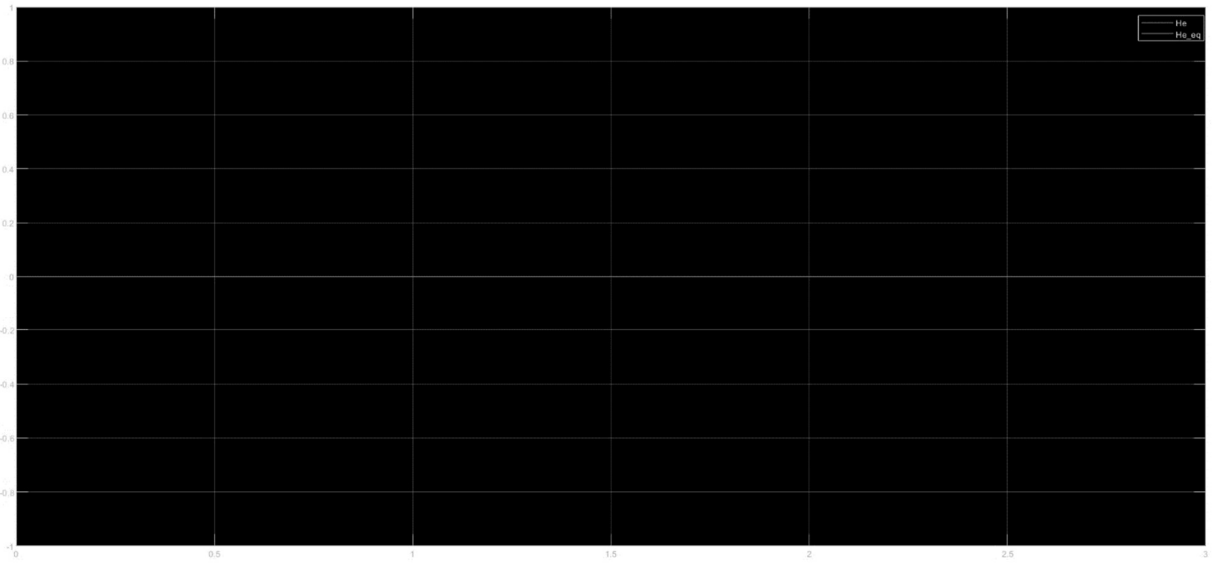


Figure 39 Compliance control without external wrench h_e , external wrench h_e graph.

$$K_p = \begin{bmatrix} 250 & 0 & 0 & 0 & 0 & 0 \\ 0 & 250 & 0 & 0 & 0 & 0 \\ 0 & 0 & 250 & 0 & 0 & 0 \\ 0 & 0 & 0 & 250 & 0 & 0 \\ 0 & 0 & 0 & 0 & 250 & 0 \\ 0 & 0 & 0 & 0 & 0 & 250 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}$$

$$K_e = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_d = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

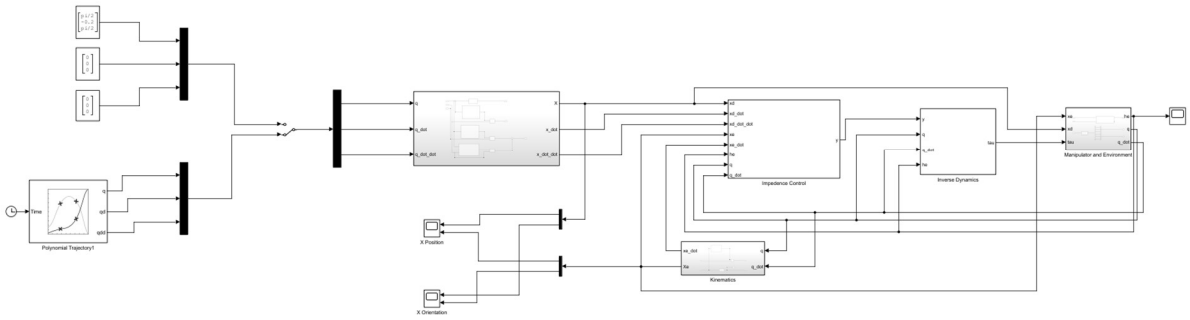


Figure 40 Impedance control schema in Simulink.

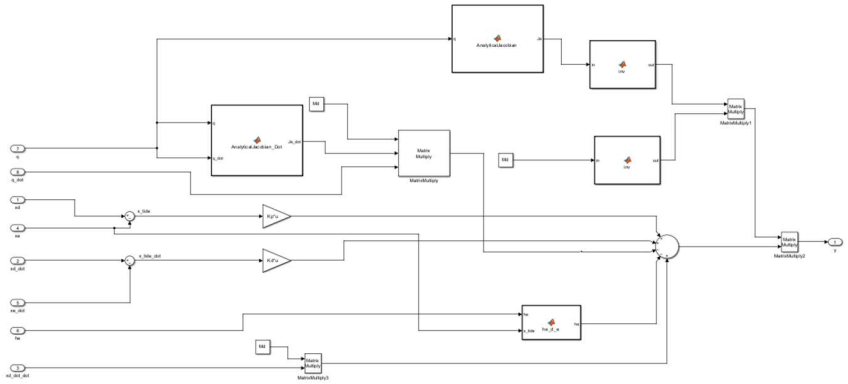


Figure 41 Impedance control block schema.

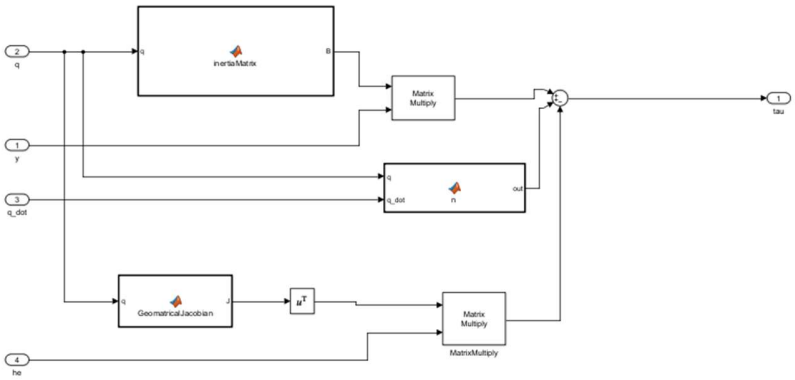


Figure 42 Inverse dynamics schema block.

Figure 43, 44 shows the position and orientation of the robot end-effector with the desired position of end-effector. It can be seen that the robot follow the trajectory while interacting with the environment. There is the error in the z-axis and x-axis due to the elastic environment present in front of the desired frame. So, the position and orientation did not converged to the desired orientation and position.

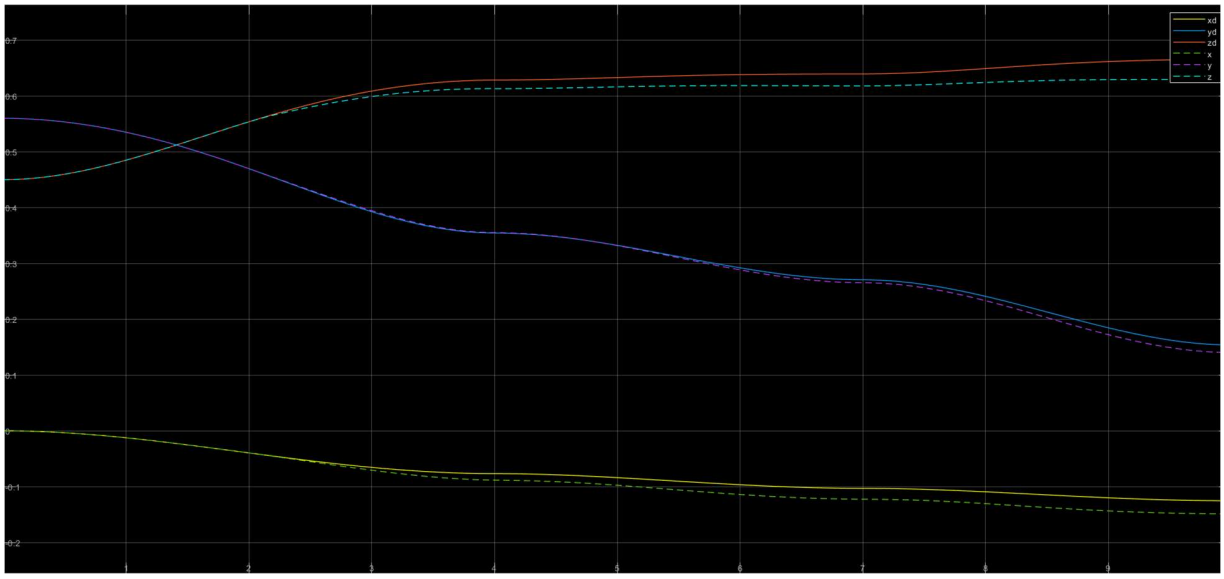


Figure 43 Impedance control position P_e graph.

The Figure 45 shows the external wrench h_e that increases when the robot interact with the environment. But it can be seen that the environment shows the elasticity behavior on the contact with the end-effector.

11.1.3. Admittance Controller

Admittance control is a type of force control that allows a robot to react to external forces in a compliant way, similar to impedance control. However, unlike impedance control, admittance control focuses on controlling the motion of the robot based on the applied force, rather than controlling the force based on the desired motion. Moreover, the admittance control separate the motion controller with the impedance controller. Which allow the robot to be more robust

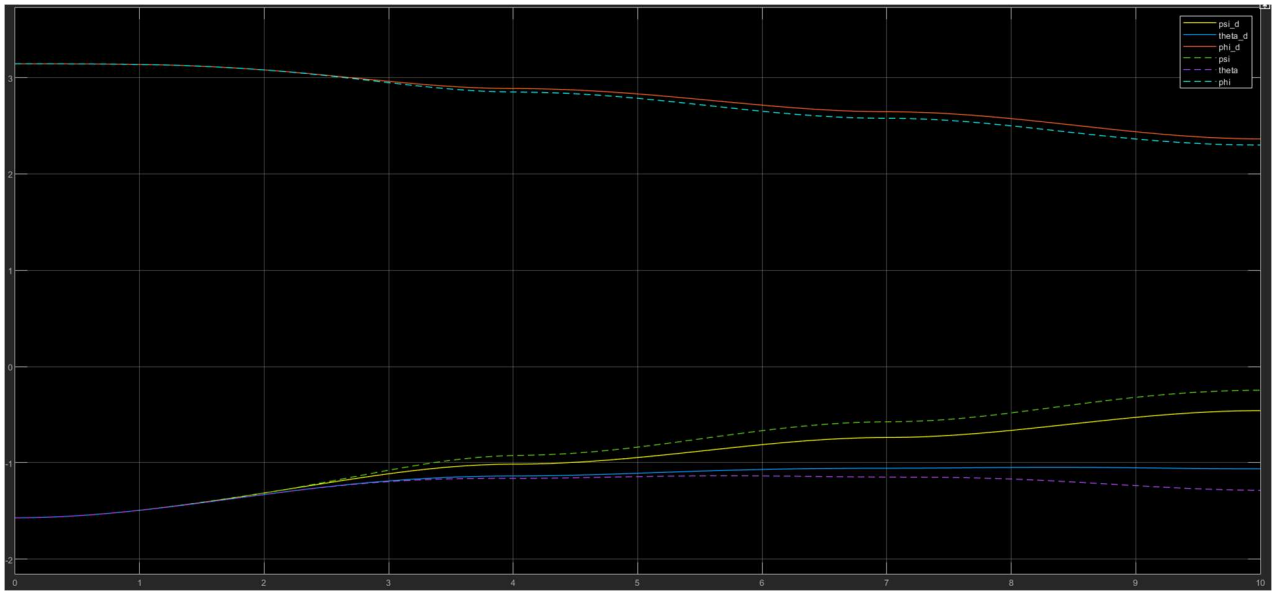


Figure 44 Impedance control position ϕ_e graph.

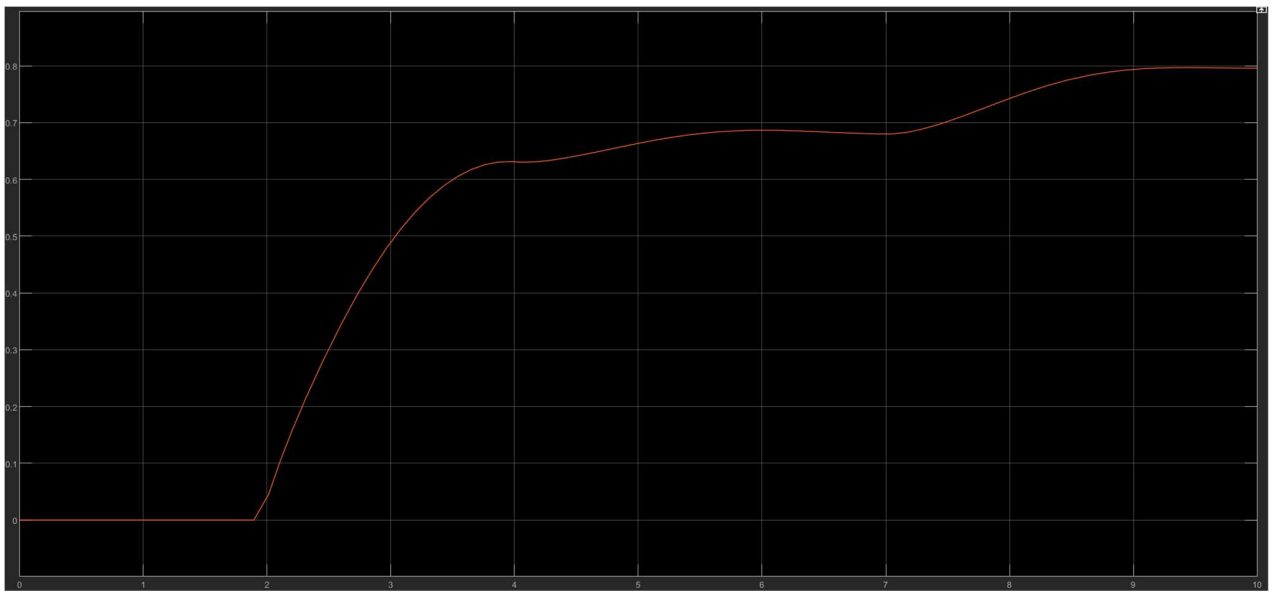


Figure 45 External wrench h_e in impedance control.

to the environment changes and accept the disturbances from the environment. The controller displayed in impedance controller except the block of the admittance controller is added to separate the motion controller with the impedance controller. In admittance controller, the compliance frame $\sum t$ is added to determine the ideal behavior of the end-effector in impedance controller. The mechanical impedance take form of

$$h_e^d = M_t \ddot{\tilde{z}} + K_{dt} \dot{\tilde{z}} + K_{pt} \tilde{z} \quad (76)$$

Where h_e^d is the desired force measure in the desired frame. $\mathrm{M_t}$, $\mathrm{K_{dt}}$, and $\mathrm{K_{pt}}$ are the mechanical impedance parameters. $\tilde{z}$ is the error between the compliant frame and the desired frame.

$$\tilde{z} = x_d - x_t \quad (77)$$

Figure 46 shows the admittance control schema in Simulink. The schema resembled to the impedance controller schema just the addition of admittance block is changed. Figure 47 introduce the admittance control block in which the desired position and orientation is converted to the compliant frame using the external interaction force. However, in this experiment, following parameters are used in the control law

$$K_p = \begin{bmatrix} 790 & 0 & 0 & 0 & 0 & 0 \\ 0 & 790 & 0 & 0 & 0 & 0 \\ 0 & 0 & 790 & 0 & 0 & 0 \\ 0 & 0 & 0 & 70 & 0 & 0 \\ 0 & 0 & 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 70 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 170 & 0 & 0 & 0 & 0 & 0 \\ 0 & 170 & 0 & 0 & 0 & 0 \\ 0 & 0 & 170 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}$$

$$K_e = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 190 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_d = \begin{bmatrix} 0.12 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

$$K_{pt} = \begin{bmatrix} 350 & 0 & 0 & 0 & 0 & 0 \\ 0 & 350 & 0 & 0 & 0 & 0 \\ 0 & 0 & 350 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}$$

$$K_{dt} = \begin{bmatrix} 125 & 0 & 0 & 0 & 0 & 0 \\ 0 & 125 & 0 & 0 & 0 & 0 \\ 0 & 0 & 125 & 0 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15 \end{bmatrix}$$

$$M_{dt} = \begin{bmatrix} 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

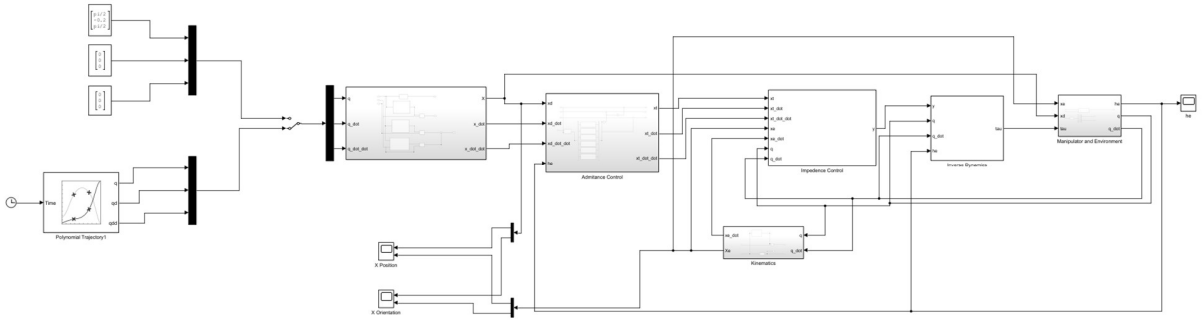


Figure 46 Admittance controller schema in Simulink.

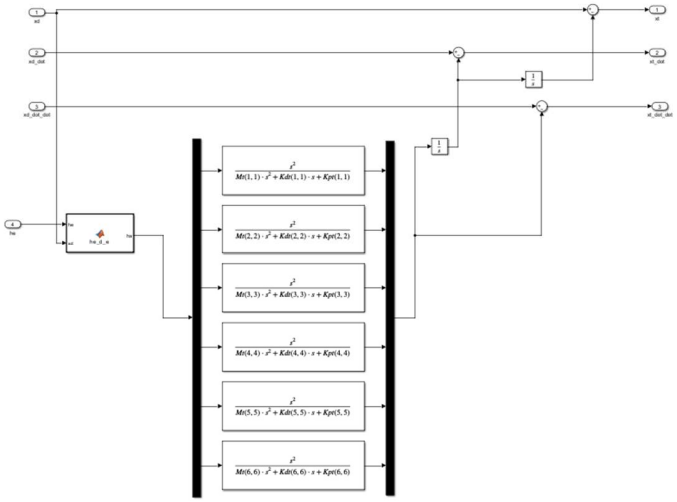


Figure 47 Admittance controller block.

The Figure 48 and 49 shows the position and orientation graph of the admittance controller. It can be seen that the admittance controller do not converge to the desired position due the environment present at the z-axis. But it can be seen that the admittance controller is more adaptive to the interaction force then the compliance controller.

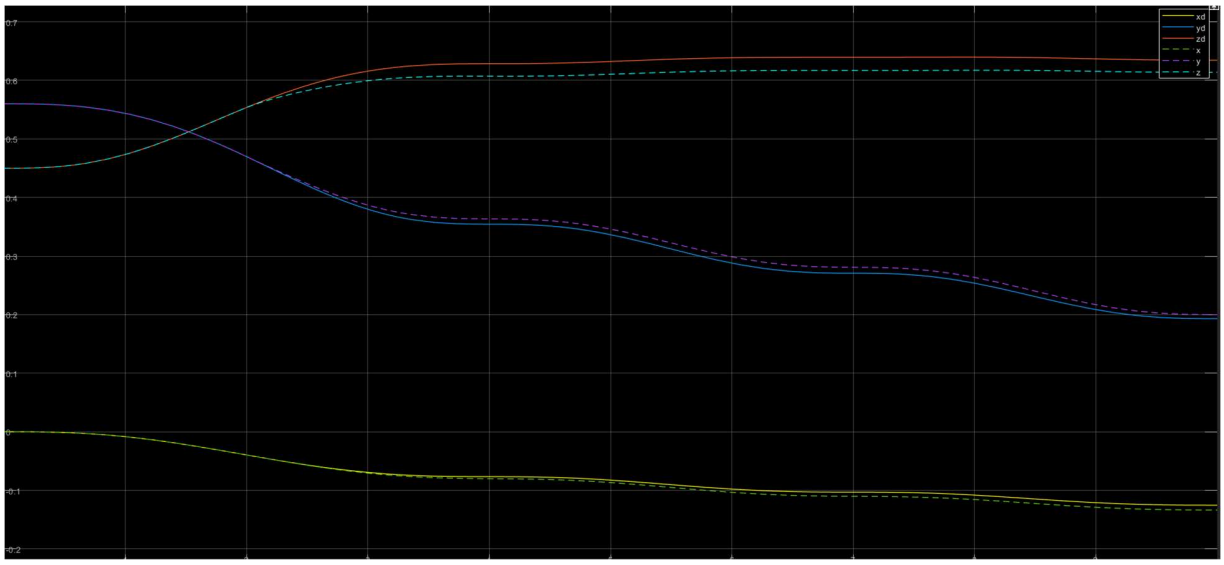


Figure 48 Admittance control position P_e graph.

Figure 50 shows the environment interaction with the end-effector. As the admittance controller has the property of the adaptation of the environment, the interaction with the environment is smooth and elastic due to the parameters of the admittance controller described above.

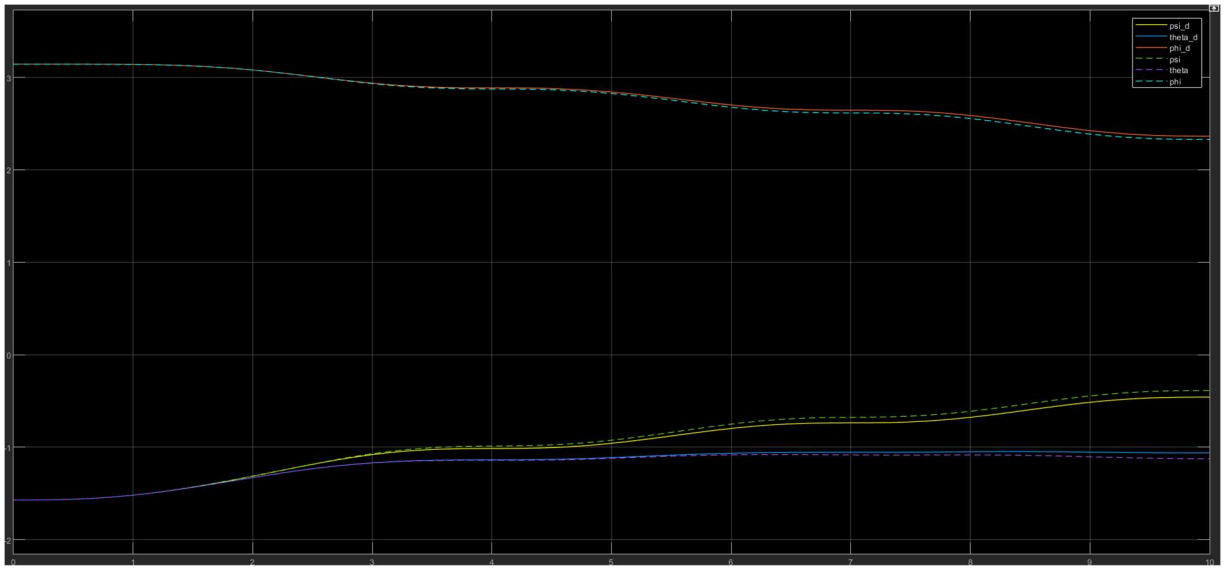


Figure 49 Admittance control orientation ϕ_e graph.

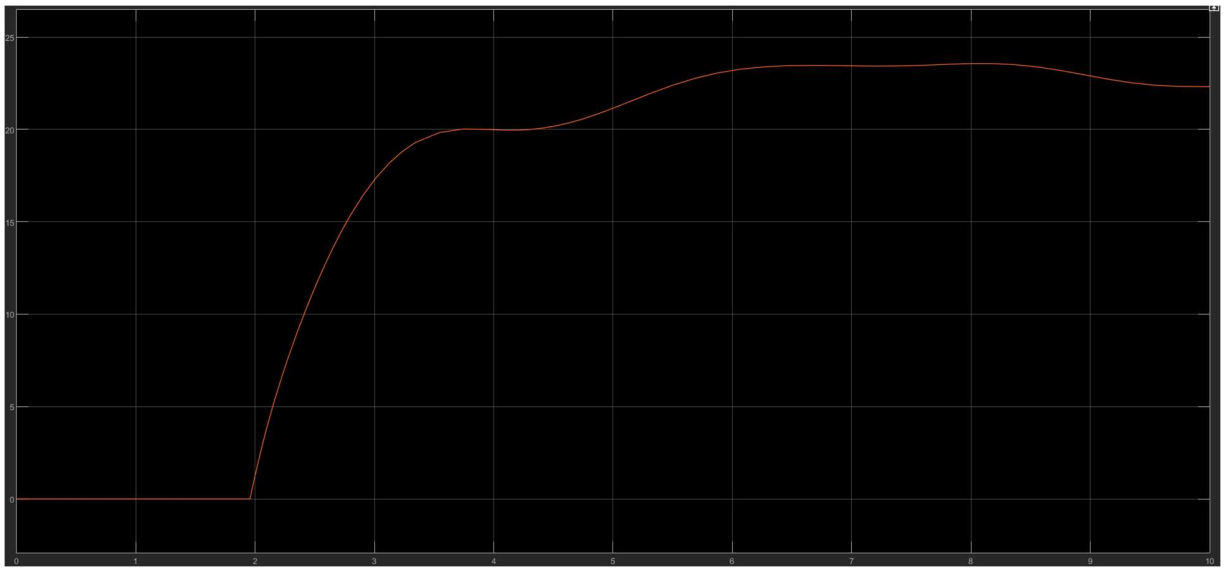


Figure 50 External wrench h_e of Admittance control.

11.2. Direct Force Control

A direct force controller is a type of force control strategy used in robotics to regulate the contact force between the robot and the environment in a direct manner. It is a modification of the motion control strategy where an outer feedback loop is added to the system, allowing for force regulation in addition to position and velocity regulation. In a direct force control scheme, the desired contact force is specified as the control input, and the force error between the desired force and the measured force is used to compute the control output. The control output is then used to update the joint positions, velocities, and accelerations to achieve the desired contact force. The control gains for the force loop are chosen such that the system is stable, and the

desired force is achieved in a timely and accurate manner. In this literature, two force controllers are used that is described below.

11.2.1. Force control

Force controller is the direct force controller method that uses the force feedback loop to control the desired force f_d. The controller used operational space to define all the forces because forces are naturally defined in operational space. However, the inverse dynamic controller defined in Figure 42 is used in the force controller. The environment is used as an elastic system as explained in above sections, that is

$$f_e = K_e(x_e - x_r) \quad (78)$$

where x_e is the position of end-effector and x_r is the rest position of the environment. It is important to note that to make the control simple, the orientation part of the position is neglected. So the analytical Jacobian became same as geometrical Jacobian.

$$J_a(q) = J(q) \quad (79)$$

so force controller ends up in the following control law

$$\tau = B(q)y + n(q, \dot{q}) + J^T(q)f_e \quad (80)$$

where

$$y = J^{-1}(q)M_d^{-1}(-K_d\dot{x}_e + K_p(x_f - x_e) - M_d\dot{J}(q, \dot{q})\dot{q}) \quad (81)$$

and x_f is the suitable reference position that needs to be related to the force error. However, x_f is defined as

$$x_f = C_F(f_d - f_e) \quad (82)$$

hence C_F is defined as a PI controller

$$C_F = K_f + K_i\frac{1}{s} \quad (83)$$

and f_d is the desired force. It is important to note that if the f_d is out of the Image(K) then there will be a drag in the position of the end-effector. For this controller following parameters are used

$$K_p = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 25 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & 25 \end{bmatrix}$$

$$M_d = \begin{bmatrix} 0.12 & 0 & 0 \\ 0 & 0.12 & 0 \\ 0 & 0 & 0.12 \end{bmatrix}$$

$$K_f = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

$$K_e = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

Moreover, in this experiment there are 2 types of desired forces i.e. $f_{d1}$and f_{d2} which is inside of Image(K) and outside of Image(k) respectively.

$$f_{d1} = [0,0,3]$$

$$f_{d2} = [0,0,6]$$

The Figure 51 shows the overall schema of the force controller, in which the outer force feedback loop can be seen. The motion control and inverse dynamics block is same as impedance control block defined in the Figure 41 and 42. The force control block which convert the force to its respective motion is defined in the Figure 52. The figure shows the PI control schema for the x_f.

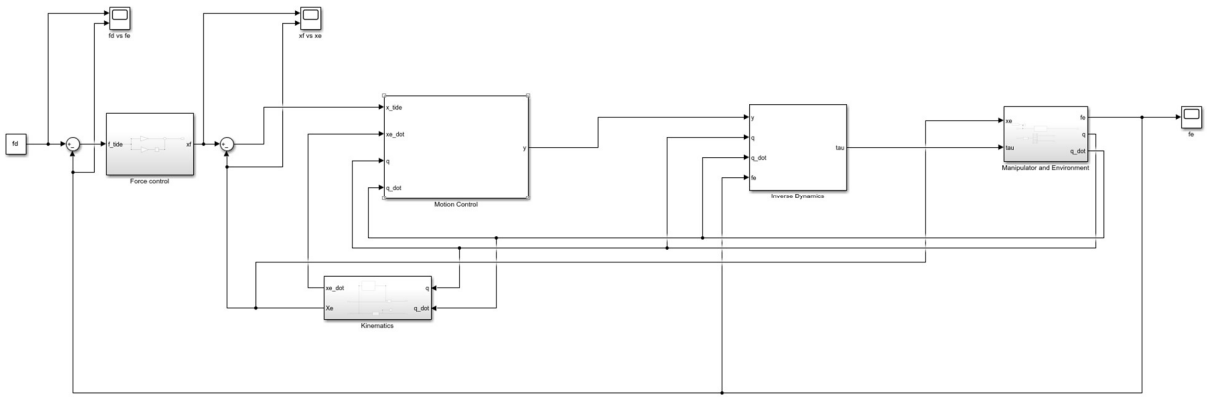


Figure 51 Force control schema in Simulink.

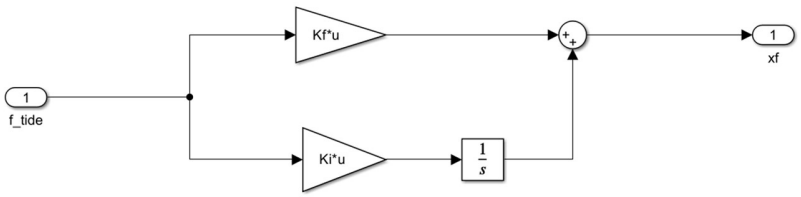


Figure 52 Force control block.

Using the above defined parameters force control, the control shows good results. As it can be seen that the force f_e converges to the desired force f_d accurately as shown in Figure 53. It is important to note that only desired force along z-axis is set otherwise all other are set to zero for sake of the simplicity.

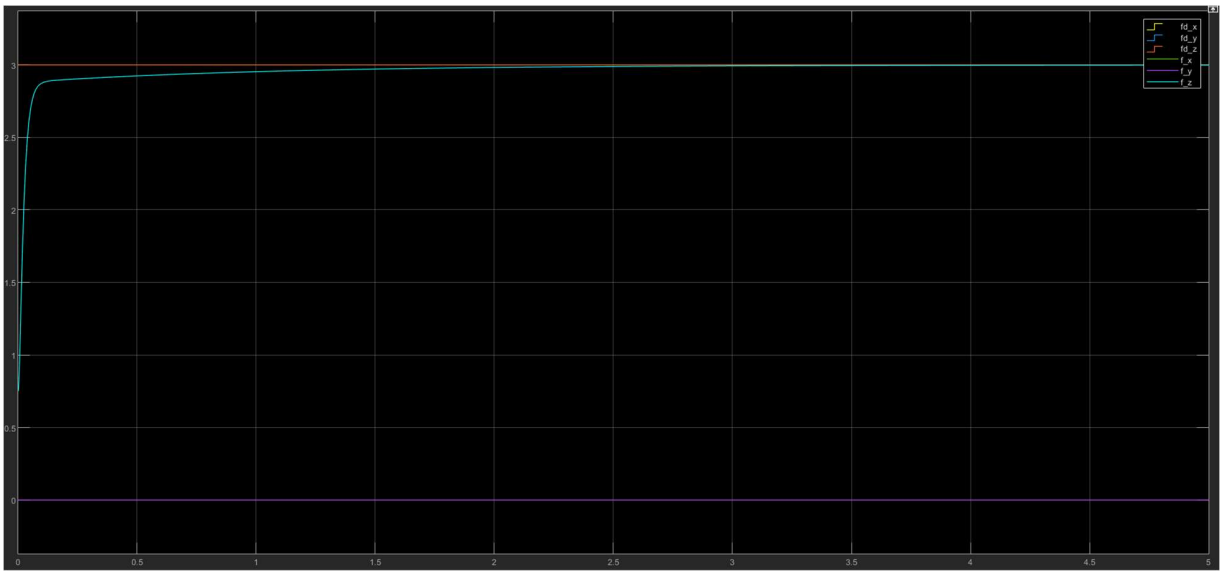


Figure 53 Control schema graph for f_{d1} and f_e.

According to theory, it is important to note that if the desired force is set inside the Image(K_e) the controller will reach to the relative position x_f. As it can be seen in the Figure 54, the K_e is set inside the Image(K_e), the position of end-effector x_e converges to the position x_f accurately. However, in Figure 55, there is a little bit of the drag because of the f_d is set out of the Image(K_e).

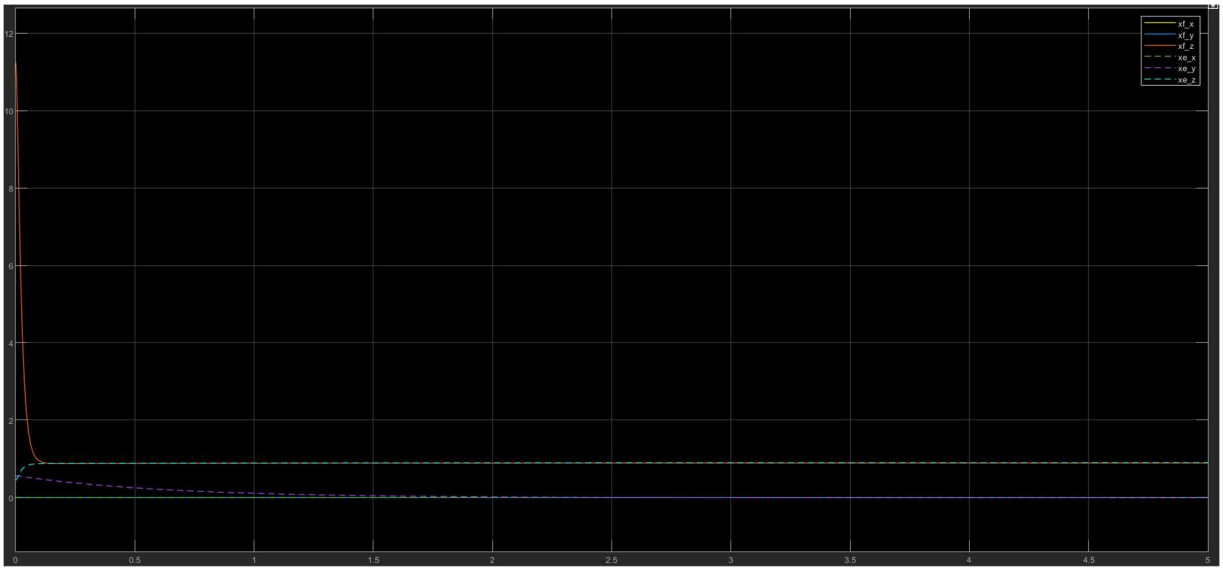


Figure 54 Force control graph between x_f and x_e for $f_{d1} = 3\ inside\ Image(K_e)$.

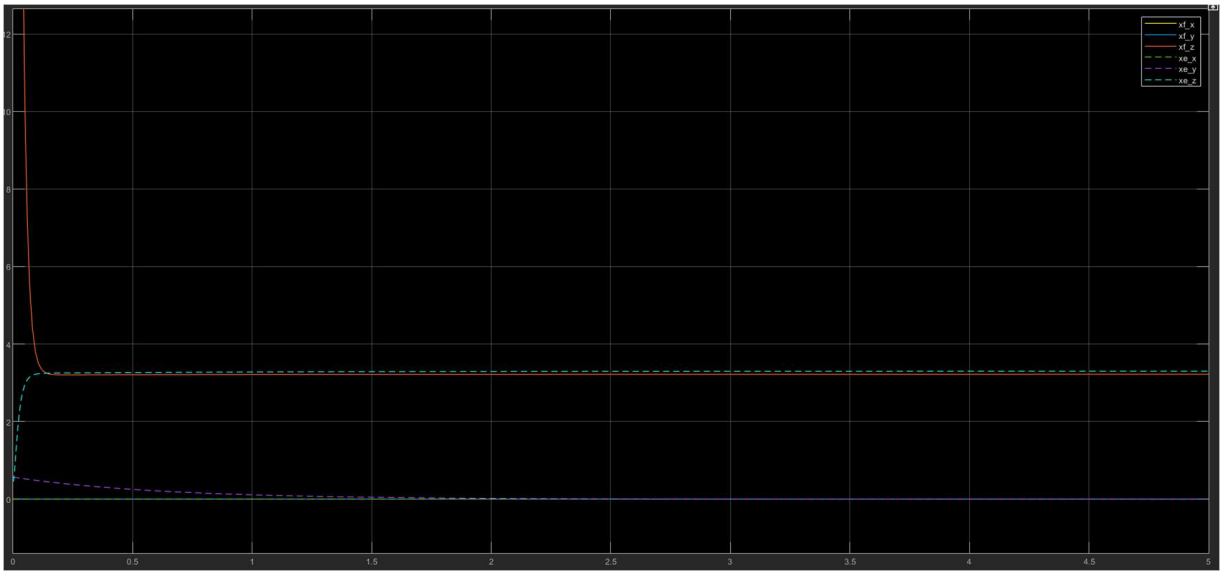


Figure 55 Force control graph between x_f and x_e for $f_{d2} = 6$ outside $Image(K_e)$.

11.2.2. Parallel force/position controller

Parallel force and position controller is the successor of the force control defined above. As there is addition of the desired position in the position control loop. So the controller can reach to the desired force as well as desired position. The desired position x_d is reached by the end-effector only if there is free motion in that direction. The control law for the parallel force and position controller explains

$$y = J^{-1}(q)M_d^{-1}(-K_d\dot{x}_e + K_p(x_f - x_e + x_d) - M_d\dot{J}(q,\dot{q})\dot{q}) \quad (84)$$

The controller uses following parameters

$$K_p = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 200 & 0 \\ 0 & 0 & 200 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}$$

$$M_d = \begin{bmatrix} 0.12 & 0 & 0 \\ 0 & 0.12 & 0 \\ 0 & 0 & 0.12 \end{bmatrix}$$

$$K_f = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 50 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}$$

$$K_e = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

$$f_d = [0,0,3]$$

It is important to note that the x_d is defined by the forward kinematics as explain in the operational space controller.

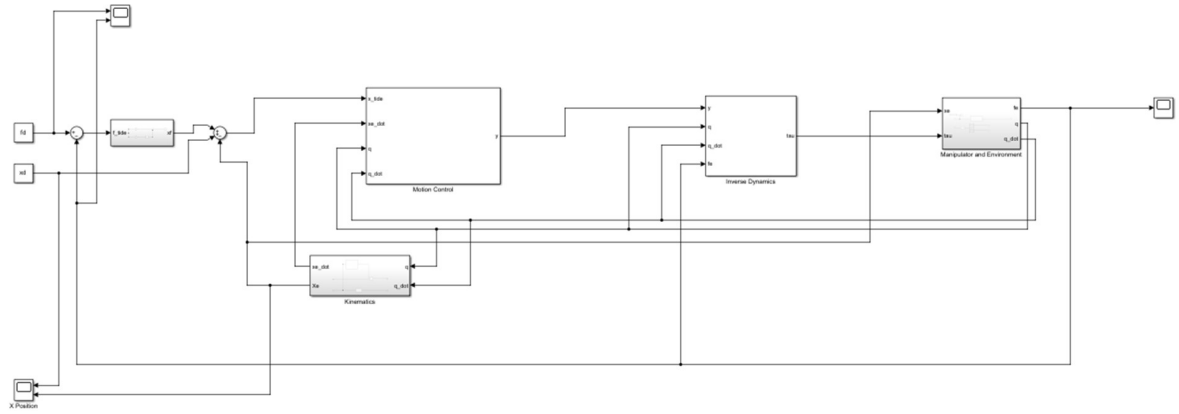


Figure 56 Parallel force/position control schema in Simulink.

Figure 57 shows that, the controller reach to its optimal force accurately. It is important to note that the environment is acting only in z-axis, all other axis are set to zero. Furthermore, the Figure 58 shows that the position is achieved accurately as well except the z-position because its motion is constrained and can achieved by the controller as per described above.

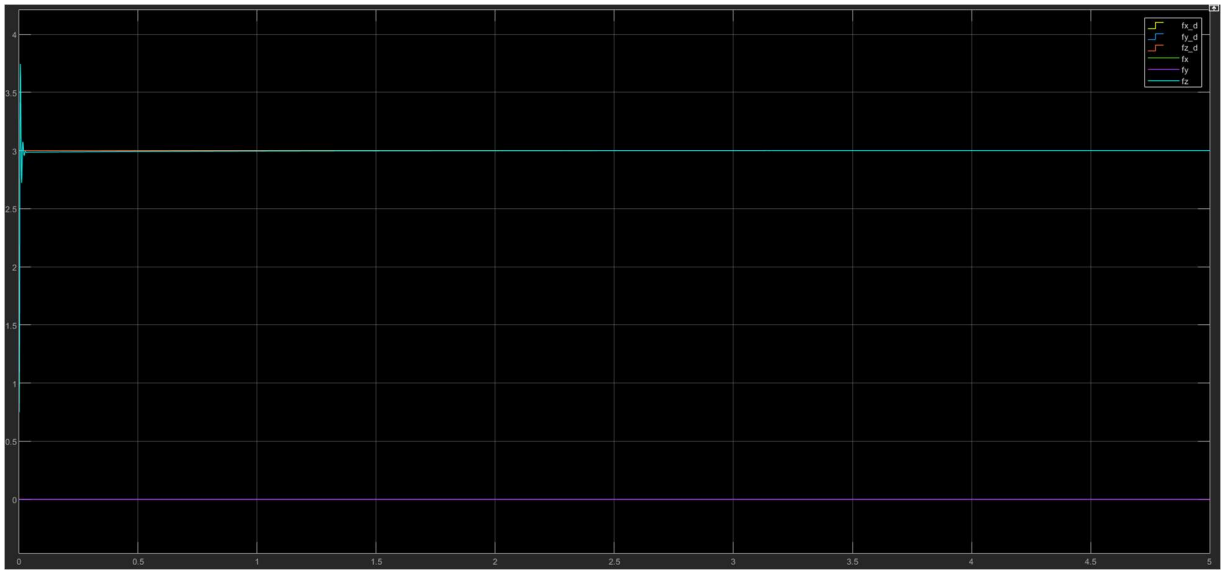


Figure 57 Force graph of parallel force and motion controller.

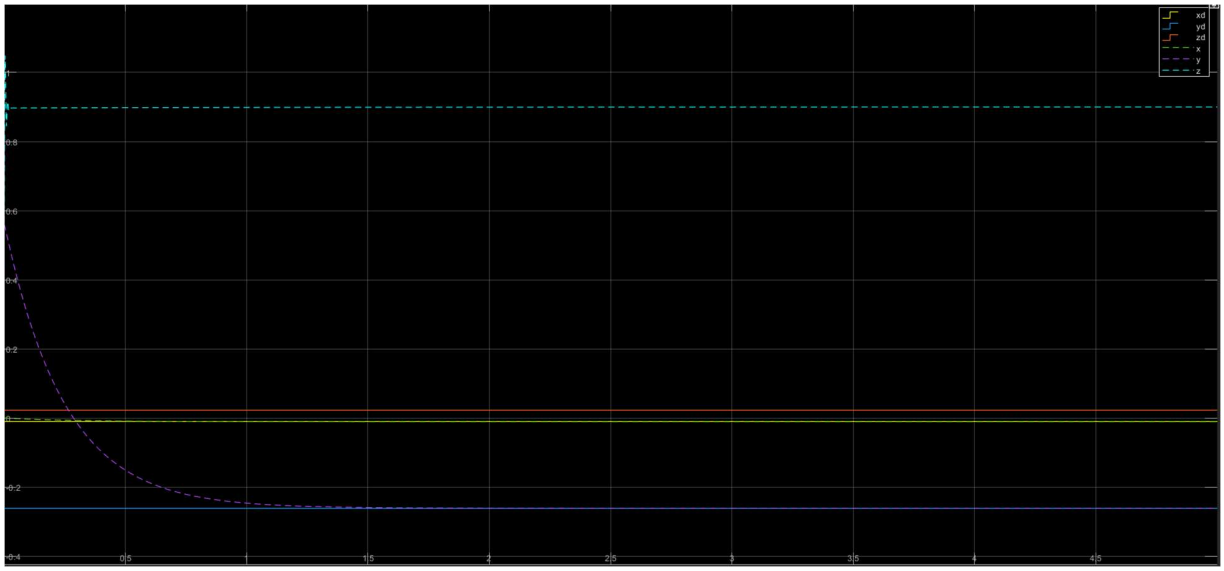


Figure 58 Position of the controller.

12. Conclusion

To conclude the literature, a fixed-base robot with three degrees of freedom, comprising two revolute joints and one prismatic joint. The robot was modeled using the Denavit-Hartenberg convention, and its kinematics, differential kinematics, dynamics, and control were studied using MATLAB tools. The forward and inverse kinematics of the robot were obtained using transformation matrices and symbolic equations, and their accuracy was cross-checked with the MATLAB Robotic Toolkit. The geometrical and analytical Jacobians were derived and utilized to analyze the velocities of the end-effector in the differential kinematics section, producing accurate results as per the Robotic Toolbox.

To obtain the dynamics of the robot, both Lagrange and Newton-Euler formulations were used and compared for accuracy, yielding similar results. Different controllers were developed for the robot in both joint space and operational space using a range of techniques, such as PD control, inverse dynamics control, adaptive control, force control, compliance control, impedance control, admittance control, and parallel force/position control. The performance of each controller was evaluated using MATLAB Simulink simulations for various tasks, including tracking a desired trajectory or applying a desired force. Additionally, the impact of disturbances, efficiency, and accuracy of different controllers were evaluated.

Overall, this literature provides a comprehensive study of the 3 DoF robot manipulator using different modeling and control techniques, offering insights into the performance of various controllers for different tasks and disturbances. The use of MATLAB tools and Robotics Toolbox has allowed for accurate analysis and cross-checking, demonstrating the effectiveness of these tools in robotics research.

The inverse dynamic controller proved well if there is a disturbance in the desired solution, while the adaptive controller proved itself when there is no accurate dynamic model of robot is present. Furthermore, the operational space control proved to be more natural and easy to use then the joint space controllers. Indirect force controllers are very complex and hard to implement while the direct force controllers are very intuitive and easy to implement.

References

[1] Z. Li, S. Li, and X. Luo, 'An Overview of Calibration Technology of Industrial Robots', *IEEECAA J. Autom. Sin.*, vol. 8, no. 1, pp. 23–36, 2021, doi: 10.1109/JAS.2020.1003381.

[2] E. Madsen, O. S. Rosenlund, D. Brandt, and X. Zhang, 'Comprehensive modeling and identification of nonlinear joint dynamics for collaborative industrial robot manipulators', *Control Eng. Pract.*, vol. 101, p. 104462, Aug. 2020, doi: 10.1016/j.conengprac.2020.104462.

[3] C. Sabnis, N. Anjana, A. Talli, and A. C. Giriyapur, 'Modelling and Simulation of Industrial Robot Using SolidWorks', in *Advances in Industrial Machines and Mechanisms*, Singapore, 2021, pp. 173–182. doi: 10.1007/978-981-16-1769-0_16.

[4] T. Yaren and S. Küçük, 'Dynamic modeling of 3-DOF RRP type serial robotic manipulator using Lagrange-Euler method', in *International Marmara Sciences Congress (Autumn)*, 2019.

[5] J. Luo *et al.*, 'Reinforcement learning on variable impedance controller for high-precision robotic assembly', in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3080–3087.

[6] V. G. Pratheep, M. Chinnathambi, E. B. Priyanka, P. Ponmurugan, and P. Thiagarajan, 'Design and Analysis of six DOF Robotic Manipulator', *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1055, no. 1, p. 012005, Feb. 2021, doi: 10.1088/1757-899X/1055/1/012005.

[7] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Eds., 'Kinematics', in *Robotics: Modelling, Planning and Control*, London: Springer, 2009, pp. 39–103. doi: 10.1007/978-1-84628-642-1_2.

[8] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Eds., 'Differential Kinematics and Statics', in *Robotics: Modelling, Planning and Control*, London: Springer, 2009, pp. 105–160. doi: 10.1007/978-1-84628-642-1_3.

[9] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Eds., 'Dynamics', in *Robotics: Modelling, Planning and Control*, London: Springer, 2009, pp. 247–302. doi: 10.1007/978-1-84628-642-1_7.

[10] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Eds., 'Motion Control', in *Robotics: Modelling, Planning and Control*, London: Springer, 2009, pp. 303–361. doi: 10.1007/978-1-84628-642-1_8.

[11] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Eds., 'Force Control', in *Robotics: Modelling, Planning and Control*, London: Springer, 2009, pp. 363–405. doi: 10.1007/978-1-84628-642-1_9.