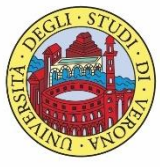
COMPUTER ENGINEERING FOR ROBOTICS AND SMART INDUSTRY

Advance Control Systems

Session 2023/2024

Author: Syed Jawwad Mehdi Rizvi (VR497161)
Email: syedjawwadmehdi.rizvi@studenti.unir.it



UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Contents

RPR - Robotic Manipulator Dynamics and Controls

Introduction

The study of robot dynamics is crucial for understanding the mobility and behavior of robotic systems. This introduction aims to provide an overview of a three-degree-of-freedom robot equipped with prismatic, revolute, and revolute joints. **Our objective is to analyze the dynamics of this robot using both the Newton-Euler and Lagrange equations.**

A mechanical system having three independent motions is represented by the three-degree-of-freedom robot. Each type of joint contributes to a specific movement. The revolute joint provides rotational motion, allowing the robot to spin around an axis. The prismatic joint enables linear motion, allowing the robot to move in a straight line. Another prismatic joint provides an additional linear degree of freedom, completing the chain of movements.

The dynamics of the robot manipulator include the **gravitational, Coriolis, and inertia matrices** derived from the Newton-Euler approach or the Lagrange approach. To enable the robot to perform various tasks, a controller is needed to manage the robot's actions. Firstly, it is crucial 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 the position and orientation of the end-effector. Alternatively, the operational space can be used instead of joint space. The operational space is more intuitive as it is defined by Cartesian coordinates from the base frame. The controller helps the robot reach desired positions and achieve desired trajectories.

As defined, robots also need to interact with the environment, so a force controller is required to manage the robot's force. The force controller helps the robot avoid damaging itself or the environment.

We hope to learn important facts about the robot's potential and constraints from this study. The findings could benefit various fields, including manufacturing, automation, medical robotics, and more. Ultimately, a thorough understanding of its dynamics will allow us to maximize the robot's potential and improve its performance in real-world scenarios.

RPP - Robot

The 3 DoF RPP robot is the one I am working with. The robot contains one revolute joint that can rotate in each direction, specifically along the z-axis, and two prismatic joints, the base frame is situated at the bottom left corner of the illustration. The end effector (ee) frame, on the other hand, is established with reference to the same base frame, with the z-axis pointing out of the end-effector.

The following table defines the Denavit-Hartenberg (DH) parameters for the RPP 3 DoF robot manipulator.

	a	**α**	***θ***	**d**
Base – Link 1	0	pi/2	0	a1
Link 1 – Link 2	a2	0	q1	0
Link 2 – Link 3	0	pi/2	pi/2	a3+q2
Link 3 – Link 4	0	pi/2	- pi/2	a4+q3

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.

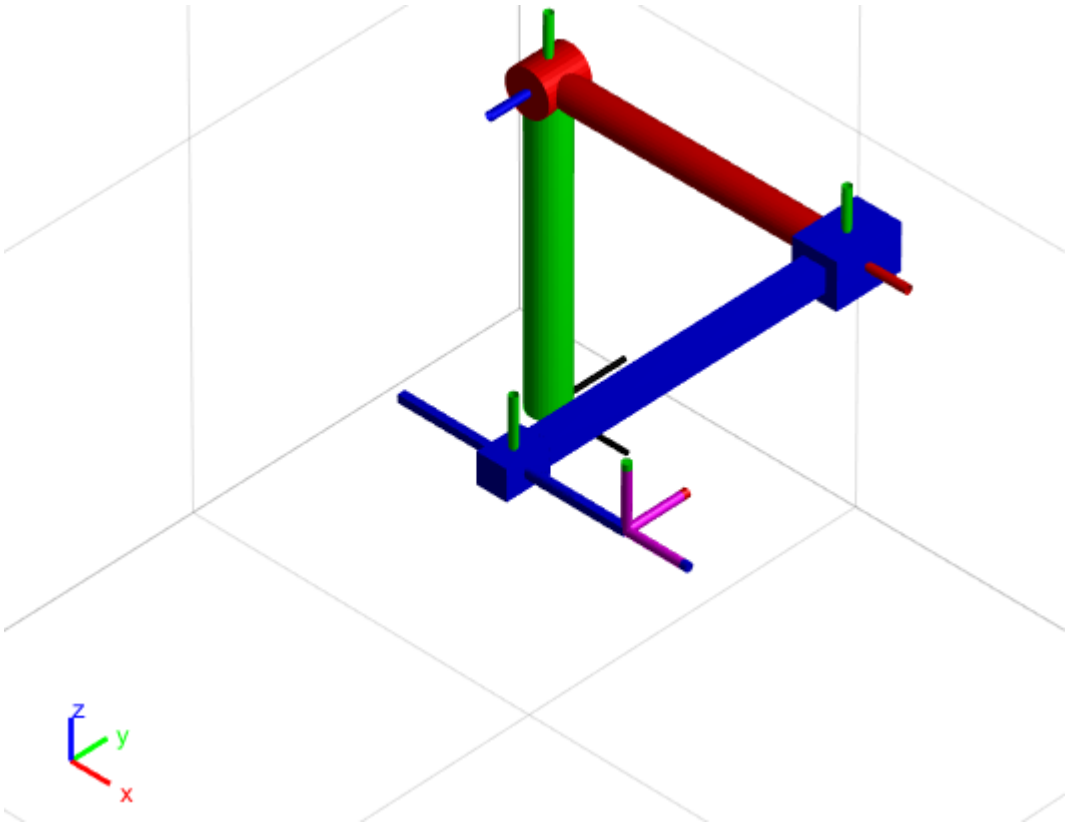


Figure 1 (RPP robotic arm manipulator URFD diagram

Kinematics

A robotic manipulator's forward and inverse kinematics can be used to represent its kinematics. The relationship between the robot's joint angles and the location and orientation of the end-effector in the workspace is referred to as forward kinematics. On the other hand, inverse kinematics describes the relationship between the end-effector's desired position and orientation and the joint angles necessary to attain those positions and orientations. The kinematics of robotic manipulators are modeled using the Denavit-Hartenberg (DH) standard. The position and orientation of each joint are determined by DH parameters. The forward kinematics of a robotic manipulator can be determined by applying the DH convention and a transformation matrix.

Forward Kinematics

Establishing a connection between joint angles and the end-effector position is essential. The forward kinematics solution, which allows the robot's end-effector to move to a specified location within its workspace, is a critical step in controlling the robot's movements. In this study, the homogeneous transformation matrix is derived using the DH table to solve the forward kinematics problem.

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

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

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

$$H_4^0 = H_1^0 H_2^1 H_3^2 H_4^3$$

Where we have,

$$H_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

H_2^1 :

```
[cos(q1), -sin(q1), 0, (2*cos(q1))/5]
[sin(q1),  cos(q1), 0, (2*sin(q1))/5]
[      0,        0, 1,             0]
[      0,        0, 0,             1]
```

H_3^2 :

```
[0, 0, 1,          0]
[1, 0, 0,          0]
[0, 1, 0, q2 + 23/50]
[0, 0, 0,          1]
```

H_4^3 :

```
[ 0, 1, 0,          0]
[-1, 0, 0,          0]
[ 0, 0, 1, q3 + 3/20]
[ 0, 0, 0,          1]
```

Here we have the final matrix after the multiplication.

H_4^0 :

```
[0, -sin(q1), cos(q1),       (2*cos(q1))/5 + cos(q1)*(q3 + 3/20)]
[1,        0,       0,                                - q2 - 23/50]
[0,  cos(q1), sin(q1), (2*sin(q1))/5 + sin(q1)*(q3 + 3/20) + 2/5]
[0,        0,       0,                                          1]
```

By substituting the values of q1, q2 and q3 to 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. The following are the results obtained from substituting the values.

q1 = pi/3,

q2 = 0.1,

q3 = 0.1,

Direct Kinematics:

```
     0   -0.8660    0.5000    0.3250
1.0000         0         0   -0.5600
     0    0.5000    0.8660    0.9629
     0         0         0    1.0000
```

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

```
    0.0004   -0.8660    0.5000    0.3250
    1.0000    0.0004   -0.0001   -0.5598
   -0.0001    0.5000    0.8660    0.9634
         0         0         0    1.0000
```

Inverse Kinematics

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 final matrix, we can analytically extract the q1, q2 and q3.

```
Px = [cos(q1)*(a4+q3)+a2*cos(q1)];
Py = [-a3-q2];
Pz = [a1+sin(q1)*(a4+q3)+a2*sin(q1)];
```

Now by manipulating the position equations we found our parameters as;

q1 = 1.0472

q2 = 0.1000

q3 = 0.1000

and these values are equal to the values which we set initially.

Differential Kinematics

Differential kinematics plays a crucial role in understanding how joint velocities relate to the linear and angular velocities of the end-effector. The configuration of the manipulator affects the Jacobian matrix, which represents this interaction. Given the joint velocity inputs, this matrix is used to determine the robot's end-effector velocity. Analytical and geometrical Jacobian matrices are the two forms of Jacobian matrices utilized in differential kinematics. The analytical Jacobian can be obtained from the geometrical Jacobian or through partial derivatives of the forward kinematics equations with respect to joint variables. Additionally, the homogeneous matrices can be concatenated to derive the geometrical Jacobian.

Geometrical Jacobian

Considers the end-effector's orientation and how the joint velocities will affect it. In a coordinate frame fixed at the robot's base, it makes use of the connection between the joint velocity and the appropriate end-effector velocity. The translational and rotational Jacobian matrices can be combined to create the geometrical Jacobian matrix. The 6x3 matrix, which incorporates the translational and angular velocities of the robotic manipulator, is the geometrical Jacobian for the 3 DoF robot. As follows is the generalised Jacobian matrix:

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

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 generalized formula is used.

$$\begin{bmatrix} Jp_i \\ Jo_i \end{bmatrix}$$

$$\begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \; for\ prismatic\ joint$$

$$\begin{bmatrix} z_{i-1} \times (P_e - P_{i-1}) \\ z_{i-1} \end{bmatrix} \; for\ revolute\ joint$$

The following numerical geometrical Jacobian matrix is obtained for the given robotic manipulator.

Geometrical Jacobian Calculated by me:

0	0	0
-1.0000	0	0
0	0	0
-0.5629	0	0.5000
0	-1.0000	0
0.3250	0	0.8660

Geometrical Jacobian Toolkit:

-0.0000	0	0
-1.0000	0	0
0.0008	0	0
-0.5629	-0.0000	0.5000
0.0003	-1.0000	-0.0001
0.3250	0.0008	0.8660

As we can match both results, the error rate is very minimum.

Analytical Jacobian

The relationship with the geometrical Jacobian can be used to obtain the analytical Jacobian, as can partial derivatives of the forward kinematics equations with respect to the joint variables. In simpler terms, the rate of change of end-effector position and orientation with respect to joint velocities can be used to determine the analytical Jacobian.

Analytical Jacobian by me:

0	0	0
1.0000	0	0
0	0	0
-0.5716	0	0.5000
0	-1.0000	0
0.3300	0	0.8660

As it can be observed that the analytical jacobian is exactly same as the geometrical Jacobian linear velocity.

Dynamics

The forward dynamics and the inverse dynamics of the manipulator can be separated into two types. In the forward dynamics problem, the manipulator's motion must be determined considering the forces and torques acting on it. It entails resolving the motion equations that express the manipulator's behavior in terms of location, velocity, and acceleration. The inverse dynamics problem, on the other hand, includes figuring out the forces and torques necessary to get the manipulator to move at the desired speed, acceleration, and position.

Both the Lagrange formulation and the Newton-Euler formulation are used to examine the dynamics of a three-dimensional RPR robotic manipulator. The results from both formulations are compared to guaranteeing the dynamic model's accuracy. The dynamics of the manipulator are estimated from the first joint, referred to as "Frame 1," rather than from the base frame, referred to as "Frame 0," as the base frame is fixed and immovable, allowing the dynamics of the base frame to be disregarded.

Lagrange formulation

The kinetic and potential energy of the manipulator must be determined for the Lagrange formulation. The inertia matrix, Coriolis and centrifugal forces, and gravitational forces of the robotic manipulator are all calculated throughout this process. That was stated in the Lagrange Formulation.

$$L = T - U$$

Where T and U are the kinetic and potential energy of the 3 DoF RPP robotic manipulator, respectively.

Kinetic Energy

By adding the kinetic energies of all the system's links, the kinetic energy is determined. This contains the kinetic energy of each link's rotation and translation. The mass of the link and its linear velocity determine the translational kinetic energy, whereas the moment of inertia of the link and its angular velocity determine the rotational kinetic energy. As previously mentioned, the total kinetic energy is

$$T = \sum_{i=1}^{3} T_i$$

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

$$T(q, \dot{q}) = \dot{q}\, TB(q)\, \dot{q}$$

With (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 RPP robotic manipulator it is referred to θ_1, d_1, and d_2. To calculate the (q) matrix following formulation can be used.

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

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.

$$J^i = \begin{bmatrix} J_p^i \\ J_o^i \end{bmatrix}$$

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

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

$$\begin{bmatrix} J_{p_j}^i \\ J_{o_j}^i \end{bmatrix}$$

$$\begin{bmatrix} z_{j-1} \\ 0 \end{bmatrix} \; for\ prismatic\ joint$$

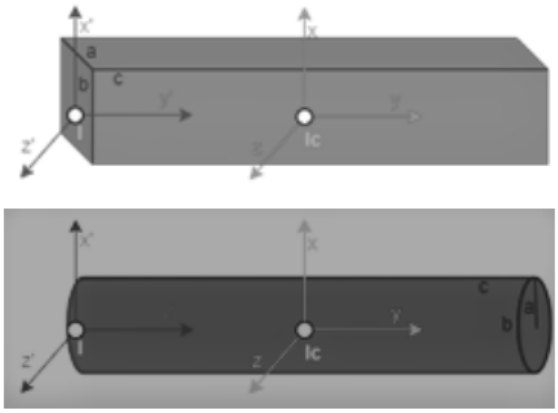
$$\begin{bmatrix} z_{j-1} \times (P_{l_i} - P_{j-1}) \\ z_{j-1} \end{bmatrix} \; for\ revolute\ joint$$

$$P_{l_i} = \mathrm{R}_\mathrm{i}^1 (\mathrm{CoM})^\mathrm{T} + P_i$$

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 RPP robot expressed in Figure 1.

```
% C.O.M
pL1 = [-l1/2; 0; 0];
pL2 = [0; 0; -l2/2];
pL3 = [0; 0; -l3/4];
```

Back to the Equation, 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.



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

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

Potential Energy

The location of the manipulator and its end-effector in relation to gravity is often connected to potential energy. The manipulator's potential energy is inversely correlated with its height above a reference point. The potential energy increases with the height of the manipulator. Calculating the potential energy is simple and may be expressed as

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

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

$$g_1 = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix}$$

Where g is the gravity constant of which value is 9.8 m/s^2.

Back to the Lagrange formulation, after finding the (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.

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

According to the Equation above, 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 (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 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

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

The gravity matrix calculated is $\begin{bmatrix} 2.3767 \\ 0 \\ 0.3439 \end{bmatrix}$.

The potential energy using gravity formulation can be achieved by

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

It is important to note that, to check the correctness of Equation above, 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 (q) matrix will be symmetric and positive definite. By computing the (q) matrix numerically we can ensure the correctness of $B(q)$ matrix.

$$\text{Inertia Matrix} = \begin{bmatrix} 0.3425 & 0 & 0 \\ 0 & 1.1583 & 0 \\ 0 & 0 & 0.0405 \end{bmatrix}$$

And with the help of cholesky decomposition, it is confirmed that the inertia matrix calculated is symmetrically positive definite.

The langrangian Equation calculated is as follows:

$$\begin{bmatrix} 2.3767 \\ 0 \\ 0.3439 \end{bmatrix}$$

Newton–Euler Formulation

By iteratively calculating the forces and torques at each joint, the Newton-Euler formulation is a technique for computing the dynamics of a robotic manipulator. It entails determining the linear and angular velocities and accelerations of the manipulator as well as the forces and torques operating on it to derive the equations of motion for the system. Using the rules of Newton and Euler to calculate the forces and torques at each joint, the procedure begins with the manipulator's ideal frame and moves toward the end-effector. Recursive algorithms can

be used to compute the Newton-Euler formation. The findings, however, are not very comprehensible. Newton-Euler formulation can be determined using the following pseudocode.

Newton Euler Formulation: Forward Equation

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

1 for i = 1 to n:

2 $R_i^{i-1} = R_i^{i-1}(q_i)$

3 $\omega_i^i = \left(R_i^{i-1}\right)^T \omega_{i-1}^{i-1}$

4 $\dot{\omega}_i^i = \left(R_i^{i-1}\right)^T \dot{\omega}_{i-1}^{i-1}$

5 if joint == REVOLUTE:

6 $\omega_i^i = \omega_i^i + \left(R_i^{i-1}\right)^T \dot{q}_i z_0$

7 $\dot{\omega}_i^i = \dot{\omega}_i^i + \left(R_i^{i-1}\right)^T (q_i z_0 + \dot{q}_i \omega_{i-1}^{i-1} \times z_0)$

8 $\ddot{P}_i = \left(R_i^{i-1}\right)^T \ddot{P}_{i-1}^{i-1} + \dot{\omega}_i^i \times r_{i-1,i}^i + \omega_i^i \times (\omega_i^i \times r_{i-1,i}^i)$

9 if joint == PRISMATIC:

10 $\ddot{P}_i = \ddot{P}_i + \left(R_i^{i-1}\right)^T \ddot{q}_i z_0 + 2\dot{q}_i \omega_i^i \times \left(\left(R_i^{i-1}\right)^T z_0\right)$

11 $\ddot{P}_{c_i}^i = \ddot{P}_i^i + \dot{\omega}_i^i \times r_{i,C_i}^i + \omega_i^i \times (\omega_i^i \times r_{i,C_i}^i)$

---

where the $\ddot{P}$ depends on the axis of the gravity from the first base frame. $r_{i-1,i}^i$ is the position of the frame 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,C_i}^i is the position of the frame i^{th} central of mass.

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 $R_{i+1}^{i} = R_{i+1}^{i}(q_i)$

3 $f_i^i = R_{i+1}^{i} f_{i+1}^{i+1} + m_i P_{C_i}^{i}$

4 $\mu_i^i = -f_i^i \times \left(r_{i-1,i}^{i} + r_{i,C_i}^{i}\right) + R_{i+1}^{i}\mu_{i+1}^{i+1} + (R_{i+1}^{i} f_{i+1}^{i+1}) \times r_{i,C_i}^{i} + I_i^i \dot{\omega}_i^i + \omega_i^i \times (I_i^i \omega_i^i)$

5 if joint == PRISMATIC:

7 $\tau_i = \left(f_i^i\right)^T \left(R_i^{i-1}\right)^T z_0$

8 if joint == REVOLUTE:

9 $\tau_i = \left(\mu_i^i\right)^T \left(R_i^{i-1}\right)^T z_0$

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 $\mu_i{}^i$ 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.

$$\tau = \begin{bmatrix} 2.3767 \\ 0 \\ 0.3439 \end{bmatrix} \quad \text{(Lagrange Formulation)}$$

$$\tau = \begin{bmatrix} 2.3767 \\ 0 \quad 0 \\ 0.3439 \end{bmatrix} \quad \text{(Newton - Eular Formulation)}$$

The dynamics of both methods produce the exact same results, based on the data. It is significant to emphasize that the outcomes are free of environmental interference and friction.

Joint Space Controllers

"Joint space controllers" control algorithms or methods used in mechatronics and robotics to regulate the angular and angular locations of robotic manipulators. In relation to robotic arms, joint space refers to the assortment of joint angles that define the robot's structure.

Joint space controllers are used to appropriately and precisely position the robot's joints in line with a desired trajectory or set of targets. These controllers take into account the kinematics, dynamics, and constraints of the robot when determining the appropriate control actions.

PD controller with gravity compensation

A common control method in robotics for regulating the motion of robotic manipulators, especially in joint space, is a PD controller with gravity compensation. To increase the accuracy and stability of the control system, it integrates a gravity correction system with a proportional-derivative (PD) controller.

The main goal of the PD controller is to reduce the discrepancy between the robot's desired joint position and actual joint position. By creating control signals depending on the mistake and its derivative, it does this. While the derivative term considers the error's rate of change, the proportional term offers a control action that is proportional to the error.

The gravity adjustment term makes sure that the control signal accurately represents the planned position of the joints by considering the gravitational forces acting on the robot's connections. The following control law is defined for the PD controller.

$$\tau = g(q) + K_p(q_d - q) - K_d\dot{q}$$

These scenarios include neglecting gravity, adding fixed gravity matrix, and adding noise to the desired q.

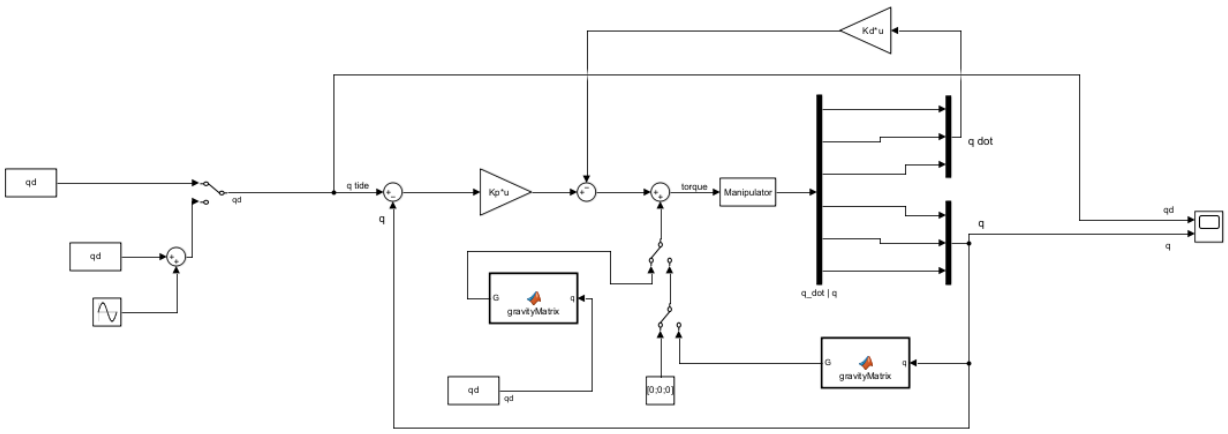


Figure 2 Joint Space PD controller with gravity compensation

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. Following parameters are set on the controller during initializing.

$$q_d = [\text{pi}/3, 0.1, 0.1]$$

$$K_p = \begin{bmatrix} 90 & 0 & 0 \\ 0 & 90 & 0 \\ 0 & 0 & 90 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 60 & 0 & 0 \\ 0 & 60 & 0 \\ 0 & 0 & 60 \end{bmatrix}$$

Figure 3 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.

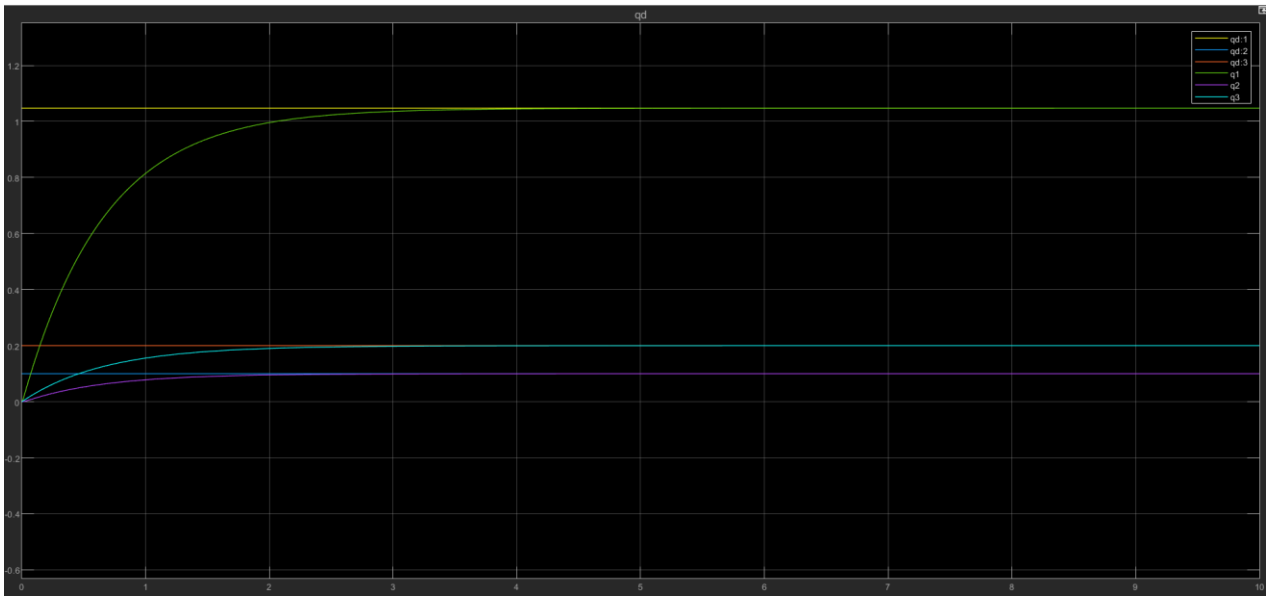


Figure 3 Joint variables with actual joint variables with gravity compensation.

The Figure 4 shows the graph of the q and q_d in which the gravity compensation is neglected, as for q_2 and q_3 there is a slight error, and they are not overlapping with q_{d2} and q_{d3}. According to motion of equation, the robot needs compensation to gravity even if the robot is not moving.

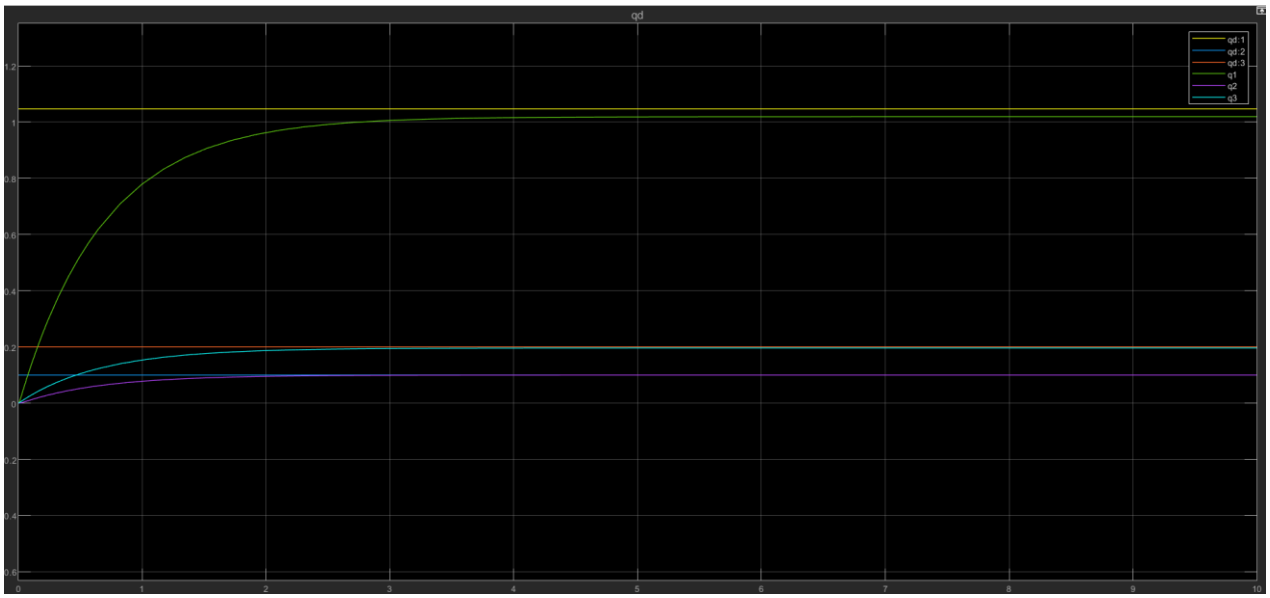


Figure 4 Joint variables with actual joint variables without gravity compensation.

Pruning to noise is one of the major drawbacks of utilizing a PD controller with gravity adjustment. To correct for the noise effect on the q_d, which limits the robot joints' range of motion, we must also account for the inertia and Coriolis forces, according to the equation of motion. Because of this, the controller was unable to precisely attain the target joint variables as depicted in the Figure.

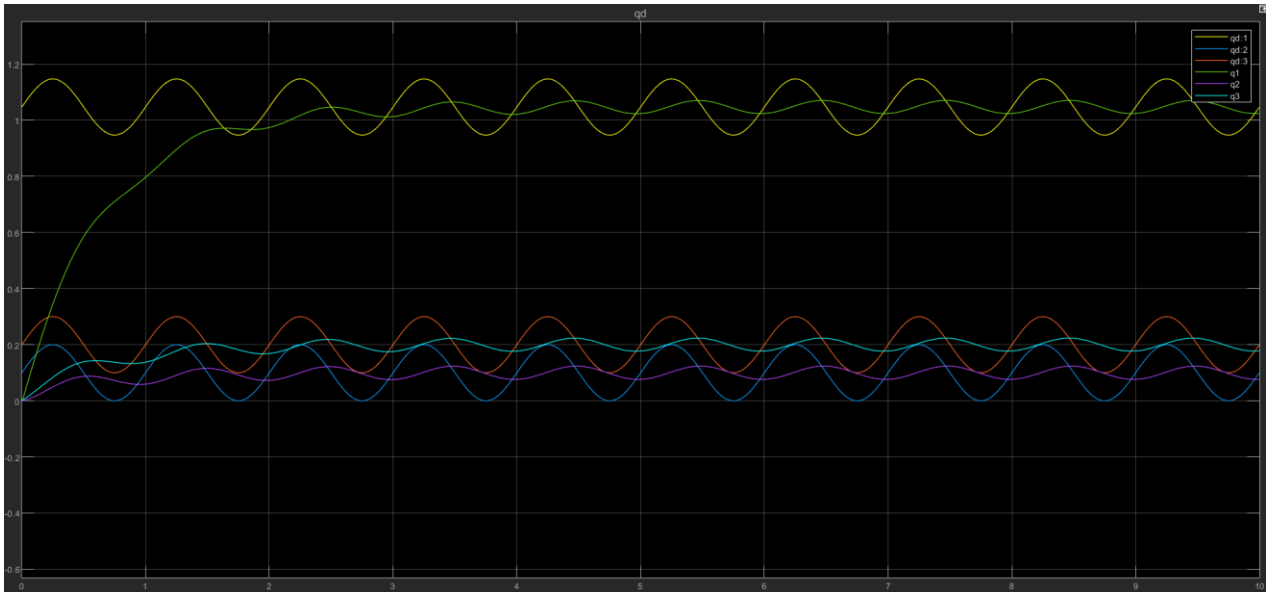


Figure 5 Joint variables with actual joint variables with gravity compensation and noise.

Inverse dynamics controller

As was previously indicated, the disadvantage of PD control with gravity compensation is that, as shown in Figure 6, the controller's effectiveness degrades when noise is present in the desired joint variable q_d. The inverse dynamic controller, which functions effectively even in noisy environments, can be a good remedy for this problem. It has been demonstrated that this controller is quite good at tracking trajectories in both the joint space and the operational space. Inertia, Coriolis, and gravity terms are among the components from the equations of motion that the inverse dynamic controller utilizes to account for all noise. It accounts for the joint variables' dynamic properties but ignores the constant term.

The control law for inverse dynamics stated that

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

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

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

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

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

The control parameters that are used in this controller are

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

$$Kd = \begin{bmatrix} 70 & 0 & 0 \\ 0 & 70 & 0 \\ 0 & 0 & 70 \end{bmatrix}$$

The control scheme for the inverse dynamics control is shown in Figure 6. It is significant to notice that the schema includes manual switches for switching between various circumstances, such as constant q_d and q_d trajectory. Additionally, this is the controller's tiny design. The controller is built from two distinct pieces, which are detailed below.

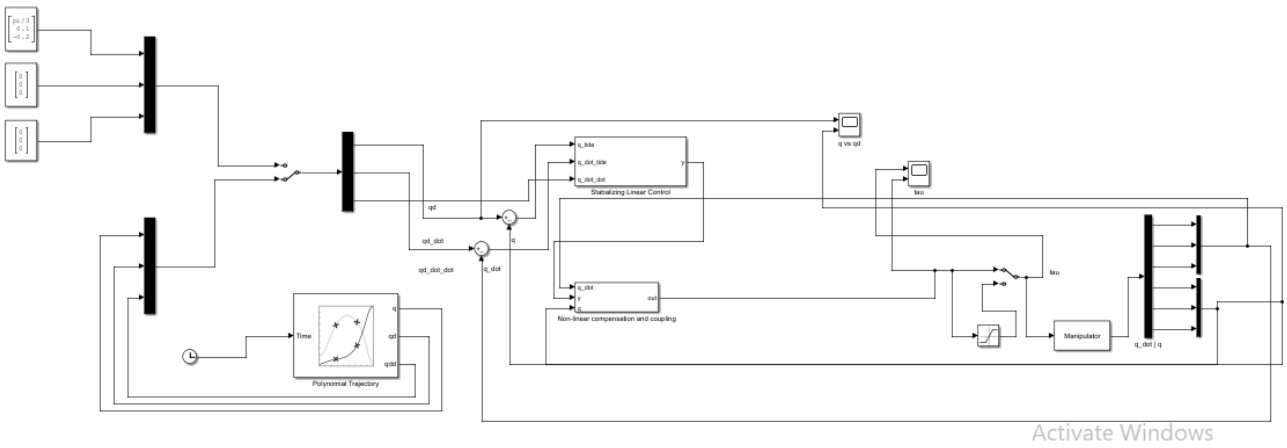


Figure 6 Joint Space Inverse Dynamic Controller

Assuring that the controller is stable and covering the aim is the stabilizing linear control. The stabilizing linear control paradigm is suggested in Figure 7. The gains that add up to the (q_d) are the K_d and K_p.

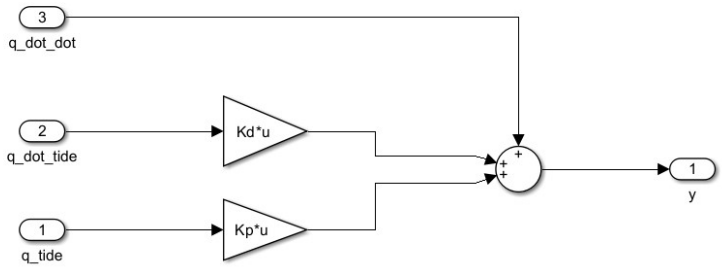


Figure 7 Stabilizing Linear Control

When the robot is travelling along the desired trajectory, as previously indicated, it is crucial to account for gravity, Coriolis, and inertia. The non-linear correction of n(q,q) and B(q) is depicted in Figure 8. It is significant to note that many scenarios are included in the schema for testing the controller. The neglection of inertia, Coriolis, or gravity compensation are some examples of these circumstances. Additionally, the inverse dynamics might incorporate noise as well.

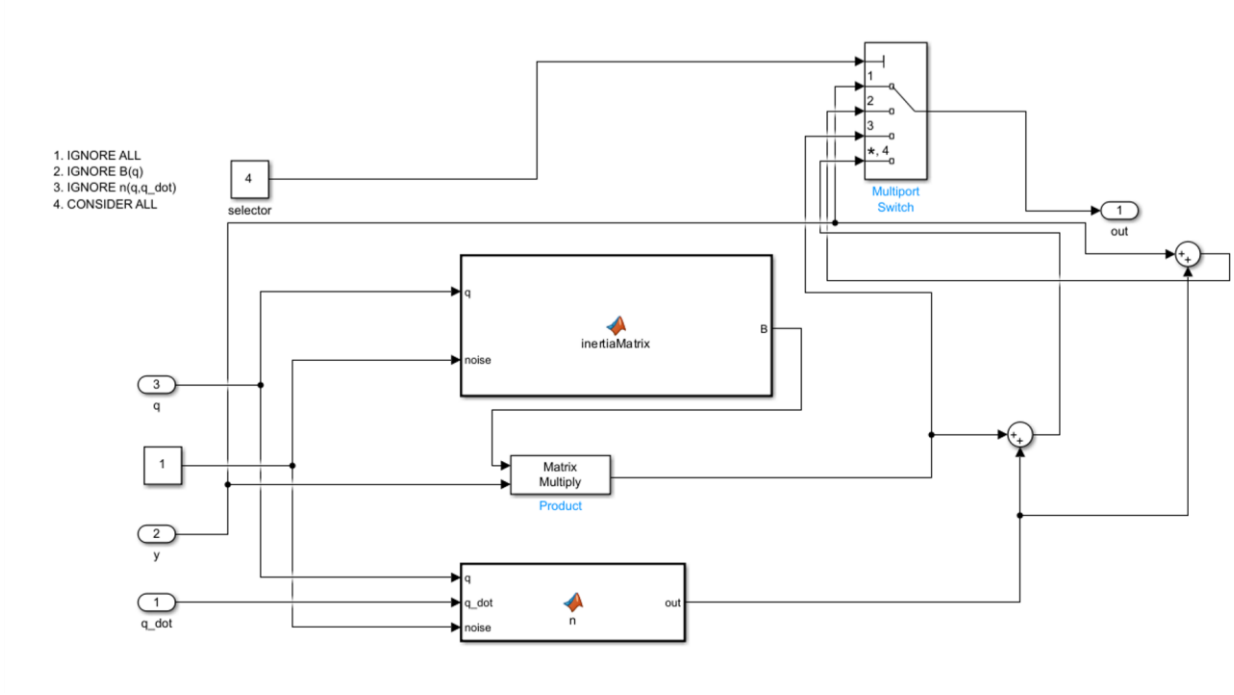


Figure 8 Non-linear compensation and coupling

Figure 9 shows the outcomes of the inverse dynamic controller. Figure shows that the q is accurately converged to the desired trajectory. It is crucial to keep in mind that the trajectory is quite dynamic and that the graph only represents 3-time steps.

Figure 9 Inverse dynamics converges to qd

But for inverse dynamics control to function, a precise dynamic model is required, just like with PD controllers with gravity compensation. The controller may cause the problem if the dynamic model is inaccurate. Figure displays the controller's output for several inverse dynamic model elements. To replicate the circumstance, the masses of the linkages are different from the actual masses of the manipulator. Figure 10 demonstrates that the graph does not appropriately converge toward the desired joint variable.

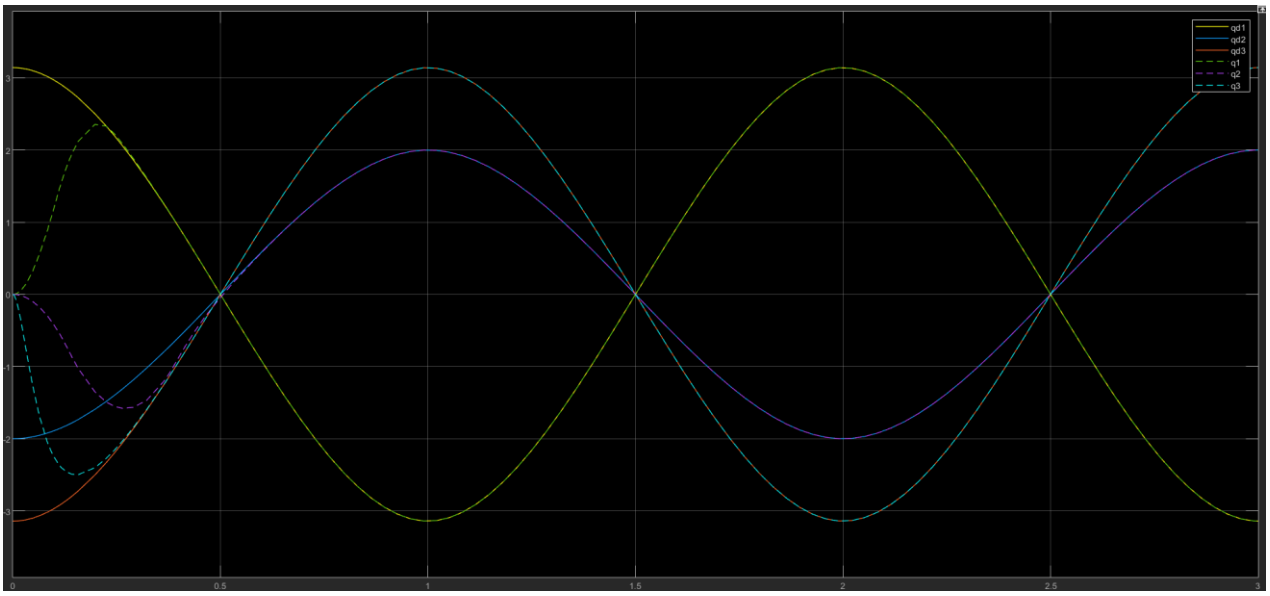


Figure 10 Inverse dynamics with slightly inaccurate model.

However, graphs are created without considering gravity, Coriolis, and inertia from the controller to determine the significance of the compensation factors. Figures, respectively, illustrate the graphs for the controllers of the inverse dynamic with and without inertia. Figure 11 depicts the controller's graph without the inertia term.

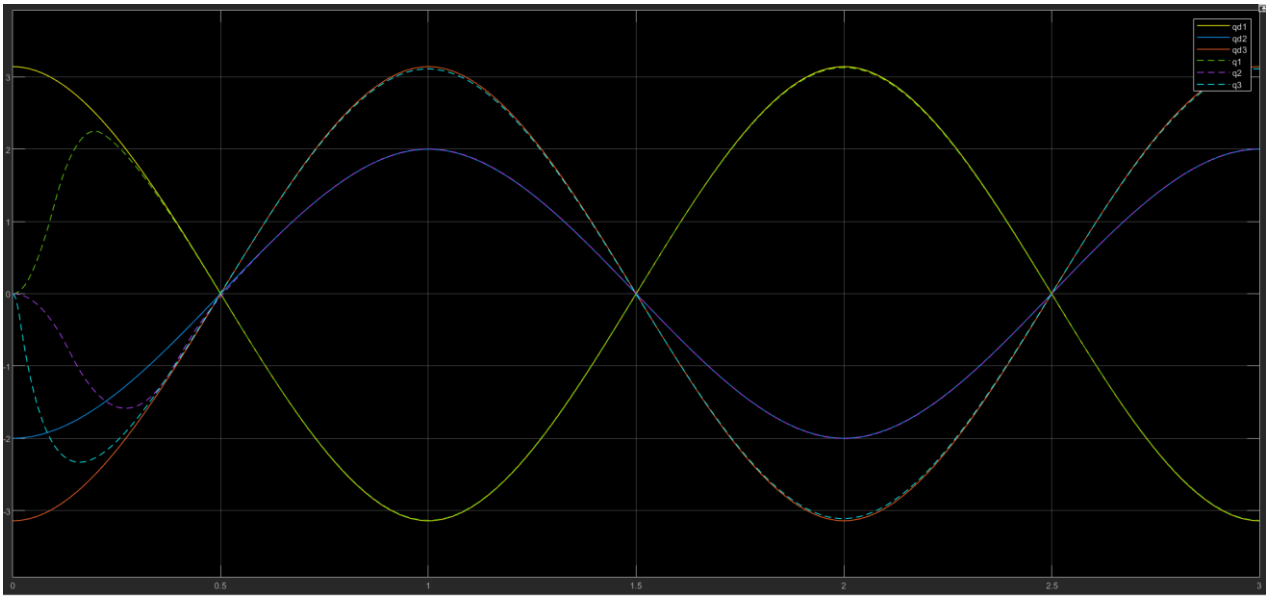


Figure 11 Inverse dynamic without inertia compensation

Similarly Figure 12 shows the controller's graph without the n(q, q) term. It is evident that the controller has a significantly larger inaccuracy than Figure 11. However, the controller continues to attain the goal values after some time. The error is not significantly higher if all compensation terms are eliminated, as shown in Figure 13, but the controller will never achieve the joint value objective values.

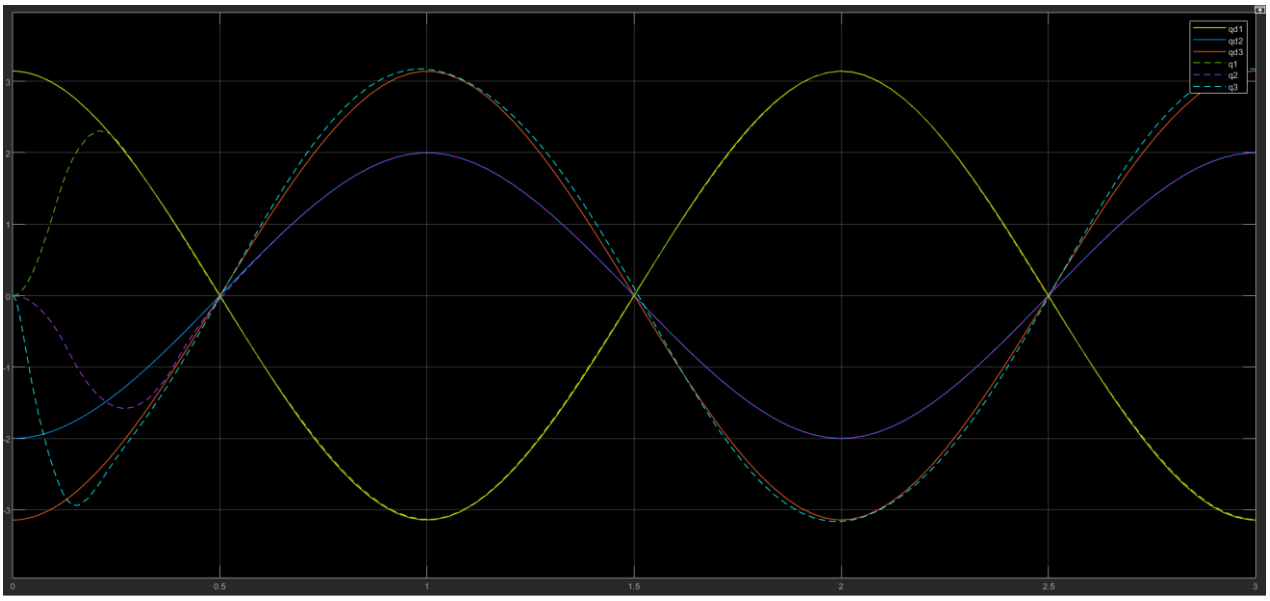


Figure 12 Inverse dynamic without gravity and Coriolis compensation

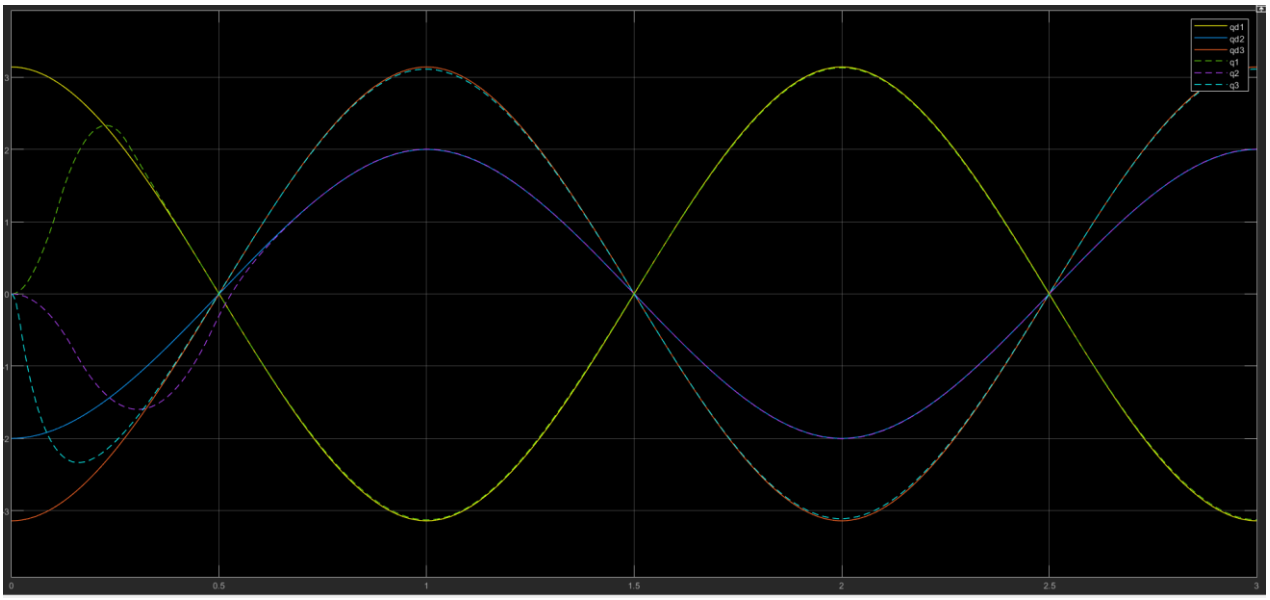


Figure 13 Inverse dynamics without any compensation

A control system's settling time can be shortened by increasing K_p values, but doing so also necessitates more torque, which might compromise the system's stability or possibly harm its mechanical or electrical components. To maintain safety, real-world manipulators frequently have restrictions on the available current or voltage. The Simulink model can be enhanced with a saturator block to detect when torque values go beyond these restrictions to solve this problem. Due to the high value of K_p, as shown in Figure 14, an unreasonably large amount of torque is applied to obtain the desired value. In the actual world, though, it can have an impact on the controller's accuracy because mechanical errors are always protected against large current values.

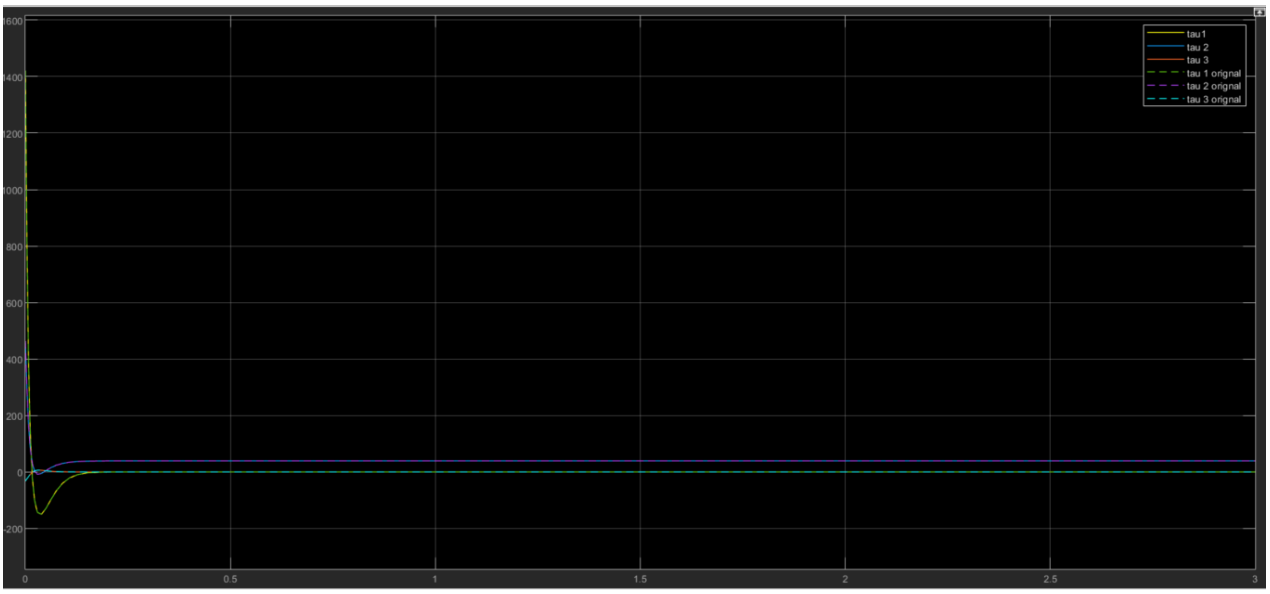


Figure 14 Torque pruning

Adaptive Controller

As was already established, the correctness of the manipulator's dynamic model has a significant impact on the effectiveness of PD control with gravity compensation and inverse dynamics control. These controllers could not converge to the appropriate positions or values if there are errors. The use of adaptive controllers can get around this restriction. By continuously learning the dynamic parameters of the manipulator, adaptive controllers can adjust to changing dynamics. These controllers solve the regressor problem for the specific manipulator by using the dynamic parameters as inputs. In this manner, the adaptive controller can calculate the unknown parameters and use them to modify its behavior.

However, putting adaptive controllers into practice is not a simple operation. Using a 1 Degree of Freedom (DoF) manipulator, adaptive controller effectiveness is demonstrated. Even if the dynamic model is erroneous, the manipulator's dynamic parameters can be learned and adjusted in real-time with an adaptive controller, improving accuracy. The following is the adaptive controller's control law:

$$\tau = (\ddot{q}_r, \dot{q}_r, q)\hat{\Theta} + K_d\sigma$$

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

$$\tau = +K_d\sigma$$

where B, 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

$$\ddot{q}_r = \ddot{q}_d + \lambda\tilde{\dot{q}}$$

$$\dot{q}_r = \dot{q}_d + \lambda\tilde{q}$$

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

$$\sigma = \dot{q}_r - \dot{q}$$

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

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

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. Furthermore, Figure 15 shows the schema of adaptive control using the following manipulator

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

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.

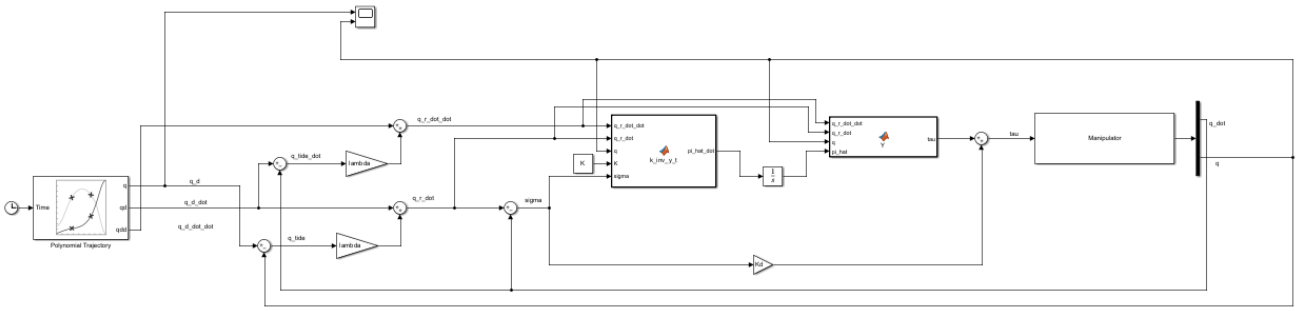


Figure 15 Schema for the adaptive control

The control parameters that are used in the adaptive controller

I = 0.1, = 0.2, g = 9.806

k = 30.2,$_d$ = 500, and λ = 100

The mass of the object is calculated using

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

where the dimensions of a connection with a density of 2.7 are a=1.1, b=1.1, and c=3.5. The center of mass is also d = c/2. The adaptive controller creates the graph in Figure 16 by using the specified parameters. Figure 17 shows that the q quickly and readily converges to q_d.

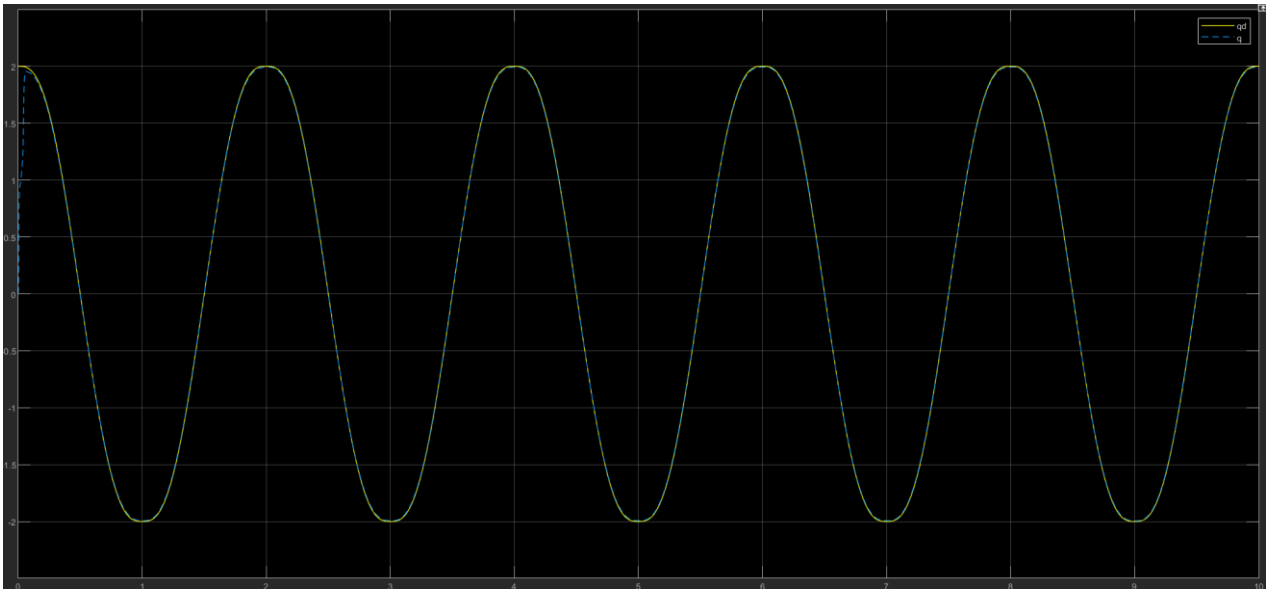


Figure 16 Adaptive control graph 1 DoF between desired and actual joint variable

Operational Space Controllers

The term "operational space" is used in robotics to define the desired motion of a robotic manipulator's end-effector in the actual world, which is commonly represented using a Cartesian coordinate system. Euler angles can also be used to describe the end-effector's orientation. Operational space offers a more straightforward way to describe and regulate the motion of the end-effector since it is independent of the robot's joint variables.

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.

The robot's motion can be managed by changing its position in actual cartesian coordinates using the operational space controller. The correct kinematic and differential kinematics, or analytical and geometrical Jacobian, are required to transition from joint space to operational space. It's also crucial to keep in mind that the problem of singularity exists in the operational space, preventing the robot from achieving the ideal position when it loses one or more degrees of freedom. The operational space controllers are also slower and more complicated than the join space controllers.

PD controller with gravity compensation

A proportional-derivative (PD) control with gains is utilized to regulate the locations of a robot. The PD controller in operational space is comparable to the joint space PD controller, but it works in a different environment. Unlike the joint space PD controller, which controls the position in joint variables, the operational space PD controller regulates the position in Cartesian coordinates. The end-effector is moved towards the desired position by the controller, which computes the difference between the desired and actual positions as well as orientations in terms of Euler angles. The controller then applies a proportional and derivative gain to this mistake to generate fresh torque. However, when the robot is subject to gravity, the weight of its linkages may impact the torques produced by the PD controller, resulting in inaccuracies in position control. A gravity compensation term is introduced to the controller to account for the gravitational forces acting on the robot's connections in order to correct for this, ensuring that the control signal accurately reflects the required joint positions. The operational space PD controller, however, might run into a singularity problem, which could prevent it from converging to the required locations or orientations.

It is significant to highlight that direct kinematics are used in this literature to determine the required position and orientation. The desired set of q_d is defined similarly to joint space, and the position P_e is extracted using the homogeneous matrix. The rotation matrix is used to obtain the orientation _e in terms of ZYZ Euler angles. The following illustrates how the desired position and orientation vector x_d is expressed as a 6x1 vector.

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

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

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

The control law for the PD controller for operational space with gravity compensation can take a following form which is also reflected in Figure 17.

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

It is significant to note that the controller's gravity correction feature can be turned off manually via a switch. The following PD controller parameters are used by the controller.

$$Kp = \text{diag}([200,200,200,50,50,50])$$

$$Kd = \text{diag}([150,150,150,20,20,20])$$

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

$$qd1 = [0.5629, -0.6600, 0.7250]$$

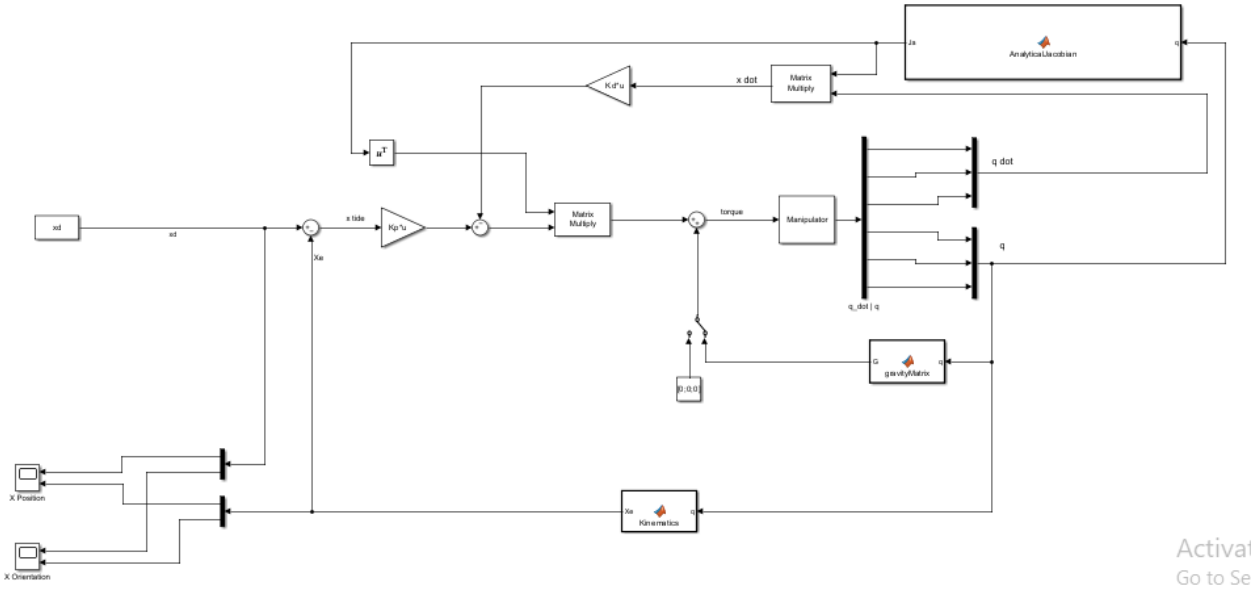


Figure 17 Operational space PD controller with gravity compensation schema in Simulink

Figures 18 and 19's graphs demonstrate how the q_d1's position and orientation have converged to its desired position and orientation. There is a time step in the orientation graph where the orientation abruptly changes to the intended position for an unidentified reason. The robot recovers from this predicament and moves in the direction of the intended location and orientation. The singularity occurs at the best estimate of orientation and position x.

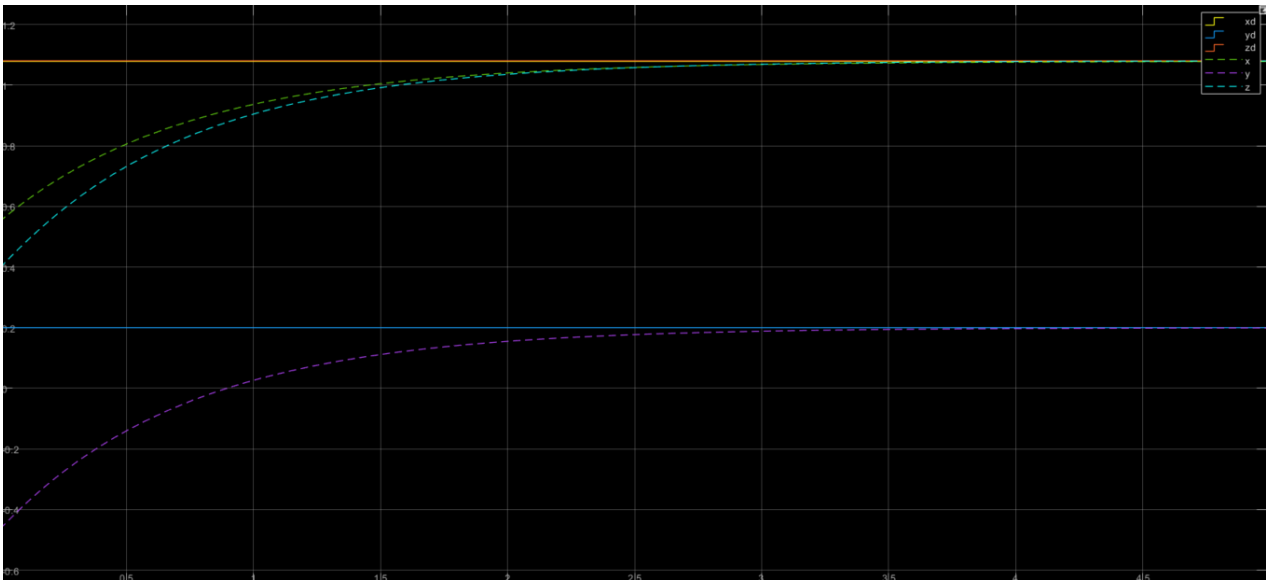


Figure 18 Graph for $qd1$ converges to desired position Pde.

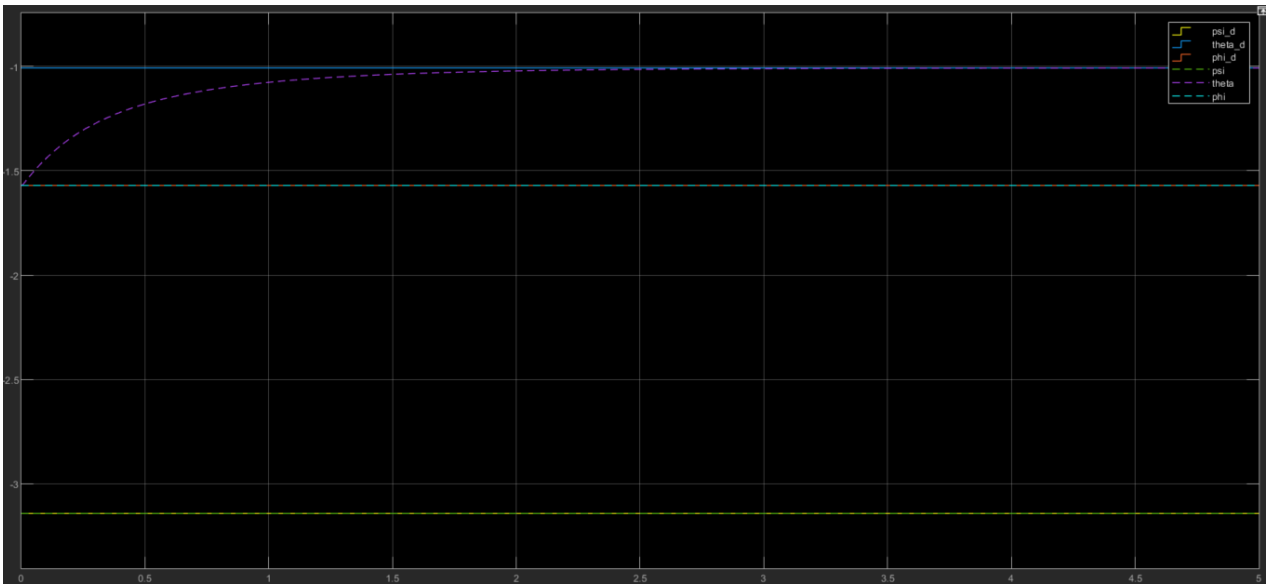


Figure 19 Graph for $qd1$ converges to desired orientation ϕde.

The controller is shown in Figures 20 without gravity adjustment. Position do not converge to the intended positions P_e1, when gravity is not corrected in the controller. The fact that link 3 of the robot is so heavily dependent on gravity.

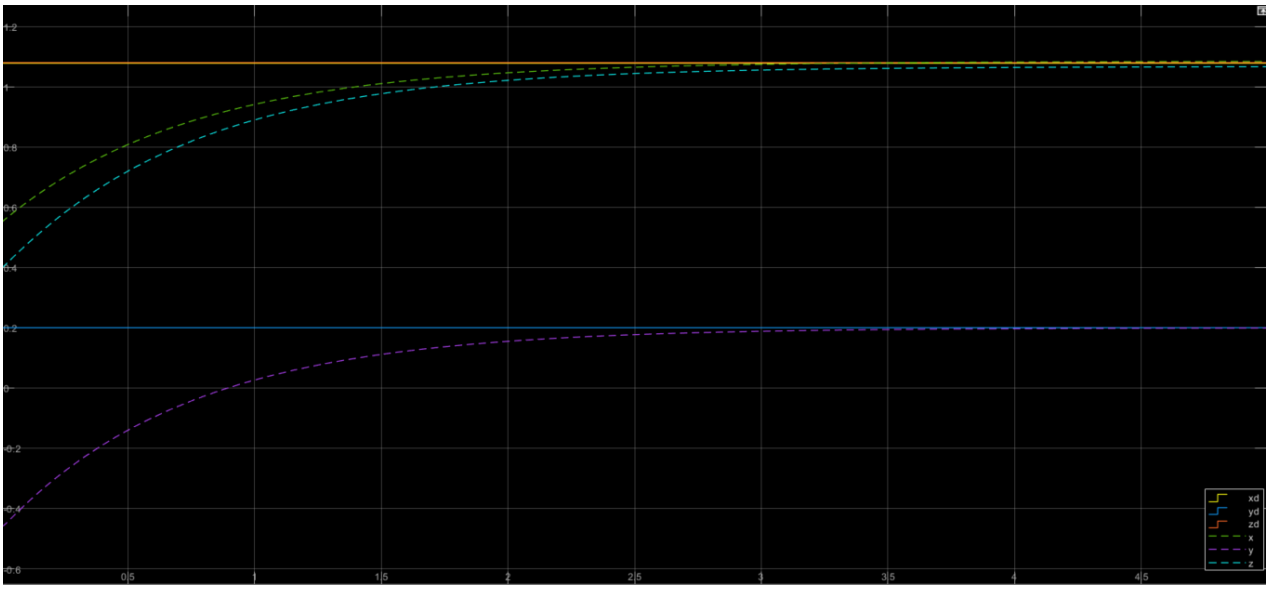


Figure 20 Graph without gravity compensation for $qd1$ does not converges to desired orientation ϕde.

.

Inverse Dynamic Controller

The operational space PD controller with gravity compensation experiences performance loss in the presence of noise in the desired position and orientation, just like the joint space PD controller. The usage of the inverse dynamic controller in the operational space is a practical remedy for this problem. The inverse dynamic controller has proven to be quite successful at tracking trajectories in the operational space, even when there is noise present. By using components from the equations of motion, such as inertia, Coriolis, and gravity factors, and taking into consideration the dynamic nature of the position and orientation, it eliminates all noise in the operating space.

It is significant to highlight that the precision of the manipulator's dynamic model has a significant impact on the effectiveness of the inverse dynamic controller. To ensure the success of this kind of controller, precise system modeling is essential. Additionally, the inverse dynamic controller may experience the singularity problem, just like the PD controller in operational space. This can prevent the controller from converging to the preferred locations and orientations. According to the control law for inverse dynamics,

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

Where y is a stabilizing linear control defined as

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

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

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

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

The Simulink schema for the inverse dynamic controller in operational space is displayed in Figure 21. The polynomial trajectory block is used to define the q_d, which is then transformed to the x_d, x _d, and x _d using the direct kinematics, analytical Jacobians, and differential chain rules, respectively, to set the x_d trajectory for the controller. Stabilizing linear control and non-linear compensation and coupling are the names of the two blocks. Figure 22 illustrates the application of analytical Jacobian and PD gains to add stability to the controllers. It is important to note that the block's output y is not the same as the y expressed in Equation; instead, it must be multiplied by J_a(-1) to yield the y term.

It is important to note that the framework includes different test scenarios to assess the performance of the controller, as illustrated in Figure 23. In these cases, the controller's actions when the inertia, coriolis, or gravity compensations are disregarded are being studied. Additionally, as part of the testing procedure, it is possible to add noise to the inverse dynamics.

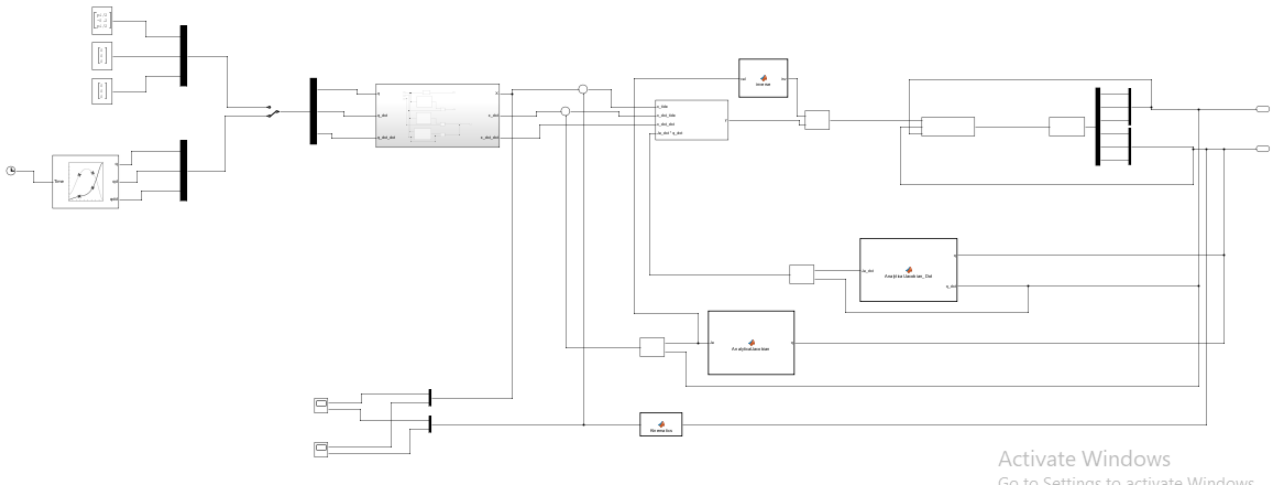


Figure 21 Operational space inverse dynamics schema in Simulink.

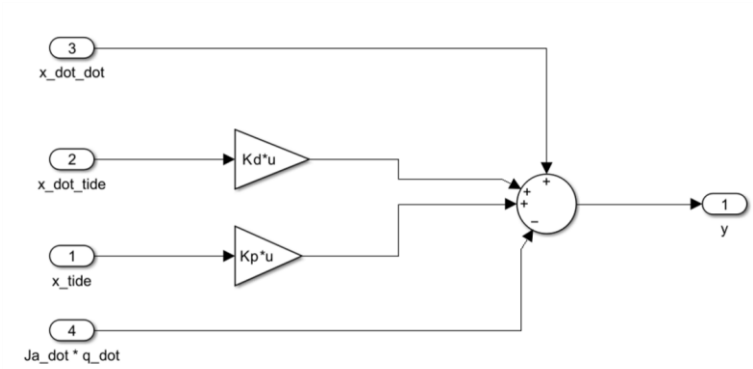


Figure 22 Stabilizing linear control.

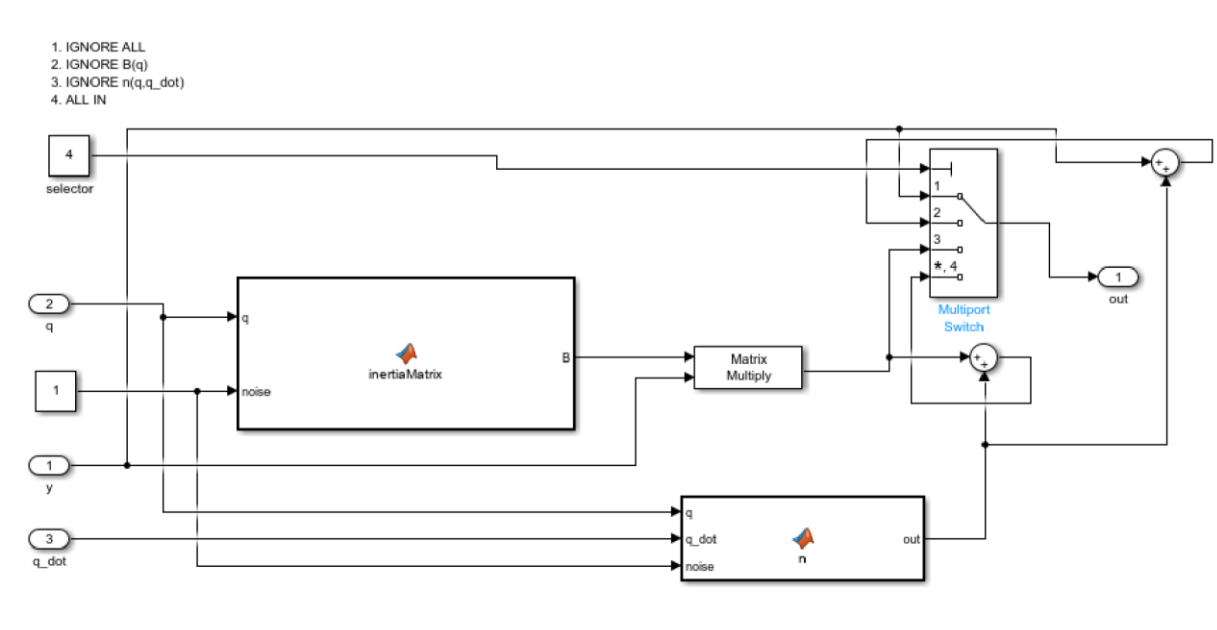


Figure 23 Non-linear compensation and coupling

As it can be clearly seen by the Figure 24 and 25, 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.

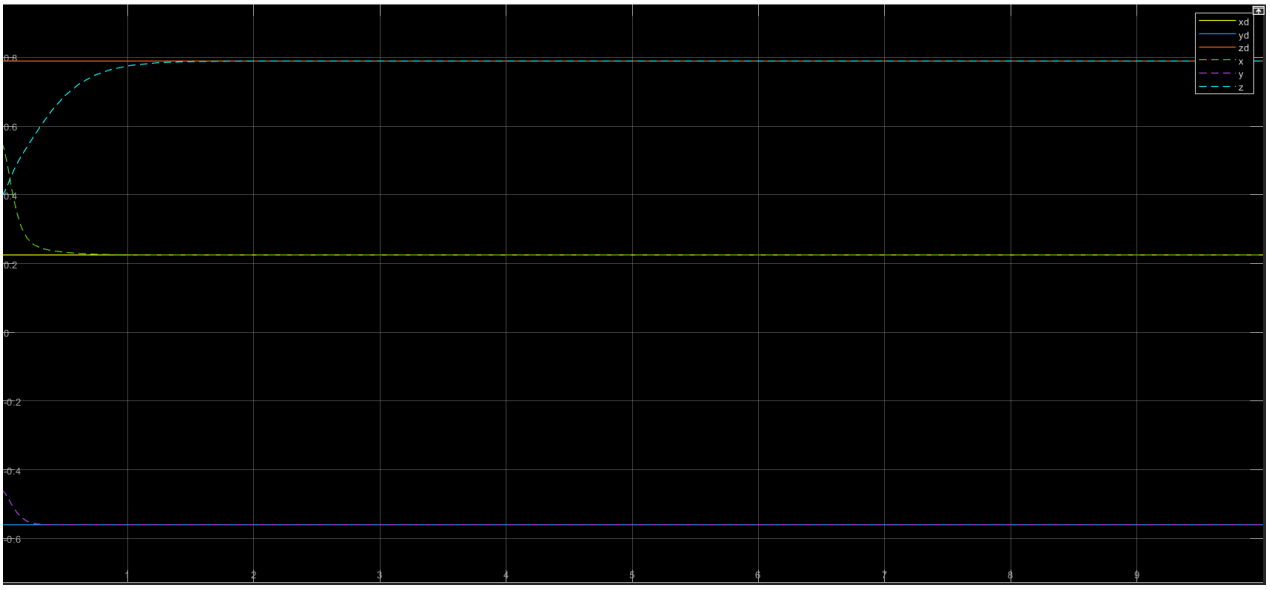


Figure 24 Inverse dynamics in operational space position Pe graph

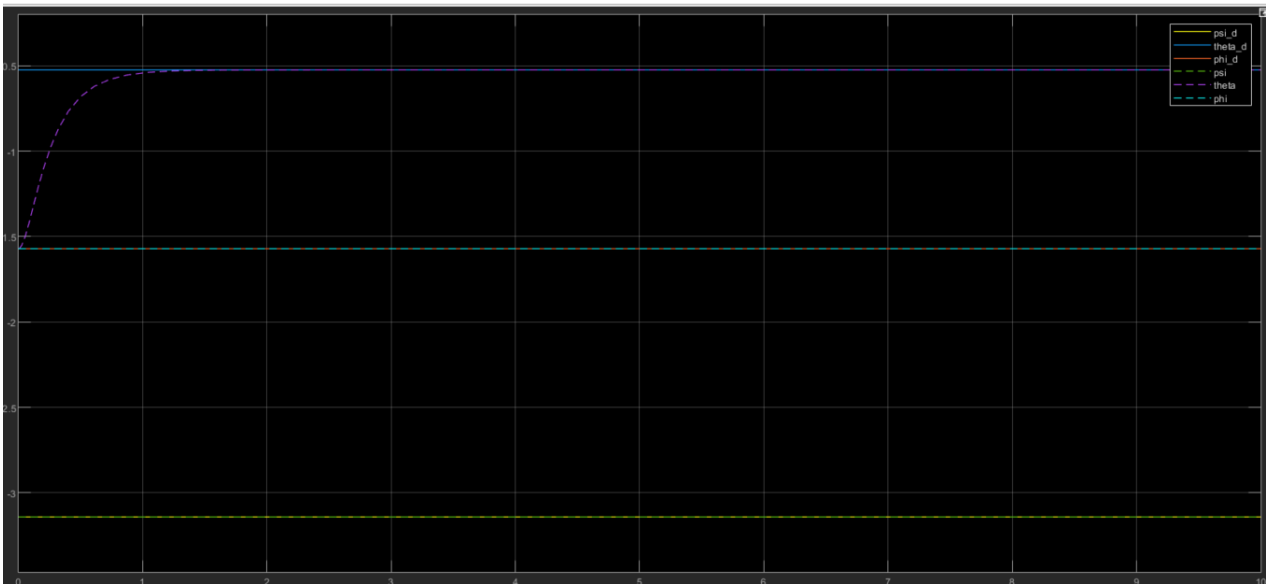


Figure 25 Inverse dynamics in operational space position ϕe graph

Force Control

Because the robotic manipulator can interact with the surroundings, it's critical to manage its force. The primary indicator of this interaction is the contact force at the end-effector of the manipulator. Estimating the potential contact forces of the manipulator is crucial. This part presents ideas like mechanical compliance, impedance, and admittance while concentrating on the operational space motion control techniques used during manipulation tasks. Indirect force control refers to these motion-based force controllers. Additionally, direct force control techniques that are created by modifying motion control strategies with an external force feedback loop are described. Despite the fact that the forces are inherently specified in operational space, for the sake of convenience, operational space models are employed in this section.

Indirect Force Control

As was already said, an indirect force control technique modifies the motion of a robotic manipulator to indirectly regulate the force applied by the robot. To be more precise, this is accomplished by creating a motion control law that generates the manipulator's intended motion trajectories, which in turn produces the desired contact force between the manipulator's end-effector and the environment. The force is a result of the motion control rather than being directly regulated. The ensuing sections define mechanical impedance, compliance, and admittance controllers as they relate to indirect force controllers.

Compliance Controller

Compliance control describes a robot's capacity to passively respond to external forces by deforming or moving in accordance with those forces. To put it another way, compliance control enables the robot to remain docile or "soft" in the face of outside forces. Compliance control aims to increase the precision and safety of the robot's motions while minimizing the impact of outside forces on the robot and the things it is managing. The virtual wall and spring model are employed in the compliance control, allowing the robot to interact with it. It's crucial to keep in mind that the virtual wall is only perpendicular to one axis for the sake of simplicity. In other words, the robot can only interact in one way with its surroundings.

The robot in this literature only engages in z-axis interactions. Additionally, the J is used in the controller because it is not functioning in this one. The definition of the control law for the compliance control is

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

Where the outer wrench h_e can be defined as

$$\boldsymbol{h_e = K_e dx_{r,e}}$$

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

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

In Equation above, 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

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

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

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

The compliance control output for position P and orientation is shown in Figures 27, 28. Because of the environment, it is evident that the position and orientation have not yet converged. Due to the environment's spring-based concept, there is little elasticity. The numbers, however, depict the unlikely case in which K occurs. As x = x x, the inaccuracy in x is lower. The z-axis of location shows the greater influence, while the y-axis also shows a very slight mistake. This results from the influence of one link on the manipulator's other link. Although the mistake in orientation is minimal, just theta is impacted and cannot get the desired orientation. The exterior impact of the end-effector on the environment is depicted in Figure 29. The manipulator begins exerting power against the environment as soon as it makes contact. As can be observed, as the force increases, the environment deforms until it reaches a point where the elasticity property causes it to become constant. The h_e force is stronger than the equilibrium force applied to the manipulator by the environment because K_p > K_e.

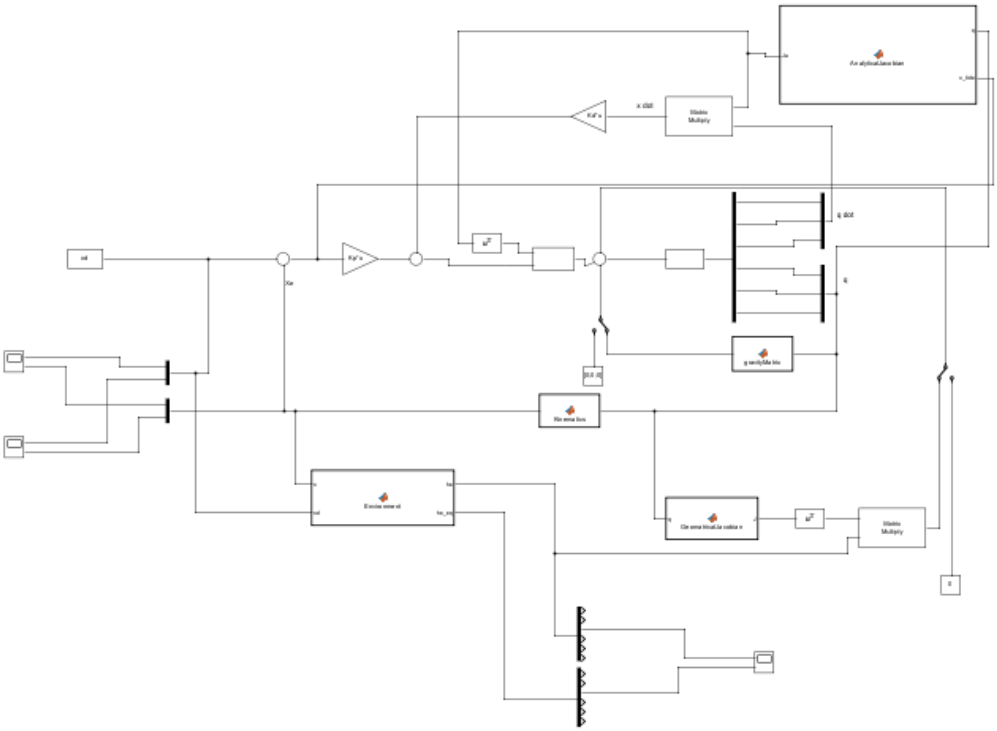


Figure 26 Compliance controller

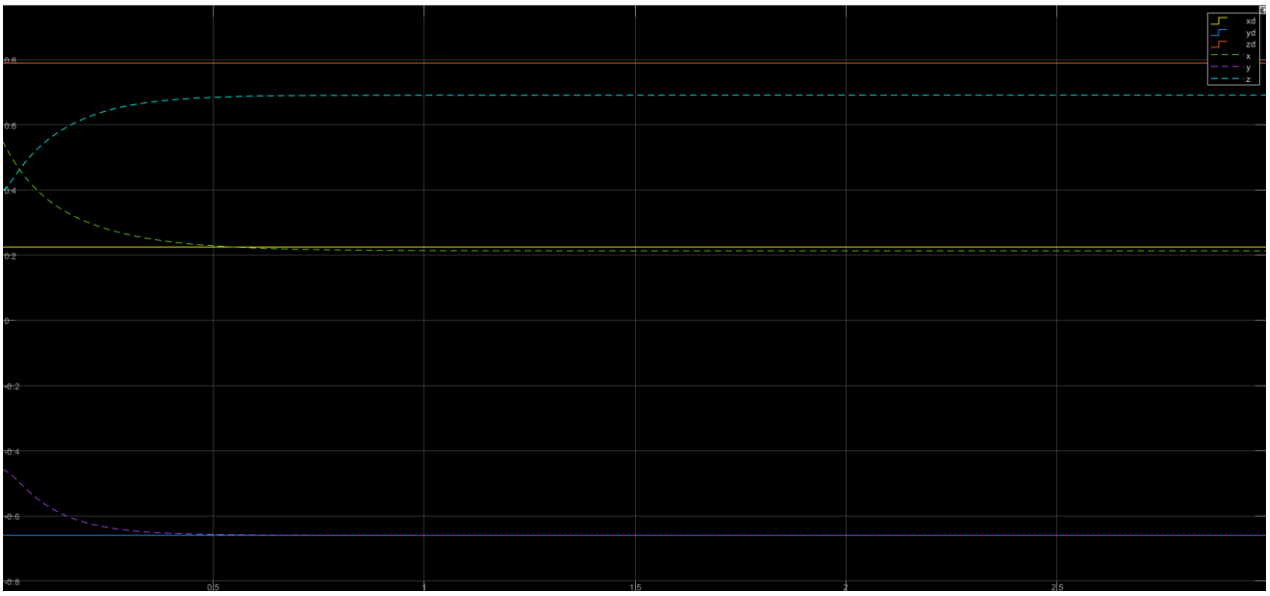


Figure 27 Compliance controller with $Kp \gg Ke$ position graph

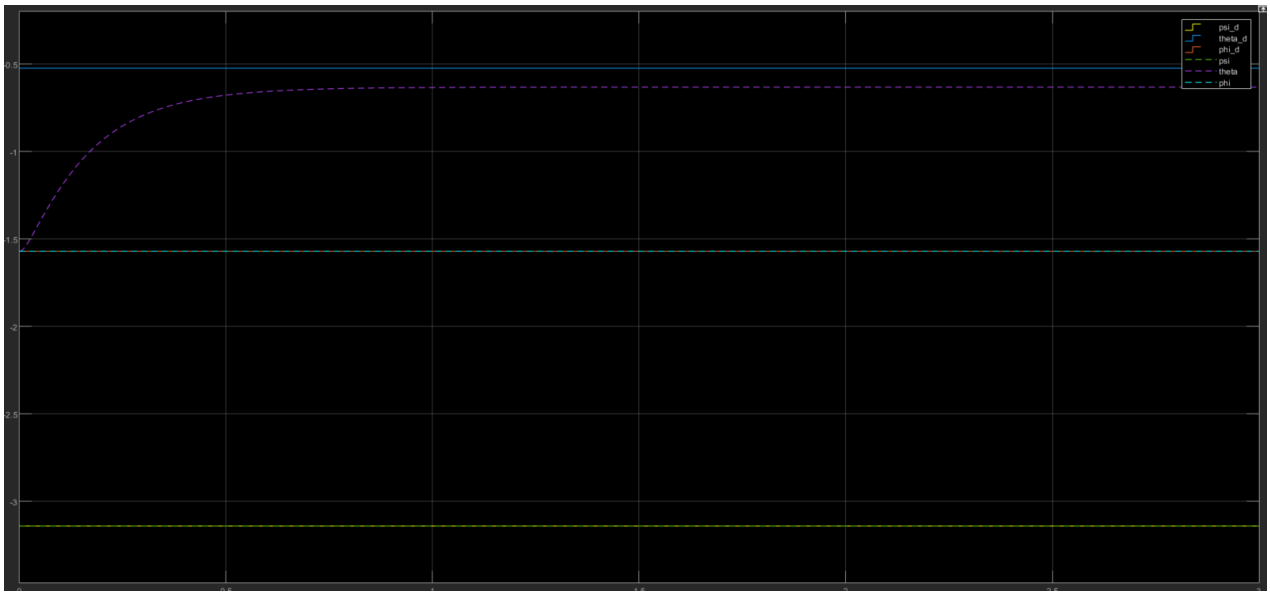


Figure 28 Compliance controller with $Kp \gg Ke$ orientation graph

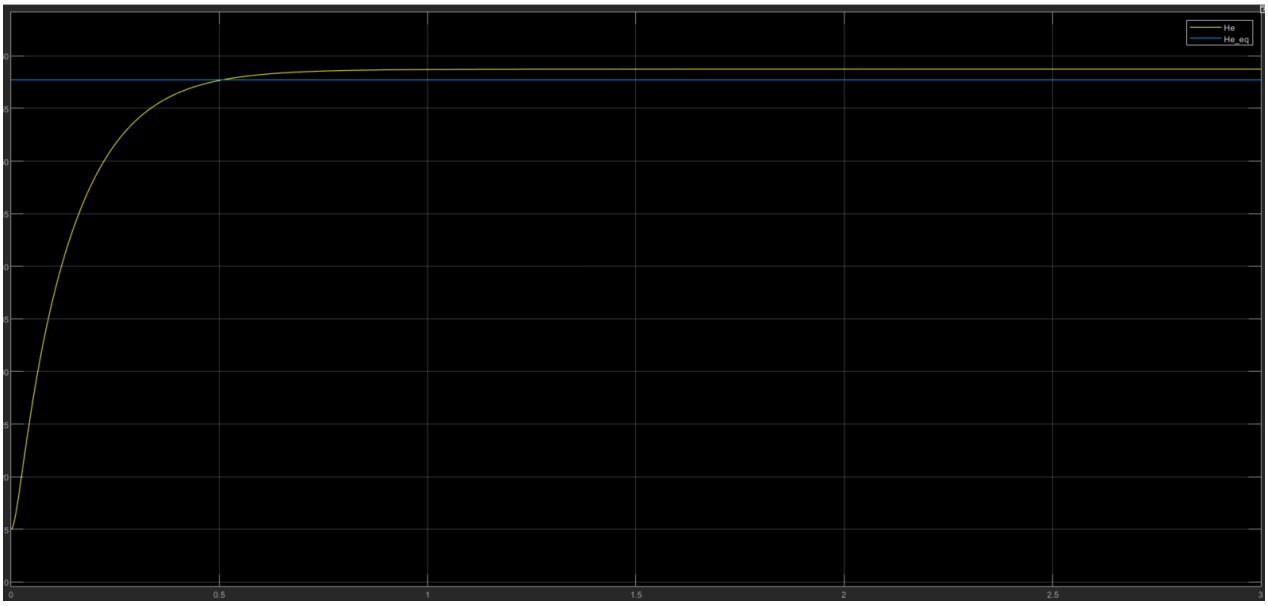


Figure 29 Compliance controller with $Kp \gg Ke$ external wrench he graph

The end-effector's location and orientation for the K_p ≪ K_e example are shown in Figures 30 and 31 because the manipulator is more brittle than the surroundings. In comparison to the scenario above, the manipulator has a larger inaccuracy in both location and orientation. The figures demonstrate how little suppleness the environment exhibits.

The manipulator needs more effort to achieve the desired position and orientation since the environment is very stiff and difficult to deform, as shown in Figure 32, therefore the required force is considerably more than the equilibrium force.

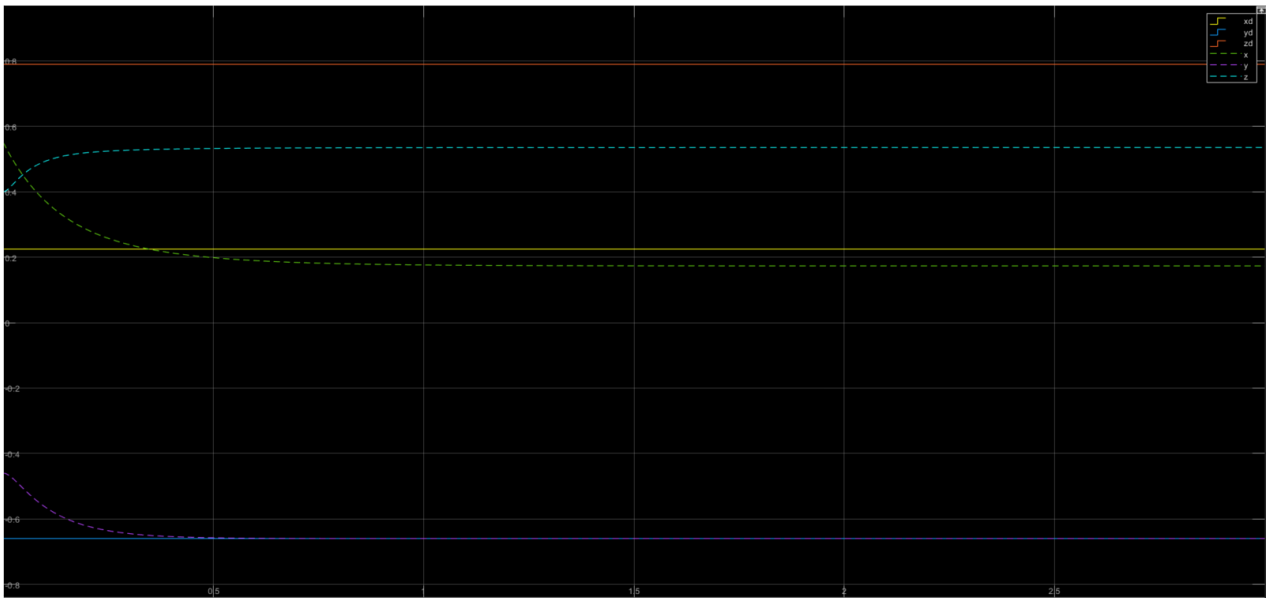


Figure 30 Compliance controller with $Kp \ll Ke$ position graph

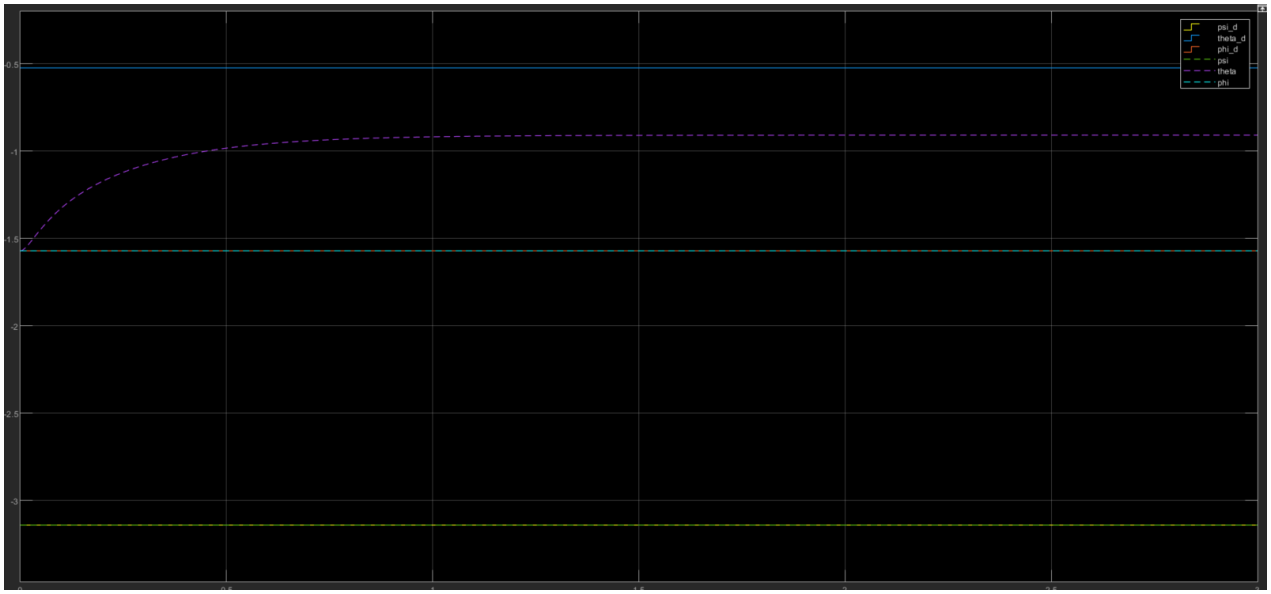


Figure 31 Compliance controller with $Kp \ll Ke$ orientation graph

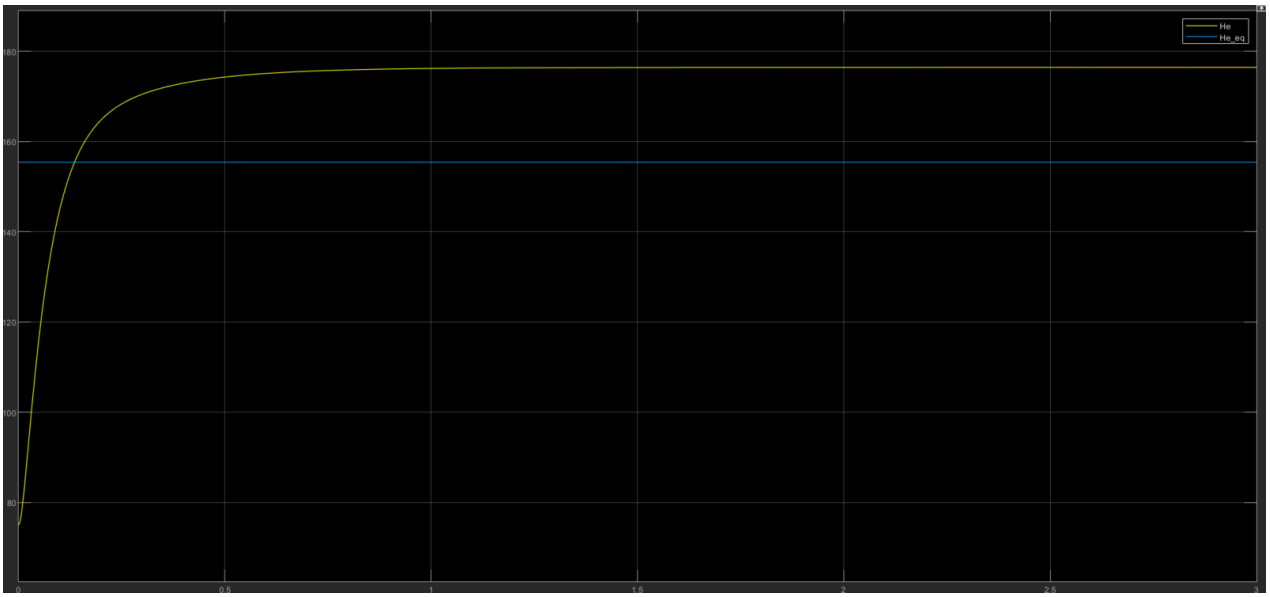


Figure 32 Compliance controller with $Kp \ll Ke$ external wrench he graph

The controller acts as a PD controller in operational space and flawlessly converges to the required position and orientation as demonstrated in Figures 33 and 34 if the robot is put to free motion, which means it does not interact with the environment. Additionally, the external wrench and the equilibrium force are both zero.

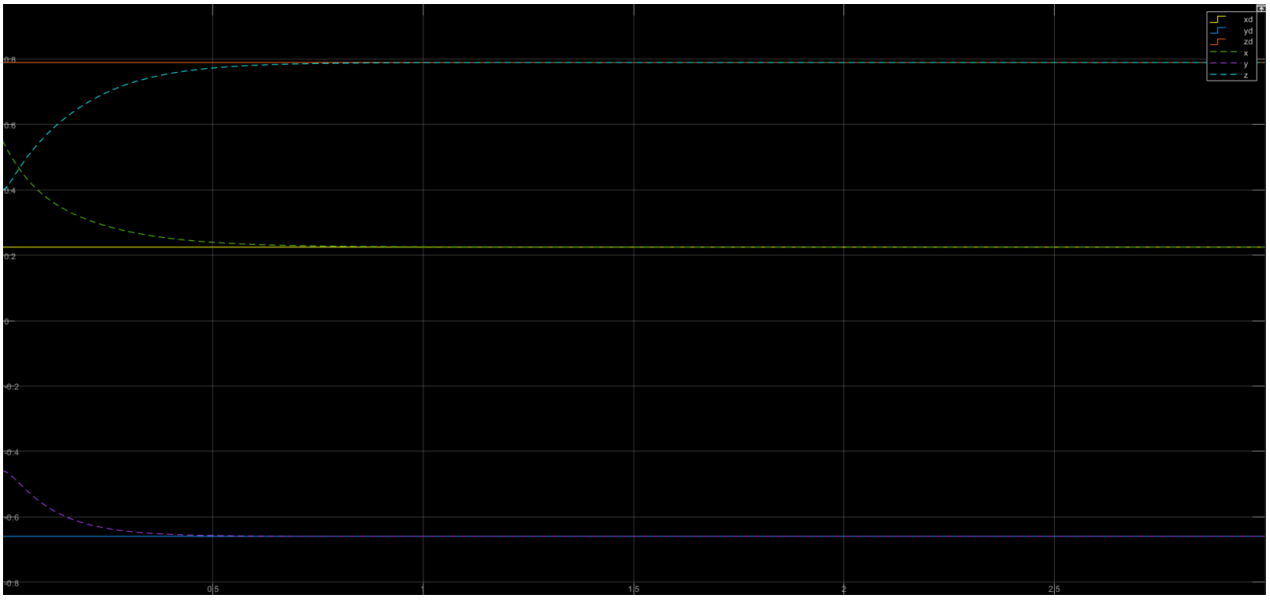


Figure 33 Compliance control without external wrench he position graph

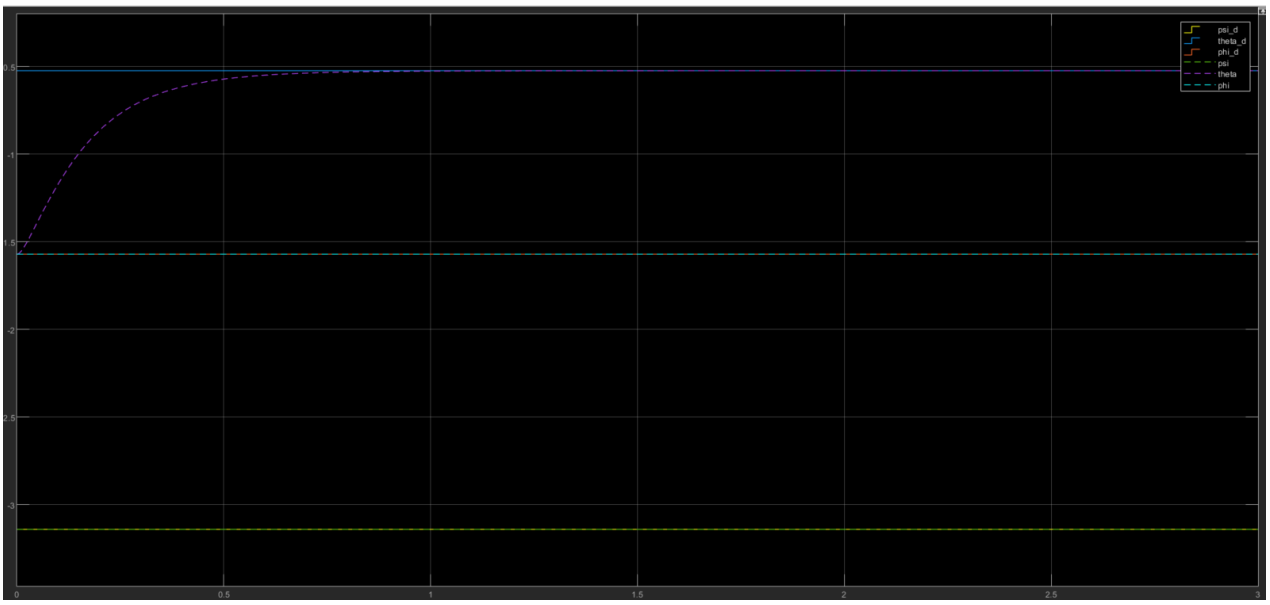


Figure 34 Compliance control without external wrench *he* orientation graph

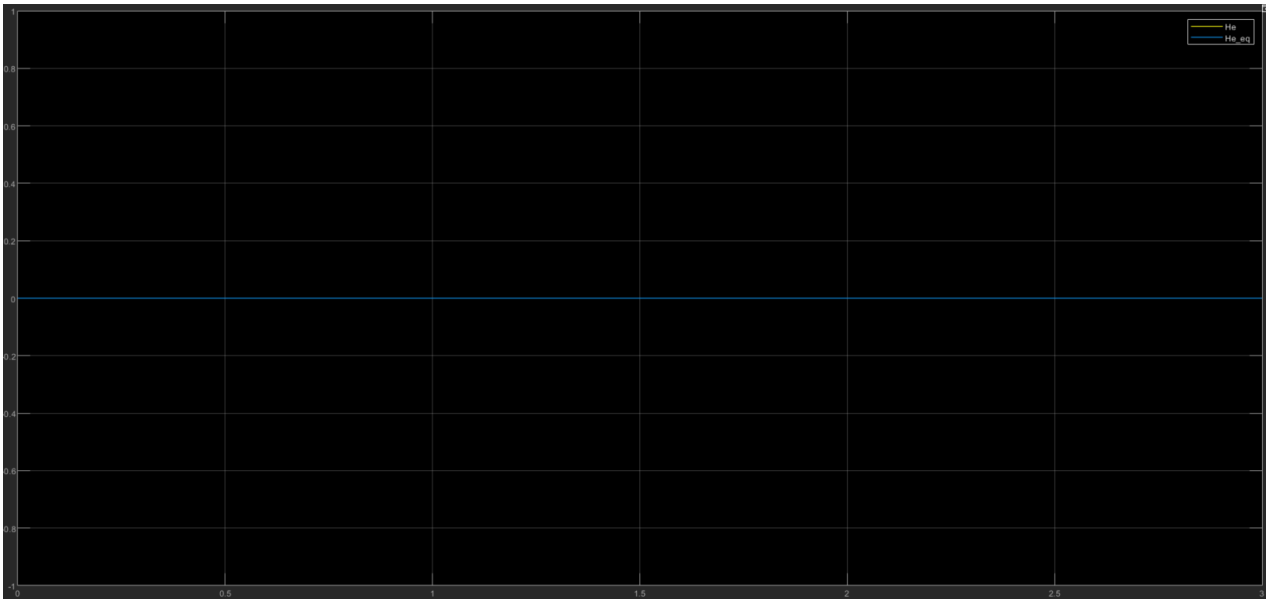


Figure 35 Compliance control without external wrench *h* , external wrench *he* graph

Impedance Controller

Impedance control is the modification of the control law for inverse dynamics controls in operational space stated in below Equation. The impedance control legislation claims.

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

where

$$y = J_{a-1}(q)M_{d-1}(M_d\ddot{x}_d + K_d\dot{\tilde{x}} + K_p\tilde{x} - M_dJ_{\dot{a}}(q, \dot{q})\dot{q})$$

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 36, 37, and 38. However, in this experiment, following parameters are used in the control law

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

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

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

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

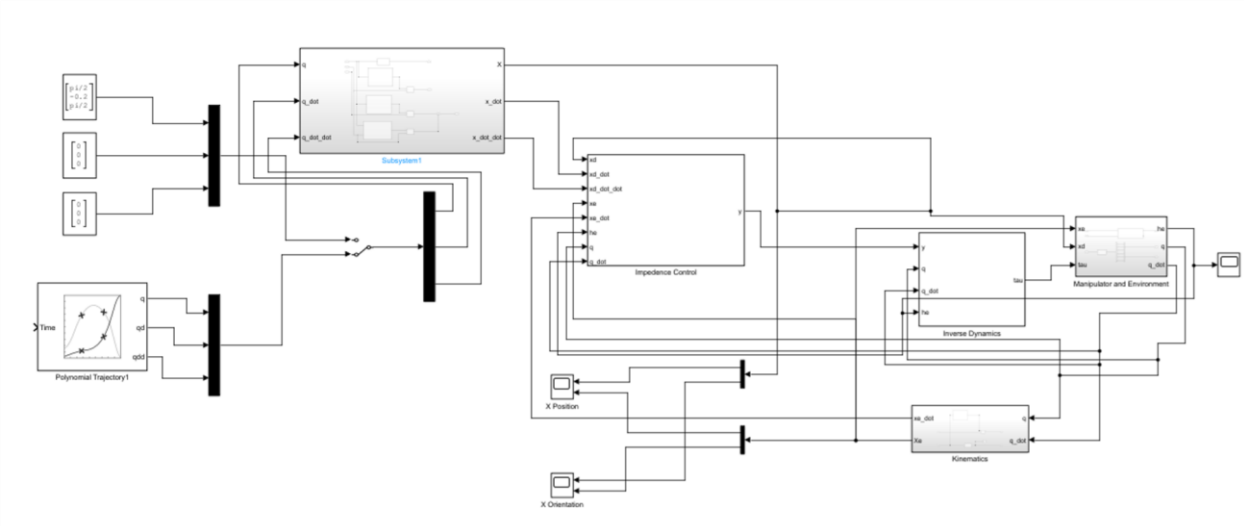


Figure 36 Impedance control schema in Simulink

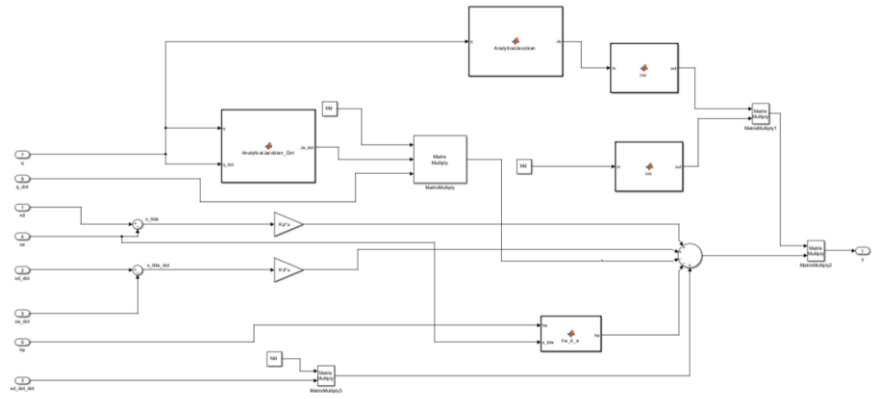


Figure 37 Impedance control block schema

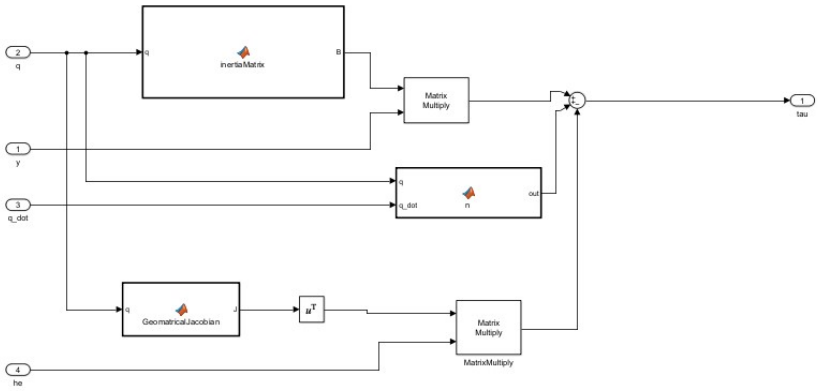


Figure 38 Inverse dynamics schema block

Figures 39 and 40 depict the end-effector of the robot in its desired position and orientation. The robot may be seen following the trajectory while interacting with its surroundings. Due to

the elastic environment that is present in front of the desired frame, there is an inaccuracy in the z-axis and x-axis. Therefore, the location and orientation did not match the desired position and orientation.

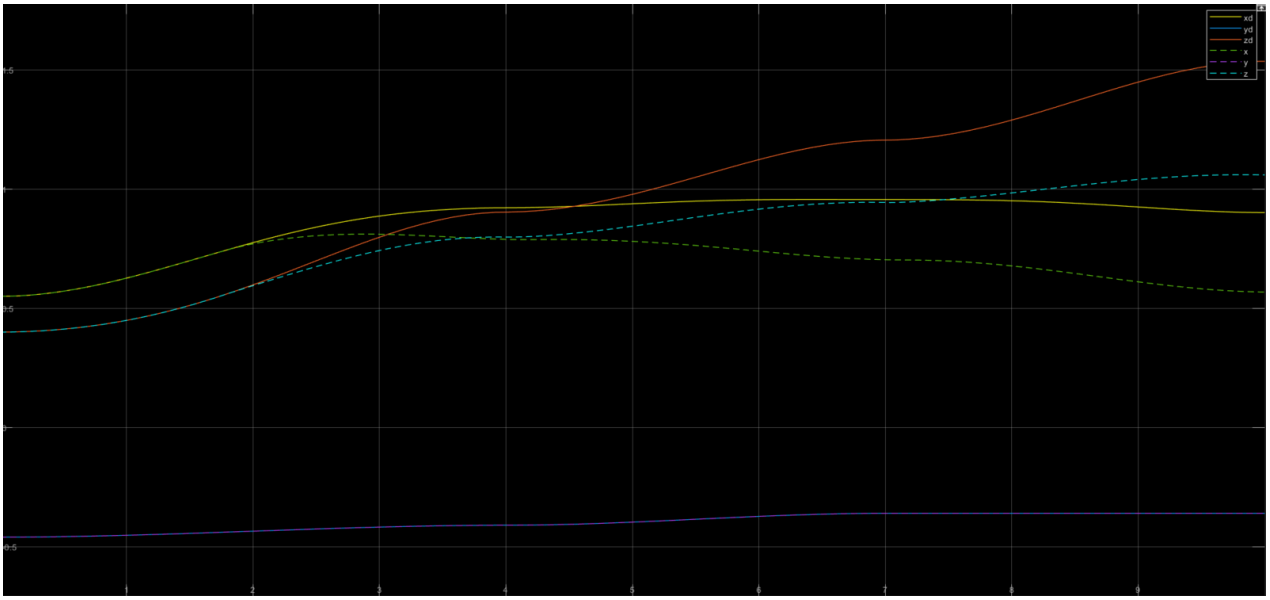


Figure 39 Impedance control position Pe graph.

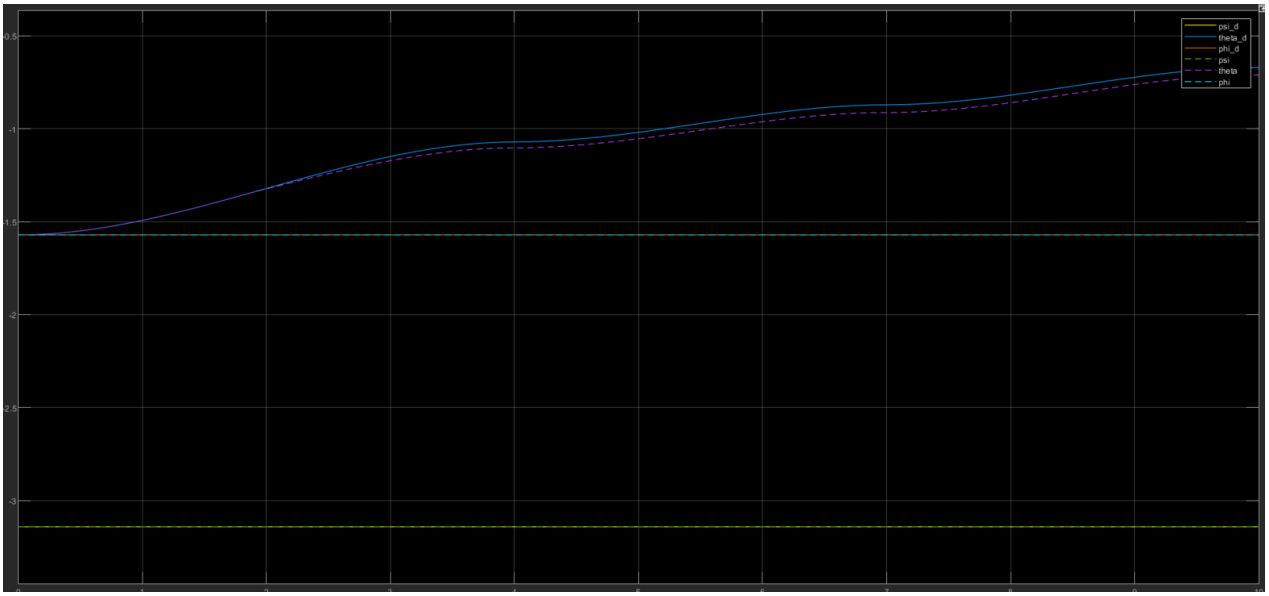


Figure 40 Impedance control position ϕe graph

The Figure 41 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.

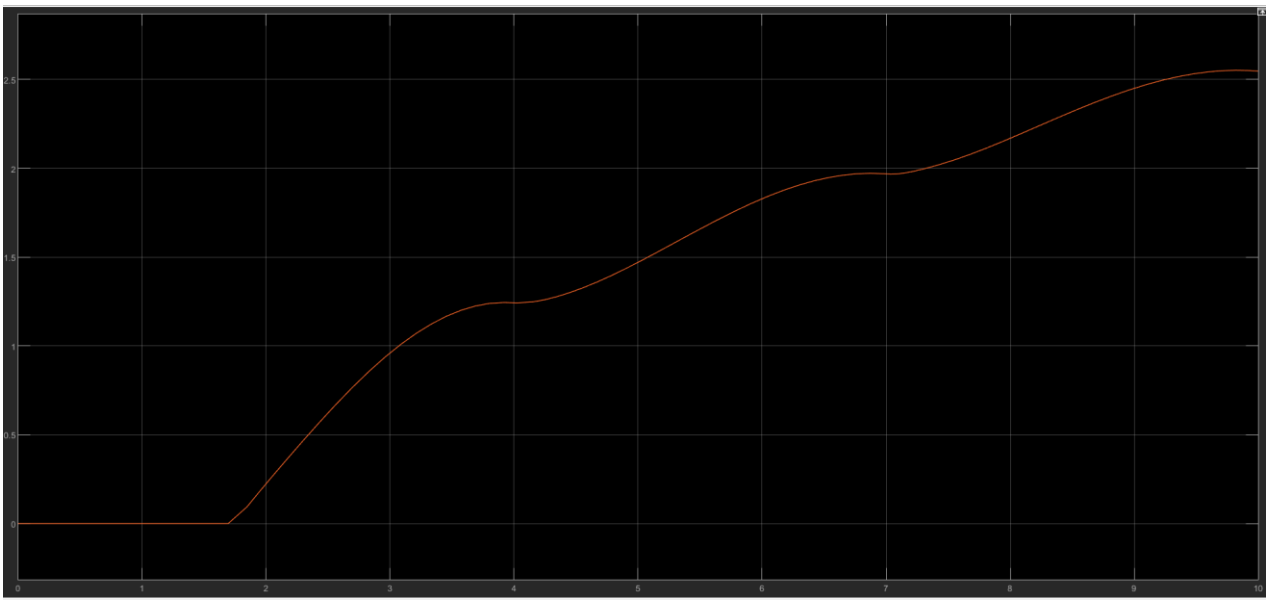


Figure 41 External wrench *he* in impedance control

Admittance Controller

Admittance control is a kind of force control that, like impedance control, enables a robot to respond to outside forces in a compliant manner. The emphasis of admittance control, in contrast to impedance control, is on controlling the robot's motion based on the applied force rather than on controlling the force based on the desired motion. Additionally, the admittance control separates the impedance controller from the motion controller that make the robot more durable to adapt to environmental changes and tolerate environmental disturbances. The block of the admittance controller has been introduced to isolate the motion controller from the impedance controller, however the controller is otherwise displayed in the impedance controller. To establish the appropriate end-effector behavior in the impedance controller, the compliance frame t is introduced to the admittance controller. Mechanical impedance manifests as

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

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

$$\tilde{z} = x_d - x_t$$

The Simulink admittance control schema is displayed in Figure 42. The only difference between the schema and the impedance controller schema was the addition of the admittance block. The admittance control block is shown in Figure 43. It uses an external interaction force to convert the required position and orientation to a compliant frame. However, in this experiment, the control law uses the following variables.

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

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

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

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

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

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

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

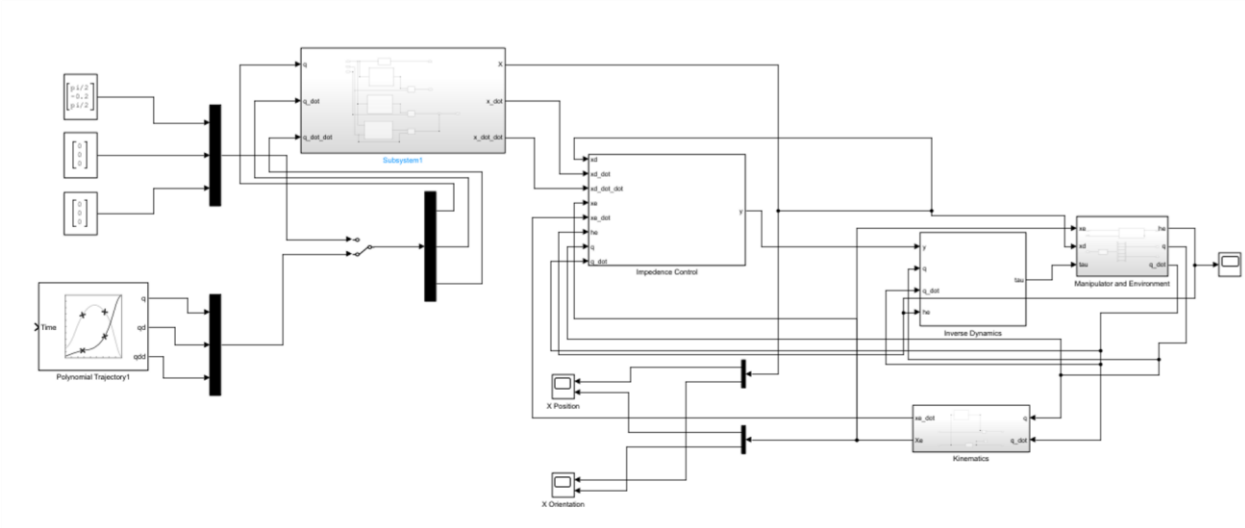


Figure 42 Admittance controller schema in Simulink

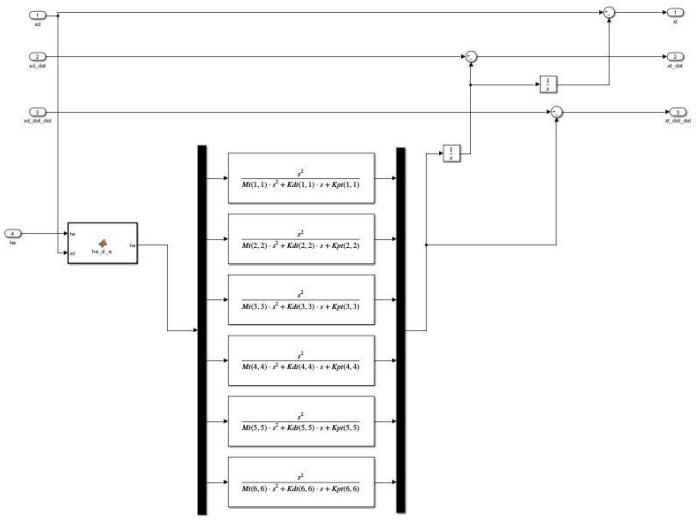


Figure 43 Admittance controller block

The Figure 44 and 45 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.

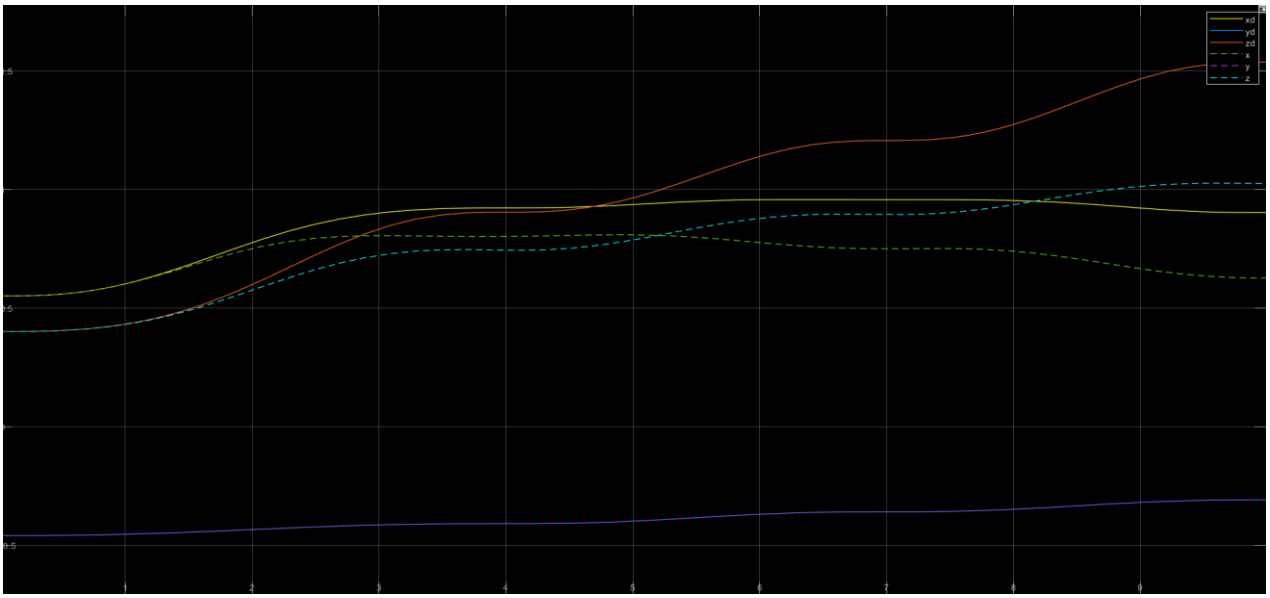


Figure 44 Admittance control position Pe graph

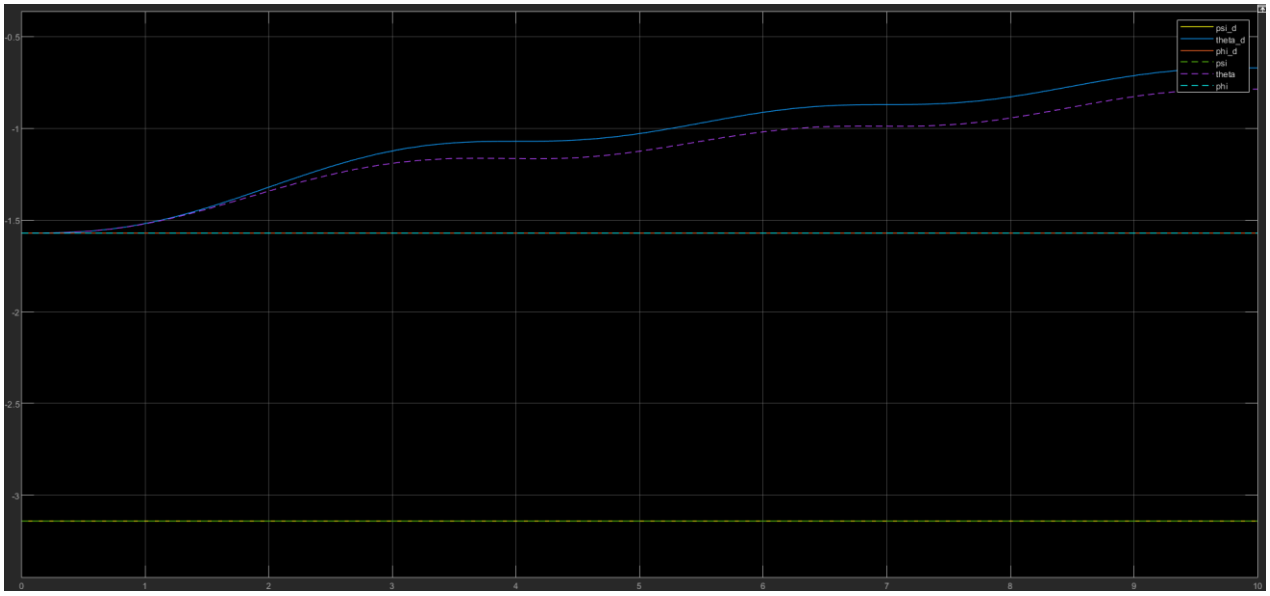


Figure 45 Admittance control orientation ϕe graph

Figure 46 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.

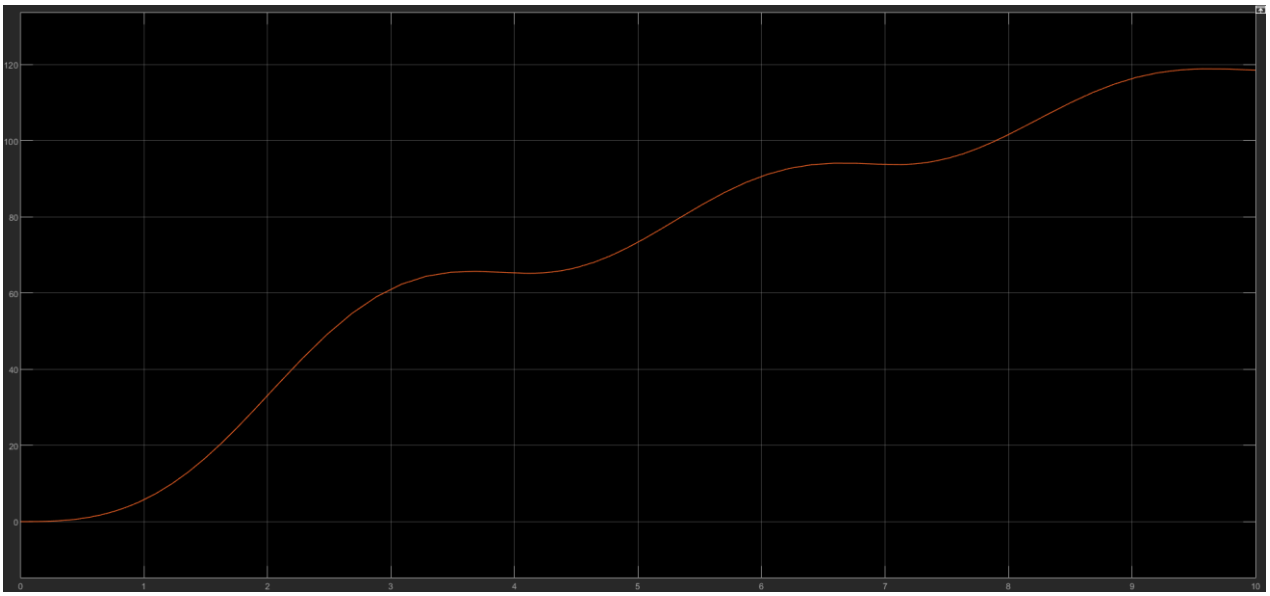


Figure 46 External wrench *he* of Admittance control

Direct Force Control

A direct force controller is a kind of force control technique used in robotics to directly manage the force of contact between the robot and its surroundings. An outside feedback loop is introduced to the system as a tweak to the motion control method, allowing for force regulation in addition to position and velocity regulation. In a direct force control method, the desired contact force is stated as the control input, and the control output is computed using the force error between the desired value and the measured force. The joint locations, velocities, and accelerations are then updated based on the control output to produce the desired contact force. The control gains for the force loop are selected to provide system stability and prompt and precise application of the desired force. Two force controllers, which are detailed below, are employed in this literature.

Force Control

Force feedback loop is used by the direct force controller method, or force controller, to control the desired force f_d. Since forces are inherently specified in operational space, the controller used it to define all of the forces. The force controller, however, makes use of the inverse dynamic controller. As described in the aforementioned sections, the environment is utilized as an elastic system, which

$$f_e = K_e(x_e - x_r)$$

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.

$$J_a(q) = J(q)$$

so force controller ends up in the following control law

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

where

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

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

$$x_f = C_F(f_d - f_e)$$

hence C_F is defined as a PI controller

$$C_F = K_f + K_i\frac{1}{s}$$

and $_d$ 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.

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 outside force feedback loop is depicted in the force controller's overall schema in Figure 47. The impedance control block shown previously identical to the motion control and inverse dynamics blocks. Figure 48 defines the force control block, which transforms the force into the appropriate motion. The PI control strategy for the x_f is depicted in the figure.

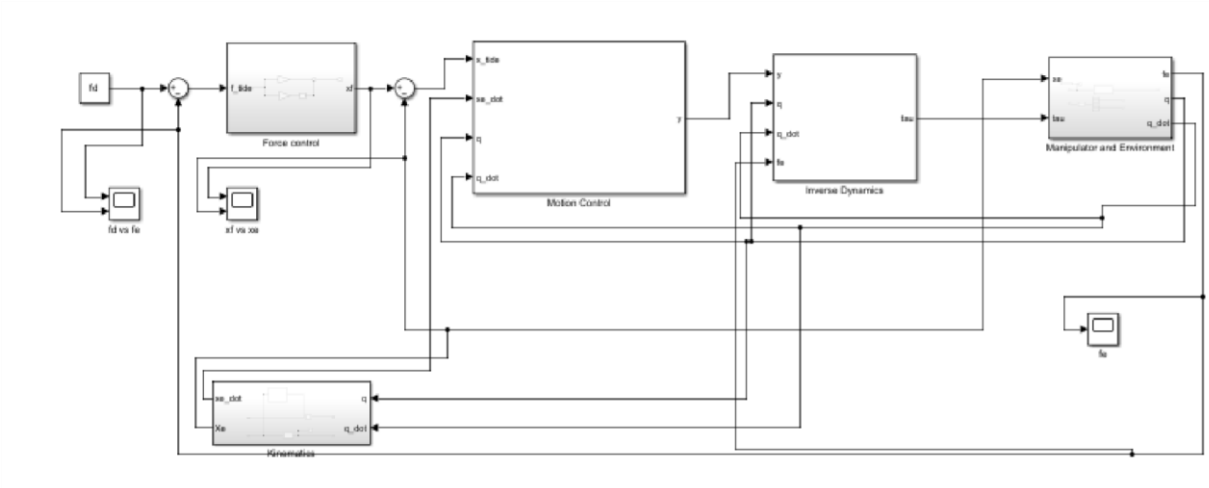


Figure 47 Force control schema in Simulink

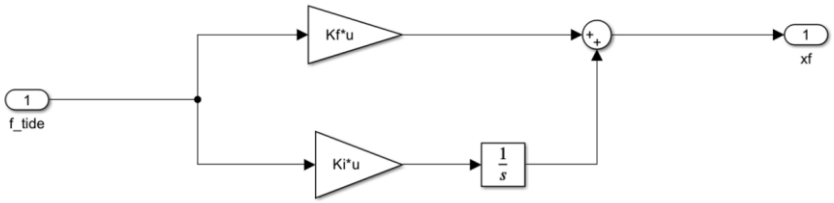


Figure 48 Force control block

When the established parameters are forced into control, the control performs well. As can be seen from Figure 49, the force f_e accurately converges to the intended force f_d. It is crucial to remember that, for the purpose of simplicity, only the required force along the z-axis is set.

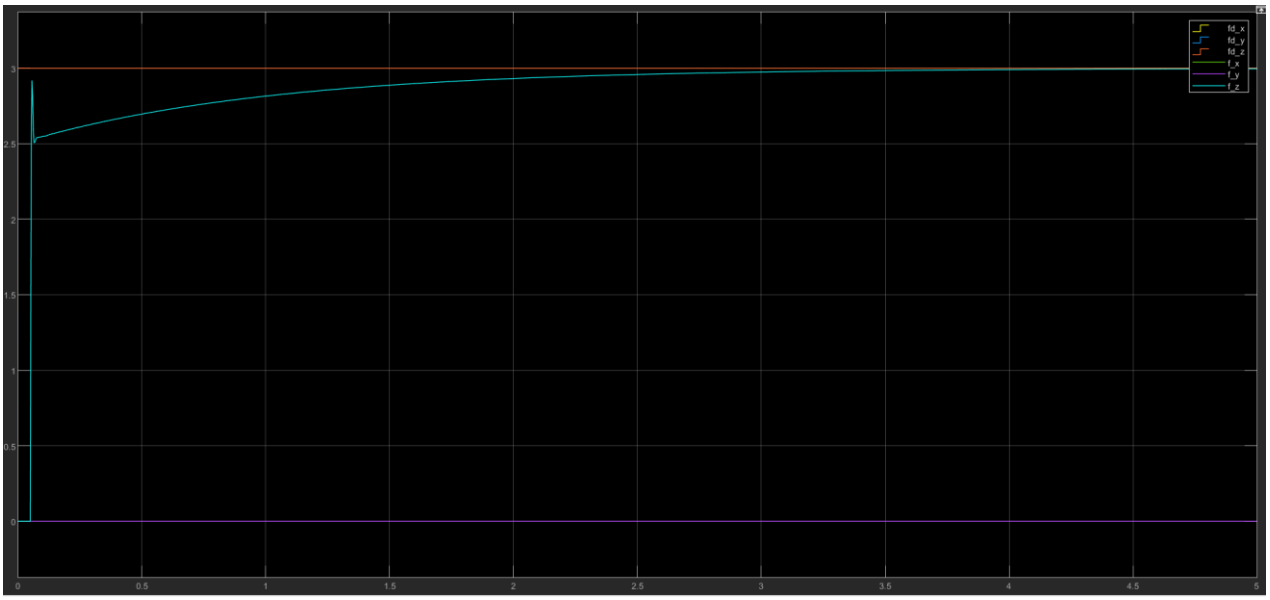


Figure 49 Control schema graph for $fd1$ and fe

Theoretically, it is significant to remember that the controller will move to the relative location x_f if the necessary force is set inside the Image(K_e). As seen in Figure 50, the K_e is placed inside the Image(K_e), and the end-effector's position (x_e) accurately converges to the position (x_f). However, there is some drag in Figure 51 since the f_d is set outside of the Image(K_e).

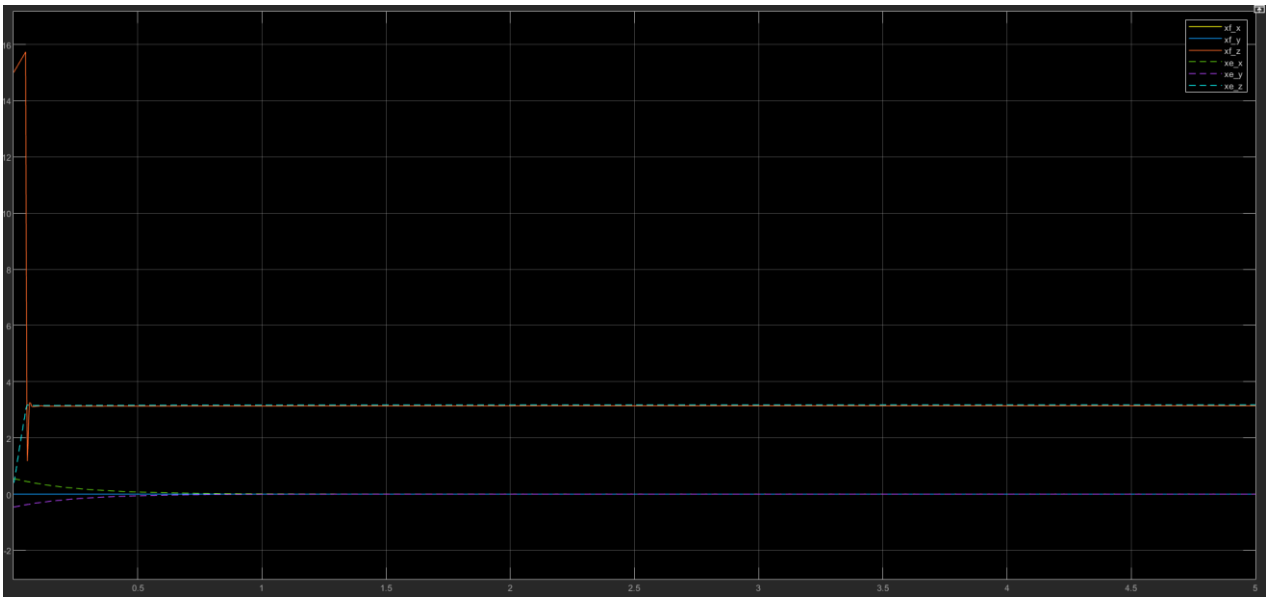


Figure 50 Force control graph between xf and xe for $fd1 = 3$ $inside$ $Imag(Ke)$.

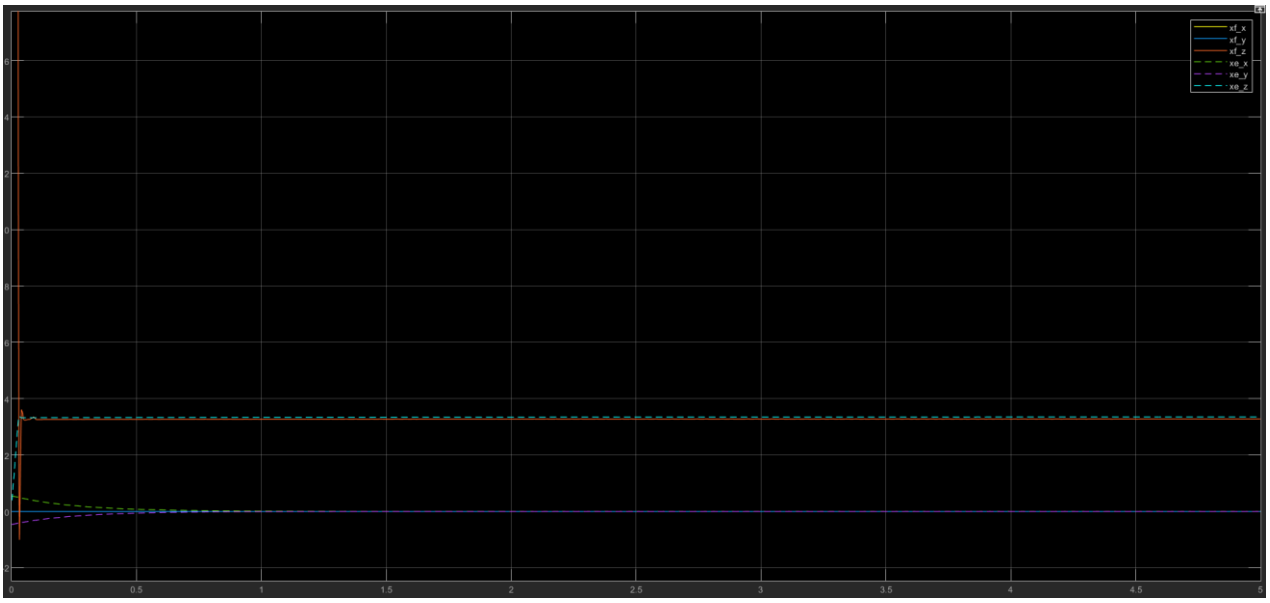


Figure 51 Force control graph between xf and xe for $fd2 = 6$ $outside$ $Imag(Ke)$

Parallel force/position controller

The force control described above is succeeded by the parallel force and position controller. As the desired position is added to the position control loop. In order for the controller to achieve the desired force and position. The end-effector can only move freely in that direction until it reaches the intended point, x. The parallel force and position controller's control law describes

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

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

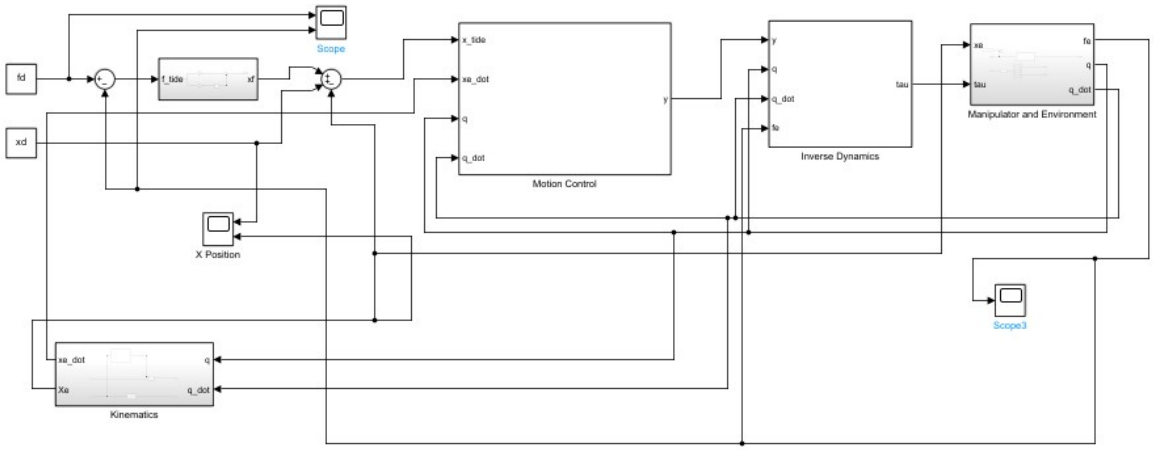


Figure 52 Parallel force/position control schema in Simulink

Figure 53 demonstrates that the controller accurately reached its ideal force. It is significant to remember that only the z-axis is affected by the environment; all other axes are set to zero. Additionally, Figure 54 demonstrates that all positions are precisely obtained, with the exception of the z-position, whose motion is constrained and can be accomplished by the controller in accordance with the previous description.

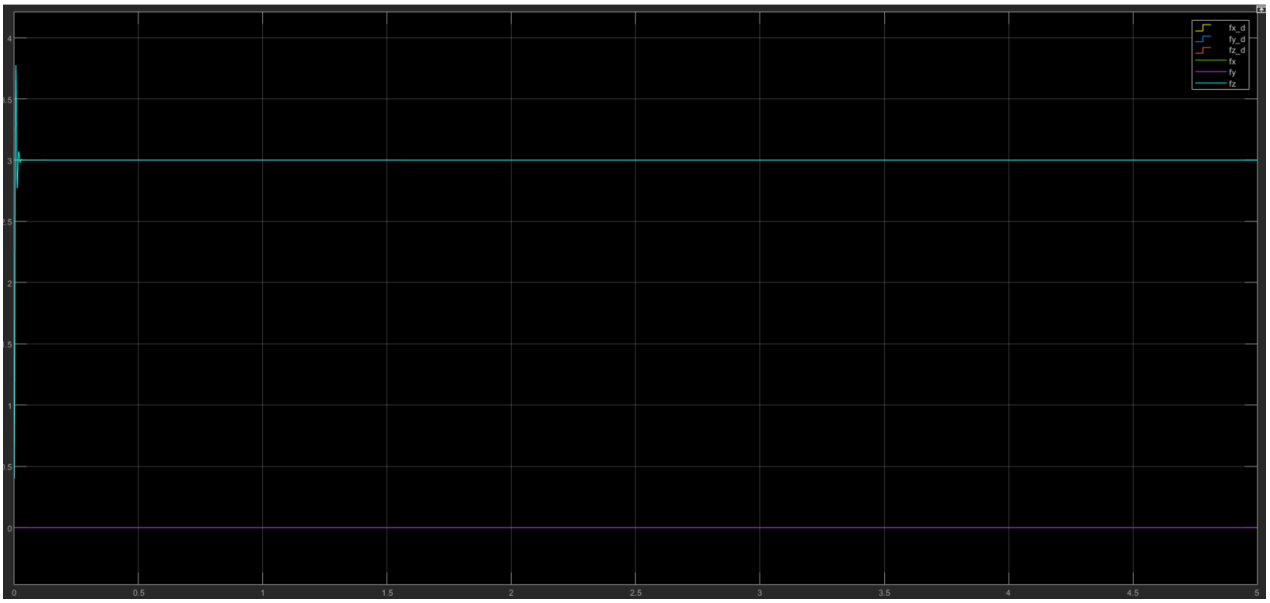


Figure 53 Force graph of parallel force and motion controller

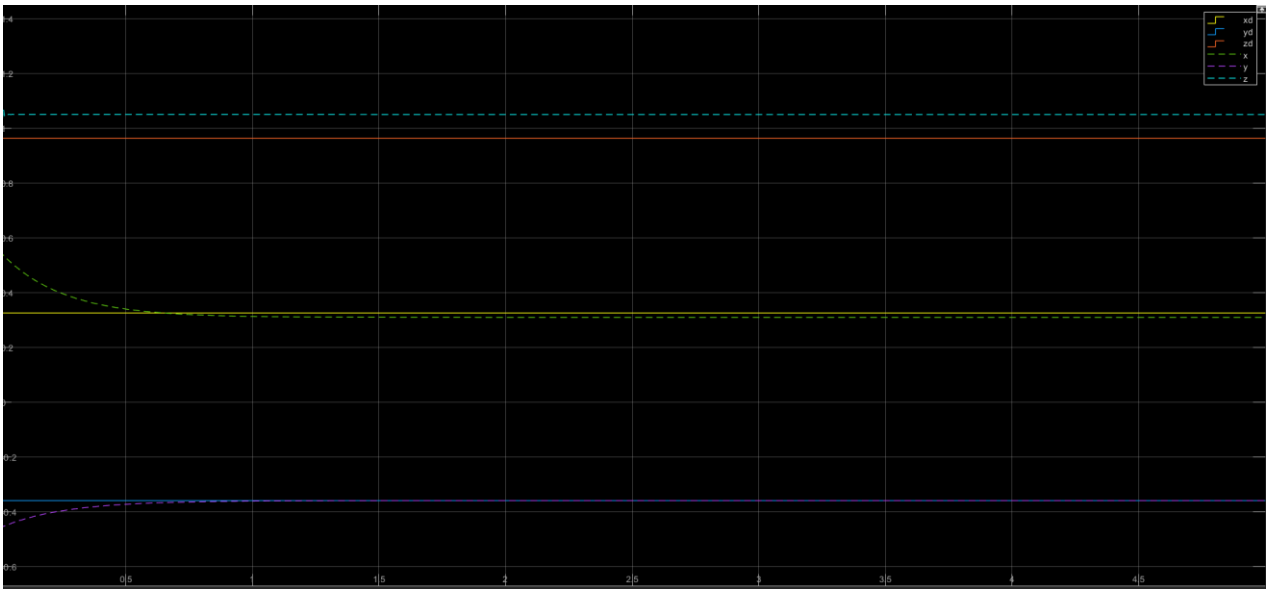


Figure 54 Position of the controller

Conclusion

In conclusion, a fixed-base robot with three degrees of freedom is presented, consisting of two prismatic joints and one revolut joint. Using MATLAB tools, analyzed the robot's kinematics, differential kinematics, dynamics, and control. The robot was modeled using the Denavit-Hartenberg convention, and its forward and inverse kinematics were calculated with transformation matrices and symbolic equations, verified using the MATLAB Robotic Toolbox.

In the differential kinematics section, the end-effector velocities were examined using geometrical and analytical Jacobians, yielding precise results. Both the Lagrange and Newton-Euler methods were employed to determine the robot's dynamics, showing broadly equivalent accuracy.

Various controllers, including PD control, inverse dynamics control, adaptive control, force control, compliance control, impedance control, admittance control, and parallel force/position control, were developed for the robot in both joint space and operational space. MATLAB Simulink simulations assessed each controller's performance for tasks such as trajectory tracking and force exertion, evaluating their effectiveness and accuracy, as well as their response to disturbances.

Overall, this work provides a comprehensive analysis of the 3 DoF robot manipulator using multiple modeling and control methodologies. It offers insights into the performance of different controllers under various conditions. The Robotics Toolbox and MATLAB tools enabled accurate analysis and validation, proving their value in robotics research. Notably, the inverse dynamics controller performed well under disturbances, while the adaptive controller excelled without an accurate dynamic model. Operational space controllers were found to be more intuitive and easier to operate compared to joint space controllers, and direct force controllers were simpler and more intuitive than indirect force controllers.