Dynamic Modeling and Control of a Multi- Prismatic Robotic System

Modeling and Control of Robots (MAE 547)

Final Project Report

TEAM MEMBERS

Avinash Bharadwaj Margabandhu

Sobha Srujana Patri

Jonathan Reggie Ebenezer

Chandra Venkata Sai Bhagavan Kommu

Naga Venkata Dheeraj Chilukuri

Table of Contents

Abstract

This project aids in performing an in-depth analysis of the dynamic nature and control systems of a robotic system consisting of n- prismatic joints. Coding is performed using MATLAB to derive the governing equations of motion for a n-prismatic link robotic system. The dynamic nature of the robotic system in analysed by plotting *q, q_dot, q_ddot* as a function of time. The robotic system is subjected to indirect force control under compliance and impedance control. The corresponding plots of the end effector positions as well as forces acting on it are made using MATLAB. A user-friendly graphic user interface (GUI) is built enabling dynamic and easy handling of the codes associated with the prismatic joints of a robotic system. It enables the user to change joint variables and observe real-time changes in the system.

INTRODUCTION:

Robotic arms play a pivotal role in a wide range of applications. The capability to understand a robotic system, the dynamics and controls involved in it are extremely critical for safe handling. This report encompasses the dynamic nature and controls of a n-link prismatic robotic system such that the user can understand how to achieve precise positioning of end effector and interaction with forces externally in nature.

METHODOLOGY

Analytical Derivations: The joint variables associated with an n-link prismatic joint have an impact on the position, orientation of end effector. Theoretically these derivations include deriving equations of motion. These equations of motion are derived leveraging on the Denavit-Hartenberg parameters (DH). In this the user specifies the lengths, twists, offsets of the links. Based on these parameters input by the user, the geometric orientation of the robot is evaluated.

The transformation matrices obtained from the process are used to evaluate the Jacobians of the motors and links associated with the robotic system. These are then applied to the mathematical processes to identify the inertial, centrifugal, Coriolis and gravity effects to evaluate the equations of motion associated the robotic system with n-prismatic links.

Dynamics Simulation: Once the user is aware of the dynamics associated with the robotic arm, they can visualize the system's behavior. This includes applying appropriate codes in MATLAB to simulate the dynamics of the robotic arm over time. The results provide the positions, velocities and accelerations of the joints. This aids the user to understand the response of the robotic arm for different joint variables.

Control Systems: This report provides support for performing two forms of indirect force control systems. Namely, compliance control and impedance control. The compliance control system consists of a proportional-derivative (PD) control system with gravity compensation. It adjusts the position of the end effector considering the gravitational forces and other interactions in the environment based on the desired and actual positions of the end effector. This control system helps with the flexible behavior of the robotic arm. The impedance control system assists with precise positioning. It regulates the damping and stiffness of the robotic system. Using the code corresponding to impedance control, the user can compute joint velocities to reduce deviations in desired and actual positions.

Graphical User Interface (GUI): The developed graphical user interface enables the user to provide inputs in terms of DH-Parameters, properties of systems, visualize simulations and monitor the performance in real-time.

IMPLEMENTATION

Analytical Derivations and Dynamics Simulations: The robotic arm is modelled on MATLAB using the DH-Parameters. The user can provide the input of the number of links of the robotic system, the DH-Parameters associated with the each of the links. The code assists the user to identify the transformation matrices, the Jacobians associated with each of the links and motors and evaluates the equations of dynamics of motion. In this report, we explain the implementation using the test case of a 3-links prismatic robotic arm. We use the following DH-Parameters for testing and implementation of the code.

Link	a(i)	alpha(i)	d(i)	theta(i)
1	1	0	4	0
2	2	-90	5	0
3	3	0	6	0

Table 1: DH-Parameters

The code can be found in Appendix. On providing the necessary inputs from the user such as DH-Parameters, mass and inertia of links and motors and gear ratio of motors, the code provides the output consisting of transformation matrices, position vectors, Jacobians of motors and links, matrices with inertial and gravity effects. As a final output, we get torque equations along with the plots of joint positions, velocities and accelerations with respect to time.

```
Command Window
Enter the number of prismatic joints: 3
Enter link length a1: 1
Enter link twist alpha1 (degrees): 0
Enter link offset d1: 4
Enter joint displacement theta1 (degrees): 0
Enter link length a2: 2
Enter link twist alpha2 (degrees): -90
Enter link offset d2: 5
Enter joint displacement theta2 (degrees): 0
Enter link length a3: 3
Enter link twist alpha3 (degrees): 0
Enter link offset d3: 6
Enter joint displacement theta3 (degrees): 0
Enter mass of link 1 (kg): 1
Enter inertia vector [Ixx Iyy Izz] for link 1: 2
Enter mass of link 2 (kg): 1
Enter inertia vector [Ixx Iyy Izz] for link 2: 2
Enter mass of link 3 (kg): 1
Enter inertia vector [Ixx Iyy Izz] for link 3: 2
Enter mass of motor 1 (kg): 1
Enter inertia vector [Ixx Iyy Izz] for motor 1:
2
Enter gear ratio for motor 1: 0.5
Enter mass of motor 2 (kg): 1
Enter inertia vector [Ixx Iyy Izz] for motor 2:
2
Enter gear ratio for motor 2: 0.5
Enter mass of motor 3 (kg): 1
Enter inertia vector [Ixx Iyy Izz] for motor 3:
2
Enter gear ratio for motor 3: 0.5
```

Figure 1: Input Parameters

```
Transformation Matrices:
T01:
[1, 0, 0, 1]
[0, 1, 0, 0]
[0, 0, 1, 4]
[0, 0, 0, 1]

T02:
[1,  0, 0, 3]
[0,  0, 1, 0]
[0, -1, 0, 9]
[0,  0, 0, 1]

T03:
[1,  0, 0, 6]
[0,  0, 1, 6]
[0, -1, 0, 9]
[0,  0, 0, 1]

Z-axes:
Z0:
0
0
1

Z1:
0
0
1

Z2:
0
1
0

Z3:
0
1
0

Midpoints (PL):
PL1:
1/2
  0
  2

PL2:
3/2
  0
9/2

PL3:
  3
  3
9/2

Motor Positions (PM):
PM1:
1
0
4

PM2:
3
0
9

PM3:
6
6
9

Linear Jacobian (Jp):
Jp for Joint 1:
     0     0     0
     0     0     0
     1     0     0

Jp for Joint 2:
     0     0     0
     0     0     0
     1     1     0

Jp for Joint 3:
     0     0     0
     0     0     1
     1     1     0

Other Jacobian for Motors (Jom):
Jom for Motor 1:
         0         0         0
         0         0         0
    0.5000         0         0

Jom for Motor 2:
         0         0         0
         0         0         0
         0    0.5000         0

Jom for Motor 3:
         0         0         0
         0         0    0.5000
         0         0         0
```

Figure 2: Output1.

```
B matrix:
[11/2,   3,   0]
[   3, 7/2,   0]
[   0,   0, 3/2]

g matrix:
[49, 147/5, 0]
[49, 147/5, 0]
[49, 147/5, 0]

Tou matrix:
[(11*ddot_d1)/2 + 3*ddot_d2 + 49, (11*ddot_d1)/2 + 3*ddot_d2 + 147/5, (11*ddot_d1)/2 + 3*ddot_d2]
[ 3*ddot_d1 + (7*ddot_d2)/2 + 49,  3*ddot_d1 + (7*ddot_d2)/2 + 147/5,  3*ddot_d1 + (7*ddot_d2)/2]
[            (3*ddot_d3)/2 + 49,             (3*ddot_d3)/2 + 147/5,             (3*ddot_d3)/2]

Tou_1:
(11*ddot_d1)/2 + 3*ddot_d2 + 49

Tou_2:
3*ddot_d1 + (7*ddot_d2)/2 + 49

Tou_3:
(3*ddot_d3)/2 + 49
```

Figure 3: Equations of Motion (output)

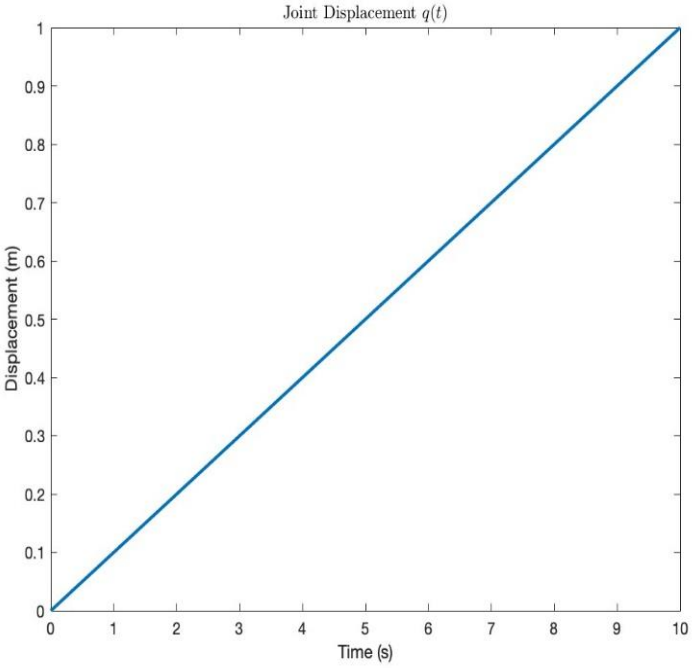

Figure 4: Joint Displacement vs Time

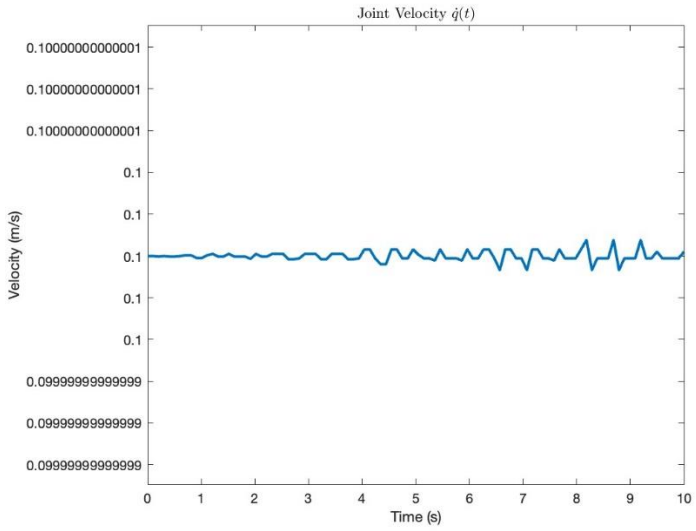

Figure 5: Joint Velocity vs Time

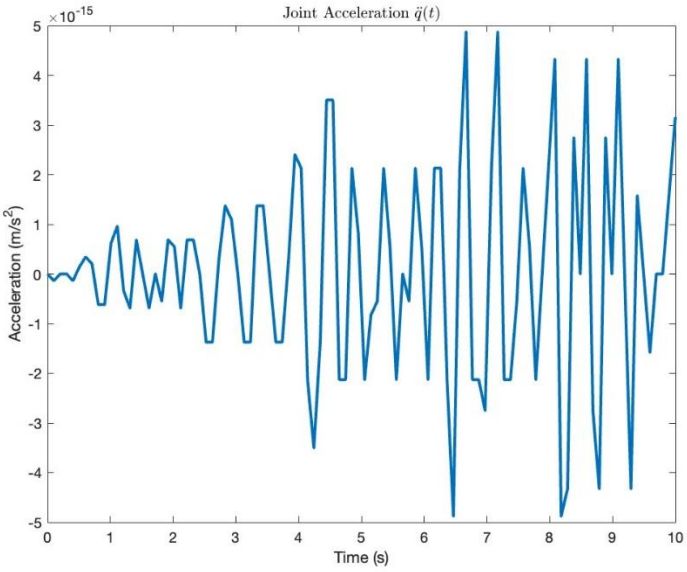

Figure 6: Joint Acceleration vs Time

Compliance Control: For performing compliance control, the user can give an input of proportional and derivative gain values along with the desired end effector position. On applying PD control with gravity compensation, we can perform compliance control and get the plots of desired vs actual end effector positions as well as the contact forces.

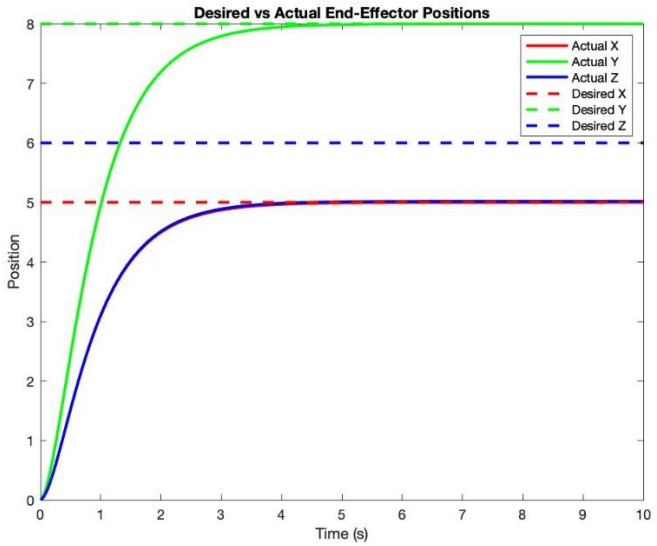

Figure 7: Desired vs Actual End-Effector Positions (Compliance Control)

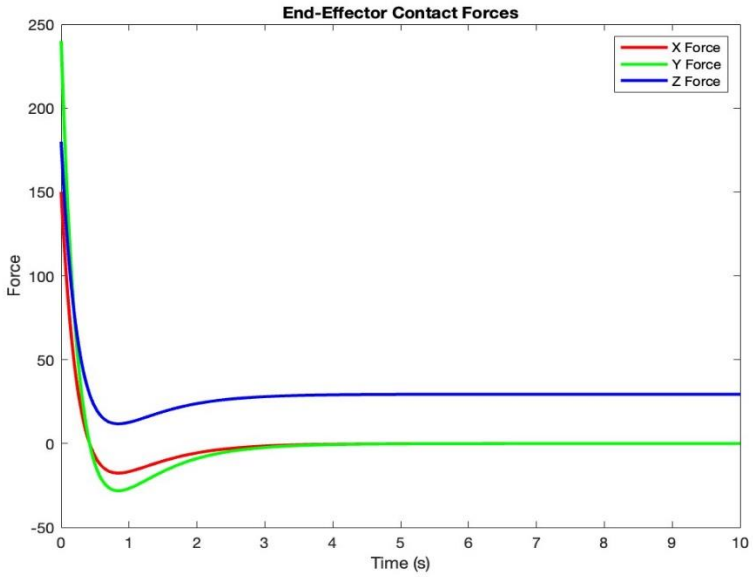

Figure 8: End Effector Contact Forces (Compliance Control)

Impedance Control: For performing impedance control, the user can give input for proportional and derivative gains along with any other mechanical forces being applied on the joints. On applying impedance control, the user can plot desired and actual end effector positions as well as contact forces.

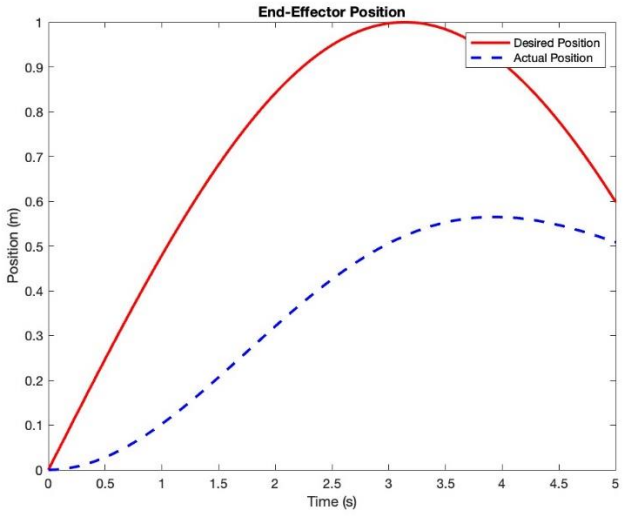

Figure 9: Desired vs Actual End Effector Position (Impedance Control)

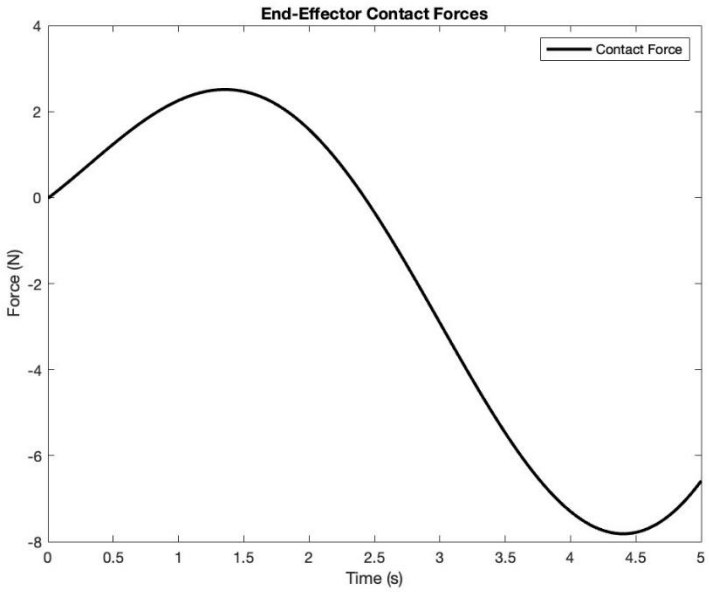

Figure 10: End-Effector Contact Forces (Impedance Control)

Graphical User Interface (GUI): Grid layout, push buttons, table and edit numerical elements are used to generate the graphical user interface with functions and call back functions integrated into the script. The user can provide all the necessary input parameters using the easy-to-understand interface and access scripts to obtain the necessary plots.

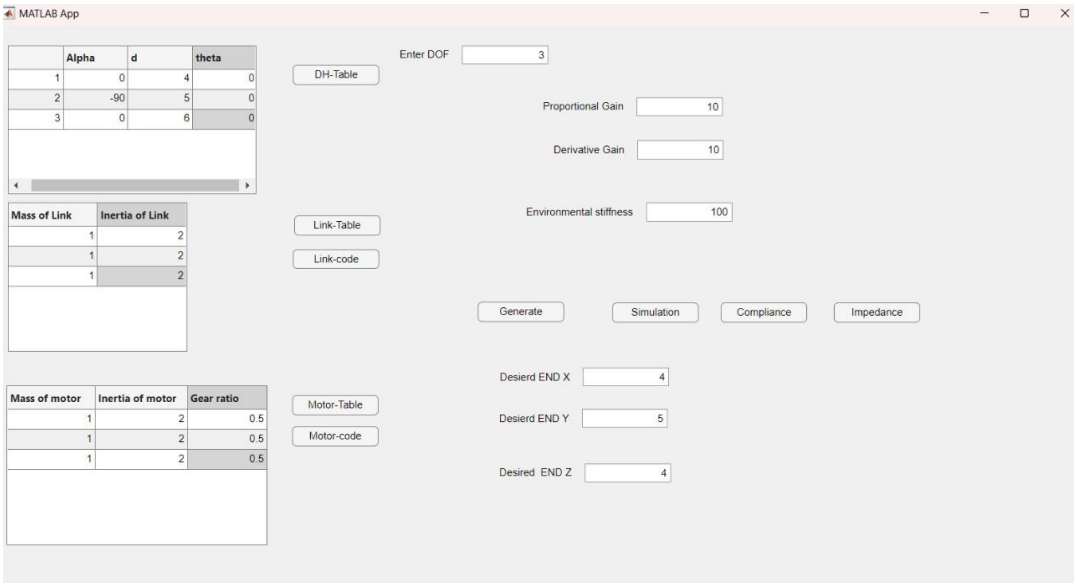

Figure 11: Graphical User Interface

RESULTS

Code Results:

With the input parameters inputted by the user(includes, Number of Links, DH Parameters, mass of link, mass of motor, inertia of motor, inertia of link, Gear ratio of motor, Proportional Gain for Impedance and its derivative, Proportional Gain for Compliance and its derivative, Environmental stiffness for Impedance and Compliance) through the GUI implemented through the AppDesigner Platform, we were able to create the Equation of motion (Using the Jacobians of the links and motor, Inertial effect, Centrifugal Coriolis effects, gravity effect, and generalized torque effects). Additionally, we were able to simulate the system dynamics, Impedance control, and the compliance control with its corresponding plots.

Plot Results:

- The plots are position, velocity, acceleration of the joints as a function of time.
- End-effector desired VS Actual for both impedance and compliance.
- Environmental forces as function of time for both Impedance and Compliance.

Learning Outcomes:

Deriving and outputting the governing equations of motion analytically, as taught in class. This involves understanding the underlying principles of the system's dynamics and expressing them mathematically. Simulating the dynamics of the system, including plotting the joint positions (q), velocities (q_dot), and accelerations (q_ddot) for a desired input over time. Implementing indirect force control through compliance control, specifically PD control with gravity compensation, to interact with the environment. This entails plotting the desired versus actual

end-effector position and contact forces over time. Implementing indirect force control through impedance control, employing inverse dynamics control to interact with the environment. This involves plotting the desired versus actual end-effector position and contact forces over time. Designing an easy-to-use and intuitive graphical user interface (GUI) using the AppDesigner Platform, facilitating user interaction with the system. Producing a clear and detailed report that serves as a step-by-step instruction manual, outlining each team member's contributions and providing relevant references.

Appendix 1

Code for analytical derivations and simulations:

https://drive.google.com/file/d/1XL_d-cU3z-H4XpKa_goEgYigedBOhTpz/view?usp=drive_link

Code for compliance control:

https://drive.google.com/file/d/1ZkEsnx8qCC9j1HOhiAIyoqV-R6KIYcOt/view?usp=drive_link

Code for impedance control:

https://drive.google.com/file/d/19BD7_hoRlDsmyD5pMjoVAlRe_R8pqm92/view?usp=drive_link

Code for GUI:

https://drive.google.com/file/d/1H-bbEi9oxoWtuYU4oSaoh9IfttXGc67b/view?usp=drive_link