Designing a Control Architecture for Phantom Premium 1.5

Yakup Can Karacaoğlu
Koç University Engineering Department
ykaracaoglu18@ku.edu.tr

I. INTRODUCTION

This project aims to develop a dynamical model and design a control architecture for the PHANToM Premium 1.5 haptic robotic arm using Simulink. The objective is to enable precise movement of the arm's end-effector along a straight path, while compensating for positional and velocity errors. To achieve this, a PID controller will be utilized, allowing the end-effector to follow a trajectory represented by a line segment in a three-dimensional space. The project focuses on establishing a robust control system that can accurately manipulate the robotic arm.

II. DISCUSSION

A. *Control Design*

For the control of the robot a closed loop control architecture is utilized. In Figure 1, an overview of the design architecture is displayed from Simulink. In the loop there are four main steps, firstly a path is generated for robot to track it, secondly a forward kinematics section to calculate current end effector position, thirdly a PID gain section to calculate an output force according to current tracking error, and Jacobian-dynamic properties section to calculate current angles according to applied forces and previous angle.

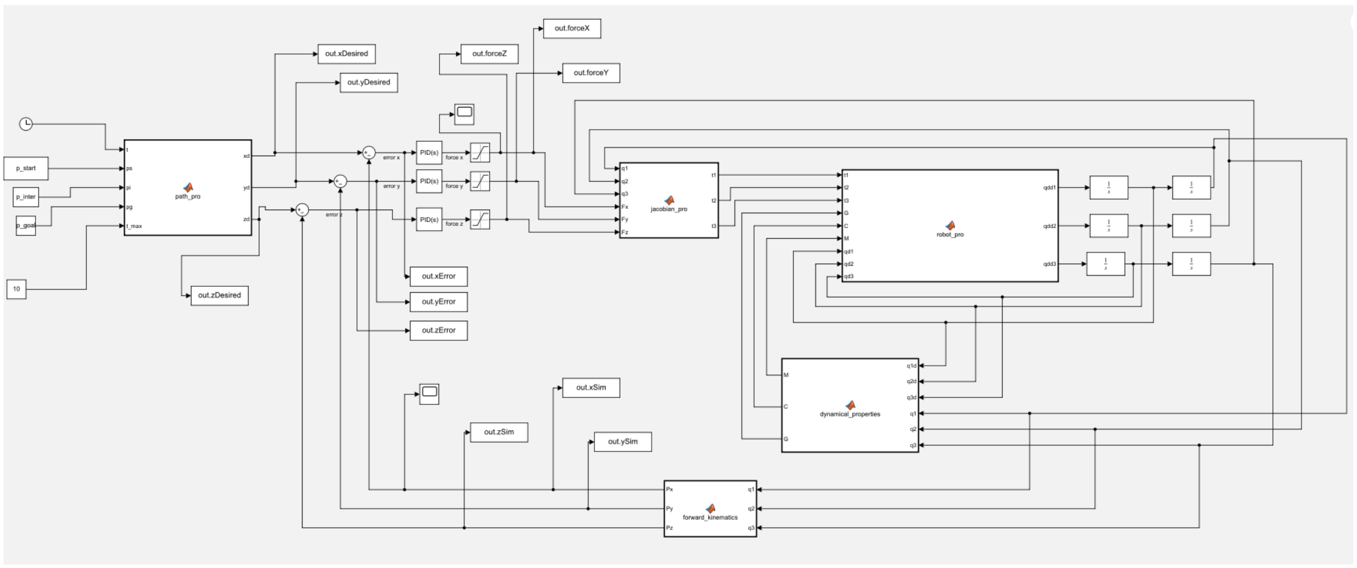

Figure 1 Control Design Overview

B. *Function Definitions*

1) Path Generator (path_pro)

This function creates a two-stage linear path that starts from a given start point, moves to intermediate point, reaches and stays at the end point for a while. The algorithm separates the path into two line segments and creates 2000 sized line spaces for every segment. It holds information of every x, y, z points in different arrays. The algorithm takes points information of start, intermediate and final from workspace as input. It also takes clock of the simulation and in every clock step it gives desired points from the created arrays. After reaching the final time of the movement (it also given as input), it keeps giving the final point information until the simulation is completed.

2) Jacobian Function (jacobian_pro)

This function calculates the torques applied to joints according to the force applied on the end effector according to equation 1. The Jacobian is calculated according to the end effector position which is given in the assignment description. The elements of the Jacobian matrix are represented in equation 2 to equation 10. The function basically takes current applied forces and previous time step's theta values to calculate current torque values.

$$\tau = J^T F \tag{1}$$

$$\frac{\partial P_x}{\partial \theta_1} = \cos(\theta_1)\,(L_1 \cos\theta_2 + L_2 \sin\theta_3) \tag{2}$$

$$\frac{\partial P_x}{\partial \theta_2} = -L_1 sin\theta_1 sin\theta_2 \tag{3}$$

$$\frac{\partial P_x}{\partial \theta_3} = L_2 sin\theta_1 cos\theta_3 \tag{4}$$

$$\frac{\partial P_y}{\partial \theta_1} = 0 \tag{5}$$

$$\frac{\partial P_y}{\partial \theta_2} = L_1 cos\theta_2 \tag{6}$$

$$\frac{\partial P_y}{\partial \theta_3} = L_2 sin\theta_3 \tag{7}$$

$$\frac{\partial P_z}{\partial \theta_1} = -\sin(\theta_1)\,(L_1 \cos\theta_2 + L_2 \sin\theta_3) \tag{8}$$

$$\frac{\partial P_z}{\partial \theta_2} = -L_1 cos\theta_1 sin\theta_2 \tag{9}$$

$$\frac{\partial P_z}{\partial \theta_3} = L_2 sin\theta_3 \tag{10}$$

3) Forward Kinematics

It is a straightforward function block to calculate the current end effector position according to current theta values. The function basically uses given end effector formulas in the transformation matrix of the robot. The end effector positions are from equation 11 to equation 13.

$$P_x = sin\theta_1(L_1 \cos\theta_2 + L_2 \sin\theta_3) \tag{11}$$

$$P_y = L_2 - L_2 cos\theta_3 + L_1 sin\theta_2 \tag{12}$$

$$P_z = -L_1 + cos\theta_1(L_1 \cos\theta_2 + L_2 \sin\theta_3) \tag{13}$$

4) Dynamic Proporties (dynamic_properties) and Robots Dynamic Equation (robot_pro)

The dynamic properties function was given and it gives inertia and Coriolis matrices, as well as the gravity vector of the robotic arm according to current theta values and their derivatives. The user-defined robot_pro function incorporated the inertia and Coriolis matrices, and the gravity vector. By utilizing this information, the function mapped the applied torque to each joint of the robotic arm to the corresponding vector of joint angles. This allowed for precise control of the arm's movement based on the input torque. The user defined function is created according to the dynamic equation from the class's textbook [1]. The equation 6.63 of the book suggests that the applied torque can be calculated according to mass, Coriolis, centrifugal, and gravitational dimensions. It is called "configuration-space equation" and represented in equation 14.

$$\tau = M(\theta)\ddot{\theta} + B(\theta)\left[\dot{\theta}\dot{\theta}\right] + C(\theta)\left[\dot{\theta}^2\right] + G(\theta) \qquad (14)$$

The user define function ignores the Coriolis term B, since the B matrix is not provided, it is assumed that it can be ignored. The equation to calculate $\ddot{\theta}^2$ is,

$$\ddot{\theta} = M^{-1} * (\tau - C\dot{\theta}^2 - G) \qquad (15)$$

Eventually, the second derivatives of the thetas are calculated according to equation 15.

C. Results

As result of the implementation the graph of the force outputs, desired and simulated end effector trajectories, tracking errors for x-y-z are displayed from Figure 2 to Figure 10.

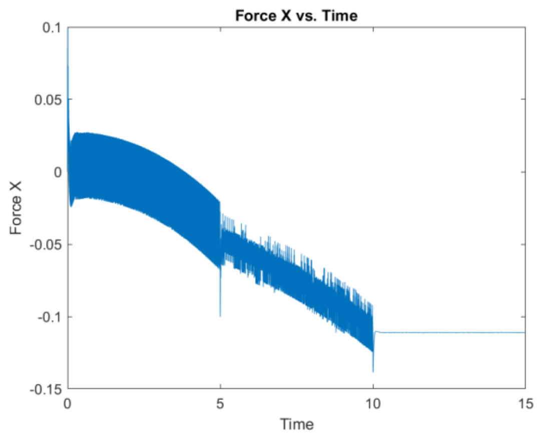

Figure 2 Force on X Direction vs Time

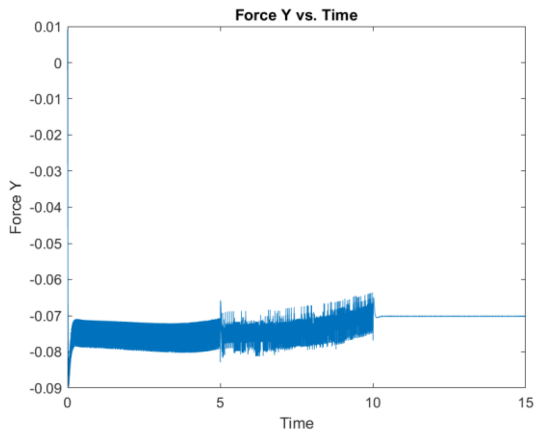

Figure 3 Force on Y Direction vs Time

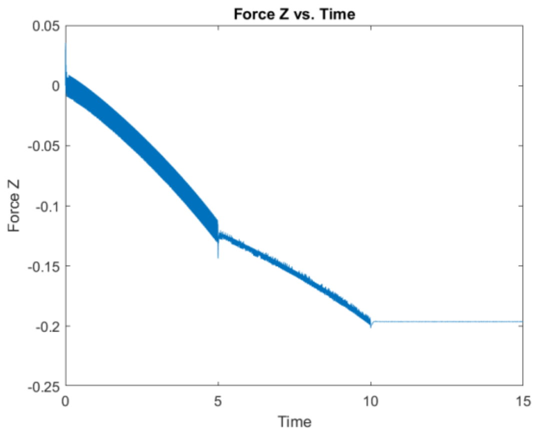

Figure 4 Force on Z Direction vs Time

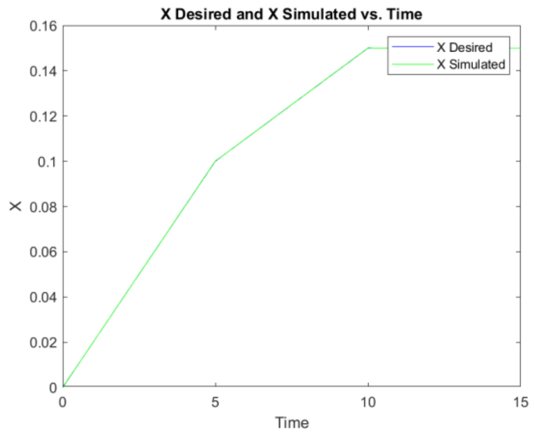

Figure 5 Desired and Simulated Path on X vs Time

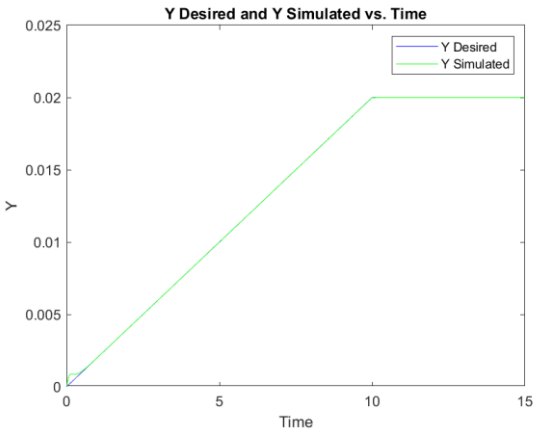

Figure 6 Desired and Simulated Path on Y vs Time

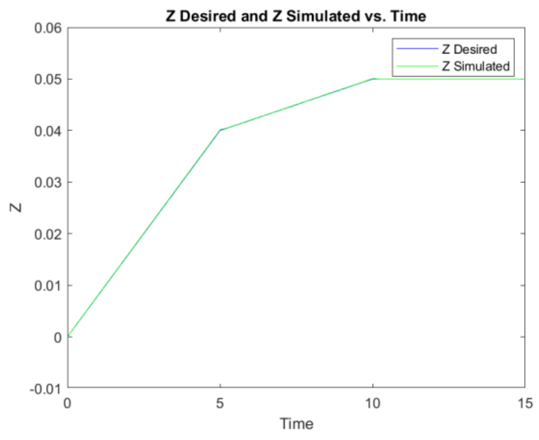

Figure 7 Desired and Simulated Path on Z vs Time

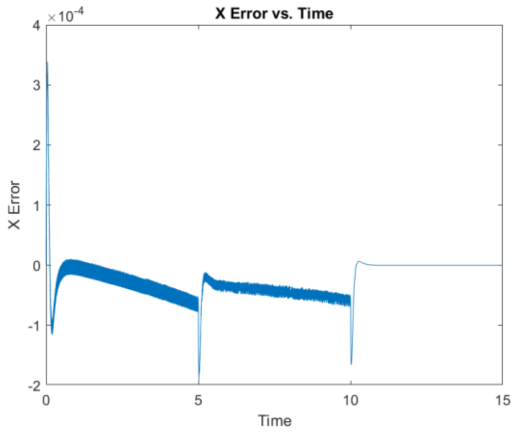

Figure 8 Tracking Error X vs Time

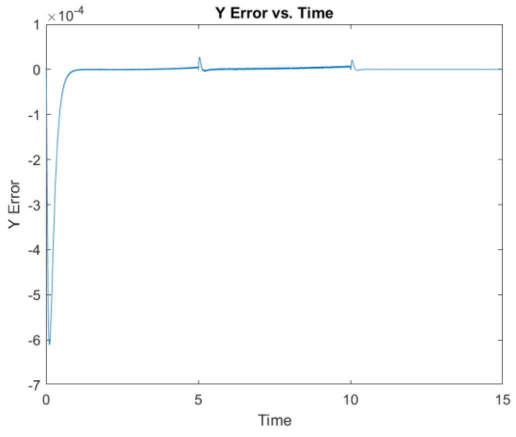

Figure 9 Tracking Error Y vs Time

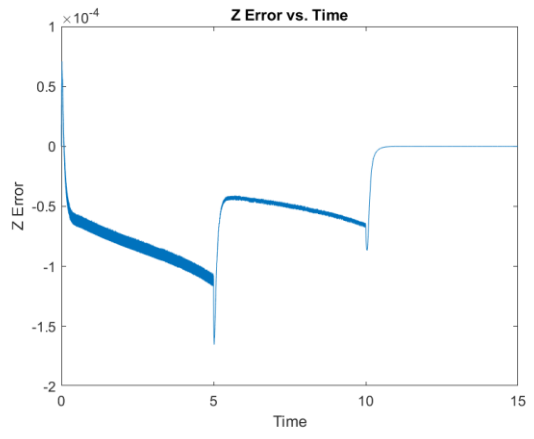

Figure 10 Tracking Error Z vs Time

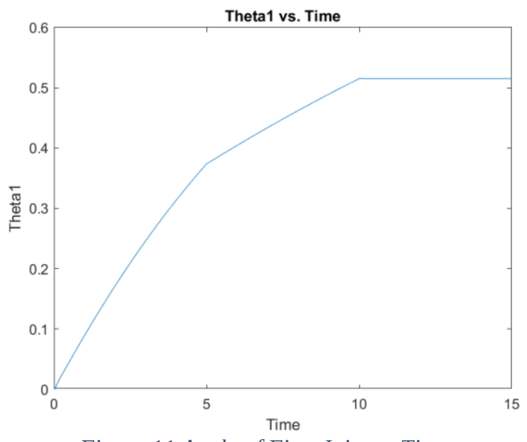

Figure 11 Angle of First Joint vs Time

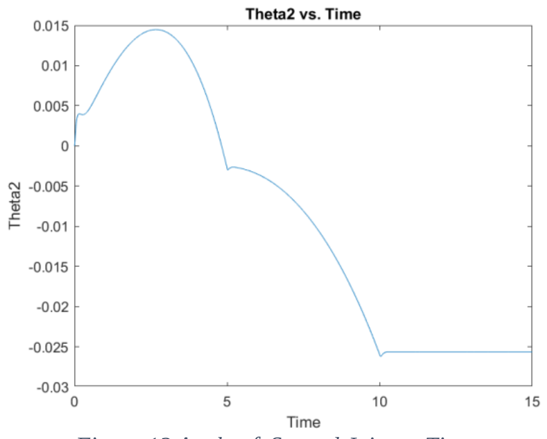

Figure 12 Angle of Second Joint vs Time

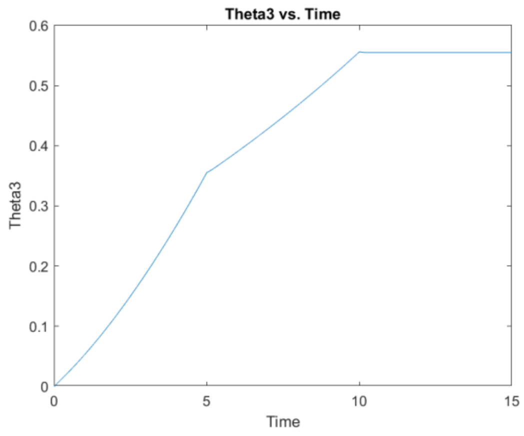

Figure 13 Angle of Third Joint vs Time

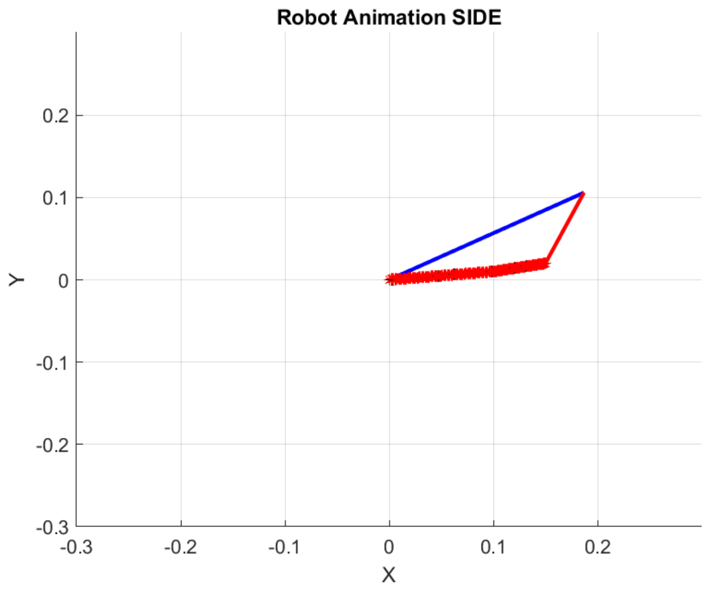

Figure 14 Robot Animation from Side View

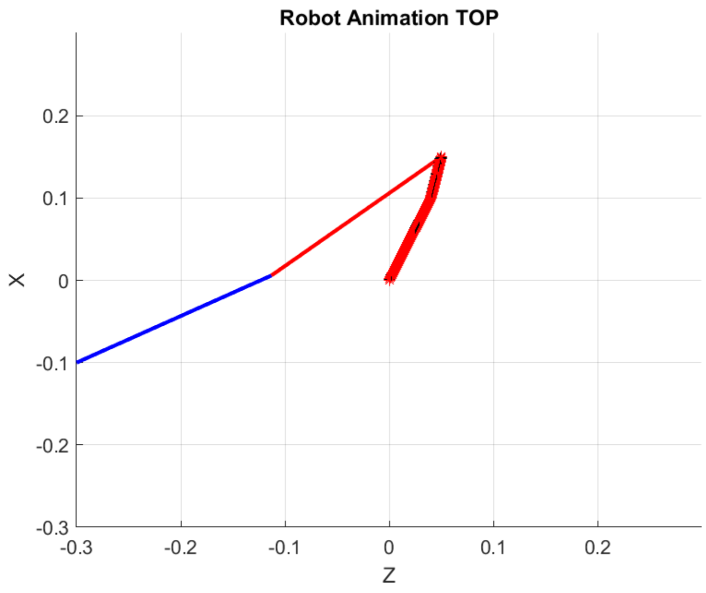

Figure 15 Robot Animation from Top View

III. Conclusion

Overall, the project focuses on establishing a robust control system that can accurately manipulate the PHANToM Premium 1.5 haptic robotic arm. By combining the path generation, forward kinematics, PID control, Jacobian calculation, and dynamic properties, the system enables precise movement along a straight path while compensating for errors and providing a responsive control mechanism.

To achieve robustness and fast response PID controllers are utilized. Their gains are arranged by using auto tuning options of Simulink. The simulated path almost perfectly matches the desired path, but there are significant oscillations of the force output. So, succession of the project can be evaluated as the control system design and tracking of the desired path are achieved, but the oscillatory behavior cannot be reduced. Even if different gain parameters are used, non-oscillatory behavior and appropriate tracking are not satisfied at the same time.

References

[1] J. J. Craig, “Section 6.8,” in Introduction to robotics: mechanics and control, Upper Saddle River, N.J: Pearson/Prentice Hall, 2005