Development and evaluation of PI, PD and PID controllers for a redundant robotic arm

Bišćević, Aaron

UNIVERSITY OF ZAGREB
FACULTY OF MECHANICAL ENGINEERING AND NAVAL ARCHITECTURE

BACHELOR'S THESIS

Aaron Bišćević

ZAGREB, 2024

UNIVERSITY OF ZAGREB
FACULTY OF MECHANICAL ENGINEERING AND NAVAL ARCHITECTURE

BACHELOR'S THESIS

DEVELOPMENT AND EVALUATION OF PI, PD AND PID CONTROLLERS FOR A REDUNDANT ROBOTIC ARM

Mentor:
Asst. Prof. Filip Šuligoj, Ph.D., Mech. Eng.

Student:
Aaron Bišćević

ZAGREB, 2024

I hereby declare that I have made this thesis independently using the knowledge acquired during my studies and the cited references.

I would like to express my sincere gratitude to my mentor, Filip Šuligoj, as well as Tara Knežević for assisting, and guiding me through the development of this thesis. Their advice and support have been crucial for the completion of this work.

I would like to extend my thanks to my friends for the countless memorable moments we've shared over the past three years. Their companionship and support have made this journey all the more enjoyable and unforgettable.

Finally, I would like to express my heartfelt thanks to my parents for their unwavering support throughout my education. Their encouragement has been invaluable in helping me reach this milestone.

Zagreb, September 2024 Aaron Bišćević

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 24 – 06 / 1**	
Ur.broj: **15 – 24 –**	

ZAVRŠNI ZADATAK

Student: **Aaron Bišćević** JMBAG: **0035239619**

Naslov rada na hrvatskom jeziku: **Razvoj i evaluacija PI, PD i PID kontrolera za upravljanje redundantnom robotskom rukom**

Naslov rada na engleskom jeziku: **Development and evaluation of PI, PD, and PID controllers for a redundant robotic arm**

Opis zadatka:

This thesis focuses on the development and evaluation of PI, PD, and PID controllers for managing a robotic arm using the Franka Control Interface (FCI). The FCI enables exceptionally fast and precise real-time control with a frequency of 1 kHz, making it ideal for implementation. The goal is to achieve controlled movement of the robotic arm, applying various control methods that contribute to increased safety and efficiency, achieved through programming in C++. Additionally, this work includes conducting experimental tests that will measure the precision of the response, the time required to reach a stable state, and potential overshoots of critical values. The results of these comparisons will provide insight into the control strategies of the robotic arm, which is crucial for improving performance in real operational conditions of the robotic arm.

The key tasks include:

- Development of PI, PD, and PID modes of control for the movement of the robotic arm using the robot's external coordinates (position of the tool center point).
- Implementation of PI, PD, and PID modes of control for the movement of the tool tip of the robotic arm according to a predefined motion profile on the Franka Emika Panda robotic arm (7 degrees of freedom of movement).
- Conducting a series of tests to measure and compare the responses of the system obtained using different control methods and compare them with theoretically designed responses.
- Comparatively assess the performance of PI, PD, and PID control modes in terms of the precision of the response, settling time, and overshoot.

The thesis must cite the literature used and any assistance received.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
24. 4. 2024.	**2. rok (izvanredni):** 11. 7. 2024. **3. rok:** 19. i 20. 9. 2024.	**2. rok (izvanredni):** 15. 7. 2024. **3. rok:** 23. 9. – 27. 9. 2024.

Zadatak zadao:

doc. dr. sc. Filip Šuligoj

Predsjednik Povjerenstva:

izv. prof. dr. sc. Petar Ćurković

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\Delta t_{2,sync,i}$	s	synchoranized time needed for all joints to complete the second part of the movement
$sign(\Delta c_i)$	m	the current pose, needed to determine in which direction to execute the movement
$c_{1,i}$	m	the path passed in the first part of the movement
$dc_{max,sync}$	m/s	maximum speed of the synchronized joint movement
t	s	the passed time
$\Delta c_{d,i}$	m	the movement required to get to the next position
$t_{1,sync,i}$	s	synchoranized time needed for all joints to complete the first part of the movement
i		discretization parameter
K_d		derivative parameter
K_i	s^{-2}	$integral parameter$
K_p	s^{-1}	$proportional parameter$
${}^b\xi_{TCP}$	m	the pose of the TCP with respect to the base of the robot
$\theta_e(t)$	rad	error responses
$\dot{\theta}_d$	rad/s	desired joint velocity in feedforward control
$\dot{\theta}$	rad/s	joint speed sent from the control law
$J^{\dagger}(\theta)$	s/m	pseudoinverse of the jacobian
Ω		angle between the starting and ending point on the unit sphere
p_0		starting point on the unit sphere
p_1		ending point on the unit sphere
q_1		starting quaternion of the robot pose
q_2		ending quaternion of the robot pose
t		interpolation parameter for the *SLERP* method

Abbreviations

DoF	degrees of freedom
FCI	Franka Control Interface
PID	proportional, integral, derivative (control)
RL	reinforcement learning
ROS	Robot Operating System
SLERP	Spherical Linear Interpolation
TCP	Tool Centre Point
UR	Universal Robots

SUMMARY

This thesis is concerned with the development and evaluation of motion control strategies in task space for a redundant robotic arm, with a particular focus on the use of P, PI, PD, and PID controllers. The motion control strategy was developed for the Franka Emika Panda robot, with the reference being given in task space coordinates. The results demonstrate the efficiency of the trajectory generation method, which employed a trapezoidal velocity profile to guarantee smooth and precise movements while synchronizing all joints for time-efficient operation. However, the trajectory generation method is inadequate when confronted with dynamic and unpredictable environments. Among the controllers, the P controller exhibited superior performance, demonstrating minimal overshoot, brief settling times, and high accuracy. The PI, PD, and PID controllers, although effective, exhibited limitations such as oscillations and longer settling times.

Keywords: motion control, PID controller, redundant robotic arm, trajectory generation, libfranka

PROŠIRENI SAŽETAK

Ovaj rad prati razvoj i evaluaciju strategija kontrole kretanja redundantne robotske ruke implementacijom proporcionalnog (P), proporcionalno-integralnog (PI), proporcionalno-derivativnog (PD) i proporcionalno-integralno-derivativnog (PID) kontrolera. Svaki od kontrolera bit će ispitani testovima te će biti analizirana dobivena točnost, statička greška i vrijeme smirivanja, kao i stabilnost robotske kretnje. Za implementaciju koda koristi se redundantna robotska ruka, Franka Emika Panda, koja zbog svog dodatnog, sedmog stupnja slobode gibanja ima povećanu fleksibilnost, odnosno istu pozu prihvatnice moguće je postići u više konfiguracija. To donosi izazove u rješavanju problema inverzne kinematike.

Rad počinje primjerima iz prakse, te je u uvodu također objašnjen cilj ovog rada. Zatim je u drugom poglavlju navedena tehnička pozadina, objašnjen je način komunikacije robota s računalom i na koji način korisnik može pomoću `libfranka` biblioteke upravljati robotom. Osim toga dan je pregled drugih metoda kontrole kretanja robota. U trećem poglavlju objašnjene su metode korištene za izradu ovog rada. Nadalje su u četvrtom poglavlju objašnjeni parametri pomoću kojih će se ocjenjivati uspješnost testova. Rad završava pregledom provedenih testova i zaključkom.

Korištenjem `libfranka` biblioteke, dobiva se mogućnost slanja naredbi, kao i pristup podacima očitanih sa senzora na robotu u stvarnom vremenu, brzinom od 1 kHz. Tako se može postići vrlo precizna i točna kontrola robota. Osim toga, za razliku od drugih biblioteka korištenih za kontrolu robotskih manipulatora, `libfranka` omogućava korisniku da upravlja najnižim nivoom robotskog *hardwarea*, čime se mogu bolje iskoristiti njegove mogućnosti.

Za kretanje robota, prvo će se razviti *point-to-point* generator trajektorije s trapezoidnim profilom brzine, koji kretanje dijeli na faze ubrzavanja, stalne brzine i usporavanja. Takav profil omogućava mirnije pokretanje i zaustavljanje, bez trzajnih pokreta (engl. *jerk motion*), te optimalno korištenje vremena za prelaz velikih udaljenosti korištenjem jednolike brzine gibanja, kao što je prikazano na slici 3.1. Ovakva metoda kretanja robota izvrsna je za primjene u industrijskim uvjetima, gdje su poslovi repetitivni, okoliš je predvidljiv i bez prepreka [1].

Za računanje rotacijskog kretanja korištena je *spherical linear interpolation* (SLERP) metoda. Pomoću nje moguće je izračunati potrebnu rotaciju između dvije orijentacije prihvatnice, izražene u kvaternionima, kako je prikazano formulom 0.1. U ovoj formuli bitan je i parametar interpolacije t, pomoću kojeg je moguće odrediti traženu rotaciju između početnog kvaterniona q_1 i konačnog q_2 [2].

$$SLERP(q_1, q_2, t) = q_1(q_1^{-1}q_2)^t \tag{0.1}$$

Za kontrolu kretanja koristit će se upravljački zakon 0.2, u kojem je uključen i *feed-forward* kao i *feed-back* upravljanje. Iako je referenca zadana u vanjskim koordinatama, na robota će se naredbe slati koristeći unutarnje koordinate. Tako je napravljeno jer, iako `libfranka` biblioteka

dopušta korisniku da referencu šalje i u vanjskim koordinatama, većina drugih biblioteka za upravljanje robotskim manipulatorima nema tu mogućnost. Tako rad postaje fleksibilan, te se upravljački zakon bez promjena može primijeniti na drugim robotima koji koriste različite upravljačke biblioteke.

Za definiranje greške, kretnja je podijeljena na translaciju i na rotaciju, te će svaka biti posebno tretirana. Za translaciju će put koji robot mora prijeći biti diskretiziran pomoću parametra diskretizacije *i*. To se radi kako bi se robotu mogli slati točno određeni segmenti puta u svakom uzorku te se tako kontrolira brzina kretanja robota. Dohvaćanjem trenutne poze robota, moguće je računati grešku koja će se slati u kontroli zakon.

Rotacijsko gibanje ponovno je rađeno pomoću $SLERP()$ metode, a interpolacijski parametar t matematički je vezan za parametar i, kako bi se osiguralo da se obje kretnje odvijaju jednakom brzinom.

Pseudokod koji će biti napisan u `C++` moguće je vidjeti u algoritmu 1.

$$\dot{\theta} = \dot{\theta}_d(t) + K_p\theta_e(t) + K_i\int_0^t \theta_e(t)\,dt + K_d\dot{\theta}_e(t) \tag{0.2}$$

Testovi su provedeni tako da se robot kretao od jedne poze do druge, te su vanjske koordinate bile zabilježene na grafovima od 5.1 do 5.9.

Generator trajektorije kretnju robota provodi vrlo fluidno, s velikom točnošću. Kretanje je napravljeno brzo i efikasno, sinkronizirajući zakret svakog zgloba robota. Iako je ovakvo kretanje vrlo efikasno, bez naglih pokreta, ograničena mu je uporaba, jer robot nije u stanju prilagoditi se nepredvidivom okolišu s preprekama. Iz tog razloga pojavljuje potreba za kontrolom kretanja.

Analizom grafova dobivenih u testovima, moguće je primijetiti da najbolje performanse ima P kontroler, koji ne pokazuje prebačaj, te ima vrlo kratko vrijeme smirivanja i veliku točnost. Osim toga, P kontroler nema oscilatornog ponašanja, kakvo je moguće vidjeti u testovima s PI kontrolerima. Također se robot ne počinje kretati u suprotnom smjeru od reference što je vidljivo kod PD kontrolera, uz brže vrijeme smirivanja.

PID kontroler je također pokazao prebačaj, uz poveće vrijeme smirivanja, bez obzira na namještanje parametara.

Ovim radom pokazan je način na koji se može implementirati kontrola kretnje na redundantnoj robotskoj ruci pomoću `C++` koda. Evaluirani su P, PI, PD i PID kontroleri pomoću testova kretanja robota. Iscrtavanjem grafova kretnje robota ustvrđeno je da od navedenih kontrolera, najbolju performansu ima P kontroler.

1. INTRODUCTION

1.1. Overview of robotic systems and applications

The advent of robotic arms has had a profound impact on numerous sectors, including manufacturing and healthcare. These automated systems offer unparalleled precision, reliability, and efficiency, making them invaluable assets in a multitude of applications. Among the numerous varieties of robotic arms, those which can be described as redundant are distinguished by their exceptional flexibility and dexterity. In contrast to standard robotic arms, which possess a limited number of degrees of freedom (DoF), that are sufficient for performing specific tasks, redundant robotic arms exhibit an augmented number of DoF, exceeding six, enabling them to attain the same objective through a multitude of configurations. This redundancy presents a range of potential applications, rendering them highly valuable in environments that require flexibility, adaptability, and fault tolerance [3].

In essence, redundancy in robotic arms can be defined as the presence of additional joint variables that exceed the minimum requirements necessary for performing a given task. To illustrate, while a conventional task may necessitate the control of a robot's position and orientation in three-dimensional space via a mere six degrees of freedom, a redundant robotic arm may possess seven or more degrees of freedom. This redundancy enables the arm to circumvent obstacles, optimize its posture for energy efficiency, and accommodate mechanical limitations without compromising task performance. Such capabilities have significant implications for a number of real-world applications [4].

In industrial settings, redundant robotic arms are commonly utilized for tasks such as welding, assembly, and material handling. The additional degrees of freedom afforded by redundant robotic arms provide enhanced maneuverability, enabling the arms to operate in constrained or cluttered spaces where traditional robots might struggle. Furthermore, redundancy enhances fault tolerance, allowing the robot to continue functioning even if one of its joints fails. This is particularly valuable in manufacturing processes where downtime due to equipment failure can lead to significant productivity losses. Extensive research has been conducted on the use of redundant robotic arms in industry, with findings indicating that they can improve both task accuracy and overall performance [5], [6].

In addition to their use in industry, redundant robotic arms have been instrumental in advancing healthcare, particularly in minimally invasive surgery. In these intricate surgical procedures, surgeons must meticulously maneuver around vital structures, frequently within confined anatomical spaces. The additional flexibility afforded by redundant arms permits more precise control of surgical instruments, thereby reducing the risk of damage to surrounding tissues. The use of redundant robotic arms could enhance surgical outcomes by providing enhanced control during complex procedures, thereby making them an increasingly essential tool in the medical

field [7].

Furthermore, redundant robotic arms are proving to be invaluable in the field of space exploration. The unique challenges presented by space environments include weightlessness, extreme temperatures, and limited accessibility for human intervention. The superior adaptability of redundant arms has been employed in a variety of missions to facilitate maintenance, assembly, and repair operations on spacecraft and satellites. To illustrate, the Canadian Space Agency's "Canadarm2," illustrated in Figure 1.1 a redundant robotic arm utilized on the International Space Station, has been pivotal in facilitating a multitude of operations, including spacecraft docking and station component repair. The additional degrees of freedom allow for precise manipulation in the absence of gravity, thereby underscoring the critical role redundancy plays in space robotics [8].

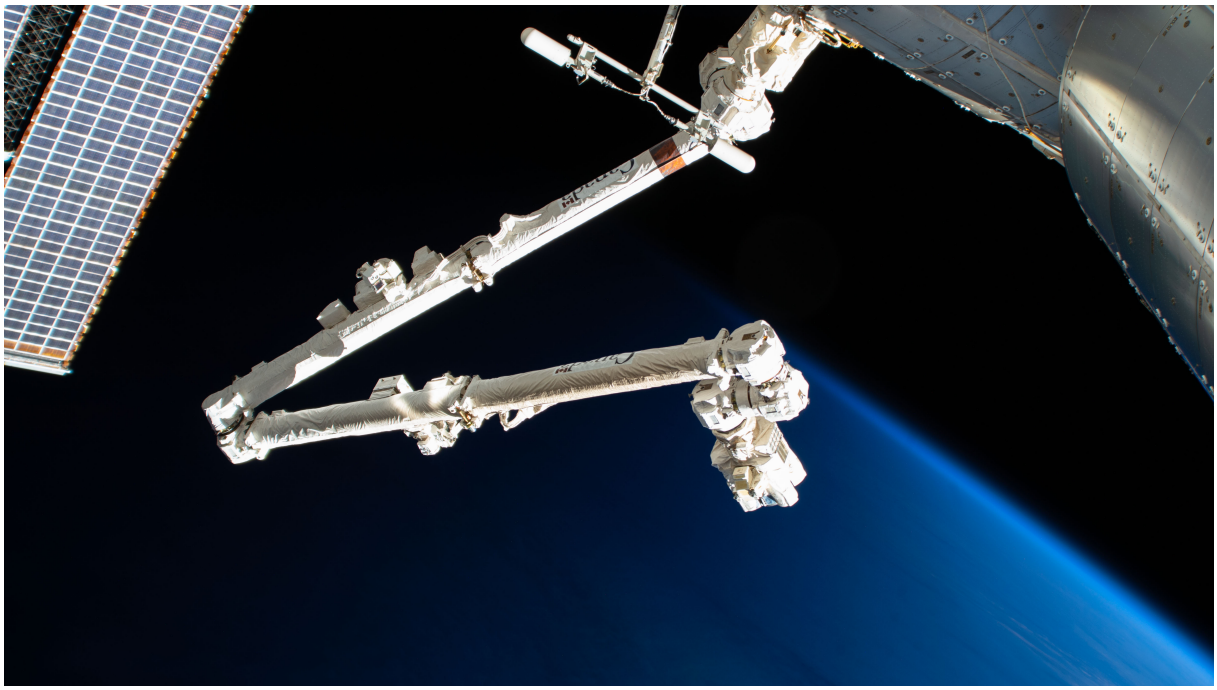

Figure 1.1: Canadarm2 at the ISS in the orbit above the Pacific Ocean [8]

Ongoing research is investigating the potential for redundant robotic arms to be optimized for autonomous systems in ways that extend beyond their immediate applications. In the context of autonomous driving and personal robotics, these robotic arms could be utilized to perform complex tasks that require a high level of adaptability and environmental interaction. As the scope of tasks delegated to autonomous robots expands, the capacity to make dynamic adjustments in real time will become a crucial necessity. The incorporation of redundancy into autonomous systems has been shown to enhance decision-making processes by providing a multitude of potential pathways for task completion [9], [10].

While redundant robotic arms offer numerous advantages, they also present certain challenges. The principal challenge is the complexity of controlling them. The increased number of

degrees of freedom necessitates the management of a more expansive set of variables, thereby imposing greater computational demands on the control algorithms. Researchers have been actively engaged in the development of optimized control strategies for redundant robots, with the objective of achieving a balance between computational efficiency and the robot's capacity to adapt to changing environments [11].

Recent advances in artificial intelligence and machine learning provide promising avenues for developing more efficient control mechanisms that can fully leverage the advantages of redundancy [9].

1.2. Objective and scope of the thesis

The objective of this thesis is to develop and evaluate motion control strategies that are based on the robot's position in task space as the primary reference point. In particular, Proportional-Integral (PI), Proportional-Derivative (PD), and Proportional-Integral-Derivative (PID) controllers will be implemented and evaluated with the objective of achieving precise motion control in a robotic system. The efficacy of these controllers will be evaluated based on key performance metrics, including settling time, overshoot, and precision. The experiments will be conducted using the Franka Emika Panda robotic arm, with the control algorithms implemented via the `libfranka` library within a `C++` framework.

The principal objective of this study is to optimize the proportional (P), integral (I), and derivative (D) control parameters to achieve optimal system performance. A series of systematic tests will be conducted to evaluate the robot's performance in executing repeated movements. The resulting data will be analyzed to determine the optimal parameter configurations. The outcomes of this research are expected to contribute to the refinement of motion control methodologies, offering insights that may prove valuable for applications in precision robotics, including fields such as automated manufacturing and robotic-assisted surgery.

1.3. Thesis organization

This thesis opens with an overview of the Franka Emika Panda robot, with a particular emphasis on its control through the use of the `libfranka` library. The fundamental principles of communication between the robot and the computer will be explained, and will be followed by a review of relevant robotic motion control techniques. This review will include a discussion of advanced motion control strategies, some of which are beyond the scope of this work but offer useful context.

The following section will examine the methods utilized in this thesis, along with the theoretical background of the two primary techniques employed: point-to-point trajectory generation and task-space motion control. These methods provide the basis for the experimental work

conducted in this research.

The subsequent section will provide a detailed account of the experimental procedures, including a description of the monitored parameters that are essential for interpreting the outcomes. A comprehensive account of the experiments conducted will be provided to establish a clear correlation between the methods employed and the resulting outcomes.

The results of the experiments are then presented, accompanied by a discussion that evaluates the findings in light of existing approaches. A critical analysis will highlight both the strengths and limitations of the techniques used in this work, offering a comparison with other research in the field.

2. BACKGROUND AND RELATED WORK

2.1. Introduction to the Franka Emika robot

The Franka Emika Panda robot, depicted in Figure 2.1, is a collaborative robot that is frequently utilized in research settings due to its straightforward force and torque control capabilities, as well as its intuitive human-robot interaction. The robotic arm is a 7DoF manipulator, which is kinematically redundant. This allows the robot to move along more complex trajectories or rearrange the joint positions while maintaining the end-effector pose [12].

2.2. Overview of the `libfranka` library

The Franka Emika Panda robot can be interfaced via the `libfranka` library. The `libfranka` library represents the `C++` implementation of the client side of the Franka Control Interface (FCI). It enables communication between the network and the control system, and provides interfaces that facilitate straightforward interaction with the system, thereby enabling the following actions to be achieved:

- Issue of commands that are not time-critical in order to control the hand and configure the arm parameters.
- Execution of real-time commands to run user-defined 1 kHz control loops.
- Robot state may be read in order to obtain sensor data at 1 kHz.
- Model library can be accessed in order to compute the desired kinematic and dynamic parameters.

The `libfranka` library enables the user to control the robot and send real-time commands at a high frequency, thereby enabling precise control of the robot. Moreover, the user is able to modify the thresholds for collision detection, Cartesian, and joint impedance. The library contains a robot model store, which facilitates the utilization of the Jacobian and inverse kinematics in a more straightforward manner than with other robots [14].

The Figure 2.2 shows the architecture of the robot-computer communication, which is facilitated by the `libfranka` library. Moreover, the robot can be directly controlled using the `libfranka` library with both non-real-time and real-time commands [15].

The `libfranka` library is distinguished from other robot control libraries by its emphasis on real-time control and its design tailored to the Franka Emika Panda robot. This library provides a direct interface to the Panda's hardware, enabling developers to implement advanced control strategies such as impedance control and force feedback and to fully utilize the robot's

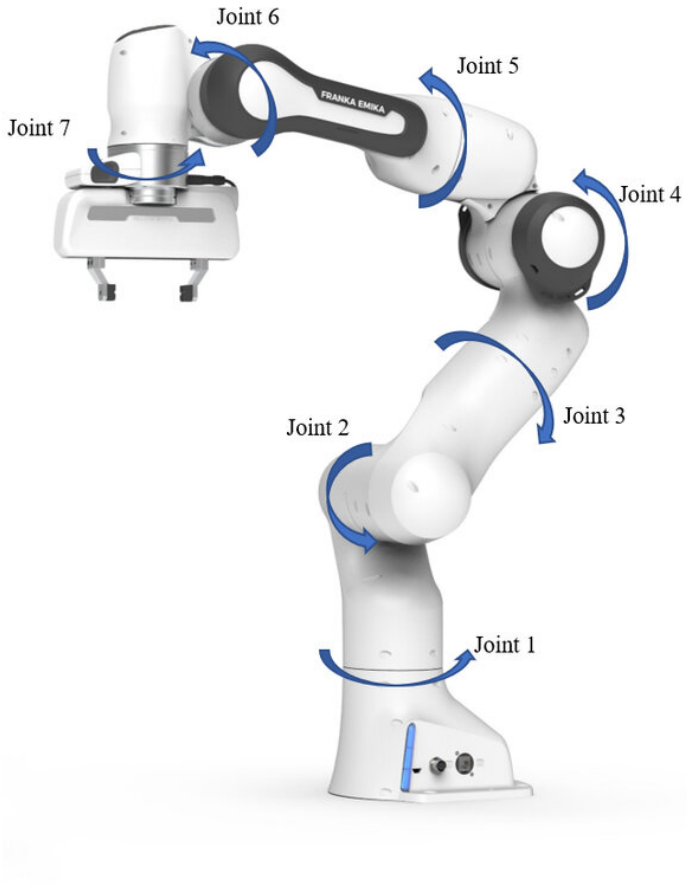

Figure 2.1: Franka Emika Panda robot with labeled joints [13]

capabilities. In contrast, other robot control libraries, such as Robot Operating System (ROS) and MoveIt, both used by Universal Robots (UR), offer broader compatibility across different robotic platforms but often lack the real-time performance and low-level control features that `libfranka` provides, rendering them less suitable for high-precision tasks requiring immediate feedback and adjustments [16].

The `libfranka` library enables the user to construct motion generators and motion controllers via the use of real-time commands. In this study, both methods will be employed and evaluated in order to demonstrate the degree of accuracy and precision that can be attained.

Firstly, a point-to-point motion generator utilising Cartesian coordinates in task space will be implemented in order to facilitate the rapid and precise navigation of the robot from one point to another. The error margin for the end-point is in the order of magnitude of 10^{-6}. As

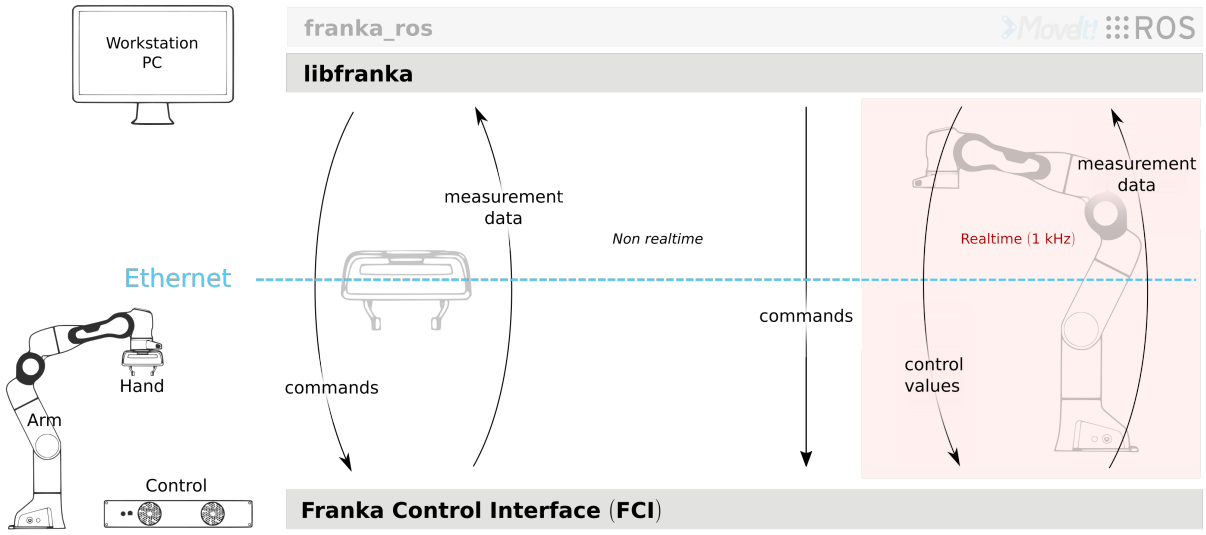

Figure 2.2: the Panda robot and FCI architecture [14]

illustrated on Figure 2.3, the desired pose, either in joint or task space is transmitted to the robot in real time, enabling the robot to reach the end point along the desired path by computing the required torques in real time.

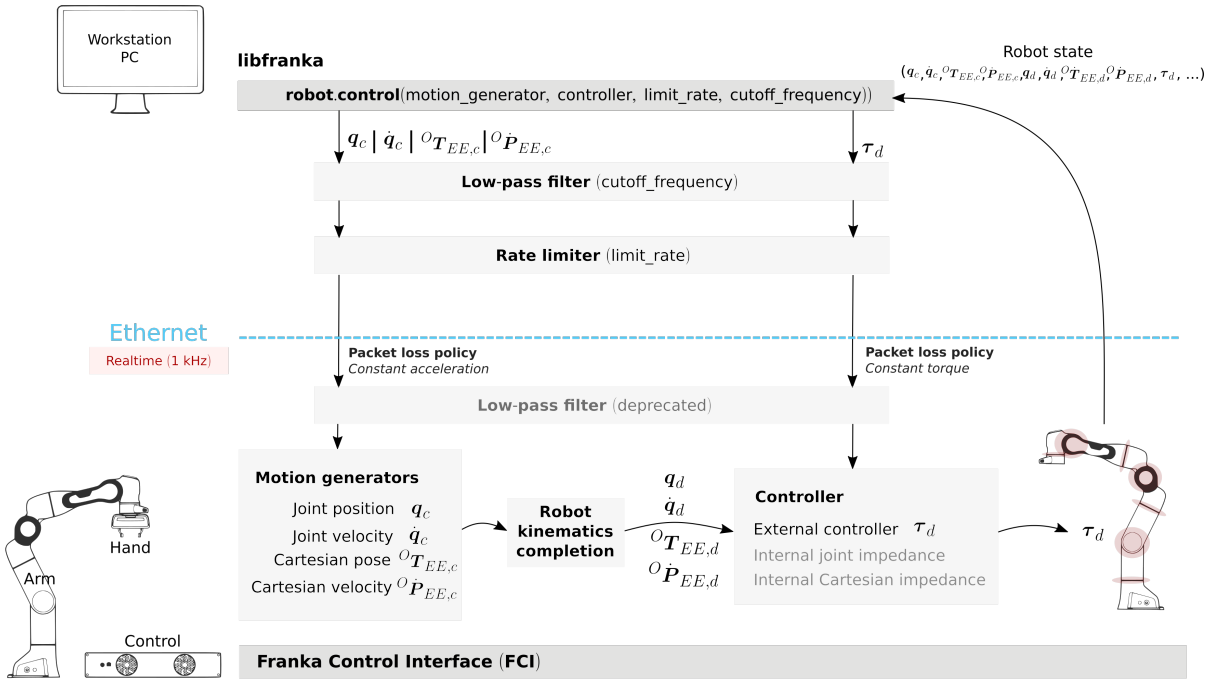

Figure 2.3: Real time loop, using commands to get the desired motor torques [14]

Subsequently, a motion controller will be developed. A control law can be devised in real time using the pose of the end-effector in task space, which will manage the movement of the end-effector in space and actively suppress the error. Moreover, the parameters of the propor-

tional, integral, and derivative controllers will be evaluated and determined. Ultimately, the optimal controller will be selected and the rationale behind its selection will be discussed.

2.3. Review of robotic motion control techniques

Robotic motion control in the operation of robotic manipulators is most needed in tasks where accuracy and precision are paramount in dynamic and unpredictable environments. Over time, a multitude of techniques have been devised to regulate the movements of robotic manipulators, each with its own set of advantages and limitations, reflecting the diverse applications and environments in which these robots are deployed.

The most popular control method in industrial robotics is classical control, a decentralized "proportional, integral, derivative" (PID) control for each degree of freedom [1]. The control method in question regulates the discrepancy between the desired pose of the end-effector and the actual output. It employs three components: proportional control to address current errors, integral control to eliminate steady-state errors, and derivative control to anticipate future errors. PID control is effective for systems with linear dynamics and is relatively straightforward to implement. However, its performance can be compromised in the presence of nonlinearities or disturbances [17].

Additional robot control techniques include model-based control methods, which have been developed to address the limitations of PID control in handling non-linearities and dynamic interactions between joints. Notable among these are Computed Torque Control (CTC) and Inverse Dynamics Control. These methods rely on the dynamic model of the robot, taking into account factors such as inertia, Coriolis forces, and gravity in order to compute the requisite joint torques for the execution of a given trajectory. Although these techniques offer enhanced precision and velocity, they necessitate the precise representation of the robot's dynamics, which can be intricate and computationally demanding. Notwithstanding these obstacles, model-based control has become a pervasive methodology in robotics, particularly in operations that necessitate elevated precision and responsiveness, such as robotic surgery and sophisticated manufacturing processes [18].

In addition to model-based approaches, adaptive and learning-based control techniques have emerged as a prominent area of interest in recent years. Adaptive control methods are designed to dynamically adjust the control parameters in response to changes in the robot's environment or payload, rendering them well-suited for applications where conditions are inherently unpredictable or variable. Learning-based methodologies, including reinforcement learning and neural network-based control, facilitate the enhancement of robot performance over time through the acquisition of experience. These techniques are particularly advantageous in complex, non-linear environments where traditional control methods may encounter difficulties. Nevertheless, the deployment of learning-based methods necessitates meticulous attention to

matters of safety and reliability, particularly when integrating robots into real-world scenarios where unanticipated behaviors could potentially result in mishaps or system failures [19].

3. METHODS

A robotic manipulator can be moved using multiple methods, two of which will be discussed in this work: trajectory generation and control design. These two approaches form the basis upon which a manipulator can produce efficient and precise movements, whether predefined or in reaction to a dynamic environment.

3.1. Trajectory generation for robotic movement

Trajectory generation is a common technique employed in the implementation of manipulators in repetitive, predictable tasks, where the environment is static and no obstacles may impede the robot's path. The trajectory generated for the end-effector to follow can be categorized into one of four categories, depending on the task at hand [1]:

1. an unconstrained path between two end-points
2. a path between two end-points, which passes through *via points*
3. a constrained path between two end-points
4. a constrained path between two end-points, which passes through *via points*.

The trajectory is generated in either the joint space (categories one and two) or the task space (categories three and four), depending on the category in question.

In the majority of cases, a joint space trajectory generator will be employed for tasks that are conducted in an environment with a considerable amount of free space, where the robot is not impeded by any obstacles. Moreover, trajectory generation in joint space is advantageous due to its reduced computational load, as no inverse geometric or kinematic models need to be calculated. Additionally, it avoids the potential issue of crossing singular configurations [1].

In other instances, where task space trajectory generation is necessary, such as when the geometry of the path taken by the Tool Centre Point (TCP) needs to be controlled due to constrained spaces, certain disadvantages become apparent. These include the potential for failure in the event of crossing a singularity configuration, the possibility of the endpoints being situated outside of the joint reach, and a reduction in configuration possibilities with regard to torques and velocity limits [1].

3.1.1. Point-to-Point trajectory planning

The most straightforward method for trajectory generation is to direct the manipulator from one point to another along a linear trajectory. In contrast to continuous path planning, where the trajectory between the initial and final points must adhere to a specific shape or pattern,

point-to-point planning prioritizes the precision of the final position over the exact nature of the path traversed.

In designing a trajectory generator, it is essential to ensure that the trajectory is a sufficiently smooth function of time and that it respects any given limits on joint velocities, accelerations, or torques [20].

It is therefore recommended that a time scaling method be implemented. The profile employed in the present work is a trapezoidal motion profile, whereby the velocity of the joints is scaled to produce a trapezoidal trajectory. This is illustrated in Figure 3.1. The method is based on dividing the movement into three distinct phases: acceleration, constant velocity and deceleration.

Such a velocity profile is especially beneficial by Point-to-Point applications, as it ensures that no sudden jerks or high torque values will be exerted to the joint motors. When dealing with manipulators with high inertia values, this can be important as using such a profile will minimize overshoot of the end-point. Moreover, the constant speed phase of the trapezoidal velocity profile trajectory generator will ensure that the robot can move efficiently, and so reduce the cycle time [21].

When creating the trapezoidal profile, transitional as well as rotational movement in task space will take place simultaneously, along all three axes, therefore it must be taken into account which movement will take the most time. This will, in turn, determine the duration of the whole manipulator motion, and how the velocities in other directions is scaled, so as to synchronize the whole movement [22].

Equations used to generate a trajectory between two end points are 3.1, 3.2 and 3.3.

$$\Delta c_{d,i} = \frac{-1}{t_{1,sync,i}^3 \cdot dc_{max,sync} \cdot sign(\Delta c_i) \cdot (0.5t - t_{1,sync,i}) \cdot t^3} \tag{3.1}$$

The Equation 3.1 is used for the first part of the movement, or the acceleration phase.

$$\Delta c_{d,i} = c_{1,i} + (t - t_{1,sync,i}) \cdot dc_{max,sync} \cdot sign(\Delta c_i) \tag{3.2}$$

The Equation 3.2 is used in the second part of the movement, or the constant velocity phase.

$$\begin{aligned} \Delta c_{d,i} = \Delta c_i + 0.5 \cdot \frac{1}{\Delta t_{2,sync,i}^3} ((t - t_{1,sync,i} - 2\Delta t_{2,sync,i} - t_{d,i}) \cdot (t - t_{1,sync,i} - t_{d,i})^3 \\ + (2t - 2t_{1,sync,i} - \Delta t_{2,sync,i} - 2t_{d,i})) \cdot dc_{max,sync} \cdot sign(\Delta c_i) \end{aligned} \tag{3.3}$$

The Equation 3.3 is used in the third part of the movement, or the deceleration phase.

The rotational movement is calculated using the spherical linear interpolation (SLERP) method. SLERP is a method used to smoothly interpolate between two points on the surface

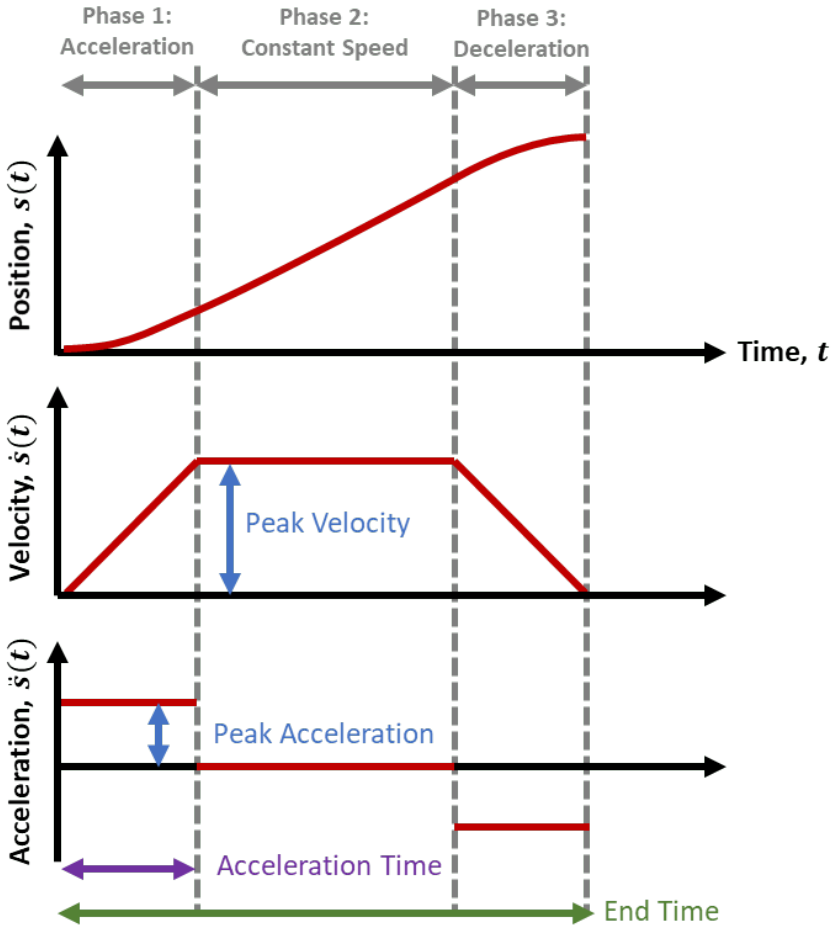

Figure 3.1: Visulatization of a trapezoidal motion profile [23]

of a unit sphere. It is particularly useful for computing rotations in computer graphics and animation. SLERP ensures that the interpolated points move at a constant velocity along the great circle arc connecting the starting and ending points. The calculation is performed as follows:

$$SLERP(p_0, p_1, t) = \frac{\sin((1-t)\Omega)}{\sin\Omega} p_0 + \frac{\sin(t\Omega)}{\sin\Omega} p_1 \tag{3.4}$$

where p_0 and p_1 are the starting and ending points on the unit sphere, t is the interpolation parameter, which ranges from 0 to 1, and where the angle between p_0 and p_1 is represented by the variable Ω.

In the context of quaternions as a means of representing rotations, the SLERP formula can be simplified to the following:

$$SLERP(q_1, q_2, t) = q_1(q_1^{-1} q_2)^t \tag{3.5}$$

where q_1 and q_2 represent the initial and final quaternions, respectively, and t denotes the interpolation parameter.

SLERP is of significant benefit to robotic motion, as it facilitates seamless and uninterrupted transitions between orientations in three-dimensional space. This is of particular importance in the field of robotics, where precise control of joint movements is essential for tasks such as manipulation and navigation. By employing the SLERP method, robotic systems are able to interpolate between two rotational states while maintaining a constant angular velocity, which results in more natural and efficient movements. This method helps to circumvent abrupt changes in motion that can lead to mechanical stress or instability, thus enhancing the overall performance and reliability of robotic applications [2].

The `Eigen/Geometry C++` library provides the slerp() method, which can be employed to calculate the necessary rotation between two poses. The method requires three variables: the initial quaternion, the final quaternion, and the interpolation parameter, which must lie between 0 and 1. As the movement progresses, the parameter t will increase from its initial value of 0 to its final value of 1.

3.2. Task space motion control

While trajectory generation, or more precisely Point-to-Point trajectory generation is a valid method for determining robotic movement, this approach fails when confronted with a dynamic environment, where the conditions are not repetitive, and are in fact constrained. In such instances, motion control becomes a requisite element. Depending on whether the required control level is low or high, the inputs may be velocity, torque, or force.

Motion control, and in particular task space motion control, can be beneficial in a number of contexts, including surgical robots, complex manufacturing processes and autonomous robots. These applications are characterised by the need for precision in an unstructured, unpredictable and dynamic environment[24].

Task space control is a better method than joint space motion control in that it utilizes readily calculable coordinates, thereby guaranteeing that the robot reaches the desired point or follows the intended trajectory with precision.

Nevertheless, the utilization of task control also presents a challenge, as it necessitates the calculation of the inverse geometric or kinematic model. This entails a considerable amount of computational power, particularly when computing for a redundant robot such as the Franka Emika Panda, which possesses seven degrees of freedom. This is because, in such cases, solving an inverse kinematics problem for the manipulator will result in an infinite number of solutions.

Nevertheless, the `libfranka` library provides a pre-built model of the robot, facilitating the calculation of the Jacobian, which is necessary for converting from task space to joint space.

3.2.1. Implementation of motion control algorithms

The Equation 3.6 is the control law used for running experiments.

$$\dot{\theta} = \dot{\theta}_d(t) + K_p\theta_e(t) + K_i\int_0^t \theta_e(t)\,dt + K_d\dot{\theta}_e(t) \tag{3.6}$$

It will be implemented using the `libfranka` robot.control() function which as an input takes a callback function, returning the desired joint velocities. The Tables 3.1 and 3.2 show how, in theory, changing each parameter of K_p, K_i and K_d will change the output of the system. It is to be expected that the robot should arrive to the desired point, without over- or undershooting. Furthermore, the response should be swift with minimal settling time.

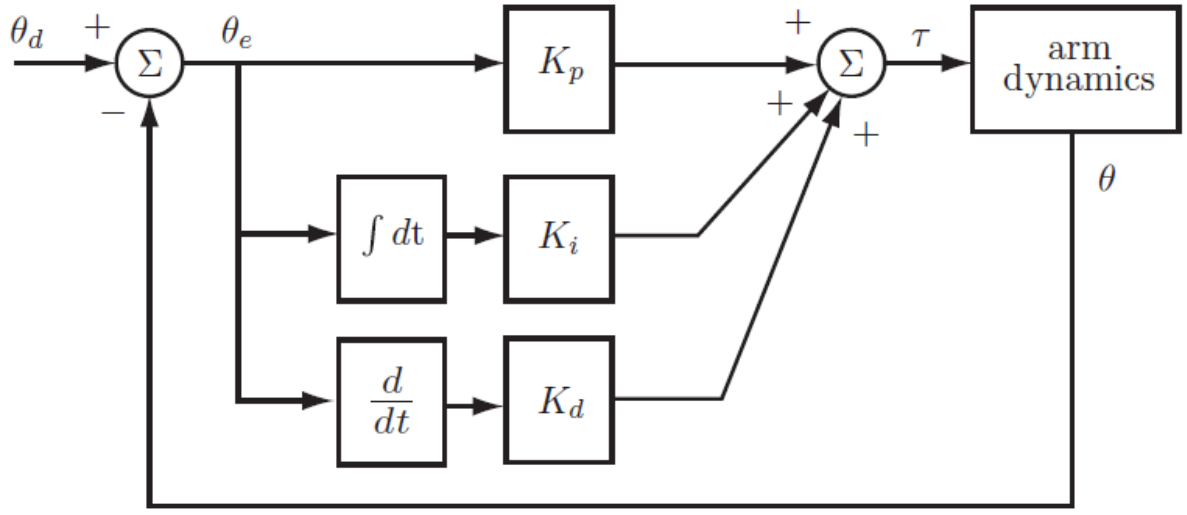

Figure 3.2: Block-model of the system [20]

The initial step will be the implementation of a basic feedforward control. The user is required to specify the desired pose in the task space. Subsequently, the pose will be multiplied by the pseudoinverse of the current Jacobian, as provided by the `libfranka` library, as illustrated in the Equation 3.7.

$$\dot{\theta}_d = J^{\dagger}(\theta)^b\xi_{TCP} \tag{3.7}$$

Subsequently, feedback control can be implemented. This will be achieved by initially defining the desired Cartesian coordinates of the endpoint in task space. The coordinates will be divided into two distinct categories: those pertaining to translational movement towards the desired pose and those associated with rotational movement towards the desired pose. These will be treated as distinct cases, as the calculation used to translate them to joint space coordinates is different. A discretization parameter i will be selected, which will determine the

number of parts into which the path/rotation will be divided and, consequently, the speed of the end-effector.

Translational movement will be calculated by determining the path which the end-effector needs to pass in order to get to the end point. This is achieved by subtracting the begging robot pose from the end pose. Because the path is discretized, as the loop is repeated, a new part of the path will be added to the desired pose.

Rotational movement will be calculated by using the SLERP method mentioned in the previous section. The initial and final rotations will be transformed into quaternions, and the interpolation parameter will be mathematically linked to the discretization parameter. This ensures that both the translational and rotational movements are discretized in an identical number of segments. The rotation resulting from the SLERP method will be expressed as a quaternion, which will then be transformed into a rotation matrix.

Finnaly, the translational and the rotational matrix will be added to a transformation matrix, and using the function from [25], which returns a solution of the inverse kinematics problem, the desired joint configurations are determined.

The rationale behind generating the reference in the task space and subsequently transmitting it to the robot in the form of robot joint configurations is that, although the Franka Emika Panda robot and the `libfranka` library do facilitate the utilization of Cartesian coordinates for commanding the robot, this is not a capability shared by all robots. For instance, the UR robot does not possess this capability. It is therefore possible to extend the application of this control method to other robots.

Proportional gain will be calculated by comparing the current configuration in the discrete time section to the desired configuration. Subsequently, the value will be multiplied by the proportional parameter, K_p.

Integral gain will be calculated by adding the value of the pose error multiplied by the discretization time to the sum of errors each time the loop is executed. Additionally, the sum of errors will be multiplied by the integral parameter, K_i.

Derivative gain will be calculated by subtracting the previous pose error (from the previous loop execution) from the current pose error and dividing the result by the discretization time. The resulting value will then be multiplied by the derivative parameter, K_d.

The objective of this study is to identify the optimal parameters that will ensure the aforementioned requirements are met.

3.3. Software and hardware setup

The robot used will be the Franka Emika Panda robot, which is a 7 DoF robot, meaning it is redundant. Although this complicates the calculations needed to get an inverse kinematics model, thanks to the `libfranka` library, which provides an estimation of the Jacobian matrix at

Table 3.1: Effect of changing a parameter on Time Domain Parameters

Parameter	Rise Time	Overshoot	Settling Time
K_p	Decrease	Increase	Small change
K_i	Decrease	Increase	Increase
K_d	Minor change	Decrease	Decrease

Table 3.2: Effect of changing a parameter on Steady-State Error and Stability

Parameter	Steady-State Error	Stability
K_p	Decrease	Degrade
K_i	Eliminate	Degrade
K_d	No effect in theory	Improve if K_d is small

every moment in real time, calculating the transpose of the Jacobian of the current position of the robot is made much easier.

The robot is connected to the computer running on Linux, with a connection frequency of 1000 Hz. This means that calculating the speed of the robot is simplified. Using the constant velocity formula $v = s/t$, and knowing that $t = 10^{-3}$ s means that depending on what value of s is sent to the robot will make the manipulator faster and slower, and slows the control of jerk motion, acceleration and sudden movements.

The pseudo-code which will be applied is shown in Algorithm 1

Algorithm 1 Task Space Motion Control with PID

$\mathbf{T_d} = [x_d, y_d, z_d]$ ▷ Initialize desired translation
$\mathbf{Q_d} = [q_{xd}, q_{yd}, q_{zd}, q_{wd}]$ ▷ Initialize desired rotation quaternion
$\mathbf{T_c} = [x_c, y_c, z_c]$ ▷ Initialize current translation
$\mathbf{Q_c} = [q_{xc}, q_{yc}, q_{zc}, q_{wc}]$ ▷ Initialize current rotation quaternion
i ▷ Set discretization parameter
K_p, K_i, K_d ▷ Set PID gains
$\mathbf{E_{prev}} = 0$ ▷ Initialize previous pose error
$\Delta\mathbf{T} = \frac{\mathbf{T_d} - \mathbf{T_c}}{i}$ ▷ Compute translation step
for $t = 1$ to i **do**
    $\mathbf{T_c} = \mathbf{T_c} + \Delta\mathbf{T}$ ▷ Update current translation

    $\mathbf{Q_c} = \text{SLERP}(\mathbf{Q_c}, \mathbf{Q_d}, \frac{t}{i})$ ▷ Compute rotation quaternion step via SLERP:

    $\mathbf{T_{mat}} = \text{ComposeMatrix}(\mathbf{T_c}, \mathbf{Q_c})$ ▷ Compute transformation matrix

    $\mathbf{J_d} = \text{InverseKinematics}(\mathbf{T_{mat}})$ ▷ Compute desired joint configuration

    $\mathbf{E} = \mathbf{T_d} - \mathbf{T_c}$ ▷ Compute current pose error

    ▷ Compute PID gains:
    $P = K_p \cdot \mathbf{E}$ ▷ Proportional gain
    $I = I + K_i \cdot \mathbf{E} \cdot \Delta t$ ▷ Integral gain
    $D = K_d \cdot \frac{\mathbf{E} - \mathbf{E_{prev}}}{\Delta t}$ ▷ Derivative gain
    $\mathbf{E_{prev}} = \mathbf{E}$ ▷ Update previous error

    $\mathbf{J} = P + I + D$ ▷ Apply joint adjustments
    $robot.control(\mathbf{J})$ ▷ Send joint commands to robot
    $\mathbf{T_c}$ ▷ Get current pose from robot feedback
end for

4. EXPERIMENTAL SETUP

4.1. Design and execution of test cases

The tests are conducted by moving the manipulator's TCP from one predefined point to another. As the pose of the end-effector changes, both translational and rotational motion will be recorded and subsequently plotted on a graph. The movement will be halted when the margin of error reaches a value of 10^{-4}.

4.2. Parameters and metrics for evaluation

In the course of the tests, specific parameters and metrics will be evaluated and compared in order to ascertain the most efficacious combination of PID controller parameters. These metrics are employed for the assessment of the performance and stability of the control system, and for the assurance that the desired system behavior is achieved under varying conditions The figure 4.1 shows some of the metrics mentioned below.

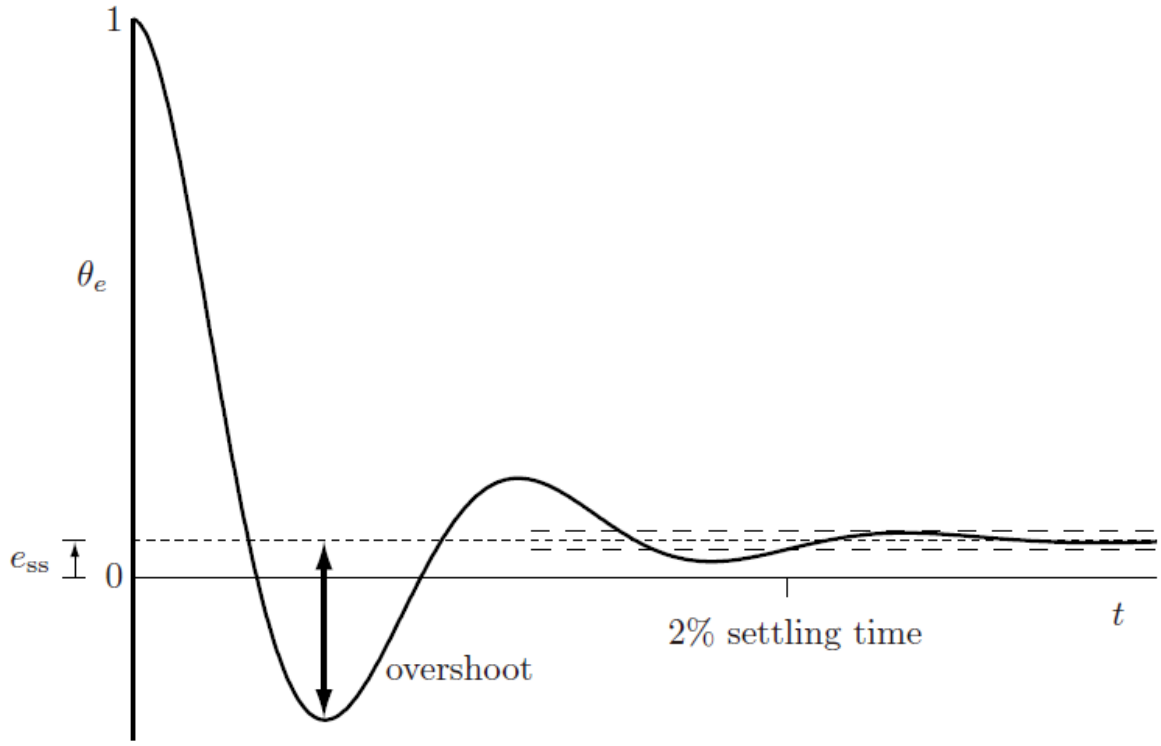

Figure 4.1: An example error response showing steady-state error e_{ss}, the overshoot, and the 2% settling time. [20]

Overshoot is used to describe the extent to which the system's response exceeds the desired value before stabilizing. A high degree of overshoot may indicate an overly aggressive control

strategy, which could potentially lead to instability or damage in applications where precision is paramount. Its reduction is crucial for achieving a smooth and controlled response.

Undershoot occurs when the system's response falls below the desired value before eventually reaching it. While less prevalent than overshoot, undershoot can also be detrimental, particularly in systems where rapid attainment of the end-point is imperative. Both overshoot and undershoot serve as indicators of the system's dynamic response and are directly influenced by the tuning of the PID parameters.

Rise time is defined as the time required for a system to transition from a specified low percentage (commonly 10%) to a high percentage (typically 90%) of the final value. This metric is fundamental to the assessment of the system's responsiveness to alterations in the desired value. A shorter rise time indicates a faster response, which is often desirable in high-performance applications. However, an excessively brief rise time can result in an excessive overshoot, higher jerks and energy consumption, necessitating a balance with other metrics.

Settling time is used to describe the period of time required for the system to respond within a specified error band (typically within $\pm 2\%$ or $\pm 5\%$ of the desired value) following an initial disturbance or change in desired value. Settling time is a critical measure of the rate at which the system stabilizes following a transient event. A shorter settling time is typically preferred, as it indicates that the system can rapidly return to a steady state following disturbances.

Steady-state error quantifies the discrepancy between the desired value and the actual output of the system once it has reached a state of equilibrium and all transient effects have ceased. A modest steady-state error suggests that the controller is efficacious in maintaining the system at the desired set point over time.

Oscillatory behavior in a system occurs when the system's output repeatedly fluctuates around a desired value, often due to underdamping or resonance. These oscillations can cause the system to continually overshoot and undershoot the desired value, leading to instability if not properly controlled. Minimizing oscillations is critical to achieving a stable and efficient system response, which can be done by choosing which of the PID controllers to augment, and which to dampen.

By carefully analyzing these metrics, the optimal combination of PID parameters can be identified to ensure that the control system performs efficiently, with minimal deviation from the desired value and a rapid return to stability after disturbances. In addition to the above metrics, it is also necessary to consider the nature of the system, a free movement of the manipulator. In such a case it should be taken into account that a steady state error is unlikely and the work of the I controller, which could slow down the system, is undesirable [26].

5. RESULTS AND DISCUSSION

5.1. Performance evaluation of trajectory generation

The trajectory generator was evaluated by comparing the desired end-effector pose to the actual pose achieved by the robot, as illustrated in Figure 5.1. The results demonstrated that the trajectory generator was capable of accurately tracking the desired path, with a maximum steady-state error of 10^{-6} meters and 10^{-6} radians. The trapezoidal velocity profile effectively constrained the jerk and ensured smooth motion. The synchronized movement across all joints resulted in efficient point-to-point motions, with the robot moving at the maximum allowable velocity and acceleration while respecting the constraints imposed by the joint limits.

While this motion profile is optimal in controlled environments, exhibiting high levels of accuracy, velocity, and fluidity, it is susceptible to failure when confronted with obstacles or disturbances.

5.2. Analysis of motion control in task space

The motion control analysis was conducted using four distinct controllers: P, PI, PD, PID. Each controller demonstrated unique characteristics with regard to overshoot, settling time, and steady-state error. The following section presents a detailed analysis of each controller.

5.2.1. P controller

The proportional controller was initially assessed, and the results are illustrated in Figure 5.3. It is noteworthy that there was no overshoot, the settling time was minimal, and the steady-state error was less than 10^{-4} m, or 10^{-4} rad. This controller ultimately demonstrated the most optimal performance, exhibiting no overshoot, oscillations, and the shortest settling time among all the controllers.

5.2.2. PI controller

As illustrated in Figures 5.4 and 5.5, the performance of the system with a PI controller is demonstrated. It is evident that the controller induces an overshoot, which is an unfavourable outcome. Moreover, the presence of oscillations is evident, as the robot fails to reach the reference point with the same degree of precision as it does with the P controller. Consequently, the settling time is considerably longer.

5.2.3. PD controller

The PD controller demonstrates some intriguing behavior, namely, motion in the opposite direction from the reference value at the outset of the movement.

An increase in the derivative gain (Kd) was observed to result in a reduction in overshoot and oscillations. However, this also led to an increase in rise time and settling time. With a proportional gain of 6 and a derivative gain of 0.01, the PD controller exhibited a well-damped response with minimal overshoot, as illustrated in Figure 5.6.

Additionally, it is noteworthy that the derivative gain must be so small. Upon testing with larger values of the D gain, the robot exhibited uncontrollable behavior, displaying highly oscillating movements.

However, when the D gain was significantly lowered, the robot resumed normal behavior. This suggests that the robot's smooth movement, without disturbances or unpredictable behaviors, observed when only using the P gain, may be attributed to the robot's inherent stability.

5.2.4. PID controller

Subsequently, the complete proportional-integral-derivative (PID) controller was calibrated. It is anticipated that the integral term will assist in the elimination of steady-state error, while the derivative term should enhance stability and damping.

It is evident that the PID controller still exhibits a move in the opposite direction from the reference value at the beginning.

With $P = 3$, $I = 1$, and $D = 0.01$, the PID controller provided a well-balanced response, as illustrated in Figure 5.8. A value of 0.05 resulted in a faster response with minimal overshoot, as illustrated in Figure 5.9. However, in both instances, the overshoot remained discernible, and the settling time was considerably longer than that observed when a P controller was employed alone. Despite the absence of evident oscillations, no attempt was successful in attaining a superior response compared to that achieved with a P controller alone.

5.3. Comparison with existing approaches

The continued popularity of PID controllers can be attributed to their simplicity and effectiveness in a multitude of control applications, particularly in industrial settings where real-time feedback is of paramount importance. Nevertheless, research has demonstrated that in dynamic and unpredictable environments, their performance can be inferior to that which might be expected. Advanced control methods, such as adaptive control and model-based control, can address this issue by making real-time adjustments to parameters or by employing precise system models to enhance precision. Adaptive controllers, for instance, are capable of dynamically updating their parameters in order to more effectively handle changing conditions. This quality

renders them a preferable choice in systems that exhibit time-varying dynamics in comparison to PID controllers [27],[28].

Notwithstanding the benefits of more sophisticated control methodologies, PID controllers remain highly regarded for their resilience and simplicity of operation. When coupled with real-time feedback, as evidenced by the findings of this study, PID controllers provide a practical solution with minimal computational demand, making them well-suited for numerous real-world applications, such as robotic manipulators. Learning-based methods, such as reinforcement learning (RL), have demonstrated potential for enabling systems to adapt and improve over time. However, they also present a number of challenges. These include substantial computational overhead and longer training times, as evidenced by the use of RL-based controllers for intricate systems such as underwater robots [29].

In contrast, while learning-based approaches can enhance performance and facilitate the completion of complex tasks over time, they are frequently less viable in real-time systems due to their elevated complexity and potential instability during training. Research indicates that a hybrid approach—utilizing PID for initial stability and RL for optimization—may serve to bridge this gap. Nevertheless, PID remains the more practical option in environments necessitating immediate and dependable control, particularly in industrial and robotic applications [30].

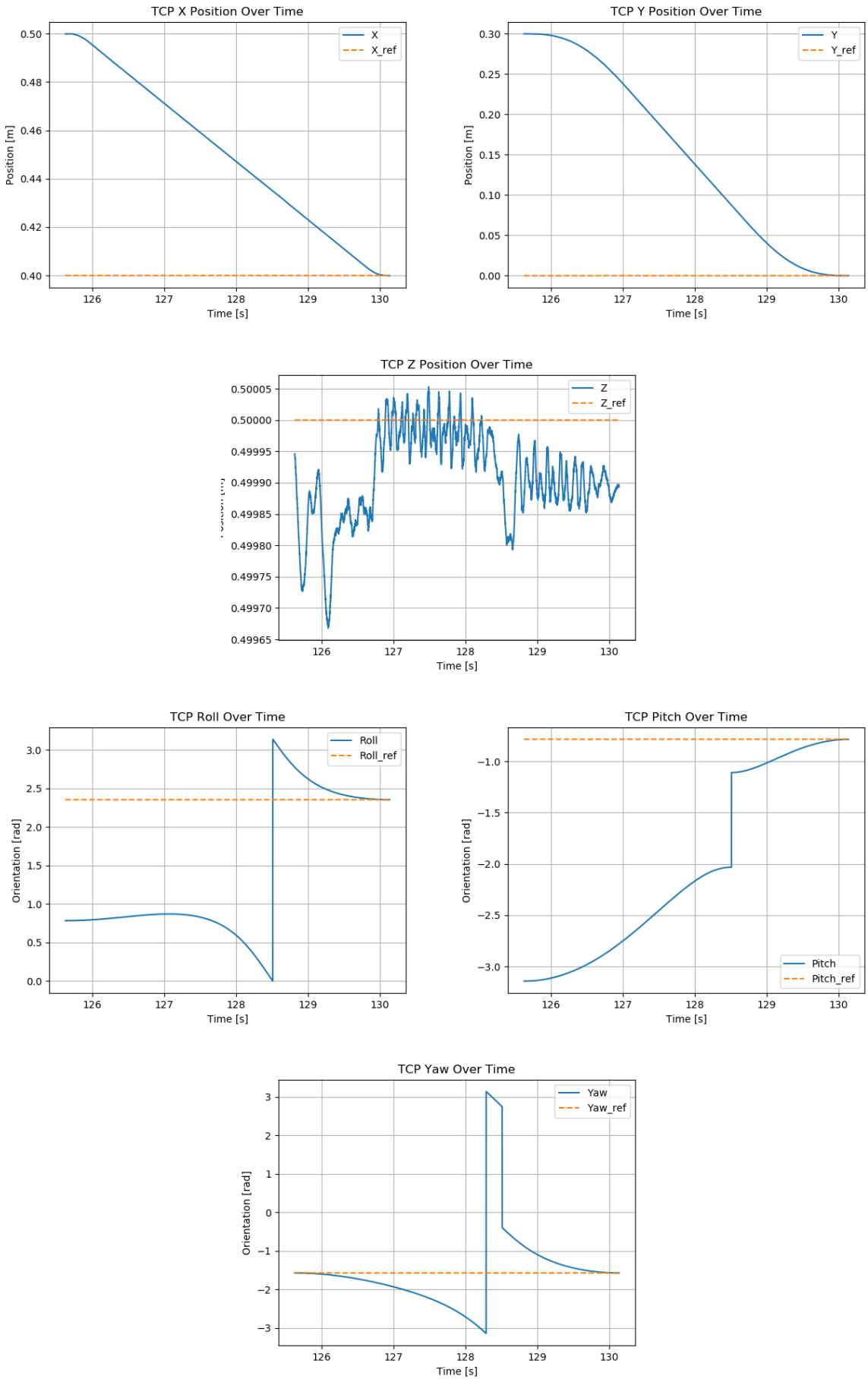

Figure 5.1: Trajectory generator performance

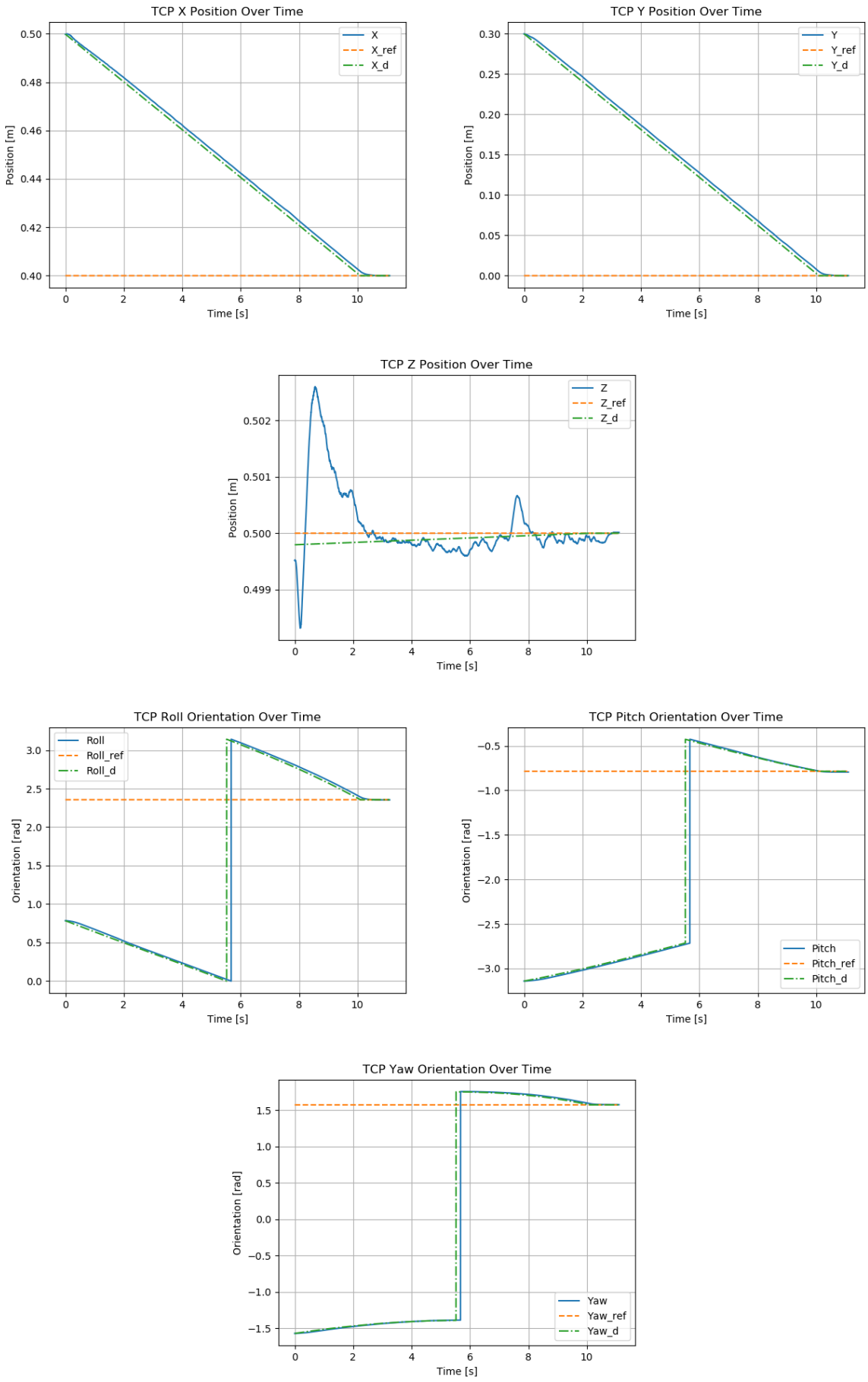

Figure 5.2: P gain at 5.5

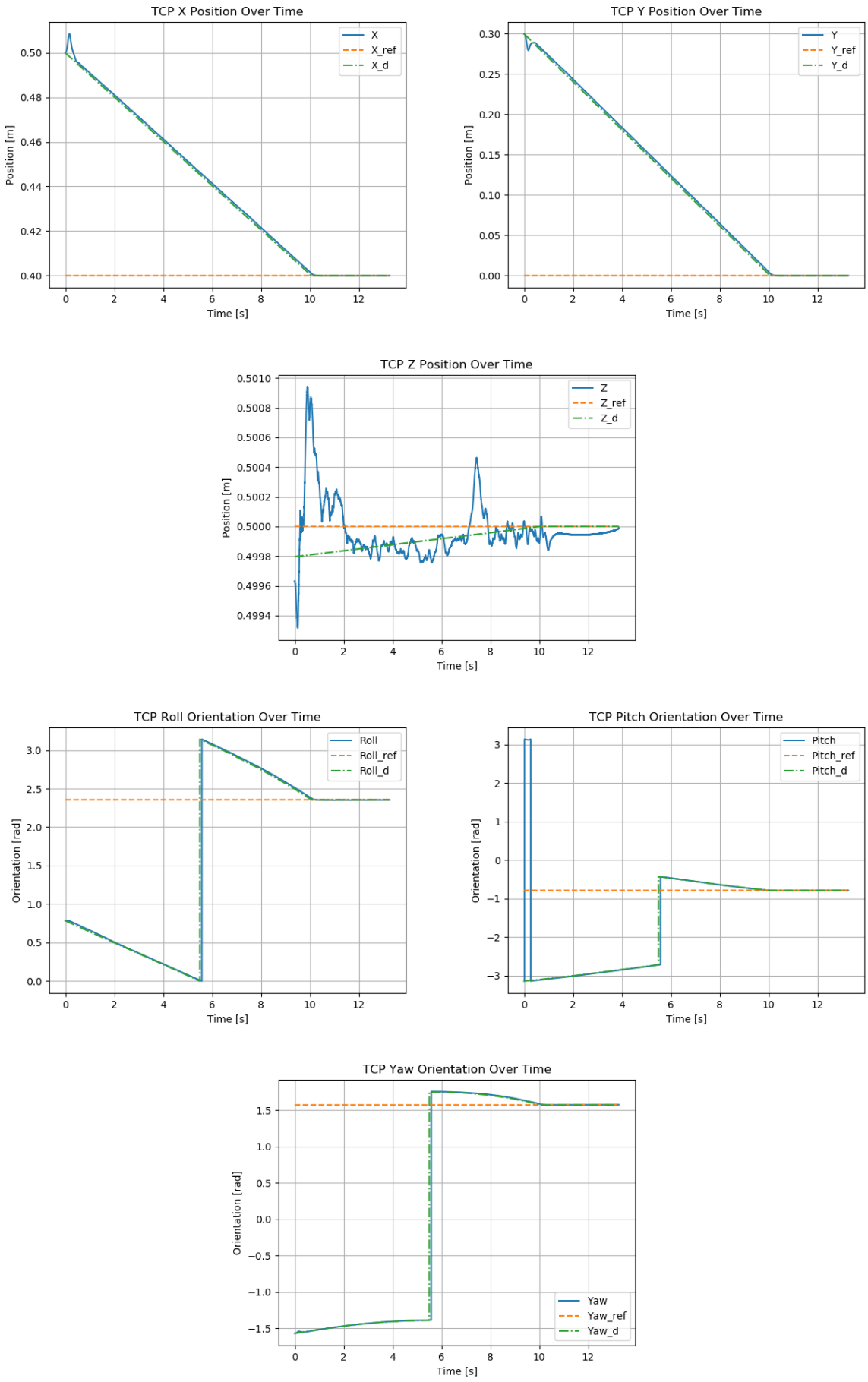

Figure 5.3: P gain at 10

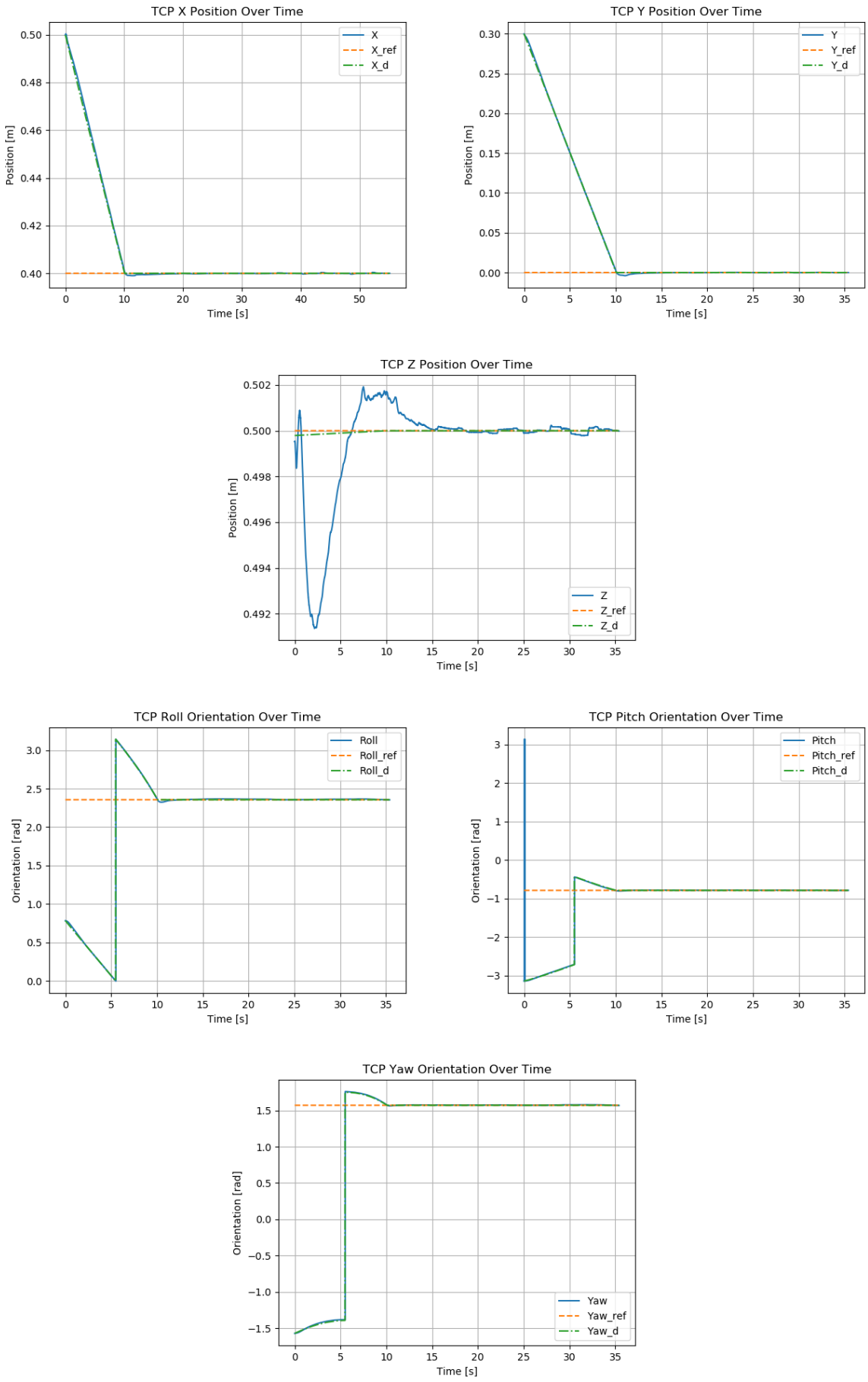

Figure 5.4: P gain at 6 and I gain at 1

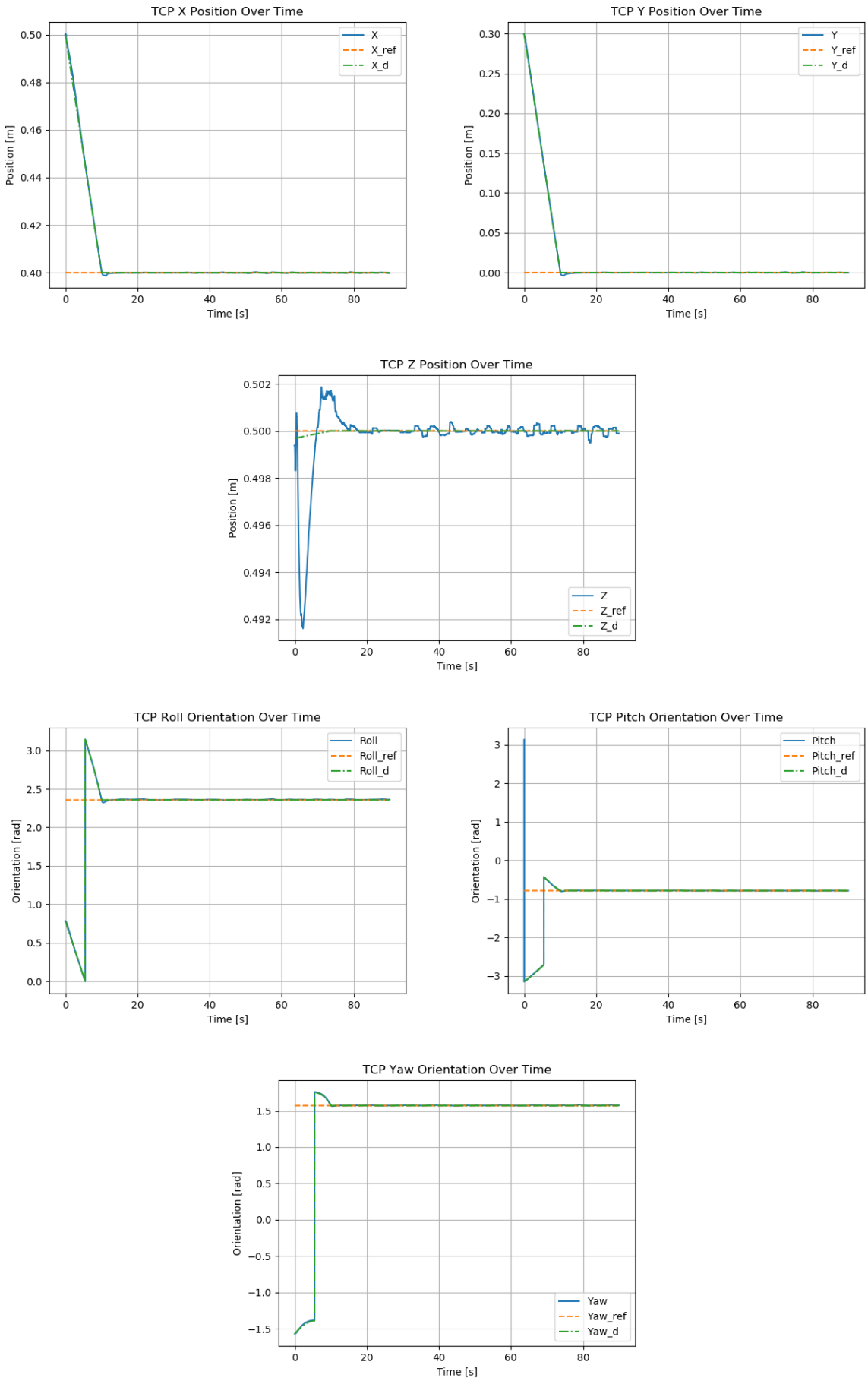

Figure 5.5: P gain at 6 and I gain at 3

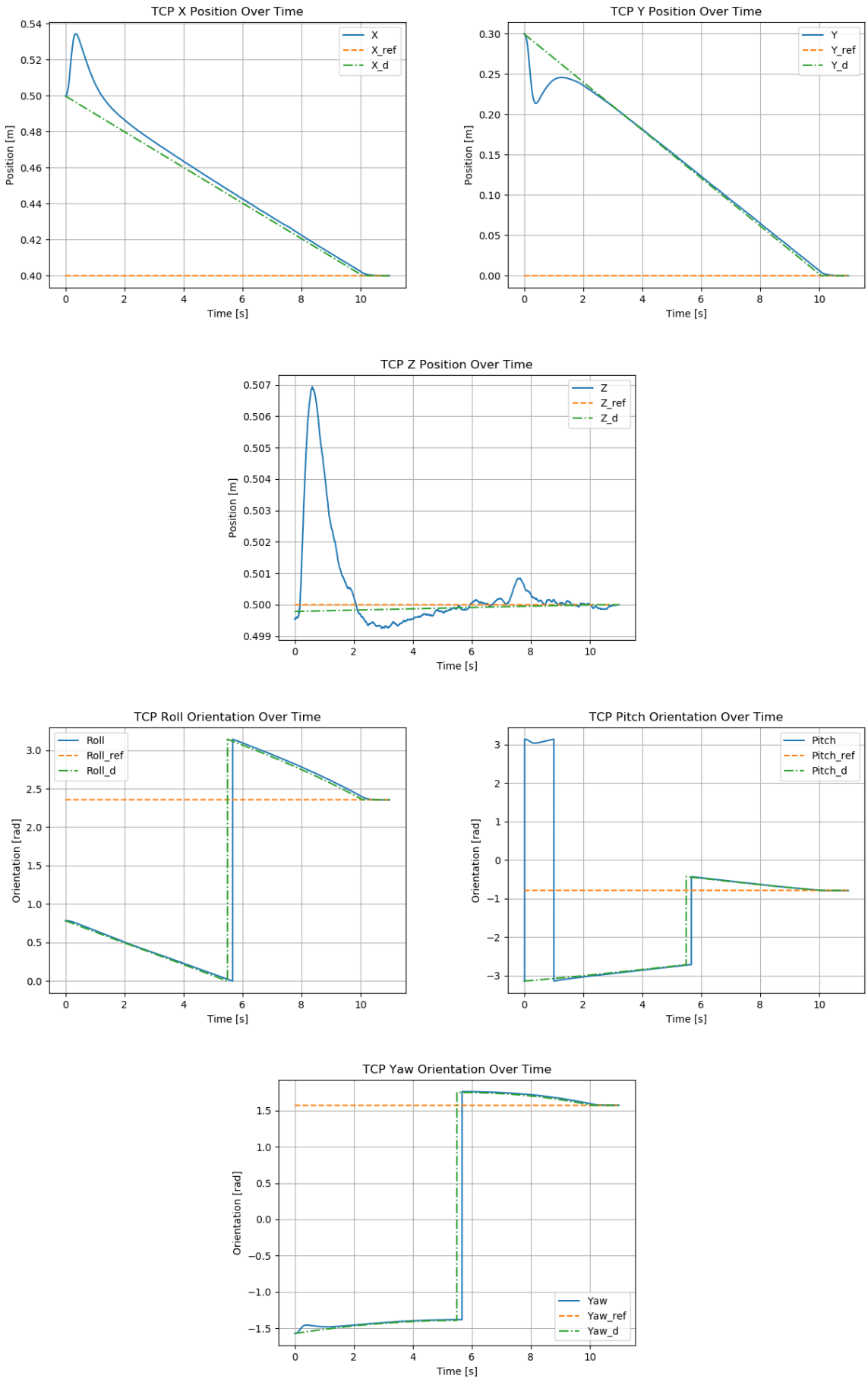

Figure 5.6: P gain at 6 and D gain at 0.01

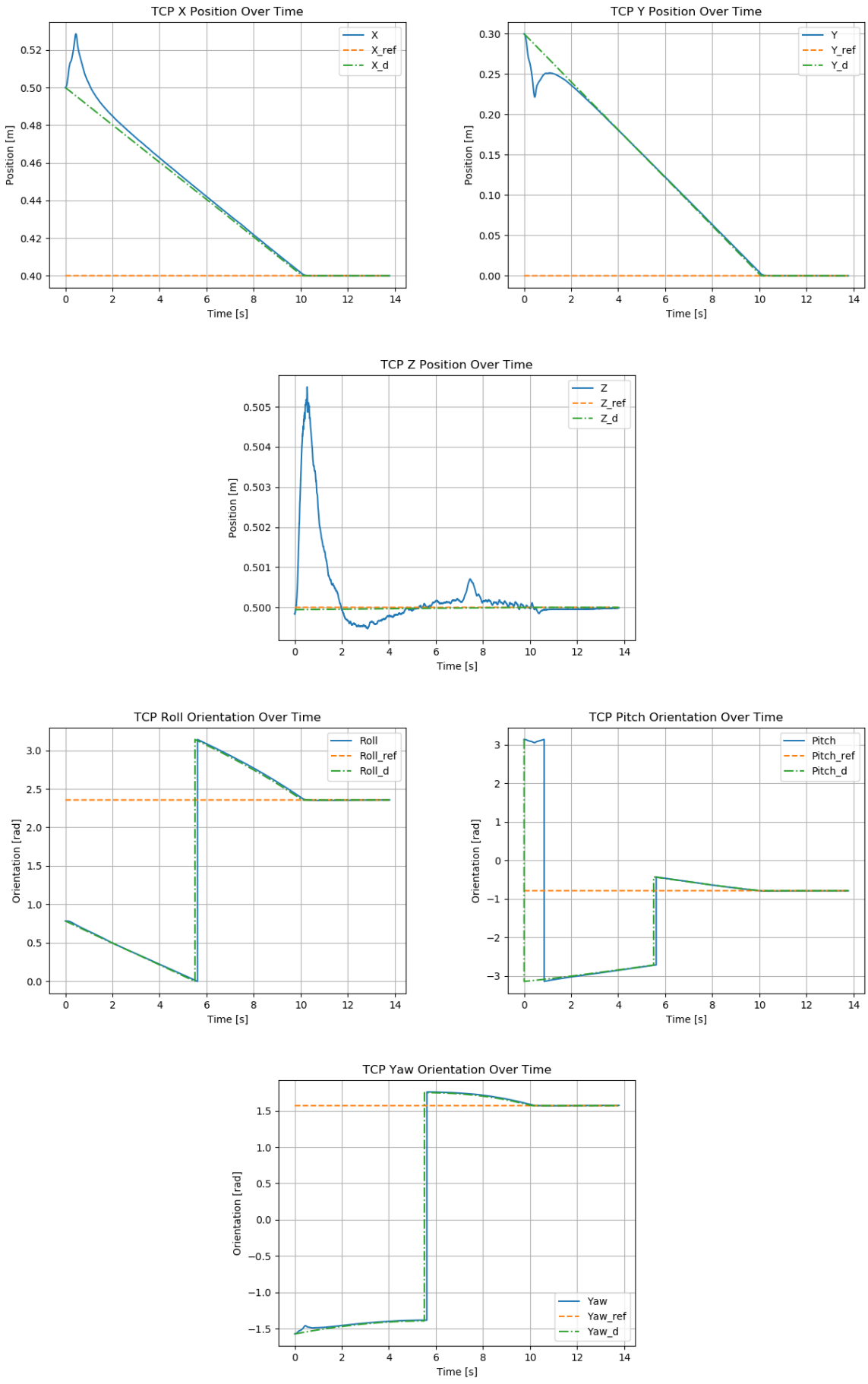

Figure 5.7: P gain at 10 and D gain at 0.05

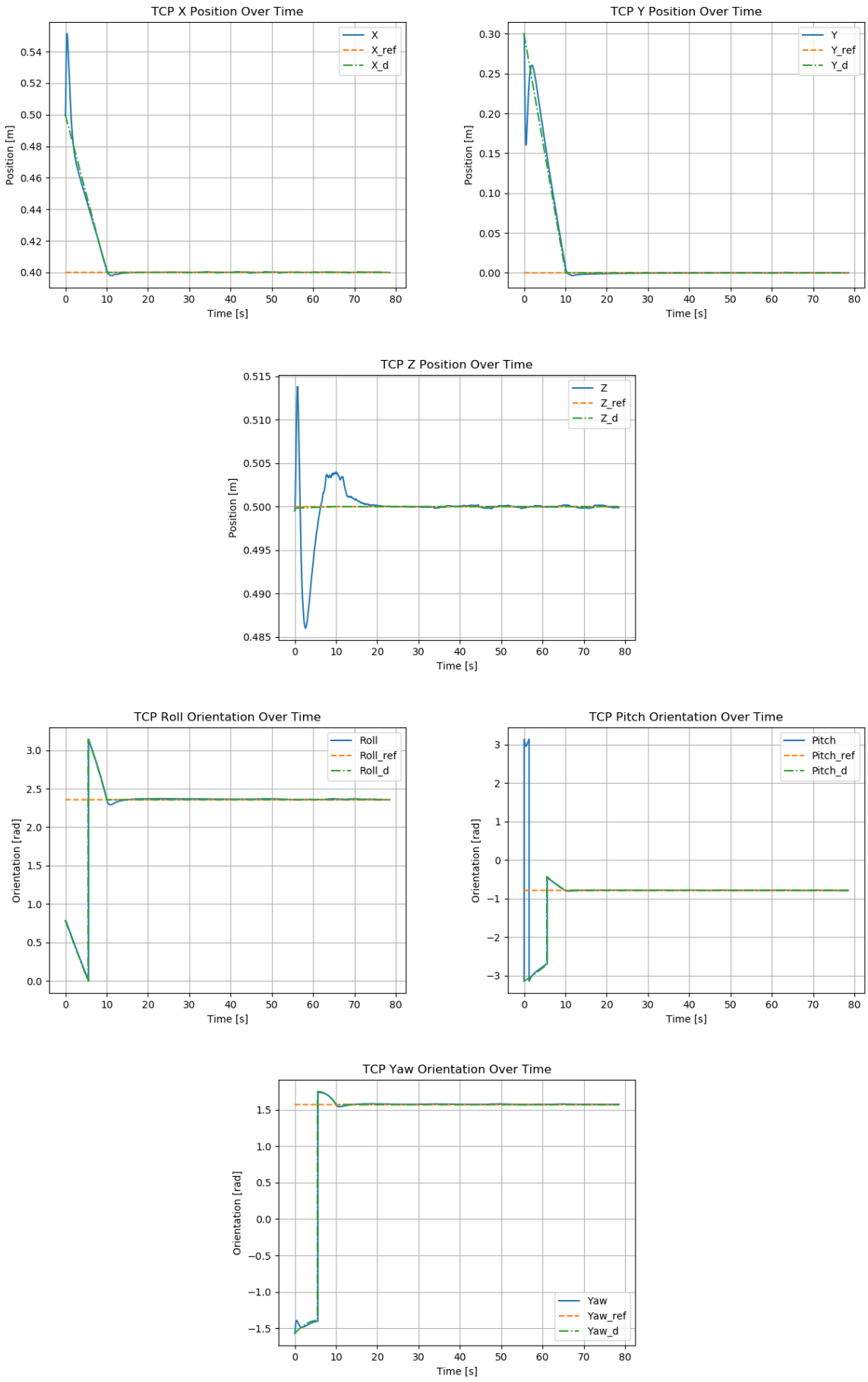

Figure 5.8: P gain at 3, I gain at 1 and D gain at 0.01

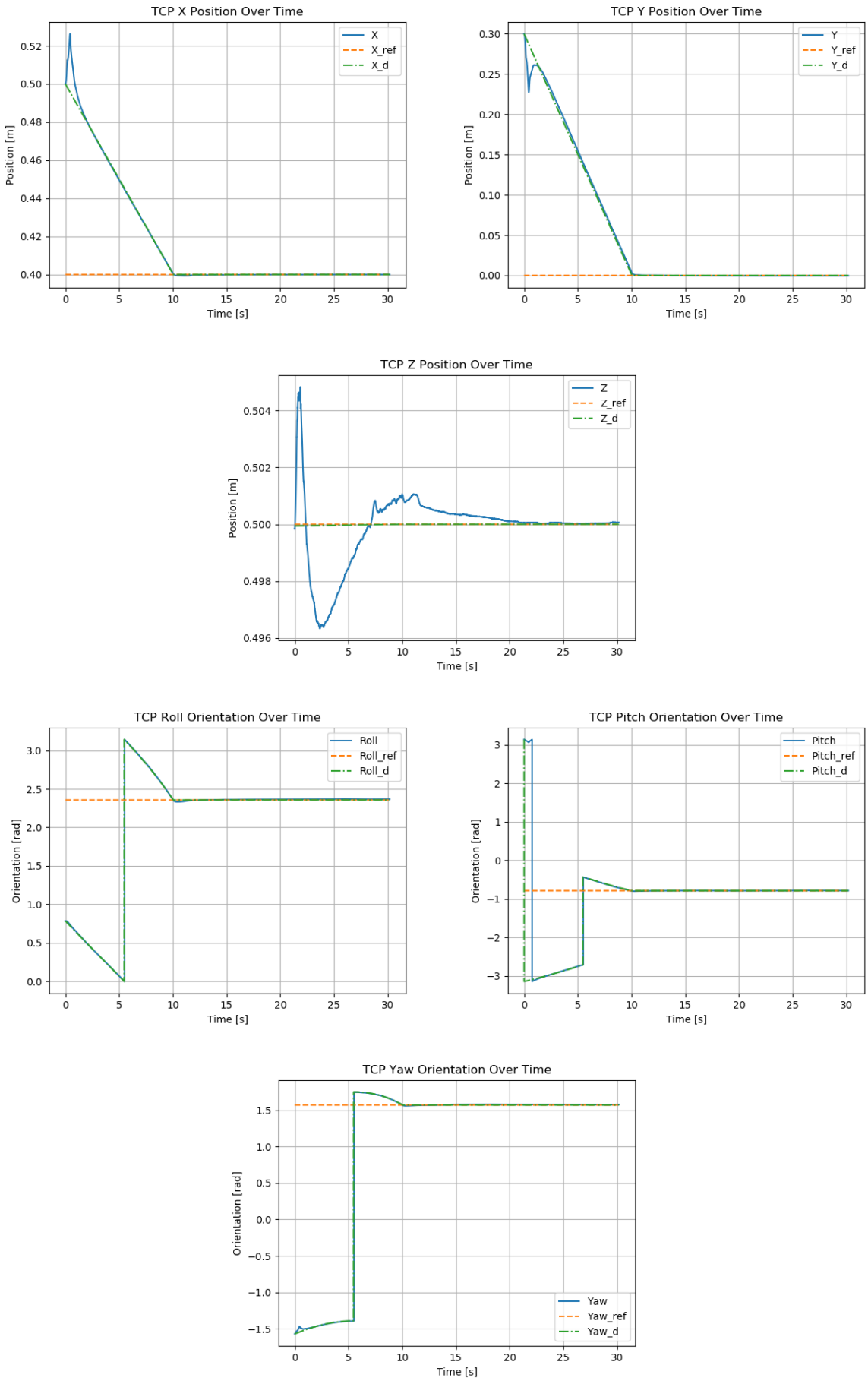

Figure 5.9: P gain at 10, I gain at 2 and D gain at 0.05

6. CONCLUSION AND FUTURE WORK

6.1. Summary of findings

This thesis presents the successful development and evaluation of motion control strategies for a redundant robotic arm. The implementation and testing of several controllers, including P, PI, PD, and PID, were conducted to achieve this objective. The findings of the experiments were presented and discussed.

The trajectory generation method, which employed a trapezoidal velocity profile, demonstrated remarkable efficacy in guaranteeing the smooth, precise, and regulated movement of the robotic arm. Synchronizing the movements of all joints enabled the trapezoidal profile to optimize both precision and time efficiency, resulting in rapid and stable robotic movements. While this approach is effective in limiting jerk motions, it is less optimal when faced with a dynamic and unpredictable environment.

Among the controllers tested, the P controller exhibited the best results, demonstrating no overshoot, minimal settling time, and approaching high accuracy. The other controllers, PI, PD, and PID, exhibited limitations, either showing signs of oscillations, long settling times, and having problems with accuracy.

6.2. Contributions and implications

The work presents the implementation and evaluation of various control techniques on a redundant robotic arm. The findings contribute to the existing body of knowledge in the field of robotic motion control, particularly in the context of task space control. By identifying the strengths and limitations of different control strategies, this research can inform future developments in industrial and medical robotics, where precision is of paramount importance.

Furthermore, the results illustrate the potential for employing straightforward control techniques, such as PID, in contexts where more sophisticated methodologies may be impractical due to computational constraints or system limitations.

6.3. Recommendations for future research

Although this study concentrated on the use of proportional-integral-derivative (PID) controllers, future research could investigate more sophisticated techniques, such as adaptive or learning-based controllers. Such methods could facilitate dynamic adjustment of control parameters in real time, allowing for the consideration of changing conditions, such as varying payloads or environmental factors.

The optimization of control parameters is a further avenue for future research. The proportional, integral, and derivative (PID) parameters in this study were manually tuned based on

visual analysis and performance metrics. Such techniques could include the application of advanced methods, such as the root-locus method. Future work could even involve the application of optimization techniques, such as genetic algorithms or machine learning, for the systematic tuning of these parameters with a view to improving performance.

The objective of optimizing robot performance is to enhance the operational efficacy of the robot. Although the focus of this study was not on robot performance, the investigation of methods for determining optimal constraints for specific usage scenarios could lead to improvements in performance.

While this thesis utilized a preexisting code to address the inverse kinematics problem of a redundant robot, future work could focus on optimizing the solution to enhance the overall outcome. The current function employed may not consistently yield the optimal result, occasionally leading to unanticipated joint configurations.

Subsequent research could apply the findings of this study to specific applications such as robotic-assisted surgery or autonomous manufacturing. Testing the control strategies in these real-world scenarios could provide further insights into their effectiveness and potential improvements.

BIBLIOGRAPHY

[1] Khalil W, Dombre E. Modeling, Identification and Control of Robots. Kogan Page Science paper edition Modeling, identification & control of robots. Elsevier Science; 2004. Available from: https://books.google.hr/books?id=nyrY0Pu5kl0C.

[2] Kremer V. The Use of Quaternions and SLERP for Character Animation; 2008. Available from: https://api.semanticscholar.org/CorpusID:61216861.

[3] Chu WL, Lin CJ, Chen YY. Redundant Robot with Pneumatic Artificial Muscles for Rehabilitation Works Using Iterative Learning Control. Applied Sciences. 2022;12(17). Available from: https://www.mdpi.com/2076-3417/12/17/8419.

[4] Elias AJ, Wen JT. Redundancy parameterization and inverse kinematics of 7-DOF revolute manipulators; 2024. Available from: https://arxiv.org/abs/2307.13122.

[5] Rong Y, Dou T, Zhang X. Optimal resource allocation method and fault-tolerant control for redundant robots. Mechanical Sciences. 2023;14(2):399-412. Available from: https://ms.copernicus.org/articles/14/399/2023/.

[6] Milecki A, Nowak P. Review of Fault-Tolerant Control Systems Used in Robotic Manipulators. Applied Sciences. 2023;13(4). Available from: https://www.mdpi.com/2076-3417/13/4/2675.

[7] Lu X, Wang C, Jin X, Li J. A Flexible Surgical Instrument for Robot-Assisted Minimally Invasive Surgery. Actuators. 2022;11(8). Available from: https://www.mdpi.com/2076-0825/11/8/206.

[8] Ma B, Jiang Z, Liu Y, Xie Z. Advances in Space Robots for On-Orbit Servicing: A Comprehensive Review. Advanced Intelligent Systems. 2023;5(8):2200397. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202200397.

[9] Long H, Li G, Zhou F, Chen T. Cooperative Dynamic Motion Planning for Dual Manipulator Arms Based on RRT*Smart-AD Algorithm. Sensors. 2023;23(18). Available from: https://www.mdpi.com/1424-8220/23/18/7759.

[10] Conti S. How redundancy and distributed control are helping make robots autonomous. Nature Reviews Physics. 2023 Sep;5(9):501-1. Available from: https://doi.org/10.1038/s42254-023-00636-6.

[11] Chiaverini S, Siciliano B, Egeland O. Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. IEEE Transactions

on Control Systems Technology. 1994 June;2(2):123-34. Available from: https://ieeexplore.ieee.org/document/294335.

[12] Haddadin S, Parusel S, Johannsmeier L, Golz S, Gabl S, Walch F, et al. The Franka Emika Robot: A Reference Platform for Robotics Research and Education. IEEE Robotics Automation Magazine. 2022 June;29(2):46-64. Available from: https://ieeexplore.ieee.org/document/9721535.

[13] Rogel A, Savery R, Yang N, Weinberg G. RoboGroove: Creating Fluid Motion for Dancing Robotic Arms. In: Proceedings of the 8th International Conference on Movement and Computing - MOCO '22. New York, NY, USA: Association for Computing Machinery; 2022. Available from: https://doi.org/10.1145/3537972.3537985.

[14] libfranka — Franka Control Interface (FCI) documentation;. [Accessed on 09/23/2024]. Available from: https://frankaemika.github.io/docs/libfranka.html.

[15] Gaz C, Cognetti M, Oliva A, Robuffo Giordano P, De Luca A. Dynamic Identification of the Franka Emika Panda Robot With Retrieval of Feasible Parameters Using Penalty-Based Optimization. IEEE Robotics and Automation Letters. 2019 Oct;4(4):4147-54. Available from: https://ieeexplore.ieee.org/document/8772145.

[16] Zhang K, Sharma M, Liang J, Kroemer O. A Modular Robotic Arm Control Stack for Research: Franka-Interface and FrankaPy; 2020. Available from: https://arxiv.org/abs/2011.02398.

[17] Ogata K. Modern control engineering. 5th ed. Upper Saddle River, Nj: Prentice Hall; 2010.

[18] Craig JJ. Introduction to robotics : mechanics and control. Upper Saddle Hall: Pearson Educacion Internacional; 2005.

[19] Nguyen-Tuong D, Peters J. Model learning for robot control: a survey. Cognitive Processing. 2011 Nov;12(4):319-40. Available from: https://doi.org/10.1007/s10339-011-0404-1.

[20] Lynch KM, Park FC. Modern robotics : mechanics, planning, and control. Cambridge: Cambridge University Press; 2017.

[21] Heo HJ, Son Y, Kim JM. A Trapezoidal Velocity Profile Generator for Position Control Using a Feedback Strategy. Energies. 2019;12(7). Available from: https://www.mdpi.com/1996-1073/12/7/1222.

[22] Siciliano B, Sciavicco L, Villani L, Oriolo G. Robotics. 1st ed. Advanced Textbooks in Control and Signal Processing. London, England: Springer; 2008. Available from: https://doi.org/10.1007/978-1-84628-642-1.

[23] Design Trajectory with Velocity Limits Using Trapezoidal Velocity Profile - MATLAB & Simulink - MathWorks Deutschland;. [Accessed on 30-08-2024]. Available from: https://de.mathworks.com/help/robotics/ug/design-a-trajectory-with-velocity-limits-using-a-trapezoidal-velocity-profile.html.

[24] Bilancia P, Schmidt J, Raffaeli R, Peruzzini M, Pellicciari M. An Overview of Industrial Robots Control and Programming Approaches. Applied Sciences. 2023;13(4). Available from: https://www.mdpi.com/2076-3417/13/4/2582.

[25] He Y, Liu S. Analytical Inverse Kinematics for Franka Emika Panda – a Geometrical Solver for 7-DOF Manipulators with Unconventional Design. In: 2021 9th International Conference on Control, Mechatronics and Automation - ICCMA; 2021. p. 194-9. Available from: https://ieeexplore.ieee.org/document/9646185.

[26] Karu ZZ. Signals and Systems Made Ridiculously Simple. ZiZi Press; 1995. Available from: https://books.google.fr/books?id=w6LmngEACAAJ.

[27] Borase RP, Maghade DK, Sondkar SY, Pawar SN. A review of PID control, tuning methods and applications. International Journal of Dynamics and Control. 2021 Jun;9(2):818-27. Available from: https://doi.org/10.1007/s40435-020-00665-4.

[28] Lee D, Koo S, Jang I, Kim J. Comparison of Deep Reinforcement Learning and PID Controllers for Automatic Cold Shutdown Operation. Energies. 2022;15(8). Available from: https://www.mdpi.com/1996-1073/15/8/2834.

[29] Lotfi F, Virji K, Dudek N, Dudek G. A comparison of RL-based and PID controllers for 6-DOF swimming robots: hybrid underwater object tracking; 2024. Available from: https://arxiv.org/abs/2401.16618.

[30] Kim I, Jeon Y, Chae J, You D. Deep Reinforcement Learning for Fluid Mechanics: Control, Optimization, and Automation. Fluids. 2024;9(9). Available from: https://www.mdpi.com/2311-5521/9/9/216.

A. APPENDIX

Link to the *Github* repository containg the code used for this study: `https://github.com/aaronr6/bachelors_thesis.git`