1 Review of Robotic Gripping Controller

Humans are hard to please, we experience fatigue, emotions, and demand for higher pay. Therefore, industries have adopted automation concept since the early 1970s, during the Third Industrial Revolution [1]. Automation uses computers and robots to assist humans in repetitive, predictable tasks and could theoretically operate 24 hours a day, 7 days a week while keeping the long-term operating cost to a minimum.

From a simple Roomba vacuum cleaner to the state-of-the-art robotic arm assembly system found in car manufacturing, the demand for personal and industrial robots is on a rapid rise. As such the global robotics market was valued by Mordorintelligence to be at USD 27.73 billion in 2020 and is expected to grow to USD 74.1 billion by the end of 2026 [2].

There is a race to develop a robot that could 'do it all'. Amazon, one of the leading e-commerce giants has gone as far as hosting a yearly competition that challenges teams around the world to design and showcase robots that are capable of separating different types of objects ranging from soft toys to toothbrushes and sort them according to the customer's order [3]. Not only that, recently an interview conducted by Tom Scott on Orcardo Grocery warehouse, a 'Hive system' is used for online grocery shopping, where approximately 2300 robots are in charge of identifying, picking, placing and sorting goods from the warehouse to each customer basket ready to be packed and delivered to the users around the UK in a 5-hour window [4].

Although robots have evolved massively since the early 1970s, it is still far from what humans are capable of. Numerous issues hinder the performance of robots. These issues motivate this research to investigate and attempt to mitigate the effects caused by the issues.

1.1 Problem Analysis

The advantages of having robots to assist humans have only been realised about half a decade ago, but humans have been developing buildings, structures, and systems for centuries. It comes as no surprise that the biggest problem faced by robots is that the world around them is optimised for humans and not robots [3]. Humans have a few powerful tools at their disposal that the current generation of robots are lacking, such as powerful processing power (Brain), the ability to move around freely and easily (Legs) and the ability to grasp and manipulate objects (Hands). AlphaGo has demonstrated the advancement in processing power while Boston Dynamics shows that robots can move freely with leg-like configuration, however, research and development into grasping and manipulation is relatively slow.

For robots to interact with the world, they require the ability to grasp and manipulate objects freely and effectively. To achieve this, it is crucial to develop both the hardware and software of a gripper.

One of the most advance grasping control systems, Dex-Net 4.0 is capable of grasping 300 items per hour with a success rate of 95% [5]. On the contrary, an average human is capable of grasping between 400 to 600 items per hour [3]. To overcome this challenge, the repetitive nature of grasping and interaction behaviour between robots and environment need to be realised and thus the controller has to be able to fulfil a range of criteria:

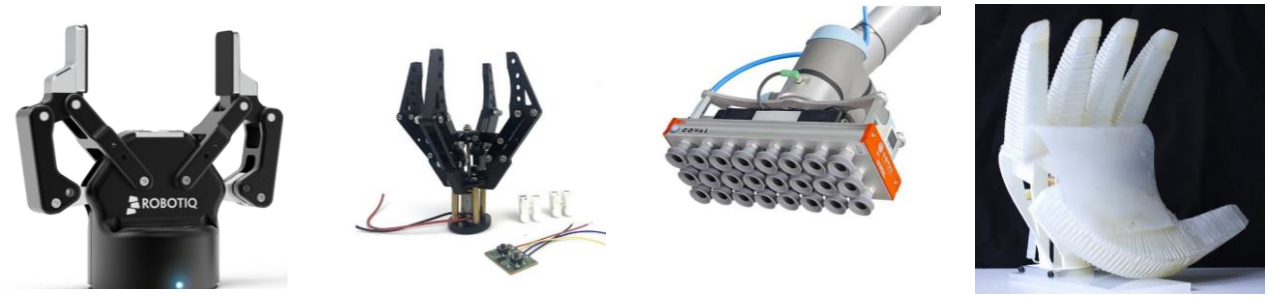
- Learn from previous grasping errors/mistakes.

- Able to correct itself from environmental disturbances.
- Ensure the deadtime between each operation is to the minimum.
- Uses the least amount of force.
- Capable of grasping a range of objects.

1.2 Approach to Robotic Grasping

Robotic grasping can be split into 2 categories, Hardware and Controller. Hardware deals with the physical structure of the gripper including what type of gripper to be used as well as what materials it should be made out of while controller plans and execute the action to complete the required task safely and securely.

Grasping requires the gripper to interact with the environment and object. Over the past decade, different gripper hardware has been developed and tested. All gripper has one common feature – finger-like structure. These fingers come in various shapes and sizes depending on the task requirement such as a simple parallel 2-plate gripper (Figure 1.1a), fingers arranged in claw structure for a more stable grasp (Figure 1.1b), suction cup gripper controlled by pressurised air to handle delicate objects by spreading the pressure out evenly (Figure 1.1c) and incorporating 'softness' to gripper by exploiting compliance properties which are capable of curling and straightening by adjusting the pressurised air sent to the fingers (Figure 1.1d).[3]



(a) Parallel Plate Gripper (Adapted from [40])
(b) Claw Gripper (Adapted from [41])
(c) Suction Cup Gripper (Adapted from [42])
(d) Compliance Gripper (Adapted from [43])

Figure 1.1: Common types of Grippers

Controller for grasping can be further split into 3 categories: Task selection, Motion planning and Motion Control. There are 3 different autonomy levels of grasping control illustrated in Figure 1.2 as identified by Ozawa [6]. Master-Slave system allows the operators to use a miniature robot to control the gripper. Where shared autonomy only requires the operator to manually select grasping tasks and provide minimal correction on the gripper operation. Recently with the development of Machine Learning, the task of the operator reduces to as little as instructing the robot with general instructions such as moving items from one location to another and the algorithm would autonomously decide on the optimal grasping strategies [3][11].

Task selection decides how the gripper should grasp an object. The simplest way is for the robot to 'copy' humans' grasping methods and actions (grasp taxonomy). There have been extensive studies on this area, with some generalising human grasping into 6 ways [7][8], while others into 16 ways [9]. Some studies have even gone as far as analysing how interaction forces are distributed during grasping action. But these different grasping methods can be further generalised based on the shape of grasping into 5 classes by Ozawa [6] as illustrated in Figure 1.3.

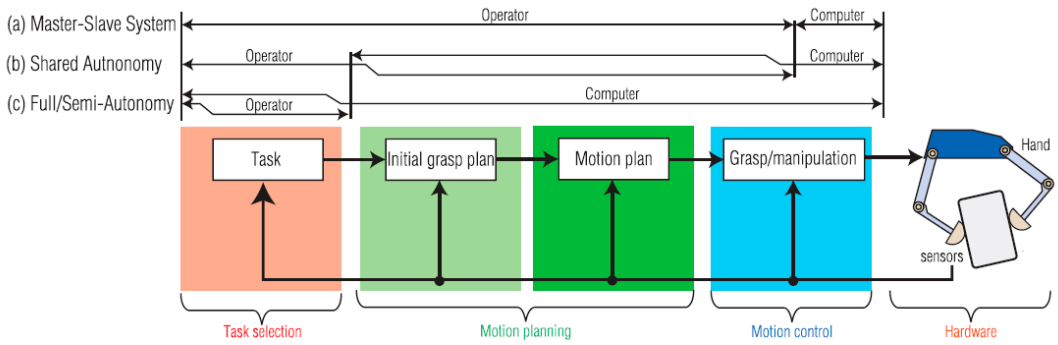


Figure 1.2: Autonomy of robotic gripper (Adapted from [6])

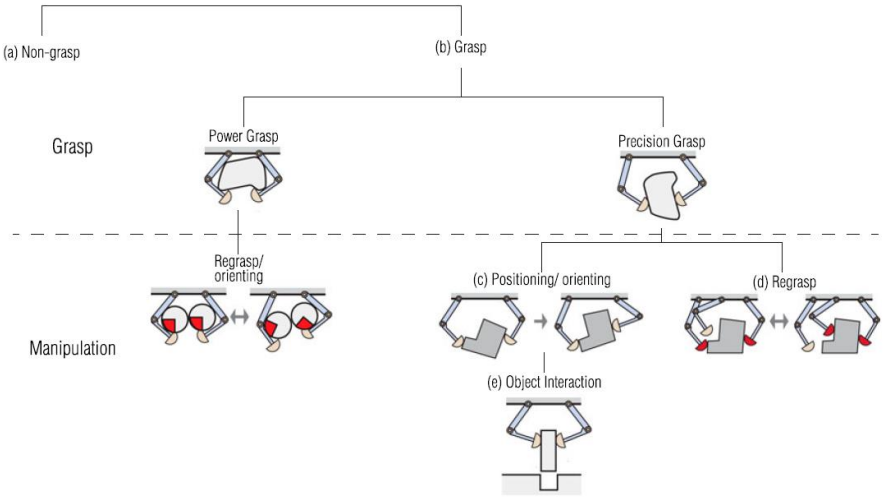


Figure 1.3: Grasp category gripper (Adapted from [6])

Motion planning plans the initial grasping point and motion for a successfully grasp. The controller requires some information about the object such as size and orientation to deduce a safe grasp, manipulate and transport strategy. For the controller to work out a grasping strategy, contacts between gripper and object are modelled. There are a few widely used contact modelling models, such as frictionless point contact which only models normal force, friction point contact which also models tangential force, and soft finger contact which also models torsional moment. Details of these basic contact models are explained by Ozawa [6].

Finally, motion control corrects any error to allow the gripper to realise the initially planned motion. The following section will investigate different types of motion control controllers for robotic grasping in chronological order.

1.3 A History of Controllers for Robotic Grasping

The role of grasping controller is to ensure that the object is grasped, manipulate, and transported securely. Generally, this requires the control of finger movement.

Sequential controller is the simplest controller where it selects a certain grasping motion from a series of pre-programmed motions. In the experiment conducted by Tamashita [12], the controller is pre-programmed with 20 primary motions and the task of the controller is to follow one of the pre-programmed motions selected by the operator. This controller cannot however grasp the object

optimally as it is not able to produce the ‘overall best’ motion but rather the ‘best’ from a list of pre-programmed motions.

Reflexive controllers increase the grasping possibility to infinite and can be imagined by observing a young baby grasping an object. As an object gets in contact with the palm, the baby instinctively closes his fingers and grasps the object. If attempts are made to remove the object, the baby would grasp harder to oppose the motion. Commonly, light or optical sensors are embedded inside each jaw of the gripper [13][14] to detect contact between the gripper and object. The flaw of the controller is its inability to detect the applied force which could damage the goods as excessive force is applied. This problem could be mitigated with some form of external sensors such as visual or force.

Hybrid controller controls both grasping position and forces simultaneously. Many hybrid controllers require the internal force to be dynamically decoupled from its motion by using model-based controllers such as the one investigated by Shin and Lee [15]. Where the controller uses the error between the desired and actual position and force signal to alter the joint acceleration to reduce excessive force applied to the object.

Often, system dynamics are hard to be abstracted/modelled. Thus, adaptive controller is one of the earliest non-model-based controllers. Adaptive controller retains excellent motion tracking ability even when there is uncertainty (mass, inertia, centre of mass) when the internal force of the contact is dynamically decoupling [16]. Other advantages of adaptive controllers over PD or hybrid controllers are decent robustness, much higher tracking accuracy, and performs well at high speed despite having uncertainties [17]. Performing well when operated at high speed is desirable as this allows the gripper to be able to pick more items per hour.

Impedance controller on the other hand is simple, as it models the interaction force using mechanical impedance. Mass-spring-damper [18] structure is often used to model the interaction forces between the gripper and environment without the need to know the system dynamics or to decouple the internal forces. To reduce the complexity of controlling a multi-finger gripper, often multiple fingers involved in manipulation are combined and represented by a single mass and inertia structure [19]. Due to its simple structure, impedance control has been gaining a lot of attention in recent years and is currently one of the most used control schemes for grasping.

All controllers discussed above does not utilise errors from previous iteration which are often rich in information [20]. Therefore, this motivates learning controllers. Iterative learning controller (ILC) proposed by Nakayama [21] uses a non-model-based P-type controller to minimise the position and force error. After 30 iterations, the gripper trajectory was identical to the ideal trajectory. Nakayama also states that the convergence of position tracking would always lead to the convergence force applied. However, research into ILC controlled grippers is very limited.

A summary of the advantages and disadvantages of each controller discussed is shown in Table 1.1.

Table 1.1: Comparison between different grasping controllers

Control Algorithm	Complexity	No. of grasping sequence	Force control	Decouple internal forces?	Knowledge of system dynamics?	Use previous iteration error?	Main issues
Sequential	Simple	Finite	No	N/A	N/A	No	Finite number of grasping options
Reflexive	Simple	Infinite	No	N/A	N/A	No	Unable to grasp with optimal force
Hybrid position/force	Moderate	Infinite	Yes	Yes	Yes	No	Needs to compute complicated mathematics Requires system dynamics
Adaptive	Moderate	Infinite	Yes	Yes	No	No	Still needs to compute complicated mathematics
Impedance	Simple	Infinite	Yes	No	Generally no	No	Does not learn from previous iterations
Learning	Moderate	Infinite	Yes	No	Depends on scheme	Yes	Limited research

1.4 Importance of Simulation

With the recent rapid development of computer hardware technology, simulation has become a crucial first step in product development and research. Simulation offers a platform where research, development and modification can be tested in a virtual environment within a short time frame with low budget commitment before building a prototype. On top of that, simulation is also capable of providing a valuable glimpse at any potential system flaws in a risk-free and safe environment. [22]

2 Detailed Specification

In the literature review, the strength and weaknesses of different grasping controllers were identified. It was discussed that robotic grasping often operates in a repetitive environment and most development has not utilised previous iteration errors which are often information-rich. The research conducted on ILC concludes that the actual trajectory of the gripper can match the ideal trajectory but converges relatively slow since it uses non-model method. The limitation of current-generation grasping controller identified in Section 1.1 remains.

A novel approach is to design a grasping controller that utilises the advantages of controllers discussed in Section 1.3 to solve the limitation discussed in Section 1.1. To achieve this, the following design requirements (DR) were identified. Upon achieving these DR, the grasping quality and speed would be improved.

2.1 Design Requirements

The following design requirements focus on the selection of simulation environment and model used to evaluate the controller algorithm.

DR1. Simulation model **closely relates to real industrial robotic configuration**. This ensures that the controller is tested on a representative system and the result can correlate to the industry.
DR2. Simulation should **model real-world interaction dynamics/forces**. This allows minimal inaccuracy between simulation environment and real-world thus ensuring the results are representative.
DR3. The simulation should **contain visual models**. Visual models are crucial to not only showcase the achievement of the controller but also to assist in discovering any potentially 'hidden' issues that might not be captured by just analysing the data.

The following design requirements highlight the aim of the controller to eliminate or reduce limitations raised in Section 1.1.

DR4. The robot (arm and gripper) shall be able to **reach and grasp the object accurately**. The normalised position error should be less than 0.01 within the first 15 iterations of operations. This ensures the robot has a fast start up time.
DR5. The robot (arm and gripper) shall be able to **reach and grasp the object quickly**. The complete reach-grasp-return trajectory should be completed within 5 to 7 seconds to give the robot a 514 to 720 means pick per hour.
DR6. The gripper shall grasp the object with the **least amount of force**. It shall not exceed 25% of the ideal force within 20 iterations of grasping the object securely. This ensures minimal excessive force is applied while considering any uncertainties in the environment and object modelling.
DR7. The robot (arm and gripper) shall be capable of **grasping objects of different mass**. The object would range between 0.2kg to 2kg. This ensures the robot is capable of operating in different environments.
DR8. The robot (arm and gripper) shall be capable of **correcting itself for a range of repetitive disturbances**. The robot will be subjected to the following disturbance scenarios (DS)
    DS1. **Constant error**. The error shall have a magnitude of 5-10%. This arises when there is an initial state error due to inaccurate tuning or wear and tear while the robot is operating.

DS2. **Impulsive error**. The error shall have a magnitude of 5-10%. This arises when the robot bumps into items or have wear and tear in certain isolated part of the system.

2.2 Overview of Iterative Learning Control (ILC)

Iterative learning control uses information-rich errors from previous iterations and learns from them. ILC has been successfully implemented in numerous other applications with some notable advantages such as the ability to control motion with high precision [23], remove disturbances asymptotically [24], and high robustness where it is capable of converging even in situations where there are uncertainties in system parameters estimates [25].

Therefore, considering the success of ILC in other applications and exploiting the repetitiveness of grasping, there is a strong motivation to employ ILC as grasping controller to meet the design requirements set out in Section 2.1.

ILC can be split into model-based and non-model based, and the following 2 subsections will investigate them further.

2.2.1 Non-Model based ILC

This type of ILC does not require knowledge of system dynamics to operate.

2.2.1.1 D-Type

The earliest ILC controller was proposed by Arimoto [26] where it uses errors generated by the previous iteration to modify the next iteration's input vector. Vectors are used to allow the algorithm to perform batch process between each iteration. The input vector update equation was defined as

$$\boldsymbol{u}_{k+1}(t) = \boldsymbol{u}_k(t) + \beta(t)\dot{\boldsymbol{e}}_k(t) \tag{2.1}$$

and the error

$$\boldsymbol{e}_k(t) = \boldsymbol{r}(t) - \boldsymbol{y}_k(t) \tag{2.2}$$

where $\boldsymbol{u}_{k+1}(t)$ is the next iteration input vector, $\boldsymbol{u}_k(t)$ is the current iteration input vector, $\beta(t)$ is the learning gain matrix, $\dot{\boldsymbol{e}}_k(t)$ is the derivative of current iteration error $\boldsymbol{e}_k(t)$, $\boldsymbol{r}(t)$ is the reference trajectory, $\boldsymbol{y}_k(t)$ is the actual system trajectory and k is the iteration number.

The system will converge to zero error when the iteration number goes infinite, and the convergence criteria are met. Otherwise, the system might experience divergence where the error increases.

2.2.1.2 P-Type

Some disadvantages of performing derivation of current iteration error include the chance of amplifying small errors and requiring more processing power. Therefore Arimoto [27] proposed a P-type ILC defined as follows

$$\boldsymbol{u}_{k+1}(t) = \boldsymbol{u}_k(t) + \beta(t)\boldsymbol{e}_k(t) \tag{2.3}$$

Since non-model-based algorithm converges slower than its model-based counterpart as they do not directly counter the system dynamics, other variations of non-model based ILC controllers such as PD, D-alpha, Higher-order, etc would be omitted here.

2.2.2 Model-Based ILC

Model-based ILC requires the knowledge of system dynamics (plant).

Plant models can be obtained through learning using data from multiple trials. Theoretically, if the abstracted model is identical to the actual plant, the system would converge in 1 iteration. However, it is impossible and unpractical to model an identical plant due to uncertainties, therefore studies have shown if the plant is not accurate, the system will take more iterations to converge. As more iterations are conducted, the difference between the expected output and actual output can be used to tune the model to increase its accuracy [28].

2.2.2.1 Inverse Plant ILC

Inverse plant ILC directly counters the plant dynamics with its inverse.

The output of the system is

$$\boldsymbol{y_k}(t) = G^0(\boldsymbol{u_k}(t)) \tag{2.4}$$

where G^0 denotes the true plant. The error equation remains unchanged as denoted in (2.2). As the plant is known, the optimal input sequence can be obtained as follows

$$\boldsymbol{u_k}(t) = -G^0(\boldsymbol{r}(t))^{-1}$$

where $G^0(.)^{-1}$ denotes the inverse of G^0.

The disadvantage of inverse plant method is any inaccuracy in the model or environmental disturbance can cause significant problems to the system. However, this method is acceptable even when there are uncertainties if the model updates the difference after each iteration [29].

2.2.2.2 Norm-Optimal ILC

Norm-optimal method works by optimising a cost function which can be described as

$$J\big(\boldsymbol{u}^*(k)\big) = \min_u J(\boldsymbol{u}(k)) \tag{2.5}$$

By using gradient descent, the optimal input that minimises the cost function can be obtained [30].

$$\nabla_u J = \left[\frac{\partial J}{\partial \boldsymbol{u}_i}\right] = 0 \tag{2.6}$$

To further improve the performance, the learning gain should change proportionally with error size. Thus, the gradient descent step size should be chosen automatically [31].

One minimisation problem identified by Amann et al. [32] can reduce each iteration normalised tracking error, automatically choose step size for cost function, and improve robustness is described as follows

$$\boldsymbol{u}_{k+1} = \arg\min_{u_{k+1}}\{J_{k+1}(\boldsymbol{u}_{k+1})\} \tag{2.7}$$

where the cost function and error are defined as

$$J_{k+1}(\boldsymbol{u}_{k+1}) = \|\boldsymbol{e}_{k+1}\|_{\mathcal{Y}}^2 + \|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|_{\mathcal{U}}^2 \tag{2.8}$$

$$\boldsymbol{e}_{k+1} = \boldsymbol{r} - \boldsymbol{y}_{k+1} \tag{2.9}$$

$$\boldsymbol{y}_{k+1} = G\boldsymbol{u}_{k+1} \tag{2.10}$$

where spaces $\mathcal{U}$ and $\mathcal{Y}$ are ℓ_2-spaces.

Detailed calculation has been omitted but can be found in the paper published by Amann et al. [32]. The final input sequence after solving the minimisation problem in (2.1) is as follows

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + (R + QG^TG)^{-1}QG^T\boldsymbol{e}_k \tag{2.11}$$

Where R and Q are diagonal matrices. R minimises the change in input sequence between iterations while Q minimises the tracking error. Careful tuning of the parameters ensures that the tracking error minimises quickly while ensuring excessive control effort is not injected to ensure smooth operation.

2.3 Overview of Impedance Control

To quantify the improvements ILC over conventional controllers, a baseline controller should be developed and subject to the same test. After reviewing Table 1.1, impedance control is chosen as a baseline comparison controller due to its simple and easy implementation while giving promising results. This chapter will briefly introduce impedance control algorithm adopted in modern grasping control.

Impedance control can be regarded as mechanical impedance and is normally used to describe complex systems using a mass-spring-damper system shown in Figure 2.1. Causality is an important concept in impedance control. Typically, the robot would act as impedance where the environment would behave as admittance since the environment is often composed of inertia where it accepts effort and produces flow.

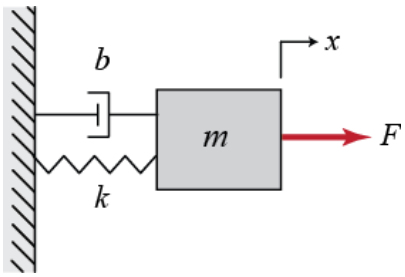


Figure 2.1: Mass-spring-damper structure (Adapted from [44])

This system can be described by the following equation

$$F = Ma + Cv + Kx \tag{2.12}$$

where M denotes the inertia, B denotes viscous damper, K denotes stiffness, a denotes acceleration, v denotes velocity and x denotes displacement.

Realising acceleration is double differentiation of displacement and velocity is the differentiation of displacement, using the concept of Laplace transform (2.12) can be written as

$$F(s) = (Ms^2 + Cs + K)X(s) \tag{2.13}$$

where it can be rearranged to form the following transfer function

$$\frac{X}{F}(s) = \frac{1}{Ms^2+Cs+K} \tag{2.14}$$

The two methods which are commonly used to implement the transfer function are integration and discretisation methods [33]. In this project, integration method would only be considered as it is more widely used in grasping operations [34][35]. Integration method was proposed by Caccavale et al. [36] for robots in the configuration presented in Figure 2.2.

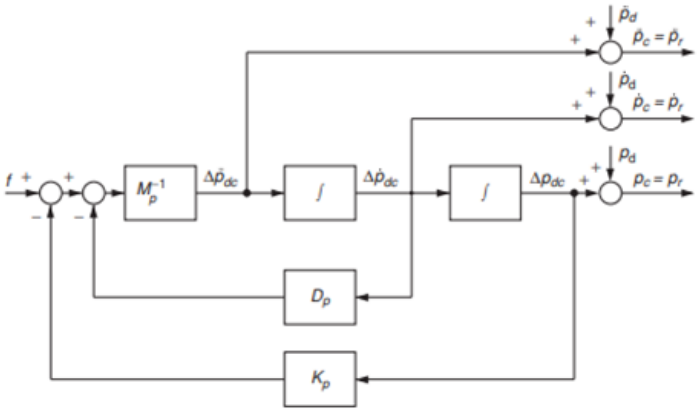


Figure 2.2: Block diagram for Impedance control using Integration method (Adapted from [36])

The operator would be tasked to choose 3 values, namely inertia (M_p), damping (D_p) and stiffness (K_p). Intuitively there is a relationship between grasping force and gripper end factor location, the algorithm will calculate the difference between the desire and actual force and adjust the gripper grasping displacement, velocity, and acceleration accordingly. This method has been tested to perform exceptionally well under different environments (soft and stiff) [33].

2.4 Summary

This chapter has identified various Design Requirements that the project would aim to achieve to improve the performance of robotic grasping. Various configuration of ILC and Impedance control algorithm has also been briefly discussed.

3 Design Overview

This section will provide a high-level overview and general approach for four crucial parts of the project:

- **Simulation environment,** including simulation software and robot selection, and modelling overview.
- Design of **robot (excluding gripper) controller** for moving the gripper to and from the desired location.
- Design of **gripper controller** including a baseline impedance controller and ILC controller that would fulfil the Design Requirements discussed in Section 2.1.
- Test and evaluate controller performance under different **disturbances** discussed in Section 2.1.

As the project is complex, each of the four parts of the project will be split further into multiple bite-size subparts to ensure each feature is thoroughly tested, verified, and analysed. Project management techniques and plans can be found in Appendix A.

3.1 Simulation Environment

DR1 requires the simulation robot model to be representative of current industrial robot. There are two commonly used configurations: gantry robot and robotic arm. Gantry robot consists of 3 independent axis actuator that moves a gripper along the X, Y and Z axis and has a cuboid work volume while robotic arm combines multiple joints to move the gripper and has a sphere work volume. Gantry robot is the simplest configuration and Ratcliffe [37] has identified the system dynamics of one such system shown in Figure 3.1. Therefore, this is an ideal robot to test the gripper controller algorithm before preceding to more complicated robotic arm system.

Figure 3.1: Proposed gantry robot by Ratcliffe (Adapted from [37])

According to Section 1.2, parallel plate grippers are the simplest and most universal. Thus, to keep the complexity down, parallel plate gripper is chosen.

There is numerous simulation software out in the market. Some are open-source, and others required licensing. Focusing on university subscribed and open-source software; the three most promising software for robotics has been identified: Gazebo, Webots, and MATLAB.

Gazebo is open-source and is considered to be one of the most widely used robotics software, it allows the designer complete control over simulation since it is integrated into ROS framework and is programmed in C++. However, Gazebo lacks good documentation, active community and is challenging to learn.

Webots is a good alternative to Gazebo as it has a simple and easy to use GUI, huge model database, active community, and good documentation. Webots is mainly programmed using C++ but there is API for other programming languages such as Java, Python and even MATLAB [38].

Although MATLAB is proprietary, the university has subscriptions for education use. MATLAB is commonly best known for data analysis as well as its ginormous collection of toolboxes including one of the most widely used control system design toolboxes. Simulink is built on top of MATLAB and is an easy-to-use graphical-based simulation environment utilising a 'drag and drop' design. Simscape multibody is an add-on of Simulink which allows the user to add graphical model to the simulation. However, the learning curve can be steep despite good documentation as well as the high CPU demand.

Table 3.1 shows a summary of the discussed simulation software. All three software are capable of simulating interaction force and contains 3D visual models which fulfil DR2 and DR3. Keeping in mind the aim of the project is to design and evaluate the gripper algorithm, it is decided that the good numerical computation and control system design toolbox that MATLAB offers is the best choice to design, test and analyse the controller performance.

Table 3.1: Features comparison between robotic simulation software

Software	License	Documentation	Programming language (main)	Robotic models (built-in)	Control toolbox
Gazebo	Apache 2.0	Fair	C++	Fair	N/A
Webots	Apache 2.0	Decent	C++	Decent	N/A
MATLAB	University subscriptions	Good	MATLAB	Limited	Good

Steps to model gantry robot and contact forces in Simulink are shown in Figure 3.2.

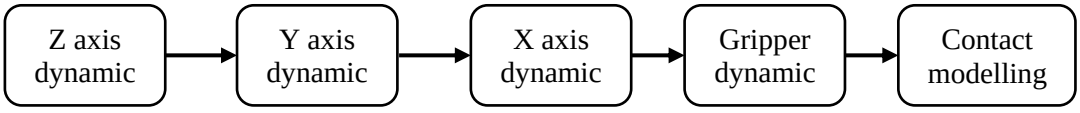


Figure 3.2: Steps for modelling gantry and environment dynamics in Simulink

3.2 Gantry Controller

After reviewing Ratcliff's [37] thesis, it is concluded that Norm-Optimal ILC performs the best. Therefore, this project would use Norm-Optimal ILC to control the X, Y and Z axis of the gantry. Equation (2.2) shows current iteration error and (2.11) shows the next iteration input vector update.

As this project focuses on the motion controller, the ideal trajectory would be given by the operator and is predefined. Since each axis of the gantry robot is independent of the other, they are decoupled, and three separate controllers will be used. The R and Q values for each controller will be tuned to meet DR4 and DR5. The schematic for the controllers is shown in Figure 3.3.

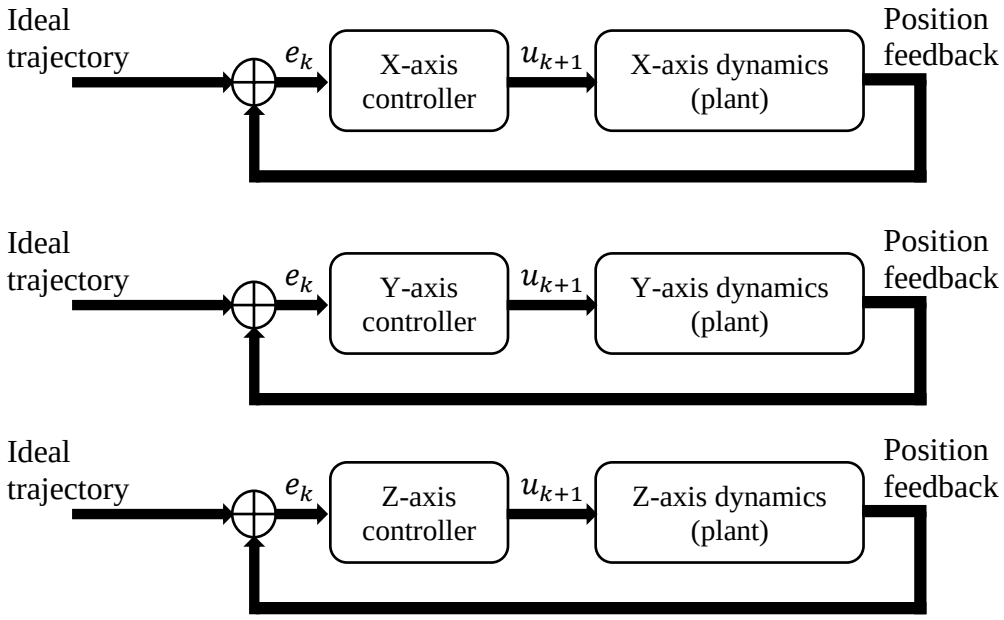


Figure 3.3: High-level schematic for X, Y and Z axis controllers

3.3 Gripper Controller

The following subsection will give an overview of the design schematic of Impedance and ILC controllers.

3.3.1 Impedance Control

Impedance controller focuses on minimising the force error between ideal and actual force by manipulating its grasping area, therefore a position controller is needed to reduce position error [34]. Since ILC updates the input vector at the end of each iteration, it is not suitable to be used as impedance controller to modify grasping distance at each sampling time. A conventional PD controller is used instead, and the schematic is shown in Figure 3.4.

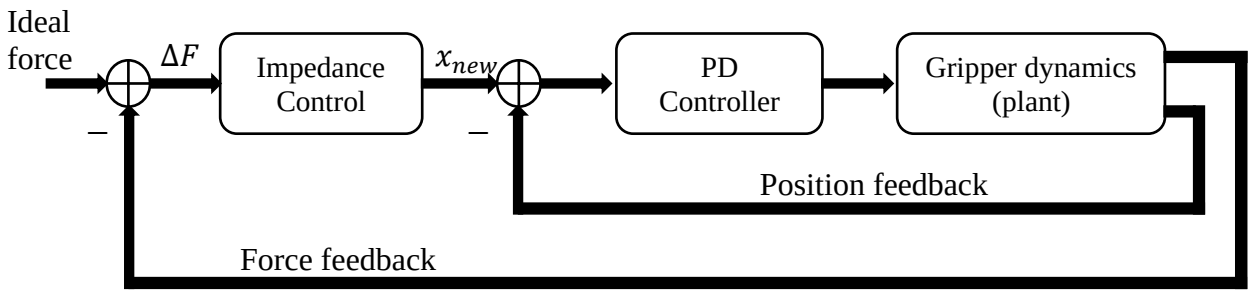


Figure 3.4: High-level schematic for impedance gripper controller

Where ΔF is force error and x_{new} is the new trajectory for the gripper to realise minimum grasping force. Upon digitising (2.12), the equation becomes

$$\Delta F(k) = M\Delta\ddot{x}(k) + B\Delta\dot{x}(k) + K\Delta x(k) \tag{3.1}$$

The equation can be rearranged to make $\Delta\ddot{x}(k)$ the subject as shown below

$$\Delta\ddot{x}(k) = M^{-1}(\Delta F(k) - K\Delta x(k) - B\Delta\dot{x}(k)) \tag{3.2}$$

Equation (3.2) can be represented in block diagram as shown in Figure 3.5.

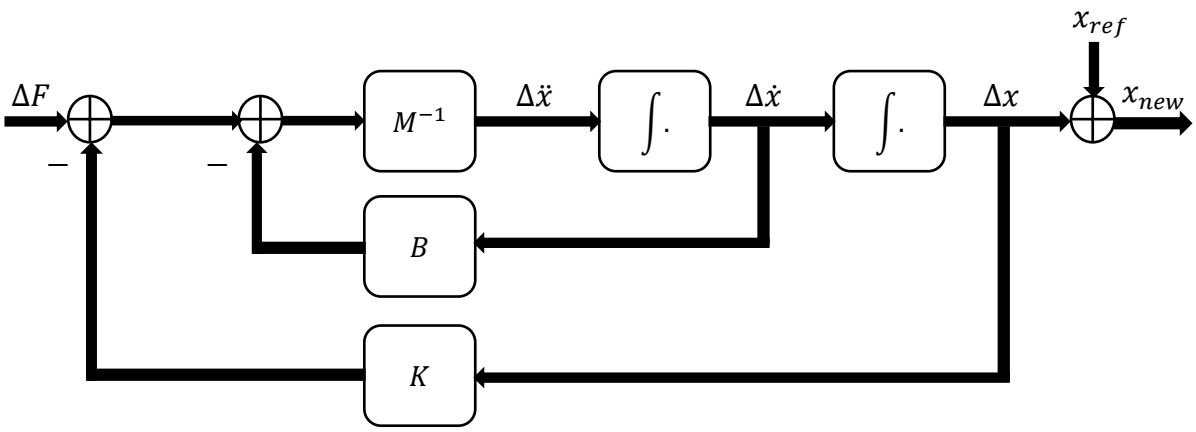


Figure 3.5: Impedance controller schematic

3.3.2 Iterative Learning Control

To realise DR4 to DR8 and evaluate the advantages and disadvantageous of each ILC scheme in Section 2.2, norm-optimal ILC is decided to be the best. However, since norm-optimal ILC requires minimising a cost function and Amann et al. [32] only minimises the tracking error, there is no single update equation that minimises both tracking and force error simultaneously. It is too complicated and time consuming for this project to evaluate an equation that would produce a single update. Thus, it is decided that the position tracking of the gripper would adopt norm-optimal approach described in (2.11), while force control will use a simple P-type update shown below

$$\boldsymbol{r}_{k+1}(t) = \boldsymbol{r}_k(t) + \beta(t)\Delta\boldsymbol{F}_k(t) \tag{3.3}$$

where r_{k+1} denotes the new reference trajectory, r_k denotes the last iteration reference trajectory, scalar gain β and $\Delta\boldsymbol{F}_k$ denotes the force error.

The schematic for ILC is shown in Figure 3.6.

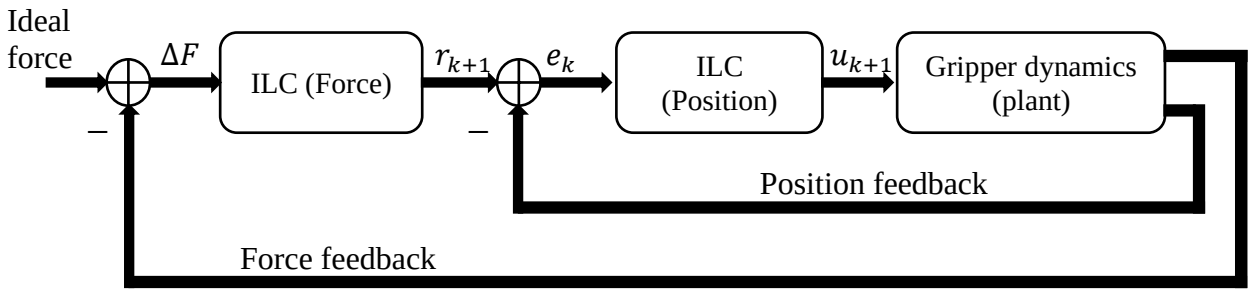


Figure 3.6: High-level schematic for ILC gripper controller

3.4 Disturbance

Random disturbance/errors are unavoidable, and it is crucial the robot can adapt itself without any human intervention. Thus, the two most commonly encountered disturbances a robot would encounter are discussed in Section 2.1.

As these two disturbances occurred in the environment where the robot situates in, one could model the disturbance as 'noise' added to the signal from the controller to the plant using the following equation.

$$\tilde{u} = u + d \tag{3.4}$$

Where d is the disturbance, u is the uncorrupted signal, and $\tilde{u}$ is the corrupted signal.

As such, the disturbance test schematic shown in Figure 3.7 is used to test the performance of each controller when the X, Y, Z axis and gripper encounter any disturbances.

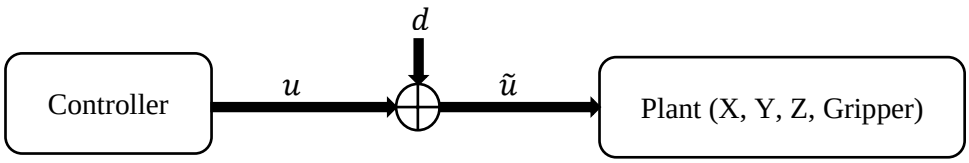


Figure 3.7: High-level schematic of controller output corrupted by disturbance

3.5 Summary

The decision on simulation environment and robot model has been decided to simulate real-world systems and contact as close as possible. Numerous high-level schematics have also been planned which would aid the process of designing control algorithms for the gantry and gripper. The following three chapters will show the process of transforming these high-level schematics into actual system.

4 Simulation Model

This chapter will translate the simulation model design discussed in Section 3.1 into MATLAB using Simulink and Simscape Multibody.

4.1 Gantry Modelling

Each axis of the gantry robot has a few important features that need to be modelled,

1. It has mass and interacts with the environment; hence it has its unique dynamics.
2. It is capable of moving on only one axis.

The goal is to model the dynamics of each axis as identified by Ratcliff [37]. The schematic to achieve this is shown in Figure 4.1.

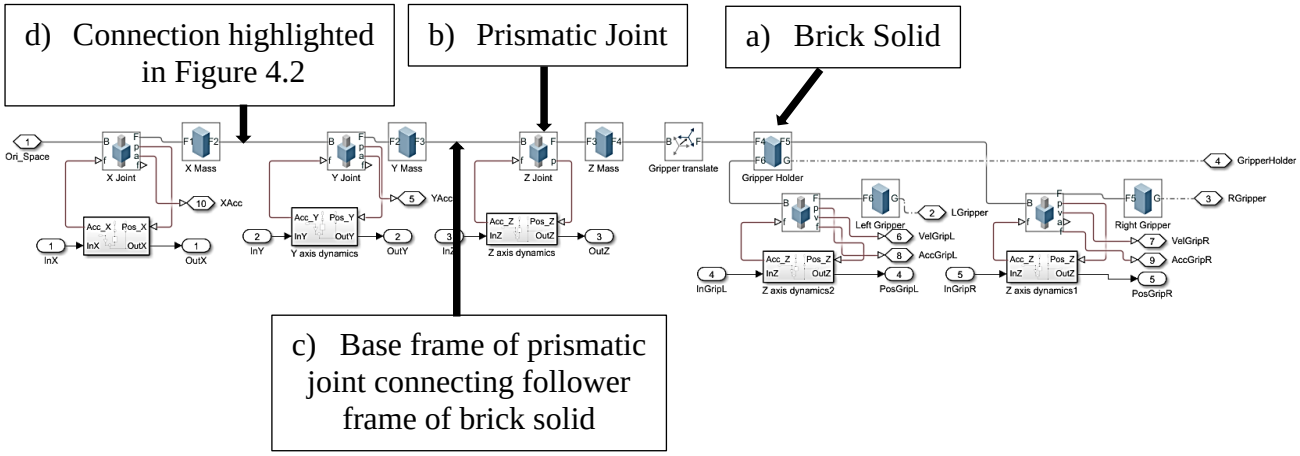


Figure 4.1: Schematic of the proposed model in Simulink

The visual of each axis can be represented by 'Brick Solid' (Figure 4.1a), where the dimension and density of each axis brick solid are as shown in Table 4.1.

Table 4.1: Values of Simulink model for each part of the gantry robot

Part	Dimension (mm)	Work area (mm)	Mass (kg)
X-axis	520 × 25 × 25	640	4.32
Y-axis	25 × 25 × 25	520	3.51
Z-axis	25 × 25 × 100	100	0.169
Gripper	70 × 50 × 35	50	0.081

Since the X-axis actuator is responsible for moving Y and Z components, and gripper along the X direction, where the Z and gripper components are situated on top of the Y structure; it can be imagined as a 'moving' Y structure along X direction. Hence the X-axis brick solid which moves in X direction will have the same dimension as Y structure to fulfil the visual aspect of the model. The same analogy can be applied to the rest of the brick solid. This can be seen in Figure 4.14.

'Prismatic Joint' (Figure 4.1b) is only able to move 'brick solid' in the Z direction. It accepts 'force' as input to move the object and can output a range of values including position, velocity, acceleration, and actuator force.

Frames are a very important concept when modelling using Simscape Multibody. There are 2 main crucial frames are base and follower frame. The frames have some important properties:

1. Base frame of the subsequent prismatic joint will follow the orientation of the follower frame of the previous brick solid (Figure 4.1c).
2. The orientation of X, Y and Z-axis of brick solid can be orientated so that the Z-axis of the follower frame will be positioned in the intended travel direction, the base frame will have the same orientation as the previous 'object' follower frame to ensure it 'stacks' on top nicely (Figure 4.2).

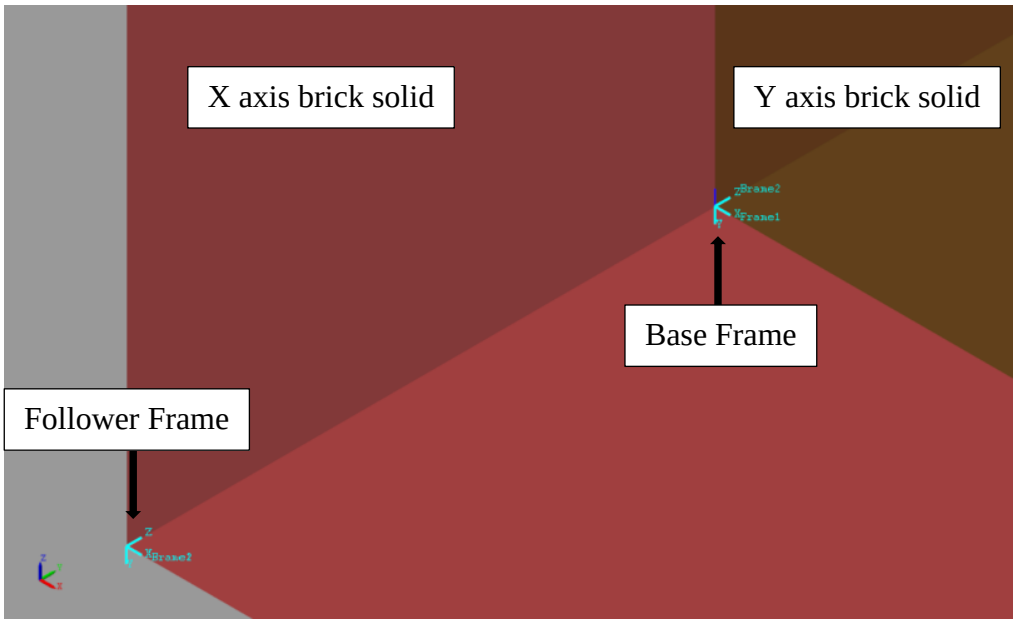


Figure 4.2: Visual representation of the base and follower frame connection (between X and Y axis) highlighted in Figure 4.1d

To ensure the model is as accurate as possible, it is crucial to model the dynamics experienced by each axis and gripper. A simple way of achieving this is to use a feedback loop PD controller to tune the dynamics of the system using the schematic shown in Figure 4.3.

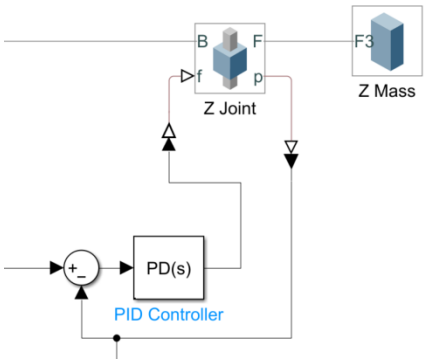


Figure 4.3: Proposed PD controller to model the dynamics of the system

Grouping prismatic joint and brick solid together and donating them as 'Plant, P' and PD controller as 'C', the transfer function of the proposed system is as follows

$$H(s) = \frac{PC}{1 + PC}$$

$$H(s) = \frac{P(k_p+k_dS)}{1+P(k_p+k_dS)} = \frac{k_p+k_dS}{P^{-1}+k_p+k_dS} \quad (4.1)$$

The following sub-sections will explain issues encountered by such system, the iterative improvement and finally a proposed final solution.

4.1.1 Trial 1

After applying step input to plant P (Figure 4.4), the Laplace transform of the system can be represented using (4.2).

$$P(s) = \frac{1}{ms^2} \quad (4.2)$$

where m is the mass.

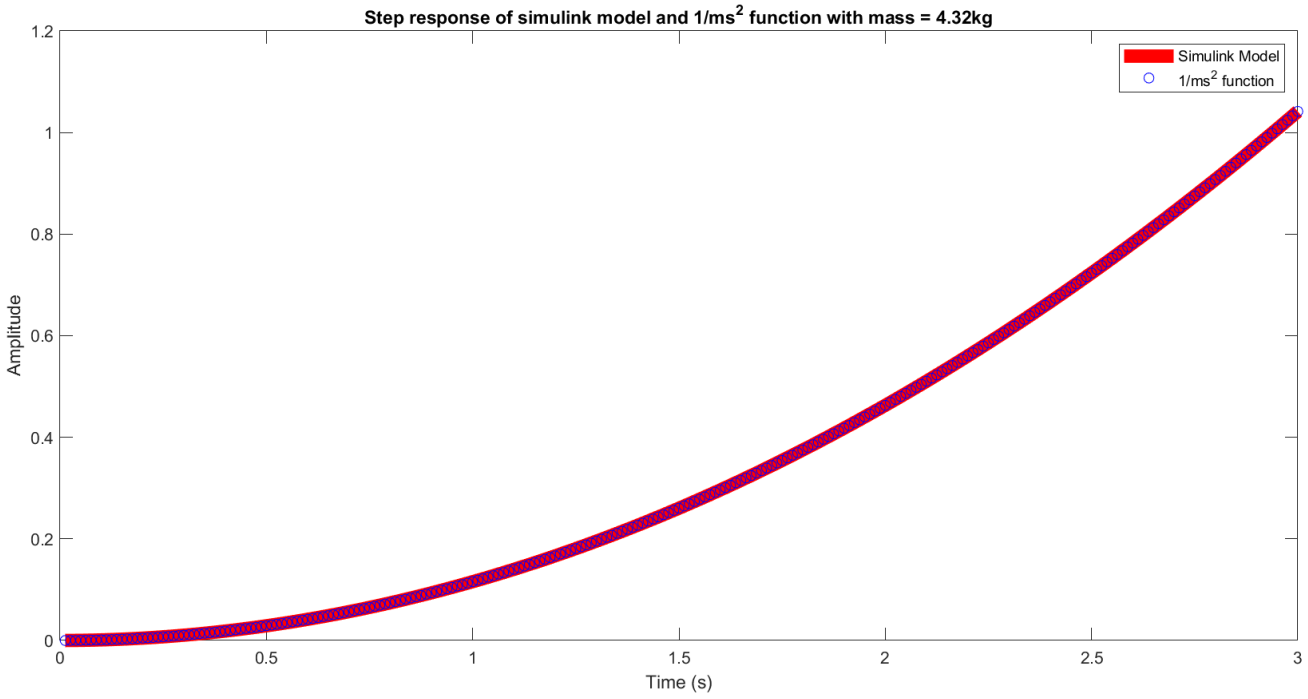


Figure 4.4: Step response for the prismatic and brick solid model VS the equivalent representation shown in $P(s) = \frac{1}{ms^2}$ (4.2).

As the system expands, the total mass acting on the first part of the system is the sum of its own and all the following mass of the rest of the connected system. For example, the total mass needed to be pushed by X-axis prismatic joint is the sum of X, Y, Z and Gripper brick solid equating to 8.098kg.

Using the schematic proposed in Figure 4.3. The PD controller is tuned using a MATLAB built-in function, PID Tuner. PID Tuner has a graphical GUI where the user can change the parameter values by typing or using a slider. The response of the resultant system can be seen instantly.

To model system dynamics, the impulse response of the actual system transfer function (Figure 4.5) has to match the step response of the model (Figure 4.6) by matching their peak response and transient time as closely as possible. The best values found are $k_p = 33634.3806$ and $k_d = 163.3421$.

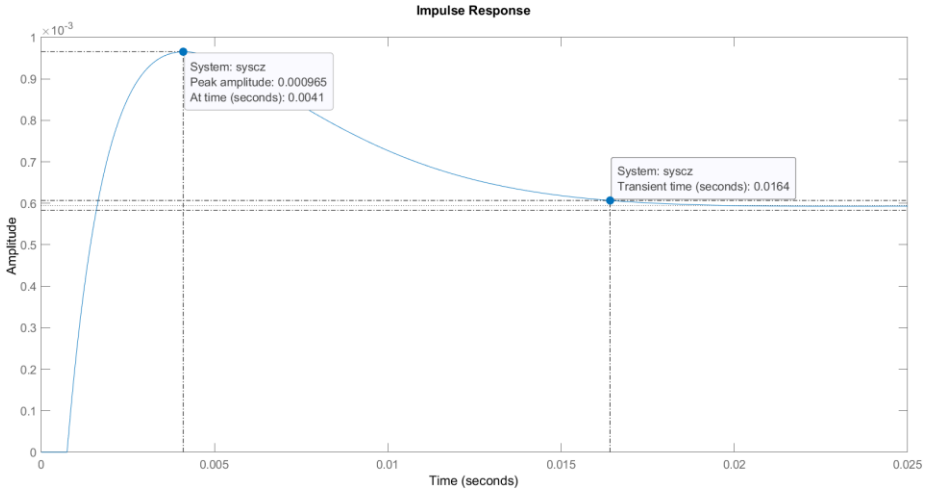


Figure 4.5: Impulse response of Z axis actual transfer function

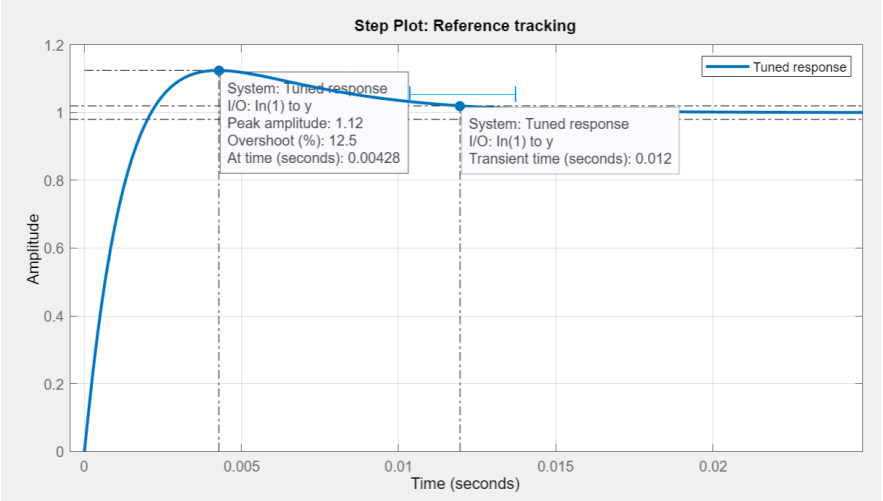


Figure 4.6: Step response of Simulink model using schematic proposed in Figure 4.3

From the two figures, the differences between impulse and step response are small (in terms of a few milliseconds) thus is acceptable since the grasping operation is in seconds.

The step response of the modelled system denoted by (4.1) and (4.2) can be denoted as follows.

$$F(s) = \frac{1}{s} \cdot H(s) = \frac{1}{s}\left(\frac{k_p + k_d s}{ms^2 + k_p + k_d s}\right) \tag{4.3}$$

Figure 4.7 places the two responses together and it can be seen that the steady-state amplitude is different despite having similar peak response and transient time. The schematic proposed in Figure 4.3 has an issue where the steady-state response always equates to one as shown by applying the Final Value Theorem to (4.3).

$$\lim_{s \to 0} sF(s) = \frac{k_p}{k_p} = 1 \tag{4.4}$$

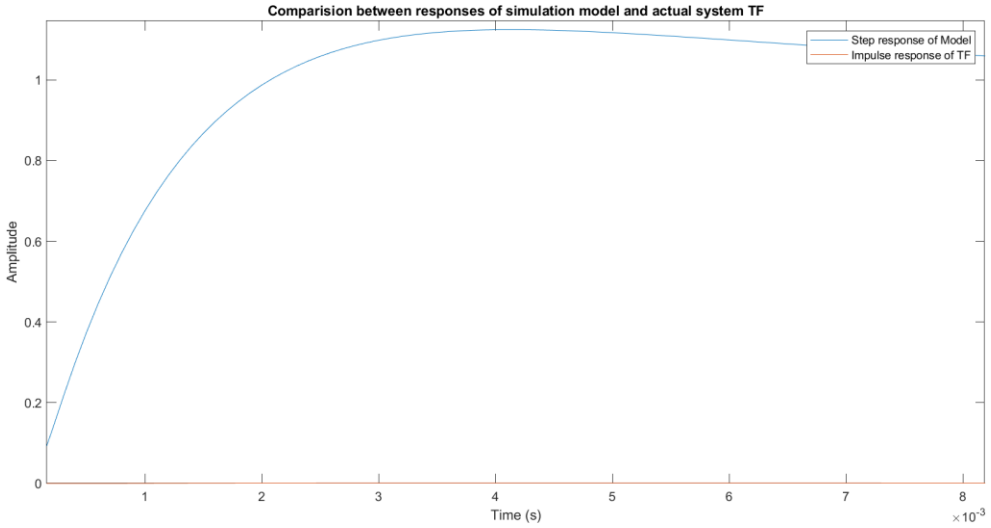


Figure 4.7: Comparison between step response of Simulink model and impulse response of actual system transfer function for Z-axis

4.1.2 Trial 2

There are two potential solutions to solve the steady-state issue, the first is to move the PD controller and place it in the feedback loop as shown in Figure 4.8. The other option is to place a scalar gain on the input signal sent to the PD controller to scale down the steady-state response accordingly as shown in Figure 4.9.

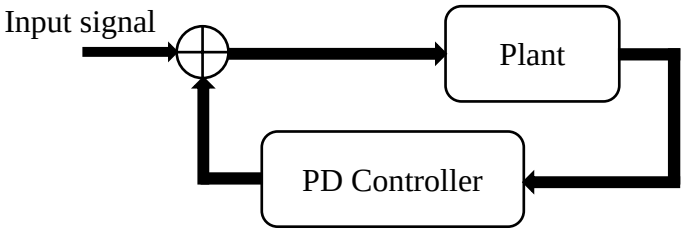


Figure 4.8: Proposed schematic where PD controller is located at the feedback loop with no scalar gain

The new transfer function of the system is as follows.

$$H(s) = \frac{P}{1 + PC}$$

$$H(s) = \frac{P}{1 + P(k_p + k_d S)} = \frac{1}{P^{-1} + k_p + k_d S} \tag{4.5}$$

Upon applying the final value theorem, the equation becomes

$$\lim_{s \to 0} sF(s) = \frac{1}{k_p} \tag{4.6}$$

where

$$F(s) = \frac{1}{s} \cdot H(s) = \frac{1}{s}\left(\frac{1}{ms^2 + k_p + k_d s}\right)$$

This shows that the steady-state value is determined by the proportional gain value chosen to tune the PD controller and therefore cannot be fully 'customise'. On top of that, the PID tuner assumes the PID controller is placed cascading with the plant, thus it is not possible to tune the dynamics directly using the PID tuner.

4.1.3 Trial 3

With the problem encountered in the previous subsection, the other feasible option is to add a scalar gain to the input signal. As the PID tuner does not work when the PD controller is placed in a feedback configuration, the following schematic is proposed.

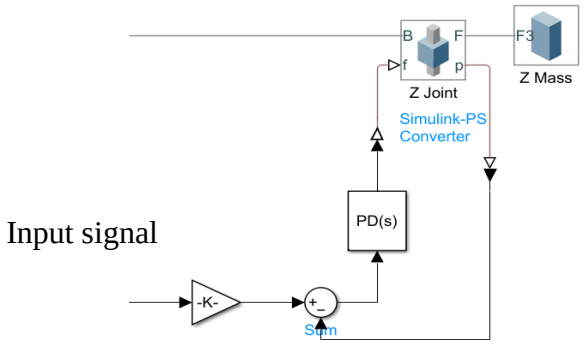


Figure 4.9: Proposed schematic for modelling gantry robot dynamics with PD controller and scalar gain

The transfer function of the system can be formulated as follows

$$H(s) = \frac{PC}{1 + PC}$$

$$H(s) = \alpha \frac{P(k_p + k_d s)}{1 + P(k_p + k_d s)} = \alpha \frac{(k_p + k_d s)}{P^{-1} + k_p + k_d s} = \alpha \frac{(k_p + k_d s)}{ms^2 + k_p + k_d s} \tag{4.7}$$

where α denotes scalar gain and is obtained visually using graph. According to Figure 4.10, the steady-state impulse response value for the Z-axis is 5.93×10^{-4}. Thus, to match this value to the step response steady-state value, α has to be 5.93×10^{-4} which is also $\frac{1}{1686.3406}$. This is also done to the rest of the system with values shown in Table 4.2.

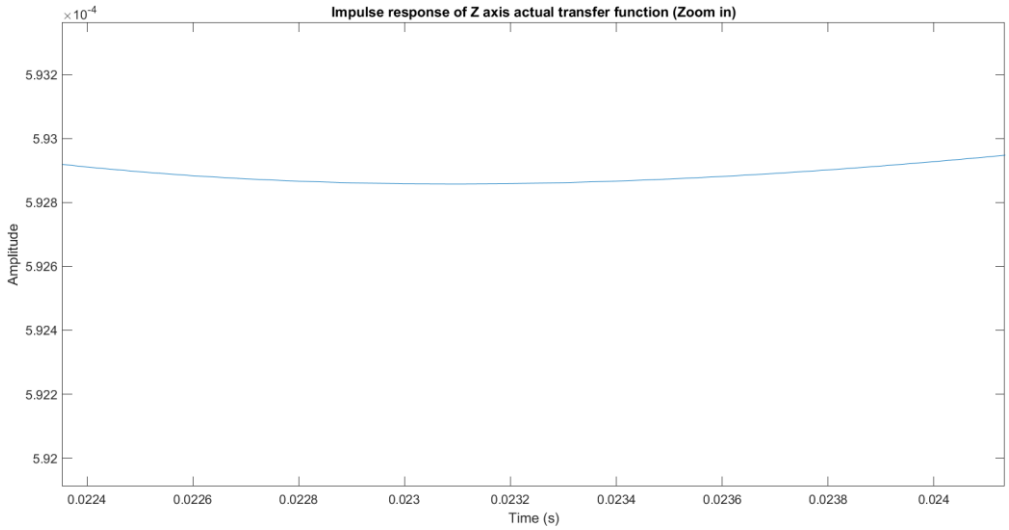


Figure 4.10: Zoom in the impulse response of Z-axis actual transfer function

Figure 4.11 shows the impulse response of the Z-axis transfer function and step response of the Z-axis model.

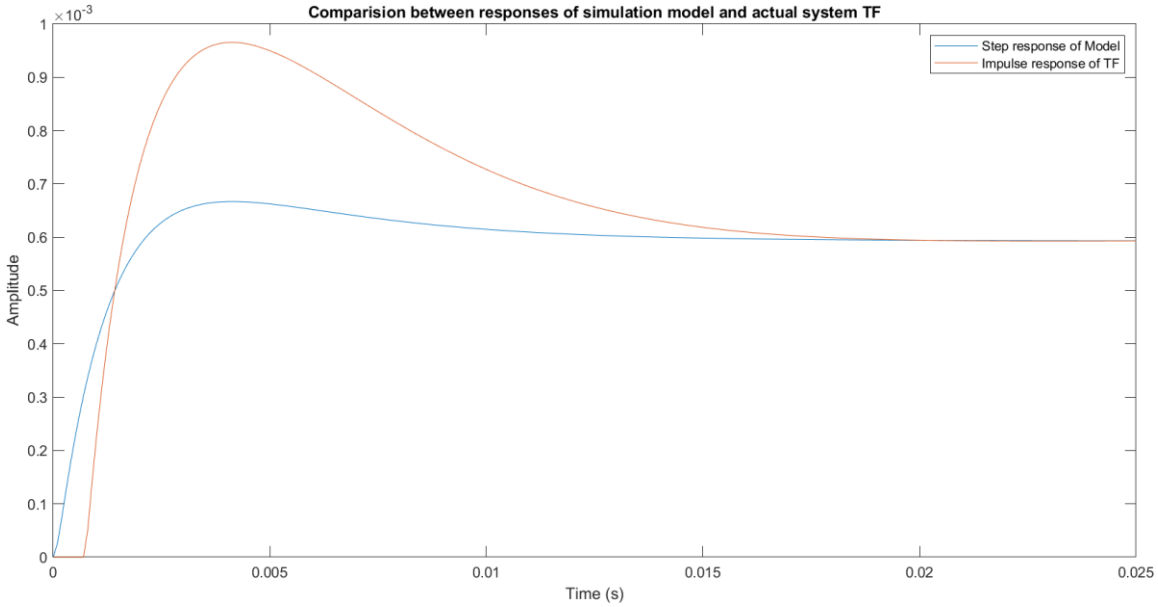


Figure 4.11: Comparison figure between step response of Simulink model with scalar gain and impulse response of actual Z-axis transfer function

Although the peak amplitude is different, the peak response and transient time are similar, and the difference is in milli scale, therefore it is deemed to be acceptable.

With this, the method to model system dynamics is completed with the model transfer function for each part of the system formulated using (4.7) by substituting the corresponding values shown in Table 4.2 and is now denoted as 'Apparent TF'. The below example is for Z-axis.

$$H(s) = \alpha \frac{(k_p + k_d s)}{ms^2 + k_p + k_d s} = \frac{1}{1686.3406} \frac{(33734.3806 + 163.3421s)}{0.2498s^2 + 33734.3806 + 163.3421s} \tag{4.8}$$

Table 4.2: Values for Simulink models of each part of the gantry robot and gripper

Part	α	k_p	k_d	Sum masses (kg)
X axis	1/12.5	27146.67	1020.03	8.04
Y axis	1/500	131139.23	1480.58	3.72
Z axis	1/1686.3	33734.38	163.3421	0.2493
Gripper (each finger)	1/1550.7	33734.38	163.3421	0.0405

The step response of the apparent transfer function is plotted along with the Simulink model step response to ensure the deduced apparent transfer function is accurate and is shown in Figure 4.12. There are only minimal differences which could be caused by numerical inaccuracy.

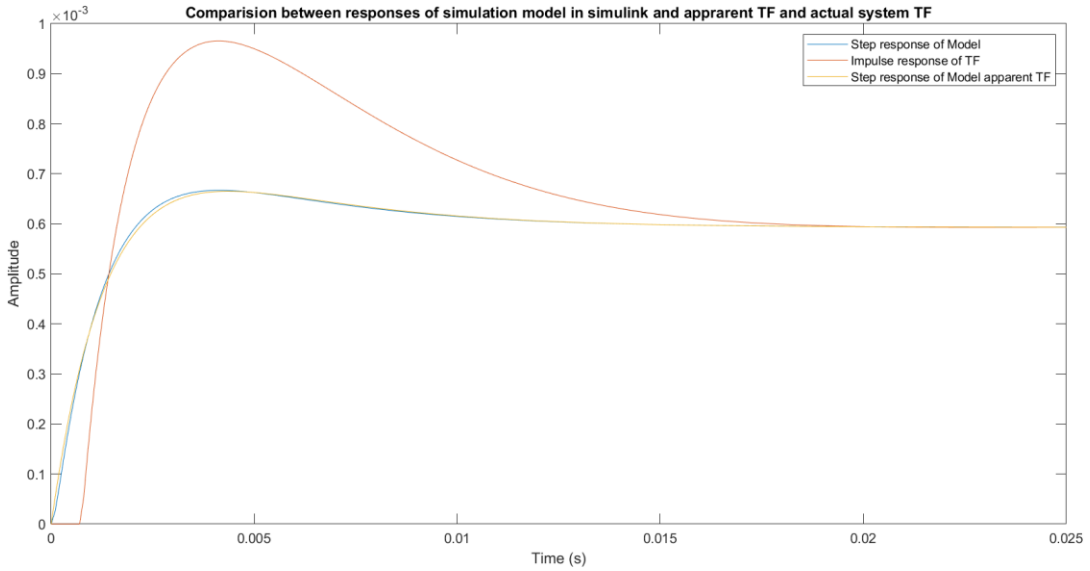


Figure 4.12: Comparison figure between step response of Simulink model with scalar gain and 'apparent transfer function' and impulse response of actual Z-axis transfer function

The final schematic of the system in Simulink is shown in Figure 4.13 where the axis dynamics block follows Figure 4.9 schematic. The visual of the whole system is shown in Figure 4.14.

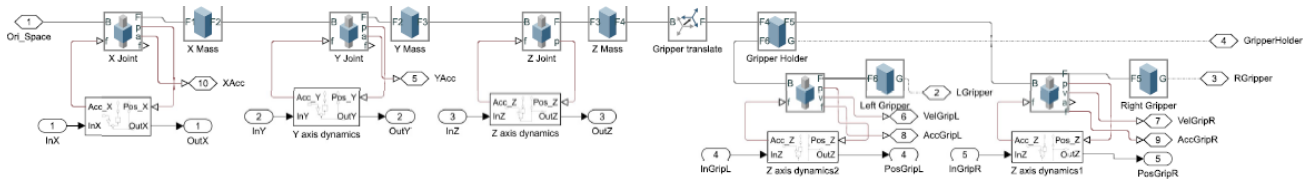


Figure 4.13: Overall schematic of gantry model

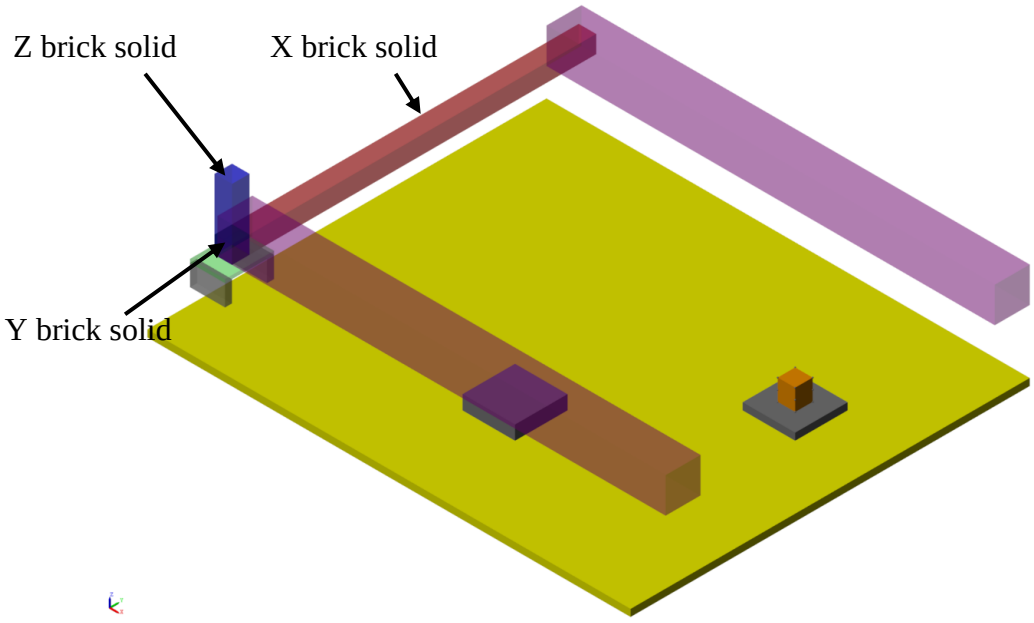


Figure 4.14: Visual view of gantry model

4.2 Contact Modelling

Simscape Multibody offers a range of blocks to model contact between two bodies. In this project, the blocks used are ‘Solid Sphere’ and ‘Spatial Contact Force’.

Spatial contact force uses a penalty force-based method to implement contact force. Penalty method allows the two-contacting object to overlap each other slightly to generate contact force as shown in Figure 4.15. The block is capable of giving several parameters as output including normal and frictional force.

The amount of normal force generated when the two bodies are in contact is modelled using a mass-spring-damper system and can be adjusted by changing the stiffness and damping value. In this project, the damping is set to 0 for easier calculation while stiffness is set to 1000N/m.

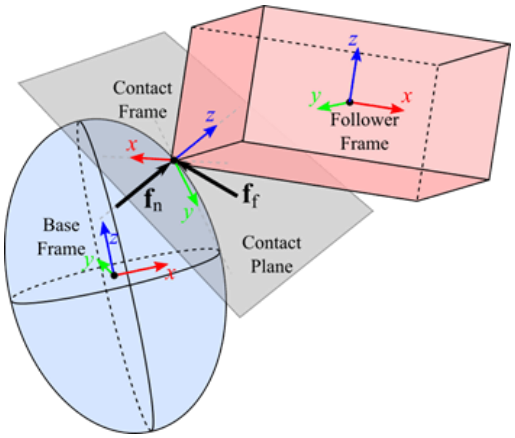


Figure 4.15: Simscape Multibody contact forces explanation diagram (Adapted from[45])

The frictional force is computed using the built-in smooth stick-slip method, where the parameters of interest are the coefficient of static and dynamic friction. Typically, the coefficient for dynamic friction is less than static, since the aim is to grasp the object, the force is ideally kept within the static region, hence it is reasonable to ignore the lower dynamic coefficient value and set both coefficients to 0.5.

To speed up modelling and improve performance, the gripper will be in contact with multiple strategically placed 1mm radius solid spheres instead of the whole object surface seen in Figure 4.17. This schematic is shown in Figure 4.16.

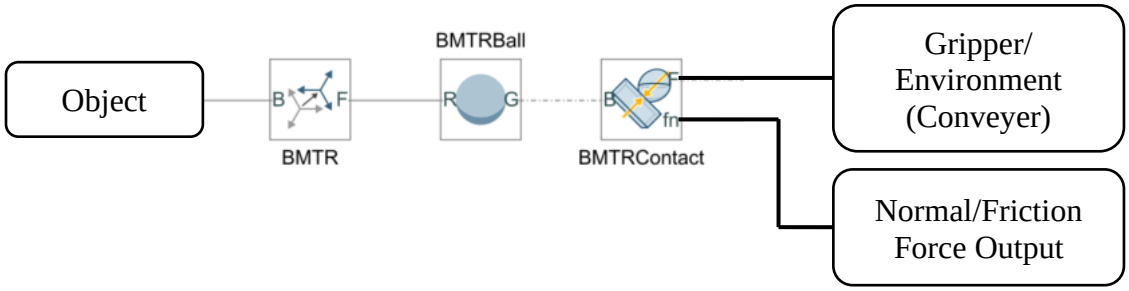


Figure 4.16: Schematic for modelling contact forces between two bodies

In total, the object will have twelve spheres spread around equally shown in Figure 4.17. This ensures that there will be at least 4 spheres in contact with either the gripper fingers or the conveyer where the object is situated to ensure that the object does not rotate/move in any direction while being grasped or placed on the conveyer.

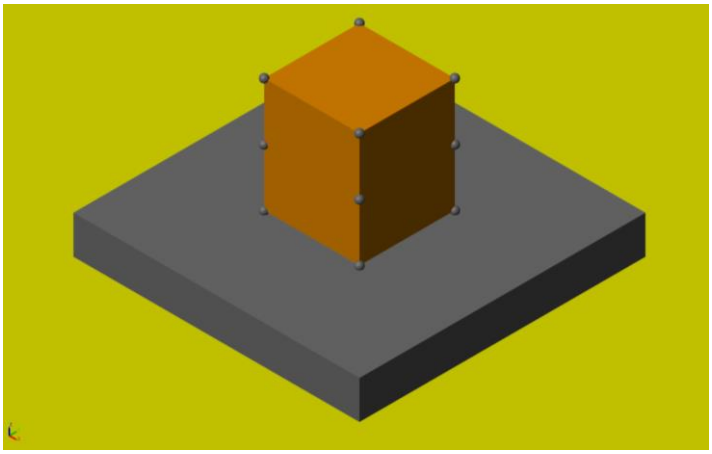


Figure 4.17: Diagram showing contact points (sphere) of the object to be grasped

4.2.1 Grasping Force Calculation

A free-body diagram of an object being grasped can be seen in Figure 4.18.

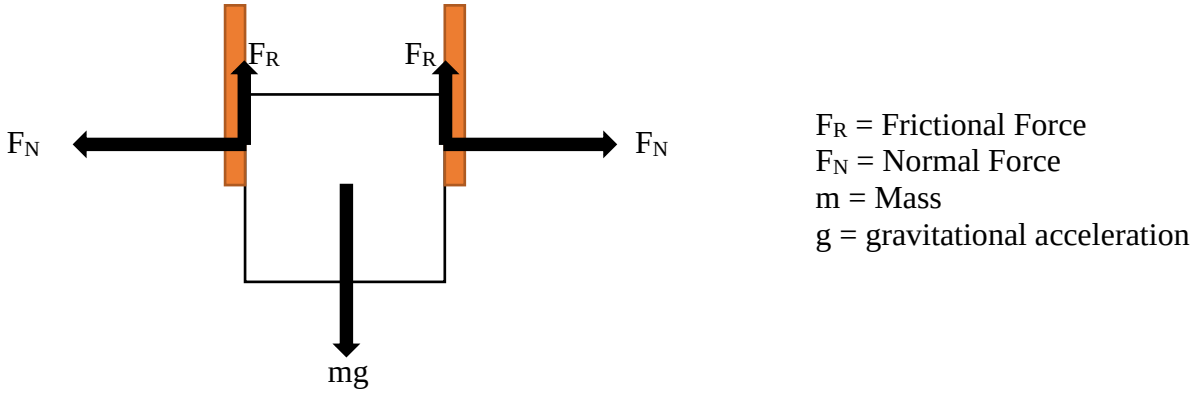


Figure 4.18: Free body diagram of grasping and lifting an object with two fingers parallel plate gripper

To ensure the object can be lifted,

$$2F_R \geq mg \tag{4.9}$$

while the relationship between frictional and normal force is

$$F_N = \mu F_R \tag{4.10}$$

where μ is the coefficient of friction. Since the magnitude of the grasping force for one finger (F_g) is the same as the normal force it experienced. The force that each gripper has to exert is calculated as follows

$$F_g = F_N = \frac{\mu mg}{2} \tag{4.11}$$

The theoretical minimum force to grasp and lift the object securely can be calculated. This force is then spread evenly over all the spheres that it comes in contact with. In this project, each finger is in contact with 4 spheres (Figure 4.19).

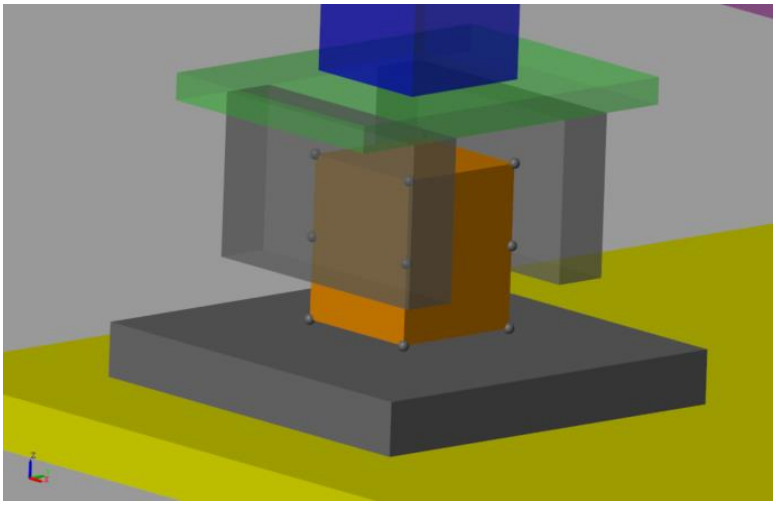


Figure 4.19: Diagram showing the contact between two bodies of interest (Object and Gripper, and Object and Conveyer)

For a mass of 2kg object, the frictional force of each gripper finger can be calculated using (4.9) to be

$$F_R = \frac{mg}{2} = \frac{(2)(9.81)}{2} = 9.81$$

Since the finger is in contact with 4 spheres, the frictional force will be evenly distributed among the 4 spheres giving the force experienced by each sphere to be

$$F_{Rsphere} = \frac{9.81}{4} = 2.45N$$

while the minimum normal force is calculated as follows using (4.11)

$$F_g = \frac{(0.5)(2)(9.81)}{2} = 4.905N$$

Figure 4.20 shows the simulation value for $F_{Rsphere}$. Since the gripper did not grasp with minimal grasping distance, the normal force is not the same as F_g.

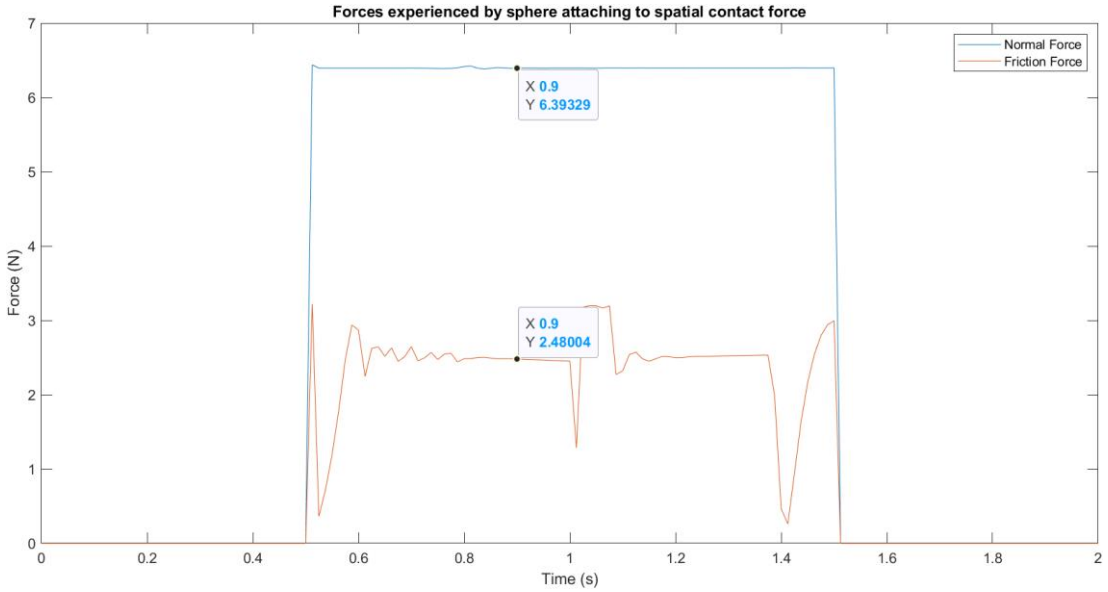


Figure 4.20: Graph showing the Normal and Frictional forces that the spatial contact force experience when grasping and lifting the object from the conveyer

4.2.2 Normal Force while Accelerating

The gripper fingers are parallel to the X-axis as seen in Figure 4.19. Therefore, the normal force experienced by each gripper changes due to the acceleration and deceleration when moved in the Y direction as illustrated in Figure 4.21.

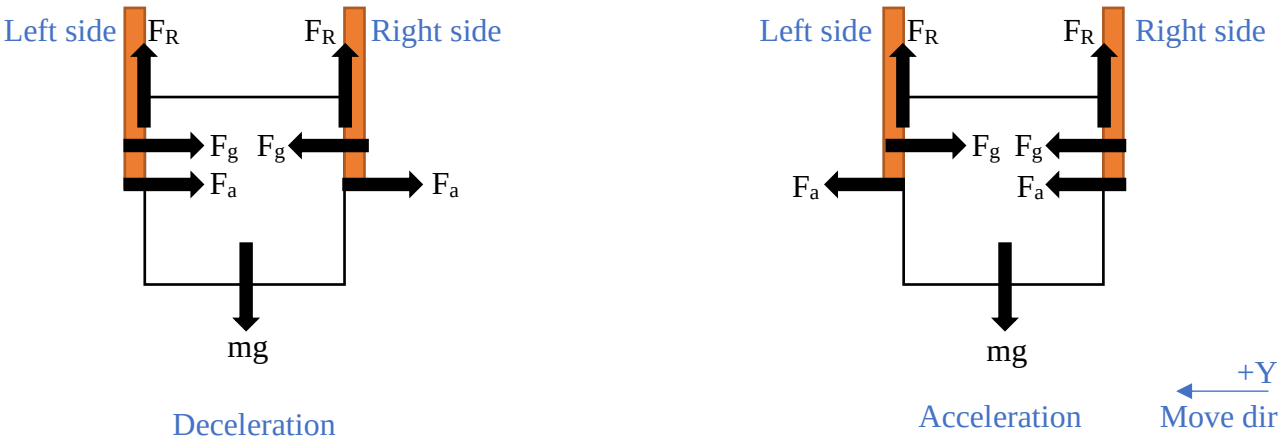


Figure 4.21: Free body diagram of forces acting on object and gripper due to acceleration and deceleration in the Y direction

In the deceleration phase, the net normal force experience by each finger is,

$$F_{NetRight} = F_g - F_a$$

$$F_{NetLeft} = F_g + F_a$$

while in the acceleration phase,

$$F_{NetRight} = F_g + F_a$$

$$F_{NetLeft} = F_g - F_a$$

These changes can be seen in Figure 4.22 when the gripper accelerates in the Y direction. The right gripper initially experiences more force due to acceleration and experiences less in the second half due to deceleration. The opposite is true for the left gripper.

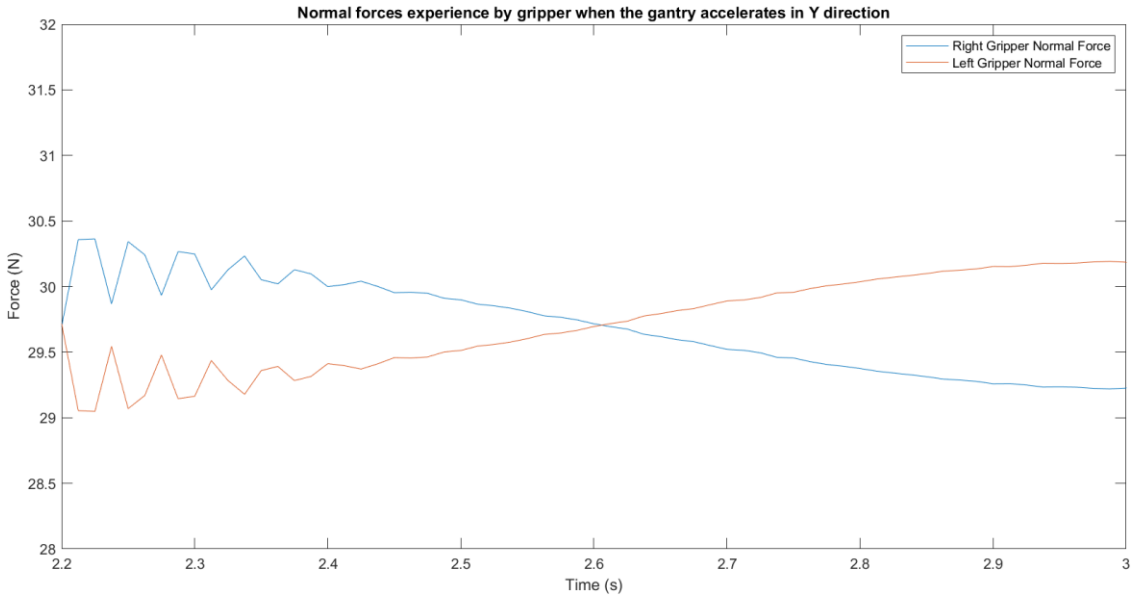


Figure 4.22: Graph showing the change in net force experienced by gripper finger due to acceleration and deceleration in the Y direction

4.3 Summary

This chapter has successfully modelled and tuned the proposed gantry system dynamics and visual aspect into MATLAB Simulink environment. Contact behaviour of the gantry system with the object and environment has also been modelled. The following chapter will design the control algorithm for moving the X, Y and Z-axis actuator.

5 XYZ Position Controller Design

This chapter will design the Norm-Optimal ILC controllers discussed in Section 3.2 for the 3-axis reach, grasp and return motion. A quick summary of requirements for the controller design is as follows.

1. Reach and grasp object within 15 iterations. Therefore, gantry controllers should converge under 10 iterations to ensure the grasping phase has sufficient iterations to learn and improve.
2. Reach and grasp operation should complete within 5 to 7 seconds.

As the controller is designed in discrete time, the duration of the whole reach and grasp trajectory is aimed to be completed in 4s with a sampling time of 0.01s giving 400 data points in one iteration.

5.1 Trajectory profile generation 1

As discussed in Section 3.2, each axis will be controlled by its independent controller. These controllers will have the algorithm shown in Figure 5.1.

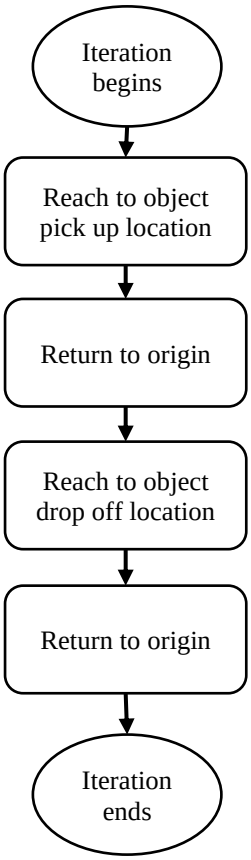
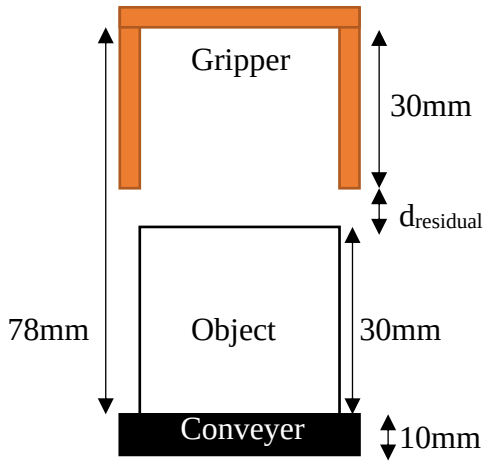


Figure 5.1: Basic control algorithm for 3 axes controllers

The object pick-up and drop-off location can be different as it is in the real system. In this project, the pick-up location is at $0.5m \times 0.3m$, while the drop-off location is at $0.3m \times 0.1m$.



Initial conveyer height = 10m
Destination conveyer height = 20mm.

$$d_{residual} = 78 - 30 - 30 = 18mm$$

To ensure object is firmly grasped, it should grasp at least half of the object.

$$\begin{aligned} d_{zi} &= d_{residual} + obj_{mid} + margin \\ &= 18mm + 15mm + 5mm \\ &= 38mm \end{aligned}$$

Destination conveyer is 10mm higher thus $d_{zf} = 28mm$.

Figure 5.2: Z-axis travel distance calculation

Figure 5.2 shows the configuration of gripper and object and the calculation for the Z-axis required travel distance.

With the target X, Y and Z values known, it is possible to generate a profile for the gantry to follow. To ensure smooth operations, the trajectory profile should have a gentle rise and fall. The rising profile is calculated using the following equation

$$y = \frac{-A\cos\left(\frac{1}{d}\pi x\right)+A}{2} \tag{5.1}$$

where the falling profile is

$$y = \frac{A\cos\left(\frac{1}{d}\pi x\right)+A}{2} \tag{5.2}$$

where A denotes steady-state amplitude, d denotes rise or fall period, x denotes sampling instance.

To ensure the trajectory can be completed in the shortest time, the deadtime for each axis should be as short as possible. This can be achieved by moving multiple axes. However, one of the top priorities is to transport the object safely, therefore Z-axis should only be moved when the X and Y-axis are within the grasping region to avoid knocking the object over prematurely. Unfortunately, that means Z-axis would not be moving together with X and Y-axis thus introducing additional deadtime for X and Y-axis while they wait for Z-axis to complete its operation.

The profiles for the 3-axis are shown in Figure 5.3.

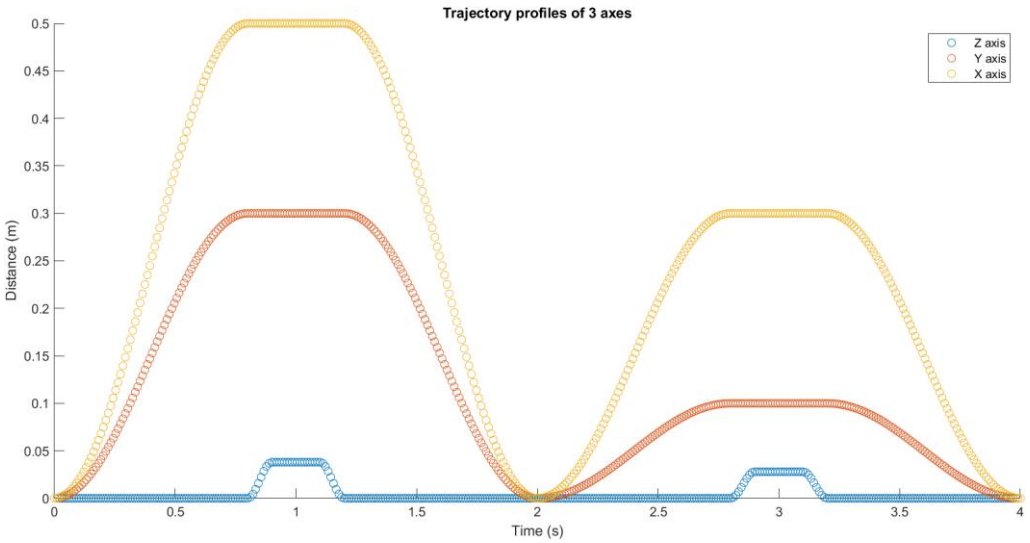


Figure 5.3: Trajectory profile for all 3 axes

Table 5.1 shows the respective values for each axis profile where A denotes amplitude, d denotes rise and fall time, and D denotes deadtime.

Table 5.1: Parameters to generate trajectory profile for each axis controller

Parameters	X-axis	Y-axis	Z-axis
A1 (m)	0.5	0.3	0.038
A2 (m)	0.3	0.1	0.028
d1 (s)	0.8	0.8	0.1
d2 (s)	0.8	0.8	0.1
d3 (s)	0.8	0.8	0.1
d4 (s)	0.8	0.8	0.1
D1 (s)	0.4	0.4	0.8
D2 (s)	0.4	0.4	0.2
D3 (s)	-	-	1.6
D4 (s)	-	-	0.2
D5 (s)	-	-	0.8

5.1.1 Tuning ILC Controller

Equations (2.9) and (2.11) used for the next iteration input update and error are repeated below

$$e_{k+1} = r - y_{k+1} \tag{5.3}$$

$$u_{k+1} = u_k + (R + QG^T G)^{-1} QG^T e_k \tag{5.4}$$

where the error vector would be the difference between the trajectory profile (Figure 5.3) and the actual distance moved by the system. The plant, G, would be the plant deduced in Section 4.1.3 for each axis.

There are two ways to tune R and Q values. The first is to use ‘lsim’ feature in MATLAB which simulates a time response of a dynamic system to arbitrary input. The second is to pass input from MATLAB to Simulink using ‘From workspace’ block to generate responses and pass the result to MATLAB using ‘To workspace’ block. The generated response for both is similar as seen in Figure 5.4, but lsim generates results instantly while Simulink takes 4 seconds to simulate each iteration. Therefore, lsim is used in the view of time-saving.

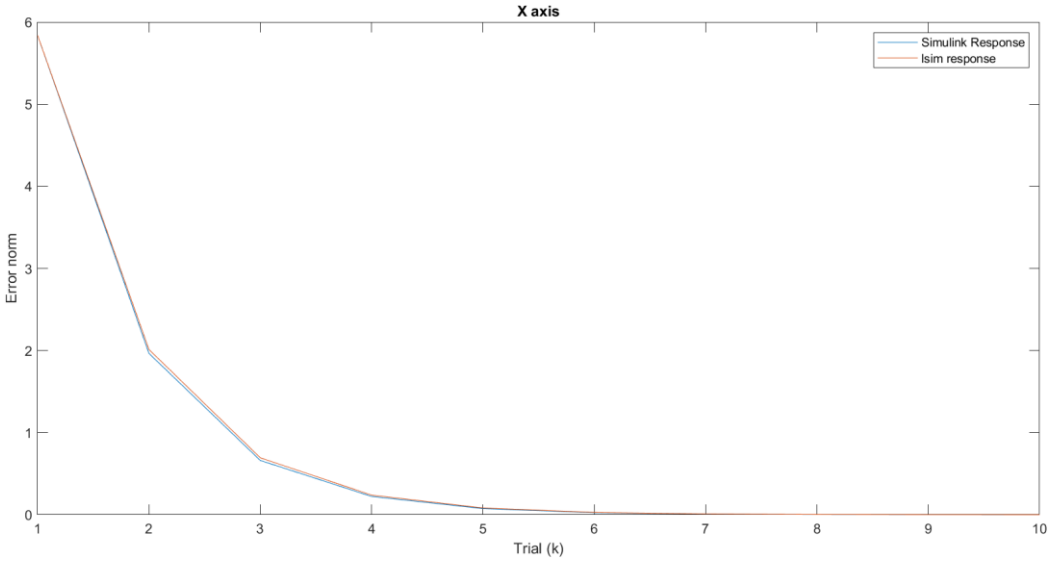


Figure 5.4: Comparison of simulated normalised position error for X-axis between using Simulink and lsim

Table 5.2 shows the values of R, Q and iteration number when the normalised position error reaches less than 0.01 (DR4). All three controllers can converge within 10 iterations.

Table 5.2: Some trials conducted to tune R and Q values controllers

Trial	X-axis			Y-axis			Z-axis		
	R	Q	Iteration	R	Q	Iteration	R	Q	Iteration
1	0.5	50	>10	0.01	40	>10	0.001	500	>10
2	0.4	50	>10	0.005	40	>10	0.0005	500	>10
3	0.3	50	10	0.005	300	>10	0.0001	500	5
4	0.2	40	9	0.001	300	9	0.0001	1000	4
5	0.1	30	7	0.001	400	7	0.0001	1500	3

5.2 Trajectory profile generation 2

The issue with the previous algorithm is that the gantry robot moves back to the origin after grasping the object which increases the distance travelled, typically the distance to travel from initial location to destination is shorter than from initial location to origin to destination. Therefore, this could put unnecessary stress on the actuators. Thus, a new trajectory profile is generated and shown in Figure 5.5.

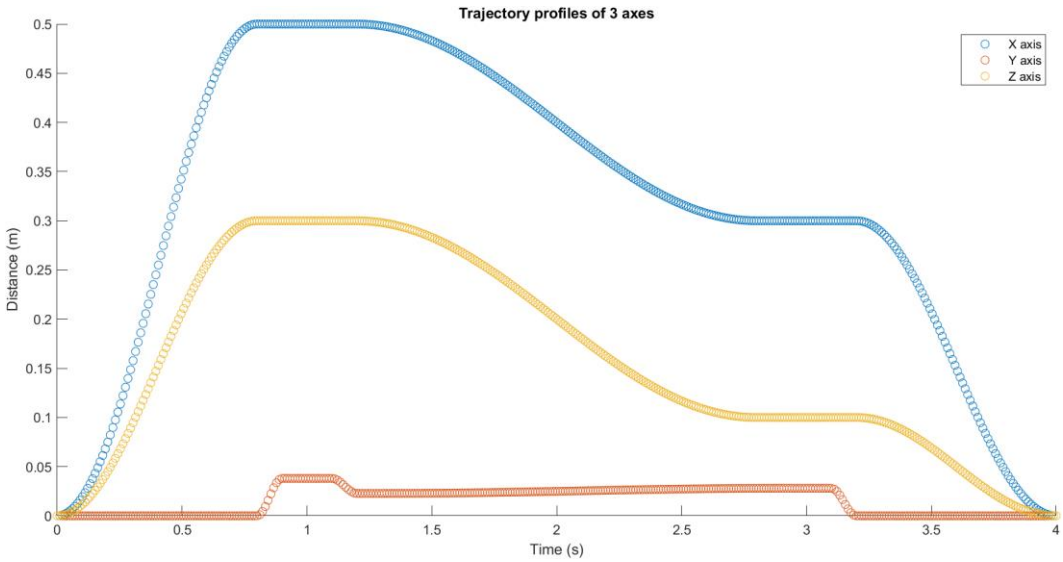


Figure 5.5: Trajectory profile for each axis controller to follow

It can be seen that between 1.2s and 2.8s, the scatter points are closer to each other in Figure 5.5 than in Figure 5.3, this indicates that the changes between each time sample are less and thus put less stress on the actuators.

To avoid the grasped object from being 'dragged' along the floor when the X and Y-axis move, the Z-axis will lift the object slightly before the deadtime of X and Y-axis ends. A closeup view of the Z profile is shown in Figure 5.6, where each part is colour coded. The new values for the trajectory profile can be found in Table 5.3.

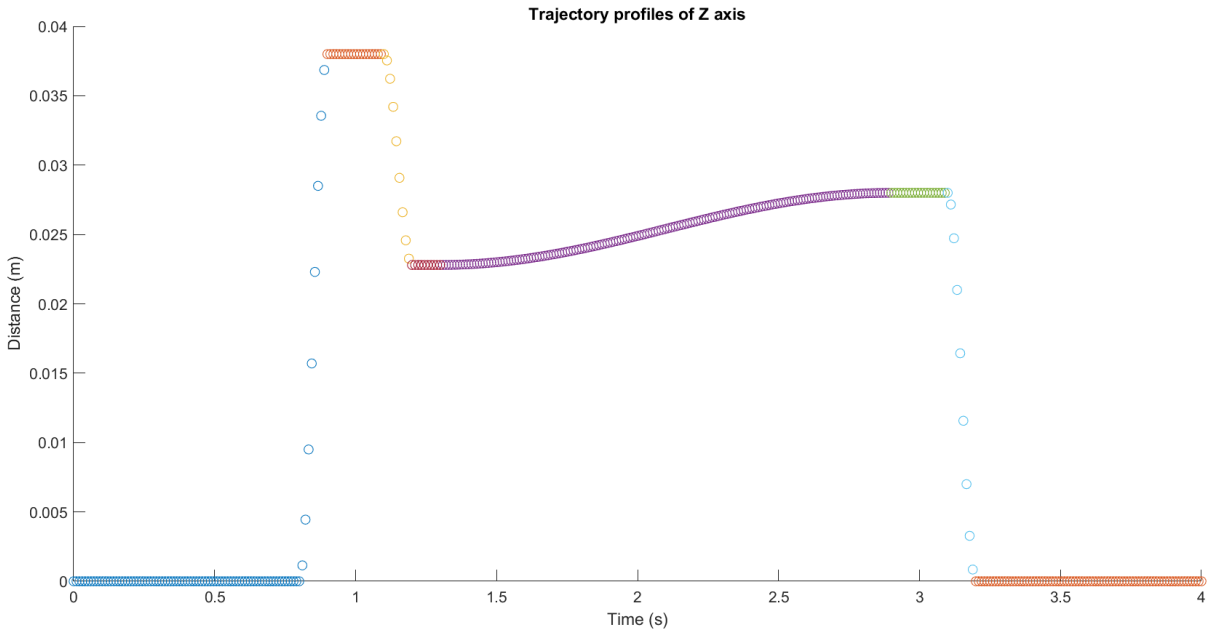


Figure 5.6: Trajectory profile of Z-axis, each colour shows different parts of the trajectory profile generation, and the timing is given in Table 5.3

Table 5.3: Values for trajectory profile generation. (D denotes deadtime, d denotes rise/fall time and A denotes amplitude.)

Parameters	X-axis	Y-axis	Z-axis
A1 (m)	0.5	0.3	0.038
A2 (m)	0.3	0.1	0.038*60%
A3 (m)	-	-	0.028
d1 (s)	0.8	0.8	0.1
d2 (s)	1.6	1.6	0.1
d3 (s)	0.8	0.8	1.6
d4 (s)	-	-	0.1
D1 (s)	0.4	0.4	0.8
D2 (s)	0.4	0.4	0.2
D3 (s)	-	-	0.1
D4 (s)	-	-	0.2
D5 (s)	-	-	0.8

This algorithm is capable of grasping the object and releasing it to its destination accurately. However, the Z-axis will start to operate in the first iteration. This causes issues where the Z-axis converges before X and Y-axis, hence causing the gripper to collide and damage the object as captured at the 4th iteration shown in Figure 5.7.

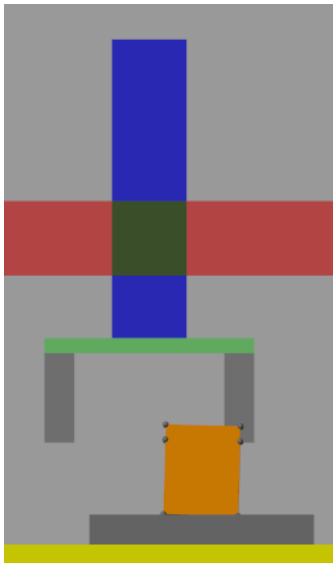


Figure 5.7: Gripper colliding with object due to Z-axis converging before X and Y-axis

5.3 Trajectory profile generation 3

In light of the issue experienced in the previous subchapter, a new control algorithm was designed where the Z-axis shall ‘wait’ until X and Y-axis have relatively small tracking errors. The new algorithm is illustrated using the flow chart shown in Figure 5.8.

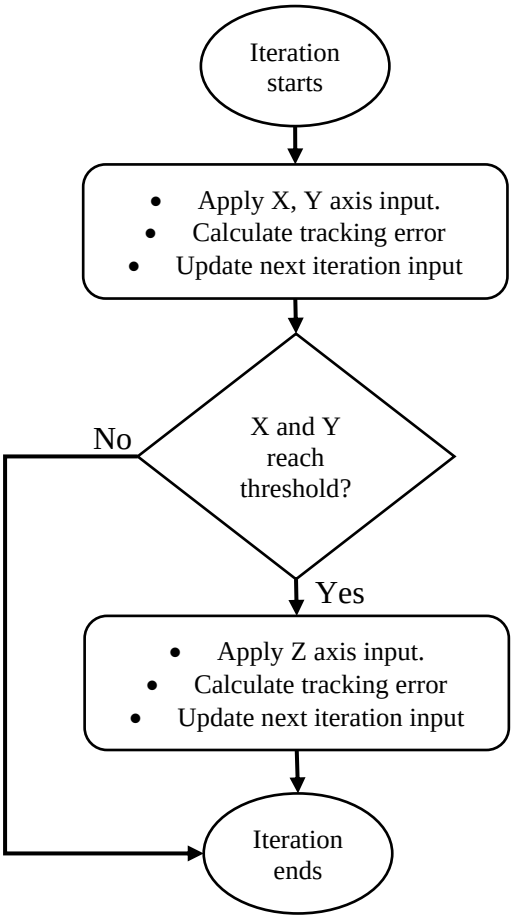


Figure 5.8: Flow chart of new control algorithm

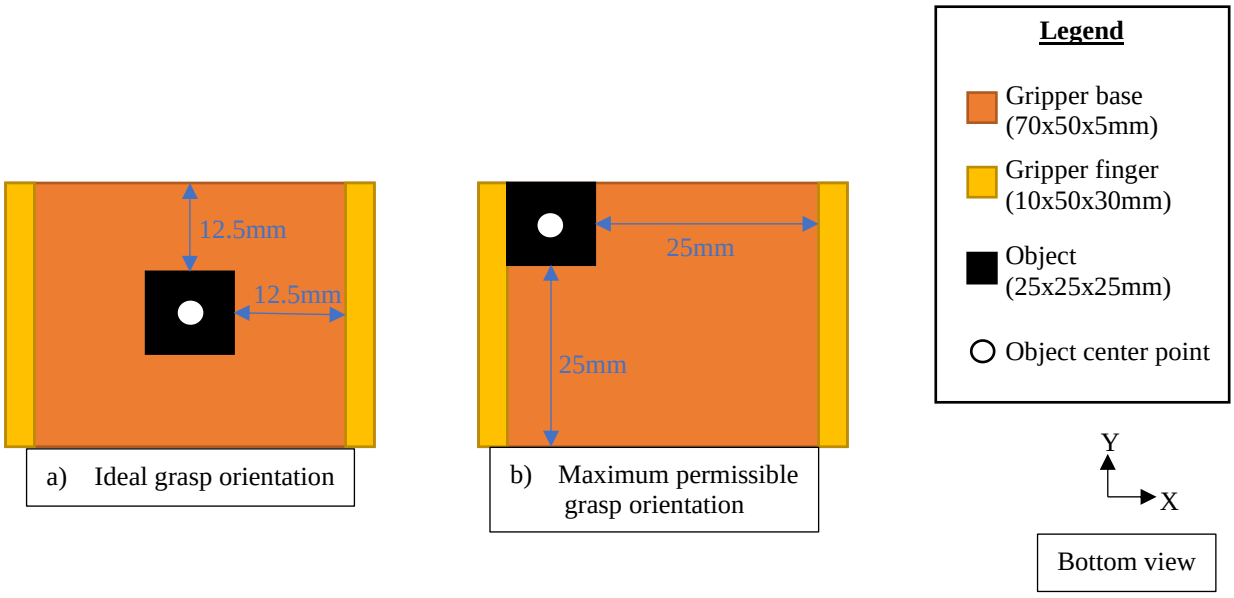


Figure 5.9: Illustration of gripper and object location for ideal grasping orientation and maximum permissible orientation

The best location to grasp the object is when it reaches the centre of the gripper as illustrated in Figure 5.9a, but for the controller to converge to zero tracking error would take infinite iterations. Therefore, as long as the object is within the gripper grasping area shown in Figure 5.9b, the object can be regarded

as safe to grasp, hence the Z-axis can start to lower the gripper. The earliest safe distance/threshold to allow Z-axis to operate is when X and Y-axis reaches 97% of the intended travel distance (A1).

To get the most reliable reading to compare with the threshold, an average data of the position feedback during each axis deadtime is considered (between 0.8s and 1.2s, and 2.8s and 3.2s as shown in Figure 5.5).

Figure 5.10 shows the normalised error of the three axes. The Z-axis normalised error remains constant during the first 4 iterations because X and Y-axis are still not within the threshold. The normalised tracking error of Z-axis reduces down to less than 0.01 in the 6th iteration which is within the 10-iteration limit mentioned at the beginning of the chapter.

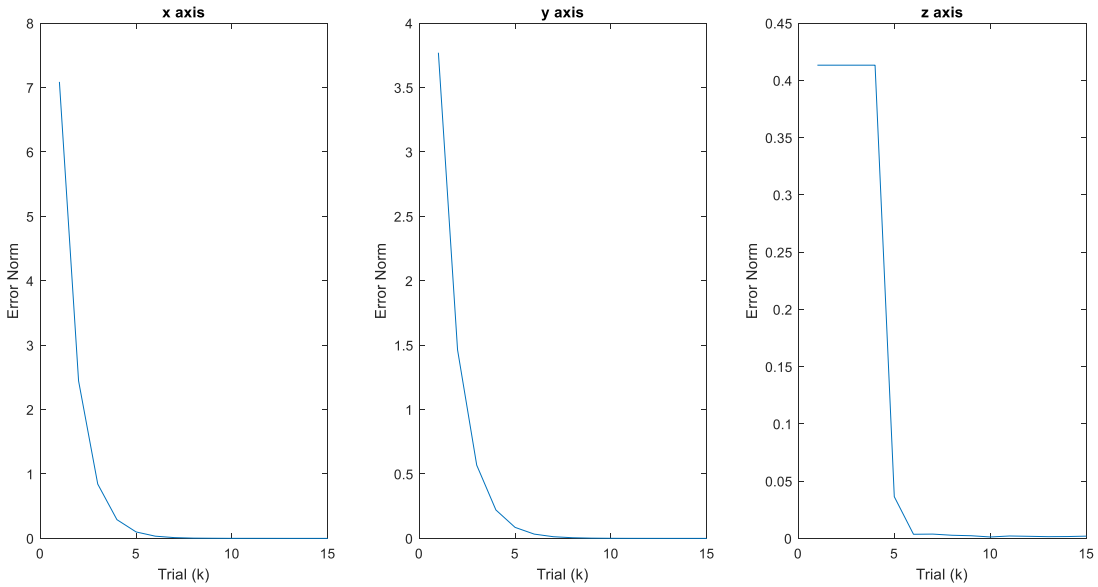


Figure 5.10: Normalised error for X, Y and Z axis with the new control algorithm

Figure 5.11 shows the position feedback of the three axes during different iterations while Figure 5.12 shows the changes to the input signal to realise the ideal tracking trajectory. This is evident that ILC learns from previous iteration mistakes and corrects itself.

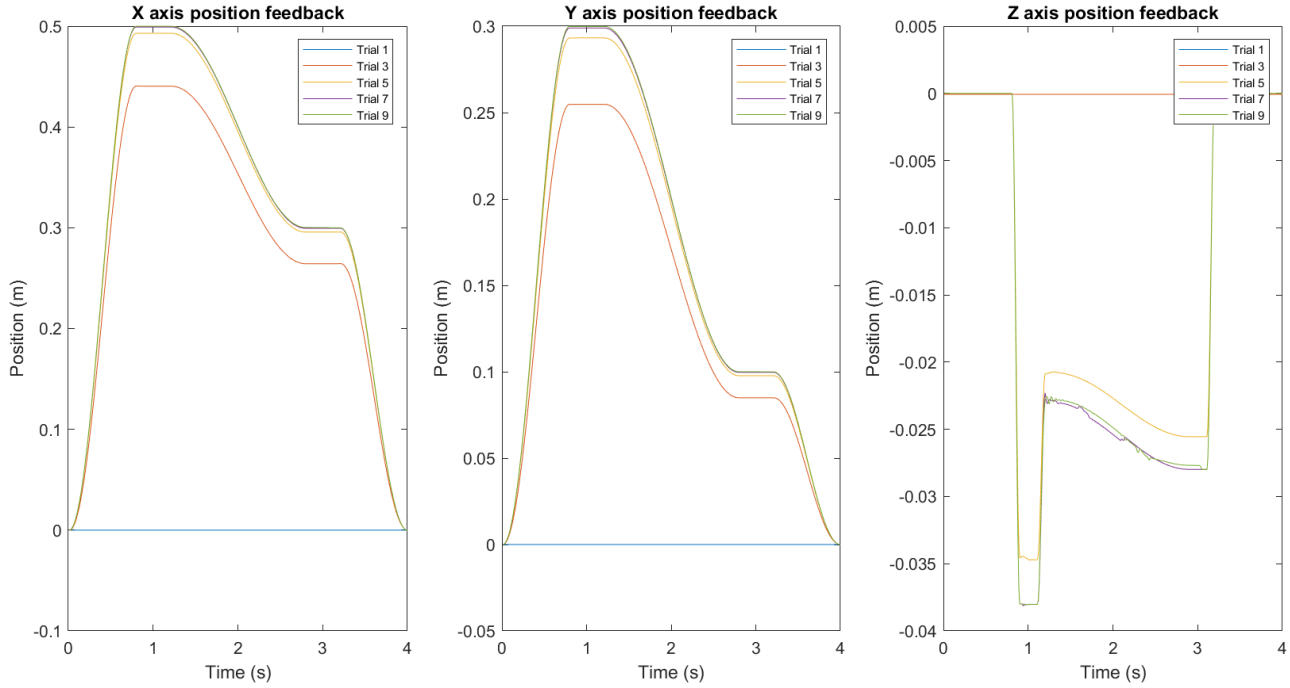


Figure 5.11: Position feedback of X, Y and Z-axis for different iterations

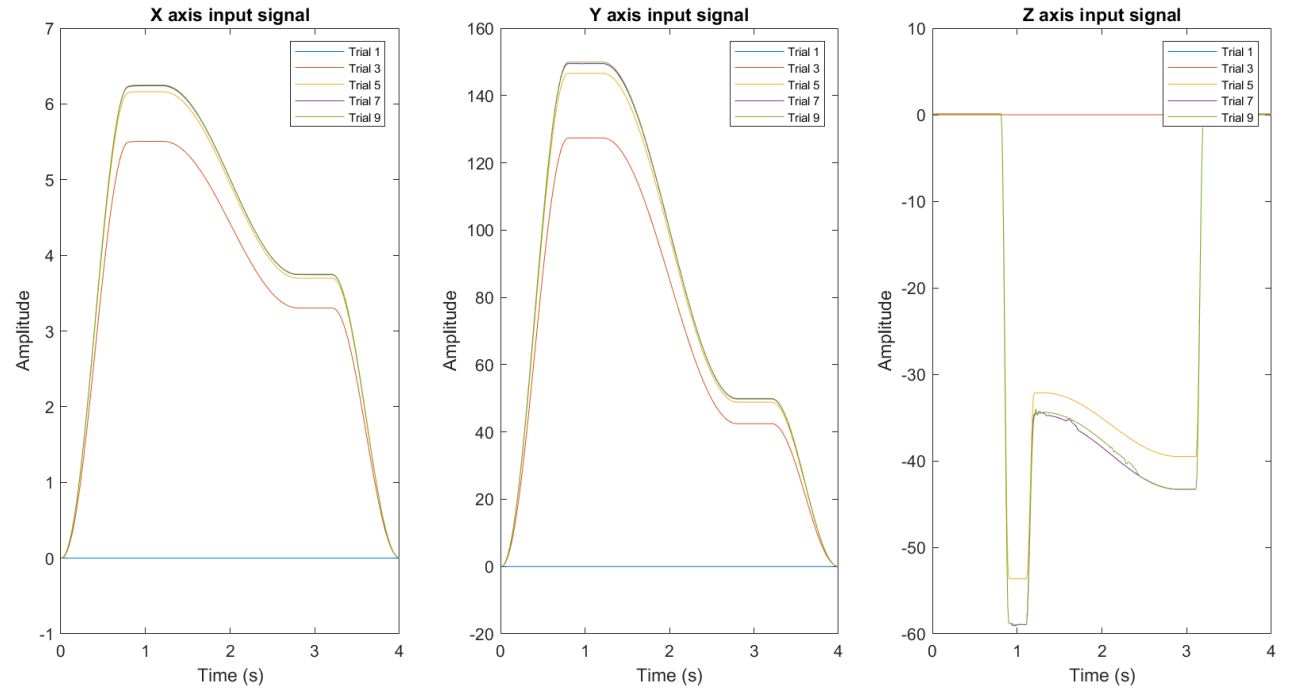


Figure 5.12: Input signal for X, Y and Z-axis for different iterations

5.4 Summary

This chapter has successfully designed a norm-optimal ILC algorithm to control the X, Y and Z-axis of the gantry system which is capable of converging within the first 10 iterations. The following chapter aims to design the controllers for the gripper to grasp the object of interest.

6 Gripper Controller Design

Before the gripper controller is designed, some modifications are required to the X, Y, and Z-axis profiles.

The rise and fall periods of Z-axis shown in Figure 5.6 are relatively short. Therefore, to reduce the strain put on the system, the total period has increased from 4s to 5s and the middle transition period has been reduced. This allows the Z-axis to operate over a longer period. The profile would still have 400 samples giving it a sampling rate of 0.0125s. The new profile is shown in Figure 6.1.

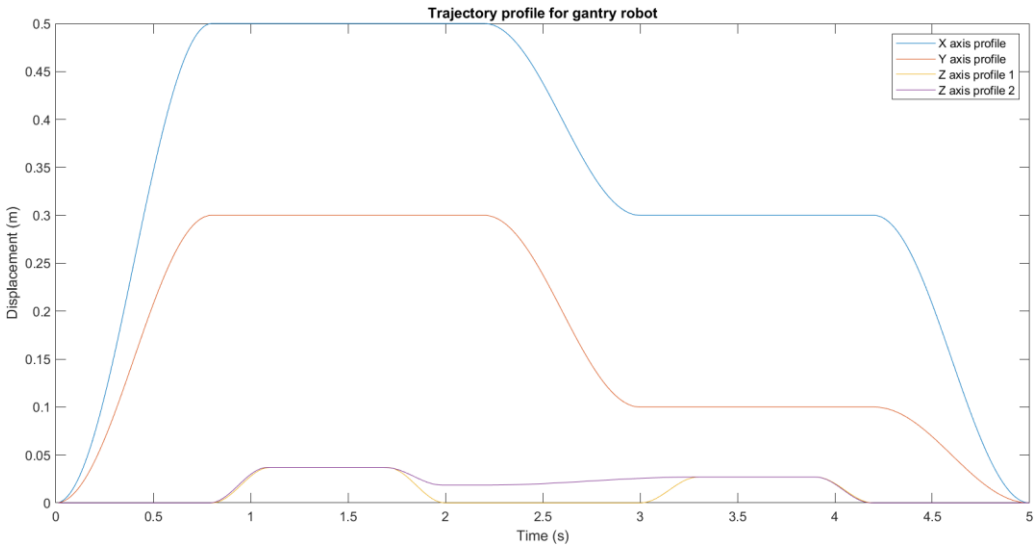


Figure 6.1: Gantry new profile with longer deadtime and longer iteration cycle

The previous lifting slightly profile (profile 2) will be discarded as it is safer to fully retract Z-axis before moving in the X and Y directions as this ensures the object will not collide with anything in the environment. The values to generate the profiles can be found in Table 6.1.

Table 6.1: Values to generate profiles seen in Figure 6.1. (D denotes deadtime, d denotes rise/fall time and A denotes amplitude.)

Parameters	X-axis	Y-axis	Z-axis 1	Z-axis 2
A1 (m)	0.5	0.3	0.037	0.037
A2 (m)	0.3	0.1	0.027	0.037*50%
A3 (m)	-	-	-	0.027
d1 (s)	0.8	0.8	0.3	0.3
d2 (s)	0.8	0.8	0.3	0.3
d3 (s)	0.8	0.8	0.3	0.8
d4 (s)	-	-	0.3	0.3
d5 (s)	-	-	-	0.3
D1 (s)	1.4	1.4	0.8	0.8
D2 (s)	1.2	1.2	0.6	0.6
D3 (s)	-	-	1.0	0.2
D4 (s)	-	-	0.6	0.6
D5 (s)	-	-	0.8	0.8

6.1 Impedance Gripper

The schematic for the impedance controller is shown in Figure 3.4 and Figure 3.5. The design will be split into 2 subsections where one focuses on the PD controller and the other on Impedance.

6.1.1 Position Control

To ensure the gripper fingers reach their desired location, a simple PD controller is added to correct any position error. Since the gripper fingers are identical to each other, it is only necessary to tune the values of one of the controllers and the other will take the same value. The schematic for the PD controller is shown in Figure 6.2.

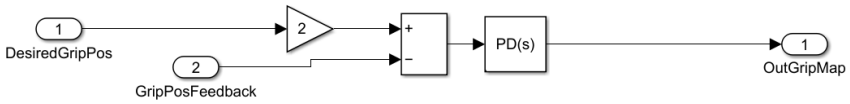


Figure 6.2: PD controller schematic for position control of gripper fingers

The PD controller aims to correct position errors as fast as possible. This can be achieved by ensuring the step response has a small transient time. Figure 6.3 shows the response of the gripper finger dynamics using PID tuner. The values can be found in Table 6.2.

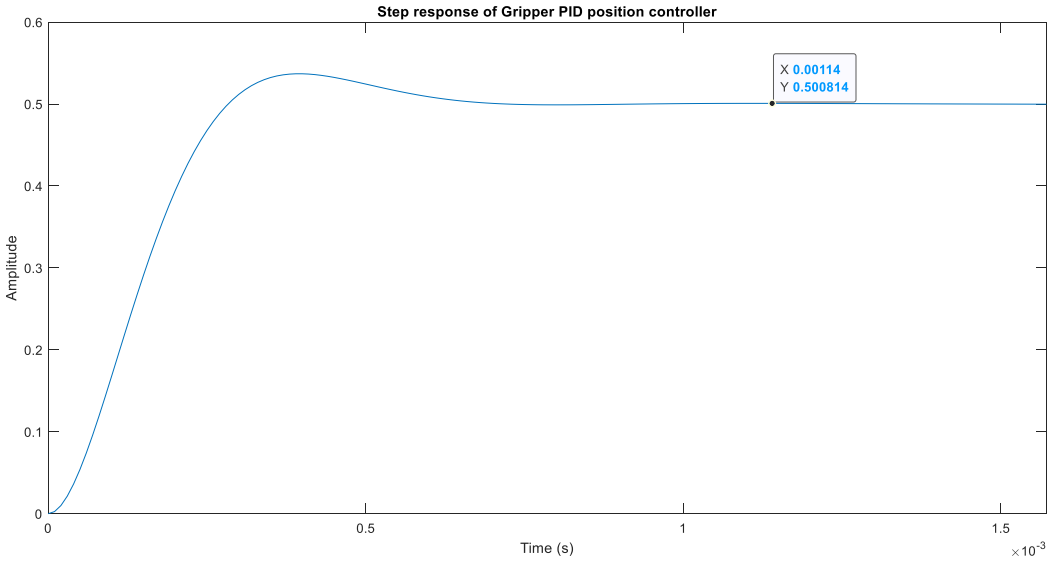


Figure 6.3: Step response of PD controller connected in series with gripper finger plant

Figure 6.3 identifies a problem where the steady-state value remains at 0.5. Therefore, a scalar gain of 2 is added to the signal to increase the steady-state value to 1.

Table 6.2: Values for PD Controller

Parameter	Value
P	1493.03
D	0

6.1.2 Force Control

There is a direct relationship between the gripper's grasping position and the force applied to the object. To change the force applied, the controller has to change the reference trajectory for PD controller to follow. Impedance control can be used for this operation by using the schematic shown in Figure 6.4 which is adapted from the general schematic shown in Figure 2.2.

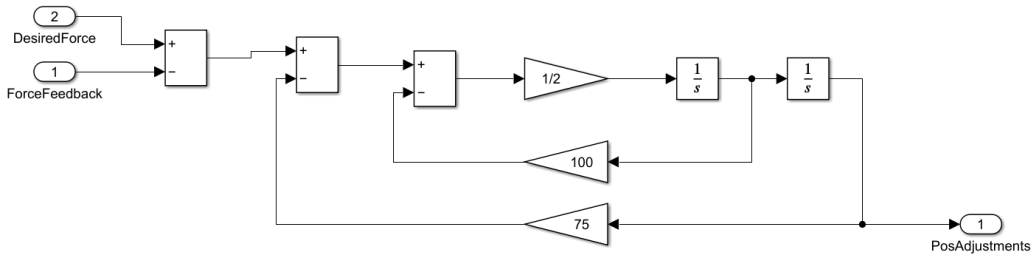


Figure 6.4: Schematic of Impedance Control

For the gripper to skim the surface of the object, it has to travel 0.0115m. This calculation can be done by referring to Figure 5.9a. A conservative approach is taken where the gripper is set to close a further 0.003m to grasp the object firmly. Figure 6.5 shows the resultant reference profile.

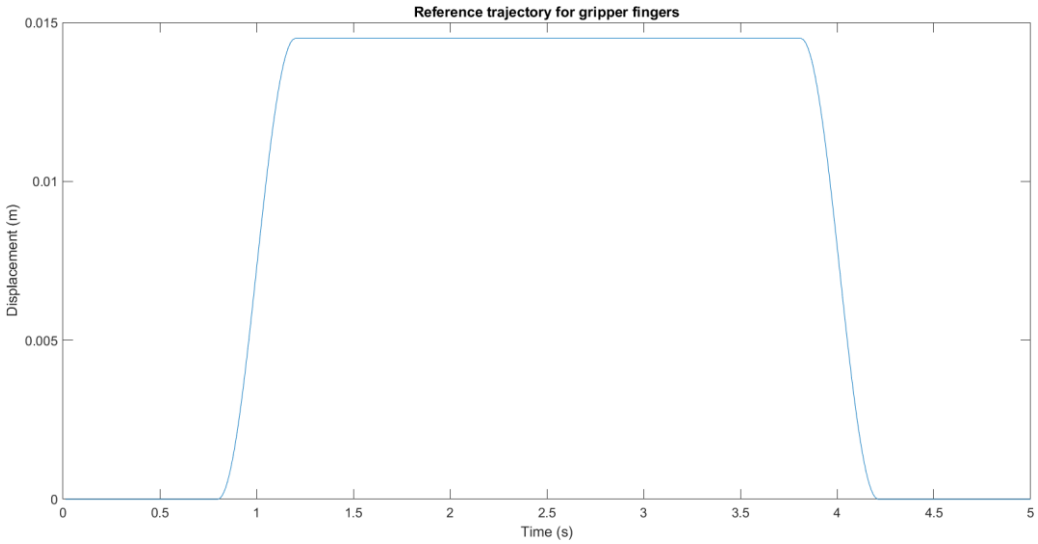


Figure 6.5: Reference trajectory for gripper fingers to follow

The minimum force to grasp an object is calculated using (4.11). This value is known as desired force and is increased by 20% to account for any modelling or calculation uncertainties. The force feedback would be measured by placing force sensors on the fingers. The resultant force adjustment will be added to the reference profile as shown in Figure 6.6.

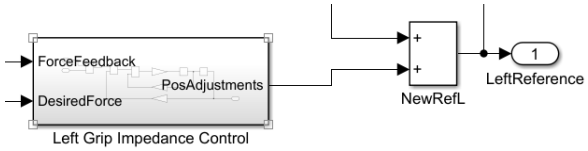
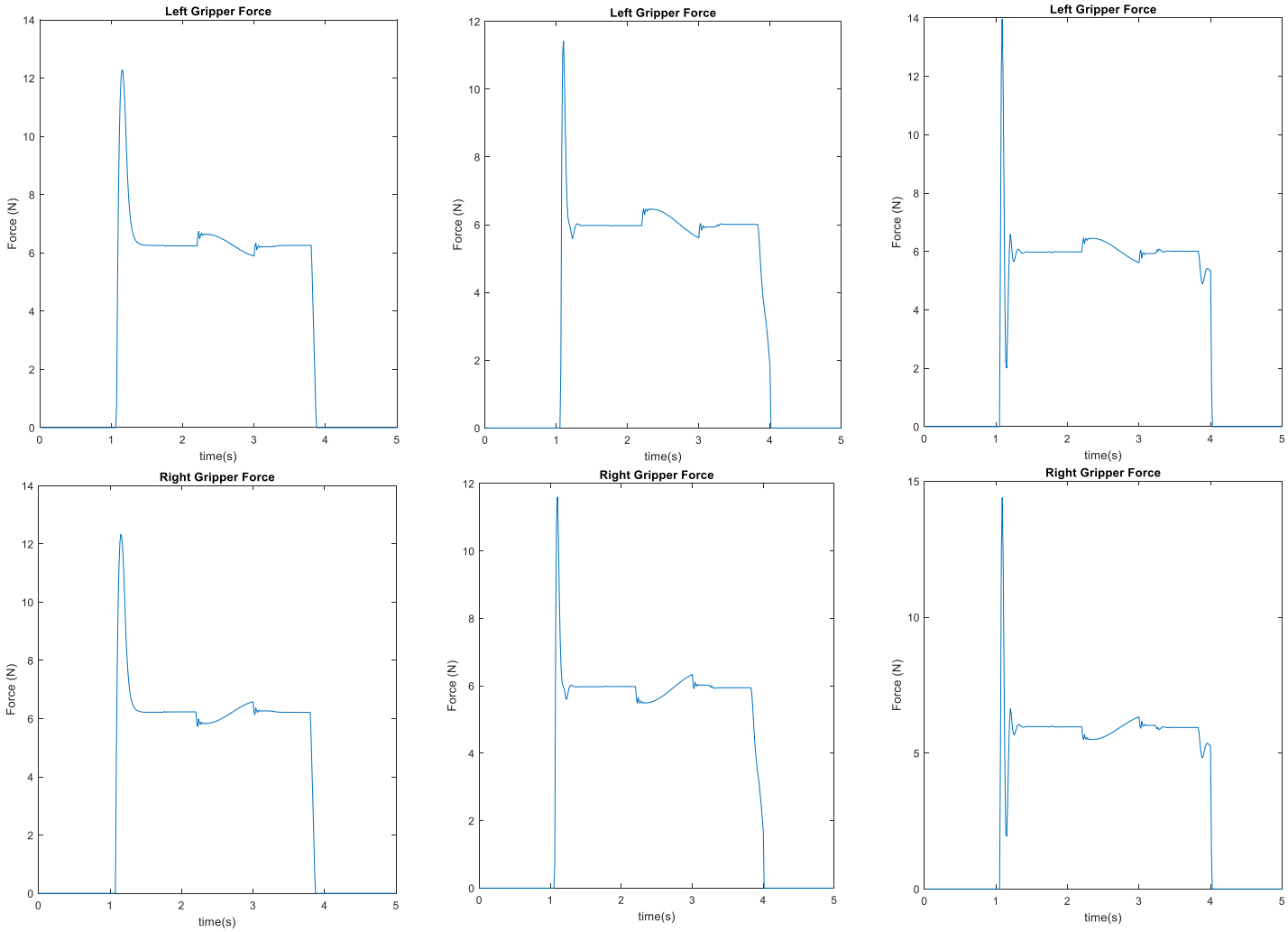


Figure 6.6: Input and output of impedance controller

The values of M, B and K need to be tuned to get the best response. Table 6.3shows a summary of the values tested. The gripper forces of some values are shown in Figure 6.7.

Table 6.3: Values for Impedance control and remarks via observing simulation and graph

Trials	M	B	K	Remarks
1	2	300	200	Huge spike, stable grasp
2	2	100	50	Smaller spike, small valley but still good grasp
3	2	50	50	Small spike reduction, multiple valleys, slip slightly
4	2	75	50	Amplitude of valley is smaller, slip slightly
5	2	100	75	Good grip, minimal to no valley



a) M = 2, B = 300, K = 200 b) M = 2, B = 100, K = 50 c) M = 2, B = 50, K = 50

Figure 6.7: Interaction forces between gripper finger and object under different B and K values.

Since the reference profile takes a conservative approach, initially it will always apply excessive force as evident in Figure 6.7. The sine wave-like phenomenon happening around 2s to 3s is caused by the acceleration and deceleration effect explained in section 4.2.2. As demonstrated in Figure 6.7a to Figure 6.7c, although the max amplitude of the spike decreases, the controller is less stiff and damped hence producing some sharp valleys where the gripper behaves like jelly and could potentially cause the gripper to drop the object. Therefore, care has to be taken when selecting the values.

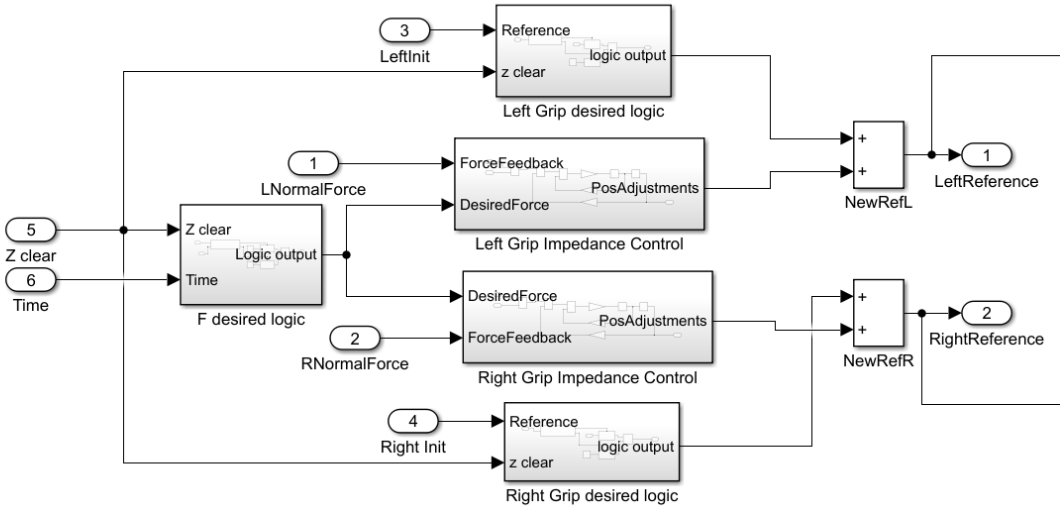


Figure 6.8: Full schematic of Impedance Control, without PD controller

To ensure the gripper can grasp the object firmly, the X, Y and Z-axis should be at a reasonable position (within a certain grasping volume) before the gripper is allowed to grasp. Therefore, a logic system is implemented as shown by the flowchart in Figure 6.9 which proceed the one in Figure 5.8. The full schematic of impedance controller is shown in Figure 6.8. Details can be found in Appendix C.

The calculation for the Z-axis threshold is similar to Section 5.3, where the threshold is set to 97% of the final value.

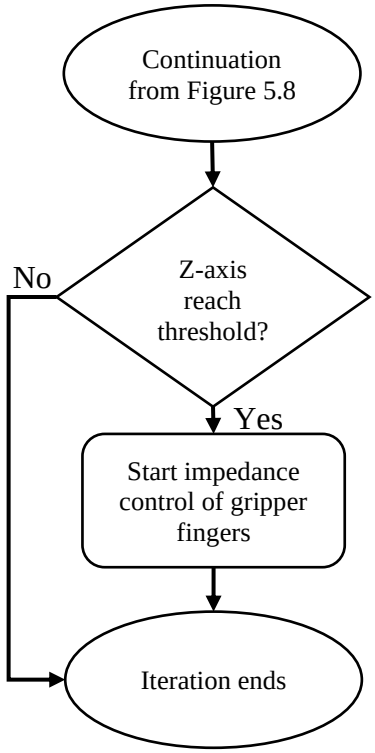


Figure 6.9: Flow chart of the full impedance control algorithm

6.2 ILC Gripper

The ILC algorithm will largely follow the one shown in Figure 6.9 with some modifications to suit the task better. This section will be split into 3 subsections with the following aim

- Ensure gripper fingers move to the correct location.
- Ensure the gripper uses the least amount of force.
- Investigate the effect of updating the plant model with the weight of the grasped object.

6.2.1 Position Control

This controller uses Norm-optimal configuration with (5.3) and (5.4) serving as update equations. Recall that the plant has been identified in Section 4.1.3. To tune the values of R and Q, similar steps used in Section 5.1.1 are used and the best values are R = 0.00005, Q = 2000.

Figure 6.10 shows the normalised error for each part of the system. From there, the gripper can be seen to converge to a normalised error of 0.01 at the 10^{th} iteration which fulfils DR4.

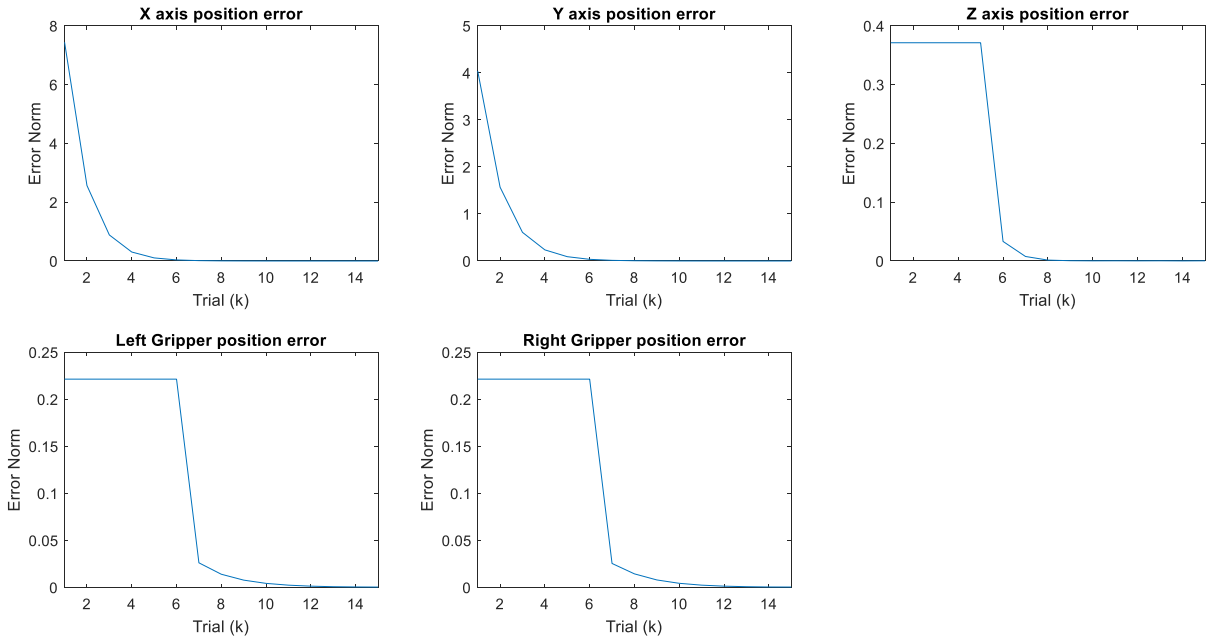


Figure 6.10: Normalised position error for the whole gantry system

The ILC operation flowchart is shown in Figure 6.13 which is a modified version of the one shown in Figure 6.9. As ILC only updates the input vector at the end of the iteration, the information passed to the controller to make decisions can be seen as delayed by one iteration, therefore, the gripper must have a firm grasp of the object before transporting it. As a result, two grasping reference trajectory profiles are generated. The first would grasp and release the object to ensure the average force feedback is within desired force value. Only if the average exceeded this value, should the second profile be used. The second profile is the same as seen in Figure 6.5, where it will grasp the object fully during the whole operation. The overall profile for the whole system is shown in Figure 6.11, while the comparison between the two gripper profiles is shown in Figure 6.12.

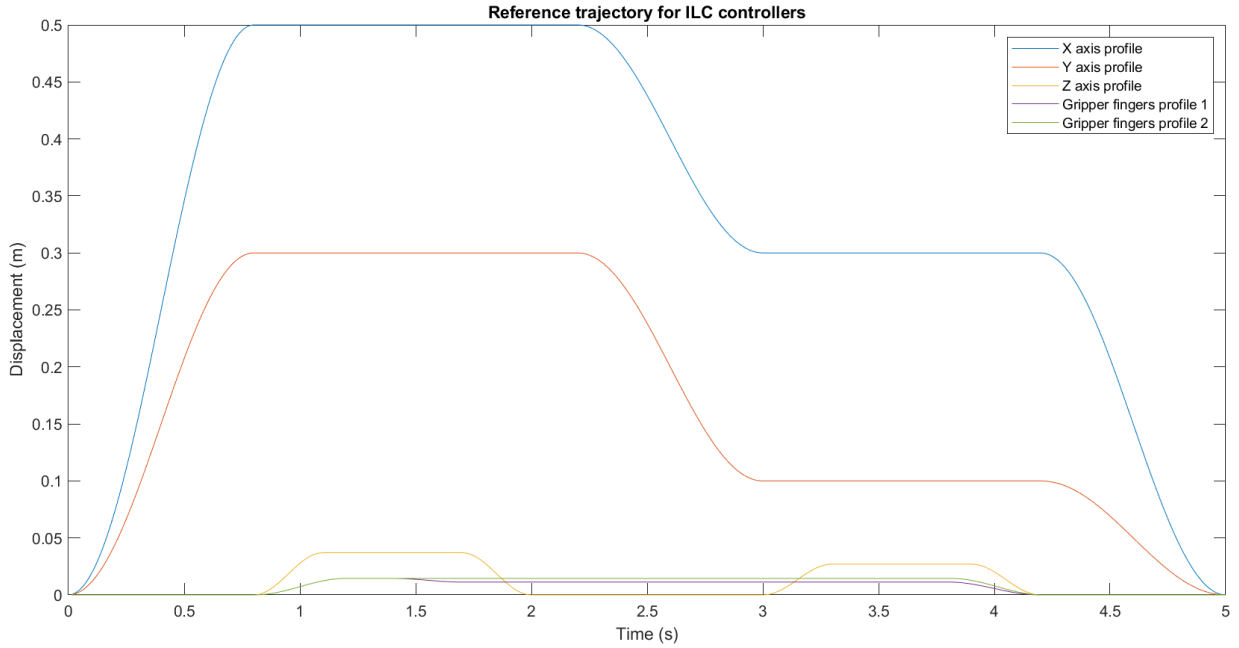


Figure 6.11: Reference trajectory profiles for X, Y, Z-axis and gripper fingers to follow

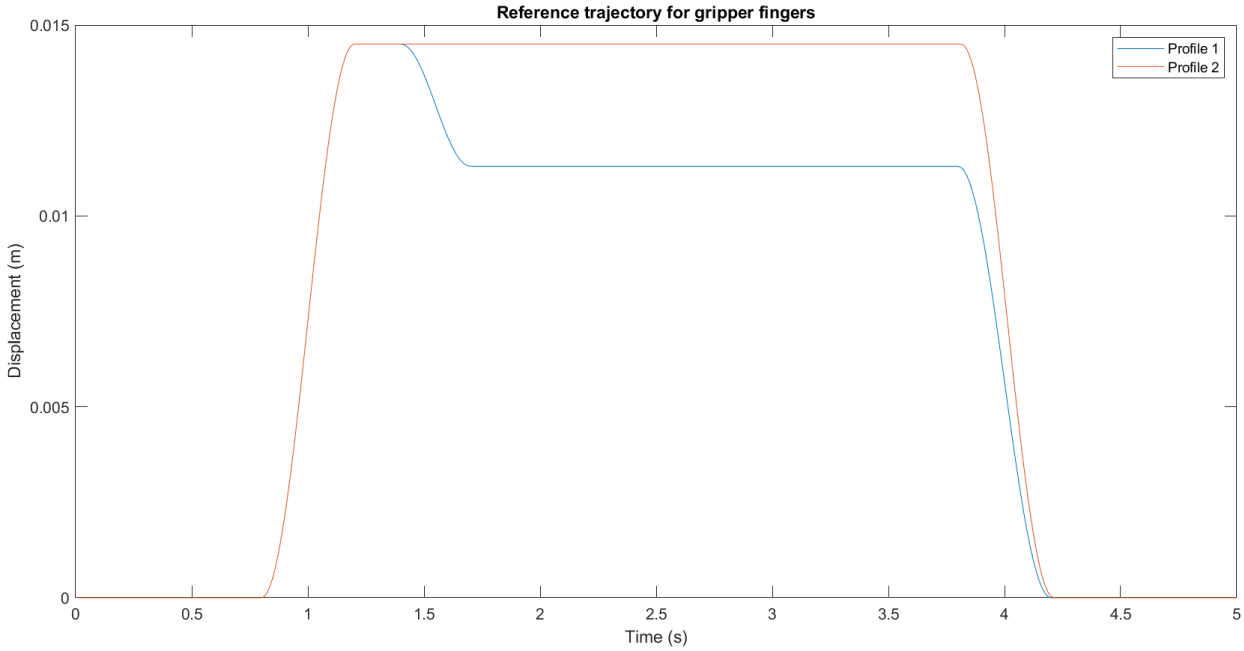


Figure 6.12: Two different reference trajectories for gripper fingers controller to operate under different conditions

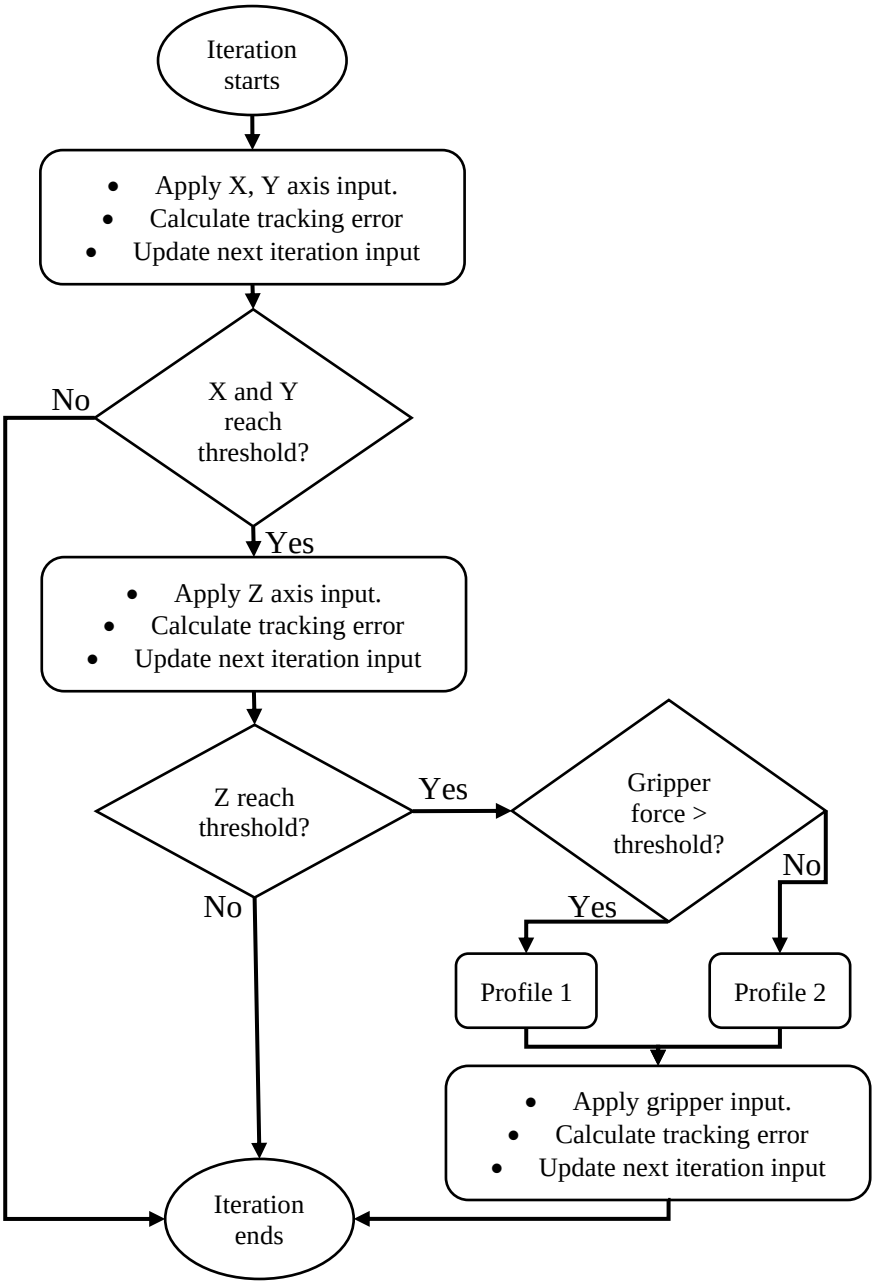


Figure 6.13: New control algorithm with different grasping profiles

For the threshold of X, Y and Z-axis to be fulfilled, they have to be within a 97% accuracy when comparing the average difference between position feedback value and reference trajectory. The force threshold is set to be 20% higher than the minimum force (F_{min}) calculated using (4.11) to account for any uncertainties or modelling errors that may arise.

6.2.2 Force Control

To ensure the gripper does not use excessive force, force control is needed. As discussed, and justified in Section 3.3.2, a simple gain ILC update would be used. The updated flow chart is shown in Figure 6.14.

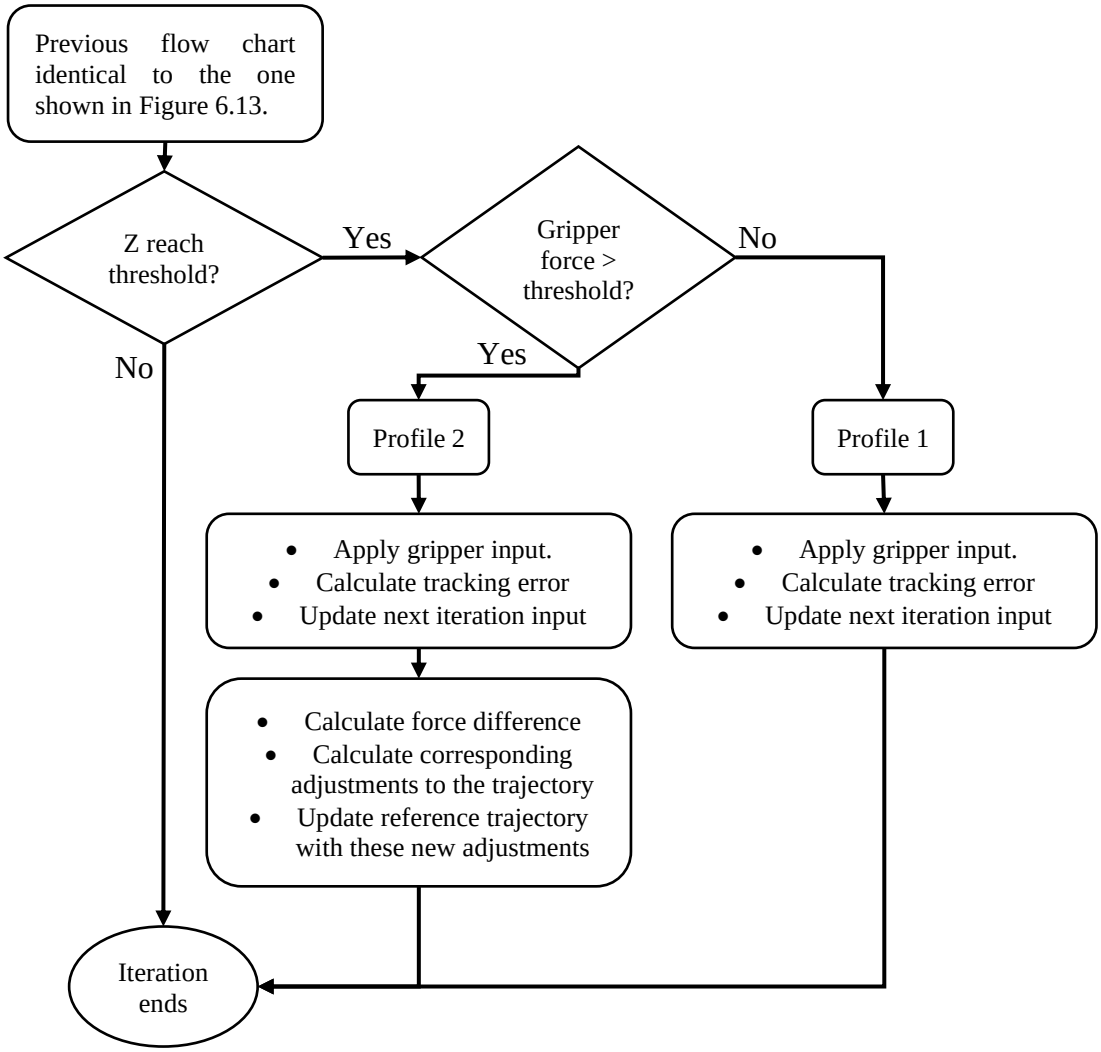


Figure 6.14: New flow chart showcasing new algorithm with features reducing the grasping force

There are a few calculations required to modify the reference trajectory for the gripper to grasp with minimal excessive force.

The first step is to calculate the force difference (F_{diff}) using the following equation

$$F_{diff} = (F_{min} * 120\%) - F_{measured} \tag{6.1}$$

where F_{min} is the minimum force, $F_{measured}$ is the measured force.

The interaction force is modelled using a spring-mass system explained in Section 4.2, therefore the force is directly proportional to the distance between two bodies overlap ($d_{overlap}$). The force can be calculated using the following formula

$$F_{applied} = d_{overlap} * stiffness \tag{6.2}$$

The stiffness is set to be 10000N/m. For an object of mass 2kg, using (4.11), the F_{min} to lift the object is 4.91N.

Combining (6.1) and (6.2), it is possible to calculate the changes to the trajectory (d_{diff}) for the fingers to reduce the force applied to the object which is shown in (6.3).

$$d_{diff} = \frac{F_{diff}}{stiffness} \tag{6.3}$$

d_{diff} can be regarded as an error signal and thus the update to profile 2 can be carried out using ILC using (3.4), resulting in the following equation.

$$\boldsymbol{ref}_{k+1}(t) = \boldsymbol{ref}_{k}(t) + \beta(t)\boldsymbol{d}_{diff}(t) \tag{6.4}$$

where $\beta(t)$ is set as a scalar gain of 0.5 to ensure the correction is smooth and not destructive.

However, as the force calculation is only done during the deadtime (between 1.2s and 3.8s), the value of d_{diff} before and after this deadtime is 0. This causes an issue shown in Figure 6.15, where it is not smooth all the way. Therefore, to produce a new grasping profile for the gripper, the whole profile has to be regenerated with the new magnitude at 1.2s and 3.8s using (5.1) and (5.2) producing profile seen in Figure 6.16.

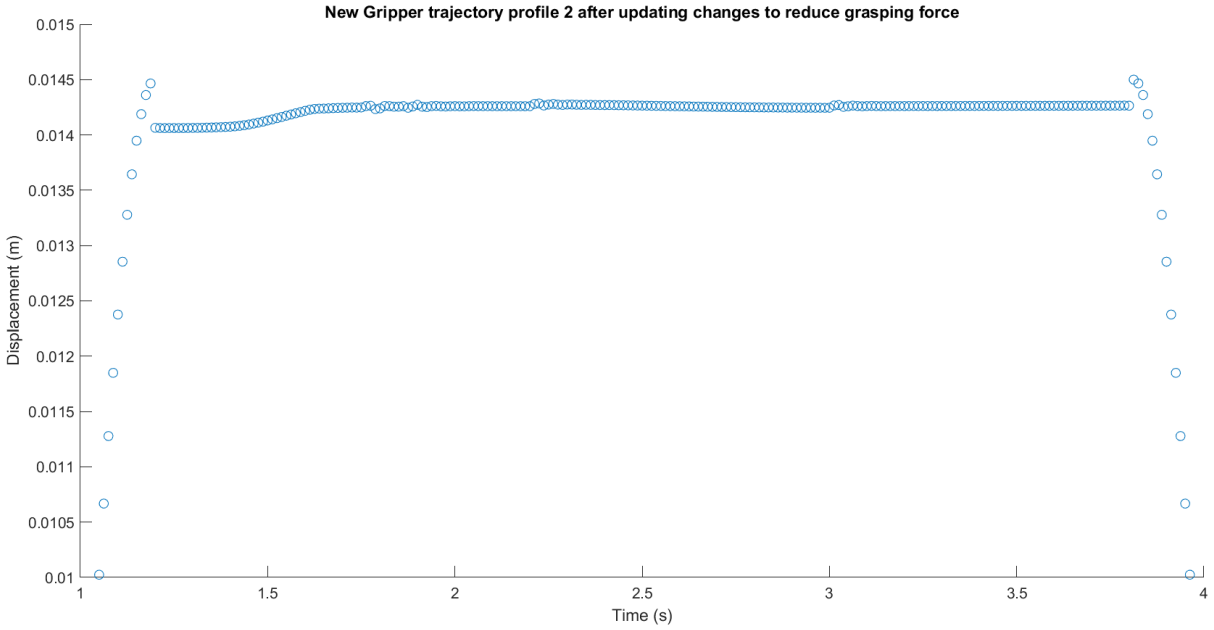


Figure 6.15: Non-smooth edge of the new reference profile via ILC

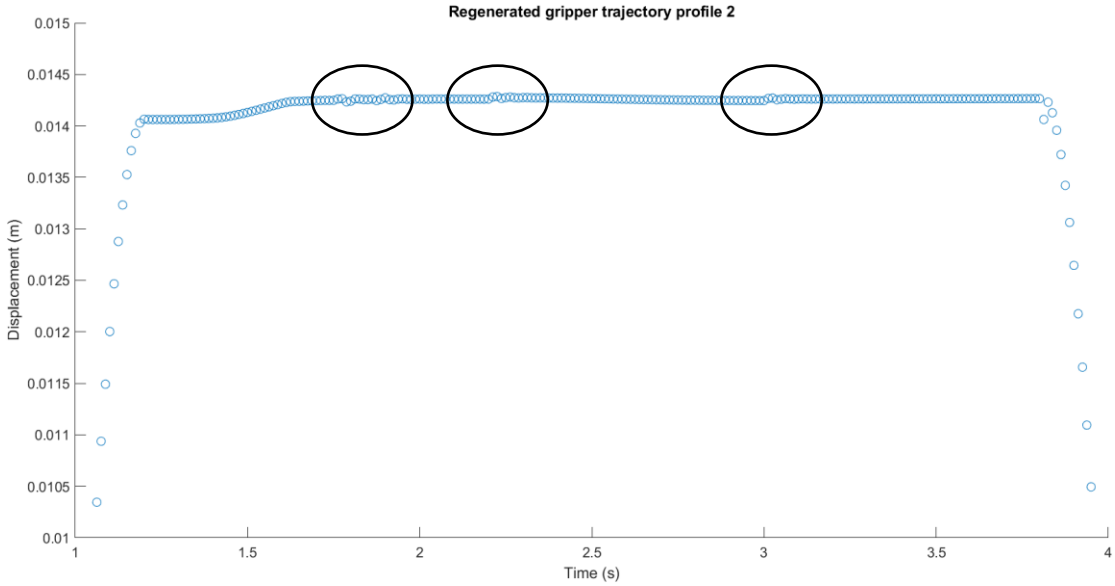


Figure 6.16: Non-smooth data within the deadtime of the grasping trajectory profile

Unfortunately, part of the profile (circled) in Figure 6.16 is not smooth. This can be the result of discretising the data. If left unfixed, it can cause the system to become increasingly unstable.

One of the solutions to resolve this issue is to increase the sampling rate to get more accurate data, however, this comes at a cost of increased computational complexity. Fortunately, another solution is to apply a non-casual filter to the signal. This can be done using MATLAB built-in function 'smoothdata' which returns a moving average using a fixed window size. Figure 6.17 shows a comparison between smoothed and non-smoothed data.

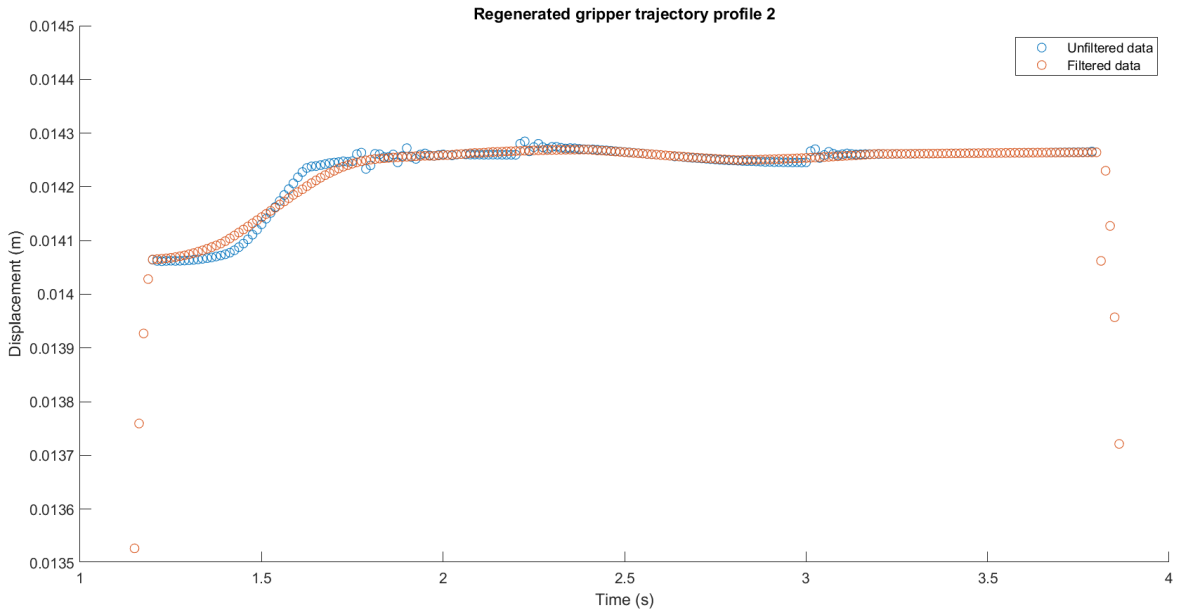


Figure 6.17: New trajectory profile before and after passing through a non-casual filter

Figure 6.18 shows the normalised position and force error. All the position controller converges at roughly the 10th iteration with minimal error. The average force the object experienced is shown in Figure 6.19. Recall that DR6 requires the gripper to use no more than 25% of minimum force to grasp the object within 20 iterations after the gripper starts grasping, the simulation shows that the gripper starts grasping at the 7th iteration and after 17 iterations, both fingers are below that value.

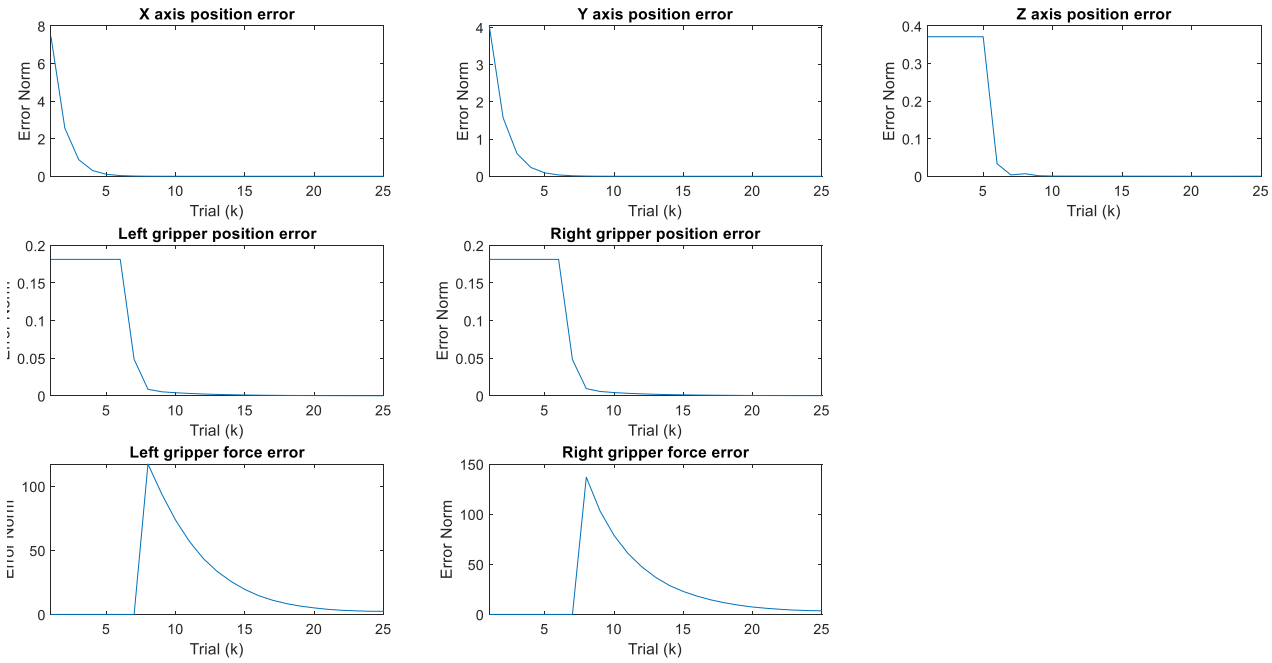


Figure 6.18: Normalised position and force error for each part of the gantry robot controller by ILC controllers

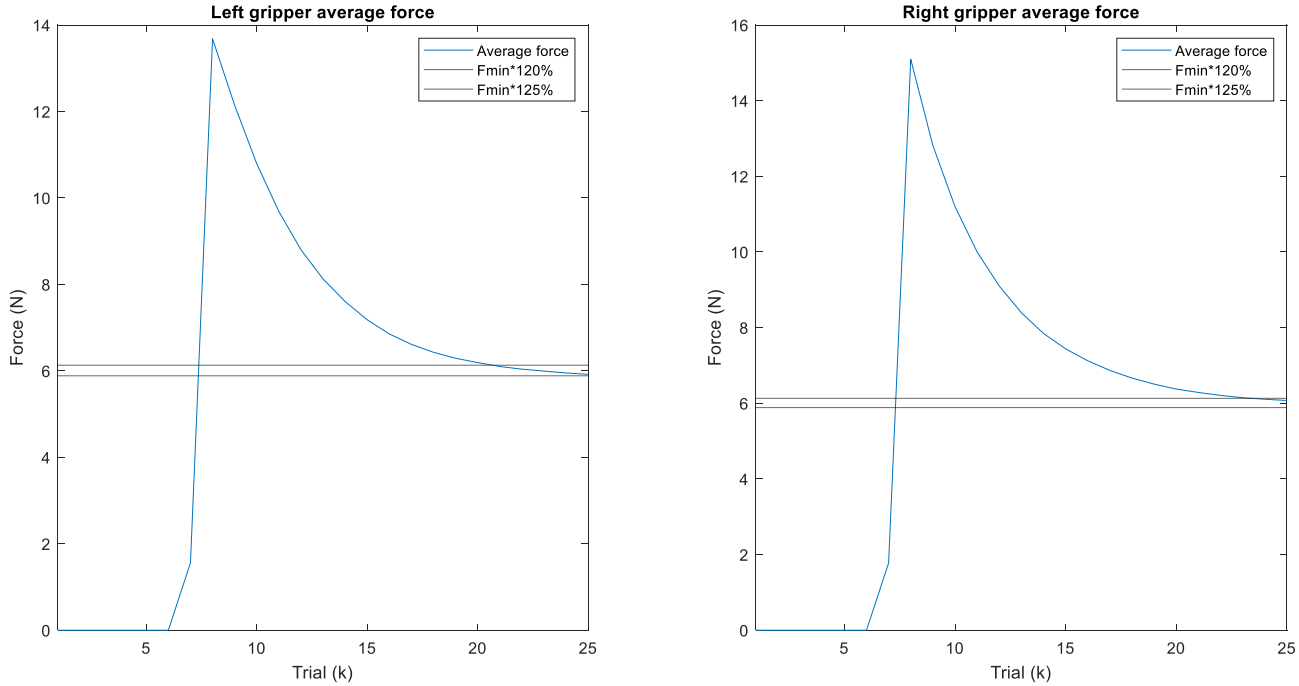


Figure 6.19: Average forces experienced by the object and the 20% and 25% increase in force over minimum force

It can also be seen in Figure 6.20 that at the 7th iteration, profile 1 is used. Once it is concluded that the gripper has a solid grasp (8th iteration), profile 2 is used. The controller has also learned and adapted the grasping profile when comparing the 8th iteration profile to the 40th iteration. The effect of the phenomenon of acceleration and deceleration force discussed in Section 4.2.2 can be seen in the wave-like structure at the 40th iteration, where the gripper profile changes to account for those forces.

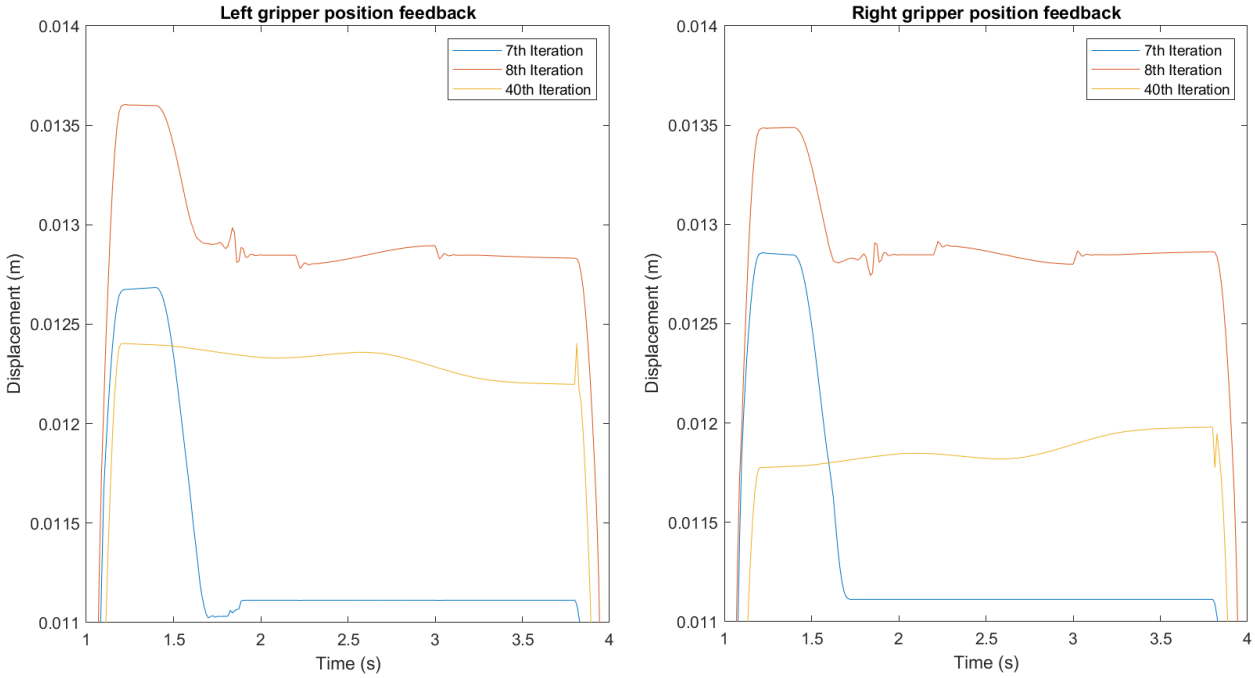
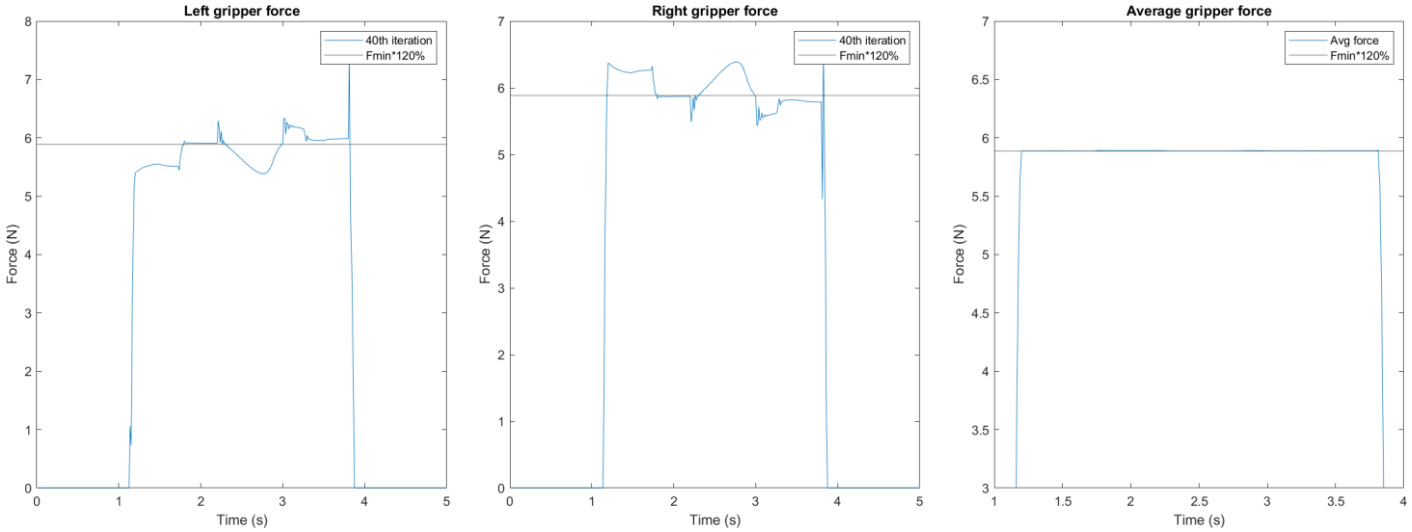


Figure 6.20: Left and Right gripper position feedback for 7th, 8th and 40th iteration

Figure 6.21 shows the force feedback experienced. It is not flat as compared to the impedance counterpart (Figure 6.7). This could be due to applying non-casual filter to smooth the new trajectory at the expense of not countering any unwanted dynamics or could be due to the controller not converging to exactly 0 normalised error which requires an infinite number of iterations. Despite this issue, the gripper can grasp the object firmly thanks to the 20% margin given and the resultant force applied is still identical to the desired force shown in Figure 6.21c.



a) Left gripper force plot

b) Right gripper force plot

c) Sum and average force of left and right gripper

Figure 6.21: Gripper forces at 40th iteration

6.2.3 Updating Controller Model

The gantry robot aims to grasp objects ranging from 0.2kg to 2kg. Investigation into the effect of not updating the plant with additional mass will be carried out at the upper and lower bounds as the effect

of the rest would lie in between these two bounds. The plant is updated by recalculating the transfer function using (4.7) after considering the additional object mass.

Since the Z-axis (3kg) and gripper (1kg) are much lighter compared to the X (47.7kg) and Y-axis (21kg) [37]; the additional mass could cause these system dynamics to change significantly. Figure 6.22 shows the grasping force experienced by grasping a 2kg object for cases where the plant is updated and not updated. The force feedback can be seen to be oscillating for non-updated plant which is the effect of an extreme mismatch between plant and actual system, causing the next iteration input vector to be significantly incorrect.

Figure 6.23 shows there is little to no effect of updating the plant for a 0.2kg case as the mass is too small to cause significant mismatch.

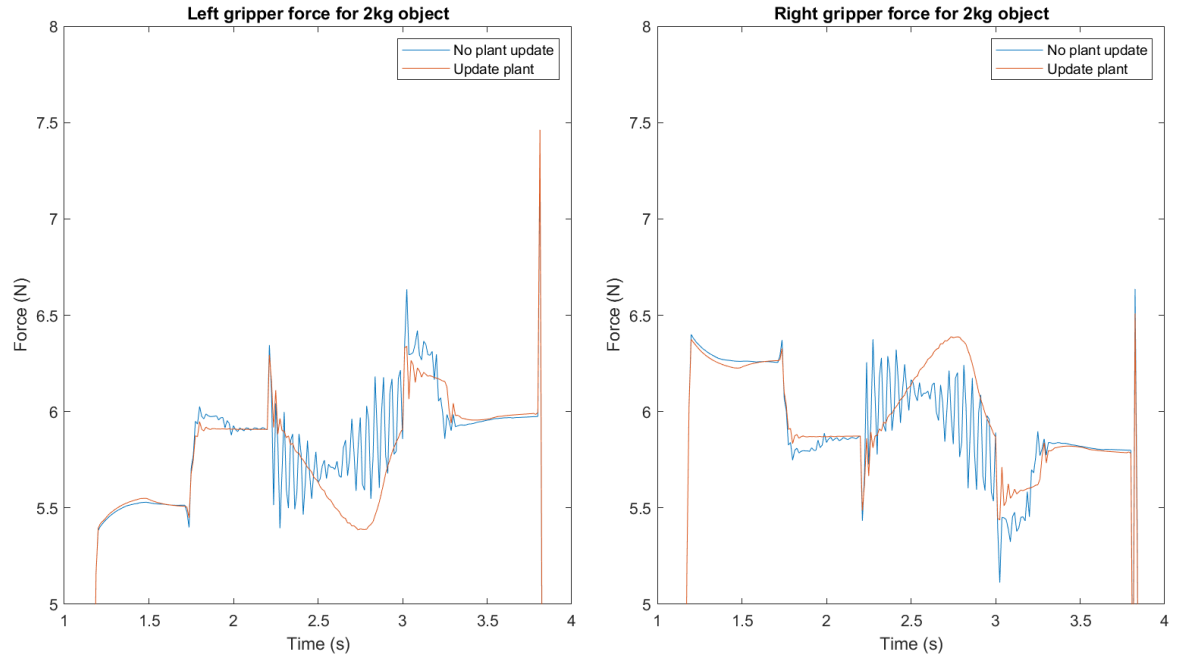


Figure 6.22: Forces experienced by the object for scenarios where the plant is updated and not updated with the object mass of 2kg

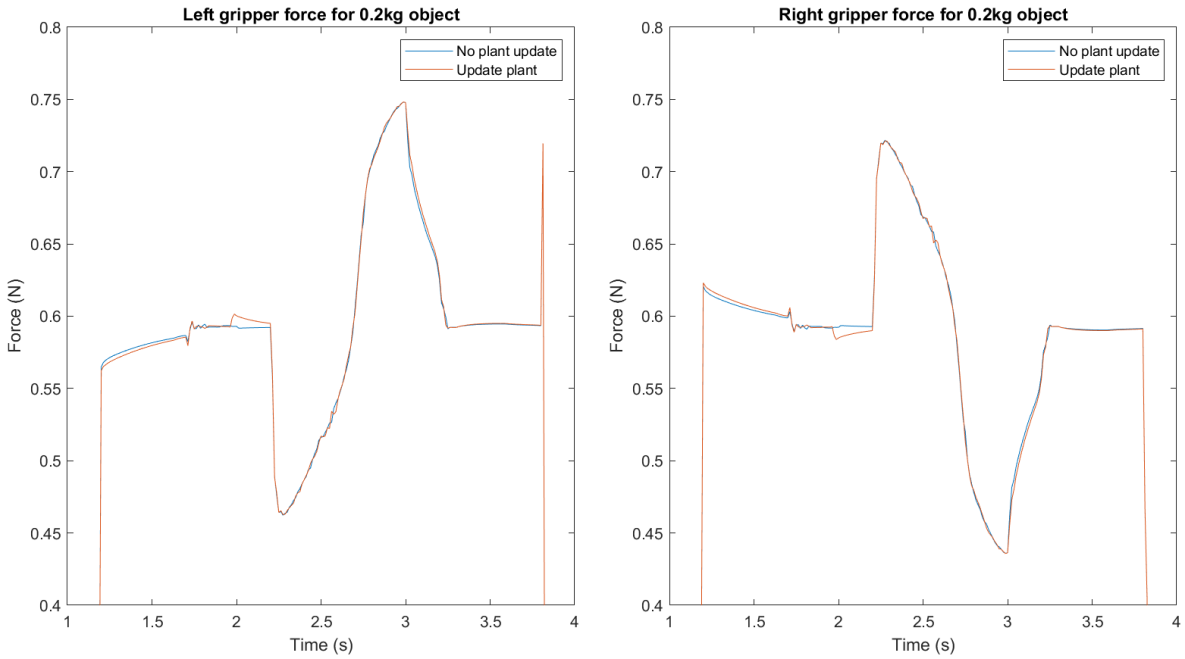


Figure 6.23: Forces experienced by the object for scenarios where the plant is updated and not updated with the object mass of 0.2kg

Figure 6.24 shows the effect of not updating the Z-axis plant model. Similar to the force scenario, there are significant consequences for not updating the model with a 2kg object as the Z-axis vibrates, essentially shaking the object and causing it to slide. This phenomenon can be seen visually in Figure 6.25 as the object starts to slide.

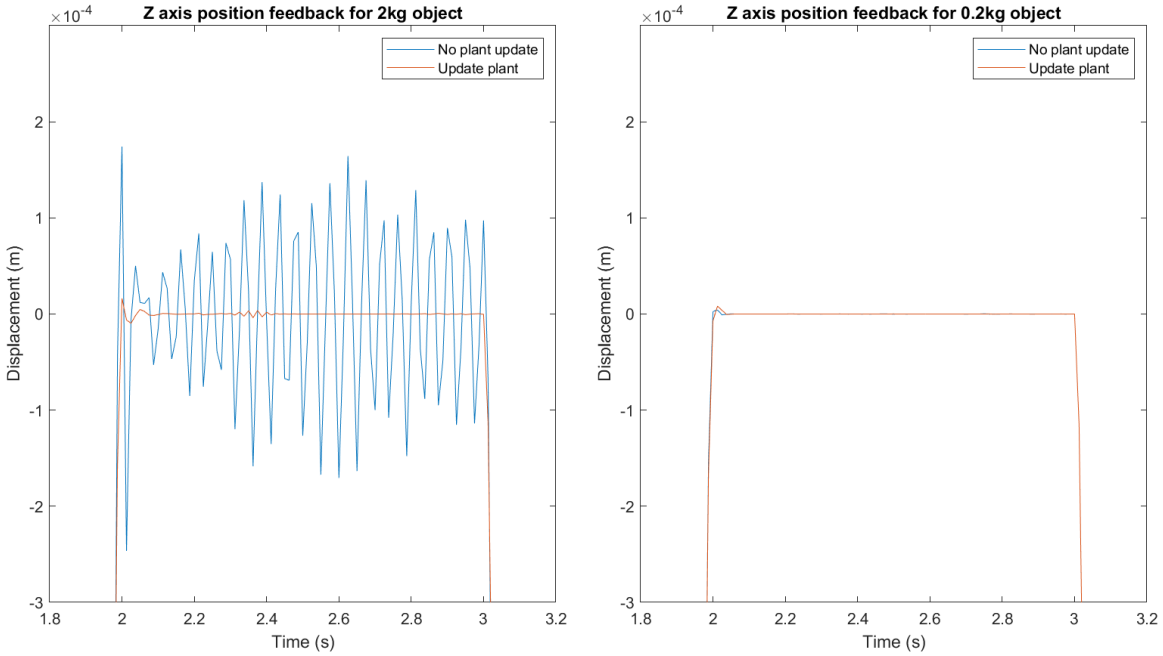
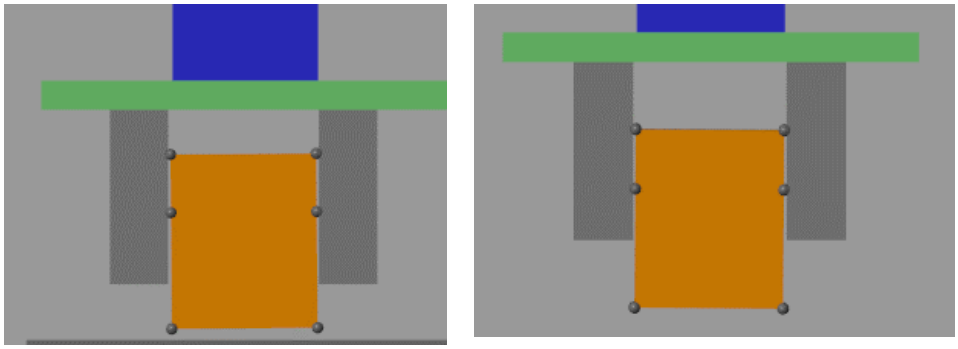


Figure 6.24: Z-axis position feedback for 0.2kg and 2kg objects for updated and non-updated plant



a) At 1.75s b) At 3.15s

Figure 6.25: 2kg object sliding when being transported using a non-updated ILC controller. Both figures are on the same scale.

6.3 Summary

Both Impedance and ILC controllers can grasp objects of mass ranging from 0.2kg to 2kg with little excessive force and fulfil DR4 to DR7. However, the disadvantage of Impedance control as shown in Figure 6.7 is the huge initial spike which could cause damage to the goods. ILC can eliminate this spike at the expense of taking more iterations to converge. The force feedback of ILC (Figure 6.21) is relatively 'bumpy' compared to the impedance counterpart (Figure 6.7), yet it is still able to grasp the object securely. As ILC uses model-based algorithm, it is crucial to update the plant to account for the object mass to a massive mismatch between plant and system which could result in oscillating behaviour causing the gripper to drop the object.

7 Controller Disturbance Test

With the completion of controller design for gantry robot and gripper. It is now crucial to evaluate the robustness of ILC controlled gripper versus its impedance counterpart. This can be done by introducing commonly disturbances encountered daily by a typical system. These disturbances are discussed in Section 2.1.

7.1 Test Setup

The high-level schematic for injecting disturbances into the system is discussed in Section 3.4. The two scenarios (DS1 and DS2) can be further split into three scenarios, namely

1. Constant disturbance added to the 1st iteration mimicking initial error.
2. Constant disturbance added to the 30th iteration mimicking error occurring during operation.
3. Impulse disturbance shows the effect of sudden, short forces interacting with the system.

The highest input signal amplitude under ideal conditions is shown in Figure 7.1. From it, the 5% and 10% values will be taken as the magnitude of disturbances, these values are shown in Table 7.1. An example of impulse disturbance is shown in Figure 7.2 with short rise and fall values found in Table 7.1.

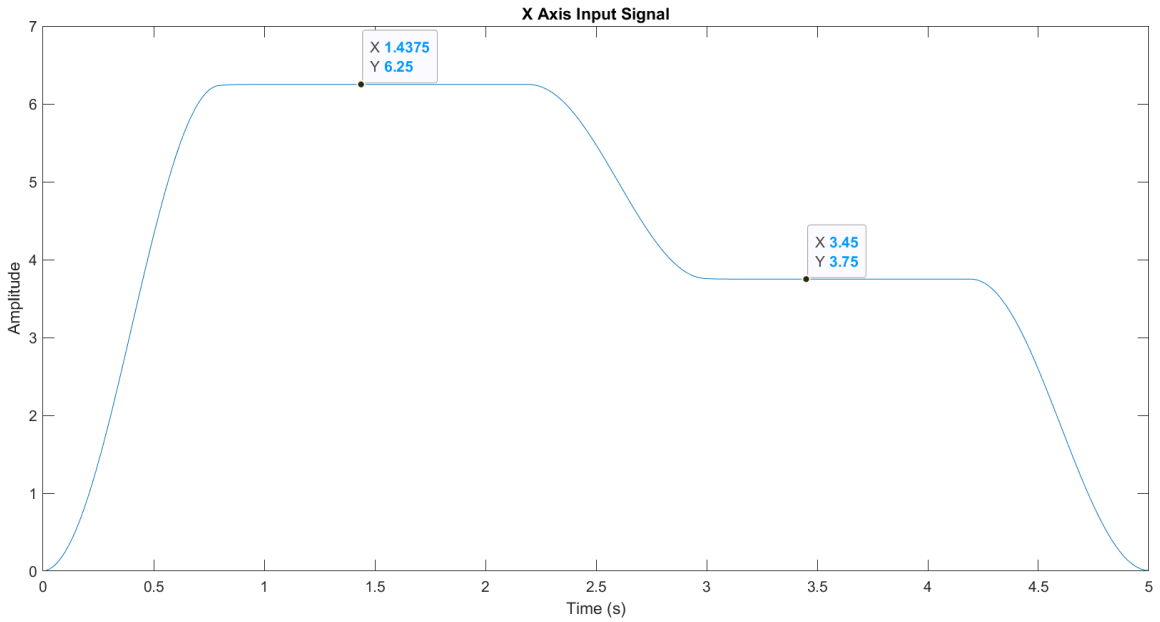


Figure 7.1: X-axis input signal at 40th iteration under no disturbance scenario

Table 7.1: Values used to generate both constant and impulse disturbance for each part of the gantry system

	Constant error				Impulse error			
	X-axis	**Y-axis**	**Z-axis**	**Gripper**	**X-axis**	**Y-axis**	**Z-axis**	**Gripper**
5% amplitude	0.3125	7.5	-2.87	1.2	0.3125	7.5	-2.87	1.2
10% amplitude	0.625	15	-5.74	2.4	0.625	15	-5.74	2.4
Rise time	-	-	-	-	0.2	0.2	0.2	0.1
Deadtime	-	-	-	-	0.2	0.2	0.2	0.2
Fall time	-	-	-	-	0.2	0.2	0.2	0.1

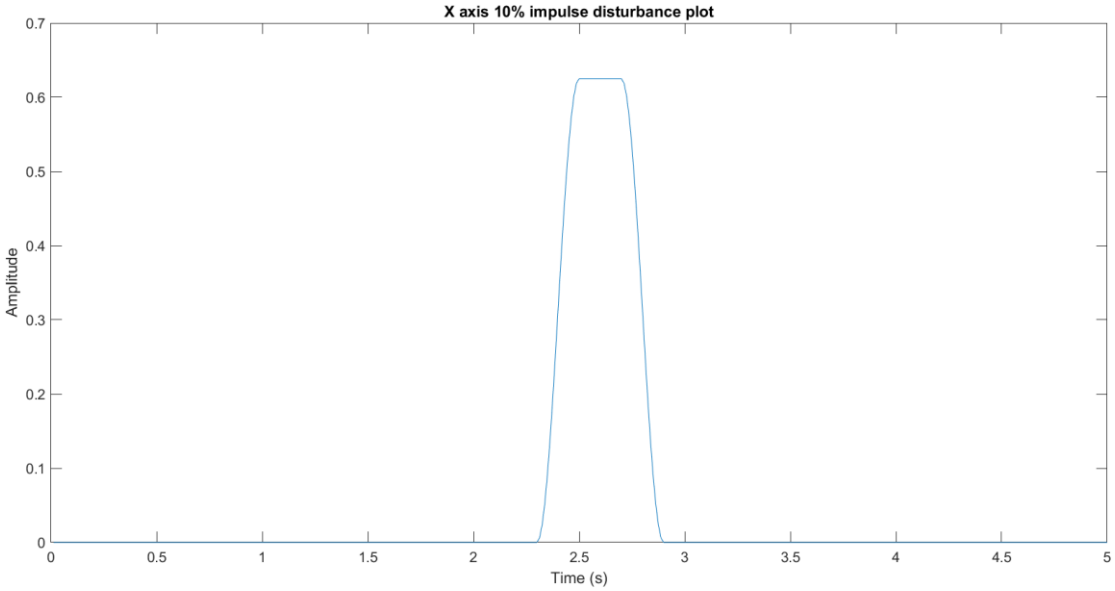
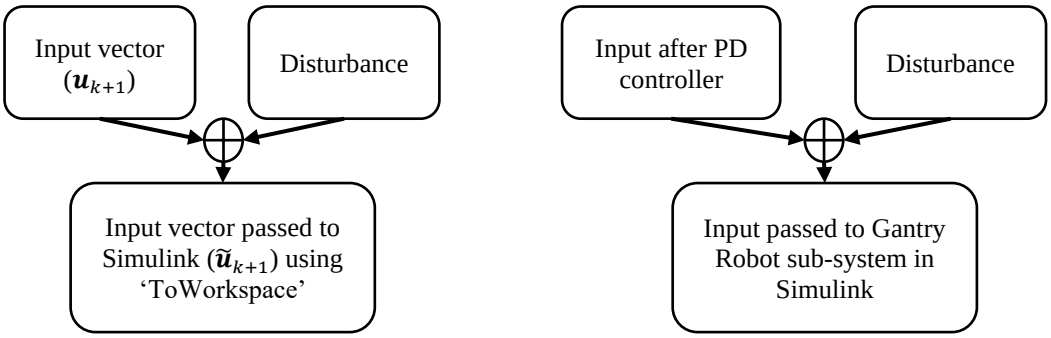


Figure 7.2: Example of Impulse disturbance, where in this case it is 10% for X-axis

For ILC based controllers, disturbances will be added to the input in MATLAB workspace before passing the value to Simulink using the schematic shown in Figure 7.3a, while for impedance-based controller, disturbances are added in Simulink before passing the value as input to gantry sub-system using schematic shown in Figure 7.3b.



a) Schematic showing disturbance added to ILC controller input

b) Schematic showing disturbance added to the signal prior passing to gantry sub-system

Figure 7.3: Schematic for adding disturbance to ILC and Impedance based controllers

The following sections will investigate and analyse the outcome of the controllers when subjected to these disturbances.

7.2 Constant Disturbance

This disturbance will be present during the whole 5s operation of the system.

7.2.1 Disturbance Occurring at 1st Iteration

Table 7.2 highlights the observations after the system is subjected to disturbances using various graphs such as normalised error, force feedback, etc and visual inspection of the simulation in Simulink.

Table 7.2: Observation of the behaviour of gripper and effect part of the gantry system under constant disturbance of various magnitude occurred at the 1st iteration

Remarks	ILC				Impedance
	X-axis	Y-axis	Z-axis	Gripper	Gripper
5%	• Normalised error same as ideal at 4th iteration • Object grasped at 8th iteration	• Normalised error same as ideal at 4th iteration • Object grasped at 8th iteration	• Normalised error same as ideal at 6th iteration • Object grasped at 8th iteration	• Normalised error same as ideal at 7th iteration • Object grasped at 8th iteration	• Behaves identically to Ideal case
10%	• Normalised error same as ideal at 4th iteration • Object grasped at 8th iteration	• Normalised error same as ideal at 4th iteration • Object grasped at 8th iteration	• Normalised error same as ideal at 6th iteration • Object grasped at 8th iteration	• Normalised error same as ideal at 7th iteration • Object grasped at 8th iteration	• Behaves identically to Ideal case
Ideal	• Object firmly grasped at 8th iteration				• Object firmly grasped at 7th iteration
	• Gripper starts grasping object at 7th iteration • Z axis starts operating at the 5th iteration				

Recall that X, Y, Z-axis, and gripper controller converges quickly as discussed in Chapter 5 and 6. Therefore, any initial error occurring at the 1st iteration will have been corrected in a few iterations. This can be seen in Figure 7.4 where the normalised error between ideal and disturbed scenarios are similar at the 4th iteration for X-axis. Figure 7.5 shows that the effect of disturbance is quickly corrected in just one iteration. This concludes that initial error does not hinder the performance of the gripper.

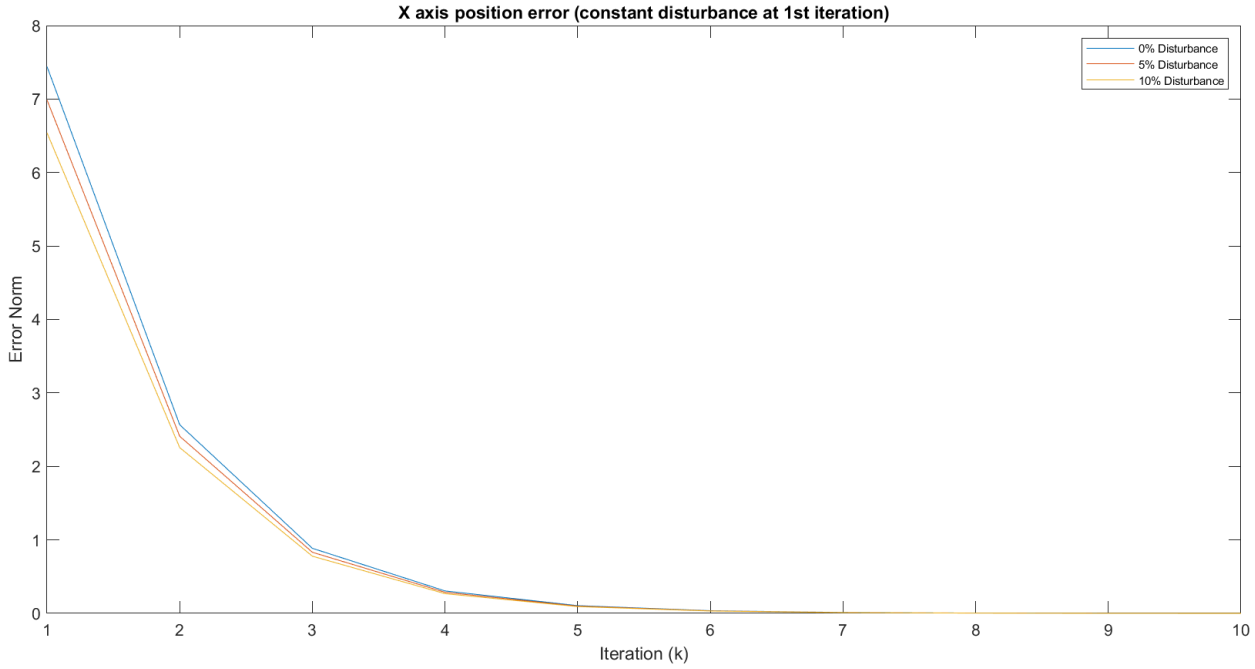
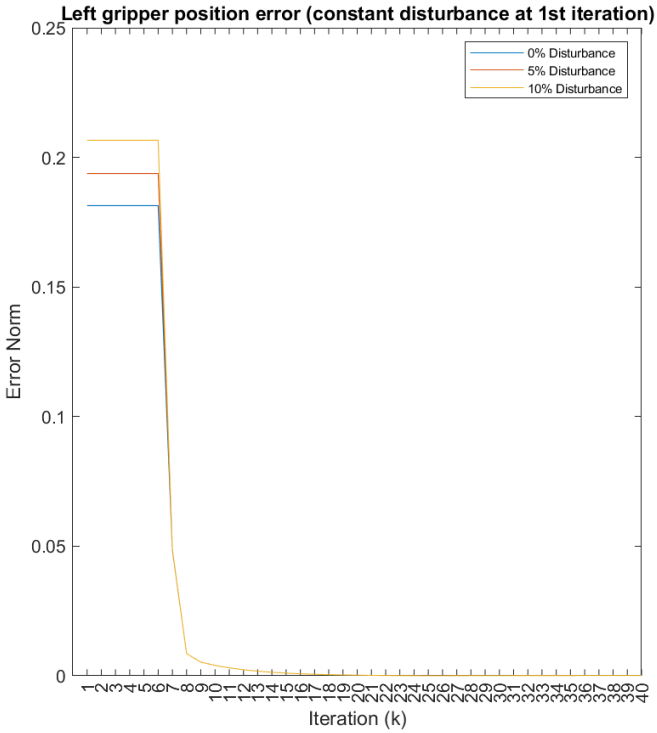
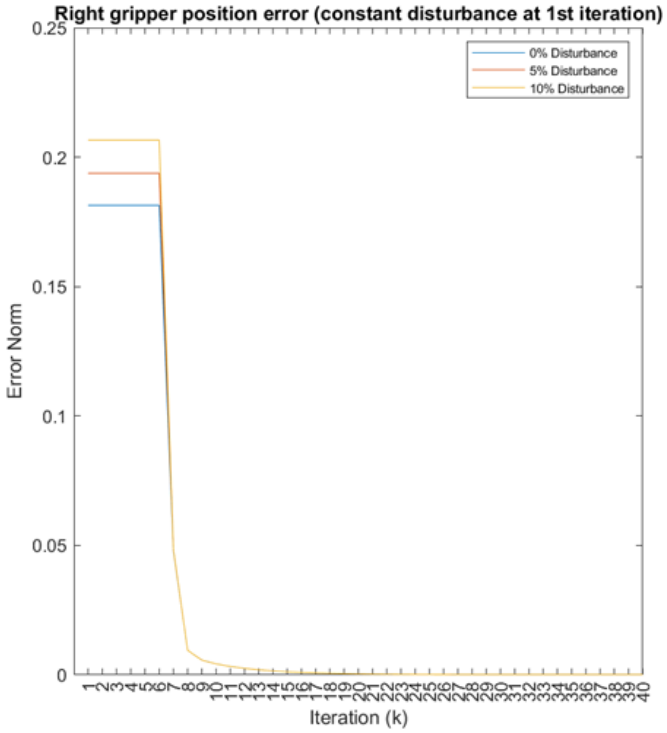


Figure 7.4: Normalised error of X-axis for case of ideal, 5% error and 10% constant error occurring at the 1st iteration



a) Left gripper normalised error for different magnitude of disturbance



b) Right gripper normalised error for different magnitude of disturbance

Figure 7.5: Normalised error for gripper when subjected to different magnitude of constant disturbance occurring at the 1st iteration using ILC controller

7.2.2 Disturbance Occurring at 30th Iteration

A simple logic system is included to only inject disturbance at the 30th iteration. 30th iteration is chosen because from Figure 6.19, the force normalised error converges to minimal after the 25th iteration, therefore, this would mimic a system which is in operation for a sufficiently long period and is now suffering from the effect of wear and tear.

Table 7.3 shows the observation of Impedance and ILC controlled gripper when part of the gantry system is subjected to a constant disturbance occurring at the 30th iteration.

Table 7.3: Observation of affected axis/part of the gantry system and gripper when each part of the gantry system is subjected to constant disturbance occurring at the 30th iteration

		Observations							
Amplitude	**Iteration**	**Disturbance on X-axis**		**Disturbance on Y-axis**		**Disturbance on Z-axis**		**Disturbance on Gripper**	
		Impedance gripper	**ILC Gripper**	**Impedance gripper**	**ILC Gripper**	**Impedance gripper**	**ILC Gripper**	**Impedance gripper**	**ILC Gripper**
5%	30th	• Overshoot grasping volume • Attempted but failed to grasp	• Overshoot grasping volume • Attempted but failed to grasp	• Overshoot grasping volume • Attempted but failed to grasp	• Overshoot grasping volume • Attempted but failed to grasp	• Same as ideal operation	• Same as ideal operation	• Stable grasp	• Failed grasp
	31st	• Overshoot grasping volume • Unstable grasp	• Overshoot slightly • Success grasp (off centre)	• Overshoot slightly • Success grasp, slid slightly	• Overshoot slightly • Object 'push' before grasping				• Failed grasp
	32nd	• Same as ideal operation	• Overshoot slightly • Success grasp (off centre)	• Same as ideal operation	• Same as ideal operation				• Initial success grasp, drop during transportation
	33rd		• Same as ideal operation						• Success grasp, slid slightly
	34th								• Same as ideal operation
10%	30th	• Overshoot grasping volume • Failed grasp	• Overshoot grasping volume • Failed grasp	• Overshoot grasping volume. • Failed grasp	• Overshoot grasping volume. • Failed grasp	• Same as ideal operation	• Same as ideal operation	• Success grasp • Slid slightly.	• Failed grasp
	31st	• Overshoot slightly • Attempted but failed to grasp	• Overshoot slightly • Attempted but failed to grasp	• Overshoot slightly • Failed grasping	• Overshoot slightly • Unstable grasping				• Failed grasp
	32nd	• Overshoot slightly • Success grasp (off centre)	• Overshoot slightly • Success grasp (off centre)	• Overshoot slightly • Stable grasp	• Overshoot slightly • More stable grasp				• Failed grasp
	33rd	• Overshoot slightly • Success grasp (off centre)	• Overshoot slightly • Success grasp (off centre)	• Same as ideal operation	• Overshoot slightly • More stable grasp				• Failed grasp
	34th	• Same as ideal operation	• Same as ideal operation		• Same as ideal operation				• Failed grasp
	35th								• Initial success grasp, drop in transportation
	36th								• Success grasp, slid slightly
	37th								• Same as ideal operation

The observations are supported by figures similar to the one shown in Figure 7.6 where the effect of disturbance is reduced to a minimal at roughly 34th iteration despite one having double the magnitude. From these observations, it can be concluded if disturbances occurred at X, Y or Z-axis, both impedance and ILC gripper converges at roughly the same iteration as X, Y and Z-axis controller

converges. Figure 7.7 shows an example of the position feedback received and the corresponding changes to the input signal to counter the disturbance.

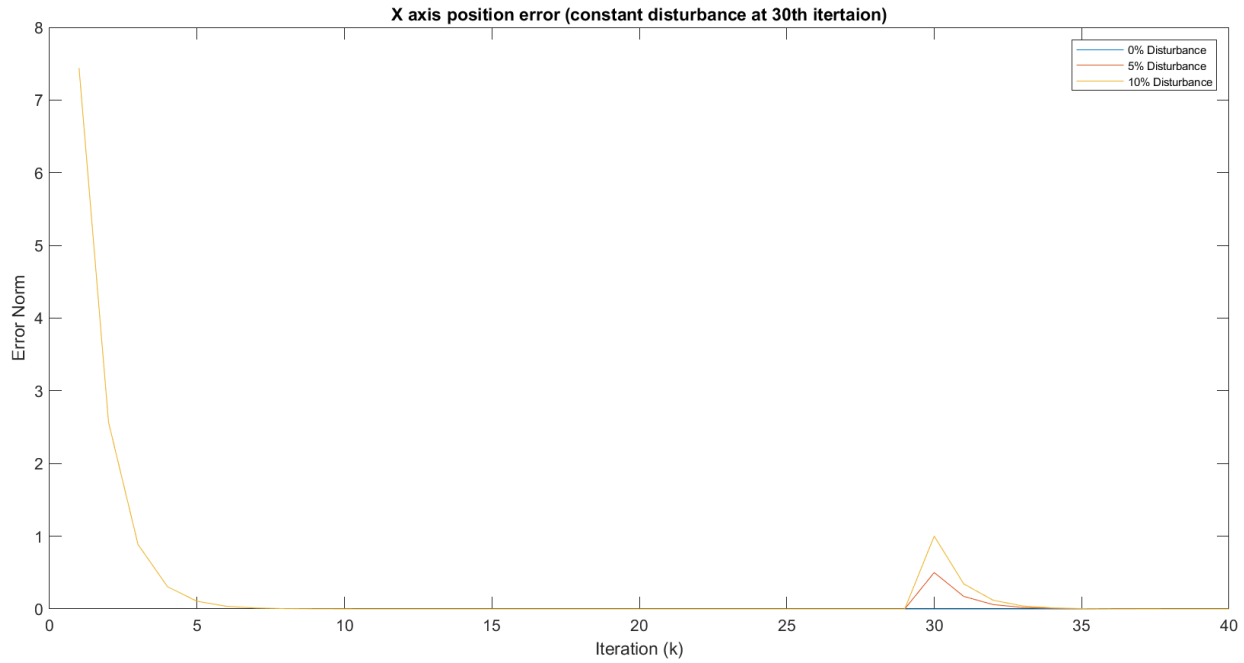


Figure 7.6: X-axis normalised position error for case of no disturbance, constant 5% and 10% amplitude disturbance occurring at 30th iteration

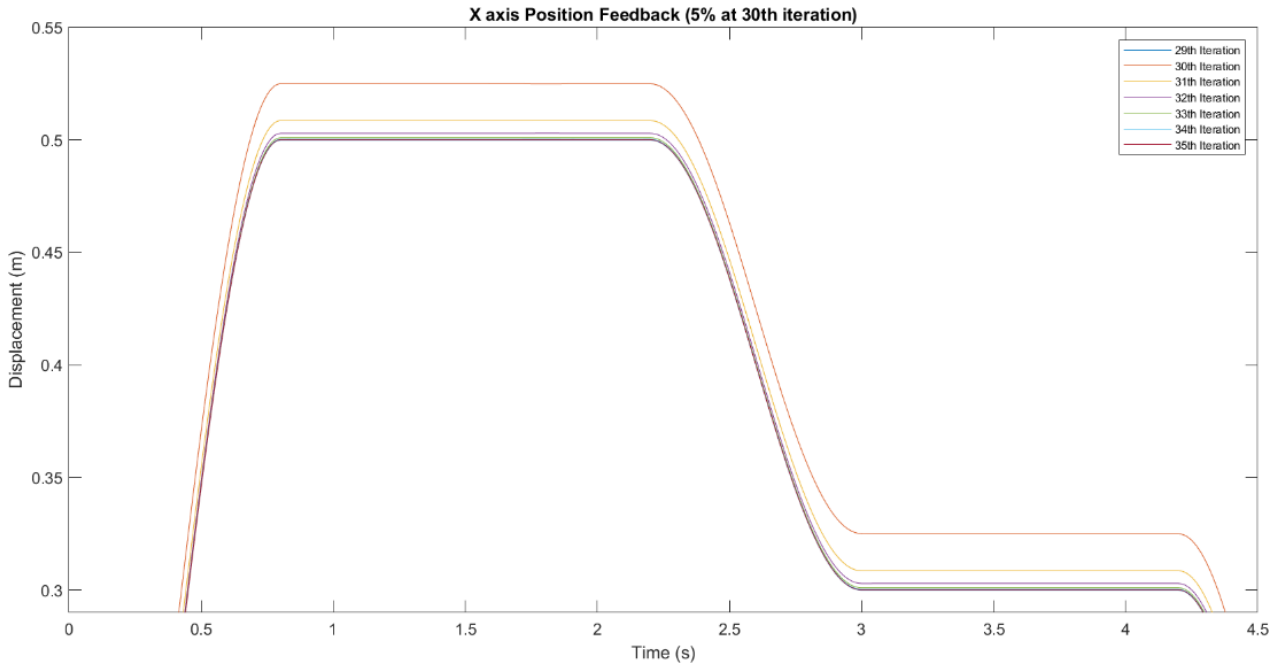


Figure 7.7a: X-axis input signal for 5% disturbance occurring at the 30th iteration

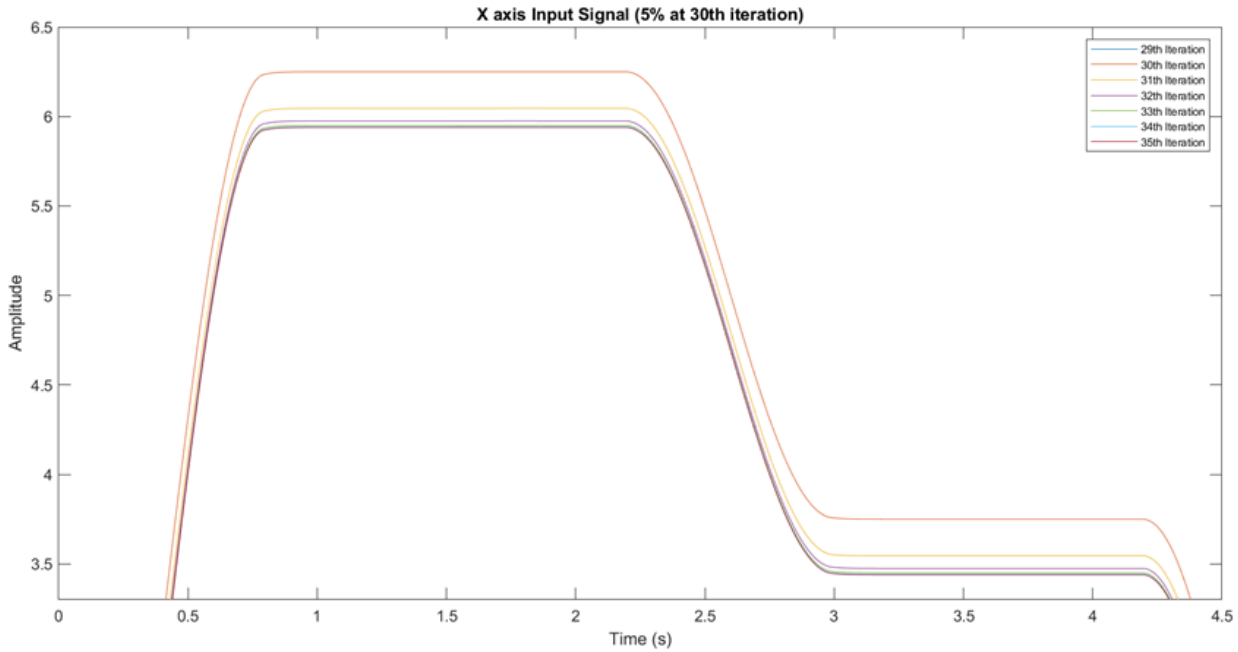
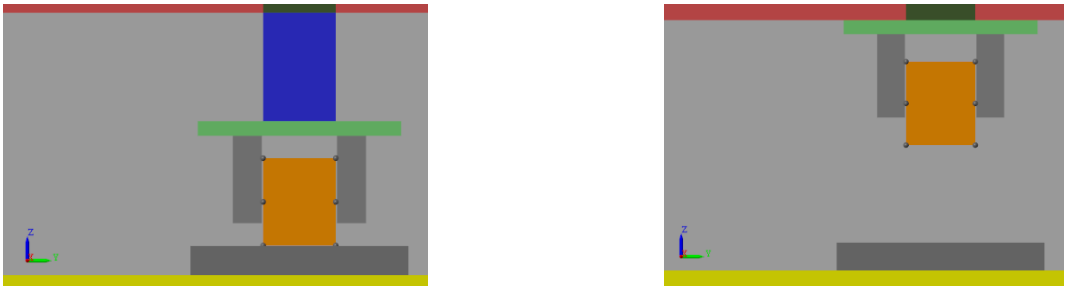


Figure 7.7b: X-axis position feedback for 5% disturbance occurring at the 30th iteration

As seen in Table 7.3, the impedance-controlled gripper always is capable of grasping the object, whereas the ILC controlled gripper takes a few iterations before it can do so. However, since impedance control does not learn from previous iterations, for the 10% amplitude disturbance, the object will always slide. ILC on the other hand can grasp the object firmly under the same disturbance after just a few iterations. To capture the sliding effect, tactile sensors are normally placed on the fingers, however, since the model does not include tactile sensors, it is identified visually illustrated in Figure 7.8.



a) Initial grasp (1.25s) b) Object slid (2.2s)

Figure 7.8: Visual simulation of grasping point when impedance-controlled gripper is subjected to 10% constant disturbance (To scale)

Figure 7.9 and Figure 7.10 show the position and force feedback of the ILC grippers when it is subjected to disturbance. At the 30th iteration, the gripper position feedback is lower than the one at the 29th iteration showing disturbance has occurred, this causes the gripper to drop the object as insufficient force is applied resulting in the force feedback to reduce to zero at roughly 1.9s. This causes the next iteration input signal to experience a significant change thus resulting in excessive force used compared to at 29th iteration. After the object has been grasped (32nd iteration), it will start correcting itself and finally reside to a similar applied force seen at the 29th iteration.

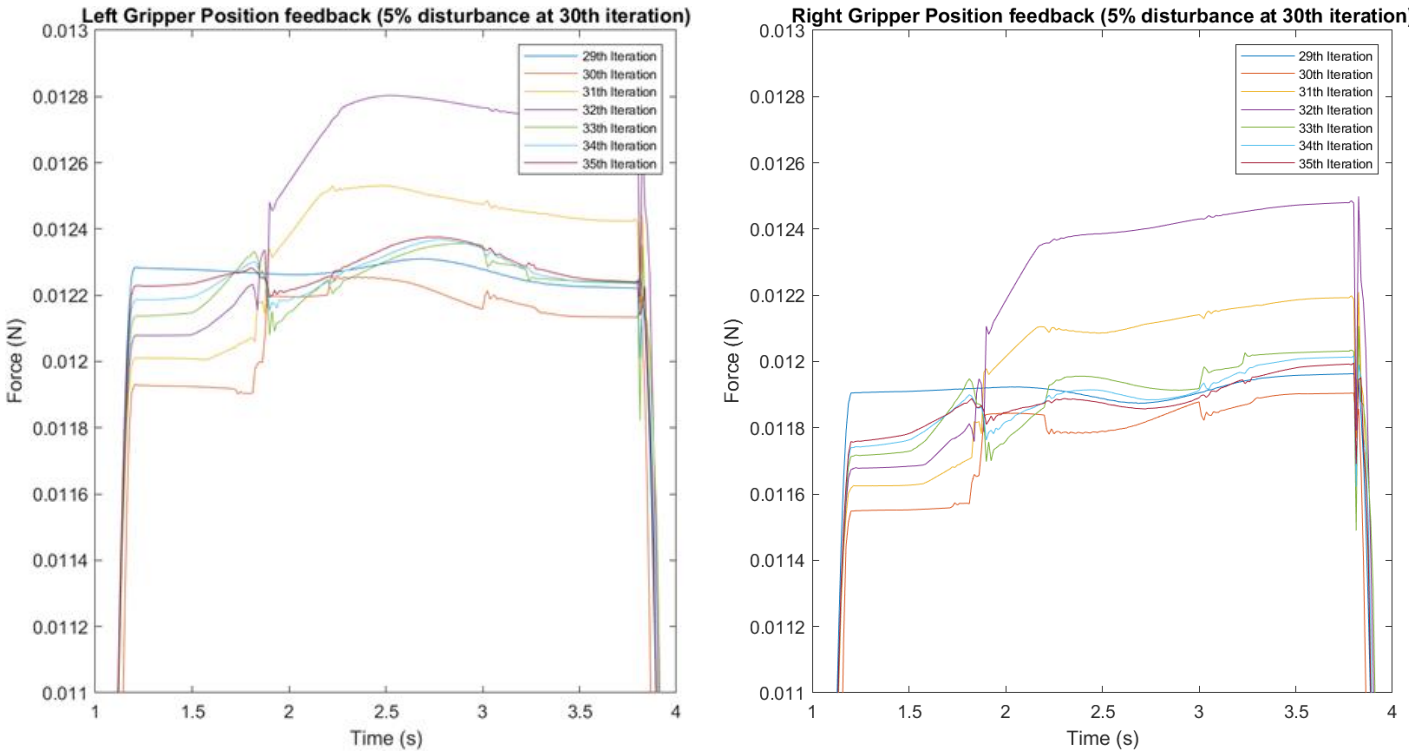


Figure 7.9: Graph showing gripper position feedback when the ILC controlled gripper is subjected to 5% constant disturbance at the 30th iteration

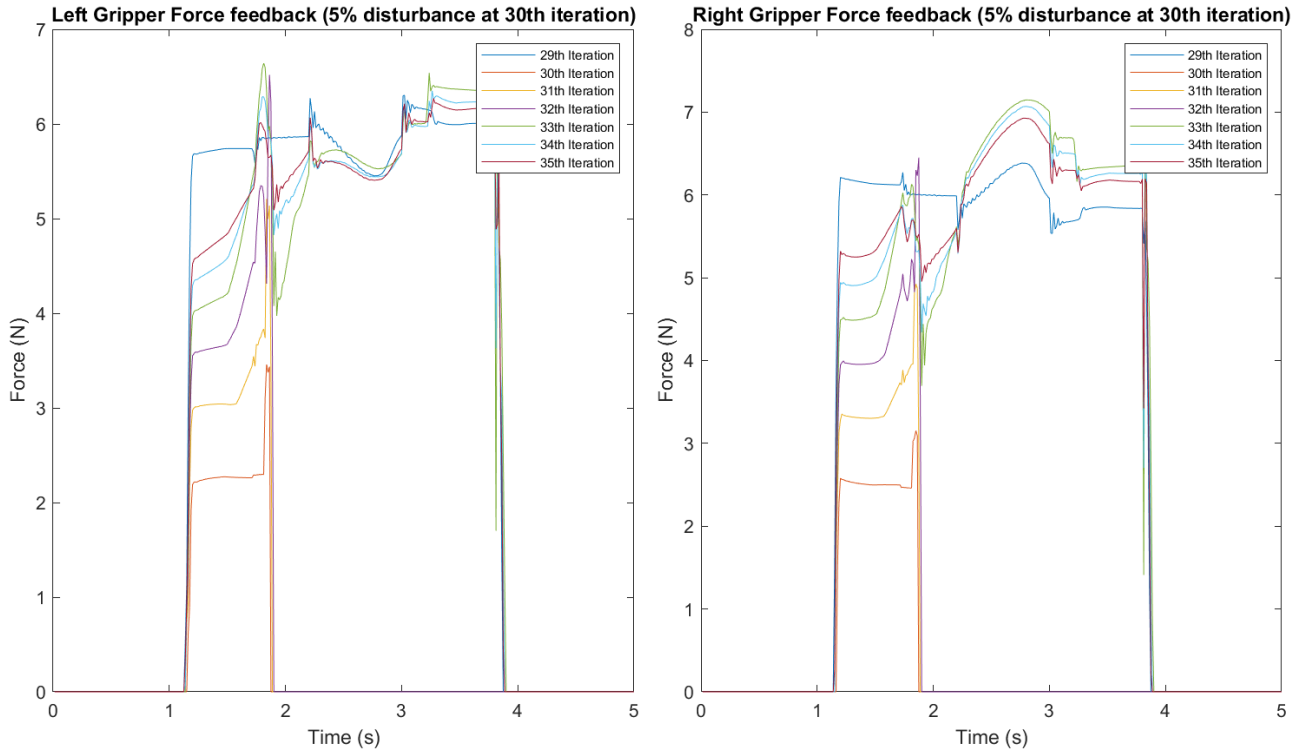


Figure 7.10: Graph showing force applied when the ILC controlled gripper is subjected to 5% constant disturbance at the 30th iteration

7.3 Impulsive Disturbance

Table 7.4 shows the observation when the gantry system is subjected to impulse disturbances. Similar to the constant disturbance case, impedance and ILC gripper performance are relatively similar to each other when X, Y and Z-axis are subject to disturbance.

Table 7.4: Observation of affected axis/part of the gantry system and gripper when each part of the gantry system is subjected to impulse disturbance occurring at the 30th iteration

		Observations							
Amplitude	**Iteration**	**Disturbance on X-axis**		**Disturbance on Y-axis**		**Disturbance on Z-axis**		**Disturbance on Gripper**	
		Impedance gripper	**ILC Gripper**	**Impedance gripper**	**ILC Gripper**	**Impedance gripper**	**ILC Gripper**	**Impedance gripper**	**ILC Gripper**
5%	30th	• Failed grasp	• Object slid	• Failed grasp	• Object slid	• Stable grasp	• Stable grasp	• Object slid	• Fail grasp
	31st	• Stable grasp	• Stable grasp	• Stable grasp	• Stable grasp				• Fail grasp
	32nd								• Stable grasp
10%	30th	• Failed grasp	• Failed grasp	• Failed grasp	• Object slid more	• Stable grasp	• Stable grasp	• Failed grasp	• Fail grasp
	31st	• Object slid	• Object slid	• Object slid and dropped	• Object slid slightly				• Fail grasp
	32nd	• Stable grasp	• Stable grasp	• Stable grasp	• Stable grasp				• Fail grasp
	33rd								• Stable grasp

Since X, Y and Z-axes use ILC controllers, the effect caused by the disturbance is minimised in a few iterations by learning by comparing position feedback between the 30th iteration and 35th iteration as shown in Figure 7.11a. Figure 7.11b shows the resultant changes to the input signal for X-axis to overcome the disturbance. Y and Z-axis can be analysed similarly.

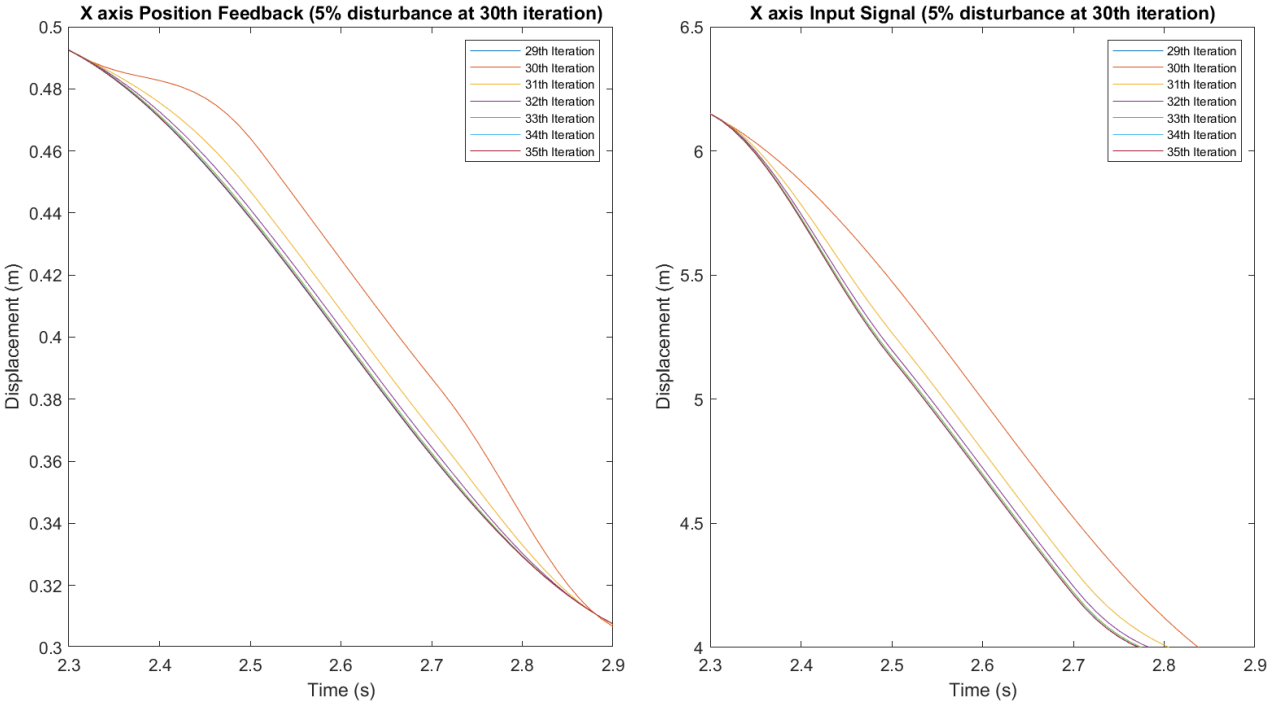


Figure 7.11: Adaptation of ILC algorithm on X-axis position feedback and input signal when subjected to 5% Impulse disturbance

The corresponding force and position feedback of impedance and ILC controlled gripper when X-axis is subjected to 5% impulse disturbance are shown in Figure 7.12 and Figure 7.13. For impedance gripper, it can be seen that at the 30th iteration, the force drops to roughly 0 around 2.6s indicating that the object has been dropped, in line with the observation in Table 7.4. During this time, the impedance controller does not know that the object has been dropped hence it assumes that there is insufficient grasping force, resulting in it closing the gripper further causing a huge position spike.

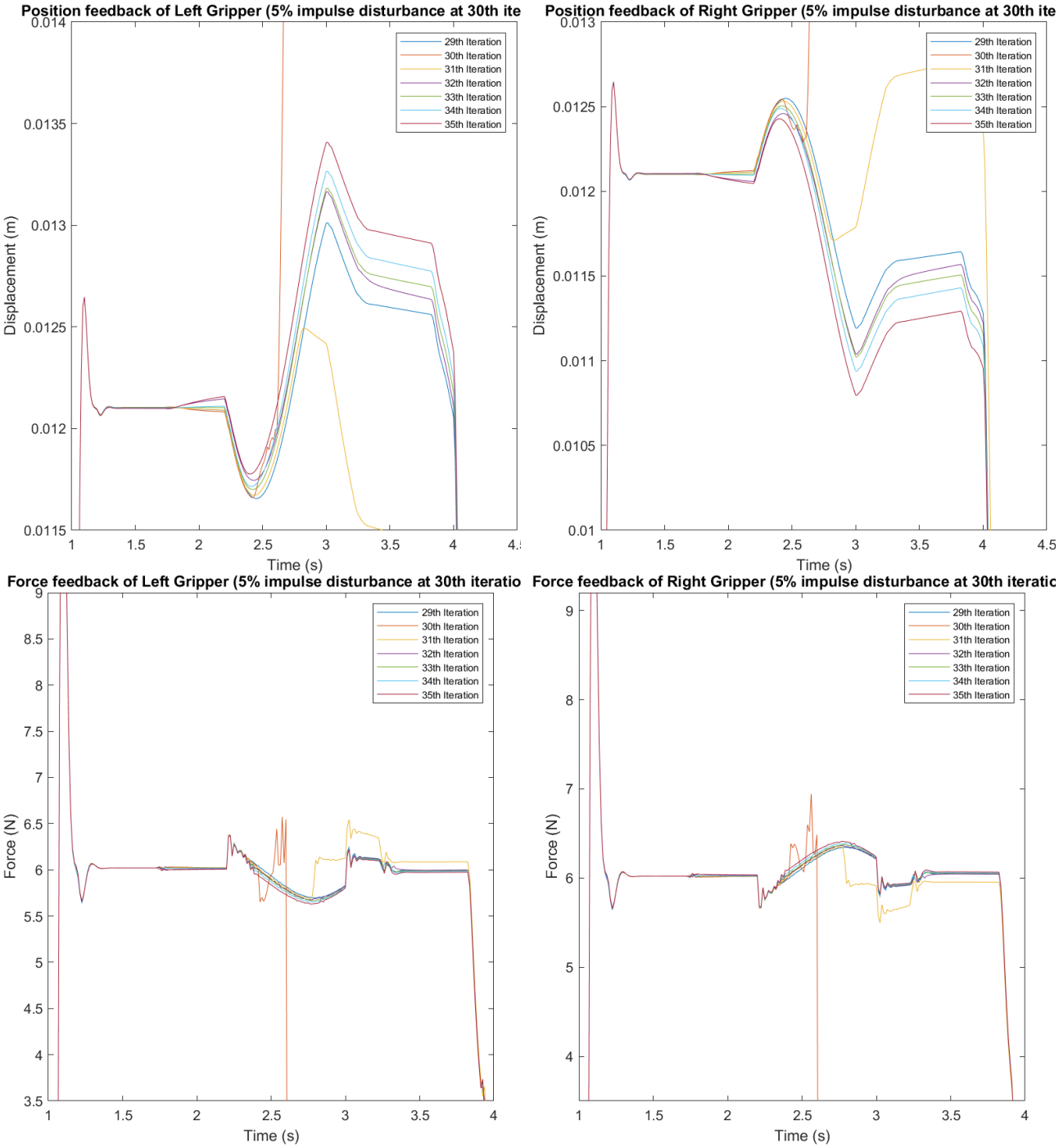


Figure 7.12: Impedance controlled gripper position feedback and force plots when X-axis is subjected to a 5% impulse disturbance

The ILC gripper on the other hand does not drop the object when X-axis is subjected to 5% impulse disturbance but rather slips. Due to the lack of tactile sensors, it is not possible to identify slippage, but the force plot seen in Figure 7.13 shows that there is a minor spike, indicating something has happened and the controller can correct grasping profile accordingly.

Both grippers perform well when the Z-axis is subjected to impulse disturbance. This is because when Z-axis encounters impulse disturbance, it is similar to shaking the object vertically and the 20% margin helps to mitigate these effects.

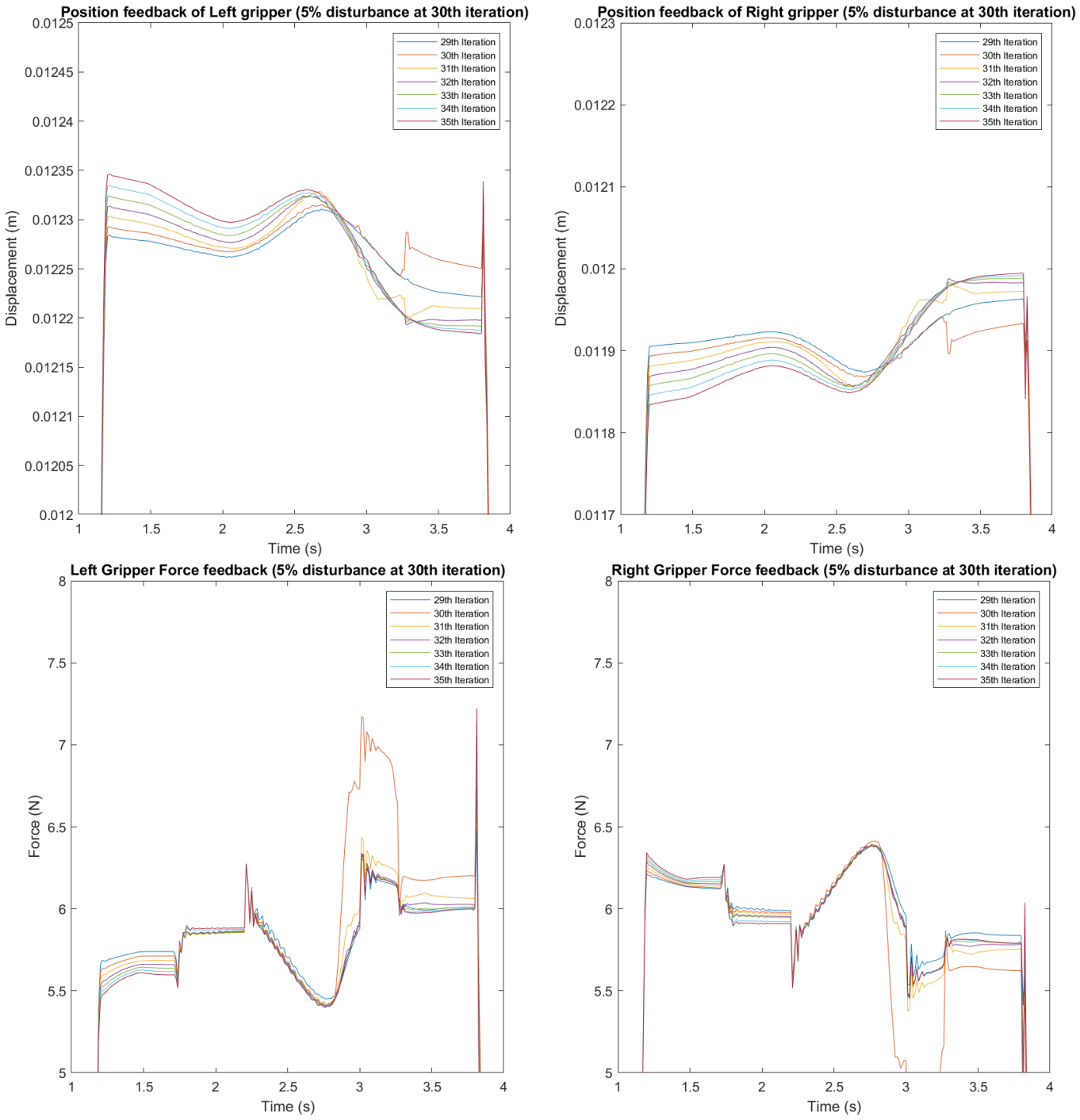
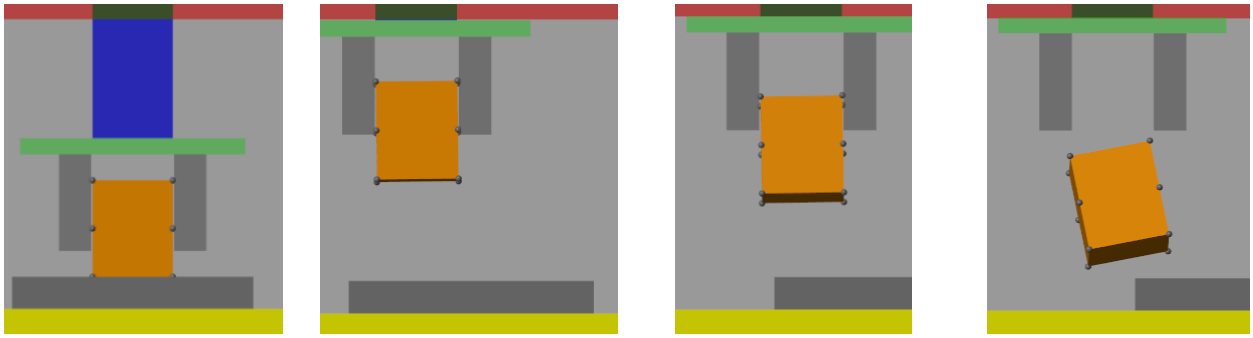


Figure 7.13: ILC controlled gripper position feedback and force plots when X-axis is subjected to 5% impulse disturbance

Since impedance does not learn from previous iterations, once the gripper becomes unstable and drops the object, it will never recover as highlighted in Table 7.4. Figure 7.14 shows the sequence of events visually. ILC controller on the other hand initially performs worse by dropping the object, but after 2-3 iterations, it converges and can grasp the object securely.



a) 1.25s b) 2.35s c) 2.4s d) 2.45s

Figure 7.14: Sequence of gripper using impedance controller subjected to 10% impulse disturbance

The effect of impulse disturbance on ILC gripper can be seen at the 30th iteration in Figure 7.15. The position feedback of the 29th and 35th iterations is almost identical, showcasing the ability of ILC to correct repetitive disturbance. The force experienced by the object is also shown in Figure 7.16. Recalling that if the force feedback drops to 0, the object is dropped.

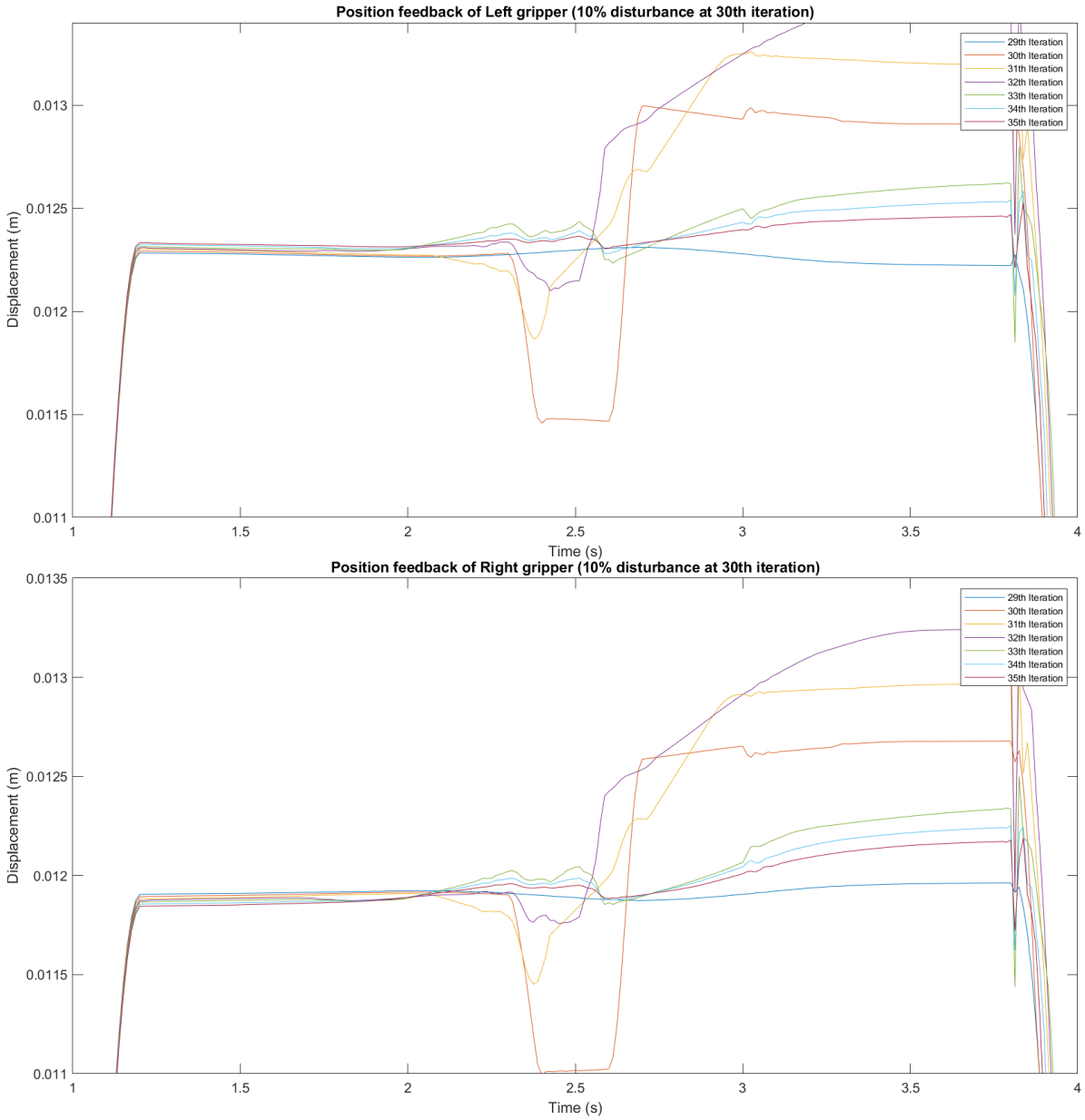


Figure 7.15: Gripper position feedback when subjected to 10% impulse disturbance

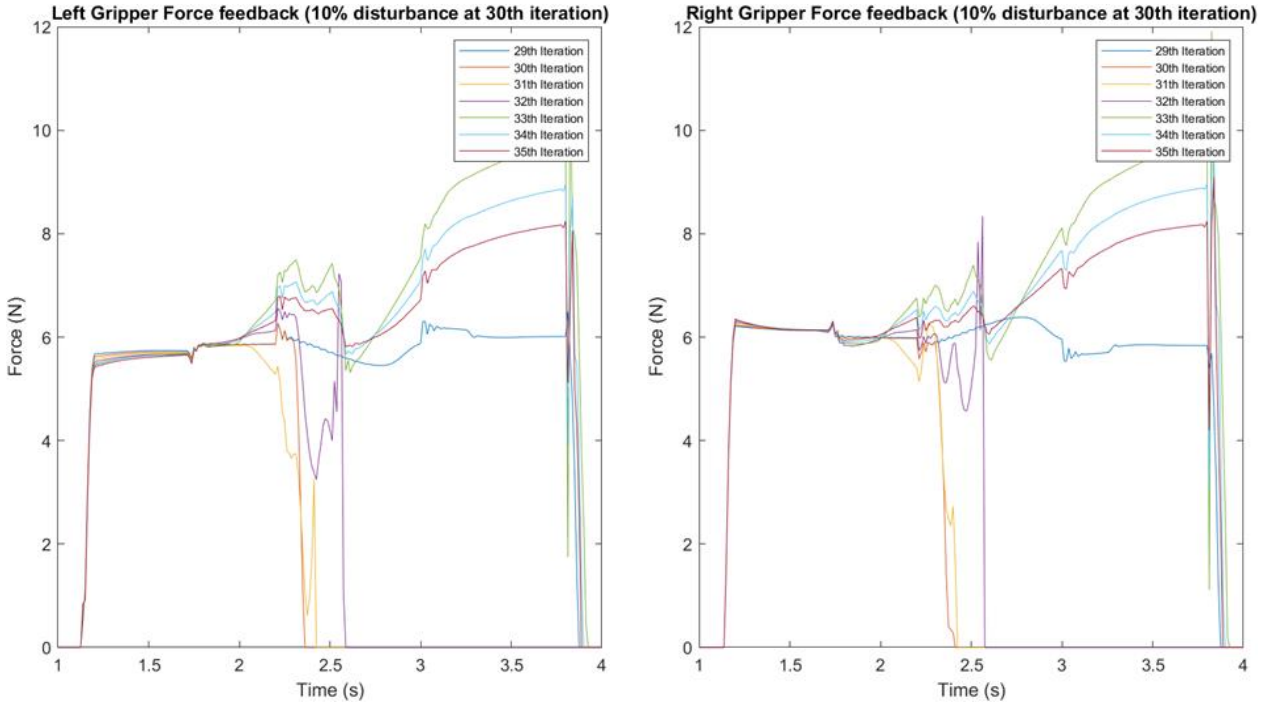


Figure 7.16: Force experienced when the gripper is subjected to 10% impulse disturbance

7.4 Summary

Both impedance and ILC controlled gripper performance relatively similarly when the gantry system is subjected to constant and impulse disturbances. Particularly, due to the additional 20% force margin specified in the control algorithm, Z-axis disturbance does not cause any issue to the gripper's ability to grasp objects securely.

The advantage of ILC over impedance control is its learning ability. As demonstrated with impulse disturbance, impedance-controlled gripper has a limit to how large disturbance it can be subjected to before dropping the object indefinitely. Although ILC gripper initially dropped the object, after a few iterations, it can correct itself and grasp the object securely.

Unfortunately, both controllers are not able to detect slippage and correct them accordingly.

8 Conclusions and Future Work

A representative industrial gantry robot and the interaction forces between robot and environment have been modelled in MATLAB Simulink using Simscape Multibody toolbox.

Upon successfully fulfilling the design requirements set out in Chapter 2, the new control algorithm is capable of grasping objects of different mass, completing the whole reach and grasp operation in 5 seconds, converging quickly upon startup, and mitigating effects caused by commonly encountered disturbances while using less than 20% excessive grasping force. To fully evaluate the success of the project, the design specification is compared against the result and analysis conducted in Chapters 4, 5, 6 and 7.

The new ILC control algorithm is capable of reducing initial force spike and minimising the effect of repetitive disturbances which the impedance control is unable.

8.1 Review against Design Requirement

Table 8.1 contains numbered Design Requirements set out previously, a brief description of the aim and how well it is met out of 10.

Table 8.1: Comparison of design requirement against the ILC controller algorithm and quality of simulation model

Design Requirement	Brief description	Achievement
1	Simulation model relates to industrial robot	7/10
2	Accurate simulation model	9/10
3	Simulation can be seen visually	9/10
4	Gantry robot reaches final location in a few iterations	10/10
5	Fast reach and grasp sequence	10/10
6	Minimal excessive force used	9/10
7	Grasp range of objects	10/10
8	Robust to commonly encountered disturbances	9/10
Total score		73/80

A detailed breakdown of the scores can be found in Table 8.2.

Table 8.2: Detailed breakdown of the scores given in Table 8.1

Design Requirement	Achievement	Additional notes
1	The model selected was a simple 3 axis gantry robot.	More flexible robotic arm robot with multiple directions of freedom (DOF) is gaining popularity in the industry.
2	The simulation model is capable of modelling the dynamics of the system and any interaction force it may encounter.	The chosen gantry robot dates back to 2005, the mass used in the simulation does not reflect the actual mass but an estimation. The gripper fingers are quite large as the contact point is modelled as spheres.
3	The 3 axes of the gantry robot and the gripper can be visually seen, and it is possible to see what is happening during simulation.	Box-like structure is the easiest to implement using Simscape multibody. To improve the visual aspect, 3D models can be generated from elsewhere such as Solidworks and imported in.
4	The gantry robot is able to complete close to ideal reach and grasp action in roughly 10 iterations.	The initial reach and grasp profile is pre-defined by the operator.
5	The whole reach and grasp profile are completed in just 5s under the assumption that the gantry robot is capable of producing the required torque.	Most industrial robot deals with bigger and heavier object and thus the duration of each iteration might increase to reduce stress and strain placed on the system.
6	The gripper uses around 20% more force when grasping an object after it converges.	Initial few iterations might use quite a significant amount of excessive force depending on the initial profile generated for the gripper
7	The gripper is able to safely transport objects ranging from 0.2kg to 2kg	Further research should aim at grasping objects of various shapes and size
8	The gripper is able to correct itself and grasp the object when there is a repetitive disturbance presence.	The gripper might initially drop the object when the disturbance is first encountered.

8.2 Future Work and Research

A few areas that could be improved and investigated are highlighted below.

8.2.1 Hybrid of ILC and Impedance

Impedance controller is great at reducing the amount of excessive force applied to the system whereas ILC is known to remove any repetitive disturbances. Therefore, to get the best of both worlds, it would be worth investigating the overall gain of combining these two algorithms.

8.2.2 Test on Physical System

The project has only focused on evaluating the performance of the new ILC controller in a simulation environment. To fully test the controller, it is necessary to subject the controller to test on the actual gantry robot to fully justify the improvements gained of ILC over Impedance controller.

8.2.3 Increase Complexity of Gripper Hardware

This project uses a basic two parallel plate fingers gripper with force sensors. To avoid objects falling out during operation, it is necessary to detect any slippage that occurs and correct it immediately. This requires multiple tactile sensors to be placed carefully on the fingers.

As identified in Chapter 1, there is multiple more advanced gripper structure. To fully exploit ILC capability and improve grasping, these more advanced grippers shall be used and compare the result with ones using impedance controller.

8.2.4 Improve ILC Algorithm to Grasp Object of Different Shapes and Sizes

Objects in the real world come not only in different masses but also in different shapes and sizes. To be able to grasp different types of objects, some sort of visual sensors should be used. With the help of machine learning, different tracking profiles can be used to grasp different objects. ILC can be used to improve the tracking profiles which theoretically should use less computational resources than machine learning methods.

8.2.5 Research Into Minimisation Equations

The ILC controller algorithm focuses on minimising two cost functions. This could potentially cause the update to conflict with each other. Research should be conducted to find the relationship between forces applied and grasping distance and minimise these two components with one cost function.