

Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Yi Farn Lim
yfl1f18@soton.ac.uk
20/04/2022

Improving Robotic Grasping using Iterative Learning Control

Project supervisor: Professor Christopher Freeman
Second examiner: Doctor Nicolas Green

A project report submitted for the award of
MEng Electrical and Electronics Engineering

Abstract

Over millions of years of evolution, nature has perfected one of the most widely used features of humans. Hands, particularly fingers allow humans to grasp and manipulate everyday object. As such, the environment has been adapted to suits human's operation. Therefore, to allow robots to operate effectively, they have to behave like humans in some way such as able to walk up fleets of stairs, grasp and manipulate object. Many research has been carried out focusing on the hardware aspect of robotic gripper such as flexible, multiple fingers, suction. However, research into the controller side of robotic grasping is limited. This causes most industrial sized robots to operate slowly, with a low mean grasp per hour, only able to operate in a tightly controlled environment while doing specific tasks. To fully exploit the potential of robots assisting human, robotic grasping has to at least be on par with humans capability.

Due to the large potential of robotics, there has been significant investments brought into this area. It is expected that robotic market would experience a growth of 17.45% per annum [2]. This has encouraged further development into control and decision algorithm on selecting grasping task, methods, and performance the task accurately and reliably. This project will focus on designing an Iterative Learning Control (ILC) algorithm to allow the gripper to grasp and transport various objects securely and safely.

Most currently used grasping controllers uses Impedance Control which does not improves itself after each iteration. As grasping is a repetitive action, the performance of the controller can be improved by learning previous iterations errors, similar to a footballer practising shooting repetitively. There has been ONE well known research of robotic grasping utilising simple learning controllers. Therefore, a novel approach is taken to use an advance model-based (Norm-Optimal) ILC Controllers for robotic grasping. To evaluate the effectiveness of this new control algorithm, a simulation environment is designed to replicate the physical system. The performance of the new controller is compared against a conventional Impedance Controller.

This project has achieved strong results and improve robotic grasping in the following areas;

- Built an accurate and realistic simulation model of the actual physical system using MATLAB Simulink with Simscape Multibody toolbox.
- Increase mean grasp per hour.
- Ability to grasp object of different weight.
- Ability to correct itself after encountering numerous common disturbances
- Lower average grasping force compared to conventional Impedance Controller

This project offers the potential of using a low complexity learning controller for a very repetitive operation over conventional non-learning controller.

Statement of Originality

- I have read and understood the [ECS Academic Integrity](#) information and the University's [Academic Integrity Guidance for Students](#).
- I am aware that failure to act in accordance with the [Regulations Governing Academic Integrity](#) may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

Acknowledgements

I would like to acknowledge and thank my supervisor, Professor Christopher Freeman who made this project possible. His guidance and enthusiasm throughout the project stages has truly made it fun and enjoyable.

I would also like to thank my family for their constant support and encouragement during my time at the university as well as welfare check on my mental and physical health.

And finally, thank you to University of Southampton for making this journey special and memorable.

Contents

Abstract	ii
Statement of Originality	iii
Acknowledgements	iv
Contents	v
1 Review of Robotic Gripping Controller	1
1.1 Problem Analysis	1
1.2 Approach to Robotic Grasping	2
1.3 A History of Controllers for Robotic Grasping	4
1.4 Importance of Simulation	5
2 Detailed Specification	6
2.1 Design Requirements	6
2.2 Overview of Iterative Learning Control (ILC)	7
2.2.1 Non-Model based ILC	7
2.2.2 Model Based ILC	8
2.3 Overview of Impedance Control	9
3 Design Overview	11
3.1 Simulation Environment	11
3.2 Gantry Controller	12
3.3 Gripper Controller	13
3.3.1 Impedance Control	13
3.3.2 Iterative Learning Control	14
3.4 Disturbance	15
4 Simulation Model	16
4.1 Gantry Modelling	16
4.1.1 Trial 1	18
4.1.2 Trial 2	20
4.1.3 Trial 3	21
4.2 Contact Modelling	24
4.2.1 Grasping Force Calculation	25
4.2.2 Normal Force while Accelerating	27
5 XYZ Position Controller Design	29
5.1 Trajectory profile generation 1	29
5.1.1 Tuning ILC Controller	31
5.2 Trajectory profile generation 2	32
5.3 Trajectory profile generation 3	34
6 Gripper Controller Design	37
6.1 Impedance Gripper	38
6.1.1 Position Control	38
6.1.2 Force Control	39
6.2 ILC Gripper	41
6.2.1 Position Control	41
6.2.2 Force Control	44
6.2.3 Updating Controller Model	48
6.3 Summary	50
7 Controller Disturbance Test	51
7.1 Test Setup	51
7.2 Constant Error	52

7.2.1	Error at 1 st Iteration.....	52
7.2.2	Error at 30 th Iteration.....	53
7.3	Impulsive Error	56
7.4	Summary	60
8	Conclusions and Future Work	61
8.1	Review against Design Requirement	61
8.2	Future Work and Research.....	62
8.2.1	Hybrid of ILC and Impedance	62
8.2.2	Test on Physical System	63
8.2.3	Increase Complexity of Gripper Hardware.....	63
8.2.4	Improve ILC Algorithm to Grasp Object of Different Shapes and Sizes	63
8.2.5	Research Into Minimisation Equations.....	63
References	64
Appendix A.	Project management.....	67
A1 -	Contingency Planning, Minimising risk, reduce impact.....	67
A2 -	Time and Progress Management.....	67
A3 -	Risk Management Strategy	68
A4 -	Semester One Gantt Chart	69
A5 -	Semester Two Gantt Chart.....	70
Appendix B.	Project Assumptions	71
B1 -	Position and Force Feedback	71
B2 -	Force Applied by Gantry Robot.....	71
Appendix C.	Simulink Model Schematic	72
C1 -	Overview	72
C2 -	Environment.....	74
C3 -	Gantry	74
C4 -	ObjectAndEnvironment	75
C5 -	ObjectAndGripper.....	76
C6 -	Impedance Controller.....	76
Appendix D.	MATLAB Operational Code	78
Appendix E.	Achieve File Overview	79
Appendix F.	Project Brief	80

1 Review of Robotic Gripping Controller

Humans are hard to please, we experience fatigue, emotions, becomes careless while demanding shorter working hours and higher pay. This motivates industry to adopt automation concept which resulted in the Third Industrial Revolution (Industrial 3.0) in the early 1970s [1]. Automation requires the use of computers as well as robots to assist humans in repetitive, predictable tasks. Since robots do not experience fatigue, does not make uncharacteristic mistakes and do not need to be constantly motivated, theoretically they are capable of operating 24 hours a day, 7 days a week while keeping the long-term operating cost to the minimal.

From a simple Roomba vacuum cleaner to bomb defusing robots in the police arsenal to the state-of-the-art robotic arm assembly system found in car manufacturing, the demand for robotics for the use in personal and industry 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 a compound annual growth rate of 17.45%, bringing the value up to USD 74.1 billion by the end of 2026 [2].

Amazon, one of the leading e-commerce giants has gone as far as hosting a yearly competition that challenges the best teams around the world to design and showcase a robotic system that is capable of separating different types of daily object ranging from soft toy to toothbrush and sort them according to the customer's order [3]. Not only that, recently an interview conducted by Tom Scott to Orcardo Grocery warehouse showcases their one of a kind 'Hive system' for online grocery shopping. There are approximately 2300 robots actively working in the Hive in charge of sort the groceries received from the supplier to each individual customer basket ready to be packed and delivered to the users around the UK. It takes as little as 5 hours from receiving the order to delivering the goods to the customer [4].

Although robots have evolved massively since the early 1970s, it is still far from what humans are capable. There are numerous issues that hinders the performance of robots. These issues motivate this research to investigate and attempts to mitigate effect caused by the issues.

1.1 Problem Analysis

The advantageous of having robots to assist humans has only been realised about half a decade ago, but humans have been developing buildings, structures, system 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 [4]. Humans have a few powerful tools at their disposal that 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). Therefore, grasping and ultimately manipulation is one of the major robotics development challenges [3].

If robots have the ability to grasp and manipulate object as freely and effectively, they are able to interact with the world just like an ordinary human. To achieve this, it is crucial to develop both the hardware and control system of a grasper.

One of the most advance grasping control system, 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 need to be realised and thus the controller has to be able to fulfil a range of criteria:

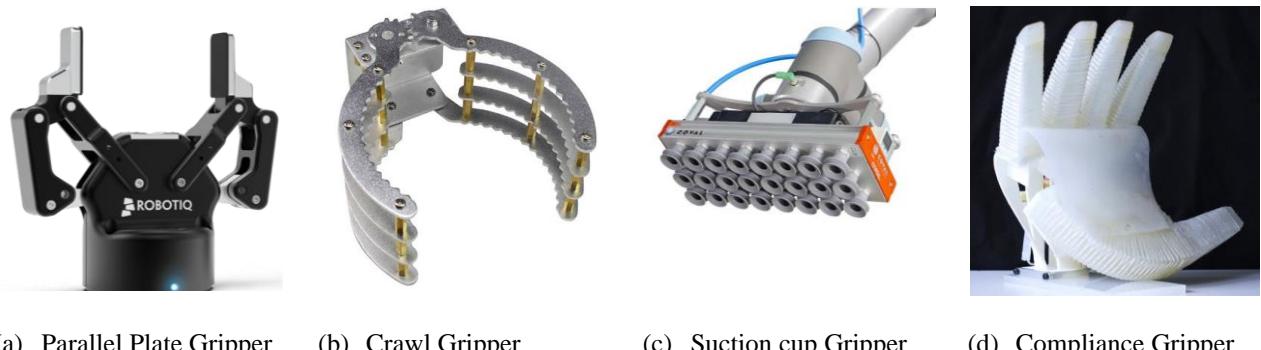
- Learn from previous grasping error/mistakes.
- Able to correct itself from environmental disturbance.
- Ensure the deadtime between each operation is to the minimum.
- Uses least amount of force.
- Capable of grasping a range of objects.

The following section will discuss the generic approach on how robots grasp an object.

1.2 Approach to Robotic Grasping

Robotic grasping problem can be split into 2 categories, Hardware and Control. 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.

Grasping requires the gripper to interact with the environment or object. Over the past decade, different gripper hardware has been developed and tested. All gripper has one feature in common, they have finger-like structure to grasp an object. These fingers, however, comes in various shapes and sizes depending on the task requirement. A simple gripper might employ a parallel plate gripper (Fig. 1.1a) which consists of 2 simple finger that closes. A variation of parallel plate gripper is fingers aligned in a claw structure (Fig. 1.1b). This type of gripper allows the gripper to be in contact with more area of the object reducing the pressure. When delicate objects are handled, the popular choice is to use a suction cup gripper where the suction is controlled by pressurised air (Fig. 1.1c). The motivation of these gripper is similar to the one seen in glass panel installation for buildings, where they are able to grasp the object while ensuring the pressure is spread over a wide area to ensure minimal stress on the object. Recently, the development has shifted to employ ‘softness’ into fingers by exploiting compliance properties in grasping (Fig. 1.1d). The fingers are able to curl and straighten by adjusting the pressurised air sent to the fingers. [3]



(a) Parallel Plate Gripper (b) Crawl Gripper (c) Suction cup Gripper (d) Compliance Gripper

Figure 1.1: Common types of Grippers

Control process of the gripper can be further split into 3 categories, namely, Task selection, Motion planning and Motion Control. Ozawa [6] has identified 3 different autonomy levels in controlling robotic gripper, which is illustrated in Figure 1.2. Master-Slave systems employs a ‘dummy’ small scale manipulator in the operators hand and the gripper would mimic what the operator does. Where shared autonomy includes the operator selecting the tasks and provides a minimal correction on the gripper operation. Recently with the development of Machine Learning, the task of the operator is as little as instructing the robot in a very high-level manner such as sort the item from location A to B and C. The robot would figure out autonomously decide on grasping strategy [3, 11].

Task selection related to what object should the gripper grasp and how the gripper should grasp the object. The simplest way to teach robots to grasp an object is by mimicking humans. Grasp taxonomy refers to how humans grasp different objects. There have been numerous studies focusing on this area. Some studies found that human grasping can be generalised into 6 ways [7, 8], while other studies categorise it into 16 ways [9], studies has even gone as far as analysing how the force is distributed in the hand when a human grasp an object. However, according to Ozawa [6], these different grasping methods can be further generalised based on the shape of grasp into 5 classes (Non-grasp, grasp, positioning/orientation, regrasp, object interaction) which is illustrated in Figure 1.3.

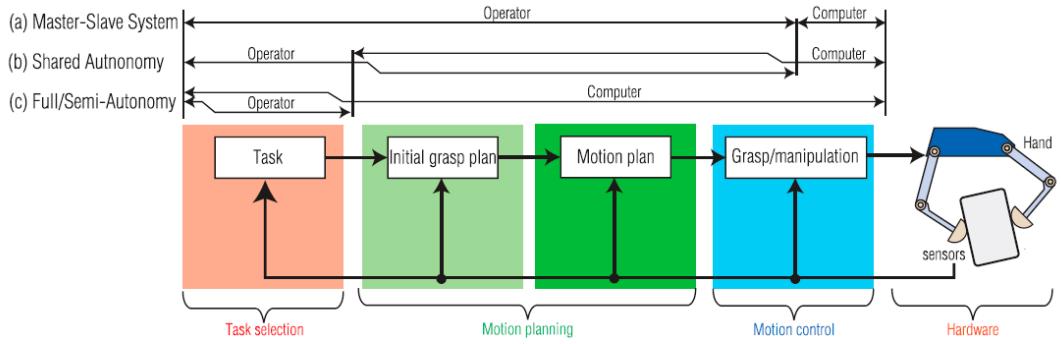


Figure 1.2: Autonomy of robotic gripper

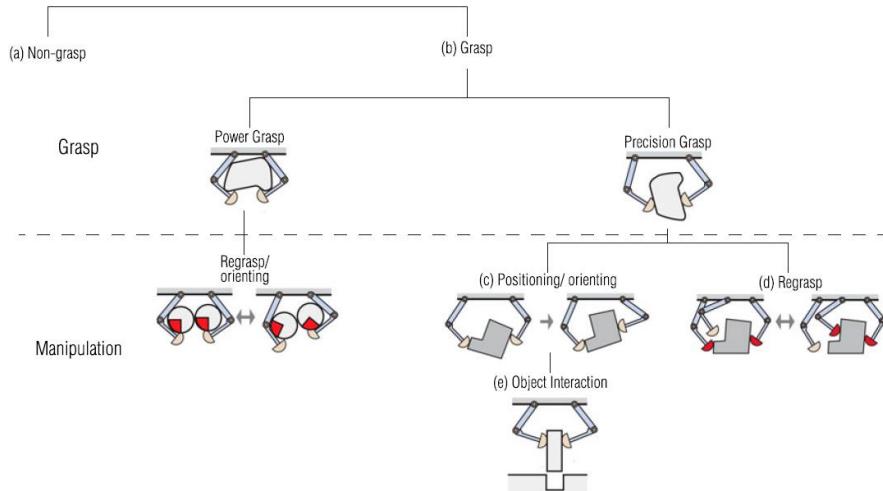


Figure 1.3: Grasp category

Motion planning involves finding the initial grasp point and motion to successfully grasp the object. The problem requires some knowledge of the object to be grasped such as the size and orientation. Based on the information provided, the control system would deduce a grasping strategy to safely grasp, manipulate and transport the object. Generally, to work out the grasping strategy, this requires a contact model where it models the forces experienced by the gripper and object when they interact. Basic model includes Frictionless point contact where it only takes into account normal force, Friction point contact where it also includes tangential force and Soft-finger contact which also models torsional moment. Details of these basic contact model are explained by Ozawa [6].

Finally, motion control ensures that the gripper is able to pick up the item by following the initial planned motion given by motion planning. The following section will investigate different types of motion control controller in 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 not only successfully grasp, but also does not fall while completing the task set by the task selector. Generally, the controller is responsible in controlling the finger movement.

The simplest form of grasping controller is one where it selects a certain grasping motion from a series of pre-programmed motion. This controller is known as sequential controller. 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 1 of the 20 pre-programmed motions selected by the operator. This controller however lacks the ability to grasp the object optimally as it is not able to produce the ‘overall best’ motion but rather the ‘best’ from the list of pre-programmed motions.

Realising the lack of flexibility of sequential controller, reflexive controller was next in line. This controller was motivated by observing a very young baby trying to grasp an item. As an object gets in contact to the palm of a baby hand, the baby instinctively closes the fingers and grasp the object. If attempts are made to remove the object, the baby would grasp even harder to oppose the motion. The controller requires some form of sensors embedded inside the gripper to detect the contact. Commonly light or optical sensors are used, and they are placed inside each jaw of the gripper [13, 14]. The fundamental flaw the controller faced is its inability to detect how much force is being applied. Theoretically, there is certain orientation that could lead to minimum force being applied such as when the object is grasped while the palm is facing upwards and start reducing the grasping effort until the object starts to slide. However, if the palm is facing downwards and there is no other visual feedback, the grasping effort might damage the object.

Reflexive controller highlights an important ingredient that is missing in robotic grasping, which is force control/measure. Naturally, this motivates a hybrid controller that control both the position and force at the same time. Hybrid controller requires dynamically decoupled the internal force from its motion. The control scheme was 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. This controller is a model-based controller which requires the knowledge of the system dynamics.

To abstract the dynamics from a system can sometimes proof to be time consuming or even difficult. Thus, there is a need of a controller that could grasp with minimal force even when the system dynamics is unknown. Adaptive controller was the way to go. Adaptive controller has been tested to have the ability to track motion trajectory even when there is mass, inertia, and centre of mass uncertainty when performing the calculation to dynamically decouple the internal force of the contact [16]. Other advantageous of adaptive controller is that it is as robust as a simple PD and hybrid controller while offering a much higher tracking accuracy and can perform well in high speed even when there is uncertainty [17]. Performing well when operated in high speed is desirable as this allows the gripper to be able to pick more items per hour.

Impedance control has been gaining attention as the importance of robotic grasping is realised and is well studied. Impedance controller adjusts the contact force between gripper and the environment by utilising mechanical impedance. This controller is simple as it directly imposes a mass-spring-damper [18] like structure to the interaction between gripper and the environment without the need to know the system dynamics and decoupling the internal force. Because of the simple structure, multiple fingers involved in manipulation can often be combined together and be represented by a simple mass and inertia structure [19], thus reducing the complexity of multi finger gripper.

All controllers discussed above does not utilise the information rich errors from previous iteration [20]. Therefore, this sparks the research into learning type controller. Iterative learning controller proposed by Nakayama [21] does not requires the knowledge of system dynamics for the position and force error to converge. As the error is reduces iteratively, by using a P-type configuration, the gripper tracking trajectory was identical to the ideal one after just 30 iterations. The paper strongly states that when the convergence of position tracking would lead to the convergence of using minimum force. However, this is only tested in simulation and research in this field is limited.

A summary of the controller discussed along with the advantage and disadvantage is shown in Table 1.1.

Table 1.1: Comparison between different grasping controllers

Name	Complexity	No. of grasping sequence	Force control	Decouple internal force?	Knowledge of system dynamics?	Use previous iteration error?
Sequential	Simple	Finite	No	N/A	N/A	No
Reflexive	Simple	Infinite	No	N/A	N/A	No
Hybrid position/force	Moderate	Infinite	Yes	Yes	Yes	No
Adaptive	Moderate	Infinite	Yes	Yes	No	No
Impedance	Simple	Infinite	Yes	No	No	No
Learning	Moderate	Infinite	Yes	No	Depends on scheme	Yes

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 in a short time with little budget commitment before building a prototype. Not only 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 strength and weakness of different grasping controller was identified. It was discussed that robotic grasping often operates in a repetitive environment and most development has not utilised the information rich error signal produced in each iteration. In the research done on ILC, it is concluded that the actual trajectory of the gripper is identical to the ideal trajectory. However, since it utilised a non-model method, it only converges after 30 iterations. The limitation of current generation grasping controller identified in chapter 1.1 still remains.

Therefore, a novel approach is to design a grasping controller that utilises the advantageous of controllers discussed in chapter 1.3 to solve the limitation discussed in chapter 1.1. As such, the following design requirements was identified.

2.1 Design Requirements

The following design requirements focus on the selection of simulation environment as well as model used to test the controller.

- DR1. Simulation model is chosen such that it is **closely related 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 **mimic real world interaction dynamics/forces**. This allows minimal inaccuracy between simulation environment and real world and ensures 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 assists in discovering any potentially ‘hidden’ issues that might not be captured or not able to represent in the form of data by the model structure.

The following design requirements highlights the aim of what the controller should achieve to eliminate or improve limitation raised in chapter 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 gives 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 that the gripper would not employ excessive force that might damage the object while considering any uncertainties in the environment and object modelling.
- DR7. The robot (arm and gripper) shall be capable of **grasping object of different mass**. The object would range between 0.1kg to 2kg. This ensures the robot is capable of grasping a range of objects.
- DR8. The robot (arm and gripper) shall be capable of adjusting itself to **correct 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 arise 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 arise when the robot bumps into items or there is a massive wear and tear in certain part of the system.

2.2 Overview of Iterative Learning Control (ILC)

Iterative learning control uses information rich errors from previous iterations and learn from it. ILC has been successfully implemented in numerous other applications. Some of the highlights of ILC being applied elsewhere includes its ability to control motion with high precision [23], able to cancel 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 chapter 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

2.2.1.1 D-Type

The earliest ILC controller was proposed by Arimoto [26] where the algorithm uses the error generated by the previous iteration to modify the input vector to the plant in the next iteration in the hopes of reducing the error. Vectors are used to allow the algorithm to perform batch process between each iteration. The equation was defined as

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

where the error was defined as

$$\mathbf{e}_k(t) = \mathbf{r}(t) - \mathbf{y}_k(t) \quad (2.2)$$

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

The condition for convergence has been well studied and the system will converge to 0 error when the iteration number goes infinite, and the convergence criteria is met. Otherwise, the system might experience divergence where the error increases.

2.2.1.2 P-Type

As D-Type ILC controller utilise the derivation of the current iteration error, there is a chance that small errors are amplified by the derivation process. Performing derivation on a vector of data can sometimes be challenging for less powerful processors, therefore Arimoto [27] proposed a P-type ILC with the equation defined as follows

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

There are numerous other variations of non-model based ILC controllers such as PD, D-alpha, Higher order, Hybrid, Anticipatory and so on that the paper would omit. The reason for this is the fact that non-model based ILC would not outperform a model based ILC as they do not directly counter the system dynamics and thus would always converge slower; where to realise the design requirements, the controller has to converge as fast as possible.

2.2.2 Model Based ILC

Model based ILC requires the knowledge of the plant, particularly the system dynamics.

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 to model an identical plant, therefore studies have shown that even if the model shows some inaccuracy, the system would still be able to converge, while taking a few more iterations. As more and 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 exploit the fact that the plant model is known by directly countering the dynamics with an inverse of the plant. The output of the system is

$$y_k(t) = G^0(u_k(t)) \quad (2.4)$$

where G^0 denotes the true plant. The error between desired and actual trajectory remains unchanged and is denoted in Equation 2.2. As the plant is known, the optimal input sequence can be obtained as follows

$$u_k(t) = -G^0(r(t))^{-1} \quad (2.4)$$

where $G^0(\cdot)^{-1}$ 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 involves solving and optimised cost function by finding its minimum point. This minimisation problem attempts to minimise the tracking error described by Equation 2.5.

$$J(\mathbf{u}^*(k)) = \min_u J(\mathbf{u}(k)) \quad (2.5)$$

$$\nabla_{\mathbf{u}} J = \left[\frac{\partial J}{\partial \mathbf{u}_i} \right] = 0 \quad (2.6)$$

By using gradient descent method shown in Equation 2.6, the optimal input that minimises the cost function can be obtained [30].

To further improves the performance, the learning gain should be high when error is large while small when the error is small. As such, the gradient descent step size should be chosen automatically [31].

A new minimisation problem has been identified by Amann et al. [32] and is described as follows

$$u_{k+1} = \arg \min_{u_{k+1}} \{J_{k+1}(u_{k+1})\} \quad (2.7)$$

where the cost function and error are defined as

$$J_{k+1}(u_{k+1}) = \|e_{k+1}\|_{\mathcal{Y}}^2 + \|u_{k+1} - u_k\|_{\mathcal{U}}^2 \quad (2.8)$$

$$e_{k+1} = r - y_{k+1} \quad (2.9)$$

$$y_{k+1} = Gu_{k+1} \quad (2.10)$$

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

Detailed calculation has been omitted but can be found at paper published by Amann et al. [32]. The final input sequence after solving the minimisation problem is Equation 2.7 is as follow

$$u_{k+1} = u_k + (R + QG^T G)^{-1} QG^T e_k \quad (2.11)$$

Where R and Q are diagonal matrices. R minimises the change in input sequence between iteration 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. The variation has been proven by mathematically and through experiments to be able to reduce normalised tracking error, able to choose step size automatically and improve robustness.

2.3 Overview of Impedance Control

To quantify the improvements ILC should make over conventional controllers, a baseline controller should be developed and subject to the same test. After reviewing the advantageous and disadvantageous of different controllers discussed in chapter 1.3, impedance control is chosen as a baseline comparison controller. Impedance control is the current lead in grasping controller mainly due to its simple and easy implementation while giving promising result. 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 system 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 by inertia where it accepts effort and produce flow.

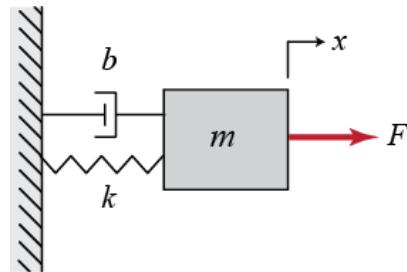


Figure 2.1: "Mass-spring-damper system" by ControlTutorialsForMatlab&Simulink is licensed under CC BY 4.0

This system can be described by the following equation

$$F = Ma + Cv + Kx \quad (2.12)$$

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, by using the concept of Laplace transform, equation 2.12 can be written as

$$F(s) = (Ms^2 + Cs + K)X(s) \quad (2.13)$$

where it can be rearranged to form the following transfer function

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

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

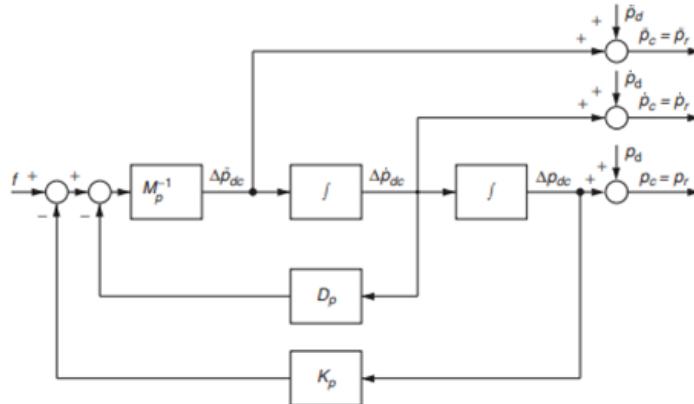


Figure 2.2: Block diagram for Impedance control using Integration method

The operator would be tasked to choose 3 values, namely the 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 environment (soft and stiff) [33]. However, as this takes the force error and adjust the gripper target trajectory, a simple PD controller is typically employed to ensure the gripper position error remains as low as possible [34].

3 Design Overview

This section will provide a high-level overview and general approach for 4 crucial parts of the project namely

- **Simulation environment** including simulation environment selection, robot selection and model building 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 set out in chapter 2.1.
- Proposed **disturbance test** procedure.

As the project is complex, each of the 4 parts of the project will be split further into multiple bite size subparts to ensure each feature is thoroughly tested, verified, and analysed. More project and time management technique and plans can be found in Appendix A.

3.1 Simulation Environment

DR 1 listed in chapter 2.1 ensures that the simulation model would reflect real world robots. There are 2 commonly used robots in the industry, 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 area while robotic arm combines multiple joints to moves the gripper and has a sphere work area. Gantry robot is the simplest robot and Ratcliffe [37] has conducted research using ILC on such system, therefore this is an ideal robot to test the gripper controller algorithm before preceding to the more complicated robotic arm system. Figure 3.1 shows the proposed gantry robot.



Figure 3.1: Proposed gantry robot by Ratcliffe

According to chapter 1.2, parallel plate grippers are the simplest and most universal. Therefore, to ensure the model represents most of the currently used gripper in the market, the decision to use parallel plate grippers in the model is chosen.

There is numerous simulation software out in the market. Some are opened source, others required licensing. Utilising university subscribed software and open source, the 3 best promising software for robotics has been identified, they are Gazebo, Webots, and MATLAB. Gazebo is licensed open sourced and is considered to be one of the 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 do not have as good documentation as the others and the community are not so active, on top of that, the interface does not follow established conventions. Webots is a good alternative to Gazebo as it has simple and easy to use GUI, huge model database, has an active community as well as good documentation and open sourced as well. Webots is mainly programmed using C++ but there is API for other programming language such as Java, Python and even MATLAB [38]. Although MATLAB is proprietary, the university has subscription for education use. MATLAB is commonly best known for data analysis as well as a ginormous collection of toolboxes, one of it is a widely used control system design toolbox. Simulink is built on top of MATLAB and has an easy-to-use graphical based environment utilising ‘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: 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

Gazebo, Webots and MATLAB with Simulink and Simscape is capable of simulation interaction force a robot might encounter in real life and contains 3D visual models which fulfils DR 2 and 3 listed in chapter 2.1. Keeping in mind the aim of the project is to design a controller for a robot and the pros and cons listed in table 3.1, it is decided that the good numerical computation and control system design toolbox that MATLAB is the best choice to design, test and analyse the controller performance.

Since MATLAB does not have a built-in gantry robotic model that matches the proposed model. The modelling and testing both the gantry robot and environmental contact in Simulink will split into sections as shown in Figure 3.2.

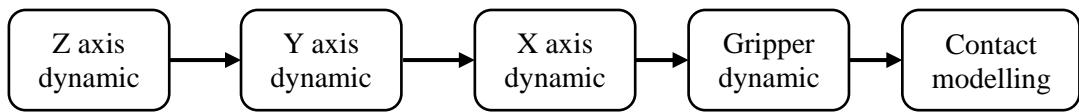


Figure 3.2: Steps on modelling gantry and environment dynamics in Simulink

3.2 Gantry Controller

After reviewing Ratcliff [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. The next iteration input vector equation is the same as equation 2.11 and denotes as

$$u_{k+1} = u_k + (R + QG^T G)^{-1} QG^T e_k \quad (3.1)$$

While the error vector will be the result of the difference between the ideal trajectory and actual trajectory. As this project focuses on motion controller, the ideal trajectory would be given by the operator and is predefined. As the project uses a gantry robot, the 3 axes can be decoupled as they are independent of each other. Therefore, 3 separate controllers will be built, each will be in charge of controlling one axis and the values would be tuned to meet DR 4 and 5 listed in chapter 2.1. The schematic for the controllers is shown in Figure 3.3.

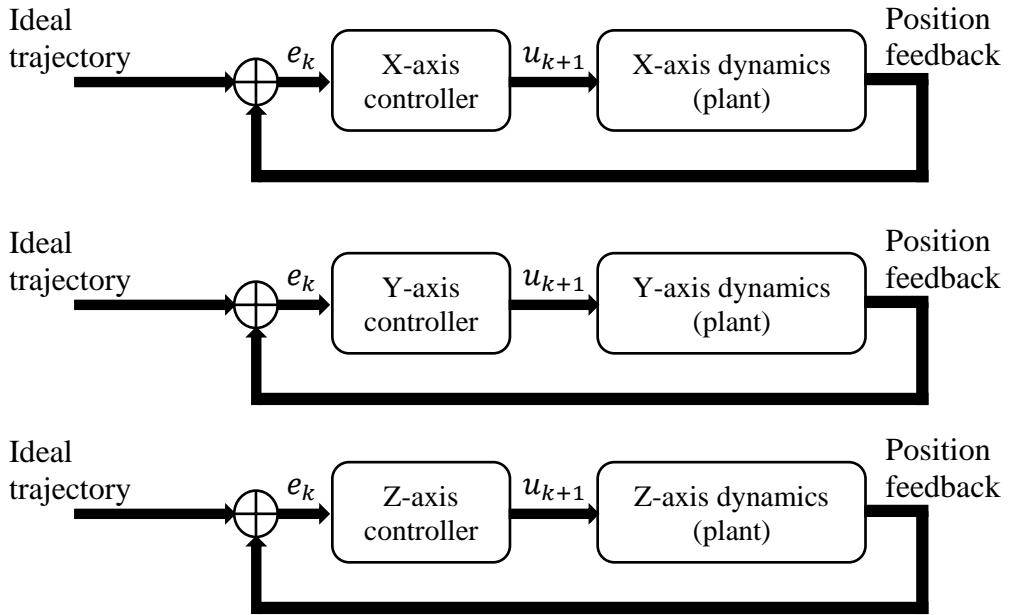


Figure 3.3: High-level schematic for X, Y and Z axis controllers

3.3 Gripper Controller

Recalling that the gripper has to satisfy a few conditions set by the design requirement in chapter 2.1 which includes the ability to grasp the different mass objects quickly and accurately while using the least amount of force and able to overcome some commonly encountered disturbance. Thus, having reviewed the success of ILC in chapter 2.2 and the experiment conducted by Ratcliff [37], ILC is one of the best solutions to design a robust and fast converging learning controller. To prove the advantageous of ILC over conventional grasping controllers, impedance control is chosen as a baseline controller. The following subsection will give an overview of the design schematic of both controllers.

3.3.1 Impedance Control

As the impedance control algorithm focuses on minimising the force error between ideal and actual by changing the opening size of the gripper and is operating in real time instead of iteration. Therefore, ILC could not be used as position control, therefore, PD control is use instead. The schematic is shown in Figure 3.4.

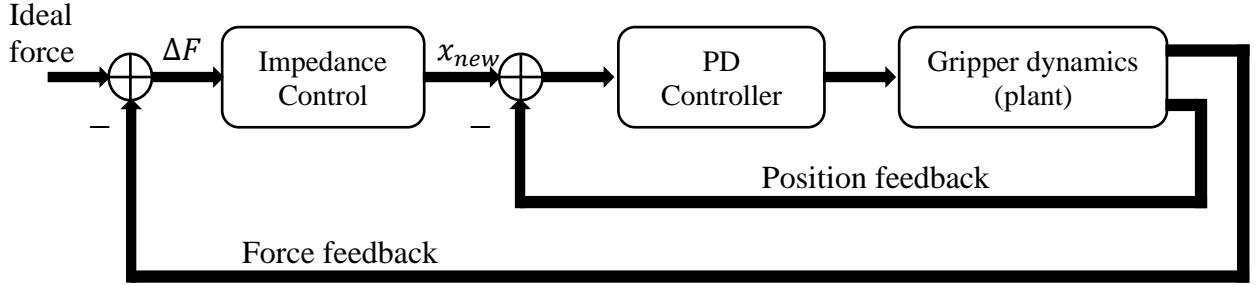


Figure 3.4: High-level schematic for impedance gripper controller

Where the force error ΔF is fed into the impedance controller and x_{new} is the new trajectory for the gripper to realise minimum force grasping. Recalling that impedance control formula shown in equation 2.12, the same equation but digitised and change of notation is as follows

$$\Delta F(k) = M\Delta\ddot{x}(k) + B\Delta\dot{x}(k) + K\Delta x(k) \quad (3.2)$$

After rearranging making $\Delta\ddot{x}(k)$ as the subject using integration method discussed by Caccavale et al. [36], equation 3.2 is denoted as follows:

$$\Delta\ddot{x}(k) = M^{-1}(\Delta F(k) - K\Delta x(k) - B\Delta\dot{x}(k)) \quad (3.3)$$

Equation 3.3 can be represented in block diagram as shown in Figure 3.5, which would be used as the impedance control update.

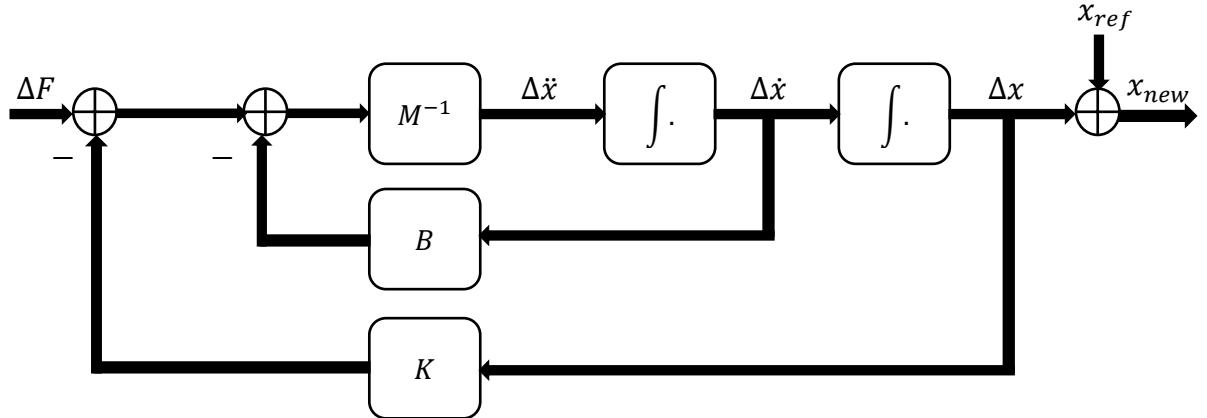


Figure 3.5: Impedance controller schematic

3.3.2 Iterative Learning Control

The schematic for ILC would be similar to impedance control schematic and is shown in Figure 3.6.

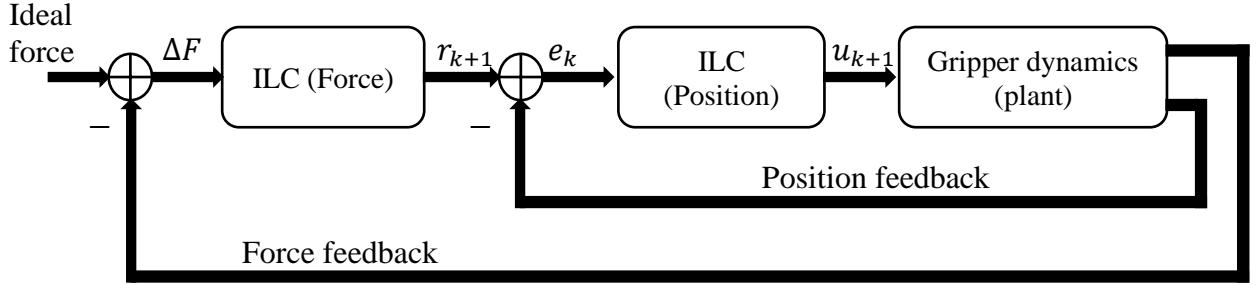


Figure 3.6: High-level schematic for ILC gripper controller

Where r_{k+1} denotes the new reference trajectory for the gripper.

The position control ILC schematic will use norm-optimal configuration due to its robustness and fast convergence feature and is calculated using equation 3.1. Norm-optimal configuration requires a plant model which for position control is the gripper dynamics. However, the minimisation problem for force control is not as straight forward as using the gripper dynamics as the plant, because there are other factors that should be modelled such as the reaction forces between the gripper, object, and environment. Therefore, a simple D-type ILC is chosen. This force ILC would change the reference trajectory for the gripper to follow, similar to the impedance case and can be calculated using the following equation.

$$\mathbf{r}_{k+1}(t) = \mathbf{r}_k(t) + \beta(t)\Delta\mathbf{F}_k(t) \quad (3.4)$$

Where r_k denotes the last iteration reference trajectory, scalar gain β and $\Delta\mathbf{F}_k$ denotes the force error.

The force and position updates will happen at the end of each iteration instead of ‘real time’ update that impedance control offers.

3.4 Disturbance

To fully quantify the advantages of ILC over impedance control for gripper, one should consider its robustness to unavoidable environmental disturbance. Two of the common disturbance that a robot would encounter is discussed in chapter 2.1 under Disturbance Scenario (DS).

As these two disturbances occurred in the environment 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.

$$\tilde{u} = u + d \quad (3.5)$$

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

As such, the disturbance test schematic shown in Figure 3.7 can be used for both X, Y and Z axis controller by ILC and gripper deployed either ILC or impedance control scheme.

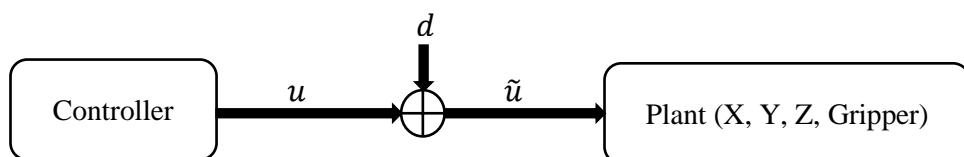


Figure 3.7: High-level schematic of controller output corrupted by disturbance

4 Simulation Model

Chapter 3.1 have decided on the simulation environment, robot to be used, and overview of the whole design. This chapter would translate the simulation model design overview into MATLAB environment using Simulink and Simscape Multibody. Simscape Multibody is one of the toolboxes provided by MathWorks that allows the designer to model and simulate multibody mechanical system.

4.1 Gantry Modelling

Simscape multibody and Simulink offers a drag and drop block design feature. Each axis of the gantry robot has a few important features

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

The goal is to replicate the dynamics of each axis as identified by Ratcliff [37] in the simulation environment.

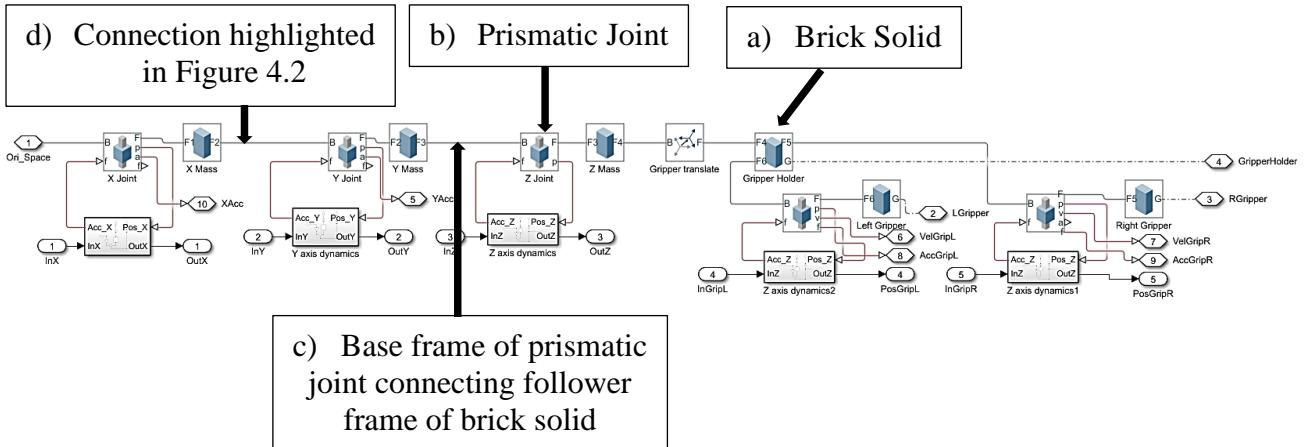


Figure 4.1: Schematic of proposed model in Simulink

The mass of the axis would be replicated using ‘Brick Solid’ shown in Figure 4.1a, where the adjustable parameters include 3D dimensions and density to specify the length and mass of the object. Details of the values can be found in Table 4.1. The length of X axis brick solid is different from its work area to ensure visually the movement is correct, where the length of the actual aluminium strut in the gantry model is facing Y direction while moving in the X direction. Same case for Y axis mass. Where the gripper finger would close in the Y axis direction.

The moving of the brick solid in 1 axis direction will be replicated by ‘Prismatic Joint’ shown in Figure 4.1b. This is a linear moving component that would move in the Z axis direction. It accepts ‘force’ as input to move the object and is capable of outputting a range of parameters including position, velocity, acceleration, actuator force and so on.

Frames are a very important concept when modelling using Simscape Multibody. There are 2 main frames that are crucial, base frame 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. Illustrated in 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 in the direction of intended travel direction, the base frame will have the same orientation of the previous ‘object’ follower frame to ensure it ‘stacks’ on top nicely. Illustrated in Figure 4.2.

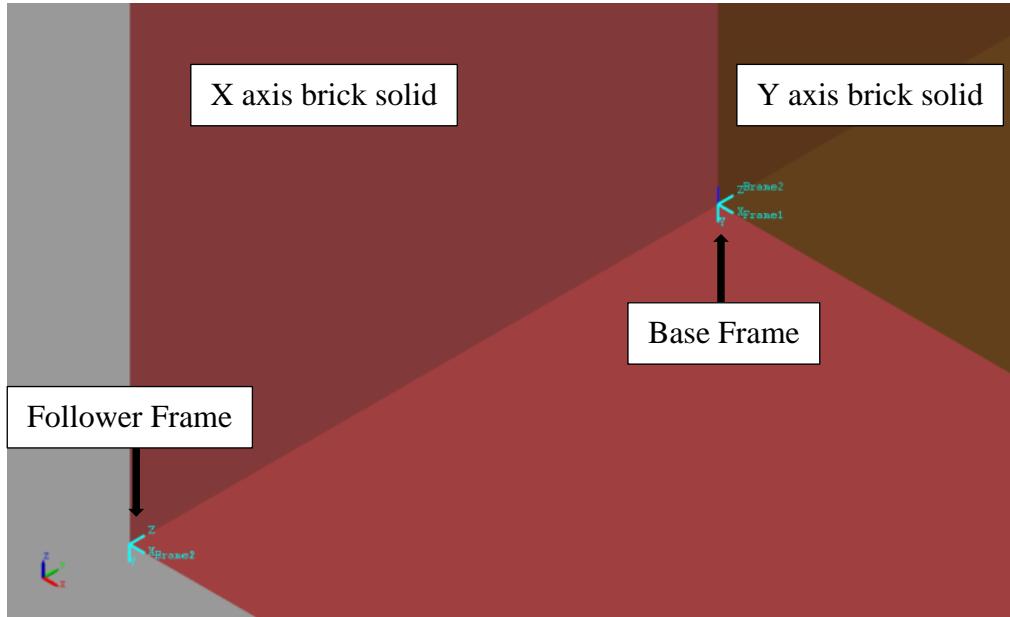


Figure 4.2: Visual representation of the base and follower frame connection (between X and Y axis) highlighted in Figure 4.1d

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

Using brick solid and prismatic joint configuration is simple and yet capable of achieving the visual aspect of axis moving to grasp object. 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 include a PD controller using position feedback and changing the input force to prismatic joint to model the dynamics of the system. An illustration of such system can be found in Figure 4.3. The following subsections will explain issues encountered by such system, iterative of improvement and finally a proposed final solution.

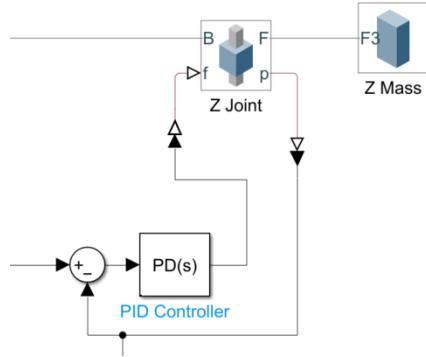


Figure 4.3: Proposed PD controller to model the dynamics of the system

While grouping prismatic joint and brick solid together as ‘Plant, P’ and using PD controller denoting as ‘C’, the transfer function of the proposed system is as follows

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

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

4.1.1 Trial 1

Test has been conducted to find out the Laplace transform of the prismatic joint connected to mass system and can be represented using Equation 4.2. The test is shown in Figure 4.4.

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

where m is the mass of the object.

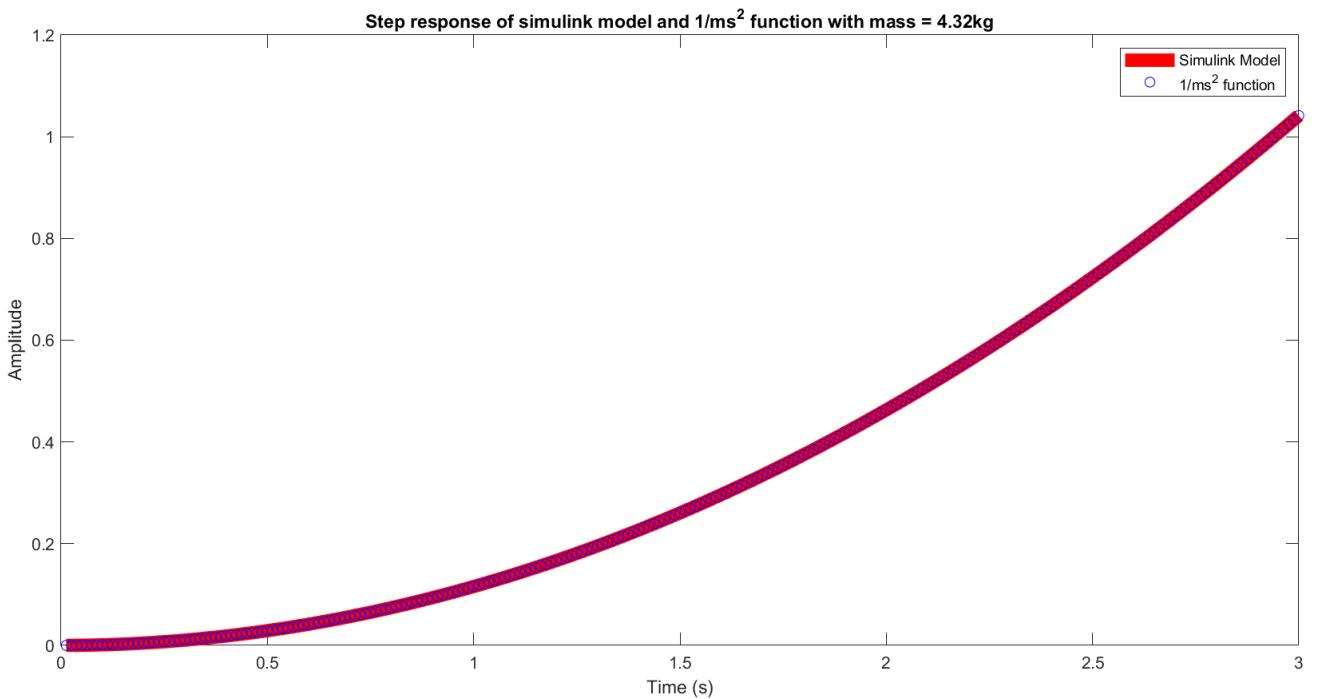


Figure 4.4: Step response for the prismatic and brick solid model VS the equivalent representation shown in Equation 4.1.

As the system expands, the mass of the first ‘set’ is its own brick solid in addition any other brick solid connects together, therefore for the value of mass needed to substitute to Equation 4.2 is the sum of X, Y, Z and Gripper mass which equates to 8.098kg. Similar steps are conducted to find the respective mass for Y, Z, and gripper.

Using the schematic proposed in Figure 4.3. The PD controller is tuned using a MATLAB built-in feature call PID Tuner. PID Tuner has a graphical GUI where the user is able to change the values of P and D by either typing the value or using a slider and see the response in real time. Figure 4.6 shows the GUI of PID Tuner. To ensure the dynamics is as similar to the actual dynamics identified by Ratcliff [37], the impulse response of actual system transfer function (Figure 4.5) is match with the step response of the model schematic (Figure 4.6). The impulse response graph only offers the peak response time and transient time in the graph; therefore, the PD is tuned so that the model peak response and transient time is match as closely as possible. The best values are $k_p = 33634.3806$ and $k_d = 163.3421$.

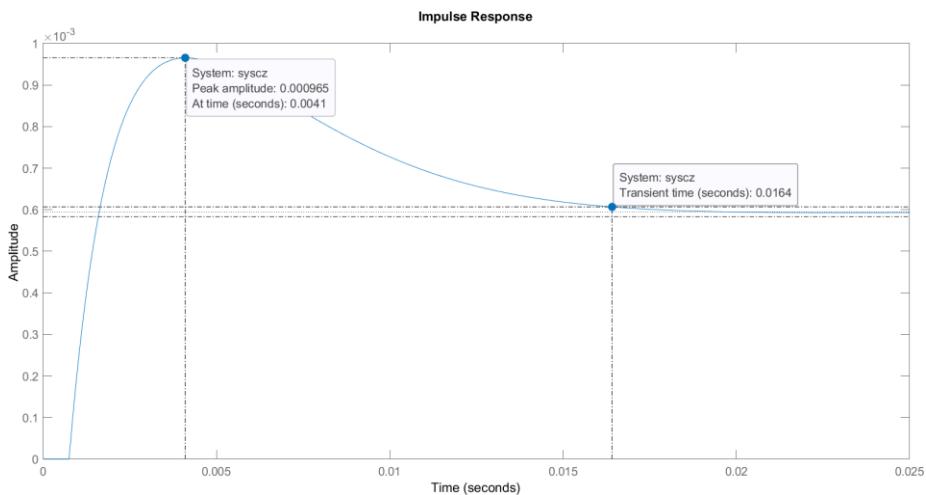


Figure 4.5: Impulse response of Z axis actual transfer function

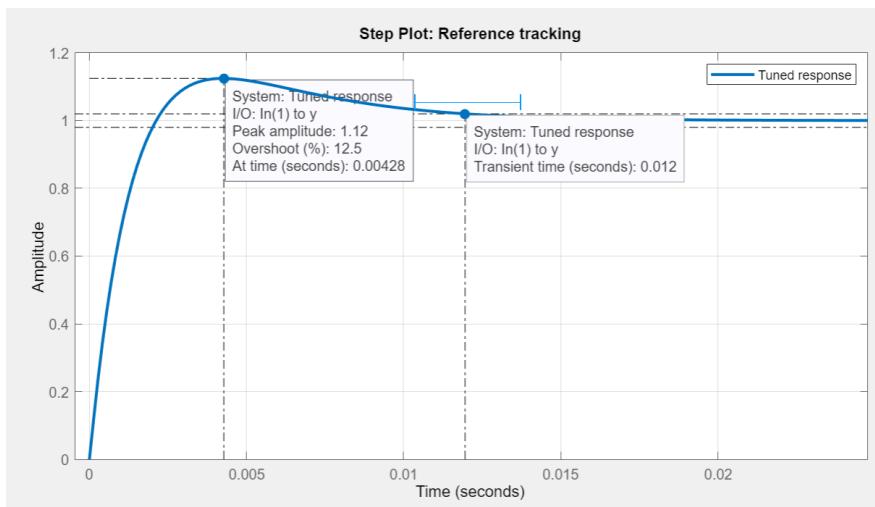


Figure 4.6: Step response of Simulink model using schematic proposed in Figure 4.3

Although there are differences between the peak response and transient time of impulse response and step response, the difference is small, in terms of few milliseconds and thus is acceptable considering the grasping operation is in seconds.

The step response of the system denoted by Equation 4.1 and 4.2 can be denoted to as follow.

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

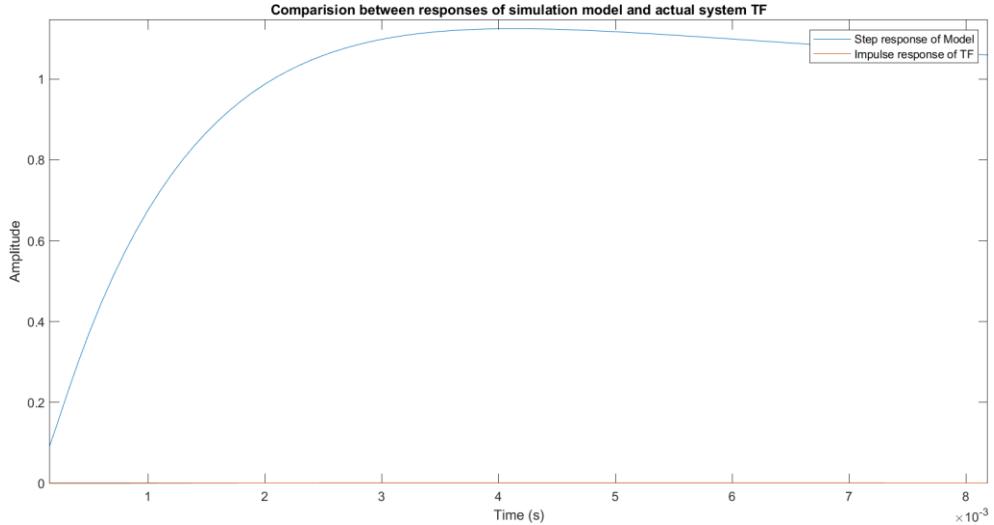


Figure 4.7: Comparison between step response of Simulink model and impulse response of actual system transfer function for Z axis

Plot in Figure 4.5 and 4.6 can be placed together and is shown in Figure 4.7. It can be seen that although the peak response and transient time for both responses are roughly the same, the steady state amplitude is different. This is due to the schematic proposed in Figure 4.3 where the steady state of the step response of the Simulink model is 1, this can be proved by applying the Final Value Theorem to Equation 4.3 and is shown below.

$$\lim_{s \rightarrow 0} sF(s) = \frac{k_p}{k_p} = 1 \quad (4.4)$$

4.1.2 Trial 2

Due to the steady state response issue experienced in previous section. There are 2 potential solutions to this, 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 to the input signal sent to the PD controller to scale down the steady state response, so it matches with the actual system.

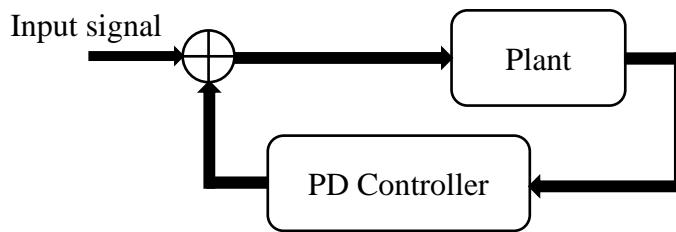


Figure 4.8: Proposed schematic where PD controller is located at the feedback loop with no scalar gain

The new transfer function to represent the system is as follows.

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

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

Upon applying the final value theorem, the equation becomes

$$\lim_{s \rightarrow 0} sF(s) = \frac{1}{k_p} \quad (4.6)$$

where

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

Therefore, the steady state value will be determined by the proportional gain value chosen to tune the PD controller. This reduces the flexibility to choose any values to ensure the Simulink model will have the same system dynamics as the actual system transfer function. On top of that, the PID tuner feature assumes the PID controller is placed in the way that it cascades with the plant. Therefore, PID tuner cannot be used to tune the PD controller to achieve dynamics similar to the actual system dynamics.

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 PID tuner does not work when the PD controller is placed in a feedback configuration, the following schematic is proposed.

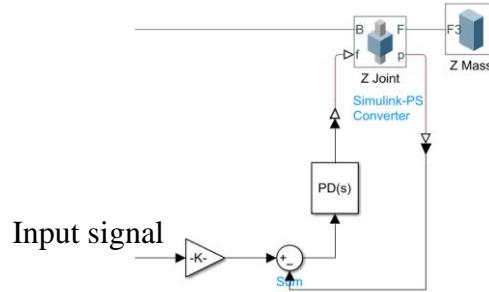


Figure 4.9: Proposed schematic for modelling gantry robot dynamics with PD controller and scalar gain

The transfer function of the system can be formulated as follows

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

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

The scalar gain will be denoted as α , where α is the steady state value of the impulse response of actual system transfer function. Where it is obtained visually using graph, Figure 4.10 denotes the value for Z axis which is taken to be 5.93×10^{-4} and thus $\alpha = \frac{1}{1686.3406}$.

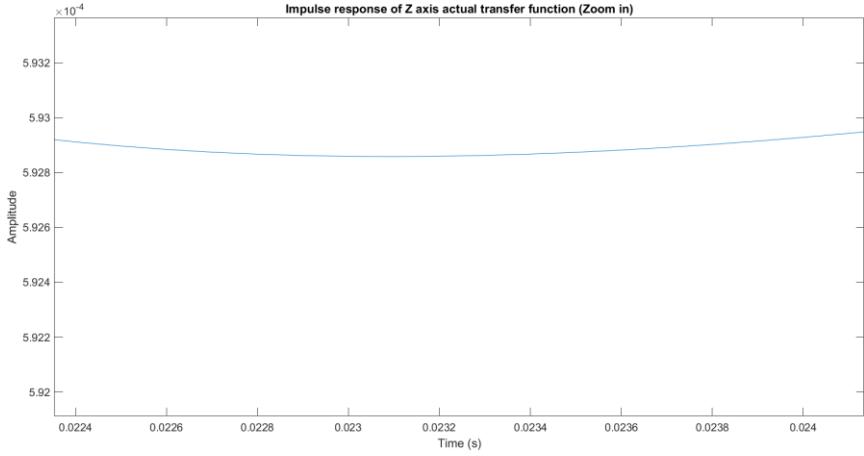


Figure 4.10: Zoom in impulse response of Z axis actual transfer function

The impulse response of Z axis actual transfer function and step response of Z axis model is placed together and is shown in Figure 4.11.

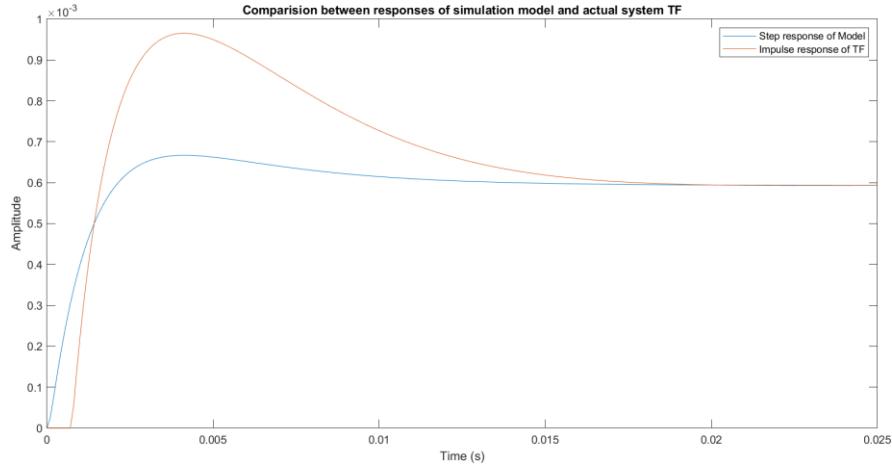


Figure 4.11: Comparison figure between step response of Simulink model with scalar gain and impulse response of actual Z axis transfer function

Although the peak amplitude is different, the peak response and transient time are similar, and the amplitude is in milli scale therefore the small inaccuracy is acceptable. Since the design overview of ILC controller presented in chapter 3.2 and 3.3 uses model based ILC which requires the plant model to be known. Therefore, using Equation 4.7 and substituting the total mass, PD parameter values, the plant model (known as apparent transfer function) is as follows

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

The plots of impulse response of actual transfer function, step response of Simulink model and step response of ‘apparent transfer function’ can be found in Figure 4.12. The step responses of Simulink model and ‘apparent transfer function’ are almost identical with some small differences which could be due to numerical inaccuracy as the values are enter manually and some decimal points are omitted.

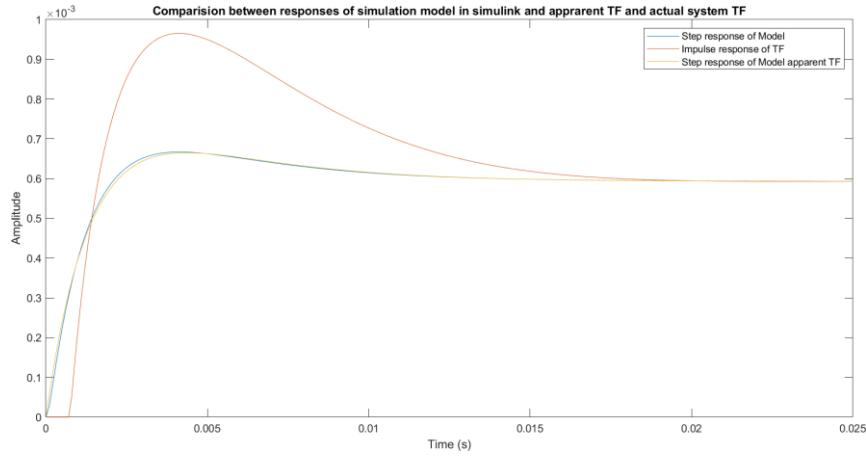


Figure 4.12: Comparison figure between step response of Simulink model with scalar gain and 'apparent transfer function' and impulse response of actual Z axis transfer function

To find and model the X, Y and gripper dynamics, the above steps are replicated, and the values can be found in Table 4.2.

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

Part	α	k_p	k_d	Sum masses
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 final schematic of the system in Simulink is shown in Figure 4.13 where the axis dynamics block follows Figure 4.9 schematic. While the visual aspect of the system is shown in Figure 4.14.

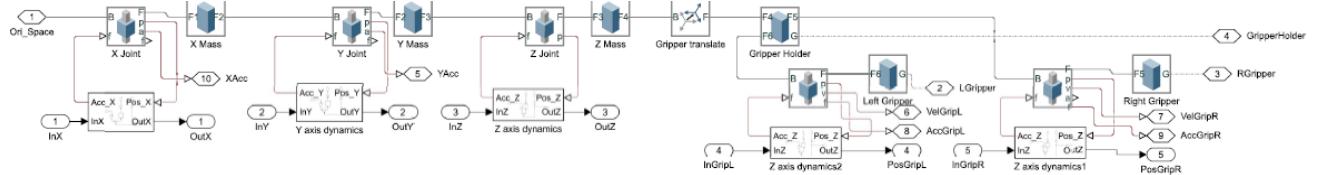


Figure 4.13: Overall schematic of gantry model

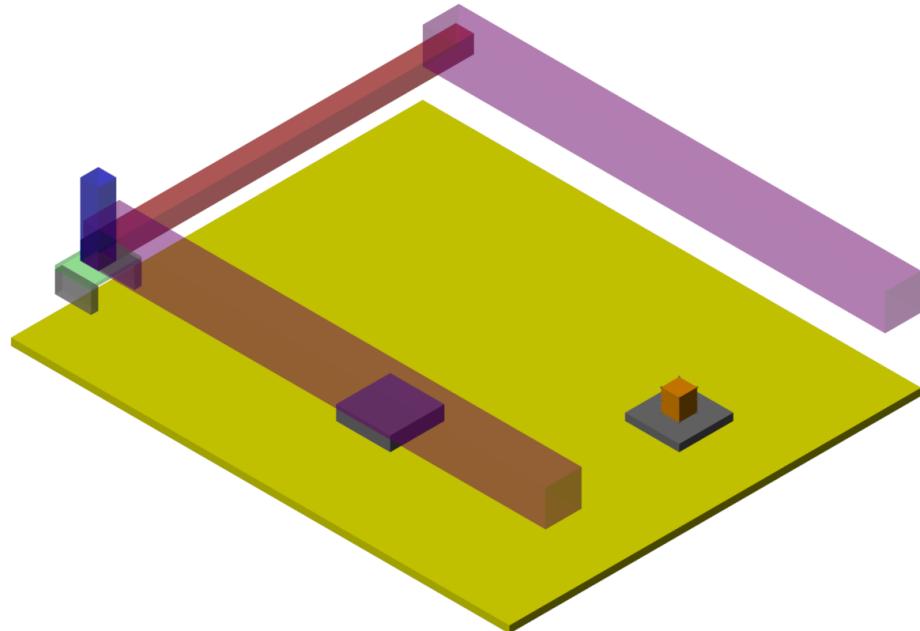


Figure 4.14: Visual view of gantry model

4.2 Contact Modelling

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

Spatial contact force uses penalty force-based method to implement contact force. Penalty method allows the two-contacting object to overlap each other a little bit to generate contact force. In Simulink Multibody it uses a spring-damper system to simulate the contact force. Figure 4.15 shows a diagram on how Simscape Multibody models the contact.

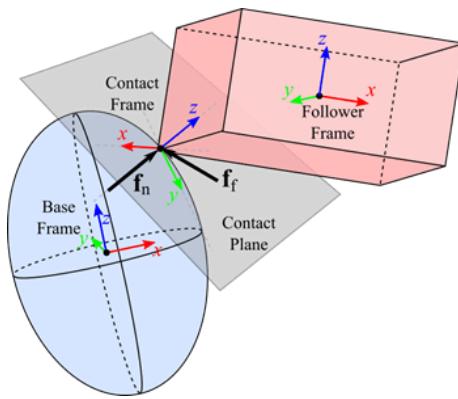


Figure 4.15: Simscape Multibody contact forces explanation diagram (adapted from MATLAB website)

Spatial contact force allows the user to adjust the amount of normal and frictional force generated when two bodies are in contact. Normal force can be adjusted by changing the stiffness and damping value. The damping is set to 0 for easier implementation. While the stiffness is set to 1000N/m. This means if the two bodies overlap each other by 1m, it will exert 1000N of force.

The frictional force will be computed using the built-in smooth stick-slip method, where the parameters of interest are coefficient of static and dynamic friction. The coefficient of dynamic and static friction is set to the same value of 0.5, this can be justified by imagining the gripper always having enough grasping force once gripped to prevent slipping.

The body that will be in contact with spatial contact force will be small 1mm radius Solid Sphere which would be placed strategically around the object to be grasp. This speeds up the modelling to improve performance as oppose to using the object surface directly.

One of the contact force connection schematics is shown in Figure 4.16.

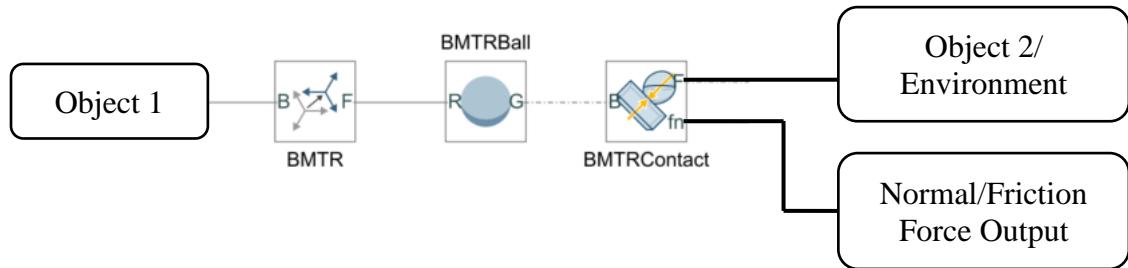


Figure 4.16: Schematic for modelling contact forces between two bodies

Object 1 would be the object to be grasped while Object 2 will be the gripper fingers and Environment will be the conveyer where the object will be lifted and lowered. The normal and frictional force can also be provided as an output for Spatial Contact Force. In total, the object to be grasped will have 12 spheres spread around equally as shown in Figure 4.17.

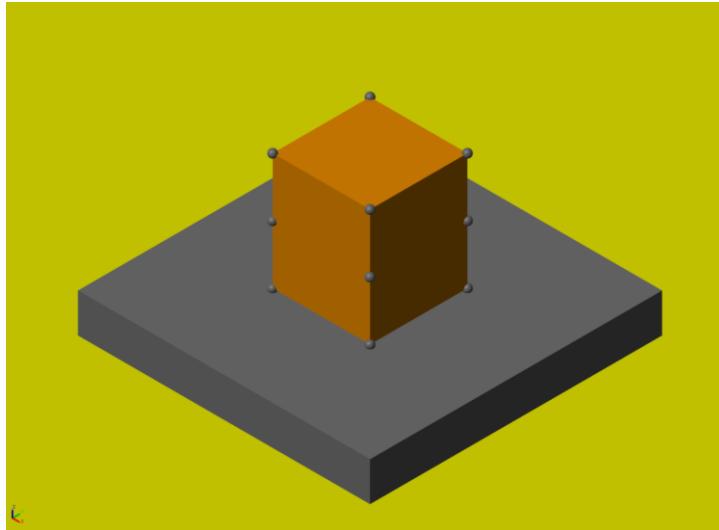


Figure 4.17: Diagram showing contact points (sphere) of the object to be grasped

This ensures that there will be at least 4 spheres in contact with either the gripper or the conveyer the object will be placed on. Each of the gripper finger will have 4 spheres and this is enough to ensure the object does not rotate in any direction while being grasp by the gripper.

4.2.1 Grasping Force Calculation

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

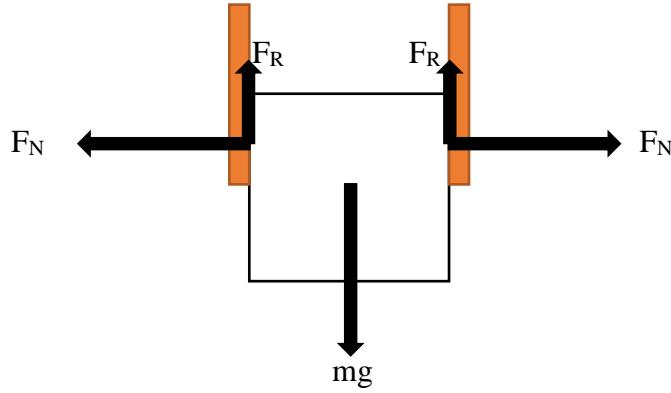


Figure 4.18: Free body diagram of grasping and lifting an object with two fingers parallel plate gripper

To ensure the object can be lifted,

$$2F_R \geq mg \quad (4.9)$$

recalling the friction force formula,

$$F_N = \mu F_R \quad (4.10)$$

where μ is set previously to 0.5. Since the magnitude of the grasping force for one finger is the same as the normal force it experienced Each gripper has to exert a force of,

$$F_g = \frac{(0.5)m(9.81)}{2} \quad (4.11)$$

Therefore, the theoretical minimum force to grasp and lift the object securely can be calculated. There is also a relationship between the amount of spatial contract force that is in contact between the gripper and object. Figure 4.19 show that each gripper finger is in contact with 4 spheres.

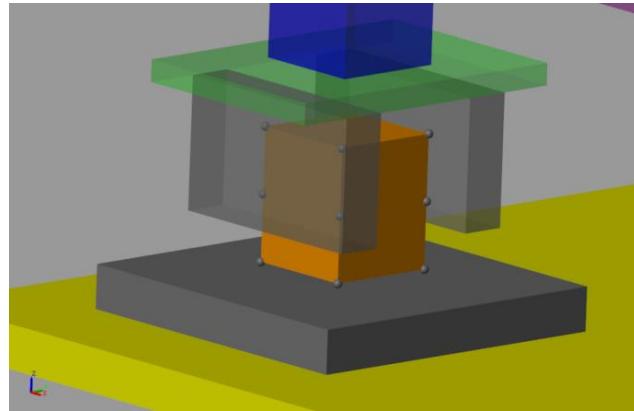


Figure 4.19: Diagram showing the contact between two bodies of interest (Object and Gripper, and Object and Conveyer)

With a mass of 2kg object, the frictional force of each gripper finger can be found using Equation 4.9.

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

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

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

which is the same value in the graph shown in Figure 4.20. The Minimum normal force is calculated as follows

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

This value is above the normal force experienced as shown in Figure 4.20.

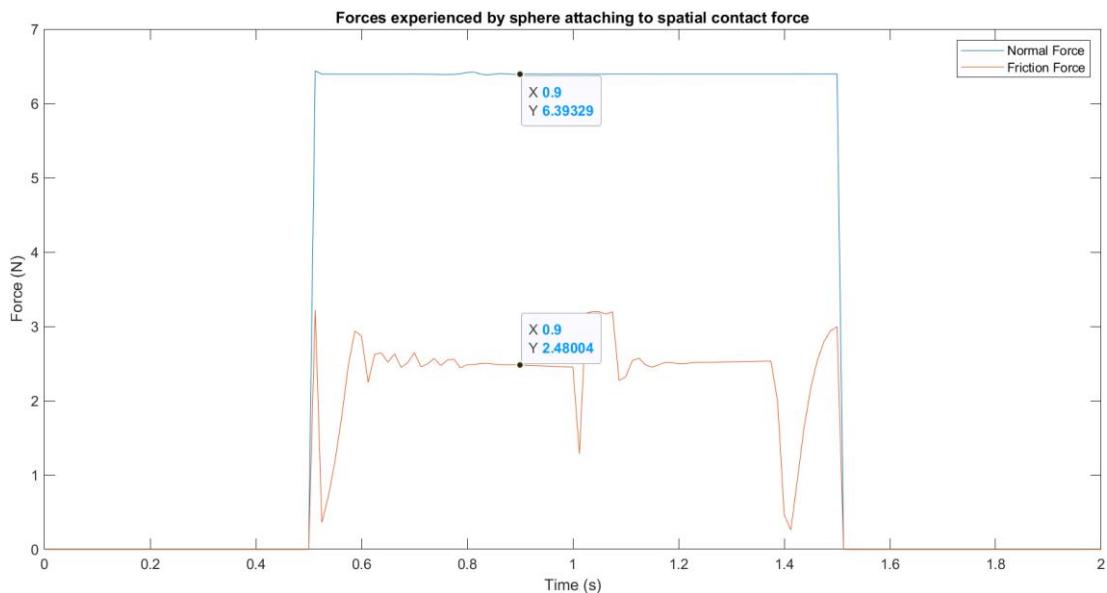


Figure 4.20: Graph showing the Normal and Frictional forces that the spatial contact force experience when grasping and lifting the object from conveyer

4.2.2 Normal Force while Accelerating

As shown in Figure 4.19, the gripper fingers are parallel to the X-axis. Therefore, as the gripper moves in Y direction, the normal force experienced by each gripper changes due to the acceleration and deceleration of moving in that direction. Figure 4.21 shows the how the normal forces change as it experiences the movement.

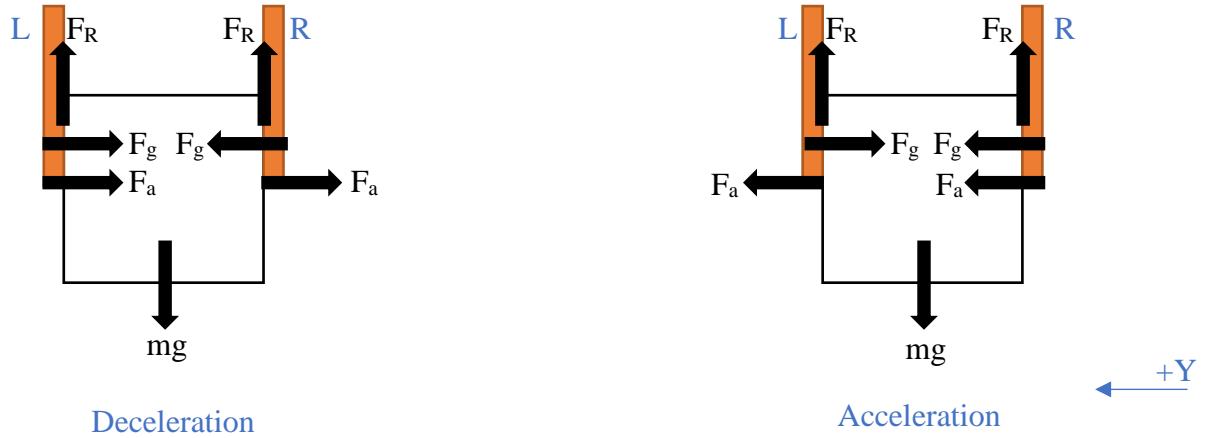


Figure 4.21: Free body diagram of forces acting on object and gripper due to acceleration and deceleration in Y direction

Assuming it moves in the +Y direction.

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

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

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

While in the acceleration phase,

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

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

The gantry starts moving in Y direction at 2.2s and stops at 3s. The changes in normal force experienced can be seen in Figure 4.22, where right gripper initially experiences more force due to the acceleration force and at the end experiences less due to the deceleration force.

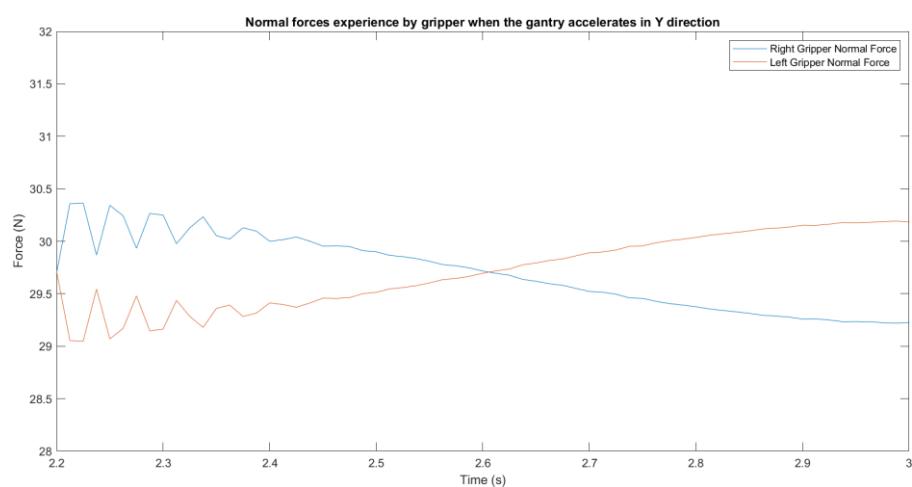


Figure 4.22: Graph showing the change in net force experienced by gripper finger due to acceleration and deceleration in Y direction

5 XYZ Position Controller Design

With the gantry model constructed in Simulink, the next step is to build a controller to control the 3 axes for reach and return phase of grasping. Discussions from Chapter 2.2 and Ratcliff's thesis [NO.] shows that ILC is a great choice for the controller scheme for repetitive robots and Norm-Optimal is one of the best solutions as it converges quickly and is robust. This chapter will design a Norm-Optimal ILC controller to control the 3 axes movement. A quick summary of requirements for the controller design is as follows.

1. Reach and grasp object within 15 iterations. Therefore, reach phase should be completed under 10 iterations to ensure 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.

5.1 Trajectory profile generation 1

As discussed in previous chapter, the gantry robot axes can be controlled independently from each other. Therefore, for the 3 axes, 3 separate controllers will be used, each controlling 1 axis. The overview of each controller algorithm is shown in Figure 5.1.

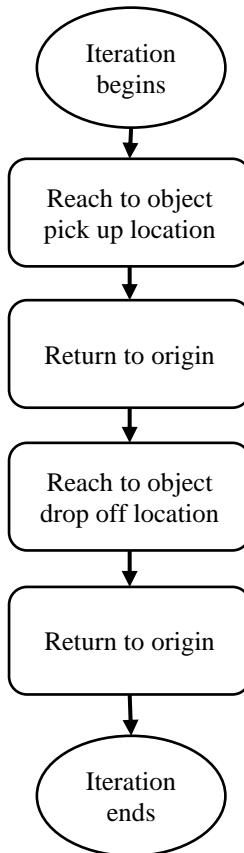
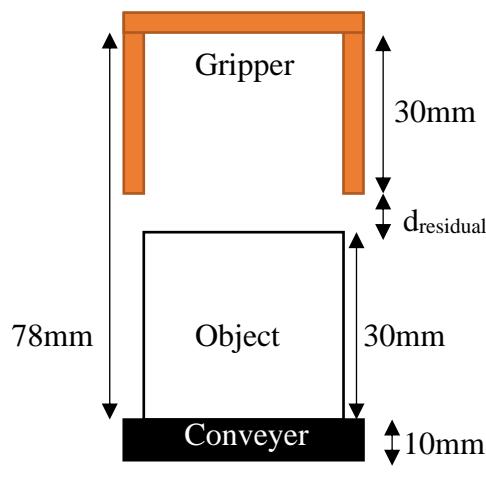


Figure 5.1: Basic control algorithm for 3 axes controllers

The object pick-up and drop off location can be different to mimic a real-world system of transporting items from one conveyer to another where the pick-up location is $0.5m \times 0.3m$ (Initial conveyer), while the drop off location is $0.3m \times 0.1m$ (Destination conveyer). The distance Z axis should travel to grasp the object is $0.038m$ and to drop the object is at $0.028m$, the calculation is shown in Figure 5.2.



Initial conveyer has a height of 10mm, and destination conveyer has a height of 20mm.

$$d_{residual} = 78 - 30 - 10 = 38mm$$

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

$$\begin{aligned} d_z &= d_{residual} + obj_{mid} + margin \\ &= 38mm + 15mm + 5mm \\ &= 58mm \end{aligned}$$

Destination conveyer is 10mm higher thus $d_z = 28mm$.

Figure 5.2: Z axis travel distance calculation

The complete profile of the 3 axes trajectory is shown in Figure 5.3.

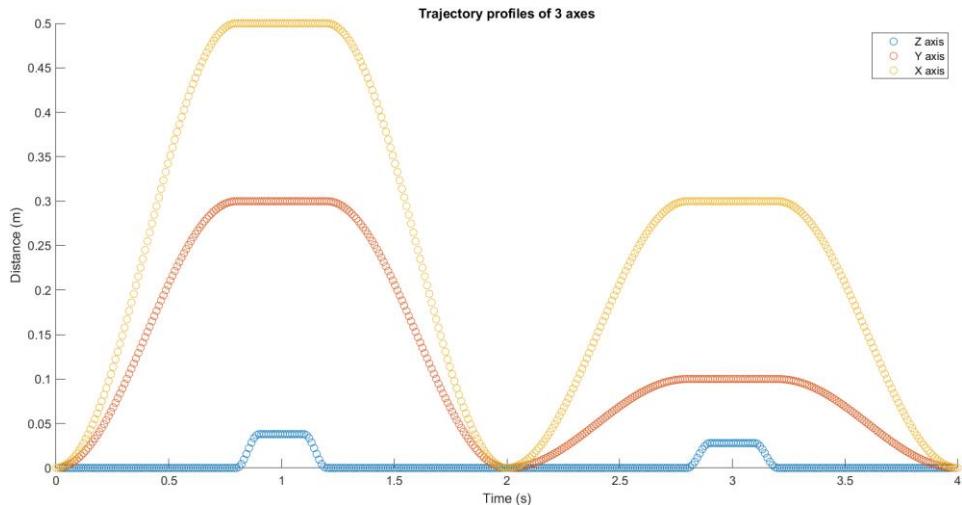


Figure 5.3: Trajectory profile for all 3 axes

To ensure smooth operations, the trajectory profile should have a gentle rise and fall to the final value. Rising profile is calculated using the following equation

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

where the falling profile is

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

where A is the steady state amplitude, d is the rise or fall time period, x is the time value input.

To ensure the complete trajectory can be completed in a short time, the deadtime for each axis when it reaches its final location should be as low as possible. This can be achieved by moving multiple axis together such as X and Y. However, Z axis can only be activated when X and Y axis are in place to ensure the object is not knocked over. Unfortunately, that means introducing deadtime for Z axis while X and Y axis moves and longer than desired deadtime for X and Y axis to ensure Z axis completes its operation.

Since one of the top priorities is to transport the object safely, the Z axis should only operate during X and Y axis deadtime as this is when the two axes are in position, thus there is additional deadtime. This can be seen in Figure 5.3. Table 5.1 shows the respective values for each axis where A denotes amplitude, d denotes rise and fall time, D denotes deadtime.

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

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

5.1.1 Tuning ILC Controller

As stated in the above subchapter, Norm-Optimal ILC will be use for controller. The formula for input update and error shown here are the same as equation 2.9 and 2.11.

$$e_{k+1} = r - y_{k+1} \quad (5.1)$$

$$u_{k+1} = u_k + (R + QG^T G)^{-1} QG^T e_k \quad (5.2)$$

The error vector would be the difference between the trajectory profile shown in Figure 5.3 and the actual distance moved. The plant, G , would be the plant deduced in Chapter 4 for each axis.

There are 2 ways to select R and Q values. The first is to use ‘lsim’ feature in MATLAB which simulate 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 response and pass the result to MATLAB environment using ‘To workspace’ block. ‘Simulink takes more time to generate response, for a 4 second each iteration simulation, it takes about 2-3 seconds to generate the response whereas lsim is instant. The generated response for both is similar as seen in Figure 5.4.

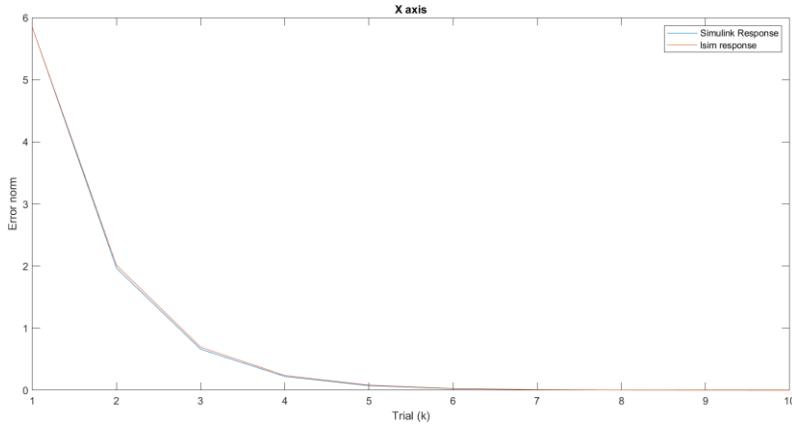


Figure 5.4: Comparison of simulated normalised position error for X axis between using Simulink and lsim

Therefore, lsim would be used to tune the values of R and Q to save time. Table 5.2 shows the R, Q and iteration where normalised position error gets less than 0.01 as indicated by DR4 in Chapter 2.1. X, Y and Z axis converges with low error within 10 iteration. A decision is made to choose R and Q values that ensures Z axis converges faster and thus allows the gripper to be able to grasp faster (less iteration).

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 origin after grasping the object but before putting it down at the destination. Typically, the distance to travel from initial location to destination is shorter than from initial location to origin then to destination. Therefore, this could cause unwanted stress in the system as it needs to accelerate hence force to complete the trajectory. Thus, it is best for the gantry robot not to go back to the origin after grasping the object. A new trajectory profile is generated and shown in Figure 5.5.

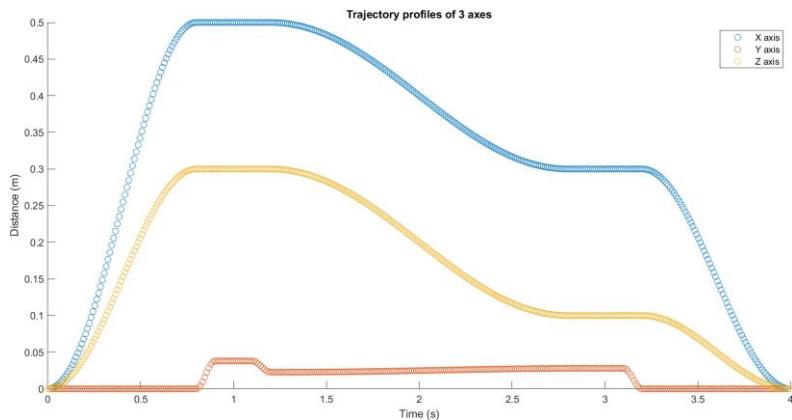


Figure 5.5: Trajectory profile for each axis controller to follow

It can be clearly 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 is less and thus put less stress on the gantry robot.

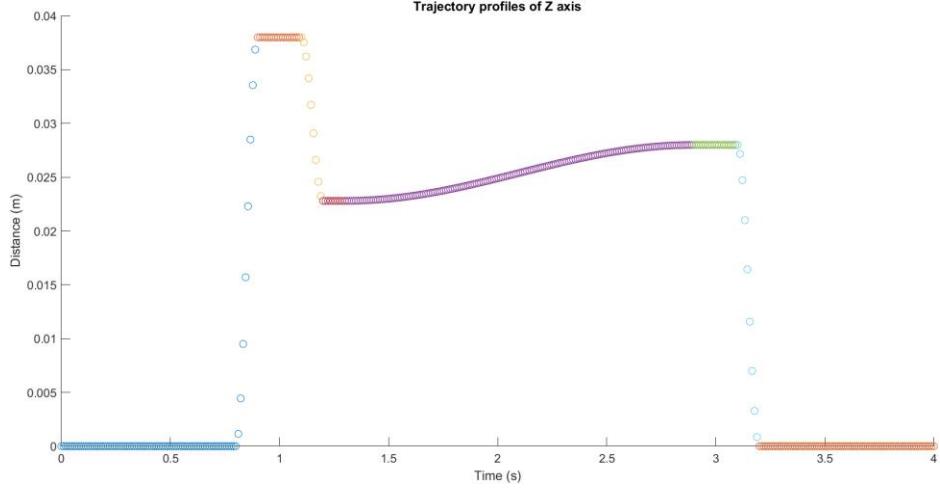


Figure 5.6: Trajectory profile of Z axis, each colour shows different parts of the trajectory profile generation, and the timing is given in Table 5.3

The Z profile has a significant change. To avoid the grasped object from being ‘dragged’ along the floor as the X and Y axis moves, the Z axis will lift the object up slightly before the deadtime of X and Y axis ends. A close up view of 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, where A denotes amplitude, d denotes rise/fall time, D denotes deadtime.

Table 5.3: Values for trajectory profile generation

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

This algorithm is capable of grasping the object and releasing at its destination effectively. However, before the Z axis will start to move in the first iteration. This can cause issues where the Z axis has converged with minimal tracking error while X and Y axis have not, hence caused the gripper to collide and harm the object. This phenomenon is captured in the 4th iteration shown in Figure 5.7.

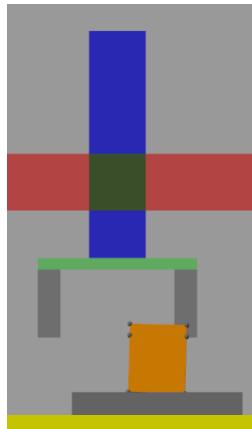


Figure 5.7: Gripper colliding with object due to X and Y axis having significant tracking error

5.3 Trajectory profile generation 3

In light of the issue experienced in previous subchapter, a new control algorithm was deduced where the Z axis shall ‘wait’ until X and Y axis have relatively small tracking error before starting to operate. The new algorithm can be illustrated using flow chart shown in Figure 5.8.

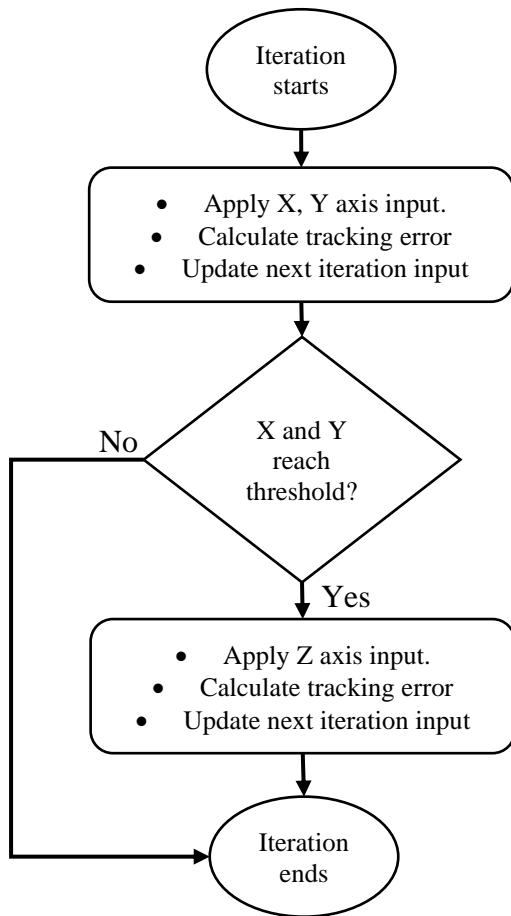


Figure 5.8: Flow chart of new control algorithm

The aim of this new algorithm is to ensure the X and Y axis final position are close to the actual object location. Therefore, the threshold set for X and Y axis can be calculated graphically as follows.

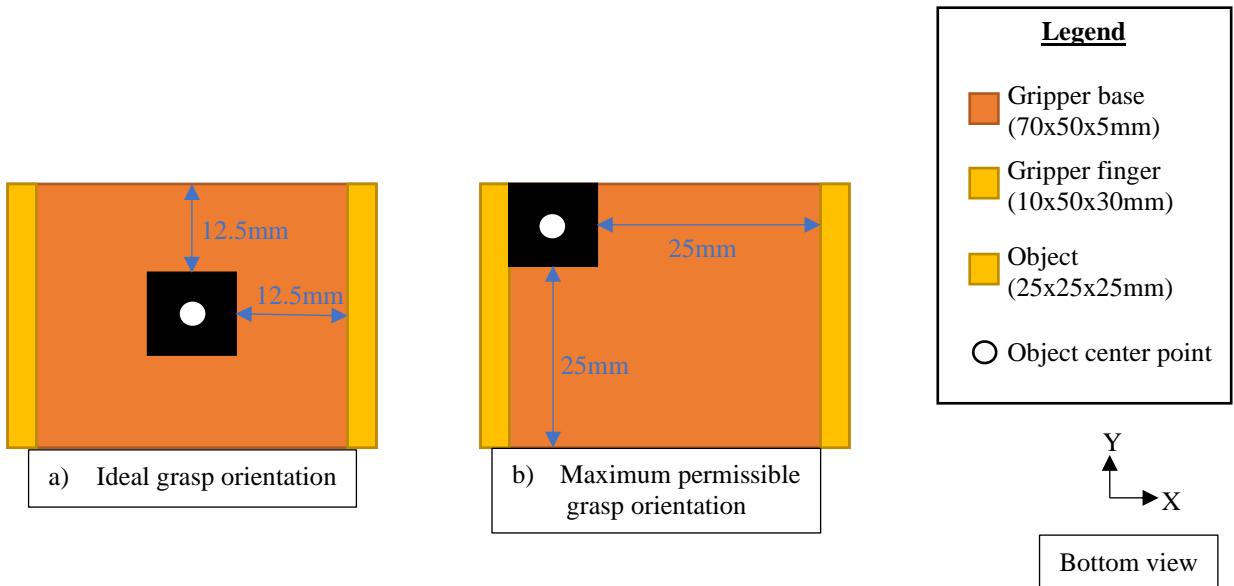


Figure 5.9: Illustration of gripper and object location for ideal grasping orientation and maximum permissible orientation

The trajectory profile of X and Y are generated based on grasping the object when it is at the centre of the gripper as illustrated in Figure 5.9a. But for the controller to converge to practically 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 the grasp, hence the Z axis can start to decent to allow the gripper to grasp the object. The earliest safe distance to allow Z axis to operate is distance from origin to object minus 25mm for both axis; for X axis, that would be the deducting 25mm from the amplitude of 500mm shown in Table 5.3.

A check is conducted to ensure X and Y axis are within the threshold shown in the flow chart. To get the most reliable reading to compare with the threshold, an average of the position feedback data is taken, this average is taken when the deadtime occurs (between 0.8s and 1.2s, and 2.8s and 3.2s as shown in Figure 5.5). Only if it is within the threshold, should the Z axis allow to operate. This ensures the object is safe in the event of overshoot or undershoot by X and Y axis.

Figure 5.10 shows the normalised error of the 3 axes, notice that Z axis normalised error remains constant during the first 4 iterations when 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 still within the 10 iteration limit mentioned at the beginning of the chapter.

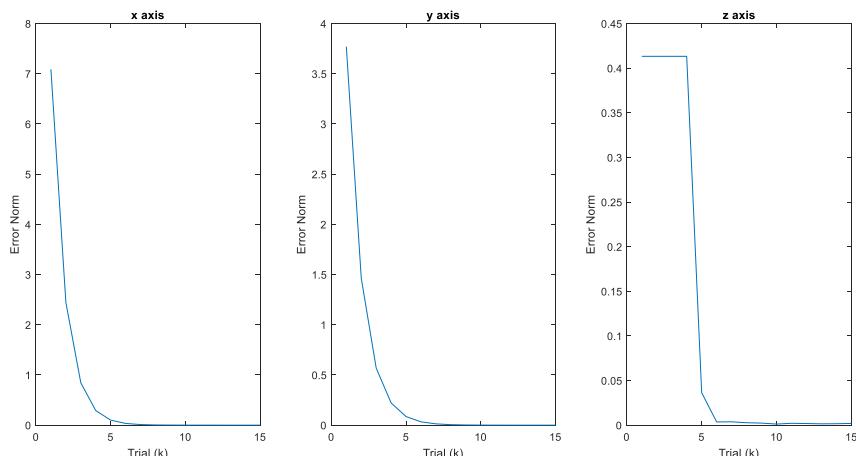


Figure 5.10: Normalised error for X, Y and Z axis with the new control algorithm

The iteration improvement of tracking is also shown in Figure 5.11 while Figure 5.12 shows the changes of input signal to realise the ideal tracking trajectory. This is evident that ILC learns from previous iteration mistakes and able to correct itself.

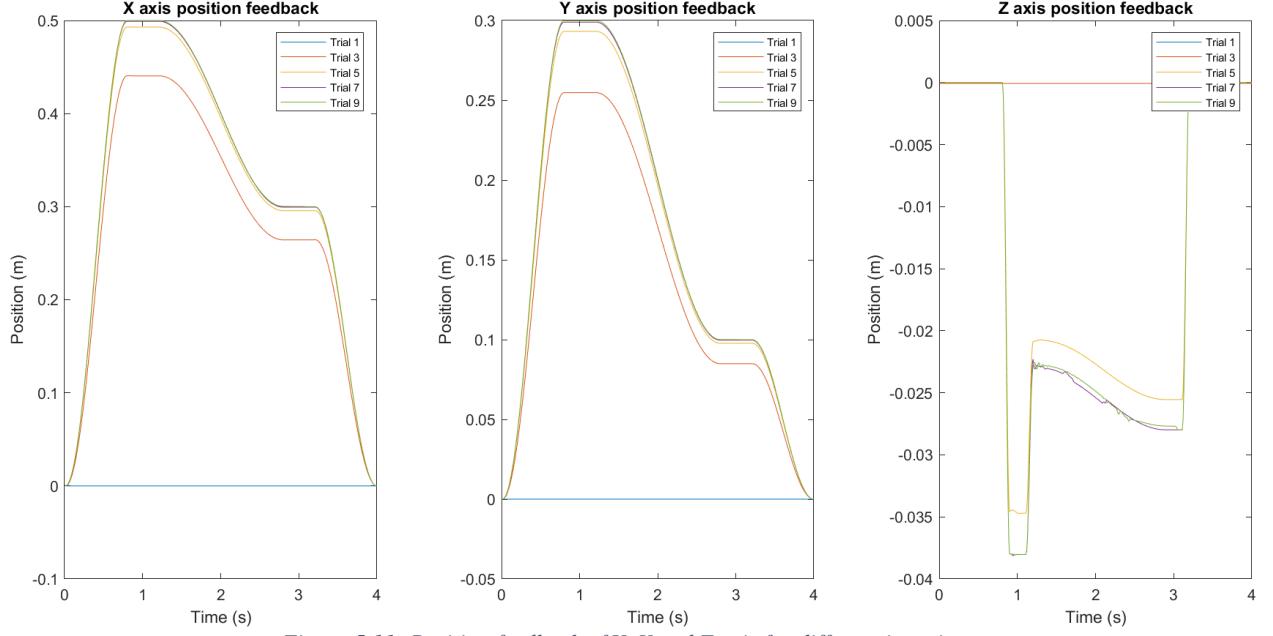


Figure 5.11: Position feedback of X, Y and Z axis for different iterations

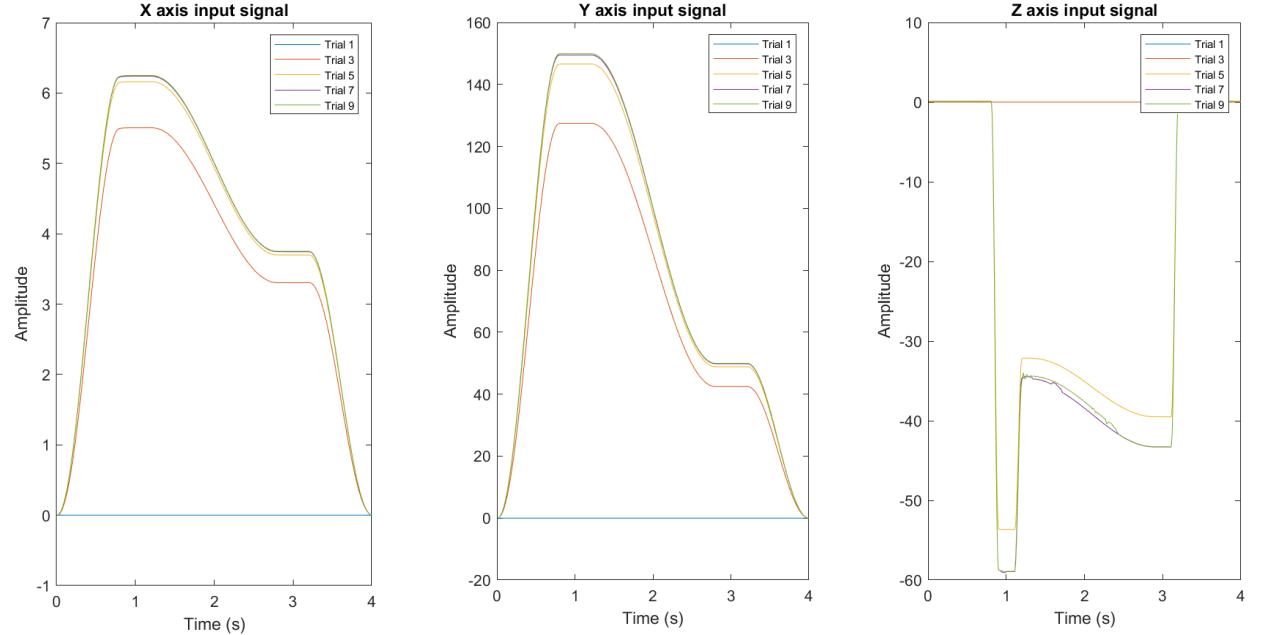


Figure 5.12: Input signal for X, Y and Z axis for different iterations

The Z axis input signal and position feedback is not smooth. This is due to the gripper being in the way when the X and Y axis moves as it assumes the gripper has grasped the object and nothing is affecting the path of Z axis. This can be resolved in the next chapter when the controller for the gripper is designed.

6 Gripper Controller Design

The rise and fall period of Z axis shown in Figure 5.6 is relatively short. Therefore, to reduce the strain put on the system, the total time period has been decided to increase from 4s to 5s, this allows longer deadtime for X and Y axis and reduced the middle transition period, thus giving more time for Z axis. The profile would still have 400 samples for 1 iteration giving it a sampling rate of 0.0125s. The new profile is shown in Figure 6.1.

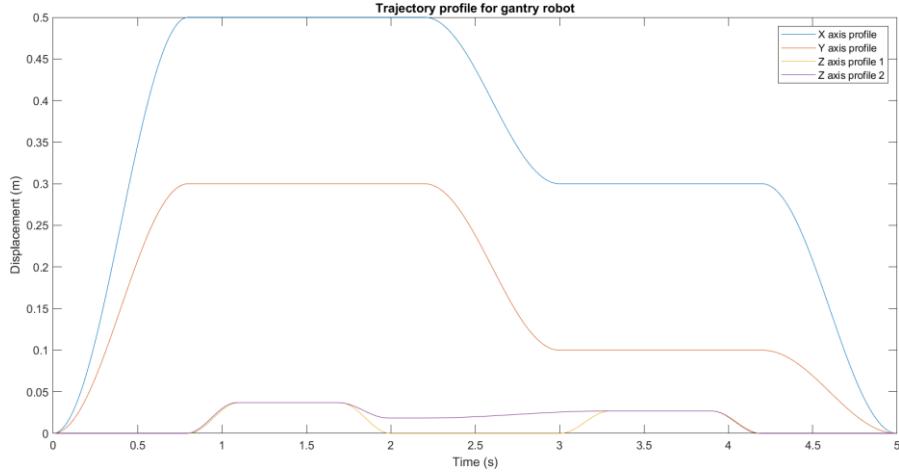


Figure 6.1: Gantry new profile with longer deadtime and longer iteration cycle

Z axis has 2 profiles. Profile 2 is the original profile structure discussed in previous chapter where the Z axis will retract a little to lift the object off the conveyer surface. However, the conveyer where the object will be picked up might not be flat and there might have parts which protrude out, hence it is safer to fully retract the Z axis prior to transport the object. The values to generate the profile shown in Figure 6.1 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	0.5	0.3	0.037	0.037
A2	0.3	0.1	0.027	0.037*50%
A3	-	-	-	0.027
d1	0.8	0.8	0.3	0.3
d2	0.8	0.8	0.3	0.3
d3	0.8	0.8	0.3	0.8
d4	-	-	0.3	0.3
d5	-	-	-	0.3
D1	1.4	1.4	0.8	0.8
D2	1.2	1.2	0.6	0.6
D3	-	-	1.0	0.2
D4	-	-	0.6	0.6
D5	-	-	0.8	0.8

As the project is interested in applying ILC to the gripper, the project will omit using Impedance control to move the axis to the final location and instead used the ILC controller produced in the last chapter plus modification of trajectory profile shown above.

6.1 Impedance Gripper

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

6.1.1 Position Control

To ensure the gripper fingers reaches 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 which is part of the Gripper PID block shown in the full schematic in Appendix B.

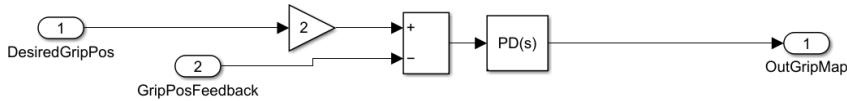


Figure 6.2: PD controller schematic for position control of gripper fingers

The PD controller has to fulfil the following criteria

- The position error has to decrease as fast as possible.

This corresponds to having small transient time when the system dynamics is subjected to step input. Figure 6.3 shows the response of the gripper finger dynamics using PID tuner.

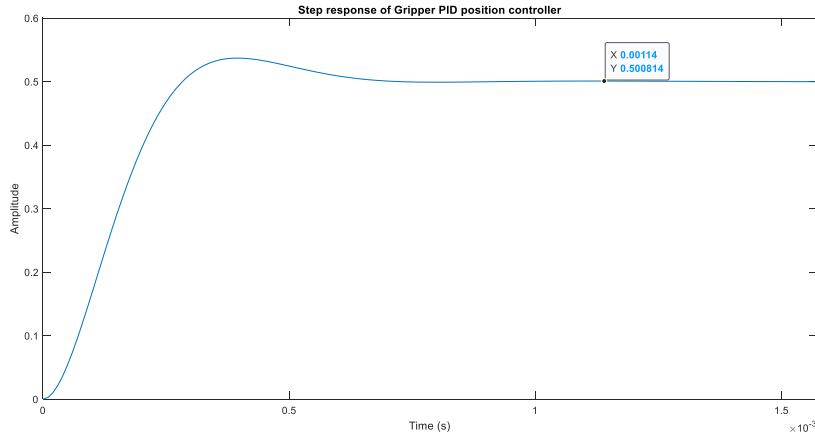


Figure 6.3: Step response of PD controller connected in series with gripper finger plant

As seen in the figure, the problem where the steady-state value does not return to 1 remains, therefore, it is crucial to have a gain of 2 added to the signal to cancel off the steady-state value effect. As evident from the step response, the gripper position error controller converges in a very short time (small transient time).

6.1.2 Force Control

Since there is a direct relationship between the gripper grasping position and force applied on the object. To ensure no excessive force is applied, the controller has to change the reference trajectory passed onto the PD controller to control the finger position. This requires an impedance controller, and the general schematic can be found in Figure 3.5 and this schematic is replicated in Simulink as shown in Figure 6.4.

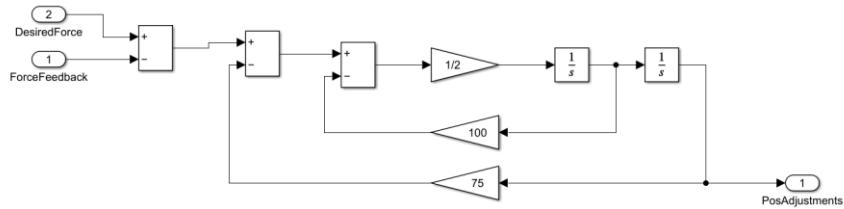


Figure 6.4: Schematic of Impedance Control

The desired force to grasp an object is calculated using Equation 4.11 while subjecting to a 20% increase to account for any uncertainties in modelling and calculation. The difference between this force and the measured normal force by placing force sensor on the gripper finger will be passed on to the impedance control where it would calculate an adjustment (PosAdjustment) to be added to the reference trajectory shown in Figure 6.5 to get the grasping force as close to the desired one as possible shown in Figure 6.6.

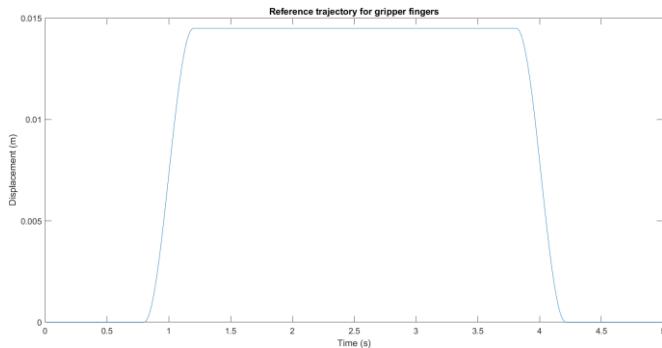


Figure 6.5: Reference trajectory for gripper fingers to follow

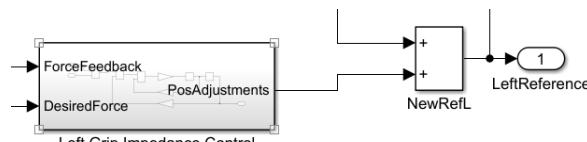


Figure 6.6: Input and output of impedance controller

The values of M, B and K needs to be tuned to get the best response. Figure 6.7a shows the measured gripper force when M is set to 2, B is set to 300 and K is set to 200. For the gripper to skim the surface of the object, it has to travel 0.0115m. A conservative approach is taken where the gripper is set to close a further 0.003m to grasp the object firmly. This would cause the gripper to always able to grasp the object but use excessive force resulting in a spike shown in Figure 6.6. The sine wave like phenomenon around 2s to 3s is caused when the object moved in Y direction as explained in chapter 4.2.2. In order to reduce the initial spike, M, B and K values are chosen based on trial and error. Some of the test values can be found in Table 6.2 and some of the force measured is plotted in Figure 6.7. Care has to be taken when selection the values, as demonstrated by Figure 6.7a to 6.7c, although the

max amplitude of the spike decreases, it produces a sharp valley at some point, this can be due to the gripper behaving like a jelly swinging side to side when moved which could cause the gripper to drop the object.

Table 6.2: 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

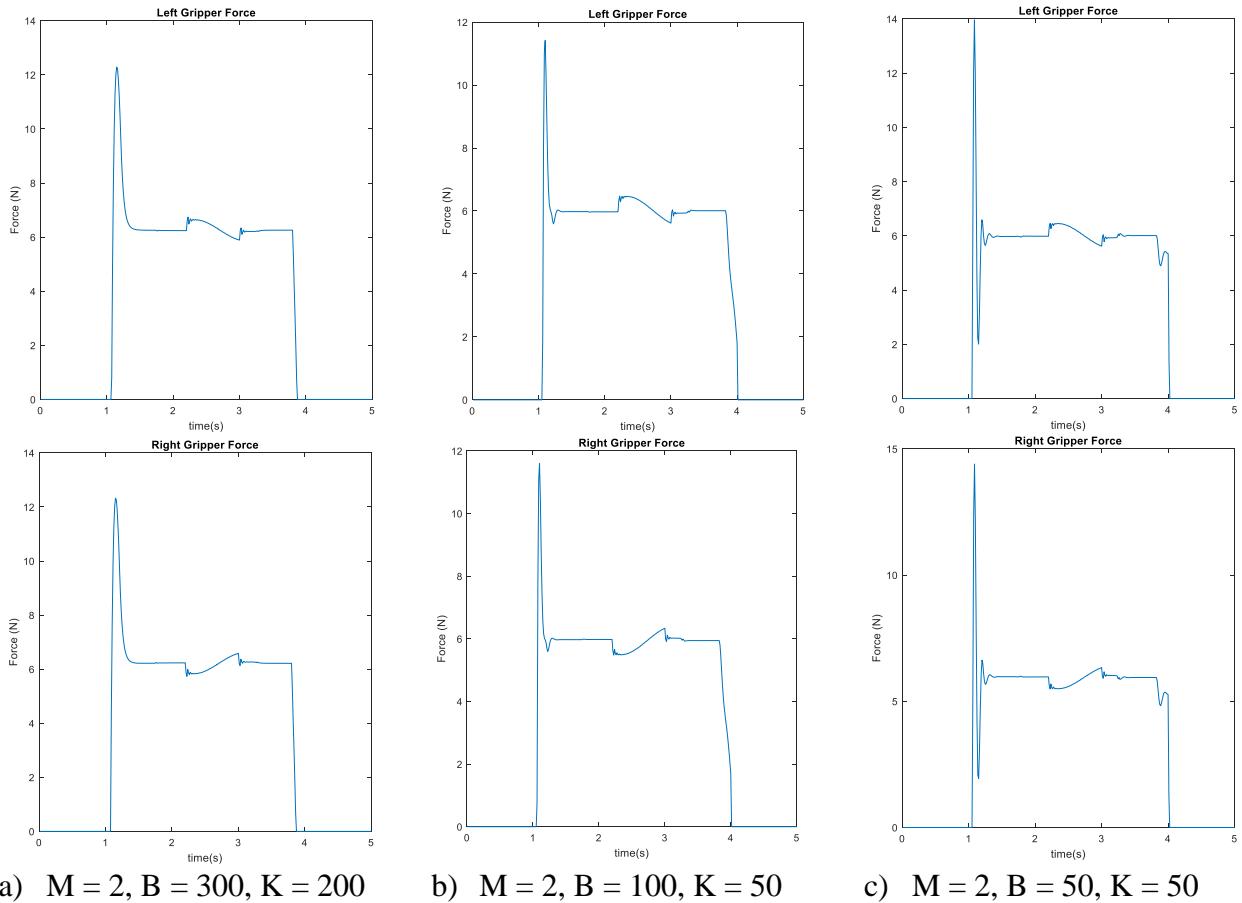


Figure 6.7: Interaction forces between gripper finger and object under different B and K values.

To ensure the gripper is able to grasp the object firmly, the X, Y and Z axis should be at a reasonable position (within certain grasping volume) before the gripper is allowed to grasp. Therefore, a logic system will need to be in place together with the schematic shown in Figure 6.6. The flow chart for such system is shown in Figure 6.9 which proceed the one in Figure 5.8. The full schematic of impedance controller is shown in Figure 6.8, with appendix B showing a more detailed version.

The threshold for Z axis is calculated similarly to Figure 5.9, where now the average position feedback has to be within 97% of the final value.

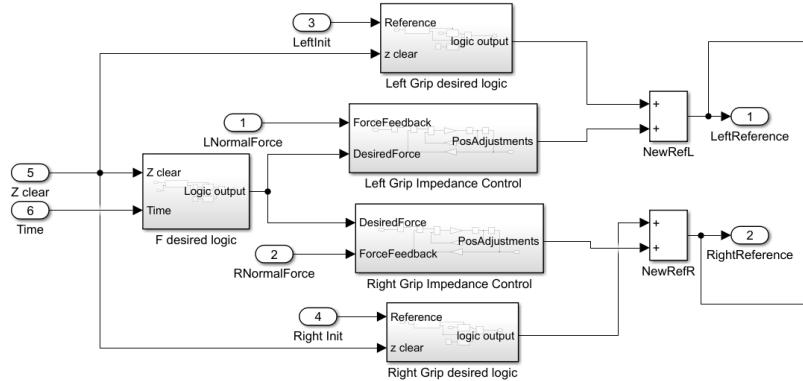


Figure 6.8: Full schematic of Impedance Control, without PD controller

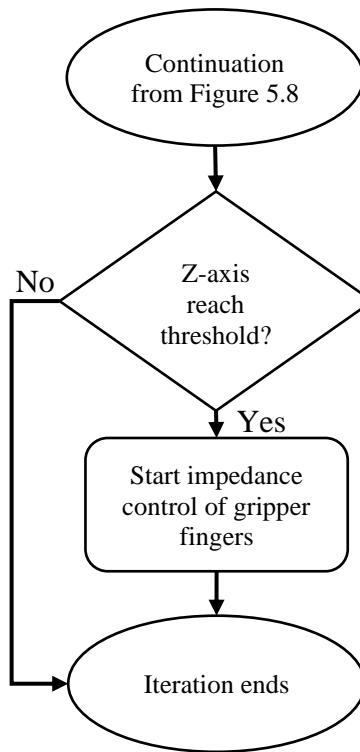


Figure 6.9: Flow chart of full impedance control algorithm

6.2 ILC Gripper

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

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

6.2.1 Position Control

The gripper fingers will use the same ILC update equations as the gantry controller and are denoted at Equation 5.1 and 5.2. The plant has been identified chapter 4.1.3. To tune the values of R and Q,

similar steps used in chapter 5.1.1 are used and the best value found are $R = 0.00005$, $Q = 2000$. This allows the controller to converge with normalised error less than 0.01 in just 3 iterations as shown in Figure 6.10. Therefore, after considering that X, Y and Z axis converges at the 6th iterations, theoretically, the gripper will converge at the 9th iteration after X, Y and Z axis converges which is in line with DR4. Figure 6.10 shows the normalised error plot for the system.

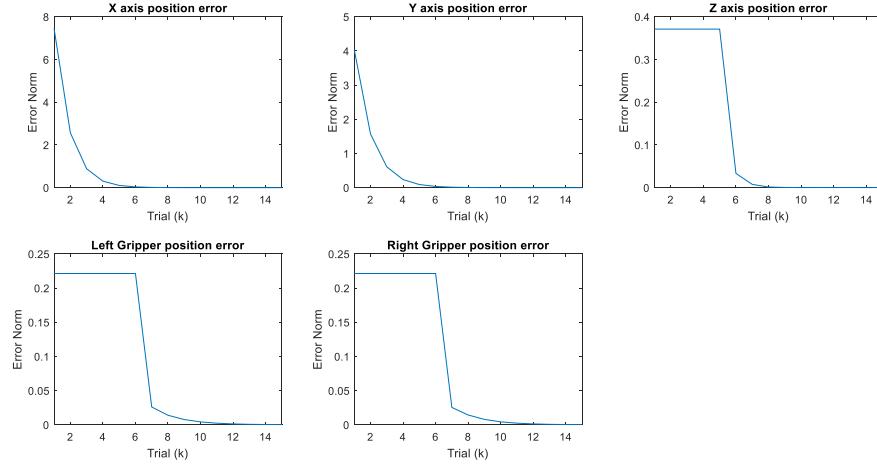


Figure 6.10: Normalised position error for whole gantry system

From chapter 5.3, there is a logic system that only permits Z axis to operate if X and Y axis are within a threshold. Similarly, in chapter 6.1.2, the impedance control only starts closing the gripper if Z axis are within a certain threshold. ILC would use the same concept with the same threshold but with some minor modification. As ILC only updates the input vector at the end of the iteration, the information passed to the controller to make decision can be seen as delayed by 1 iteration, therefore, it is crucial that the gripper has a firm grasp of the object before transporting. As a result, 2 reference trajectory profile is generated for gripper fingers to follow. The first one would grasp and release to obtain the force reading when the gripper reaches the object. Only when the value exceeded a force threshold, should the second profile be used. The second profile is the same as seen in Figure 6.5, where it is able to grasp the object while transporting. The overall profile for the whole system is shown in Figure 6.11, while the comparison between the 2 gripper profiles is shown in Figure 6.12.

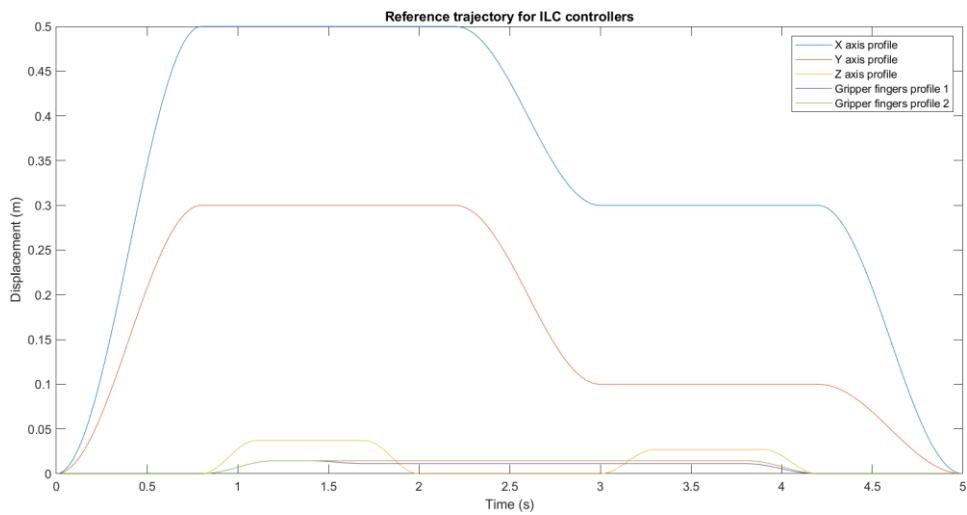


Figure 6.11: Reference trajectory profiles for X, Y, Z axis and gripper fingers to follow

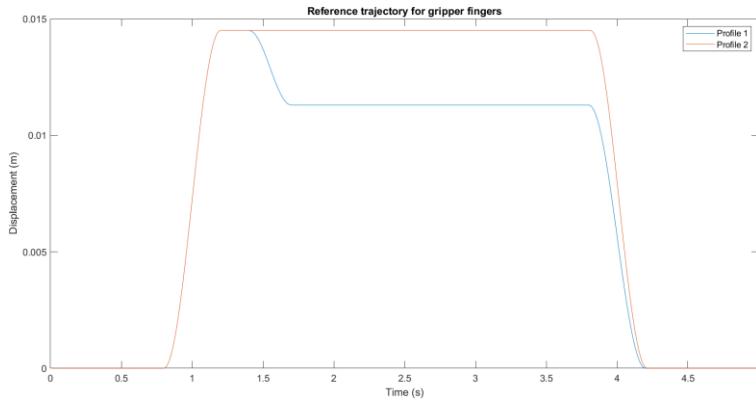


Figure 6.12: Two different reference trajectory for gripper fingers controller to operate under different conditions

The modified flow chart of the new algorithm is illustrated in Figure 6.13.

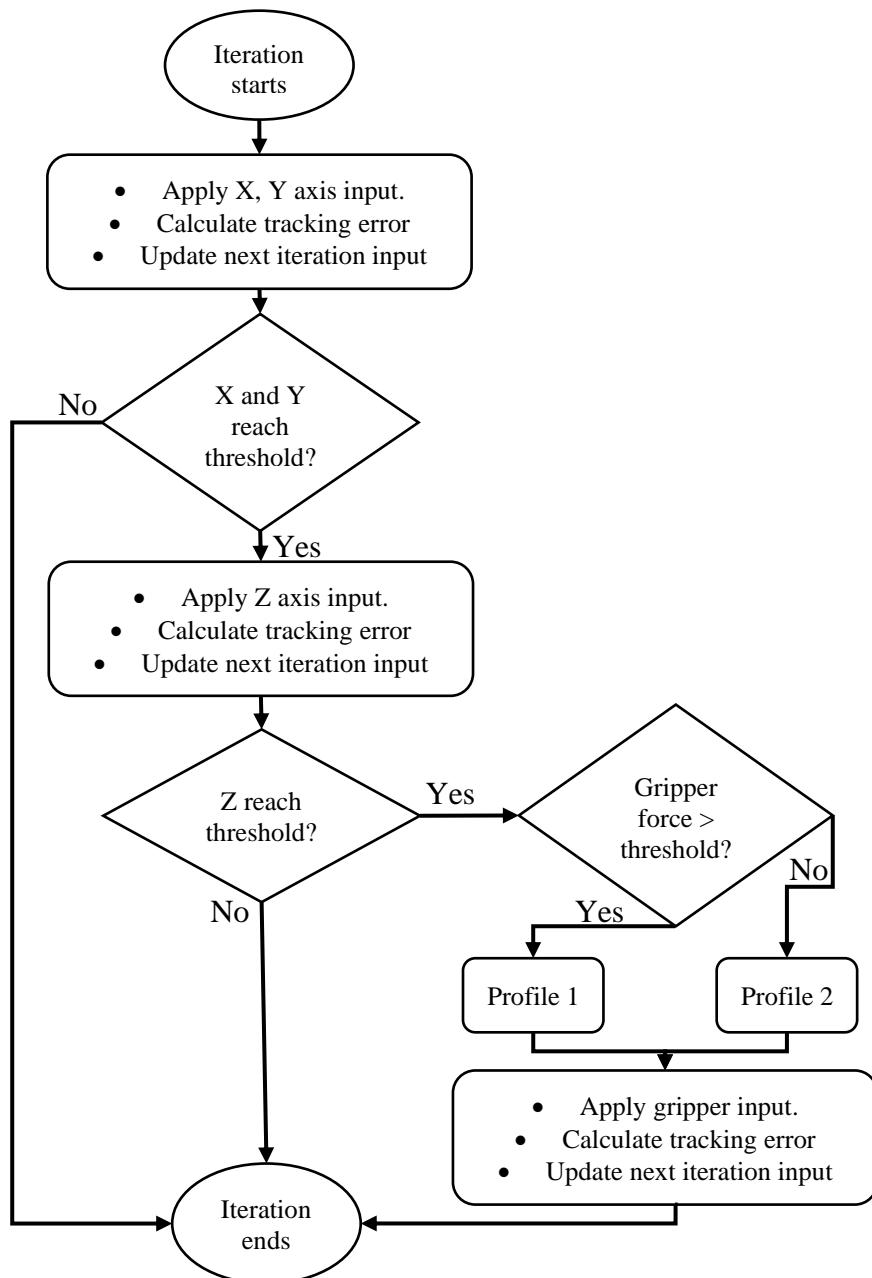


Figure 6.13: New control algorithm with different grasping profiles

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 Equation 4.11 to account for any uncertainties or modelling errors that may arise.

6.2.2 Force Control

In order to ensure the gripper does not use excessive force, a force control is needed. As mentioned, and justified in chapter 3.3.2, a simple gain ILC update would be used. This requires the flow chart shown in Figure 6.13, to be updated to the one shown in Figure 6.14. In the interest of space, duplicated areas of the flow chart have been omitted.

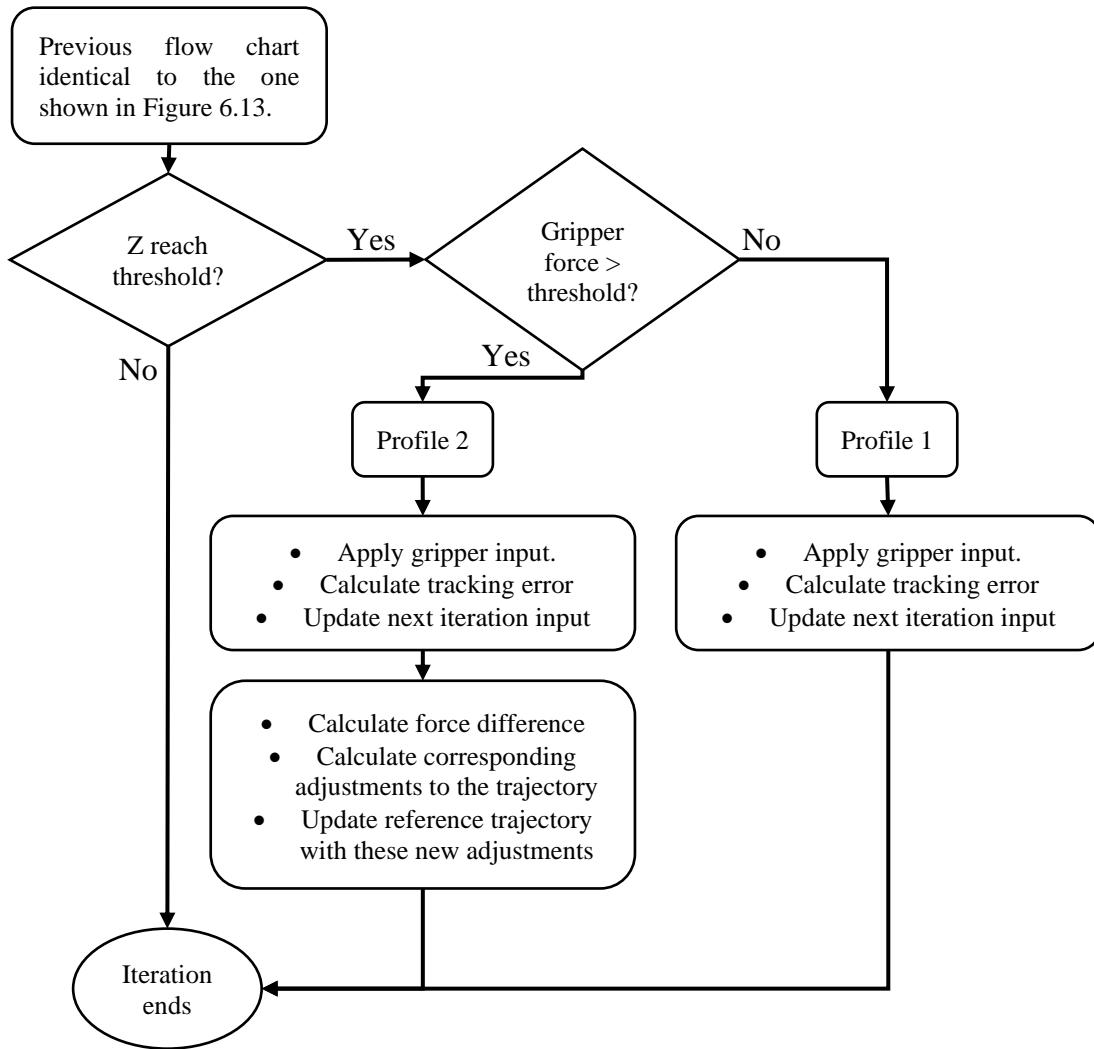


Figure 6.14: New flow chart showcasing new algorithm with features reducing grasping force

As the gripper only grasp the object during part of the trajectory, only the forces measured at the deadtime of profile 2 are used.

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

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

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

As the interaction force is modelled like a spring-mass system in Simulink explained in Chapter 4.2, the force is directly proportional to the distance two bodies overlap. This can be calculated using the following formula

$$F_{applied} = d_{overlap} * stiffness \quad (6.2)$$

The stiffness is set to be 10000N/m, therefore in profile 2 where the gripper moves 0.0145m and for the gripper and object to touch on the surface, the gripper has to move 0.0115m, this leaves the overlap distance to be 0.003m, hence the force applied is 30N. For an object mass of 2kg, using equation 4.11, the F_{min} to lift the object is 4.91N. F_{diff} can be obtained when the value of F_{min} is placed in equation 6.1 and $F_{measured}$ is the force sensor measure. Combining that with equation 6.2, it is possible to calculate the displacement reduction for the fingers to reduce the force applied on the object and is shown in equation 6.3.

$$d_{diff} = \frac{F_{diff}}{stiffness} \quad (6.3)$$

The changes are applied to the grasping profile 2, thus reducing the peak amplitude of the profile. d_{diff} can be regarded as an error signal and thus the update can be carried out using ILC using equation 3.4, thus the update equation is as follows

$$\mathbf{ref}_{k+1}(t) = \mathbf{ref}_k(t) + \beta(t) \mathbf{d}_{diff}(t) \quad (6.5)$$

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 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.

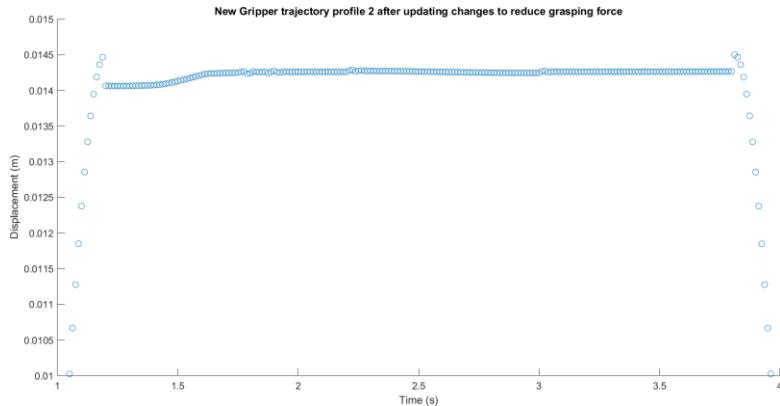


Figure 6.15: Non-smooth edge of the new reference profile via ILC

Therefore, to produce a new grasping profile for the gripper, the whole profile has to regenerate with the new magnitude at 1.2s and 3.8s using equation 5.1 and 5.2 producing profile seen in Figure 6.16.

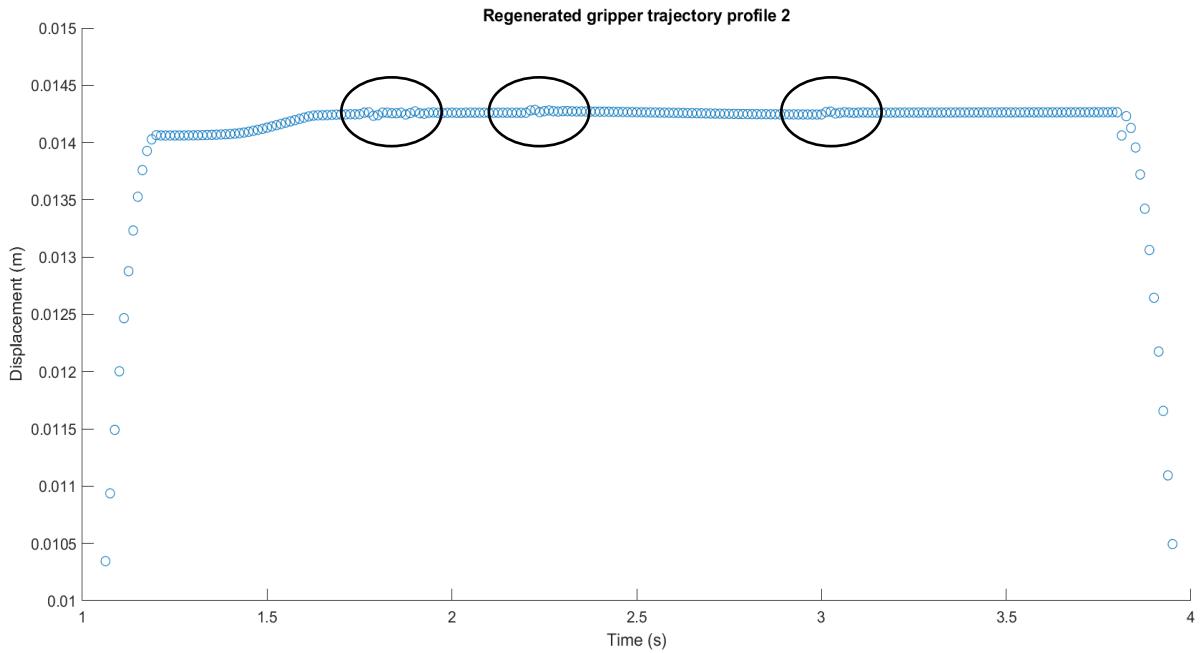


Figure 6.16: Non-smooth data within the deadtime of the grasping trajectory profile

Unfortunately, part of the profile circled in Figure 6.16 is not smooth, this is caused by discrete controllers and the effect superimpose each other which could result in the gripper behaving in a destructive way, similar to a noise effect. 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 increase computational complexity. Fortunately, another solution is to apply a non-causal filter to the signal as all the data points within that iteration are known. This can be done using MATLAB built in function call ‘smoothdata’ which returns a moving average using a fix window size. The result of using a filter and not using one is shown in Figure 6.17.

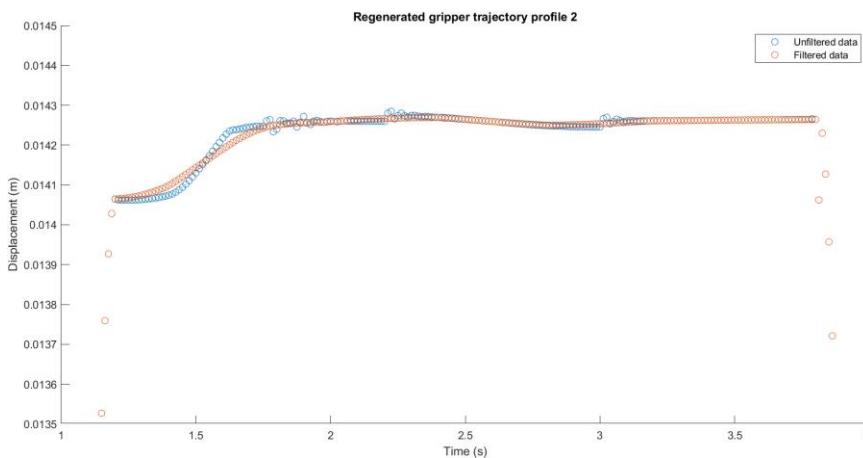


Figure 6.17: New trajectory profile before and after passing through a non-causal filter

With the control algorithm fully implemented, the performance of the ILC controller with force and position control is investigated. Figure 6.18 shows the normalised position and force error. All the position controller converges at roughly 10th iteration with minimal error. The average force the object experienced is shown in Figure 6.19. Recall that DR 6 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 start grasping at 7th iteration and after 17 iterations, both fingers is able to accomplish it.

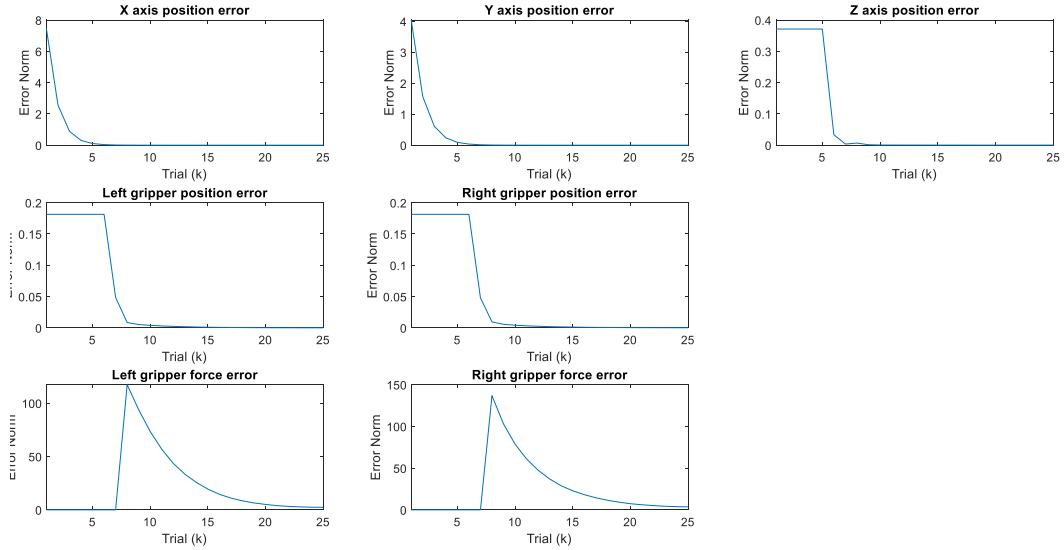


Figure 6.18: Normalised position and force error for each part of the gantry robot controller by ILC controllers

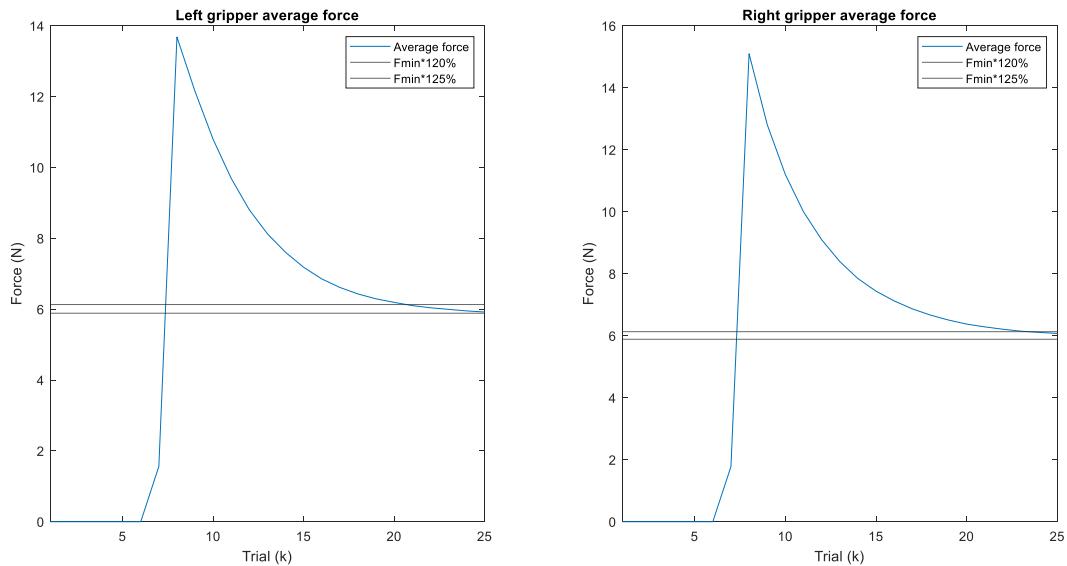


Figure 6.19: Average forces experienced by the object and the 20% and 25% increase in force over minimum force

It can be seen in Figure 6.20 that initially at 7th iteration, profile 1 is in action and is testing to see if the gripper has sufficient force to grasp the object securely. Once it does in the 8th iteration, profile 2 is used and after position and force ILC updates, at 40th iteration this is the ‘best’ grasping profile the gripper should follow to realise all the DR. The effect of the phenomenon of acceleration and deceleration force identified in chapter 4.2.2 can be seen at the 40th iteration peak and valley in Figure 6.20, where the gripper profile changes to ensure it considers those forces.

One would expect the force plot shown in Figure 6.21 to be flat since there is force control, this disagreement could be caused by a few reasons. The first is due to applying non-causal filter in force control as this makes the data smoother at the expense of force control. The other could be due to the small error signal as Norm-optimal ILC converges to 0 error when it reaches infinite iteration. Despite this issue, the gripper is able to grasp the object firmly due to the 20% margin given and the fact that left and right gripper complement each other as shown in Figure 6.21c.

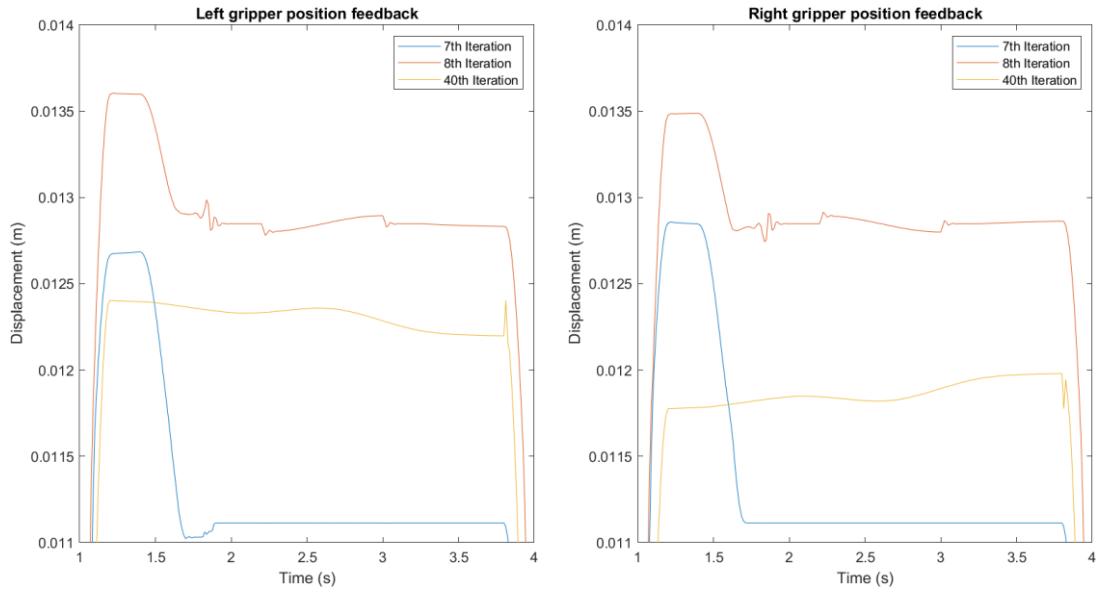


Figure 6.20: Left and Right gripper position feedback for 7th, 8th and 40th iteration

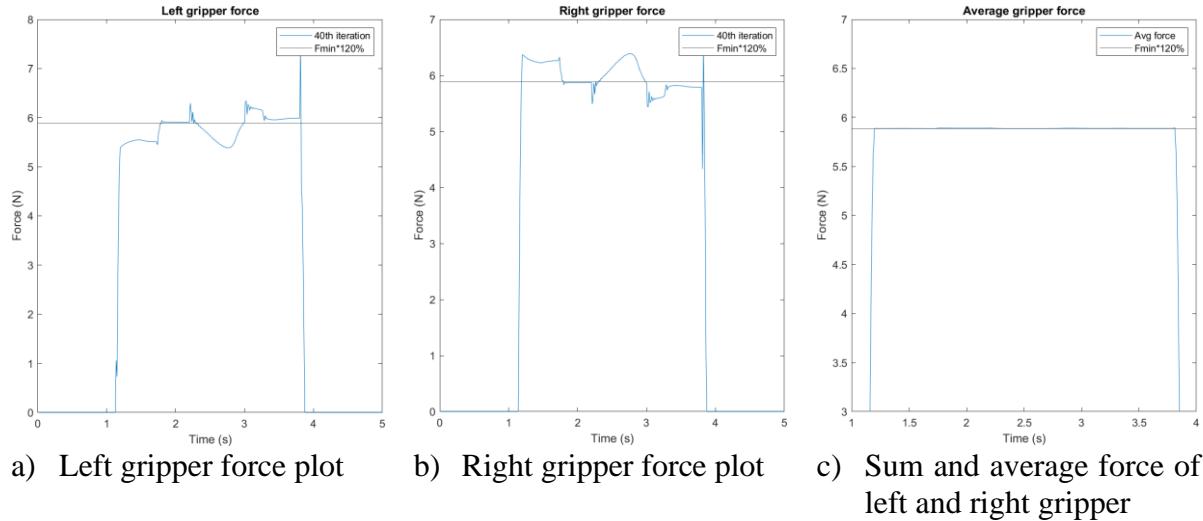


Figure 6.21: Gripper forces at 40th iteration

6.2.3 Updating Controller Model

As the aim of the gantry robot is to grasp object ranging from 0.2kg to 2kg. The bound is object weighing 0.2kg and 2kg. Thus, investigation of the effect of updating the plant will be carried out at the upper and lower bound as the effect of the rest would lie between the two bounds. The plant is updated by adding the mass of object to each part of the gantry mass shown in Table 4.1 and the plant transfer function is recalculated using equation 4.7.

Due to Z axis (3kg) and gripper (1kg) being on a lighter side compared to X (47.7kg) and Y (21kg) axis [39], the additional object mass could cause the system dynamics to change significantly. Figure 6.22 show the grasping force experienced by the object of 2kg when the plant is updated versus not updated. The force feedback can be seen to be oscillating for non-updated plant, this is caused by extreme mismatch of the plant and actual system, causing the next iteration input vector to be significantly incorrect.

Object of 0.2kg force is shown in Figure 6.23. There is little to no effect of updating the plant as the mass is too light to cause significant difference in system dynamics.

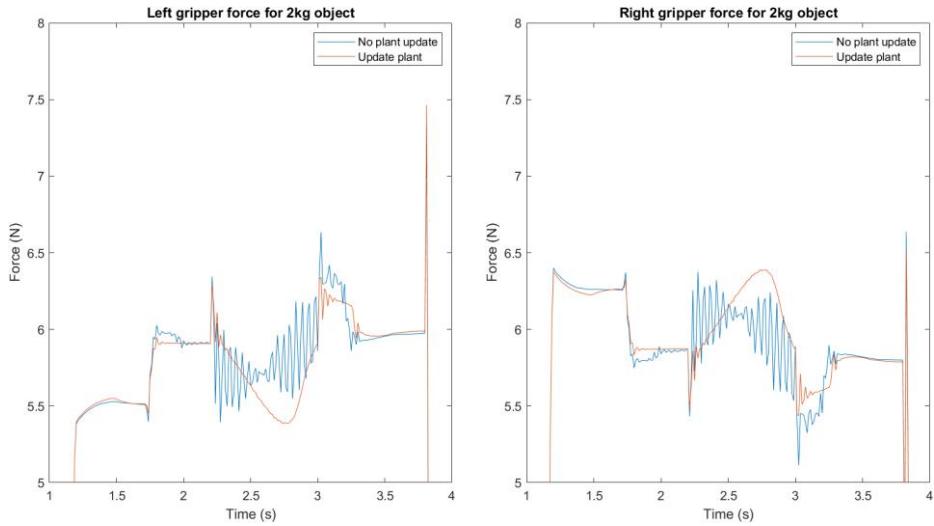


Figure 6.22: Forces experienced by the object when for scenario where plant is updated and not updated with the object mass of 2kg

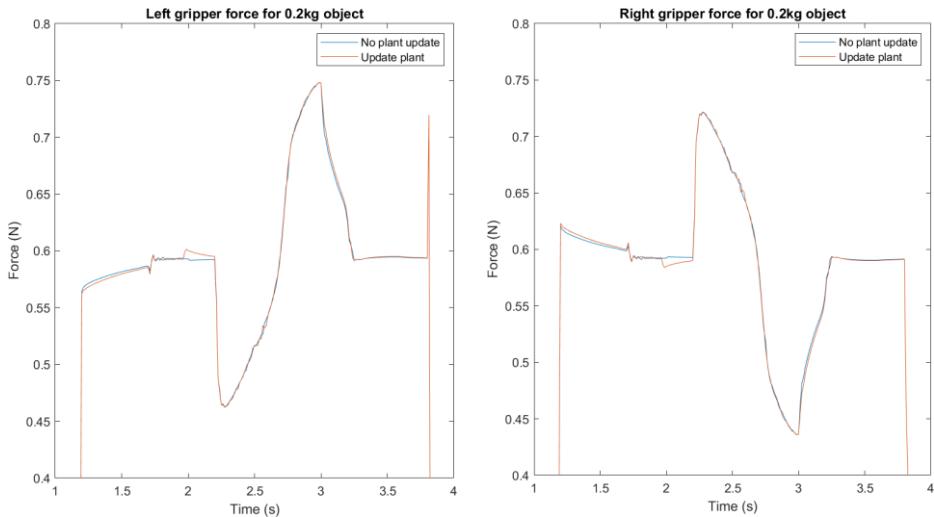


Figure 6.23: Forces experienced by the object when for scenario where plant is updated and not updated with the object mass of 0.2kg

The effect of updating plant model for both 0.2kg and 2kg object is shown in Figure 6.24. Similar to the force scenario, the consequences of not considering the 2kg object dynamics is quite serious as the Z axis vibrates, essentially shaking the object off. Whereas there is no such issue for 0.2kg object. This is caused by the system dynamics change significantly when 2kg object is grasp.

Both Figure 6.22 and 6.24 highlighted that the issue of unstable grasp only occurs when the object is grasped, this shows that it is important to consider major changes to the system dynamics. On top of graphs, simulation in Simulink shows that the object slid while transporting as seen in Figure 6.25.

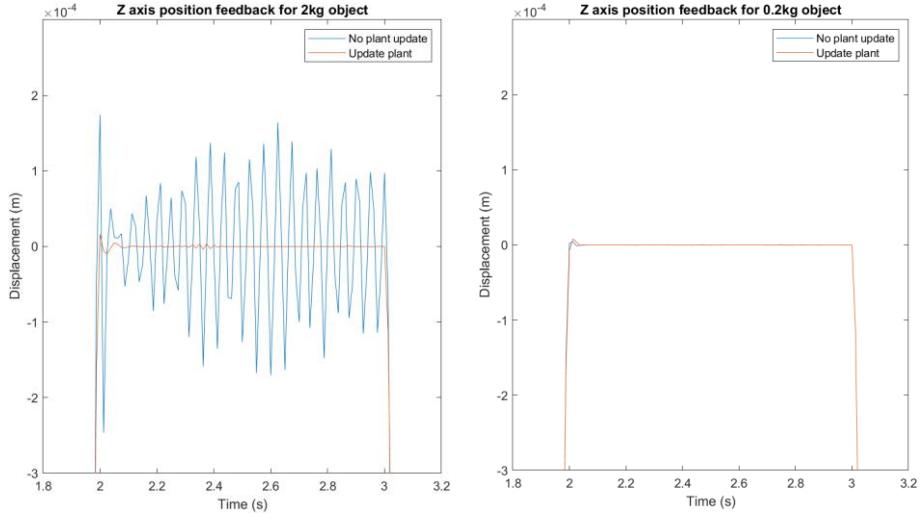


Figure 6.24: Z axis position feedback for 0.2kg and 2kg object for updated and non-updated plant

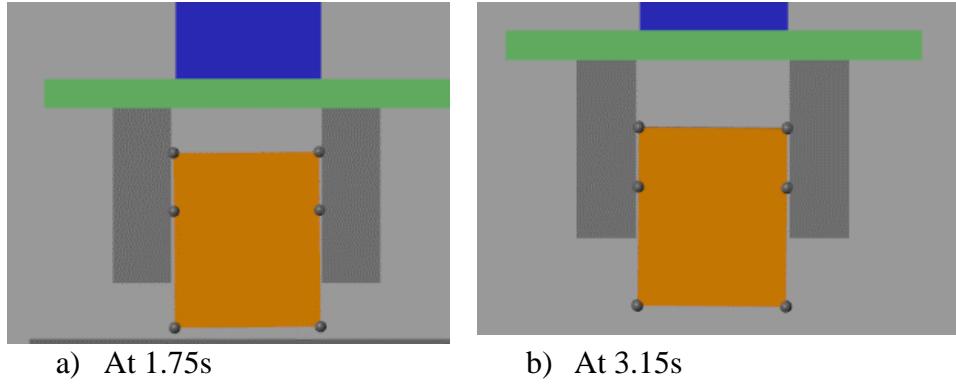


Figure 6.25: 2kg object sliding when being transported using non-updated ILC controller. Both figures are to the same scale.

6.3 Summary

Both Impedance and ILC is able to grasp object of mass ranging from 0.2kg to 2kg. However, the disadvantage of Impedance control as shown in Figure 6.7 is the huge initial spike which could damage the goods upon grasping. ILC is able to eliminate spike, but the force feedback is relatively not ‘flat’ when comparing Figure 6.21a and 6.21b to 6.7. Besides that, it is crucial to update the plant of the gripper and Z axis when using a model based ILC controller. If it is not updated, the robot might have some oscillation which could result in the object sliding off or dropped by the gripper. Although Norm-optimal ILC has been known to be a more robust ILC controller, the 2kg object proves to be a little too heavy to cause some sliding issue, while no such issue is present when grasping the 0.2kg object.

7 Controller Disturbance Test

7.1 Test Setup

The disturbance will be introduced to the system via method discussed in Chapter 3.4. In total, there are 3 different disturbances. Firstly, a constant initial error added to the 1st iteration to mimic initial error. Secondly, a constant initial error added to the 30th iteration mimicking error occurring during operation. Lastly, an impulse error showing the effect of external force interacting with the system. The first two will be a constant disturbance with magnitude of 5% and 10% of the highest amplitude input signal under ideal condition shown in Figure 7.1, the values are shown in Table 7.1. Impulse error will have short rise and fall time seen in Figure 7.2 and can be found in Table 7.1 as well.

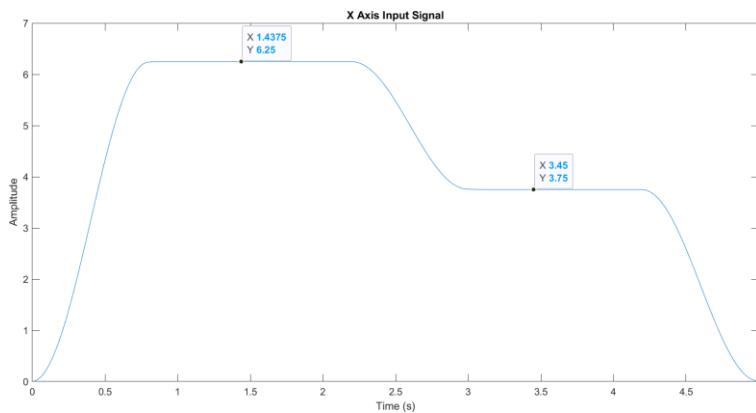


Figure 7.1: X axis input signal at 40th iteration under no disturbance scenario

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

	Constant error				Impulse error			
	X	Y	Z	Gripper	X	Y	Z	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
Dead time	-	-	-	-	0.2	0.2	0.2	0.2
Fall time	-	-	-	-	0.2	0.2	0.2	0.1

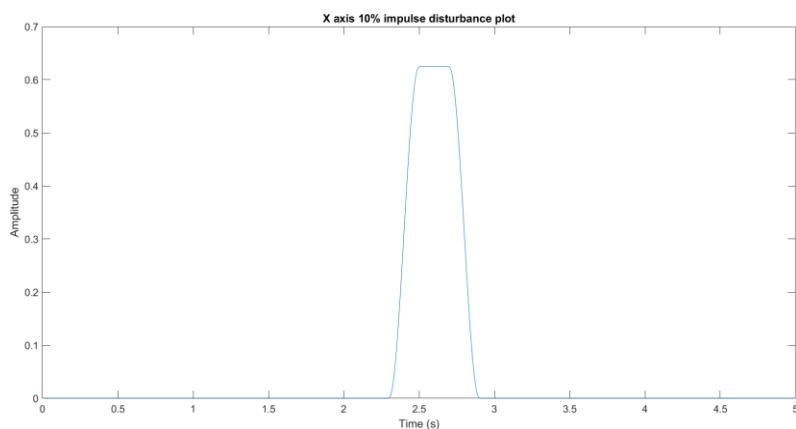


Figure 7.2: Example of Impulse disturbance, where in this case it is 10% for X axis

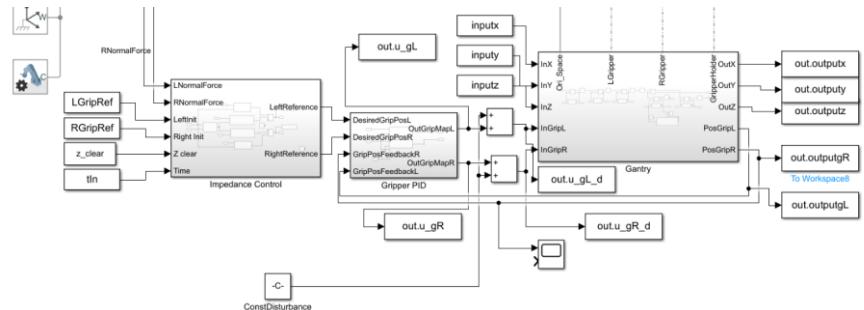
For ILC based controllers, the disturbance will be added to the input prior to passing the value to Simulink as demonstrated in Figure 7.3a, while for impedance-based controller, the disturbance will be added in Simulink prior to passing the value to PD controller as seen in Figure 7.3b.

```

if (i >= error_start)
    u_z_d = u_z+disturbance_z;
else
    u_z_d = u_z;
end
inputz = timeseries(u_z_d,t);

```

a) Code showing disturbance added to ILC controller input



b) Simulink schematic where disturbance is added using blocks prior to the signal passing to the PD controller

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 Error

This section will investigate the effect of the first 2 disturbances described at the beginning of previous subchapter.

7.2.1 Error at 1st Iteration

Table 7.2 highlights the observations from the test via both 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 affect 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%	<ul style="list-style-type: none"> Normalised error same as ideal at 4th iteration Object grasped at 8th iteration 	<ul style="list-style-type: none"> Normalised error same as ideal at 4th iteration Object grasped at 8th iteration 	<ul style="list-style-type: none"> Normalised error same as ideal at 6th iteration Object grasped at 8th iteration 	<ul style="list-style-type: none"> Normalised error same as ideal at 7th iteration Object grasped at 8th iteration 	<ul style="list-style-type: none"> Behaves identically to Ideal case
10%	<ul style="list-style-type: none"> Normalised error same as ideal at 4th iteration Object grasped at 8th iteration 	<ul style="list-style-type: none"> Normalised error same as ideal at 4th iteration Object grasped at 8th iteration 	<ul style="list-style-type: none"> Normalised error same as ideal at 6th iteration Object grasped at 8th iteration 	<ul style="list-style-type: none"> Normalised error same as ideal at 7th iteration Object grasped at 8th iteration 	<ul style="list-style-type: none"> Behaves identically to Ideal case
Ideal	<ul style="list-style-type: none"> Object firmly grasped at 8th iteration 				<ul style="list-style-type: none"> Object firmly grasped at 7th iteration
	<ul style="list-style-type: none"> Gripper starts grasping object at 7th iteration Z axis starts operating at 5th iteration 				

When comparing between ideal cases and disturbance cases, it can be concluded the ILC controllers are able to quickly correct any error. The gripper efficiency was not affected because any affected X, Y and Z axis was able to resolve any error caused by the disturbances in a few iterations as seen from the X axis response shown in Figure 7.4. This is true for ILC controller as well which can be seen in Figure 7.5. For impedance controller, since the PD controller has fast response, it is able to correct itself before grasping the object.

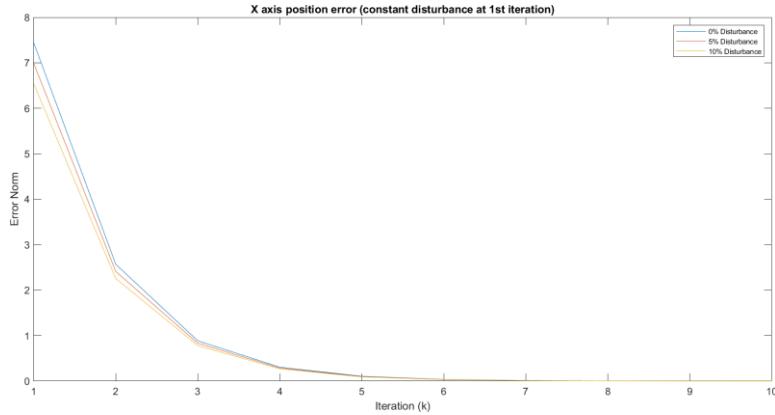


Figure 7.4: Normalised error of X axis for case of ideal, 5% error and 10% constant error occurring at the 1st iteration

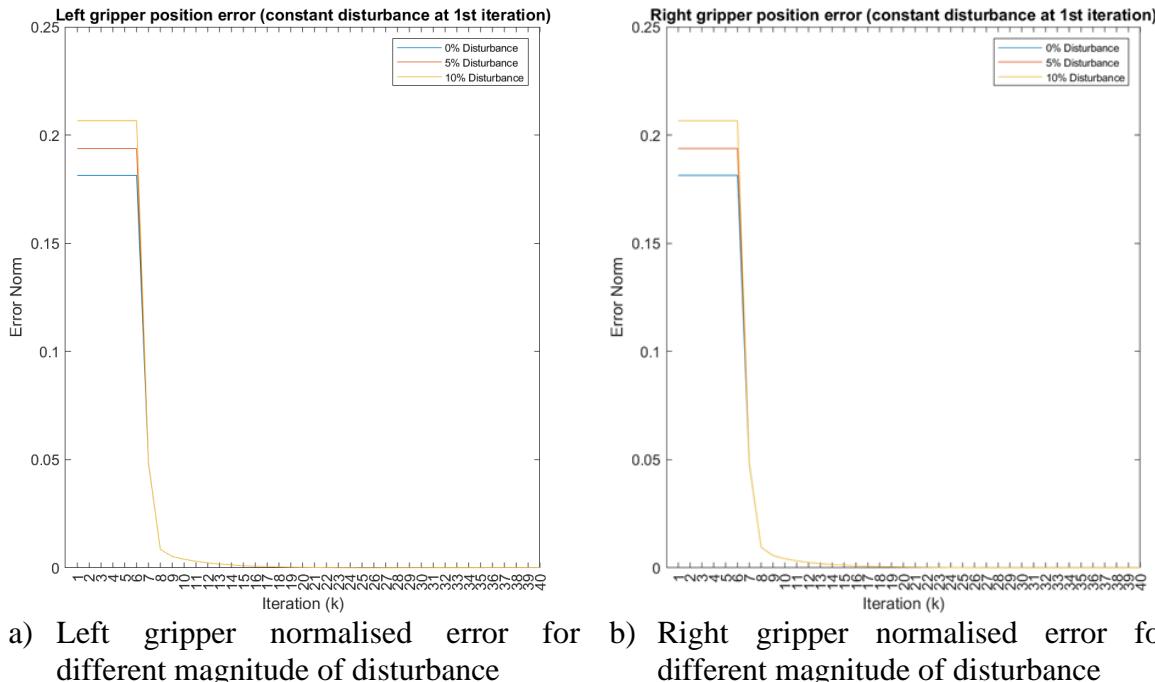


Figure 7.5: Normalised error for gripper when subjected to different magnitude of constant disturbance occurring at the 1st iteration using ILC controller

Any initial condition error present during the start-up of the system does not hinder the performance of both Impedance and ILC based gripper controller.

7.2.2 Error at 30th Iteration

A simple logic system is implemented to only add disturbance at the 30th iteration which is included in the schematics shown in Figure 7.3. 30th iteration is chosen because the force control converges to

minimal error after 25th iteration as shown in Figure 6.19, therefore, this would mimic a system which is in operation for a sufficiently long period of time and now suffering from the effect of wear and tear.

Table 7.3 shows the effect on both Impedance and ILC gripper when part of the gantry system is subjected to a constant disturbance occurring at 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 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%	30 th	<ul style="list-style-type: none"> • Overshoot grasping volume • Attempted but failed grasp 	<ul style="list-style-type: none"> • Overshoot grasping volume • Attempted but failed grasp 	<ul style="list-style-type: none"> • Overshoot grasping volume • Attempted but failed grasp 	<ul style="list-style-type: none"> • Overshoot grasping volume • Attempted but failed grasp 	<ul style="list-style-type: none"> • Same as ideal operation 	<ul style="list-style-type: none"> • Same as ideal operation 	<ul style="list-style-type: none"> • Stable grasp 	<ul style="list-style-type: none"> • Failed grasp 			
	31 st	<ul style="list-style-type: none"> • Overshoot grasping volume • Unstable grasp 	<ul style="list-style-type: none"> • Overshoot slightly • Success grasp (off centre) 	<ul style="list-style-type: none"> • Overshoot slightly • Success grasp, slid slightly 	<ul style="list-style-type: none"> • Overshoot slightly • Object ‘push’ before grasping 				<ul style="list-style-type: none"> • Failed grasp 			
	32 nd	<ul style="list-style-type: none"> • Same as ideal operation 	<ul style="list-style-type: none"> • Overshoot slightly • Success grasp (off centre) 	<ul style="list-style-type: none"> • Same as ideal operation 	<ul style="list-style-type: none"> • Same as ideal operation 				<ul style="list-style-type: none"> • Initial success grasp, drop in transportation 			
	33 rd								<ul style="list-style-type: none"> • Success grasp, slid slightly 			
	34 th								<ul style="list-style-type: none"> • Same as ideal operation 			
10%	30 th	<ul style="list-style-type: none"> • Overshoot grasping volume • Failed grasp 	<ul style="list-style-type: none"> • Overshoot grasping volume • Failed grasp 	<ul style="list-style-type: none"> • Overshoot grasping volume. • Failed grasp 	<ul style="list-style-type: none"> • Overshoot grasping volume. • Failed grasp 	<ul style="list-style-type: none"> • Same as ideal operation 	<ul style="list-style-type: none"> • Same as ideal operation 	<ul style="list-style-type: none"> • Success grasp • Slid slightly. 	<ul style="list-style-type: none"> • Failed grasp 			
	31 st	<ul style="list-style-type: none"> • Overshoot slightly • Attempted but failed grasp 	<ul style="list-style-type: none"> • Overshoot slightly • Attempted but failed grasp 	<ul style="list-style-type: none"> • Overshoot slightly • Failed grasping 	<ul style="list-style-type: none"> • Overshoot slightly • Unstable grasping 				<ul style="list-style-type: none"> • Failed grasp 			
	32 nd	<ul style="list-style-type: none"> • Overshoot slightly • Success grasp (off centre) 	<ul style="list-style-type: none"> • Overshoot slightly • Success grasp (off centre) 	<ul style="list-style-type: none"> • Overshoot slightly • Stable grasp 	<ul style="list-style-type: none"> • Overshoot slightly • More stable grasp 				<ul style="list-style-type: none"> • Failed grasp 			
	33 rd	<ul style="list-style-type: none"> • Overshoot slightly • Success grasp (off centre) 	<ul style="list-style-type: none"> • Overshoot slightly • Success grasp (off centre) 	<ul style="list-style-type: none"> • Same as ideal operation 	<ul style="list-style-type: none"> • Overshoot slightly • More stable grasp 				<ul style="list-style-type: none"> • Failed grasp 			
	34 th								<ul style="list-style-type: none"> • Failed grasp 			
	35 th								<ul style="list-style-type: none"> • Initial success grasp, drop in transportation 			
	36 th								<ul style="list-style-type: none"> • Success grasp, slid slightly 			
	37 th								<ul style="list-style-type: none"> • Same as ideal operation 			

From the observations, it can be concluded that for disturbances occurring at X, Y or Z axis, both impedance and ILC gripper converges at roughly the same iteration. Figure 7.6 shows the normalised position error experienced by the X axis when subjected to the disturbances. It can be seen that the normalised error for 5% and 10% disturbance case converges to the ideal case after 34th iterations. This enable X axis to ‘remove’ the repetitive error in just 4 iterations. As expected, the controller will take longer to correct a disturbance with higher magnitude. Figure 7.7 shows the position feedback received and the corresponding changes to the input signal to counter the disturbance.

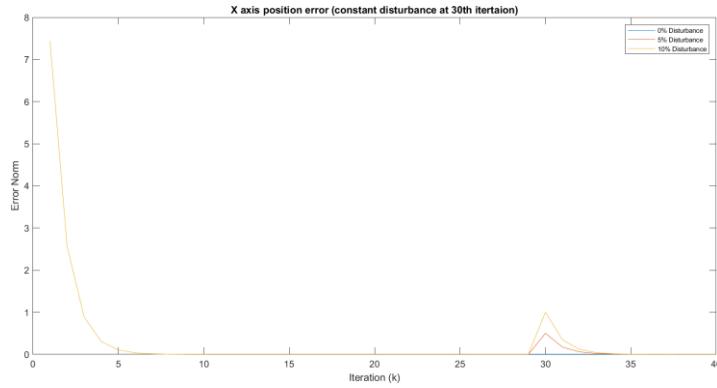


Figure 7.6: X axis normalised position error for case of no disturbance, constant 5% and 10% amplitude disturbance occurring at 30th iteration

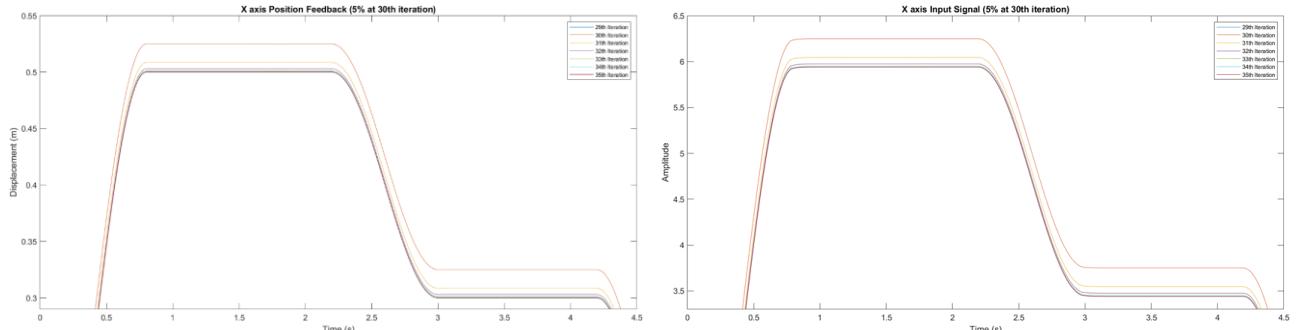
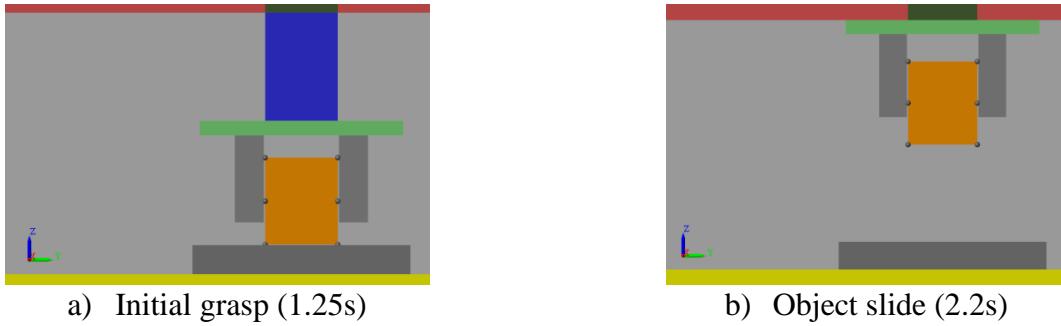


Figure 7.7: X axis position feedback for 5% disturbance occurring at 30th iteration and the corresponding changes to input signal

The most interesting part comes when disturbance occurs at the gripper itself. As seen from Table 7.3, the impedance-controlled gripper always is able to grasp the object, whereas the ILC controlled gripper takes a few iterations before it is able the grasp. However, because impedance control does not learn from previous iterations, for the 10% amplitude disturbance, the object will always slide no matter how many iterations has passed. ILC on the other hand, is able to grasp the object firmly under the same disturbance after just a few iterations. To capture the sliding effect, tactile sensors are normally placed at the fingers, however, since the model does not include tactile sensors, it would have to be identified manually. Figure 7.8a shows the initial grasping position and 7.8b shows the object has slide down slightly.

Figure 7.9 shows the position feedback of the grippers when it is subjected to disturbance. It can be seen that at 30th iteration, the gripper position feedback is lower than the one at 29th iteration. For 20th and 31st iteration, insufficient grasping force was applied as shown in Figure 7.10 causing it fail to grasp the object as highlighted in Table 7.3; This causes significant change to the next iteration input update using equation 6.1 to 6.5 resulting in excessive force used compared to the one at 29th iteration. After the object has been grasped (32nd iteration), it will start correcting itself and finally residing to similar applied force seen at 29th iteration.



a) Initial grasp (1.25s)

b) Object slide (2.2s)

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

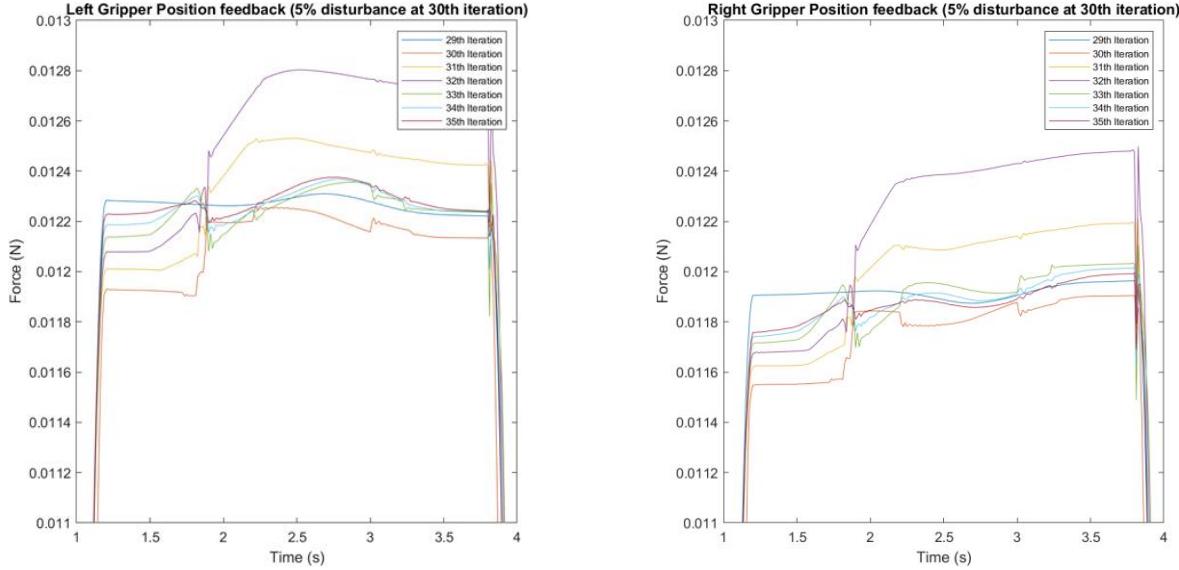


Figure 7.9: Graph showing gripper position feedback when the ILC controlled gripper is subjected to 5% constant disturbance at 30th iteration

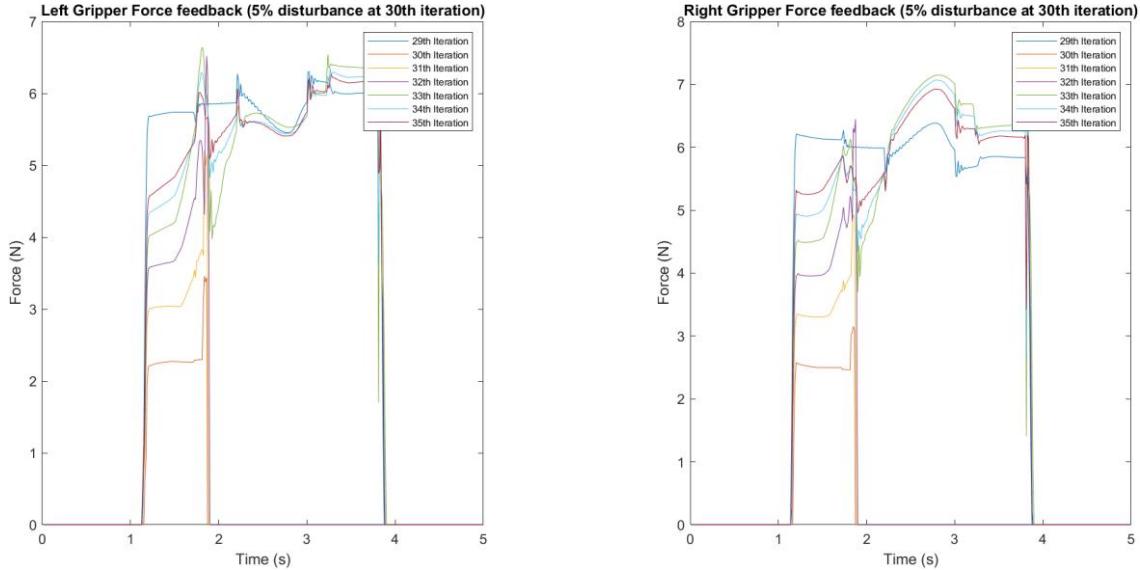


Figure 7.10: Graph showing force applied when the ILC controlled gripper is subjected to 5% constant disturbance at 30th iteration

7.3 Impulsive Error

The result of subjecting the gantry system to different magnitude of impulse disturbance is shown in Table 7.4. Similar to the constant disturbance case, impedance and ILC gripper performance 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 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%	30 th	• Failed grasp	• Object slid	• Failed grasp	• Failed grasp	• Stable grasp	• Stable grasp	• Object slid	• Fail grasp
	31 st	• Stable grasp	• Stable grasp	• Stable grasp	• Stable grasp				• Fail grasp
	32 nd	• Stable grasp							
10%	30 th	• Failed grasp	• Failed grasp	• Failed grasp	• Object slid more			• Failed grasp	• Fail grasp
	31 st	• Object slid	• Object slid	• Object slid and dropped	• Object slid slightly			• Fail grasp	
	32 nd	• Stable grasp	• Stable grasp	• Stable grasp	• Stable grasp			• Fail grasp	
	33 rd			• Stable grasp					

Since X, Y and Z axes uses ILC controller, the effect caused by the disturbance is able to be minimised in a few iterations. Figure 7.11a shows that at 30th iteration, there is a massive error but by 34th iteration, there is little difference compared to the one at 29th iteration. Figure 7.11b on the other hand shows that the input signal for X axis changes to account for the disturbance.

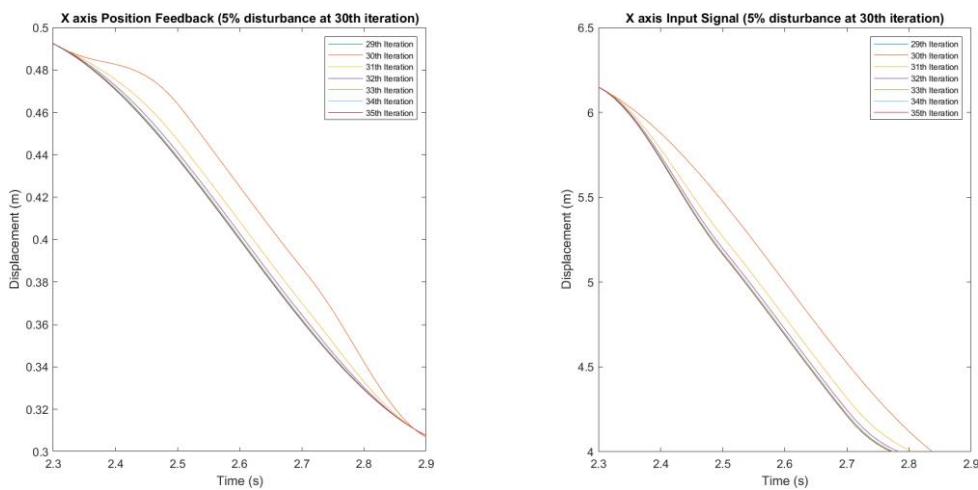


Figure 7.11: Adaptation of ILC algorithm on X axis position feedback and input signal when subjected to 5% Impulse disturbance

The corresponding force and position feedback of the impedance gripper when X axis is subjected to 5% impulse disturbance is shown in Figure 7.12, while ILC gripper is shown in Figure 7.13. For Impedance gripper, it can be seen that at 30th iteration, the force drops to roughly 0 around 2.6s, this

indicates that the object has been dropped, in line with the observation in Table 7.4. During this time, the impedance controller does not have knowledge that the object has been dropped, as a result, it assumed that there is not enough force, hence the position feedback plot shows it rises sharply.

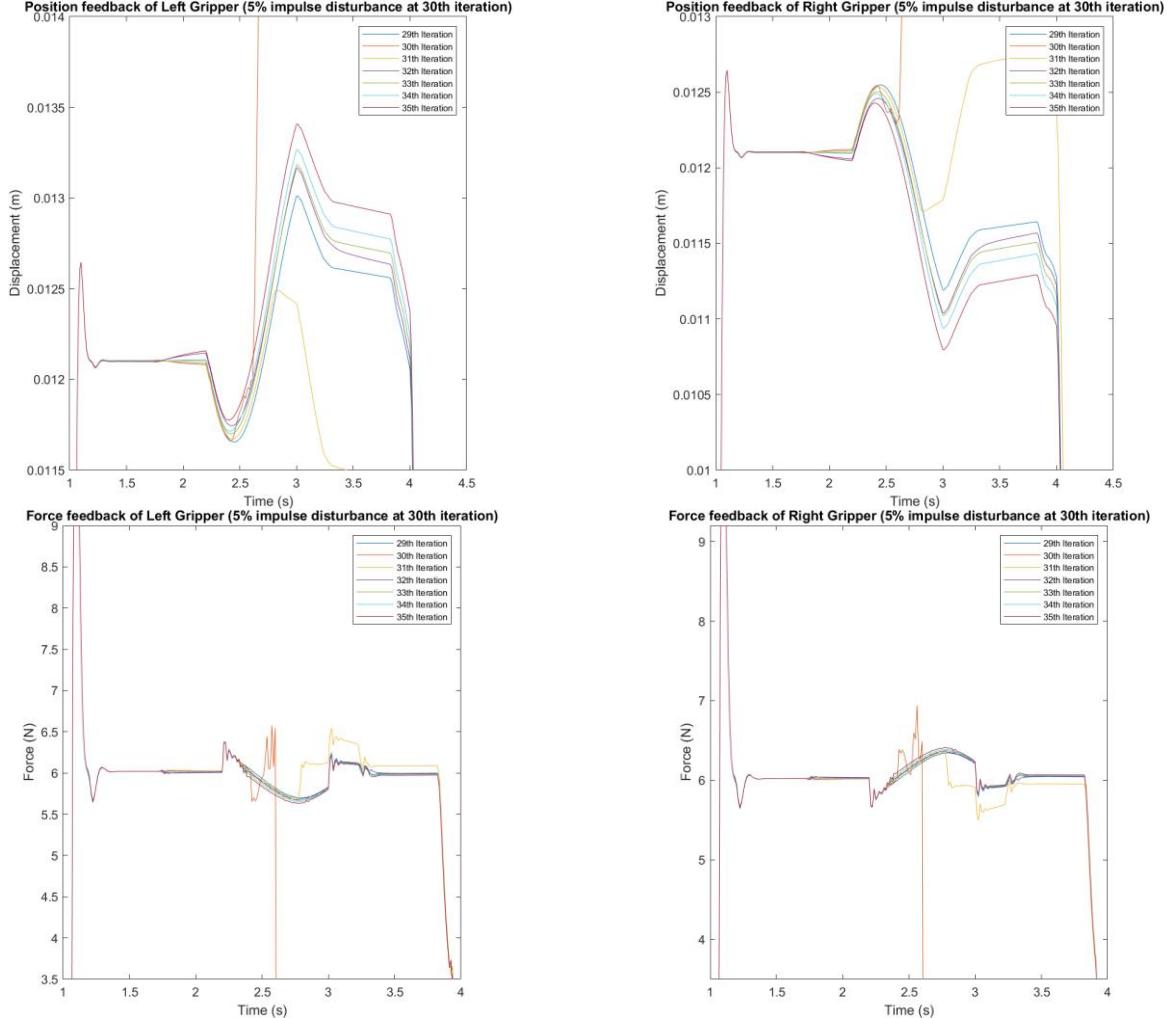


Figure 7.12: Impedance controlled gripper position feedback and force plots when X axis is subjected to 5% impulse disturbance

The ILC gripper does not drop the object when subjected X axis with 5% impulse disturbance, but rather slip. Due to the lack of tactile sensor, it is not possible to identified slippage, but the force plot seen in Figure 7.13 shows that there is a minor spike, indicating something has happened.

Both grippers performance superbly when the Z axis is subjected to impulse disturbance. When Z axis encounters impulse disturbance, it is similar to shaking the object in the direction of up and down once, with the help of excess 20% force applied to both algorithms, the gripper is still able to grasp the object securely.

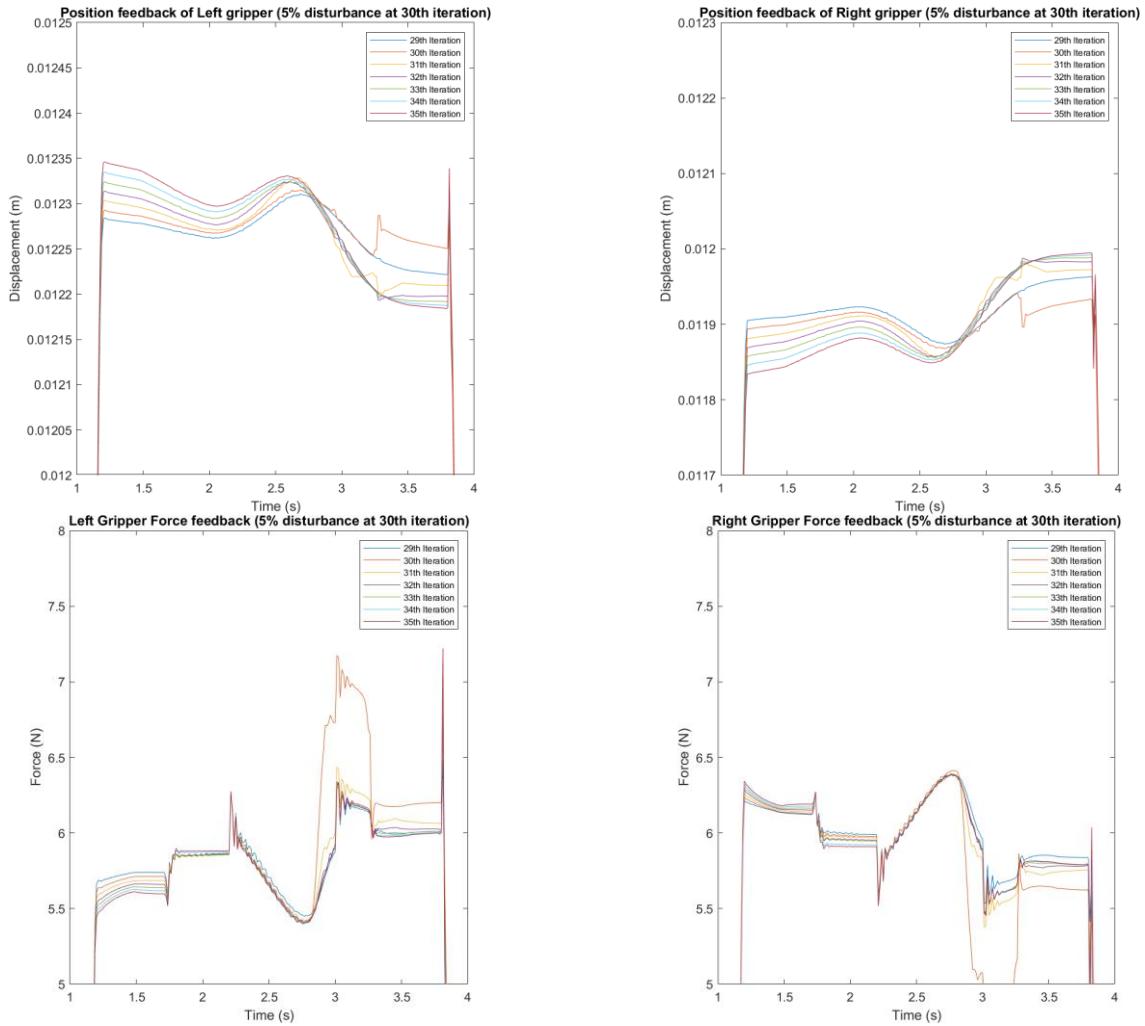


Figure 7.13: ILC controlled gripper position feedback and force plots when X axis is subjected to 5% impulse disturbance

In previous tests, impedance-controlled gripper does not encounter much issue, just minor slippage when the disturbance amplitude is 10%. However, since impedance does not learn from previous iteration mistakes, as seen in Table 7.4, once the gripper becomes unstable and drop the object, it will always do the same thing. Figure 7.14 shows the sequence where the gripper drops the object under 10% impulse disturbance. ILC controller on the other hand initially performs worse by dropping the object, but after 2-3 iterations, it converges and is able to grasp the object.

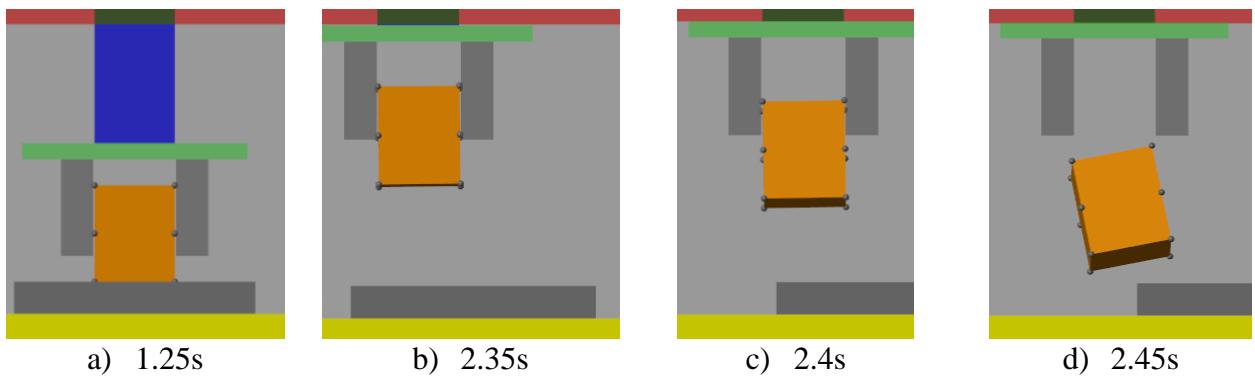


Figure 7.14: Sequence of gripper using impedance controller subjected to 10% impulse disturbance

The effect of impulse disturbance can be seen at the 30th iteration in Figure 7.15. The position feedback of 29th and 35th iteration is almost identical, showcasing the ability of ILC to correct repetitive error

over impedance controller. The force experienced by the object is also shown in Figure 7.16. It can clearly be seen that part of 30th, 31st and 32nd plot goes down to 0 indicating the object has been dropped, this is also highlighted in Table 7.4.

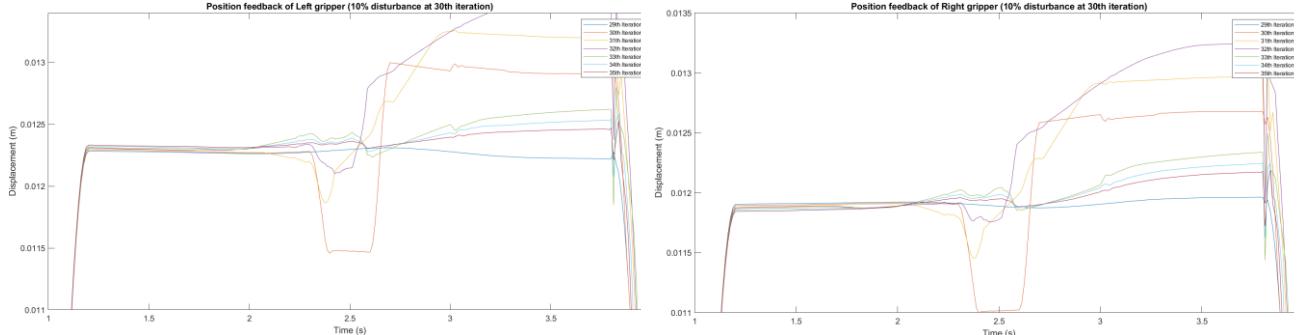


Figure 7.15: Gripper position feedback when subjected to 10% impulse disturbance

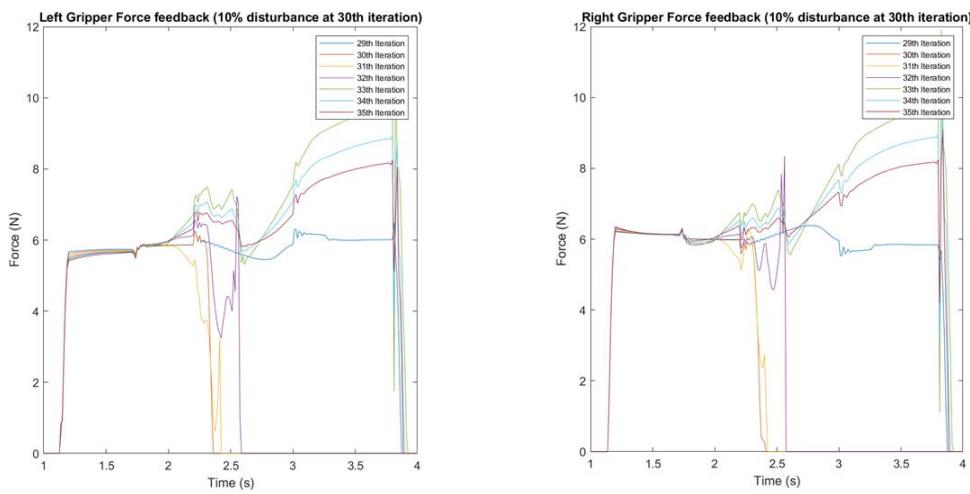


Figure 7.16: Force experienced when gripper is subjected to 10% impulse disturbance

7.4 Summary

Both impedance and ILC controlled gripper performance relatively similar when the gantry system is subjected to constant and impulse disturbance. Particularly, due to the additional 20% force margin specified in the control algorithm, Z axis disturbance does not posses any trouble to the gripper controllers.

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 not able to grasp the object securely. Although ILC gripper initially dropped the object, after a few iterations, it is able to 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

The model of a typical industrial gantry robot has been successfully replicated in MATLAB Simulink simulation environment with the help of Simscape Multibody toolbox. Any interaction forces between environment and object to be grasped is also modelled.

The ILC controlled gripper is able to reduce the spike in force issue encountered by impedance control after a few iterations. However, the force profile is not as smooth as impedance control owing to the fact that ILC updates its input after every iteration. As the ILC controller in this project uses a model-based algorithm, it is crucial to update the plant model to include the added mass caused by grasping the object to reduce the chance of the system becoming unstable.

The ILC controller has proven itself to be great at eliminating repetitive disturbances that would otherwise cause significant damage (drop grasped object) if impedance controller is used.

To fully evaluate the success of the project, the specification layout in Chapter 2 is compared against the result and analysis conducted in Chapters 4-7.

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 related 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.	However, a more flexible robotic arm robot with multiple direction of freedom (DOF) is gaining popularity in industry.
2	The simulation model is capable of modelling the dynamics of the system and any interaction force it may encounter.	Unfortunately, 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 axis of the gantry robot and the gripper can be visually seen, and it is possible to see what is happening when it is simulating.	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.	-
5	The whole reach and grasp profile is 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 for 1 iteration might increase.
6	The gripper uses around 20% more force when grasping an object after it converges.	Initial few iterations use quite excessive force depending on the initial profile generated for the gripper
7	The gripper is able to safely transport object ranging from 0.2kg to 2kg	Further research should aim at grasping object of various shapes and size
8	The gripper is able to correct itself and grasp the object when there is 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 superb at removing 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 focus on evaluating the performance of the new ILC controller in a simulation environment. To fully test the controller, it is necessary to use the actual gantry robot and compare the results obtained with the simulation data to fully justify the improvements gain for switching to ILC controller from Impedance controller.

8.2.3 Increase Complexity of Gripper Hardware

The gripper in this project uses a basic two parallel plate fingers gripper with force sensors. To avoid object 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 at the fingers.

As identified by Chapter 1, there are multiple more advance gripper structure. To fully exploit ILC capability and improve grasping, these more advance 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 comes not only in different mass, but also different shapes and sizes. To be able to grasp different types of objects, some sort of visual camera should be used. With the help of machine learning, different tracking profiles should be used to grasp different objects. ILC can be used to improve the tracking profiles used 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 function. This could potentially cause the update to be conflicting each other. Research should be conducted to find the relationship between force applied and grasping distance and minimise these two components to produce only just one update equation instead of the current two.

References

- [1] Dima, "Short history of manufacturing: from Industry 1.0 to Industry 4.0", KFactory, 2021. [Online]. Available: [https://kfactory.eu/short-history-of-manufacturing-from-industry-1-0-to-industry-4-0/#:~:text=The%20Third%20Industrial%20Revolution%20\(Industry,memory%2Dprogrammable%20controls%20and%20computers](https://kfactory.eu/short-history-of-manufacturing-from-industry-1-0-to-industry-4-0/#:~:text=The%20Third%20Industrial%20Revolution%20(Industry,memory%2Dprogrammable%20controls%20and%20computers).
- [2] "Robotics Market Size, Share (2022 - 27)", Mordorintelligence.com. [Online]. Available: <https://www.mordorintelligence.com/industry-reports/robotics-market>.
- [3] R. Hodson, "A gripping problem", Nature, vol. 557, 2018.
- [4] T. Scott, "How many robots does it take to run a grocery store? ", [Online]. Available: https://www.youtube.com/watch?v=ssZ_8cqfBIE.
- [5] J. Mahler et al., "Learning ambidextrous robot grasping policies," Science Robotics, vol. 4, no. 26, p. eaau4984, 2019, doi: doi:10.1126/scirobotics.aau4984.
- [6] O. Ryuta and T. Kenji, "Grasp and dexterous manipulation of multi-fingered robotic hands: a review from a control view point," Advanced Robotics, vol. 31, no. 19-20, pp. 1030-1050 , year = 2017, doi: 10.1080/01691864.2017.1365011.
- [7] NAPIER JR. The prehensile movements of the human hand. J Bone Joint Surg Br. 1956 Nov;38-B(4):902-13. doi: 10.1302/0301-620X.38B4.902. PMID: 13376678.50
- [8] Kamakura N, Matsuo M, Ishii H, Mitsuboshi F, Miura Y. Patterns of static prehension in normal hands. Am J Occup Ther. 1980 Jul;34(7):437-45. doi: 10.5014/ajot.34.7.437. PMID: 6446851.
- [9] M. R. Cutkosky, "On grasp choice, grasp models, and the design of hands for manufacturing tasks," in IEEE Transactions on Robotics and Automation, vol. 5, no. 3, pp. 269-279, June 1989, doi: 10.1109/70.34763.
- [10] B. Abbasi, E. Noohi, S. Parastegari and M. Žefran, "Grasp taxonomy based on force distribution," 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), 2016, pp. 1098-1103, doi: 10.1109/ROMAN.2016.7745245.
- [11] Y. Li, Q. Lei, C. Cheng, G. Zhang, W. Wang, and Z. Xu, "A review: machine learning on robotic grasping," 03, 2019.
- [12] T. Yamashita, 'Engineering approaches to function of fingers', 1963.
- [13] Mouri, T., Kawasaki, H., and Ito, S., "Unknown Object Grasping Strategy Imitating Human Grasping Reflex for Anthropomorphic Robot Hand", Journal of Advanced Mechanical Design, Systems, and Manufacturing, vol. 1, no. 1, pp. 1–11, 2007. doi:10.1299/jamds.1.1
- [14] M. Teichmann and B. Mishra, "Reactive Robotics I: Reactive Grasping with a Modified Gripper and Multifingered Hands," The International Journal of Robotics Research, vol. 19, no. 7, pp. 697-708, 2000, doi: 10.1177/027836490001900706
- [15] K. G. Shin and C. -p. Lee, "Compliant control of robotic manipulators with resolved acceleration," 1985 24th IEEE Conference on Decision and Control, 1985, pp. 350-357, doi: 10.1109/CDC.1985.268883.
- [16] P. R. Pagilla and M. Tomizuka, "Adaptive control of two robot arms carrying an unknown object," Proceedings of 1995 IEEE International Conference on Robotics and Automation, 1995, pp. 597-602 vol.1, doi: 10.1109/ROBOT.1995.525349.
- [17] J. -. E. Slotine and Li Weiping, "Adaptive manipulator control: A case study," in IEEE Transactions on Automatic Control, vol. 33, no. 11, pp. 995-1003, Nov. 1988, doi: 10.1109/9.14411.
- [18] S. Pertuz, C. Llanos, and D. Muñoz, "Simulation and Implementation of Impedance Control in Robotic Hand," 04, 2018

- [19] S. Pertuz, C. Llanos, and D. Muñoz, "Simulation and Implementation of Impedance Control in Robotic Hand," 04, 2018
- [20] D. A. Bristow, M. Tharayil and A. G. Alleyne, "A survey of iterative learning control," in IEEE Control Systems Magazine, vol. 26, no. 3, pp. 96-114, June 2006, doi: 10.1109/MCS.2006.1636313.
- [21] T. Nakayama, S. Arimoto and T. Naniwa, "Coordinated learning control for multiple manipulators holding an object rigidly," Proceedings of 1995 IEEE International Conference on Robotics and Automation, 1995, pp. 1529-1534 vol.2, doi: 10.1109/ROBOT.1995.525492.
- [22] S. SHAIKH and M. SHAIKH, "IMPORTANCE OF MODELING AND SIMULATION: A VITAL SYSTEMS ENGINEERING TECHNIQUE," International Journal of Advances in Science Engineering and Technology, vol. 5, no. 1, pp. 121–124, Feb. 2017.
- [23] H. Elci, R. W. Longman, Minh Phan, Jer-Nan Juang and R. Ugoletti, "Discrete frequency based learning control for precision motion control," Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 1994, pp. 2767-2773 vol. 3, doi: 10.1109/ICSMC.1994.400292.
- [24] W. Messner, R. Horowitz, W. -. Kao and M. Boals, "A new adaptive learning rule," in IEEE Transactions on Automatic Control, vol. 36, no. 2, pp. 188-197, Feb. 1991, doi: 10.1109/9.67294.
- [25] Dong-Il Kim and Sungkwan Kim, "An iterative learning control method with application for CNC machine tools," in IEEE Transactions on Industry Applications, vol. 32, no. 1, pp. 66-72, Jan.-Feb. 1996, doi: 10.1109/28.485814.
- [26] S. Arimoto, S. Kawamura and F. Miyazaki, "Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronics systems," The 23rd IEEE Conference on Decision and Control, 1984, pp. 1064-1069, doi: 10.1109/CDC.1984.272176.
- [27] S. Arimoto, S. Kawamura, F. Miyazaki and S. Tamaki, "Learning control theory for dynamical systems," 1985 24th IEEE Conference on Decision and Control, 1985, pp. 1375-1380, doi: 10.1109/CDC.1985.268737.
- [28] M. Q. Phan and J. A. Frueh, "Learning control for trajectory tracking using basis functions," Proceedings of 35th IEEE Conference on Decision and Control, 1996, pp. 2490-2492 vol.3, doi: 10.1109/CDC.1996.573465.
- [29] O. Markusson, H. Hjalmarsson and M. Norrlöf, "Iterative learning control of nonlinear non-minimum phase systems and its application to system and model inversion," Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228), 2001, pp. 4481-4482 vol.5, doi: 10.1109/CDC.2001.980908.
- [30] M. Togai and O. Yamano, "Analysis and design of an optimal learning control scheme for industrial robots: A discrete system approach," 1985 24th IEEE Conference on Decision and Control, 1985, pp. 1399-1404, doi: 10.1109/CDC.1985.268741.
- [31] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control for discrete-time systems with exponential rate of convergence," IEE Proceedings - Control Theory and Applications, vol. 143, no. 2, pp. 217-224. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/ip-cta_19960244
- [32] N. Amann, D. H. Owens and E. Rogers, "Iterative learning control for discrete time systems using optimal feedback and feedforward actions," Proceedings of 1995 34th IEEE Conference on Decision and Control, 1995, pp. 1696-1701 vol.2, doi: 10.1109/CDC.1995.480384.
- [33] G. Lahr, J. V. Rocha Soares, H. Garcia, Adriano, A. A. G. Siqueira, and G. Caurin, "Understanding the Implementation of Impedance Control in Industrial Robots," 10, 2016
- [34] S. Pertuz, C. Llanos, and D. Muñoz, "Simulation and Implementation of Impedance Control in Robotic Hand," 04, 2018.

- [35] José and K. Frank, "Modelling and Simulation of Robot Arm Interaction Forces Using Impedance Control," IFAC Proceedings Volumes, vol. 41, no. 2, pp. 15589-15594, 2008, doi: <https://doi.org/10.3182/20080706-5-KR-1001.02636>
- [36] F. Caccavale, C. Natale, B. Siciliano and L. Villani, "Integration for the next generation: embedding force control into industrial robots," in IEEE Robotics & Automation Magazine, vol. 12, no. 3, pp. 53-64, Sept. 2005, doi: 10.1109/MRA.2005.1511869.
- [37] J. Ratcliffe, "Iterative Learning Control Implemented on a Multi-Axis System", June 2005.
- [38] J. Collins, S. Chand, A. Vanderkam and D. Howard, "A Review of Physics Simulators for Robotic Applications," in IEEE Access, vol. 9, pp. 51416-51431, 2021, doi: 10.1109/ACCESS.2021.3068769.
- [39] Aerotech sales website, “AGS1000 Direct-Drive Gantry”. Available at:
<https://uk.aerotech.com/product/gantries-en-uk/ags1000-direct-drive-gantry/>

Appendix A. Project management

Project management including time and progress management is an important part of ensuring the project is completed in an organised and timely manner. The following sections identified several risks and methods to minimise them and plans to realise a smooth project progress.

A1 - Contingency Planning, Minimising risk, reduce impact

Setbacks and issues due to unexpected events, failures and changes are unavoidable in projects. An effective solution involves the following steps:

1. Identify potential setbacks/issues.
2. Plan out methods of minimising the chances or damage of these setbacks/issues.
3. Create contingency plans to overcome them.

Setbacks/issues can be classified as risks encountered during a project lifetime. Some risks can be foreseen while others can't. Similarly, some risks are controllable and effective risk management can reduce the impact while others are uncontrollable and can only be managed. Appendix A1 highlights any identified risks. As not all risks are able to be identified at first glance, the list is modified when any unidentified risks become apparent.

Appendix A1 classified each risk by how likely it is to occur, the severity of the risk, methods that could reduce the risk and any contingency plans should the risk becomes apparent.

A2 - Time and Progress Management

Effective progress and time planning can be done with the help of Gantt charts. Gantt charts in this project are split into 2 timelines Semester 1 (Appendix A2) and semester 2 (Appendix A3). Semester 1 plans were mostly completed during the first half of the project timeline.

Semester 1 Gantt chart provides a buffer region which allows any uncompleted works to be carried out without causing domino effect to the rest of the project progress. Part of building the gripper model and contact modelling was not carried out during this buffer region and the rest of the progress was carried out during Semester 2.

Since it takes longer than expected to complete the models, part of the Semester 1 was carried forward to Semester 2. Therefore, there is not sufficient time to test the controller design onto the physical system and compare the performance with the simulation results and evaluate the controller. It has been discussed in Chapter 8 as a potential future work area.

A3 - Risk Management Strategy

The risks are rank out of 5, where 5 is very likely/severe and 1 is unlikely/low severity. A small explanation of each risk is provided below the table

Risk	Risk likelihood	Risk severity	Risk reduction methods	Contingency plans
Loss of files	1	5	N/A	Daily backup of files to both pen drive and private GitHub repo
Simulation model does not work	3	5	N/A	Use MATLAB built in Simulink model
Simulation takes too many resources/too long	2	3	Reduce the step size of the simulation or run section by section	Request to use university computational resources
Simulation files incompatible with other versions	3	2	Ensure MATLAB does not get upgraded/downgraded during the project duration	Locate computer from university/friend which uses that version

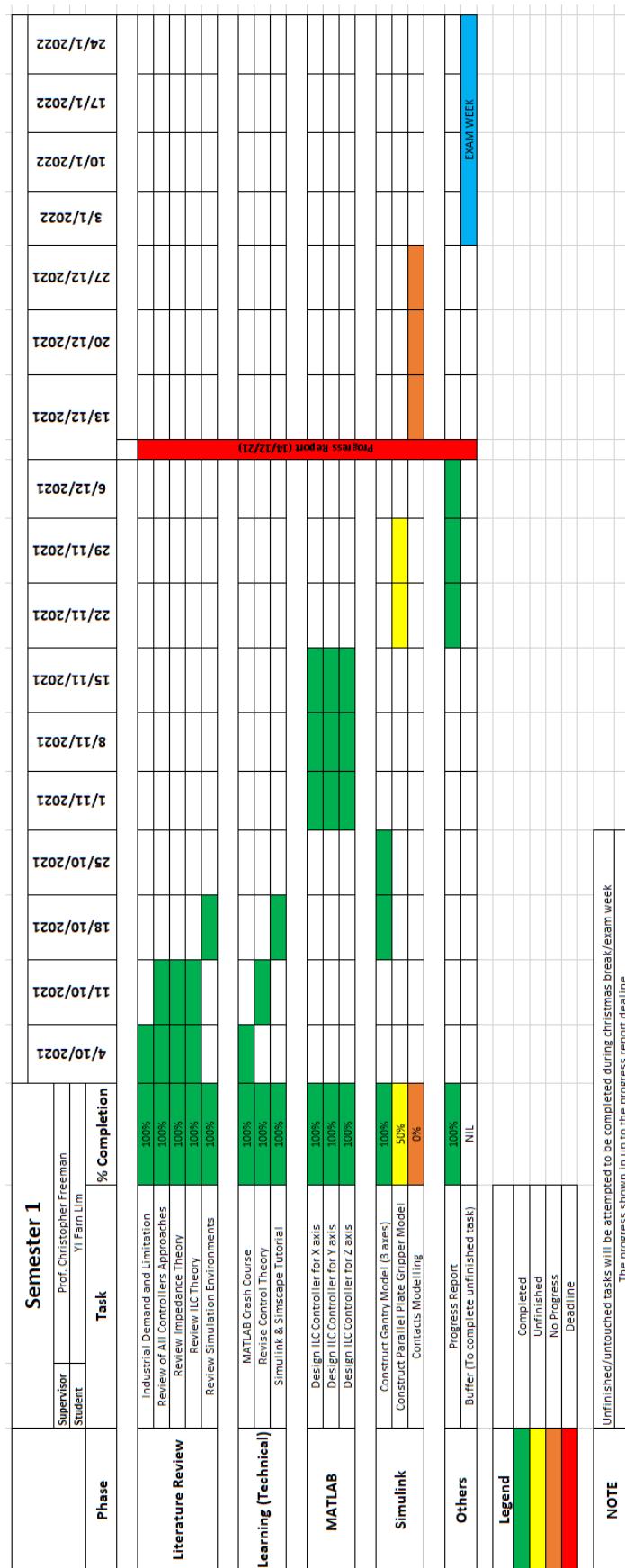
Loss of files – Files might be corrupted or lost due to issues arising from computer/malware/theft.

Simulation model does not work – The attempts on building a working model which mimics the interested robot system fails or it takes much longer than expected to complete.

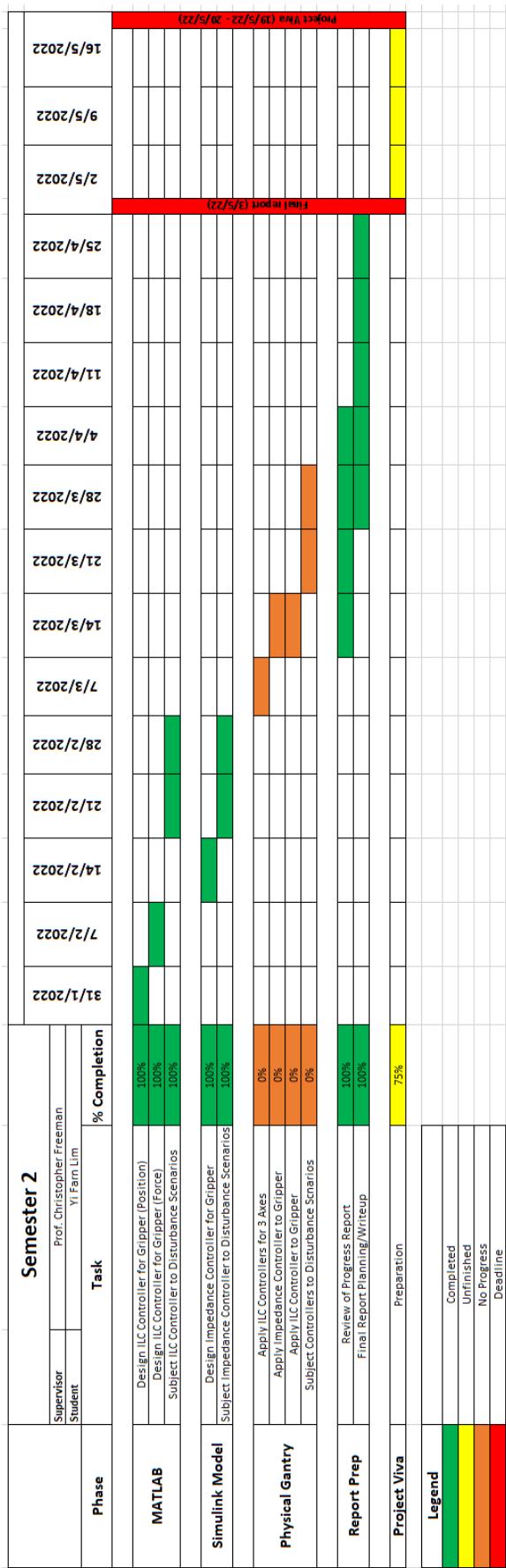
Simulation takes too many resources/too long – MATLAB has been known to consume a significant portion of resources to simulate, this can cause the simulation to be slow and takes a long time to complete.

Simulation files incompatible with other versions – MATLAB files are version dependent; it is not possible to run files created in newer versions on older ones. This could be a factor when simulating using university machines while using personal to write codes.

A4 - Semester One Gantt Chart



A5 - Semester Two Gantt Chart



Appendix B. Project Assumptions

B1 - Position and Force Feedback

It is assumed that the position and force feedback provided to the controller is accurate. Although wear and tear are expected to build up and cause inaccurate data being fed into the controller. A frequent inspection of force sensors or encoder should ensure the error caused by damaged equipment is kept to a minimal. Hence, this is a valid assumption for simulation.

B2 - Force Applied by Gantry Robot

Unfortunately, the specification of the linear motor used by Ratcliff when constructing the Gantry robot is no longer available on the manufacturer website. The closest system that the specification is currently available is AGS1000 series by Aerotech. The maximum load the gantry robot can carry is 15kg.

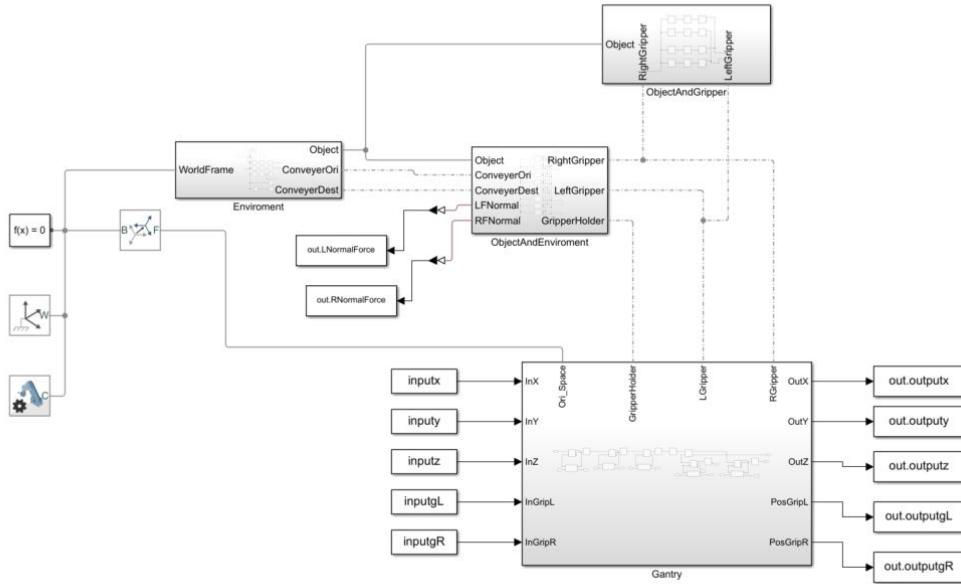
The gripper of choice is a 0.2kg universal 2-finger parallel gripper (JGP-P 50-2-As) from Schunk which recommends grasping workpiece of 2kg. This gives a total load to be transported by the gantry robot to be 2.2kg which is within the 15kg recommendation.

According to the gantry manufacturer, the peak force to move in X direction is 916N while to move in Y direction is 458N. With a gentle rise and fall time of 0.8s for X and Y axis and 0.3s for Z axis, it is safe to assume that the force required would not exceed the recommendation provided by the manufacturer.

Appendix C. Simulink Model Schematic

This section will be dedicated to showing the full schematic for the model built in Simulink as well as detailed explanation of each part. The aim is to provide full information regarding how the simulation model is built.

C1 - Overview

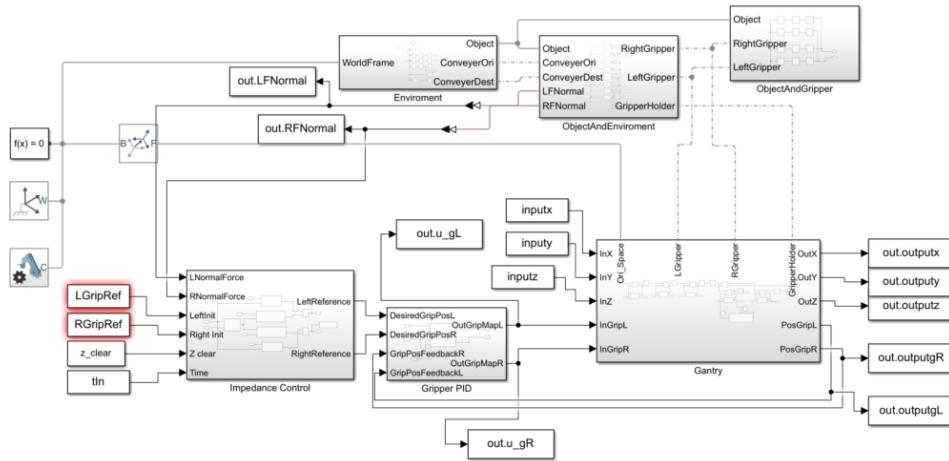


This shows the highest level of the schematic which is used for both ILC and Impedance controller. There are 4 major blocks that made up the whole simulation model: ‘Environment’, ‘Object and Environment’, ‘Object and Gripper’, and ‘Gantry’. Each will be broken down further in the following subsections.

The definition for each ToWorkspace and FromWorkspace block is found at the table below.

Name	Type	Description
inputx	FromWorkspace	Pass X axis input vector from workspace (ILC)
inputy	FromWorkspace	Pass Y axis input vector from workspace (ILC)
inputz	FromWorkspace	Pass Z axis input vector from workspace (ILC)
inputgL	FromWorkspace	Pass Left Gripper input vector from workspace (ILC)
inputgR	FromWorkspace	Pass Right Gripper input vector from workspace (ILC)
out.outputx	ToWorkspace	Pass X axis position feedback to workspace
out.outputy	ToWorkspace	Pass Y axis position feedback to workspace
out.outputz	ToWorkspace	Pass Z axis position feedback to workspace
out.outputgL	ToWorkspace	Pass Left Gripper position feedback to workspace
out.outputgR	ToWorkspace	Pass Right Gripper position feedback to workspace
out.LNormalForce	ToWorkspace	Pass Left Gripper force feedback to workspace
out.RNormalForce	ToWorkspace	Pass Right Gripper force feedback to workspace

The schematic for Impedance controlled gripper is slightly different. This is due to implementing full impedance control in Simulink instead of passing values from workspace as done with ILC. The schematic can be found below. The major difference is the addition of ‘Impedance control’ and ‘PD control’, these will be further decomposed in the following subchapters.

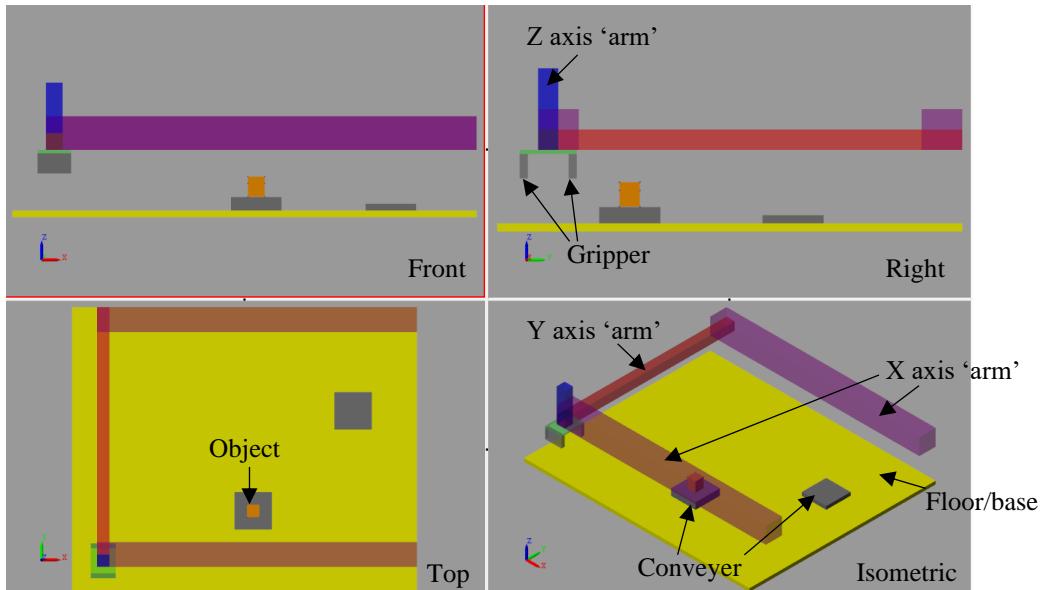


The definition for new ToWorkspace and FromWorkspace blocks can be found below

Name	Type	Description
LGripRef	FromWorkspace	Initial trajectory profile for Left Gripper
RGripRef	FromWorkspace	Initial trajectory profile for Right Gripper
z_clear	FromWorkspace	Logic signal to signal X, Y and Z axis has reach threshold
tIn	FromWorkspace	Discrete time samples
out.u_gL	ToWorkspace	Pass Left Gripper input signal to Workspace
out.u_gR	ToWorkspace	Pass Right Gripper input signal to Workspace

Data passed to workspace are either use as feedback for controllers or to plot, analyse and evaluate the controller performance.

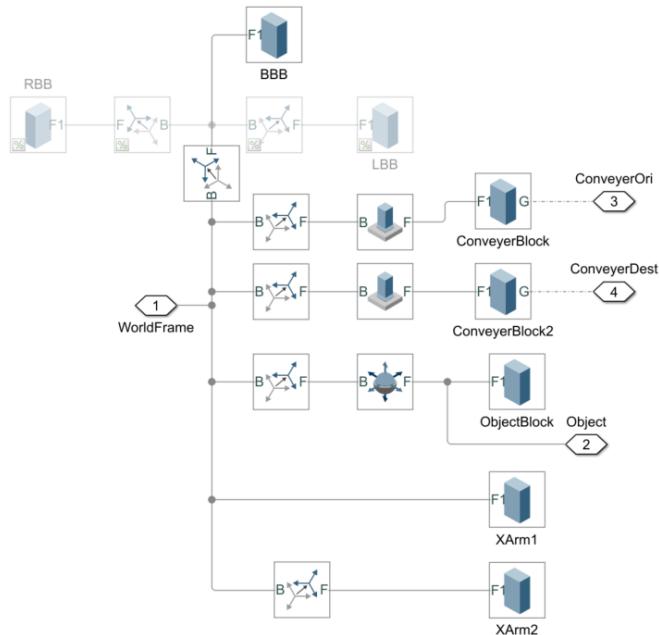
A visual representation of the model is shown below along with labels.



The following subsections will breakdown each individual subsystem of the model.

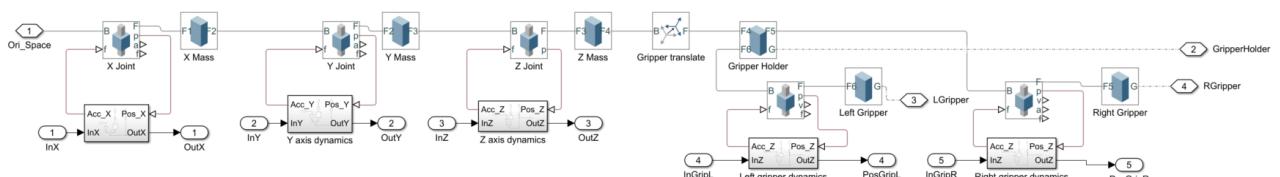
C2 - Environment

This subsystem contains all visual elements that do not affect the operation of the gantry robot which includes the object to be grasped (ObjectBlock), conveyer ‘blocks’(ConveyerBlock and ConveyerBlock2), floor (BBB) and X axis arm to match the actual gantry (XArm1 and XArm2). The schematic is shown below

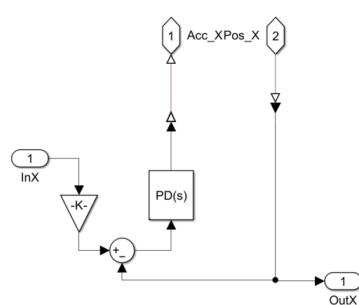


C3 - Gantry

This subsystem houses the blocks required to model the dynamics of the gantry robot and grippers.

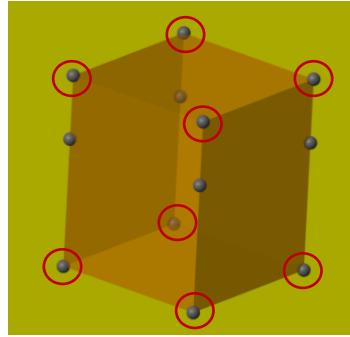


The dynamics of each part of the robot is modelled using a PD controller each with a similar schematic as follows.

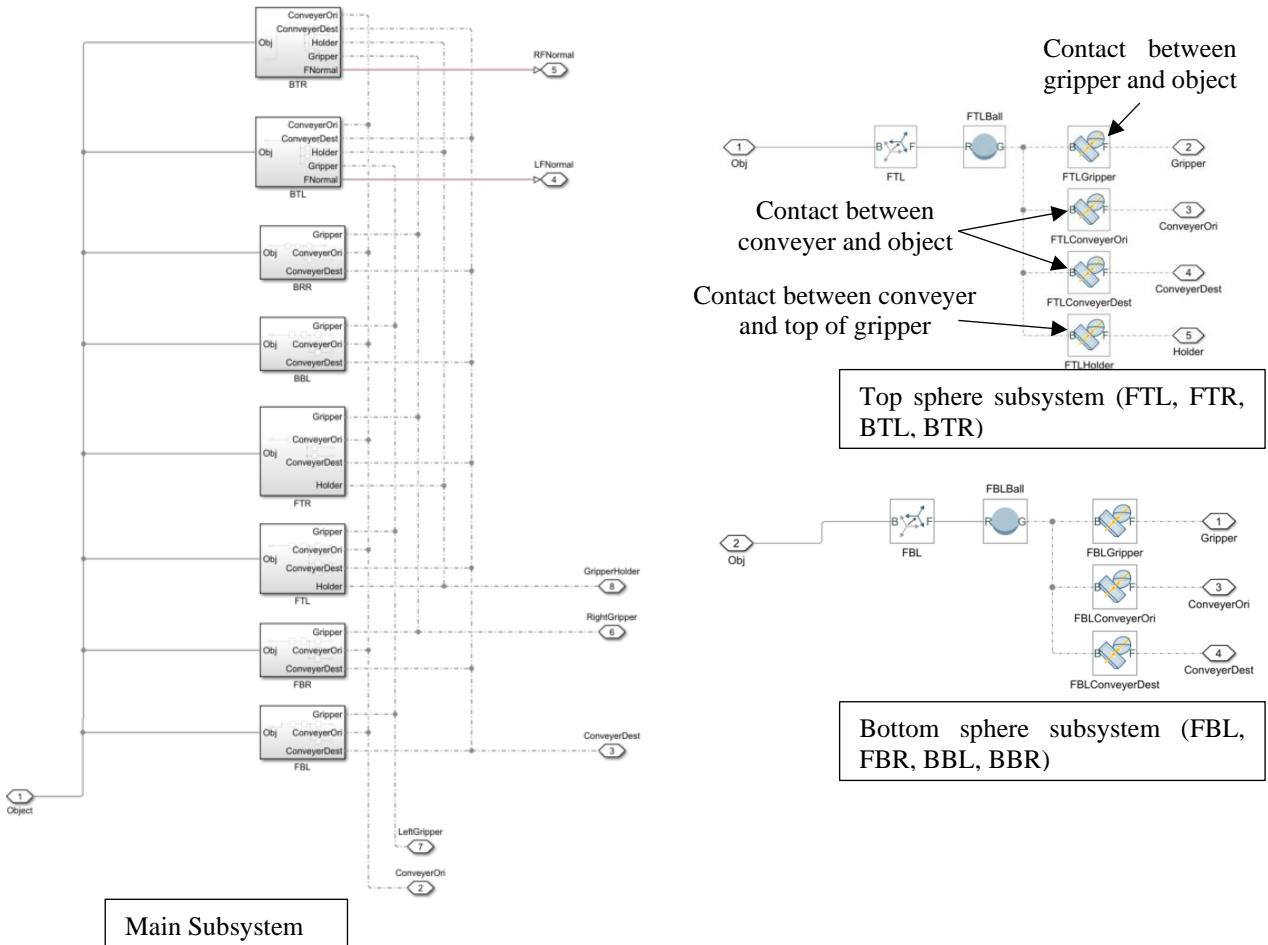


C4 - ObjectAndEnvironment

This subsystem focuses on the interaction forces and condition between the object and the environment which includes the conveyer where the object is located at and the top of the gripper (not finger). This makes up each top and bottom 4 spheres as circled below. As the object is a cube, A contact sphere located at each edge of the object ensures the object interact with the environment adequately and would not topple/rotate without external intervention.

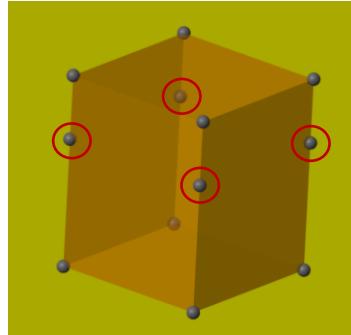


The schematic for such system is shown below. The contact schematic is slightly different for the top 4 spheres compare to

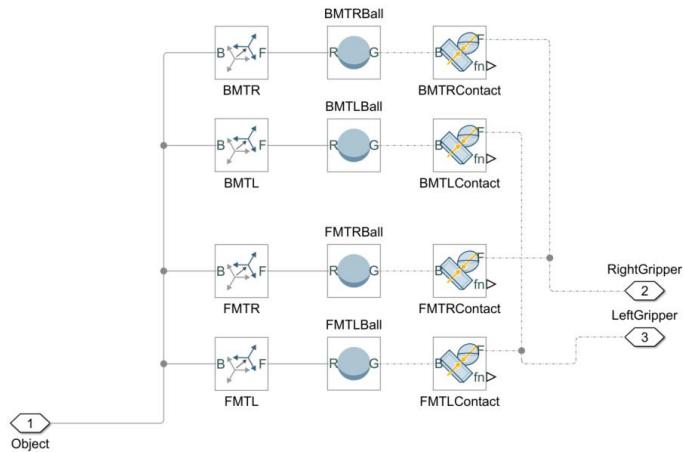


C5 - ObjectAndGripper

This subsystem is similar to the previous one, where this focuses on the contact sphere located at roughly the centre of the object.



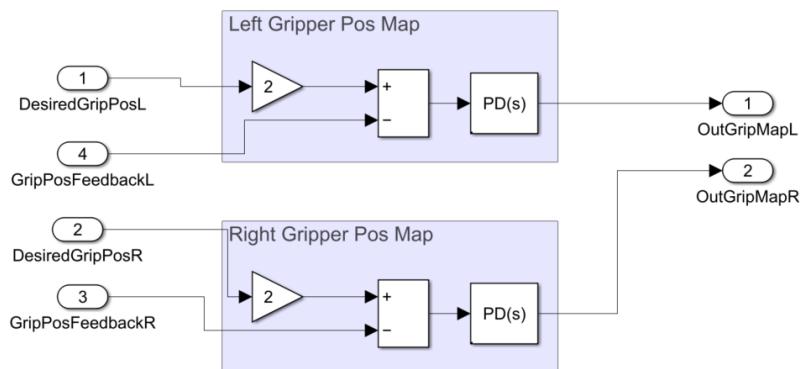
As this contact sphere is located roughly at the centre, it only has to interact with the gripper. Therefore, the schematic is relatively simple and is shown below.



C6 - Impedance Controller

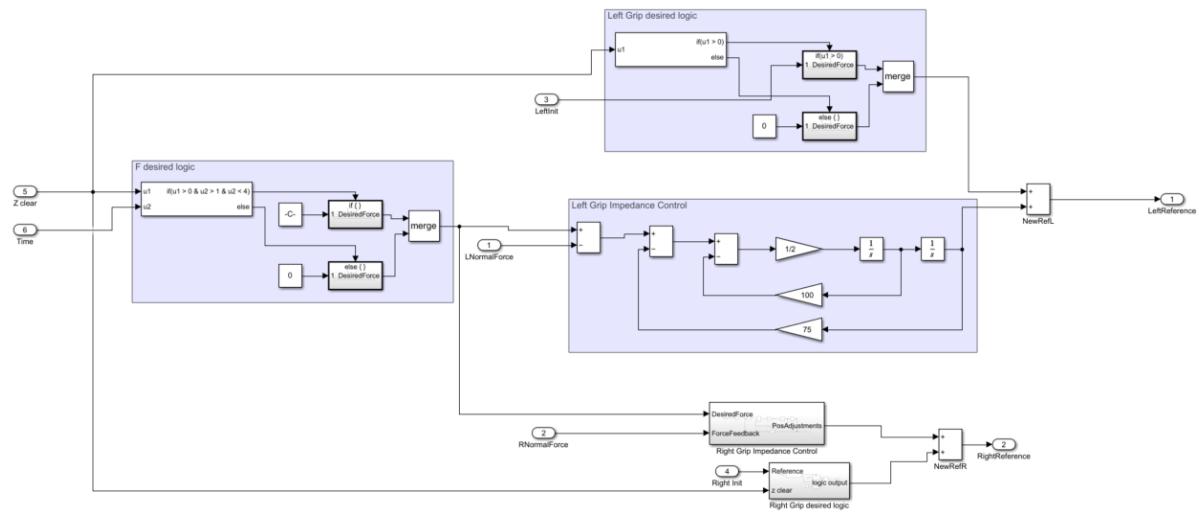
The impedance controller can be split into 2 sections. One is the force correction impedance control schematic and the other is the position correction PID schematic.

The position correction PID schematic is a simple feedback PID controller.



The force correction impedance control schematic is more complicated. Each finger controller is made up of 3 additional subsystems namely ‘F desired logic’, ‘Grip desired logic’, and ‘Grip Impedance Control’

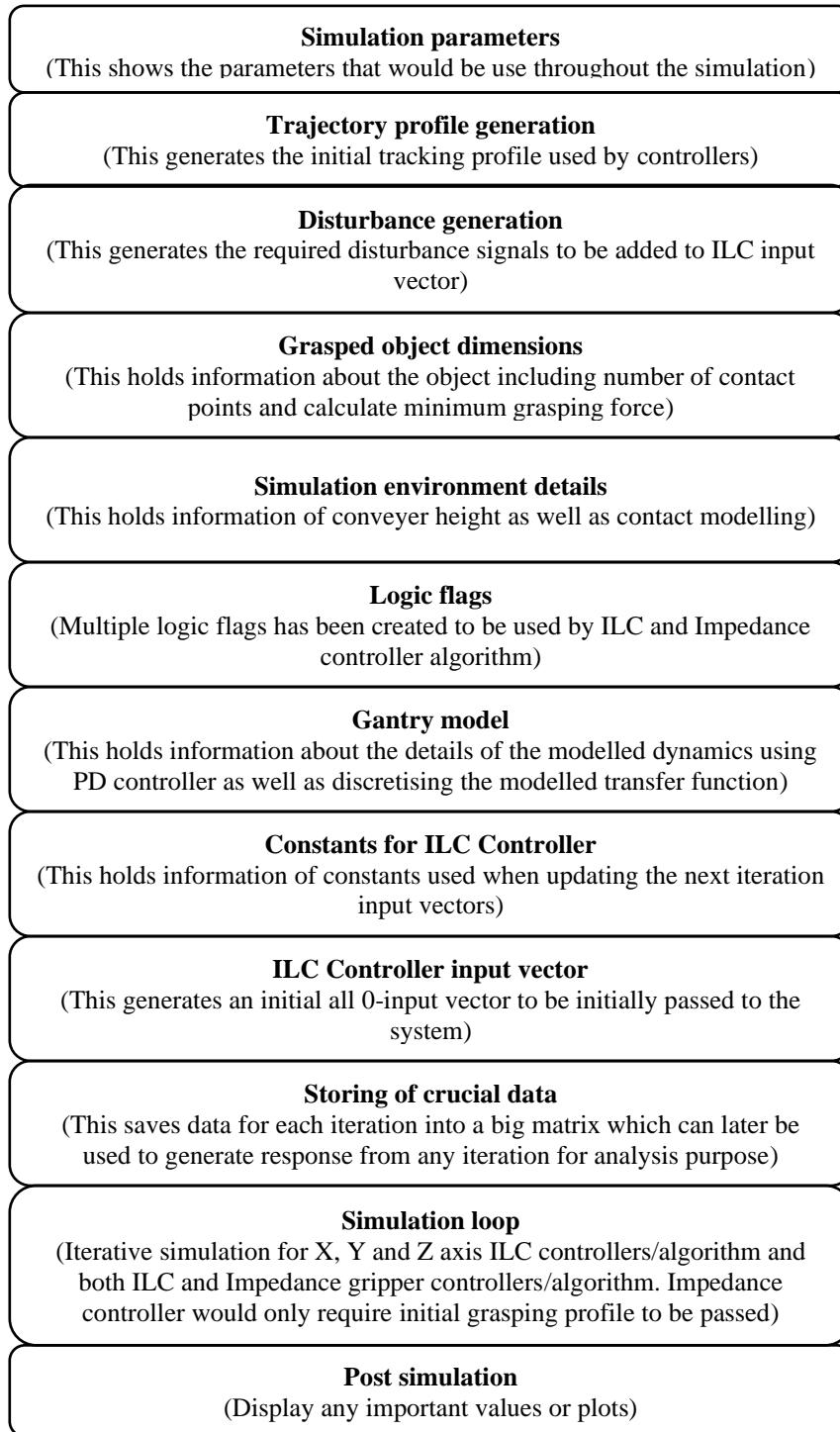
F desired logic tells the impedance-controlled gripper when the X, Y and Z axis has reached the threshold and thus it is safe to grasp the object. Grip desired logic passes the initial grasping trajectory profile when the gripper is safe to grasp. Grip Impedance Control on the other hand attempts to match the actual force applied by the gripper to the desired value by changing the grasping trajectory.



Appendix D. MATLAB Operational Code

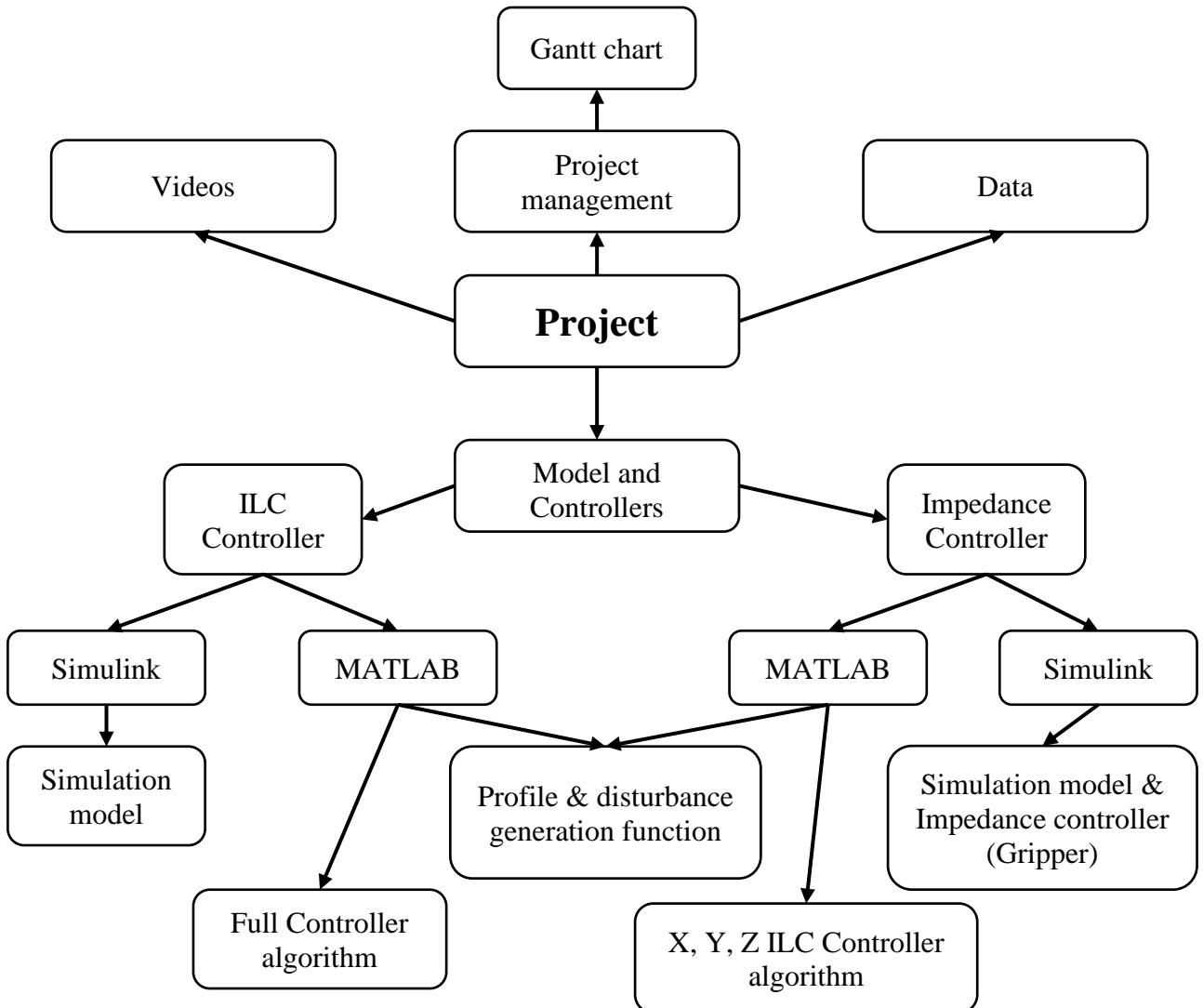
This section focuses on the MATLAB part of the project. Two different scripts would be used, one for ILC operation while the other for Impedance.

As the scripts are relatively long, therefore this section will highlight the main features of the code in chronological order as appeared in the actual raw script. Detail of the scripts can be found with the files attached to the submission which are explained in Appendix E.



Appendix E. Achieve File Overview

The following flow chart shows the location of each supporting document submitted alongside this report.



Model and Controllers – Contains all the files to build and run the model and controllers.

ILC Controller – Contains files for Gripper ILC Controller

Simulink – Files built and opened using MATLAB Simulink R2021b, including full simulation model (gantry and contact modelling) and/or gripper impedance controller.

MATLAB – Files built and opened using MATLAB R2021b, including trajectory profile and disturbance generation and/or ILC controller algorithm for X, Y, Z and/or Gripper.

Videos – Contains short videos of simulation for various interesting scenarios/cases

Project Management – Files used to effectively manage the progress and time of the project.

Gantt charts – Excel files for Gantt chart planning and progress tracking.

Appendix F. Project Brief

Problem

In this technologically rich era, our lives are filled with robotics as well as automation. From as simple as an autonomous robot vacuum machine to a more sophisticated multi degree of freedom robotic arm, all these systems required a state-of-the-art controller. As a result, the needs for faster, more precise, and accurate controller are ever more demanding.

There are numerous controllers out there capable of doing 300 picks per hour. On the other hand, the average human can grasp 600 times per hour. This makes robotic grasping twice as slow as an average human. One of the reason robotic grasping is much slower is due to the limited research conducted on robotic grasping, especially ones that incorporates Iterative learning control (ILC). ILC is crucial for robots that does repetitive motions such as a robotic grasping arm as it enables the controller to learn and improves from its own error and mistakes due to imperfection or just environmental factor.

ILC is not a new concept, in fact, it has been successfully applied onto numerous systems such as industrial robots, CNC machines tools, injection-molding machines and many more. However, applying ILC onto robotic grasping is relatively new and to the best of my knowledge there have been no experiments conducted into robotic grasping.

Goal

This project aims to:

- Read up literature, journals to identify different controllers as well as their advantages and points of improvements.
- Model accurately a 3-axis gantry robot as well as a simple gripper arm using Simulink simscape multibody toolbox.
- Attempts to design a new control algorithm which includes ILC to improve the accuracy and speed of the robotic gripper and simulate it using the model built on MATLAB.
- Test the control algorithm onto a real 3-axis gantry robot and identify any improvement or changes to be made to improve the controller and compare the result to the ones obtained via simulation.
- Further test the performance of the controller on a 6-DOF robotic arm.

Scope

My project aims to learn and identify different controller's strengths and weaknesses and adapt or attempts to design a new controller which incorporates ILC to improve the agility of robotic grasping. The project will start off by researching on different controller as well as modeling the gantry robot for simulation testing. The new controller will be tested first on MATLAB via simulation and will later be put to the test onto a real gantry robot. Works done previously on the gantry robot such as the model will be used as a reference and be improved further should any mismatch arises.

Furthermore, as an additional test, the controller will be programed onto a 6-DOF robot and be tested on it after receiving satisfactory results from the gantry robot.

I hope to design a new controller that is optimized for robotic grasping while incorporating ILC to improve its agility.