1. Overview of Robotics Grasping

Humans are prone to errors and fatigue especially when they are assigned to repeatable tasks. Robots are immune to that and can operate 24/7. However, the aim of robots is not to replace humans but to assists [1]. The simplest form is to just merely copy how human does tasks. There have been numerous developments on robotic hardware and robots are able to follow trajectory quite remarkably [1]. Manipulators are generally lacked in development. The importance of manipulator is the ability for robot to interact with an object. One reason that is holding back manipulators development is the lack of sensors that feedback to manipulator controller. Sensors generally comes in quite big in size and thus only a handful of sensors can be placed on a manipulator.

An effective robot manipulator needs to have the ability to grasps and manipulator various goods. Most manipulators today are capable of doing one specific task in a tightly controlled environment well [1][2]. As e-commerce grows, the demand for robot manipulators that can manipulate various goods are ever more demanding [1].

Amazon, one of the largest e-commerce company hosts an annual competition that challenge teams to design robot manipulator hardware and software that could identify and sort different items accurately [1].

Another challenge faced by robot manipulator is their speed. An average human could grasp around 400 to 600 items per hour [1]. One of the advance robot manipulator control system, Dex-Net 4.0 is only able to grasp roughly 300 items per hour [2]. The speed of robotic grasping has to increase to be at least on par with human's capability.

1.1 Problem Analysis

The main issue faced by current generation of robot manipulators is the speed at which they perform the tasks. To increase the speed, the whole robot system would have to work in a smooth and parallel manner. Most of the time, to increase the speed, simpler but yet able to yield satisfactory outcome controller should be use as it allows the planning and calculation to be short and allows the robot to perform next iteration with minimal dead time.

Fortunately, there has been countless controller designs out in the market with different range of complexity and performance. In the following section, some of these controllers will be discussed and analysed.

1.2 Robotic Grasping Approaches

The main aim of building a controller for robot manipulator is to allows the manipulator to move from one point to another securely while grasping an object. It came as no surprise that one of the earliest controllers was sequential controller which controls robot manipulator with a range of pre-programmed motion [3].

There is only a finite way of allows manipulator to grasp object with sequential controller. Reflective controller on the other hand relies on some form of sensing and thus has theoretically infinite ways of grasping an object. One of the approaches is to mimic a 10-month-old baby where the baby would grasp when an object came into contact with the palm. This allows the manipulator to adapt and grasp different objects [4]. Light sensor on the manipulator with react algorithm [5] was also tested as well as automatically selecting pre-grasp shape according to bitmap data sent to controller from a visual sensor was experimented with excellent result [6].

Sensors has been used to guide the manipulator to grasp an object, the next issue is to ensure the manipulator do not use excessive force. This motivates hybrid control which uses both force/torque and position data to control robot manipulator. The force feedback sensor is located at the wrist while the position sensor is situated at the manipulator. This combination proven by simulation and experiment that the manipulator is stable and accurate when subjected to different scenarios [7].

If the plant model is known, model-based controller can be used. One of the techniques is to adopt an idea of "inverse problem" also sometimes called "computed torque" where the input torque is calculated using a function that takes the difference between the desired and current position, velocity, and acceleration [8]. Ferrari and Canny on the other hand discussed two criteria required to plan an optimal grasp by using the idea of "force closure grasp" where a grasp is force closure if the grasp quality remains unchanged when an external force is exerted on it [9]. This method does not work well when the grasper has infinite ways to grasp an object which usually arises when the grasper is bigger than the object leading to the grasper having the possibility of grasping an object from any side in any orientation.

Another model-based controller is to utilise a learning controller. Nakayama et al. [10] proposed a learning controller that has k number of manipulators mimicking a single robot with certain geometric constrain attempting to hold one single object. The convergence of tracking position and force is proven by both simulation and theoretical as long as the multiple manipulator under consideration is initialised to the same posture each time. P type ILC along with temporally scaled force control has also been investigated with a 7DOF and has shown to improve both the speed and robustness of the system however theoretical proof was not demonstrated although the robot converges through experiment [11].

When the pre-knowledge of the dynamics parameters of a system is not known, one technique is to use adaptive grasp controller. Adaptive controller has been investigated to be able to estimate an initially unknown manipulator dynamics within the first second of operation. It also has the ability to match the robustness of as a PD controller while outperforming PD or computer torque controller in tracking performance [12]. Although the initial unknown dynamics could cause the behaviour of the controller to be unknown in the first few seconds of operation potentially causing harm to the system.

Schneider and Cannon [13] demonstrated that dynamic object impedance control is capable of performing most tasks in a fast and accurate manner. They show that a complex system such as a multi-arm robotic system can be controlled by a simple but yet powerful model-based impedance control.

Soft fingertips manipulators have some advantages over the traditional rigid fingertips such as being able to increase the surface of contact which in turn increases the grasp stability. However, soft manipulators make the grasping problem more complicated. Most cases, the controller for rigid manipulator can be used on soft manipulator however, it has been concluded that it is inadequate as the rolling distance changes according to the material [14]. Therefore, a new type of controller called passivity-based was introduced [2].

With the recent development of machine learning, there has been an increase of interest in using machine learning to assist in grasping. Particularly, cameras along with machine learning algorithm is capable of detecting different objects and plan out a solution to grasp them. Another approach is to use a human trainer to feed instructions into a neural network, after several hundred iterations, the model has acquired the skill of grasping an object reliably [1].

A summary of the controllers mentioned above can be found in figure 1 along with their dates of proposed.

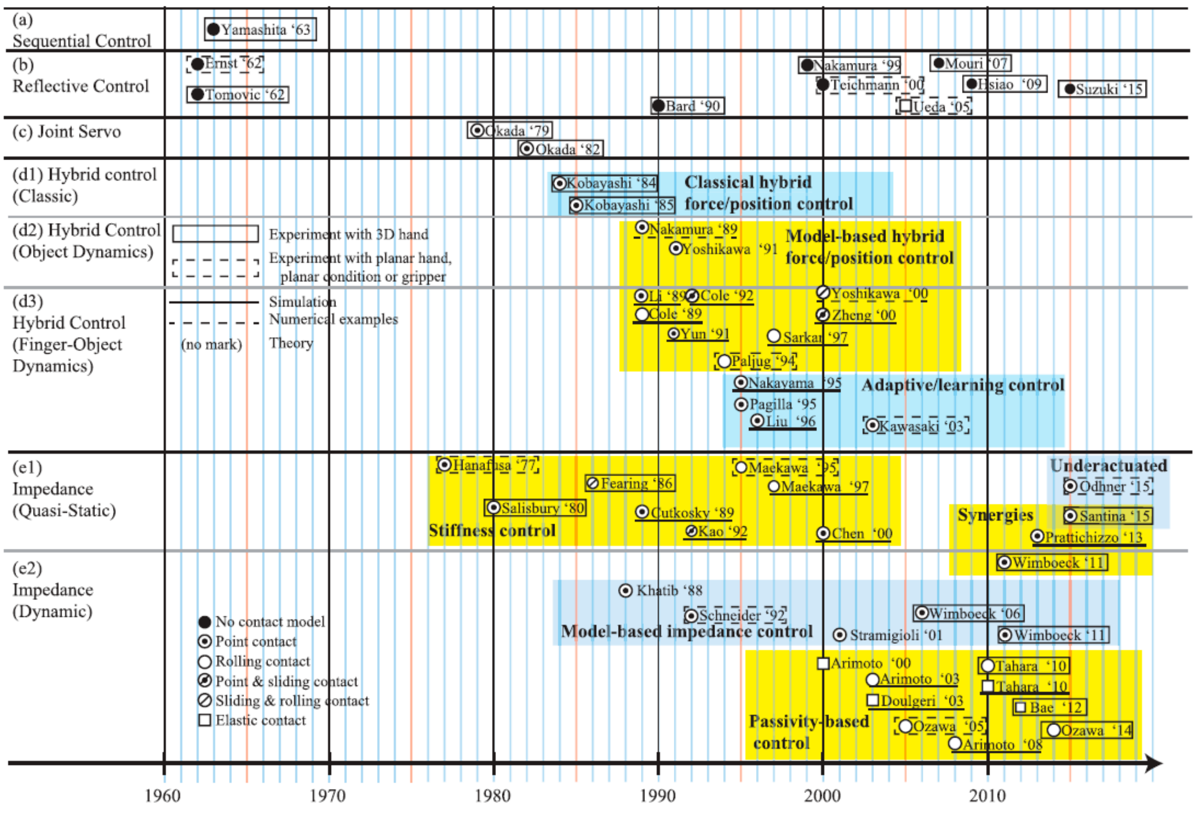

Figure 1: Order of different controllers developed for robotic grasping [2]

2. Overview of Iterative Learning Control

Most of the time, the error or imperfection of previous iterations contains valuable information. Iterative learning control (ILC) aims to exploit this area and learn from error occurred in previous iterations.

ILC is similar to repetitive control (RC). The difference between ILC and RC is the initial condition of ILC is reset every iteration while RC final condition is carried forward as initial condition in next iteration.

ILC are typically separated into 2 categories. Non-model based and model based, and several iterations are discussed briefly in the next section.

2.1 Non-model Based ILC

2.1.1 D Type

The first ILC was proposed by Arimoto et al. [15] where the derivative error from previous iteration is integrated into the current iteration to improve the performance as long as the previous iteration gives desired output response. The equation governing is defined as follows:

$$e_k(t) = y_d(t) - y_k(t) \tag{1}$$

$$u_{k+1}(t) = u_k(t) + \beta \dot{e}_k(t) \tag{2}$$

where k is the number of iterations, $y_d(t)$ is the desired trajectory, $y_k(t)$ is the trajectory output of the plan for the current iteration, $u_{k+1}(t)$ is the input of the next iteration, $u_k(t)$ is the input of current iteration, β is a constant mxm gain matrix and $\dot{e}_k(t)$ is the derivative of current iteration error.

The model will converge to zero error as long as the convergence criteria is satisfied when the number of iterations tends to infinity.

2.1.2 P Type

P type ILC is very similar to D type. Instead of using the derivative of the error, the error in its original form is used, yielding the following equation following Arimoto et al. [16]:

$$u_{k+1}(t) = u_k(t) + \beta e_k(t) \tag{3}$$

This controller however may lack the ability to detect and correct small noise signal as there is no derivative term which is known to significantly amplified small signals. [17]

2.1.3 PD Type

Arimoto et al. also proposed a mix type ILC where the P and D type works together similar to a PD feedback controller. This iteration is popular among the use for non-linear system as it does not require a plant to operate [18]. Although it can achieve convergence more readily than using P or D alone [17], it is not a guarantee that such success is always achieved unless the number of iterations is low [18].

2.2 Model Based ILC

2.2.1 Inverse Plant ILC

When the plant of the system is known, one of the simplest methods of ILC is just to invert the plant and use that to calculate the desired optimal input [19] yielding the equation in the form of:

$$u_d(t) = f_p^{-1}(y_d(t), t) \tag{4}$$

where $u_d(t)$ is the desired input signal, f_p^{-1} is the inverse of the plant and $y_d(t)$ is the desired output.

The next input iteration and error is formulated as eqn 5 and 6.

$$u_{k+1}(t) = u_k(t) + f_p^{-1}(t) e_k(t) \tag{5}$$

$$e_{k+1} = r - G u_{k+1} - d = (I - G f_p^{-1}) e_k \tag{6}$$

Since this uses the plant model, it is capable of converge in just one iteration. However, the performance is greatly hinder with any modelling error and random disturbances. Hence, it is generally not suitable to be used in real system where modelling error is unavoidable.

2.2.2 NORM-Optimal ILC

Norm-optimal ILC is consider the ‘best’ ILC controller. The algorithm developed by [20] aims to solve the optimisation problem in eqn 7 by ensuring reduction of norm of error is achieved in each iteration.

In this implementation, the step size for each iteration is automatically chosen and is capable of improving robustness by utilising the feedback from current iteration and feedforward data from previous iterations.

$$u_\infty = \arg min_u\{\left|\left|e\right|\right|^2 : \ e = r - y, y = Gu\} \tag{7}$$

The algorithm proposed that on completion of k iteration, the input of the next iteration is simply the solution of eqn 8 optimisation problem:

$$u_{k+1} = \arg min_{u_{k+1}}\{J_{k+1}(u_{k+1})\} \tag{8}$$

where the optimality criterion is given in eqn 9. Eqn 9 differs slightly from [20] as 2 scalar matrix Q and R are added to aid tuning in the hopes of balancing the choice of reducing tracking error and limit the change of input during iterations.

$$J_{k+1}(u_{k+1}) = Q||e_{k+1}||^2 + R||u_{k+1} - u_k||^2 \tag{9}$$

After taking the derivative of eqn 9 with respect to u_{k+1} , one would yield the following equation as the input for the next iteration:

$$u_{k+1} = u_k + (R + QG^TG)^{-1}QG^Te_k \tag{10}$$

Details of the steps and proof of convergence are omitted here, but they can be located in [20].

3. Detailed Specification

In the literature review, a range of different grasper controller was discussed. It can be concluded that almost all controllers were focusing on the quality of the grasp instead of the speed. Most controllers were tested on a single object. The problem discussed in section 1.2 still stands.

Therefore, a novel approach is to design a controller that can grasp objects of different dimensions while doing it in a fast manner without deteriorate the quality of grasping too much. This motivates the following design requirements.

3.1 Design Requirements [DR]

DR 1	-	Develop **simulation environment** for ease of testing different controller and evaluate controller performance for reaching and grasping.
DR 2	-	Develop **controller** for reaching and grasping that has the following properties
DR 2.1	-	Achieve **tracking error of less than 5%** in 25 iterations.
DR 2.2	-	Complete a cycle of reach and grasp **in less than 8 seconds**.
DR 2.3	-	Adapt to grasp objects that are **20% to 40% different in size and weight** compared to trained object.
DR 2.4	-	Reduce the grasping force to be **less than 20% of minimal frictional force**.

This specification will lead to if followed/met a grasping controller that is capable of grasping at least 450 different objects per hour accurately while ensuring the object does not get damaged due to excessive force.

4. Proposed Approach

In light with the design requirement specified in section 2. Several approaches were considered and settled on one final approach for each design requirement. This section will justify the approach selected.

4.1 Simulation Environment

There is numerous well established open-sourced simulation environment out in the market such as Gazebo and Webots. Gazebo is a decent simulator however it can be challenging to use for newcomers and the online resources to learn is quite limited. Webots on the other hand is simpler than Gazebo and has much wider resources on learning; still it is relatively complex compared to MATLAB Simulink.

MATLAB is a well-established numerical calculation software and has been used by many in the industry. Simulink is an add-on to MATLAB, while Simscape Multibody is built on top of Simulink. Simscape Multibody offers a Lego building structure where users connect different types of blocks such as joints, mass to create a realistic simulation model. This not only allows users to visualise the design and behaviour in a 3D world but also ensures it can be learnt quickly making it ease to use. Another advantage of using Simscape is its integration with MATLAB, MATLAB has a widely used control toolbox that could do almost anything, from step function to changing a transfer function to state-space form. As such, it is easy to switch between simulation purely in numerical form and simulation in 3D behaviours in Simscape with controller calculation done in MATLAB.

4.2 Controller

In the literature review, it can be concluded that most grasp controllers focus on the grasp quality by using different technique such as sensors, softness, machine learning and much more. Machine learning allows the controller to learn faster and could theoretically outperform traditional controllers. However, machine learning is very computational expensive and most of the time a simple controller would be enough to grasp an object. Most grasper does not consider error occurred in the previous iteration. Only two papers [10][11] has used learning controller. One was done in Simulation while the other uses a simple P-type ILC controller.

Since robotic grasping is a repetitive motion, it would be beneficial to learn from previous error. Hence, ILC is chosen to both learn previous iteration errors and to fulfil DR2 requirements. In chapter 2, it was discussed that model based ILC shown high convergence rate, this ensures the controller will reduce the error in the fastest manner while ensuring stability. Norm-Optimal ILC performs better when subject to unavoidable modelling uncertainties and external disturbances over Inverse Plant ILC.

4.3 Model

The model chosen to be used is built by Ratcliffe [17] during his PhD. This is a standard, simple gantry robot which can be found in many industries. The grasper would be changed from an electromagnetic grasper to a simple 2 Fingers Schunk MPC. A picture of the gantry robot can be seen in figure 2.

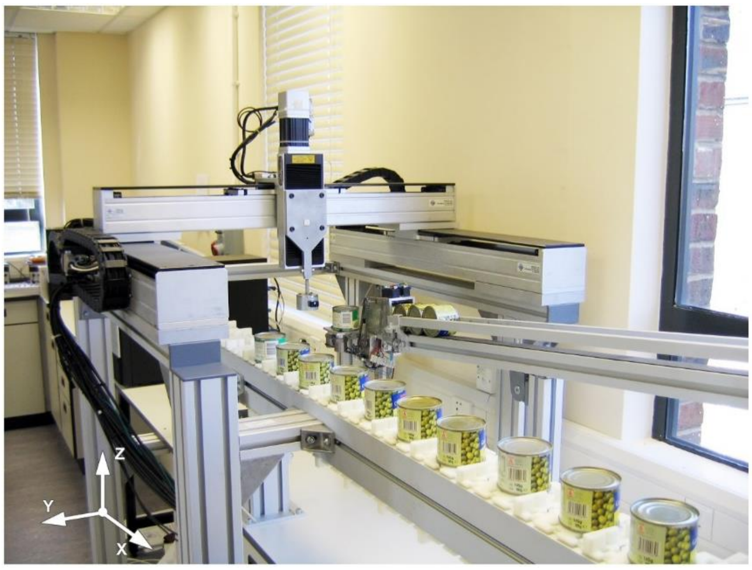

Figure 2: Gantry robot setup

4.4 High Level Design Overview

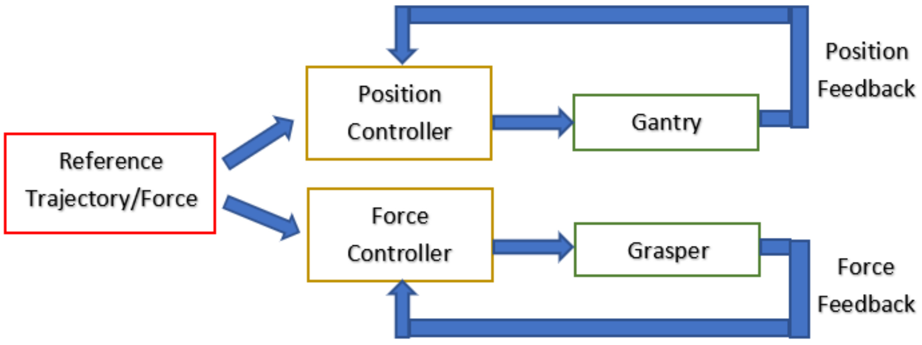

Figure 3: Design flow for the systems.

Figure 3 shows the high-level design flow of the proposed system. Position controller will have 3 independent controller each controlling each axis on the gantry robot similar to the controller in [17]. Whereas the force controller will control how much force is applied to the grasper to grasp an object.

The controller will be interchanged with Impedance, P-type ILC and Norm-Optimal ILC to evaluate their performance through simulation and experiment and thus justifying the used of Norm-Optimal ILC.

5. Gantry Model in Simulation Environment

The gantry model used developed by [17] allows each axis to be independently modelled and controlled. The x and y axis are modelled with 7 order transfer function while z axis is modelled by 4 order. It is difficult to model the actual gantry onto Simulink as it is very complex. The alternative is to use a simple model to model the dynamics of the system.

Using a solid block, the mass of each gantry can be modelled according to their dimension and density. Then the servo dynamics are modelled with a PD feedforward controller through a unity gain feedback. This allows the Simulink model dynamics to match the gantry dynamics as close as possible. Figure 3 shows the building blocks for the Simulink model. Translation motion of the system is modelled by prismatic joint specified in certain direction by the follower and base parameters.

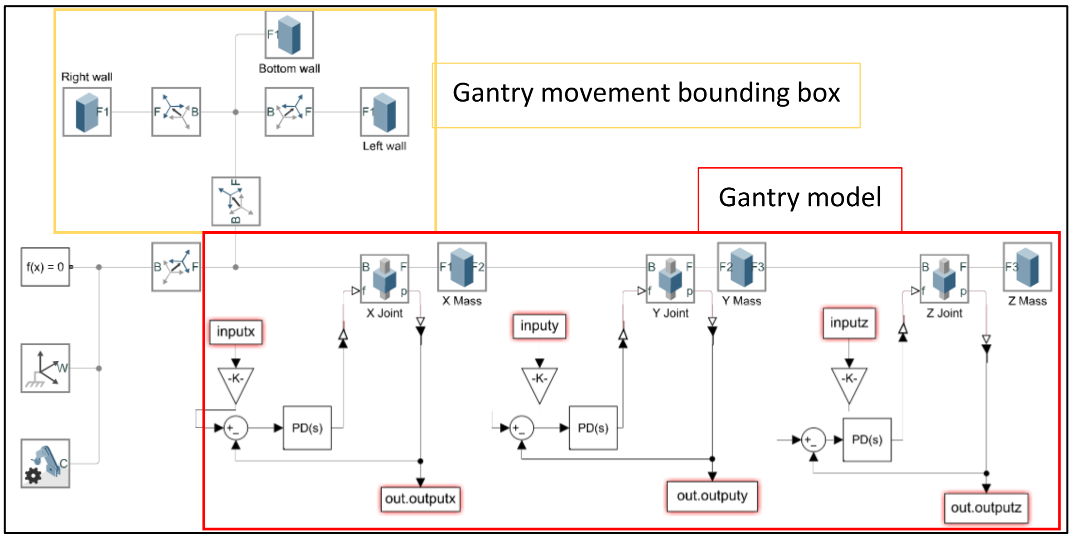

Figure 4: Simulink model of Gantry robot (Without grasper)

The Simulink model shown in figure 4 can be split into 2 major section. The bounding box provides the visualisation of the absolute physical limit of the gantry robot. The other section defines the dynamics of the gantry model in a simulation environment. Figure 5 shows the 3D model of the gantry robot.

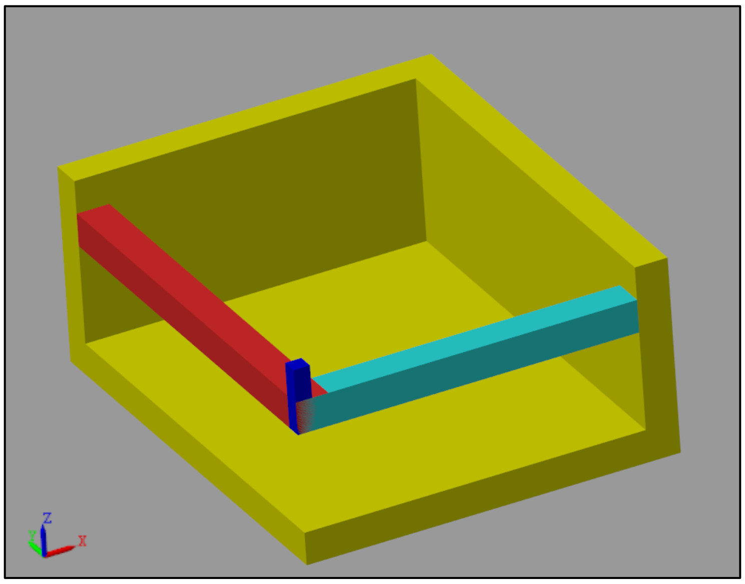

Figure 5: 3D visual of Simulink model of Gantry robot (Without grasper). Yellow shows the bounding box of the gantry robot axis movement, while red, cyan, and blue shows the gantry axes.

Unfortunately, Simulink does not have a built-in unit impulse input, but does have step input. However, the transfer function of each axis can be formulated in the form of $\frac{1}{s}(A)$, where $\frac{1}{s}$ is an integrator in S-domain. Coincidently, a step response is the same as an impulse response passed through an integrator. Therefore, in order to match the dynamics of actual model and Simulink model; the step response from the Simulink model has to be the same as the impulse response of the actual gantry model.

By tuning the PD controller, the response characteristics of Simulink model can be tuned to match the transfer function from the gantry model and is demonstrated in figure 6 with z axis.

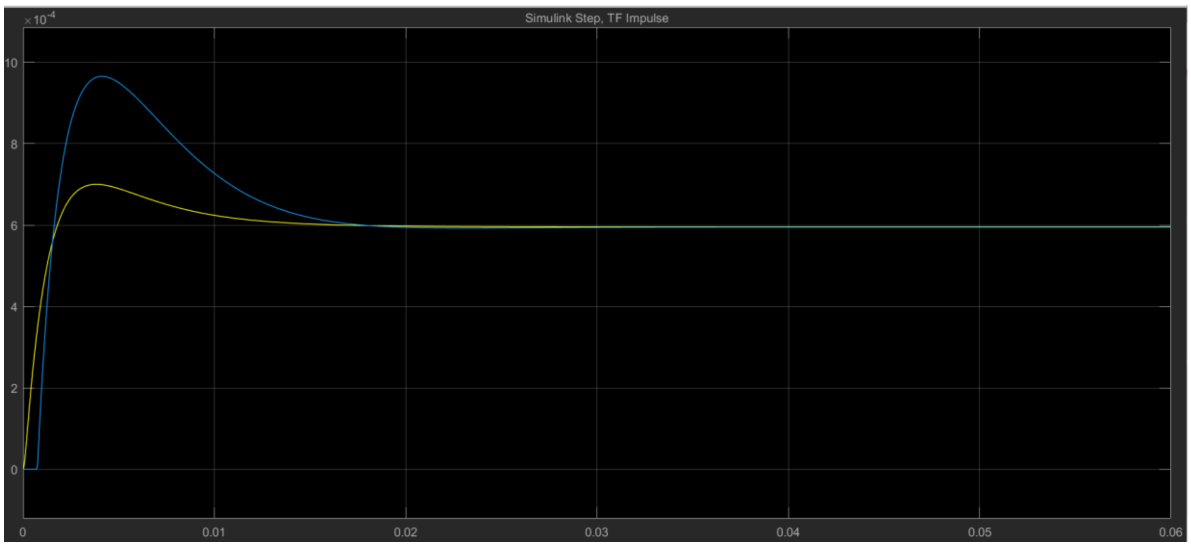

Figure 6: Impulse response of transfer function (Blue) and step response of Simulink model (Yellow) for Z axis

The solid object in Simulink environment has a transfer function of $\frac{1}{mS^2}$. When a PD controller is placed in a feedforward position, the whole system transfer function becomes:

$$\frac{K_dS+K_p}{mS^2+K_dS+K_p} \tag{11}$$

Which when evaluated with Final Value Theorem yields an output of 1. This shows the limitation of modelling a high order system with a 2nd order equation. Therefore, a gain block has to be added to reduce the steady state output to come in line with the transfer function response. This changes the form of eqn 11 to eqn 12 where m is the mass of the system under consideration. For Z axis it is just the Z axis itself, for Y axis it is Y axis plus Z axis. This can be seen in figure 2 where Y axis consists of the weight of itself plus Z axis.

$$\frac{K_dS+K_p}{(mS^2+K_dS+K_p)*Gain} \tag{12}$$

By using eqn 12, the transfer function of Z axis model can be formulated by substituting the tuned values for Kp, Kd, Gain and Mass. From figure 6, the percentage overshoot for Simulink model is less than the actual gantry response by a magnitude difference is in the form of 10^{-4}, which could be neglected. The transient time matches the actual system quite well. Similar tuning approach is done for X and Y axis and the values can be found in table 1.

	Kp	Kd	Gain	Mass (Sum of all affected axes)
X axis	27146.67	1020.03	1/12.5	4.32 kg
Y axis	131139.23	1480.58	1/500	3.51 kg
Z axis	33734.38	163.34	1/1550.67	0.1688 kg

Table 1: Values for Simulink model to yield response similar to gantry transfer function

6. ILC Controller for Trajectory Tracking

The controller used in this experiment is norm-optimal ILC. The G matrix from eqn 10 can be obtained by taking the impulse response of the discretised plant transfer function and remove all 0 elements on the diagonal side to obtain a 'lifted' form and toeplitz it to obtain the form:

$$G = \begin{pmatrix} CB & & & \\ CAB & CB & & \\ \vdots & \vdots & & \\ CA^{N-1}B & CA^{N-2}B & \dots & CB \end{pmatrix} \tag{13}$$

The R and Q matrix are carefully selected to fulfil DR 2.1 and 2.2 of chapter 3 and the values are shown in table 2.

	R	Q
X axis	0.1	30
Y axis	0.001	400
Z axis	0.0001	1500

Table 2: R and Q values for each axis Norm-Optimal ILC Controller

The controller with parameters states in table 2 was first evaluated with the true transfer function obtained from Ratcliffe [17] thesis and the graphical response of the L2 normalised error is shown in figure 7. Table 3 shows the L2 normalised error as well as the tracking accuracy evaluated by using eqn 14.

$$\frac{L2\ norm\ error\ at\ 15th\ iteration}{L2\ norm\ error\ at\ the\ beginning} \times 100\% \tag{14}$$

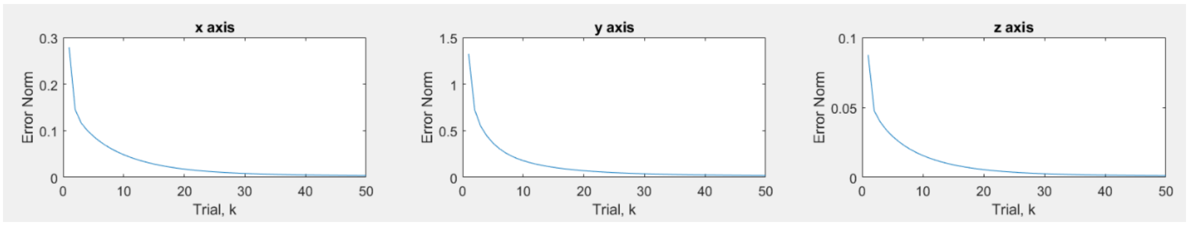

Figure 7: Norm tracking error for TF obtained from Ratcliffe thesis evaluated with 50 iterations with travelling time of 2 second

	Actual model numerical result			Simulink model result		
	Final norm error	Initial norm error	Percentage accuracy (%)	Final norm error	Initial norm error	Percentage accuracy (%)
X axis	0.0037	0.2789	4.21	3.21e-7	0.2789	3.11e-4
Y axis	0.0192	1.3254	3.82	4.32e-9	1.3254	3.32e-7
Z axis	0.0012	0.0876	4.16	5.61e-17	0.087	7.3e-14

Table 3: Results obtained from employing ILC with parameters found in table 2 to gantry actual model and Simulink model.

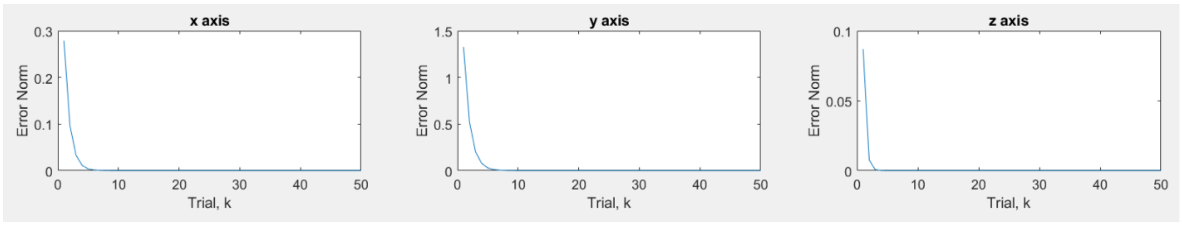

Figure 8: Norm tracking error for Simulink model evaluated with 50 iterations with travelling time of 2 second

Next, the controller was evaluated with the Simulink model. The corresponding normalised error reduction is shown in figure 8 while table 3 held the simulation result. It can be concluded that the same ILC performs remarkable better on the Simulink model compared to the actual model. This is caused by the simple second order system not able to capture the full dynamics of the actual model in Simulink, however both model converges, and Simulink provide an informative environment to visualised the performance.

7. Plans Going Forward

The gantry robot has been modelled into the simulation environment in chapter 5. Chapter 6 applies ILC to allow each axis to learn from previous error and improve performance. Plans going forward includes:

- Modify Simulink model to includes a 2-finger grasper along with contact points and object to grasp.
- Evaluate theoretical convergence to support simulation result.
- Evaluate the input force to the system to ensure it stays within the design limit of the gantry robot servo motor. Input force can be adjusted by the time set to complete each iteration.
- Compare the performance of ILC with other controllers (Impedance and P type ILC).
- Adapt to grasp different objects specified in DR 2.3.
- Adapt to use grasping force described in DR 2.4.
- Potentially overcome the limitation of simple second order model by implementing a more complex model to capture more system dynamics.
- Apply controller algorithm to experiment with physical gantry robot.