TABLE OF CONTENT

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

3D	Three dimensions
CAD	Computer aided design
CAE	Computer aided engineering
CAM	Computer aided manufacturing
ILC	Iterative learning control
P	Proportional
PD	Proportional-derivative
PID	Proportional-integral-derivative
TwinCAT	The Windows Control and Automation Technology

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

d_1	First link length [m]
d_2	Second link length [m]
d_3	Third link length [m]
L	Distance between first link joint with ground and third link joint with ground [m]
θ_1	Angle between first link and horizontal line passing through two joints on ground [rad]
θ_2	Angle between first link and second link [rad]
θ_3	Angle between third link and horizontal line passing through two joints on ground [rad]
θ_4	Angle between third link and second link [rad]
τ	Torque produced in motor [$N.m$]
$\dot{\theta}_1$	Derivation of θ_1 with time [$\frac{rad}{s}$]
$\dot{\theta}_2$	Derivation of θ_2 with time [$\frac{rad}{s}$]
$\dot{\theta}_3$	Derivation of θ_3 with time [$\frac{rad}{s}$]
$\ddot{\theta}_1$	Derivation of $\dot{\theta}_1$ with time [$\frac{rad}{s^2}$]
P	Power [$Watt$]
r_m	Motor shaft radius [m]
r_1	First link shaft radius connecting to ground [m]
m_1	First link mass [kg]
m_2	Second link mass [kg]
m_3	Third link mass [kg]
T	Kinetic energy [J]

V	Potential energy [J]
$\mathbf{Q}_{n.c}$	Vector of non-conservative forces [N]
L_{c_1}	Distance between center of gravity of first link and joint of first link with ground [m]
L_{c_2}	Distance between center of gravity of second link and the joint at either two ends [m]
L_{c_3}	Distance between center of gravity of third link and joint of third link with ground [m]
${}^{i}c_{j}$	Position of center of gravity of link j in i coordinate [m]
t	Time [s]
$\theta_1^d(\mathrm{t})$	First link desired angle [rad]
$\theta_1(\mathrm{t})$	First link real angle [rad]
$\mathbf{v}_{c_j}$	Velocity of first link center of gravity [$\frac{m}{s}$]
${}^{i}\boldsymbol{\omega}_{j}$	Angular velocity of link j in i coordinate [$\frac{rad}{s}$]
${}^{c}\mathbf{I}_{j}$	Moment of inertia of link j at its center of gravity along perpendicular to moving plane [$kg.m^2$]
$\mathbf{F(\theta,\dot{\theta})}$	Coriolis force matrix [N]
$\mathbf{M(\theta)}$	Mass matrix [kg]
n	Gear ratio
t_f	Total duration of trajectory from initial position to final position [s]
θ_0	Initial position of first link [rad]
θ_f	Final position of first link [rad]
$\mathbf{K}_p$	Matrix of proportional gains for computed torque
$\mathbf{K}_d$	Matrix of derivative gains for computed torque
$\mathbf{K}_i$	Matrix of integral gains for computed torque

k_d	Derivative gain for ILC
k_p	Proportional gain for ILC
k_i	Integral gain for ILC
T_r	Response time [s]
t_d	Delay time [s]

INTRODUCTION

The goal of virtual prototyping in mechanical design is to decrease the time and cost of manufacturing. Computer aided design (CAD), computer aided manufacturing (CAM) and computer aided engineering (CAE) are software that allow engineers to minimize the time and cost of manufacturing. CATIA, one of the Dassault system productions, is a well-known CAD/CAM/CAE software. However, unless a real model of the system is produced, or an additional module is added to the software, engineers cannot observe all the facets and defects of the system. To avoid all the defects with minimal time and cost, several prototyping methods have been developed. In these methods engineers aim to detect the problems of the system through testing the physical strength and geometrical limitations of the model. Until now, the focus has been on the mechanical aspect of the system. The present thesis will focus on the functionality of the system. In other words, our goal is to minimize the cost and time of dynamic modeling and controller design. In robot modeling, it is difficult to calculate the dynamics of the system with existing methods such as Lagrange or Newton-Euler. With these methods, mistakes are often made. We will avoid this risk by using CAD/CAM/CAE software to observe the dynamics of the system. Simulink is equipped with lots of libraries including SimMechanics. In SimMechanics we can use geometrical constraints to verify the behavior of the system in forward and inverse dynamics, so we no longer need to use the time-consuming Lagrange and Newton-Euler methods.

However, this procedure is still complicated, and can be further simplified. Imagine we have to design a robot containing hundreds of parts, including parallel or serial links, that works in a 3D space. Even in SimMechanics, modeling hundreds of parts with geometrical and mechanical constraints is not an easy task. We must also consider the task of manually adding the inertial properties of these hundreds of parts in SimMechanics. So each software has its own advantages and disadvantages: CATIA is extraordinary in 3D modeling but unable to analyze the dynamic, whereas SimMechanics is perfect in dynamic analysis but not user-friendly for 3D modeling. If only there were a way to use the benefits of each software and avoid their drawbacks, then we could save lots of time and effort.

In a cooperative project with IREQ (Institut de recherche d'Hydro-Québec), we developed a macro to convert CAD files from CATIA V6 to SimMechanics files, while respecting all the geometrical and inertial properties of the system. In this thesis, the macro is validated through a four bars parallel robot. We chose to use this simple robot, with just one degree of freedom, for validation because the analytical model can be found manually. So the macro will be validated by comparing the model obtained manually and the one obtained with the macro. In order to validate the system in the context of prototyping, two control laws will be generated using the macro: a computed torque position controller and an iterative learning controller. The computed torque controller calculates the required torque for the actuator based on the actual position and velocity of the robot and the desired acceleration. This controller uses the inverse dynamics of the model to calculate the required torque.

What about the popular PID controller? While these controllers are effective and easy to implement, they have some drawbacks which have caused engineers to turn to other kinds of controllers. One of these flaws is the tracking error. After stability, the most critical factor in control engineering is error. In industrial applications, most procedures are repeated through different cycles, such as during pick-and-place tasks. Since one system is under operation with similar trajectory and initial conditions, everything remains unchanged. In systems controlled by PID controllers, the same error pattern will therefore emerge along the trajectory in different cycles in the systems. However, iterative learning control (ILC) has the potential to solve this problem. ILC, which was first introduced by Arimoto (1985), takes advantage of the knowledge of error patterns to reduce the error in future cycles. We will use this technique to train the system, so that knowledge of the results in the previous cycle will be used to compensate for the error and converge it to near-zero.

The objective of this thesis is to provide a method of finding the dynamics of a mechanism without involving the more complicated analytical methods. For our purposes, it suffices to have the CAD model of the mechanism in CATIA V6. The dynamic model of a system allows us to better design a controller such as computed torque. In order to reduce the

tracking error of the robot, which undergoes a repetitive procedure with identical conditions at each iteration, we use ILC in a serial structure with the computed torque controller.

This work enabled users to save time when finding the dynamic model of a robot. It provides an easy way to design a computed torque controller for the system, and shows how to reduce the tracking error through the use of an ILC controller.

The first chapter presents a review of literature for robots, virtual prototyping, and ILC controllers. Chapter Two concentrates on a four bars parallel robot. It explains the kinematic and dynamic modeling of the robot, and discusses the method of trajectory planning and computed torque design for this robot. Chapter Three presents a method of virtual prototyping, and Chapter Four explains the different types of ILC. The simulation results are presented in Chapter Five, and the experimental results in Chapter Six. Chapter Six also contains some useful pieces of information about real robot set-up.

CHAPTER 1

REVIEW OF LITERATURE

1.1 Robots

A robot is a mechanical system that controls several degrees of freedom of the end-effector (e.g., the hand of the robot) (Tsai, 1999). The links, all of which are rigid, are connected to each other through revolute joints for rotation and prismatic joints for linear displacements. Robots are classified into two main categories, according to the configurations of their kinematic chains: serial robots and parallel robots.

1.1.1 Serial robots

Serial robots are made up of links that are attached in succession to each other by a one degree of freedom joint (Tsai, 1999). This configuration makes an open kinematic chain. Figure 1.1 shows an example of a serial robot. Serial robots typically experience problems regarding the mass and inertia of new links when they are added to the previous links. The actuators, from the end effector to the base, end up becoming larger and heavier. Their additional mass results in augmentation of the position error from the base to the end effector (Joubair, 2012).

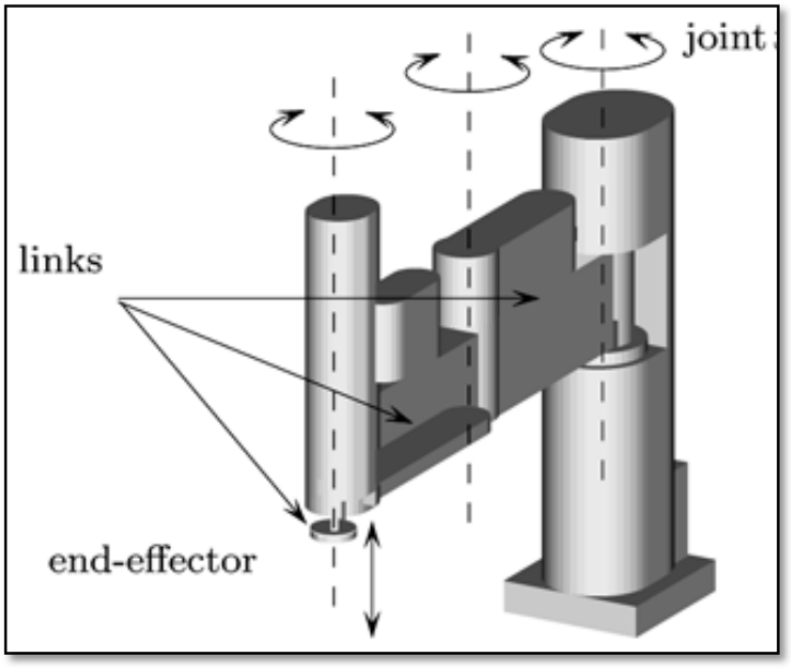


Figure 1.1 An example of serial robot: the Scara Robot
Taken from Merlet ((2006), p. 2)

1.1.2 Parallel robots

Usually, one can reduce the large mass and inertia of serial robot links by distributing the load across several links, which are attached in a parallel configuration from the end effector to the ground. In doing so, one creates a parallel robot, an example of which is shown in Figure 1.2. A parallel manipulator is generally defined as a closed-loop kinematic chain mechanism whose end-effector is linked to the base by several independent kinematic chains (Merlet, 2006). Parallel robots are more rigid, due to their closed-loop mechanical structure. Typically the actuators are attached to the base, enabling good precision since the mass of the moving part is reduced (Joubair, 2012). Unlike with serial robots, however, not all the joints of a parallel robot are actuated. While some of them are active (also known as actuated), others are passive. Because of the geometrical constraints imposed by closed-loop mechanical chains, the singularity problem is more likely to occur with parallel robots than with serial ones.

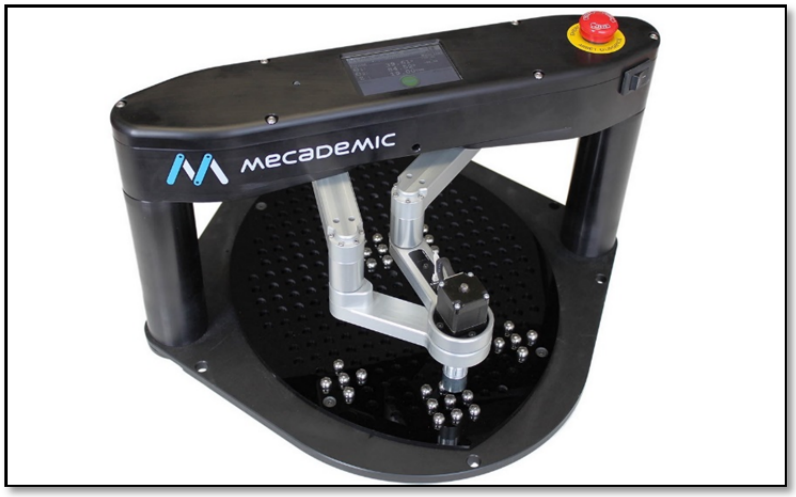


Figure 1.2 DexTar Robot; an example of parallel robot
Taken from Bonev (2013)

1.2 Virtual prototyping

We will give just a brief overview of virtual prototyping here, because the purpose of this thesis is not to present a new prototyping tool; instead we are validating a prototyping tool developed by other people in our laboratory. Virtual prototyping is a relatively new concept in control engineering. In order to have accurate and robust control over the system, one must have comprehensive knowledge of its dynamics. This knowledge is used to develop an inverse dynamic controller that compensates for the nonlinearity in the system, which is caused by forces such as inertia, centrifugal force, and gravity (Jagannathan, 2001). For example, Yeon et al. developed an inverse dynamic controller by using SimMechanics to design a computed torque for a HyRoHILS robot (Yeon et al., 2005). With this method, Yeon et. al used the geometrical and mechanical properties of the CAD file to generate the dynamic model.

1.3 Iterative learning control

One of the primary aims in control engineering is to reduce the error between a reference, which may be a trajectory, and the system output. But when conventional controllers such as

PID are combined with computed torque, and the trajectory is repetitive, the tracking error typically does not change in different cycles.

In industry several processes are repetitive, particularly tasks accomplished by robots. The process is repetitive when all the conditions are identically repeated in all cycles. For example, a gripper picks up an object with mass m from point x_1 and carries it through a specific trajectory; it delivers the object to point x_2 and then returns back to x_1 to pick up the next object with the same mass m. This would be one process. In the next cycle the robot undergoes the same process and does the same duty. Under these conditions it is normal to see the same error in each cycle of the process, since all the factors such as friction, dynamic model, input, controller and trajectory are remaining constant.

If we can change the input of the system in a specific way so that the output will more closely approach the desired trajectory, then we are successful in reducing the error. But what is this specific way? This method is called Iterative Learning Control or ILC. An ILC block diagram is shown in Figure 1.3.

With ILC, the robot tries to learn its error and correct its commands accordingly in order to reduce the error in the next cycle. If we repeat this procedure several times, we expect that at each cycle the error will be smaller than in the previous one, and so eventually the error will converge to near-zero.

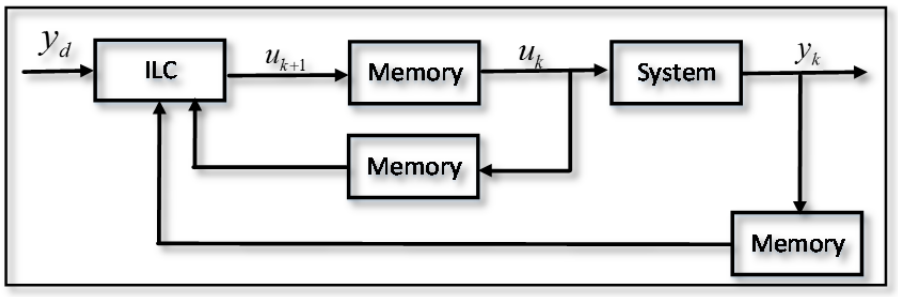


Figure 1.3 ILC block diagram

The output will be stored in memory in the k *-th* cycle, and will be used during the $(k+1)$ *-th* cycle to change the input.

1.3.1 Iterative learning control definition

There are several definitions of ILC, but the following two descriptions represent the general consensus:

- "The learning control concept stands for the repeatability of operating a given objective system and the possibility of improving the control input on the basis of previous actual operation data" (Arimoto, Kawamura and Miyazaki, 1986).
- ILC is a "recursive online control method that relies on less calculation and requires less a priori knowledge about the system dynamics. The idea is to apply a simple algorithm repetitively to an unknown plant, until perfect tracking is achieved" (Bien and Huh, 1989).

1.3.2 Iterative learning control history

The idea of ILC was initially suggested in 1974 by (Edwards, 1974), with its first formulation presented in Japan by Uchiyama in 1978 (Uchiyama, 1978). A learning configuration is shown in Figure 1.4.

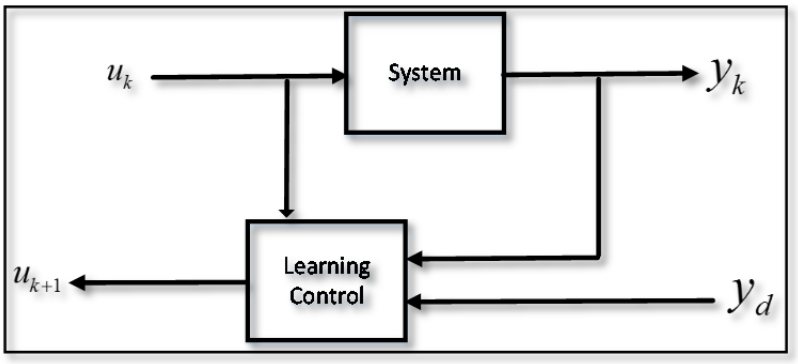


Figure 1.4 Learning control configuration

Later during 1980s, the idea was further developed by a group of researchers under Arimoto's supervision (1984). In the resulting article, Arimoto proposed this updating law:

$$u_{k+1} = u_k + \Gamma \dot{e}_k \tag{1.1}$$

The convergence is guaranteed according to the well-known small gain conditions. Here Γ is a constant matrix, which is the gain of the updating law. This algorithm is PD-type ILC because it uses the derivative of the error. They also proposed other types of ILC (Arimoto, 1985), such as the following PID-type:

$$u_{k+1} = u_k + \Phi e_k + \Gamma \dot{e}_k + \Psi \int e_k dt \tag{1.2}$$

Togai and Yamano (1985) and Furuta and Yamakita (1987) focused on learning control for discrete-time linear systems. So far, all the algorithms we have discussed used a unity weighting on input. By contrast, Mita and Kato (1985) used separate weighting for the input and error. Their updating law is:

$$U_{k+1}(\mathrm{s}) = L(s)[U_k(s) + aE_k(s)] \tag{1.3}$$

Atkeson and McIntyre (1986) used the same approach to learning control as Arimoto et al., applied to linear robotic manipulator models. Hidge and Judd (1988) considered learning control from a frequency domain point of view, which is similar to Mita and Kato's work. They also took the effect of disturbance into account, and applied it to the linear robotics models. Oh, Bien and Suh (1988) proposed an approach to learning control for linear time varying systems. Arimoto et al. (1984) also considered learning control for robotics, and Craig (1984) and Gu and Loh (1989) suggested learning and adaptive control methods that are similar to those of Arimoto. Harokopos (1986) worked on minimizing the functional cost for robotics. Bondi, Casalino and Gambardella (1988) presented a high-gain feedback model for learning control and applied it to nonlinear systems including manipulators. Yamakita and Mita (1991) researched the domain of nonlinear systems through the use of Gateaux derivatives. Messner, Horowitz, Kao and Boals (1991) developed a new method for

nonlinear manipulators. Hauser (1987) suggested the idea of using learning control to find the inverse dynamics for nonlinear systems. Heinzinger, Fenwick, Paden, and Miyazaki (1989) investigated the robustness of nonlinear systems in the ILC domain. Sugie and Ono (1991) also worked on the robustness of iterative learning control. Lastly, (Ahn, Moore and Chen, 2007) produced a survey of ILC research including journals articles and Ph.D. dissertations from 1998 to 2004.

1.3.3 Assumptions used for ILC

As mentioned by Gauthier in his Ph.D. dissertation (Gauthier, 2008), the five key assumptions used for ILC are:

- The initial state of the system remains identical throughout the process. Then, $x_k(0) = x(0), \forall k$ and this initial condition gives the starting point of the desired trajectory $y_k(0) = C_k(0)x_k(0), \forall k$;
- The system can be time-varying, but each cycle should be the same as the other cycles. Then, for a system expressed in the state-space domain,
  $A_k(t) = A(t),\ B_k(t) = B(t),\ C_k(t) = C(t),\ \forall t \in [0,T]$ and $\forall k$;
- The desired trajectory $y_d(t)$ (with $t \in [0,T]$) must be feasible. Then it should be possible to have an input $u^*(t)$ for $t \in [0,T]$ such that the output follows the desired trajectory. The input is a $L_2[0,T]$ continuous function of time;
- $u^*(t)$ should be unique to obtain the desired trajectory $y_d(\mathrm{t})$;
- The cycle duration $T \in \square$ is similar from cycle to cycle (but there are some exceptions)

All five of these assumptions are held throughout this thesis.

CHAPTER 2

ROBOT MODELING

2.1 Introduction

Parallel robots play an important role in industry. Their robustness, high speed capabilities, and precision give them great advantages when it comes to certain manufacturing tasks.

This thesis uses a parallel robot with three links (four bars, including the ground) and one actuated joint. Figure 2-1 depicts a scheme of this robot.

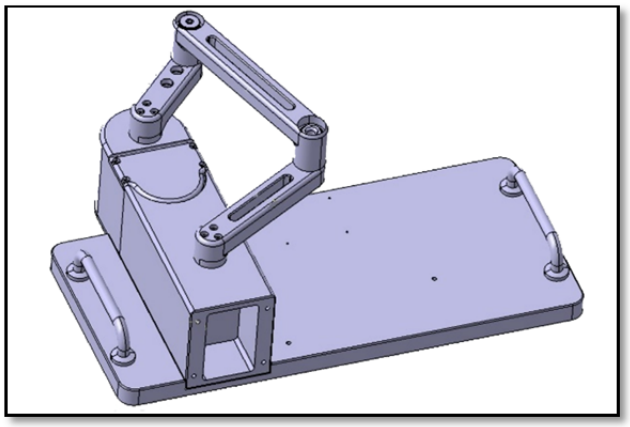


Figure 2.1 Four-bar parallel robot

We chose this robot for validation because although it has just one degree of freedom, its dynamic model is nonlinear due to its parallel structure. So although it is a simple hardware system, its dynamics are complex.

2.2 Kinematic model

2.2.1 Direct kinematic

According to Figure 2-2, we find both θ_2 and θ_3 relative to θ_1, which is the input angle (actuated joint). Figure 2-2 shows the geometric model of the robot links with their parameters.

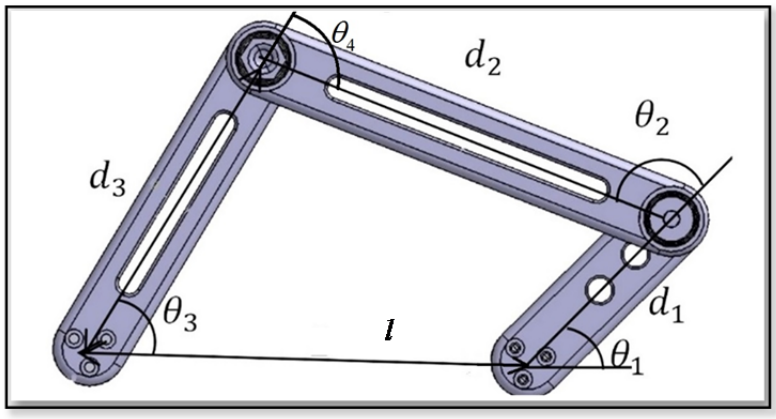


Figure 2.2 Robot links with parameters

In this figure, *l* is the distance between the two joints attached on the base, d_1 is the length of the first link (the actuated link with motor), d_2 is the length of the second link, d_3 is the length of the third link, θ_1 is the angle between the first link and the line passing through two joints on the ground, θ_2 is the angle between the first and second link, θ_3 is the angle between third link and the line passing through the two joints on the ground, and θ_4 is the angle between the second and third link.

To find θ_2 and θ_3 relative to θ_1, we consider the links as vectors in two dimensional Cartesian space. This consideration is shown in Figure 2.3.

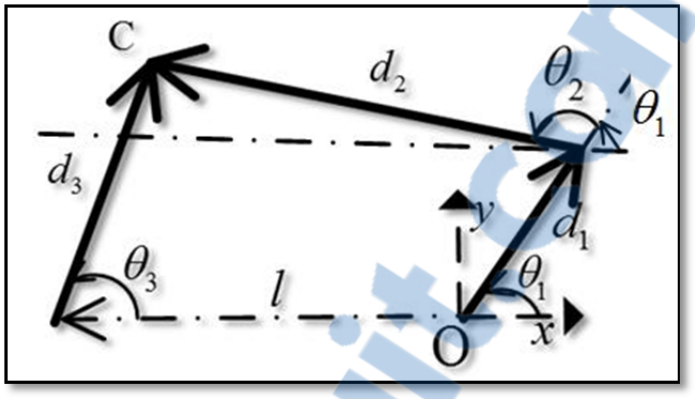


Figure 2.3 Vector representation of robot in Cartesian space

O is the *x*-*y* coordinate origin, and C is the connecting point of d_2 and d_3. So we can find two equations (one for the x- and one for the y-direction) relating θ_1 to θ_2 and θ_3 by posing the equality between the two positions C at the ends of d_2 and d_3. Vectors can be projected on *x* and *y* directions independently.

$$\begin{cases} x\text{-direction} \rightarrow d_1\cos(\theta_1) + d_2\cos(\theta_1+\theta_2) = -l + d_3\cos(\theta_3) \\ y\text{-direction} \rightarrow d_1\sin(\theta_1) + d_2\sin(\theta_1+\theta_2) = d_3\sin(\theta_3) \end{cases} \tag{2.1}$$

To solve this set of equations we will keep the θ_2 terms on the left side of equation and the rest on the right. Then we will add the square of first equation to the square of the second equation to get rid of θ_2.

$$\begin{cases} (d_2\cos(\theta_1+\theta_2))^2 = (-l + d_3\cos(\theta_3) - d_1\cos(\theta_1))^2 \\ (d_2\sin(\theta_1+\theta_2))^2 = (d_3\sin(\theta_3) - d_1\sin(\theta_1))^2 \end{cases} \tag{2.2}$$

By summing the two equations and expanding them we have:

$$\begin{aligned} &d_1^2 + d_3^2 - d_2^2 + l^2 + 2d_1 l\cos(\theta_1) - 2d_3 l\cos(\theta_3) \\ &-2d_1 d_3\cos(\theta_1)\cos(\theta_3) - 2d_1 d_3\sin(\theta_1)\sin(\theta_3) = 0 \end{aligned} \tag{2.3}$$

We will now find an equation which relates $\cos(\theta_3)$ to θ_1. For this purpose we use $\sqrt{1-\cos(\theta_3)^2}$ instead of $\sin(\theta_3)$. The equation is:

$$\begin{aligned} &d_1^2 + d_3^2 - d_2^2 + l^2 + 2d_1 l\cos(\theta_1) - 2d_3 l\cos(\theta_3) \\ &-2d_1 d_3\cos(\theta_1)\cos(\theta_3) = 2d_1 d_3\sin(\theta_1)\sqrt{1-\cos(\theta_3)^2} \end{aligned} \tag{2.4}$$

This equation can be sorted according to the terms of $\cos(\theta_3)$ as follows:

$$\mathrm{C} = d_1^2 + d_3^2 - d_2^2 + l^2 + 2d_1 l\cos(\theta_1) \tag{2.5}$$

$$\mathrm{A} = -2d_3 l - 2d_1 d_3\cos(\theta_1) \tag{2.6}$$

$$\mathrm{B} = 2d_1 d_3\sin(\theta_1) \tag{2.7}$$

For simplicity we will use the following notation:

$$\cos(\theta_3) = \mathrm{X} \tag{2.8}$$

Equation (2.4) can be rewritten in the following form:

$$\mathrm{C} + \mathrm{AX} = \mathrm{B}\sqrt{1 - \mathrm{X}^2} \tag{2.9}$$

To find X we will square both sides of the equation (2.9):

$$\mathrm{C}^2 + \mathrm{A}^2\mathrm{X}^2 + 2\,\mathrm{ACX} = \mathrm{B}^2(1 - \mathrm{X}^2) \tag{2.10}$$

and rewrite it in the following form:

$$a\,\mathrm{X}^2 + b\,\mathrm{X} + c = 0 \tag{2.11}$$

where

$$a = \mathrm{A}^2 + \mathrm{B}^2 \tag{2.12}$$

$$b = 2\,\mathrm{CA} \tag{2.13}$$

$$c = \mathrm{C}^2 - \mathrm{B}^2 \tag{2.14}$$

Equation (2.11) is a simple second degree equation. To solve this equation we will calculate the discriminant as follows:

$$\Delta = b^2 - 4ac \tag{2.15}$$

And finally the answers to X (or $\cos(\theta_3)$) are:

$$\cos(\theta_3)_1 = \frac{-b + \sqrt{\Delta}}{2a} \tag{2.16}$$

$$\cos(\theta_3)_2 = \frac{-b - \sqrt{\Delta}}{2a} \tag{2.17}$$

By solving these two equations we find the two possible answers for θ_3. So there are two answers for angle θ_1, which corresponds to the positions of links 2 and 3 in up or down positions, as shown in Figure 2.4.

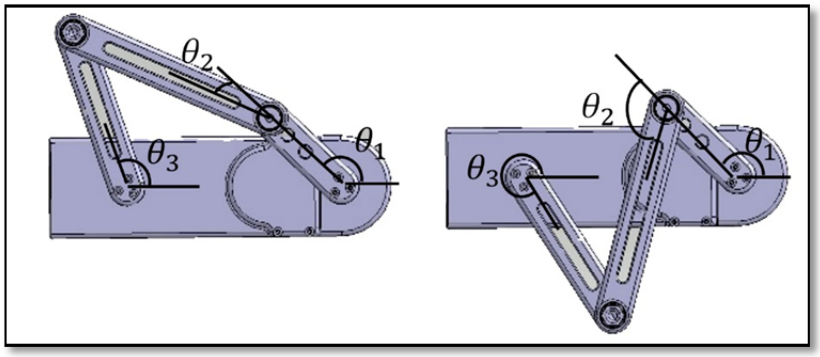


Figure 2.4 Possible solutions for θ_1 Position
up (left) – Position down (right)

The left configuration in Figure 2.4 is the current configuration of robot. Since the third link does not go through a full rotation, due to the length of the link, the correct answer according to our configuration choice is the one with θ_3 between 0 and π radians. Once we have found θ_3, it is easy to find θ_2 by using equation (2.1).

2.2.2 Singularities

For this robot, there are two positions in which we have serial singularities. Figure 2.5 presents these two singular positions, at $\theta_2 = 0$ and $\theta_2 = \pi$. The dimensions of the links are chosen to avoid parallel singularities. There are two conditions to ensure this avoidance:

$$\begin{cases} d_1 + d_2 \leq d_3 + l \\ d_1 \rightarrow \text{actuated link} \end{cases}$$

where d_1 is the length of the shortest link, which is equal to 68.58 mm. This link is actuated. d_2 is the length of the longest link, which is equal to 149.225 mm. The lengths of the two remaining links are denoted by d_3 and l, respectively, with l equal to 149.225 mm and d_3 equal to 114.30 mm. In this robot design, these two conditions are respected.

$$68.58 + 149.225 \leq 149.225 + 114.30$$

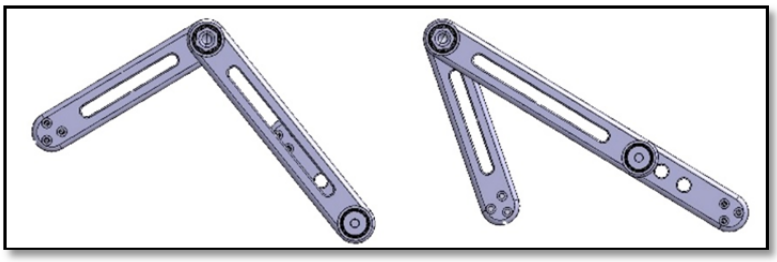


Figure 2.5 Two serial singularities

2.3 Dynamic model

For the dynamic model we will consider that the motor is directly mounted on the first link and we will neglect any flexibility. In this section, the matrix of mass and vector of nonlinear forces will be derived according to the Lagrange method (Craig, 2005). This is done in order to compute the necessary torque for the motor to follow a desired trajectory. The ratio of the driven pulley to the drive pulley is 4:1, meaning the angular velocity of the drive pulley is four times greater than the angular velocity of the driven pulley. It is also known that (when the mechanical loss is ignored) the power generated by the motor is equal to the power consumed by the robot, as described by the following:

$$P_{in} = P_{out} \tag{2.18}$$

$$P_{in} = \tau_m \dot{\theta}_m \tag{2.19}$$

$$P_{out} = \tau_1 \dot{\theta}_1 \tag{2.20}$$

where P_{in} is the power generated at input, P_{out} is the power consumed at output, τ_m is the torque generated by the motor, $\dot{\theta}_m$ is the angular velocity of the motor, τ_1 is the torque applied to the first link, and $\dot{\theta}_1$ is the angular velocity of the first link. Suppose that the absolute linear velocity of the timing belt is the same value at every single point of it, so where the belt is connected to the drive pulley (point A in Figure 2.6) and driven pulley (point B in Figure 2.6) pulleys, we can conclude that:

$$r_m \dot{\theta}_m = r_1 \dot{\theta}_1 \tag{2.21}$$

$$\frac{\dot{\theta}_1}{\dot{\theta}_m} = \frac{r_m}{r_1} \tag{2.22}$$

$$\tau_m \dot{\theta}_m = \tau_1 \dot{\theta}_1 \tag{2.23}$$

$$\tau_m = \tau_1 \frac{r_m}{r_1} \tag{2.24}$$

where r_m is the radius of the motor or drive pulley, and r_1 is the radius of the driven pulley.

For our specific design,

$$\frac{r_m}{r_1} = \frac{1}{4}$$

Thus,

$$\tau_m = 0.25\tau_1$$

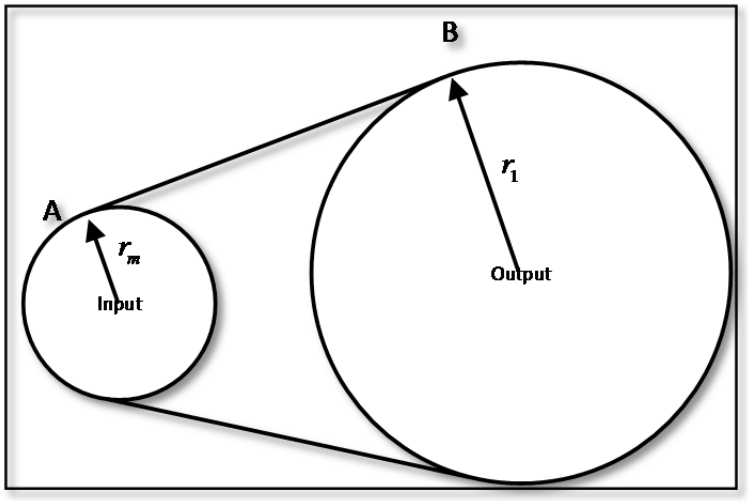


Figure 2.6 Schema of drive pulley and driven pulley connected by timing belt

We can find the right torque for the motor by simply dividing the computed torque for the axis of the first link by four.

Since the four-bar robot is a closed loop mechanism, its dynamic model has to take into account the constraint cause by the closed loop. One way to obtain this model is the Lagrange-d'alembert formulation (Murray et al., 1994). With this approach, the first step is to open the mechanical loop and calculate the open loop model with the Lagrange approach by using:

$$\frac{d}{dt}\frac{\partial T^T}{\partial \dot{\boldsymbol{\theta}}} - \frac{\partial T^T}{\partial \boldsymbol{\theta}} - \mathbf{Q} = 0 \tag{2.25}$$

where T is the kinetic energy, $\boldsymbol{\theta}$ is $(\theta_1, \theta_2, \theta_3)$, $\dot{\boldsymbol{\theta}}$ is $(\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3)$. $\mathbf{Q}$ is the vector of generalized forces,

$$\mathbf{Q} = -\frac{dV}{d\boldsymbol{\theta}} + \boldsymbol{\tau} + \mathbf{Q}_{nc} \tag{2.26}$$

where $\mathbf{Q}_{nc}$ and $\boldsymbol{\tau}$ are the vectors of non-conservative forces and V is the potential energy.

As shown in Figures 2-7 and 2-8, we chose to open the loop at point C. For the open loop configuration, we thus consider link one and two as the first chain and link three as the second chain.

Since all arms are moving in a plane perpendicular to the direction of gravity, the potential energy is constant and the variation is zero:

$$dV = 0 \tag{2.27}$$

Then, the kinematic energy of the 3 links in the open loop configuration is:

$$T = \frac{1}{2}\sum_{i=1}^{3} m_i \mathbf{v}_{c_i}^T \mathbf{v}_{c_i} + {}^{i}\boldsymbol{\omega}_i^T \, {}^{c}\mathbf{I}_i \, {}^{i}\boldsymbol{\omega}_i \tag{2.28}$$

where $\mathbf{v}_{c_j} \in \mathbb{R}^3$ is the velocity of the center of gravity of link $\mathbf{I}$, ${}^{c}\mathbf{I}_i \in \mathbb{R}^{3\times3}$ is the moment of inertia of link *i* about its center of gravity, and ${}^{i}\boldsymbol{\omega}_i$ is the angular velocity of link *i*. To find the velocity of the center of gravity, we will first find the position of this point, and then we will derivate it. The position of the first link's center of gravity is:

$${}^{0}\mathbf{c}_1 = \begin{bmatrix} \mathrm{c}_1 L_{c_1} \\ \mathrm{s}_1 L_{c_1} \\ 0 \end{bmatrix} \tag{2.29}$$

where L_{c_1} is the center of gravity of the first link, ${}^{i}\mathbf{c}_j$ is the position of the center of gravity of link *j* at the *i* coordinate, c_1 is $\cos(\theta_1)$, and s_1 is $\sin(\theta_1)$. Derivation of (2.29) gives the velocity of the center of gravity of the first link. The velocity of the first link's center of gravity, $\mathbf{v}_{c_1}$, is:

$$\mathbf{v}_{c_1} = \begin{bmatrix} -s_1 L_{c_1} \dot{\theta}_1 \\ c_1 L_{c_1} \dot{\theta}_1 \\ 0 \end{bmatrix} \tag{2.30}$$

The angular velocity of the first link is:

$$ {}^{1}\boldsymbol{\omega}_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \tag{2.31} $$

where ${}^{i}\boldsymbol{\omega}_j \in \mathbb{R}^3$ is the angular velocity of link j projected in i coordinate. The velocity of the center of gravity of the second link is calculated similarly to the first link. First, the position of this point must be found. It is:

$$ {}^{0}\mathbf{c}_2 = \begin{bmatrix} \mathrm{c}_1 d_1 + \mathrm{c}_{12} L_{c_2} \\ \mathrm{s}_1 d_1 + \mathrm{s}_{12} L_{c_2} \\ 0 \end{bmatrix} \tag{2.32} $$

where s_{12} is $\sin(\theta_1 + \theta_2)$, and c_{12} is $\cos(\theta_1 + \theta_2)$. Then the derivative of (2.32) gives the velocity of the center of gravity of the second link:

$$ \mathbf{v}_{c_2} = \begin{bmatrix} -\mathrm{s}_1 d_1 \dot{\theta}_1 - \mathrm{s}_{12} L_{c_2} (\dot{\theta}_1 + \dot{\theta}_2) \\ \mathrm{c}_1 d_1 \dot{\theta}_1 - \mathrm{c}_{12} L_{c_2} (\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix} \tag{2.33} $$

The angular velocity for the second link is:

$$ {}^{2}\boldsymbol{\omega}_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix} \tag{2.34} $$

Just like before, we will find the position of the center of mass of the third link:

$$ {}^{0}\mathbf{c}_3 = \begin{bmatrix} \mathrm{c}_1 L_{c_1} - l \\ \mathrm{s}_1 L_{c_1} \\ 0 \end{bmatrix} \tag{2.35} $$

Consequently the velocity of this point is:

$$ \mathbf{v}_{c_3} = \begin{bmatrix} -\mathrm{s}_3 L_{c_3} \dot{\theta}_3 \\ \mathrm{c}_3 L_{c_3} \dot{\theta}_3 \\ 0 \end{bmatrix} \tag{2.36} $$

And the angular velocity of the third link is:

$$ {}^{3}\boldsymbol{\omega}_{3} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_{3} \end{bmatrix} \tag{2.37} $$

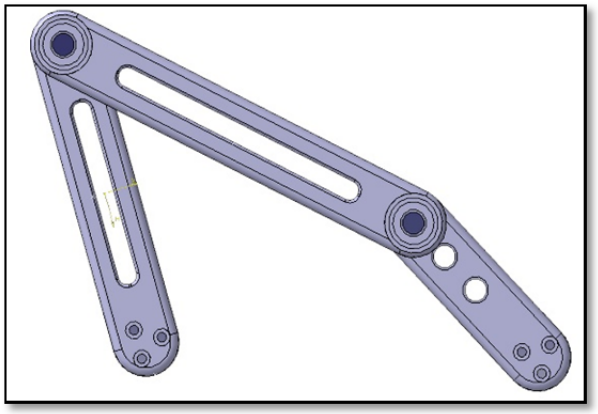


Figure 2.7 Closed chain configuration

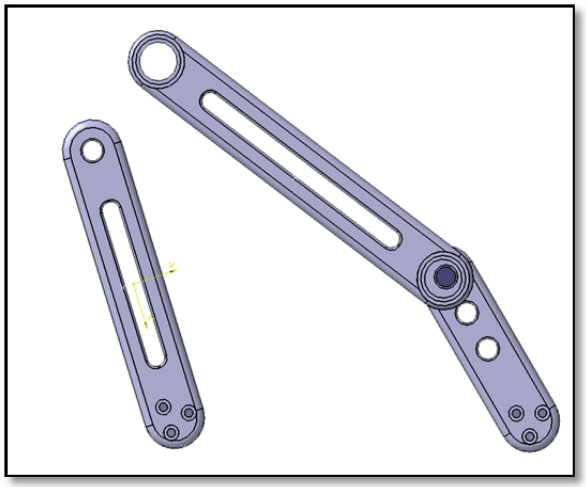


Figure 2.8 Open chain configuration

The kinetic energy of the first chain, based on equation(2.28), is given by:

$$ T_{1} = \frac{1}{2}\left[m_{1}\mathbf{v}_{c_1}^{T}\mathbf{v}_{c_1} + {}^{1}\boldsymbol{\omega}_{1}^{T}\,{}^{c}\mathbf{I}_{1}\,{}^{1}\boldsymbol{\omega}_{1} + m_{2}\mathbf{v}_{c_2}^{T}\mathbf{v}_{c_1} + {}^{2}\boldsymbol{\omega}_{2}^{T}\,{}^{c}\mathbf{I}_{2}\,{}^{2}\boldsymbol{\omega}_{2} \right] \tag{2.38} $$

And the kinetic energy of the second chain is:

$$T_2 = \frac{1}{2}[m_3 \mathbf{v}_{c_3}^T \mathbf{v}_{c_3} + {}^3\boldsymbol{\omega}_3^T \, {}^c\mathbf{I}_3 \, {}^3\boldsymbol{\omega}_3] \tag{2.39}$$

By expanding (2.38) we now have:

$$T_1 = \frac{1}{2}\left[\begin{array}{l} m_1 \begin{bmatrix} -\mathrm{s}_1 L_{c_1}\dot{\theta}_1 \\ \mathrm{c}_1 L_{c_1}\dot{\theta}_1 \\ 0 \end{bmatrix}^T \begin{bmatrix} -\mathrm{s}_1 L_{c_1}\dot{\theta}_1 \\ \mathrm{c}_1 L_{c_1}\dot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix}^T \begin{bmatrix} I_{x_1} & 0 & 0 \\ 0 & I_{y_1} & 0 \\ 0 & 0 & I_{z_1} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} + \\ m_2 \begin{bmatrix} -\mathrm{s}_1 d_1\dot{\theta}_1 - \mathrm{s}_{12} L_{c_2}(\dot{\theta}_1 + \dot{\theta}_2) \\ \mathrm{c}_1 d_1\dot{\theta}_1 - \mathrm{c}_{12} L_{c_2}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix}^T \begin{bmatrix} -\mathrm{s}_1 d_1\dot{\theta}_1 - \mathrm{s}_{12} L_{c_2}(\dot{\theta}_1 + \dot{\theta}_2) \\ \mathrm{c}_1 d_1\dot{\theta}_1 - \mathrm{c}_{12} L_{c_2}(\dot{\theta}_1 + \dot{\theta}_2) \\ 0 \end{bmatrix} + \\ \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix}^T \begin{bmatrix} I_{x_2} & 0 & 0 \\ 0 & I_{y_2} & 0 \\ 0 & 0 & I_{z_2} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix} \end{array}\right] \tag{2.40}$$

The kinetic energy for the third link is:

$$T_2 = \frac{1}{2}\left[m_3 \begin{bmatrix} -\mathrm{s}_3 L_{c_3}\dot{\theta}_3 \\ \mathrm{c}_3 L_{c_3}\dot{\theta}_3 \\ 0 \end{bmatrix}^T \begin{bmatrix} -\mathrm{s}_3 L_{c_3}\dot{\theta}_3 \\ \mathrm{c}_3 L_{c_3}\dot{\theta}_3 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix}^T \begin{bmatrix} I_{x_3} & 0 & 0 \\ 0 & I_{y_3} & 0 \\ 0 & 0 & I_{z_3} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_3 \end{bmatrix} \right] \tag{2.41}$$

After doing the multiplication in (2.40) and (2.41) the kinetic energy takes the following forms:

$$T_1 = \frac{1}{2}\left[m_1 L_{c_1}^2 \dot{\theta}_1^2 + I_{z_1}\dot{\theta}_1^2 + m_2 \begin{pmatrix} d_1^2\dot{\theta}_1^2 + L_{c_2}^2\left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 + \\ 2 d_1 L_{c_2}\, \mathrm{c}_2\, \dot{\theta}_1\left(\dot{\theta}_1 + \dot{\theta}_2\right) \end{pmatrix} + I_{z_2}\left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 \right] \tag{2.42}$$

$$T_2 = \frac{1}{2}\left[m_3 L_{c_3}^2 \dot{\theta}_3^2 + I_{z_3}\dot{\theta}_3^2 \right] \tag{2.43}$$

In order to find the equation of motion from kinetic energy we must do some derivations, as explained in (2.25):

$$\frac{\partial T_1^T}{\partial \dot{\theta}_{1,2}} = \begin{bmatrix} \frac{\partial T_1^T}{\partial \dot{\theta}_1} \\ \frac{\partial T_1^T}{\partial \dot{\theta}_2} \end{bmatrix} = \begin{bmatrix} m_1 L_{c_1}^2 \dot{\theta}_1 + I_{z_1} \dot{\theta}_1 + m_2 \begin{pmatrix} d_1^2 \dot{\theta}_1 + L_{c_2}^2 \left(\dot{\theta}_1 + \dot{\theta}_2\right) + \\ d_1 L_{c_2} \mathrm{c}_2 \left(2\dot{\theta}_1 + \dot{\theta}_2\right) \end{pmatrix} + I_{z_2} \left(\dot{\theta}_1 + \dot{\theta}_2\right) \\ m_2 \left(L_{c_2}^2 \left(\dot{\theta}_1 + \dot{\theta}_2\right) + d_1 L_{c_2} \dot{\theta}_1\right) + I_{z_2} \left(\dot{\theta}_1 + \dot{\theta}_2\right) \end{bmatrix} \tag{2.44}$$

$$\frac{\partial T_2^T}{\partial \dot{\theta}_3} = \left[m_3 L_{c_3}^2 \dot{\theta}_3 + I_{z_3} \dot{\theta}_3 \right]$$

Note that by $\frac{\partial T^T}{\partial \dot{\theta}_{1,2}}$ we mean that T^T should be partially derived with respect to $\dot{\theta}_1$ and $\dot{\theta}_2$ respectively.

The time derivative of equation (2.44) is:

$$\frac{d}{dt}\frac{\partial T_1^T}{\partial \dot{\theta}_{1,2}} = \begin{bmatrix} m_1 L_{c_1}^2 \ddot{\theta}_1 + I_{z_1} \ddot{\theta}_1 + m_2 \begin{pmatrix} d_1^2 \ddot{\theta}_1 + L_{c_2}^2 \left(\ddot{\theta}_1 + \ddot{\theta}_2\right) - \\ d_1 L_{c_2} \mathrm{s}_2 \dot{\theta}_2 \left(2\dot{\theta}_1 + \dot{\theta}_2\right) + \\ d_1 L_{c_2} \mathrm{c}_2 \dot{\theta}_2 \left(2\ddot{\theta}_1 + \ddot{\theta}_2\right) \end{pmatrix} + I_{z_2} \left(\ddot{\theta}_1 + \ddot{\theta}_2\right) \\ m_2 \left(L_{c_2}^2 \left(\ddot{\theta}_1 + \ddot{\theta}_2\right) - d_1 L_{c_2} \mathrm{s}_2 \dot{\theta}_1 \dot{\theta}_2 + d_1 L_{c_2} \mathrm{c}_2 \ddot{\theta}_1\right) + I_{z_2} \left(\ddot{\theta}_1 + \ddot{\theta}_2\right) \end{bmatrix} \tag{2.45}$$

$$\frac{d}{dt}\frac{\partial T_2^T}{\partial \dot{\theta}_3} = \left[m_3 L_{c_3}^2 \ddot{\theta}_3 + I_{z_3} \ddot{\theta}_3 \right]$$

The partial derivative of kinetic energy based on θ is:

$$\frac{\partial T_1^T}{\partial \theta_{1,2}} = \begin{bmatrix} \frac{dT_1^T}{d\theta_1} \\ \frac{dT_1^T}{d\theta_2} \end{bmatrix} = \begin{bmatrix} 0 \\ -m_2 d_1 L_{c_2} \mathrm{s}_2 \dot{\theta}_1 \left(\dot{\theta}_1 + \dot{\theta}_2\right) \end{bmatrix} \tag{2.46}$$

$$\frac{\partial T_2^T}{\partial \theta_3} = [0] \tag{2.47}$$

$$\frac{d}{dt}\frac{\partial T^T}{\partial \dot{\boldsymbol{\theta}}} - \frac{\partial T^T}{\partial \boldsymbol{\theta}} - \mathbf{Q} = 0 \tag{2.48}$$

And the vector of generalized forces is:

$$\mathbf{Q} = -\frac{dV}{d\boldsymbol{\theta}} + \boldsymbol{\tau} + \mathbf{Q}_{nc}$$

where $\boldsymbol{\tau}$ is the vector of the applied torque to the joints, $\mathbf{Q}_{nc}$ is the vector of the non-conservative forces like friction or external forces which in this case will be ignored, and V is the potential energy. As mentioned before, the differentiation in potential energy in this robot is zero since there is no elevation differentiation in robot arms.

Thus,

$$\mathbf{Q} = \begin{bmatrix} \tau \\ 0 \\ 0 \end{bmatrix}$$

By taking the derivations and doing the simplification, the following equation is achieved:

$$\begin{bmatrix} m_1 L_{c_1}^2 \ddot{\theta}_1 + I_{z_1}\ddot{\theta}_1 + m_2 \begin{pmatrix} d_1^2\ddot{\theta}_1 + L_{c_2}^2\left(\ddot{\theta}_1+\ddot{\theta}_2\right) - \\ d_1 L_{c_2}\,\mathrm{s}_2\,\dot{\theta}_2\left(2\dot{\theta}_1+\dot{\theta}_2\right) + \\ d_1 L_{c_2}\,\mathrm{c}_2\,\dot{\theta}_2\left(2\ddot{\theta}_1+\ddot{\theta}_2\right) \end{pmatrix} + I_{z_2}\left(\ddot{\theta}_1+\ddot{\theta}_2\right) \\ m_2\left(L_{c_2}^2\left(\ddot{\theta}_1+\ddot{\theta}_2\right) - d_1 L_{c_2}\,\mathrm{s}_2\,\dot{\theta}_1 + d_1 L_{c_2}\,\mathrm{c}_2\,\ddot{\theta}_1\right) + I_{z_2}\left(\ddot{\theta}_1+\ddot{\theta}_2\right) \end{bmatrix} - \mathbf{Q}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.49}$$

$$\left[m_3 L_{c_3}^2\ddot{\theta}_3 + I_{z_3}\ddot{\theta}_3\right] - \mathbf{Q}_2 = [0]$$

where $\mathbf{Q}_1 = \begin{bmatrix} \tau \\ 0 \end{bmatrix}$ and $\mathbf{Q}_2 = [0]$.

This equation can be written in the following form:

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{F}\left(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\right) = \boldsymbol{\beta}\tau \tag{2.50}$$

where $\mathbf{M}(\boldsymbol{\theta}) \in \mathbb{R}^{3\times 3}$ is the mass matrix, and $\mathbf{F}\left(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\right) \in \mathbb{R}^{3}$ is the matrix of the Coriolis force.

The mass matrix is:

$$\mathbf{M}=\begin{bmatrix} m_1\left(L_{c_1}\right)^2+Ic_1+m_2\begin{pmatrix} d_1^2+(L_{c_2})^2+ \\ 2d_1(L_{c_2})c_2 \end{pmatrix}+Ic_2 & m_2\begin{pmatrix} (L_{c_2})^2+ \\ d_1(L_{c_2})c_2 \end{pmatrix}+I_{z_2} & 0 \\ m_2((L_{c_2})^2+d_1(L_{c_2})c_2)+I_{z_2} & m_2(L_{c_1})^2+I_{z_2} & 0 \\ 0 & 0 & I_{z_3}+m_3(L_{c_3})^2 \end{bmatrix} \quad (2.51)$$

The Coriolis force matrix is:

$$\mathbf{F}=\begin{bmatrix} -m_2d_1L_{c_2}s_2\dot{\theta}_2\left(2\dot{\theta}_1+\dot{\theta}_2\right) \\ m_2d_1L_{c_2}s_2\dot{\theta}_1^2 \\ 0 \end{bmatrix} \quad (2.52)$$

And $\boldsymbol{\beta}$ is the coupling matrix:

$$\boldsymbol{\beta}=\begin{bmatrix}1 & 0 & 0\end{bmatrix}^{\mathrm{T}} \quad (2.53)$$

Up to now, we have not considered the constraints. The second step is to close the mechanical loop by taking into account the kinematic constraint given by equation (2.1). The constraint in x direction is g_x:

$$g_x = d_1\cos(\theta_1)+d_2\cos(\theta_1+\theta_2)-d_3\cos(\theta_3)-l=0 \quad (2.54)$$

And the constraint in y direction is g_y:

$$g_y = d_1\sin(\theta_1)+d_2\sin(\theta_1+\theta_2)-\mathrm{d}_3\sin(\theta_3)=0 \quad (2.55)$$

Thus, **g** is the vector of constraints:

$$\mathbf{g}=\begin{bmatrix} g_x \\ g_y \end{bmatrix} \quad (2.56)$$

By differentiating this constraint relative to time, we can write:

$$\mathbf{J_g}\dot{\boldsymbol{\theta}} = 0 \tag{2.57}$$

where the Jacobian matrix of the constraint **g** is given by:

$$\mathbf{J_g} = \begin{bmatrix} \frac{\partial g_x}{\partial \theta_1} & \frac{\partial g_x}{\partial \theta_2} & \frac{\partial g_x}{\partial \theta_3} \\ \frac{\partial g_y}{\partial \theta_1} & \frac{\partial g_y}{\partial \theta_2} & \frac{\partial g_y}{\partial \theta_3} \end{bmatrix} \tag{2.58}$$

After doing all the partial derivatives in(2.58), the following Jacobian is derived:

$$\mathbf{J_g} = \begin{bmatrix} -d_2\,\mathrm{s}_{12} - d_1\,\mathrm{s}_1 & -d_2\,\mathrm{s}_{12} & d_3\,\mathrm{s}_3 \\ d_2\,\mathrm{c}_{12} + d_1\,\mathrm{c}_1 & d_2\,\mathrm{c}_{12} & -d_3\,\mathrm{c}_3 \end{bmatrix} \tag{2.59}$$

where s_{12} is $\sin(\theta_1+\theta_2)$, c_{12} is $\cos(\theta_1+\theta_2)$, s_3 is $\sin(\theta_3)$, and c_3 is $\cos(\theta_3)$. It is possible to separate the Jacobian into two parts:

$$\begin{aligned} \mathbf{J_{g_1}} &= \begin{bmatrix} -d_2\,\mathrm{s}_{12} - d_1\,\mathrm{s}_1 \\ d_2\,\mathrm{c}_{12} + d_1\,\mathrm{c}_1 \end{bmatrix} \\ \mathbf{J_{g_2}} &= \begin{bmatrix} -d_2\,\mathrm{s}_{12} & d_3\,\mathrm{s}_3 \\ d_2\,\mathrm{c}_{12} & -d_3\,\mathrm{c}_3 \end{bmatrix} \end{aligned} \tag{2.60}$$

where $\mathbf{J_{g_1}}$ is associated with coordinate 1, and $\mathbf{J_{g_2}}$ is associated with the two other coordinates, namely 2 and 3. According to (Murray et al., 1994) the constraint can be incorporated into the model by using the Lagrange-d'Alembert formulation. Equation (2.50) can be reduced to the following form:

$$\bar{\mathbf{M}}_1(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}}_1 + \bar{\mathbf{F}}_1(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \bar{\boldsymbol{\beta}}_1 \tau \tag{2.61}$$

where

$$\bar{\mathbf{M}}_1(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{I} & -\mathbf{J}_{g_1}^T \mathbf{J}_{g_2}^{-T} \end{bmatrix} \mathbf{M}(\boldsymbol{\theta}) \begin{bmatrix} \mathbf{I} \\ -\mathbf{J}_{g_2}^{-1} \mathbf{J}_{g_1} \end{bmatrix} \tag{2.62}$$

$$\bar{\mathbf{F}}_1(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \begin{bmatrix} \mathbf{I} & -\mathbf{J}_{g_1}^T \mathbf{J}_{g_2}^{-T} \end{bmatrix} \left(\mathbf{F}_1(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta}) + \mathbf{M}(\boldsymbol{\theta}) \begin{bmatrix} 0 \\ -\frac{d}{dt}(-\mathbf{J}_{g_2}^{-1} \mathbf{J}_{g_1}) \end{bmatrix} \boldsymbol{\theta} \right) \tag{2.63}$$

$$\bar{\boldsymbol{\beta}}_1 = \begin{bmatrix} \mathbf{I} & -\mathbf{J}_{g_1}^T \mathbf{J}_{g_2}^{-T} \end{bmatrix} \boldsymbol{\beta} \tag{2.64}$$

Equation (2.61) is the final dynamic equation.

2.4 Trajectory

To validate the controllers that will be implemented on the four-bar prototype, we choose a seventh degree polynomial trajectory:

$$\theta_1 = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 + a_6 t^6 + a_7 t^7 \tag{2.65}$$

$$\dot{\theta}_1 = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4 + 6a_6 t^5 + 7a_7 t^6 \tag{2.66}$$

$$\ddot{\theta}_1 = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3 + 30a_6 t^4 + 42a_7 t^5 \tag{2.67}$$

$$\dddot{\theta}_1 = 6a_3 + 24a_4 t + 60a_5 t^2 + 120a_6 t^3 + 210a_7 t^4 \tag{2.68}$$

To ensure the smooth start and stop of the trajectory, the velocity, acceleration, and derivative of the acceleration (jerk) should be zero in the initial and final positions. By solving (2.65), (2.66), (2.67) and (2.68) together for the initial and final conditions, and knowing that the position at $t_0 = 0$ is θ_0, and at $t_f = \frac{\theta_f - \theta_0}{\dot{\theta}}$ is θ_f, the parameters will be determined with the knowledge that θ_f is the destination point, and $\dot{\theta}$ is the desired velocity with which we want the link to go from the initial position to the final one.

In order to respect these conditions, the parameters of the trajectory can be chosen as:

$$a_0 = \theta_0 \tag{2.69}$$

$$a_1 = a_2 = a_3 = 0 \tag{2.70}$$

$$a_4 = \frac{-35(\theta_0 {t_f}^3 - \theta_f {t_f}^3)}{tf^7} \tag{2.71}$$

$$a_5 = \frac{1008(\theta_0 {t_f}^5 - 1008\theta_f {t_f}^5)}{12tf^{10}} \tag{2.72}$$

$$a_6 = \frac{-70(\theta_0 - \theta_f)}{tf^6} \tag{2.73}$$

$$a_7 = \frac{20(\theta_0 - \theta_f)}{tf^7} \tag{2.74}$$

For our validation, the desired starting point of the first link is zero radian, and it has to reach π radians in 0.5 seconds . The corresponding trajectory is shown in Figure 2-9.

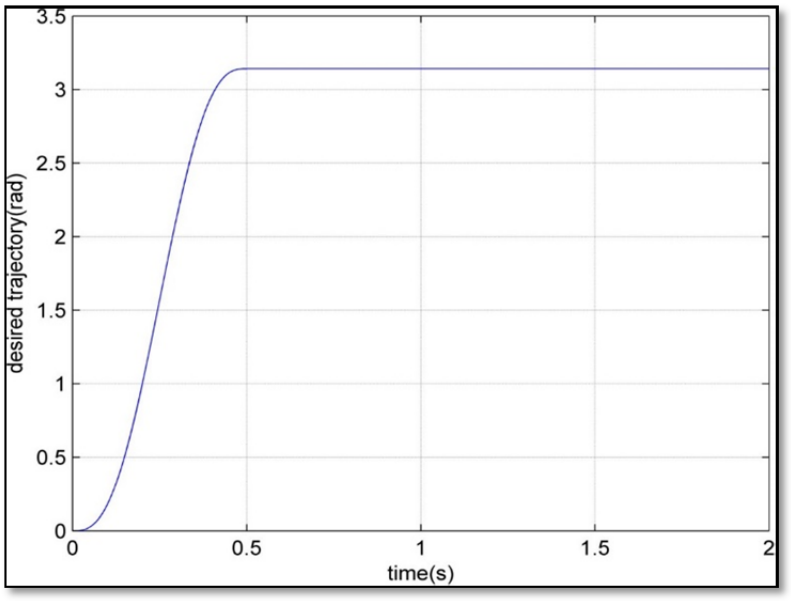


Figure 2.9 Desired trajectory for the angle of the first link of the robot

2.5 Command law

In this section, we use the mass matrix and Coriolis-force matrix determined in the previous section to present the computed torque that is used to control the angle of the first link of the robot.

2.5.1 Computed torque

The knowledge of the dynamic model of the robot enables the design of precise and energy-efficient controllers such as computed torque. This model base controller allows the robot to

follow the desired trajectory very quickly and with a small error. The more precise the model, the smaller the error will be. Another name for "computed torque" is "linearizing controller," since if we consider the dynamic model of the robot:

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}}+\mathbf{F}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})=\boldsymbol{\tau} \tag{2.75}$$

Here $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$, and $\ddot{\boldsymbol{\theta}}$ denotes the real angular position, real angular velocity, and real angular acceleration of the robot joints, so the linearizing control law will be:

$$\boldsymbol{\tau}=\mathbf{M}\left(\boldsymbol{\theta}\right)\mathbf{u}+\mathbf{F}\left(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\right) \tag{2.76}$$

All the terms cancel out and the linear system is now:

$$\ddot{\boldsymbol{\theta}}=\mathbf{u} \tag{2.77}$$

$$\mathbf{u}=\ddot{\boldsymbol{\theta}}_d+\mathbf{K}_d(\dot{\boldsymbol{\theta}}_d-\dot{\boldsymbol{\theta}})+\mathbf{K}_p(\boldsymbol{\theta}_d-\boldsymbol{\theta}) \tag{2.78}$$

where $\mathbf{K}_d$ and $\mathbf{K}_p$ are the matrices of derivative and proportional gains.

If the matrices of $\mathbf{K}_d$ and $\mathbf{K}_p$ are diagonal, then the dynamic of error is:

$$(\ddot{\boldsymbol{\theta}}_d-\ddot{\boldsymbol{\theta}})+\mathbf{K}_d(\dot{\boldsymbol{\theta}}_d-\dot{\boldsymbol{\theta}})+\mathbf{K}_p(\boldsymbol{\theta}_d-\boldsymbol{\theta})=\mathbf{0} \tag{2.79}$$

For all the actuated links we have:

$$\ddot{e}_i+K_{d_i}\dot{e}_i+K_{p_i}e_i=0 \tag{2.80}$$

where $i=[1,\mathrm{n}]$.

The gains can be calculated in order to impose the poles of the characteristic equation. For example, the following gains will generate a zero overshoot with response time of T_r:

$$\mathbf{K}_p=\lambda^2\mathbf{I} \tag{2.81}$$

$$\mathbf{K}_d=2\lambda\mathbf{I} \tag{2.82}$$

where λ is a pole of multiplicity two given by:

$$\lambda=\frac{4.73}{T_r} \tag{2.83}$$

The linearization controller of (2.74) can also be combined with a PID:

$$\mathbf{u}=\ddot{\boldsymbol{\theta}}_d+\mathbf{K}_d(\dot{\boldsymbol{\theta}}_d-\dot{\boldsymbol{\theta}})+\mathbf{K}_p(\boldsymbol{\theta}_d-\boldsymbol{\theta})+\mathbf{K}_i\int(\boldsymbol{\theta}_d-\boldsymbol{\theta})dt \tag{2.84}$$

where $\mathbf{K}_i$ is the matrix of integral gain.

If the matrices of $\mathbf{K}_d$, $\mathbf{K}_p$ and $\mathbf{K}_i$ are diagonal, then the dynamic of error is:

$$(\ddot{\boldsymbol{\theta}}_d - \ddot{\boldsymbol{\theta}}) + \mathbf{K}_d(\dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}}) + \mathbf{K}_p(\boldsymbol{\theta}_d - \boldsymbol{\theta}) + \mathbf{K}_i \int (\boldsymbol{\theta}_d - \boldsymbol{\theta}) dt = 0 \tag{2.85}$$

Again the gains can be calculated in order to impose the poles of characteristic equation. For example the following gains will generate a zero overshoot with response time of T_r:

$$\mathbf{K}_p = 3\lambda^2 \mathbf{I} \tag{2.86}$$

$$\mathbf{K}_i = \lambda^3 \mathbf{I} \tag{2.87}$$

$$\mathbf{K}_d = 3\lambda \mathbf{I} \tag{2.88}$$

where λ is a pole of multiplicity three given by:

$$\lambda = \frac{6.27}{T_r} \tag{2.89}$$

The PD computed torque and PID computed torque are illustrated in Figure 2-10.

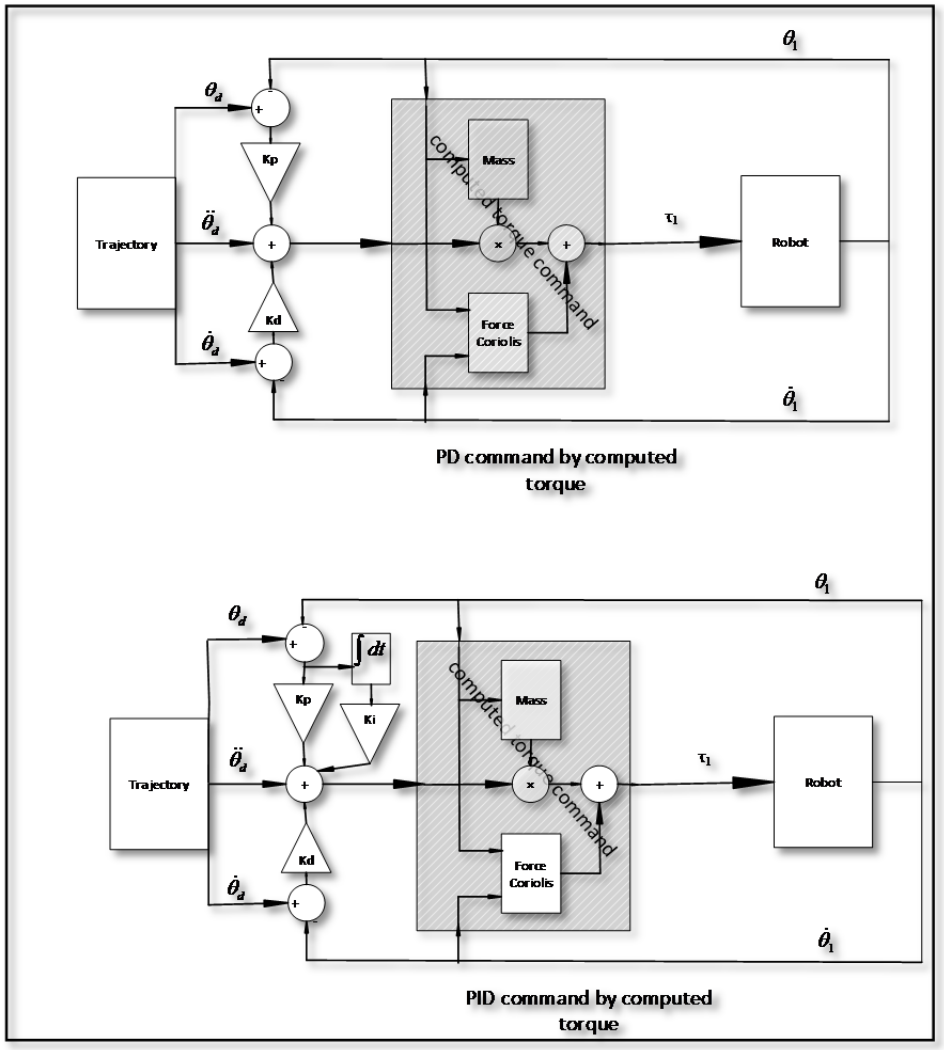


Figure 2.10 PD computed torque (top)
and PID computed torque (bottom)

2.6 Validation of the model

For validation, the robot is modeled manually in SimMechanics according to the table of mechanical properties shown in Table 2.1. For this purpose we used the inverse of the dynamic model in SimMechanics as the computed torque controller. The next step is to compare, for the same trajectory, the torques generated by the SimMechanics method with those generated by the analytical method. If the torques are similar then the results validate

our analytical model. The comparison of torques for the SimMechanics method and the analytical method can be found in Figures 2-11 and 2-12.

Table 2.1 Mechanical parameters of the four-bar robot

Parameters	Variables	Values	Units
First link length	d_1	68.58	mm
Second link length	d_2	149.225	mm
Third link length	d_3	114.3	mm
Distance between two joints attached to ground	l	149.225	mm
First link center of gravity[1]	L_{c_1}	29.748	mm
Second link center of gravity	L_{c_2}	$149.225/2 = 74.6125$	mm
Third link center of gravity[1]	L_{c_3}	53.54	mm
First link mass	m_1	0.088	kg
Second link mass	m_2	0.166	kg
Third link mass	m_3	0.138	kg
First link moment of inertia[2]	${}^{c}I_1$	6.449×10^{-5}	$kg.m^2$
Second link moment of inertia[2]	${}^{c}I_2$	4.63×10^{-4}	$kg.m^2$
Third link moment of inertia[2]	${}^{c}I_3$	2.265×10^{-4}	$kg.m^2$
Pulleys speed reduction ratio	n	1 : 4	-

[1] The center of gravity is measured from the joint attached to the ground.
[2] The moment of inertia is about the z-axis parallel to the gravity direction. It is measured about the center of gravity.

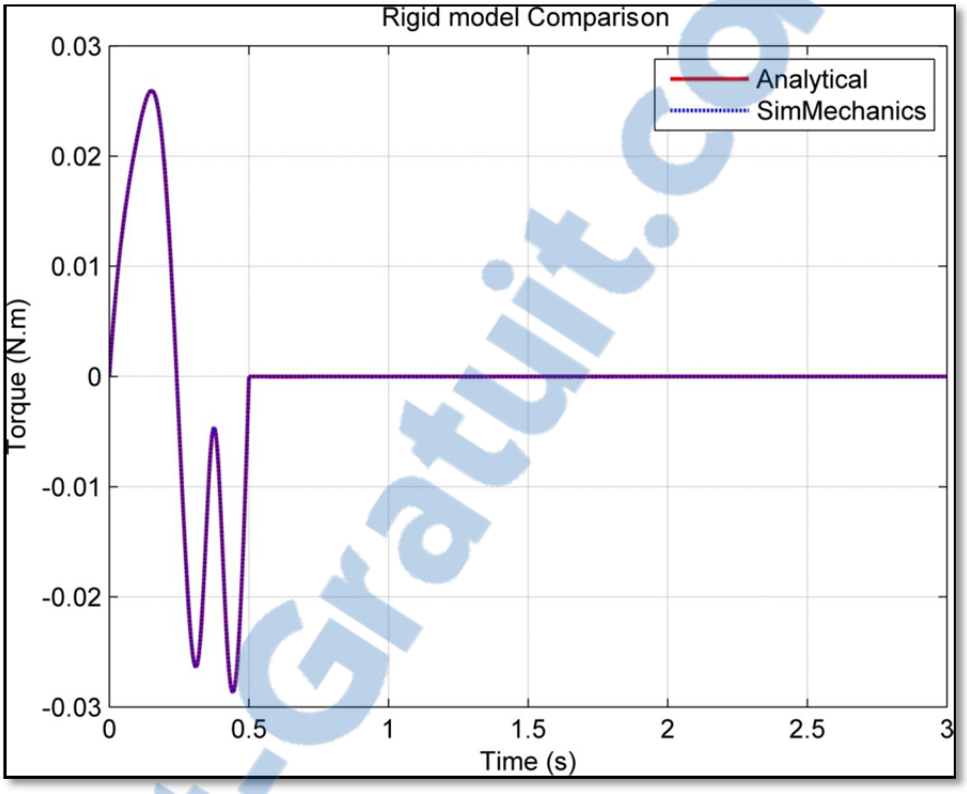


Figure 2.11 Comparison of analytical and SimMechanics methods for the rigid model

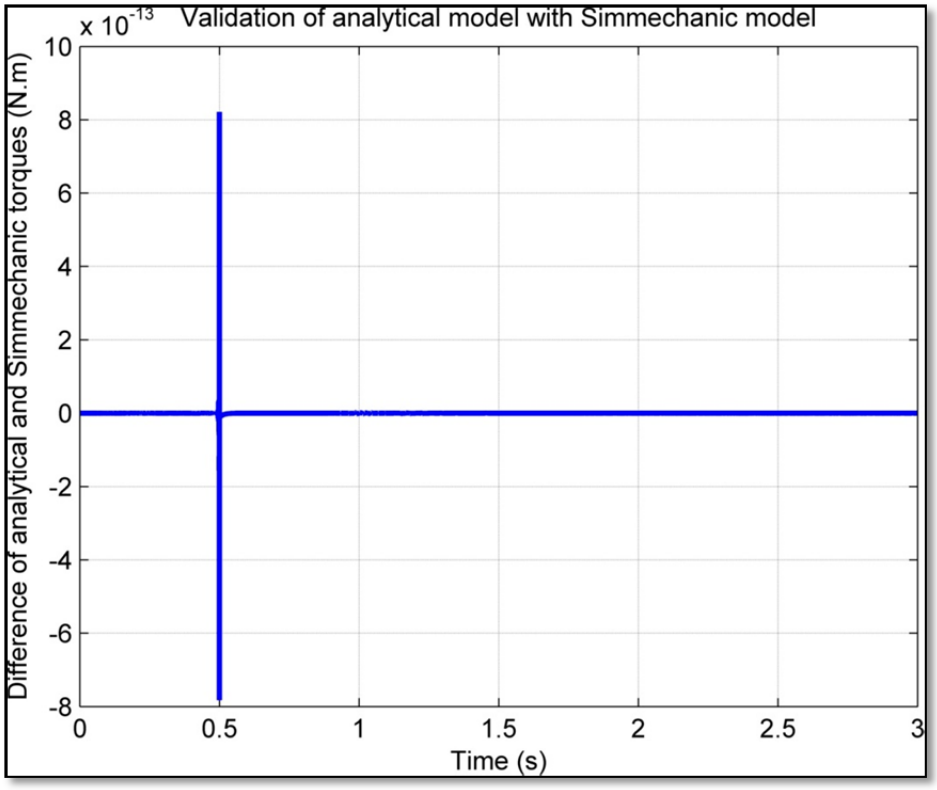


Figure 2.12 Comparison of the analytical and SimMechanics model by the difference of torques

Figure 2.11 shows that the two models are very similar. Figure 2.12 represents the difference between the torque of the analytical model and the torque of the SimMechanics model. The order of difference shows that the two models act very similarly for the same trajectory. Consequently we can conclude that our analytical model is accurate.

CHAPTER 3

VIRTUAL PROTOTYPING

The aim of virtual prototyping is to save time in modeling the dynamics of a system without involving the complicated equations. For this purpose we need to use the related software.

3.1 Modeling the robot in CATIA V6

To validate the virtual prototyping we modeled the four-bar mechanism in CATIA V6 with all of the detailed parts. The proper materials were selected for the links and other parts. This allows the software to determine the mechanical properties, such as mass and moment of inertia, for each part. By modeling the parts geometrically, the software is also able to determine the center of gravity and inertia. So far the mechanical properties of the system are known by the software. We need only to impose the mechanical constraints on the system. The mechanical constraints are applied to the parts to form the final product, as shown in Figure 2.1. The robot is made of five subassemblies: Base, housing, link 1, link 2 and link 3. Each subassembly contains different parts.

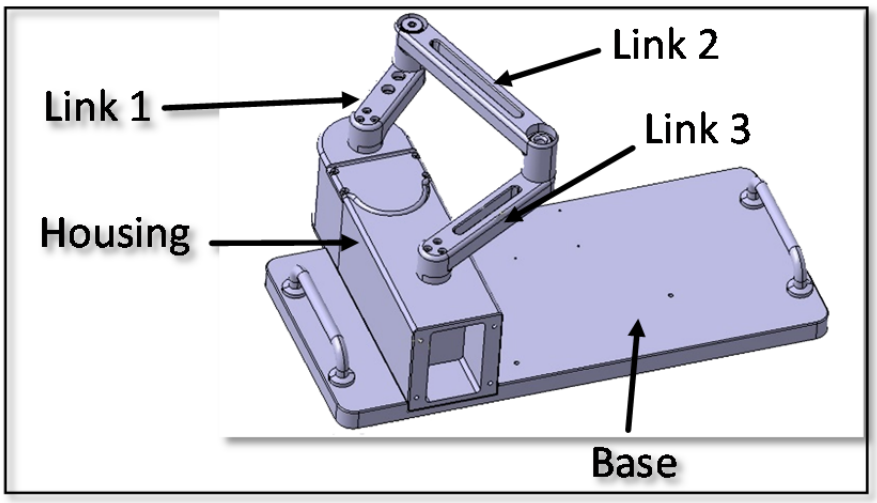


Figure 3.1 Robot composed of five subassemblies

The macro, which was developed by members of our research laboratory (The CORO Lab) and is explained in the next section, considers only the constraints between subassemblies. It does not matter which kind of constraint is used inside the subassemblies between parts, because the macro treats each subassembly (with its distinct parts) as a rigid body. We will model the robot so that the collection of all moving objects that are immobile relative to each other is considered a subassembly. For example, in Figure 3.2 all the parts are moving, but they are immobile relative to each other, and thus they form one subassembly. To simplify the model we divided the ground into two subassemblies, base and housing, which are connected to each other with a fixed constraint. The complete list of subassemblies is: base, housing, first link, second link and third link. The base and housing are connected to each other with a fixed joint (welding joint) and the other subassemblies are connected by revolute joints.

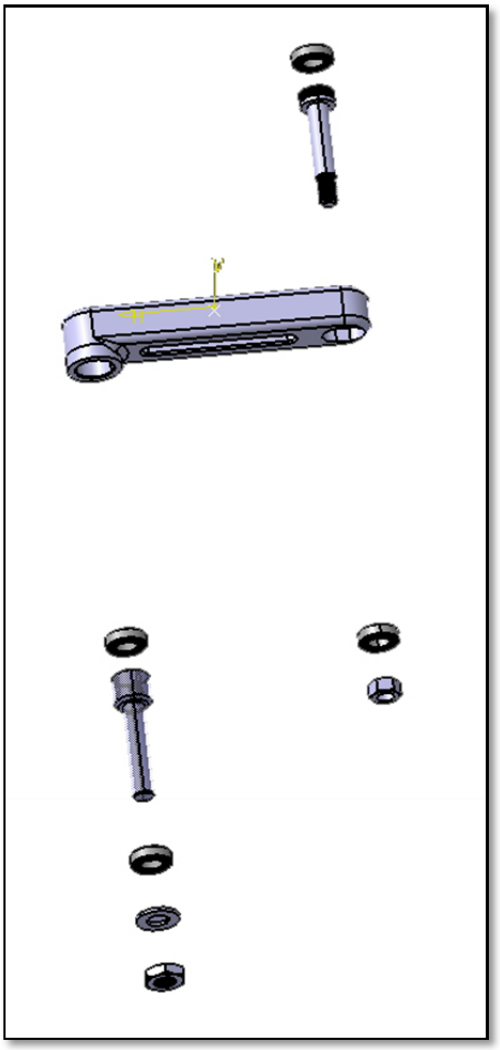


Figure 3.2 Exploded view
of parts for the third link subassembly

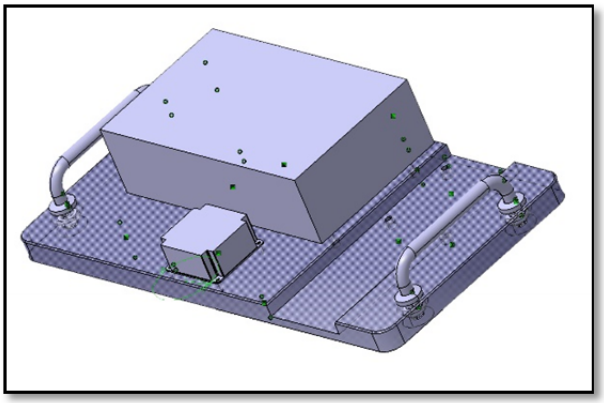


Figure 3.3 Base subassembly
as one rigid part

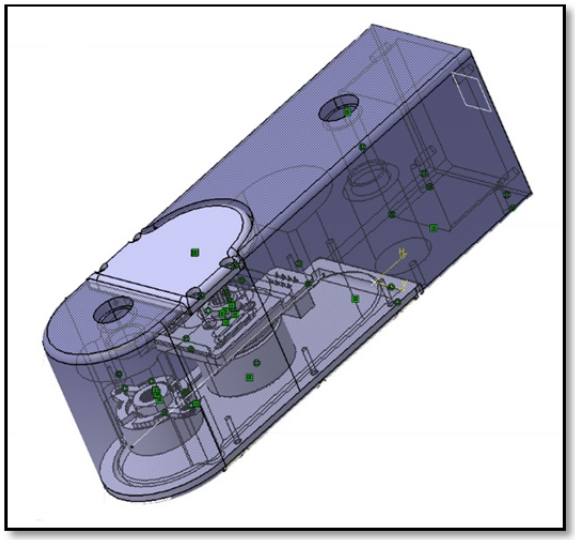


Figure 3.4 Housing subassembly
as one rigid part

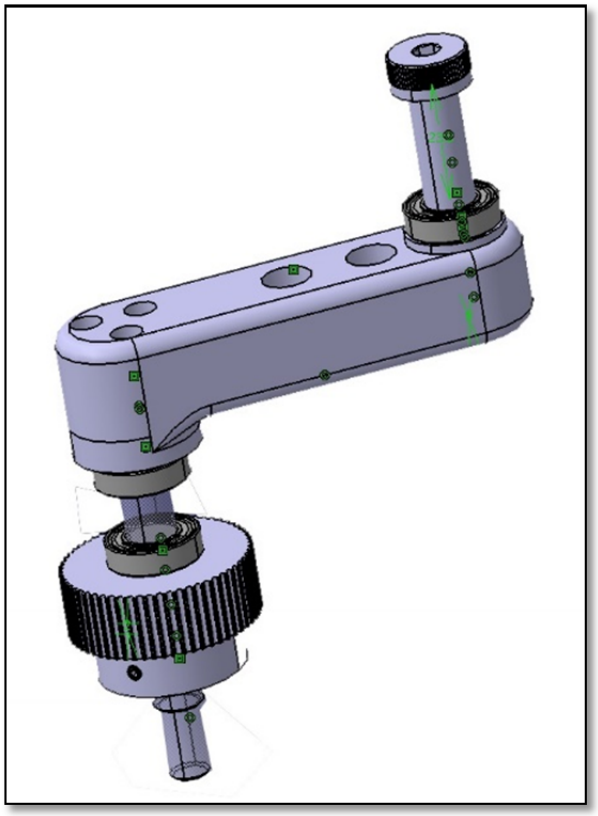


Figure 3.5 Link one subassembly
as one rigid part

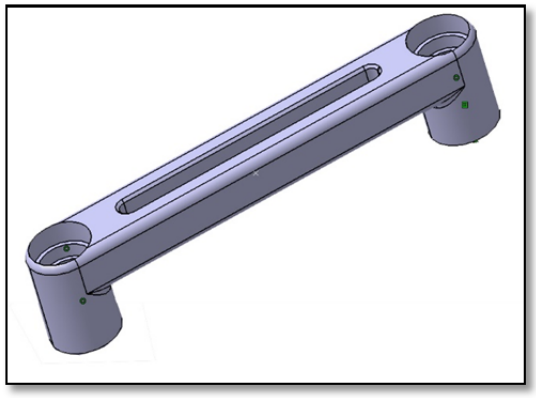


Figure 3.6 Link two subassembly
as one rigid part

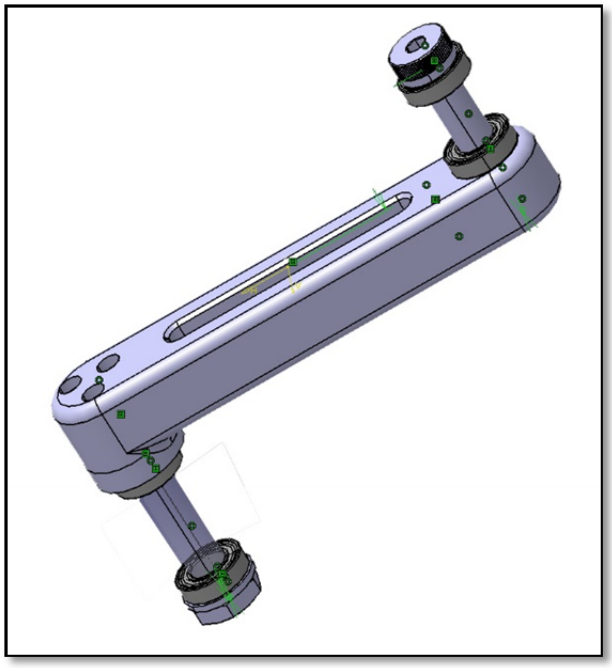


Figure 3.7 Link three subassembly as one rigid part

The five subassemblies are depicted, respectively, from Figure 3.3 to Figure 3.7.

In order to apply the constraints to the robot, after having modeled all the individual parts in 'Start\ Mechanical Design\ Part Design' one should then follow 'Start\ Mechanical Design\ Assembly Design' to import the parts to make each subassembly.

The appropriate constraints are imposed on the parts to locate the parts in the desired order. These assemblies can be seen in Figure 3.4, Figure 3.5, Figure 3.6, and Figure 3.7. Then the final product is made by importing the base and housing into a blank assembly design environment. The two subassemblies are attached with a fixed constraint. Then link one is imported. The revolute constraint is imposed on the housing assembly and on the end of link one with the gear. The second link is imported. The revolute joint is imposed on the other ends of the first and second link. Similarly the third link is imported and the revolute joint is applied to the second and third link. Finally the other end of the third link is connected to the housing subassembly with a revolute joint. Now we have the complete CAD model of the

robot with all its mechanical and geometrical properties. We just need a program to access these properties.

3.2 Develop a macro

Another member of our team has developed a macro to extract the geometric and dynamic information from the CAD model. To use this macro, the subassemblies should be connected together with either fixed, revolute, or prismatic joints. For the present robot we used only fixed and revolute joints to connect the subassemblies together.

To begin using the macro, the CAD file should be opened in CATIA V6. Then from menu 'Tools/ Macro/ Macros,' as shown in Figure 3.8, select 'ExtractDynamics' as shown in Figure 3.9 in the Macros window. Click on the edit button. In the new window, click on the play icon from the top menu and follow the instructions.

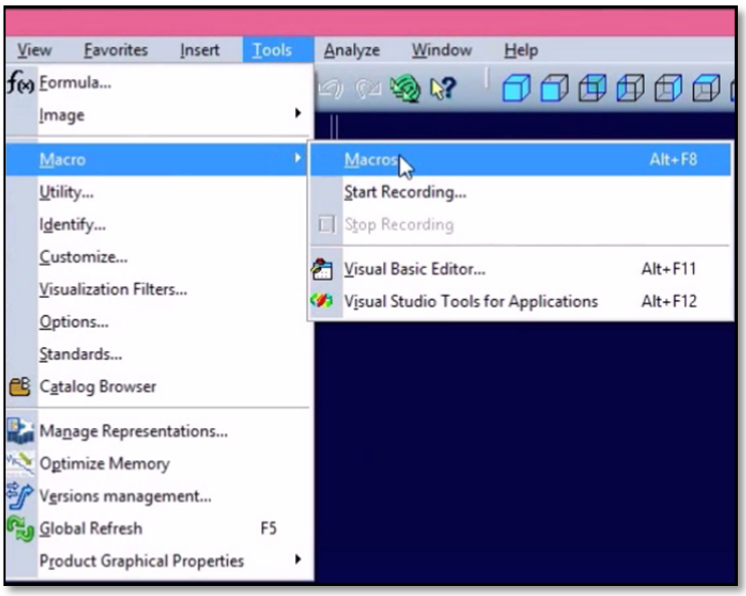


Figure 3.8 Tools/Macro/Macros

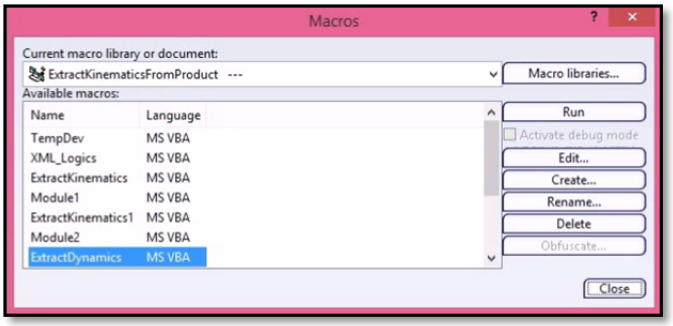


Figure 3.9 Macros window

The macro will detect the five subassemblies and their constraints, as shown in Figure 3.10.

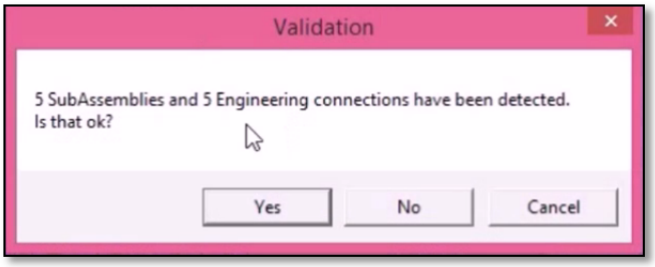


Figure 3.10 Validation message

We then select the features that define each constraint. For example, the revolute joint between links three and two is on a bolt on the third link and a hole on the second link. So we should select the hole and the bolt as the features for the connection. Figure 3.11 shows the procedure of selecting the feature for the constraint between link two and link three.

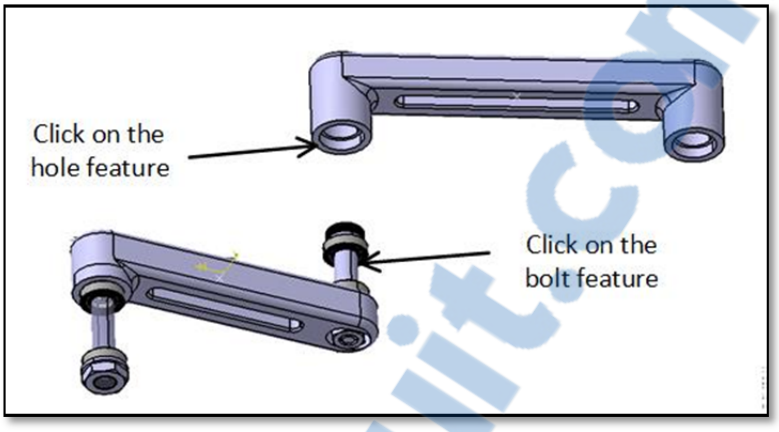


Figure 3.11 Feature selection for constraints

This procedure should be done for all of the connections. At the end, the macro will generate the XML file that should be saved on the computer. This file includes all geometric properties for each subassembly, their respective centers of gravity, and the inertial parameters, such as mass and moment of inertia. The file connects all of them with the proper constraints.

3.3 Adding sensors and actuators in SimMechanics

The XML generated by the CATIA MACRO file should be copied in the Matlab current folder. Using the command 'mech_import filename.xml' in the Matlab workspace will generate a model for SimMechanics. See Figure 3-11 for an example.

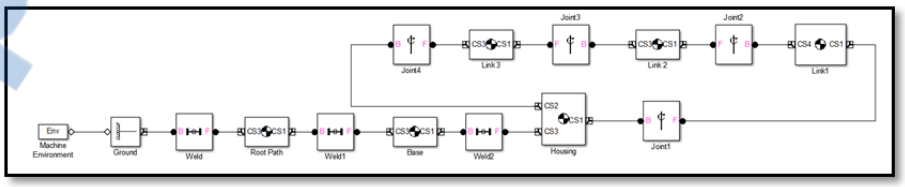


Figure 3.12 SimMechanics model of robot

Since CATIA does not involve the movement of the mechanism, the generated model does not have an actuator and sensor. However, we can manually add the sensor and actuator from

the Simulink library in ‘Simscape\ SimMechanics\ SimMechanics First Generation\ Sensors & Actuators.’ For this purpose drag and drop the ‘Joint Sensor’ and ‘Joint Actuator’ to the SimMechanics file that contains the modeled robot.

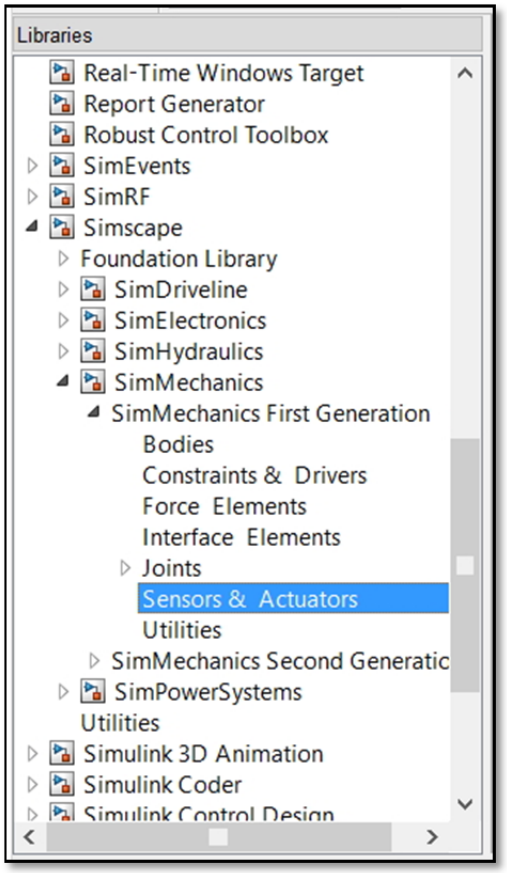


Figure 3.13 Simulink Library

Double-click on the joint where there should be the motor and encoder. In our case, it is the joint connecting the first link to the housing. In the section ‘Connection parameters,’ change the parameter ‘Number of sensor / actuator ports’ to two, because we need to connect one actuator and one sensor to this joint.

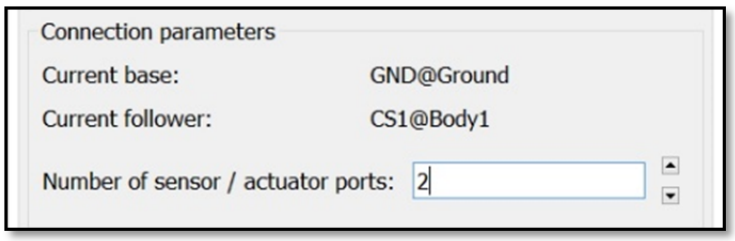


Figure 3.14 Adding the number of actuators and sensors

Then, connect the sensor and actuator that have been added from the library. Double-click on the actuator, and in the 'Actuation' part set the parameter 'Actuate with' to 'Generalized Forces,' and set the 'Applied torque units' as desired (in our case to 'N*m').

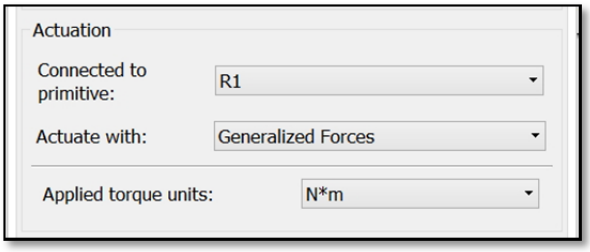


Figure 3.15 Selecting the actuation mode and its unit

Then double-click on the joint sensor. Under 'Measurements,' check the box to select 'Angular Velocity' and set the 'Units' to 'rad/s.'

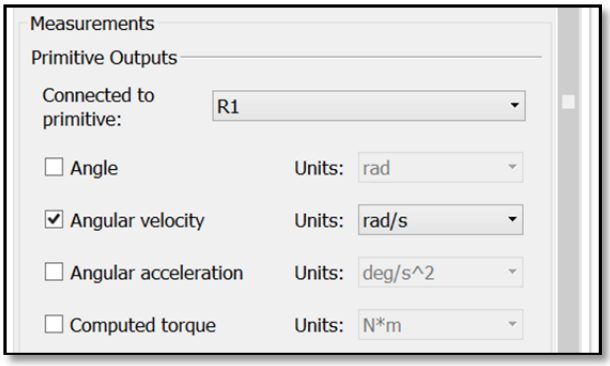


Figure 3.16 Selecting measurement parameters and the units

If angular velocity has been chosen, we could add an integral block from the Simulink library to calculate the position (in our case, the position of the first link). In that case, discontinuity will be avoided at each 2π rad position.

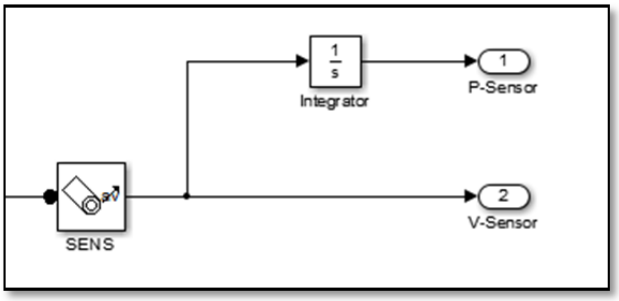


Figure 3.17 Position and velocity sensors

Finally the robot model can be completed with one input for torque and two outputs for position and velocity.

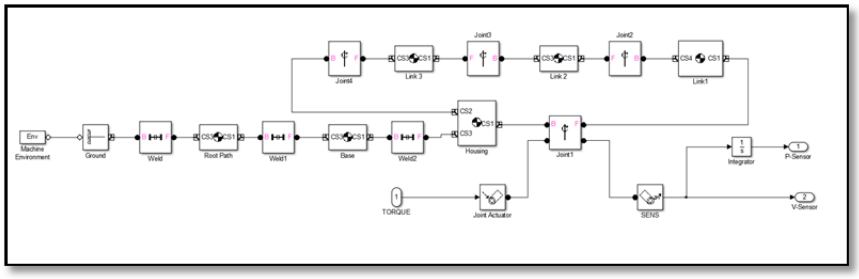


Figure 3.18 Complete robot model

CHAPTER 4

ITERATIVE LEARNING CONTROL

4.1 ILC algorithms

In this chapter, we will explore different approaches to iterative learning control (ILC), and then two types of ILC will be used to validate the macro that was presented in Chapter Three. As discussed previously, in repetitive processes with identical conditions in each cycle, the error remains unchanged throughout the process. One way to decrease the error over time is with ILC. For the cycle-to-cycle processes which each cycle is exactly the same as the next one, it is possible to benefit from this method of control.

The term "iterative learning control" was first introduced by Arimoto (Moore, 1993). ILC is a way to improve the transient response performance of systems that operate repetitively over a fixed time interval (Moore, 1993). The idea of iterative learning control is to modify the input of the system by training the system in such a way as to converge y_k (output) as closely as possible to y_d (desired periodic trajectory) – put simply, to minimize the error (Moore, 1993). Generally speaking, there are two types of ILC updating law: P-type and PD-type (Yang Quan and Moore, 2002). We will study P-type and PD-type ILC as applied to the parallel robot described in previous chapters.

4.1.1 P-type ILC

The most basic ILC algorithm is the P-type ILC. The following formula is used to obtain the input $u_{k+1}(t)$ at time *t* inside the *k+1 -th* cycle:

$$u_{k+1}(t) = u_k(t) + k_p e_k(t) \tag{4.1}$$

where the error $e_k(t)$ is:

$$e_k(t) = y_d(t) - y_k(t) \tag{4.2}$$

$u_k(t)$ is the input to the system in the k *-th* iteration, k is the iteration number, k_p is the proportional gain, $y_d(t)$ is the desired trajectory, $y_k(t)$ is the output of the system in the k *-th* iteration, and $t \in [0, T_{cycle}]$ is the time inside the cycle of duration T_{cycle}.

Please note that the uppercase **K** denotes the computed torque gains, whereas the lowercase k denotes the ILC gains.

Figure 4.1 shows the block diagram of this controller.

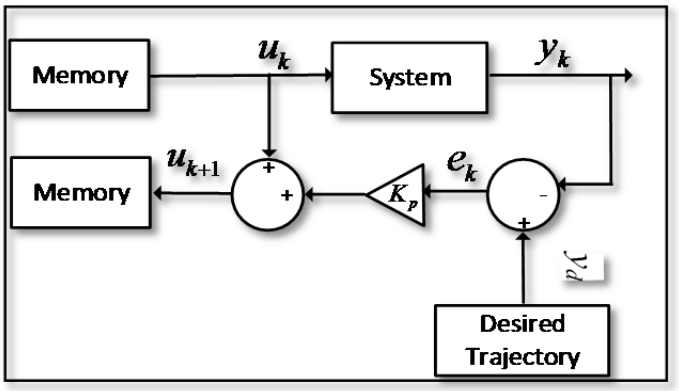


Figure 4.1 P-type ILC block diagram

The way that it works is that we run the system with the desired trajectory for the first time. In the meanwhile the error of the system is being saved in the memory with respect to the time for next iteration. In the next iteration we add a fraction of error with respect to time to the previous input which will be the new input to the system. Again in the meanwhile the system is saving the new error. And the same as before in the next iteration we add a fraction of new error to the new input for producing the current input.

4.1.2 PD-type ILC

The second type of ILC used here is the PD-type controller, in which there are both proportional and derivative gains. This type was introduced by Arimoto (Moore, 1993). The PD-type ILC we used has the following updating law:

$$u_{k+1}(t) = u_k(t) + k_p e_k(t) + k_d \left(\frac{de_k(t)}{dt} \right) \tag{4.3}$$

where k_d is the derivative gain, and $\frac{de_k(t)}{dt}$ is the derivation of error.

Here we add a fraction of error as well as the fraction of its derivation.

4.2 Combining ILC with computed torque

In this section we combine the ILC controller with the computed torque controller. This combination enables a 'shortcut' to be taken in the process of error reduction. Since the ILC does not have any information on trajectory error in the first iteration, the computed torque controller does all of the work here. This is the main benefit of using the computed torque controller in combination with the ILC controller – the computed torque controller can act in the first iteration and the subsequent ones, whereas the ILC can only act after the first iteration.

When the two controllers are combined, the output signal from the ILC enters the computed torque block where the required torque for the desired trajectory is calculated. This is a serial structure, a scheme of which is depicted in Figure 4.2. We chose to use a serial structure because we need the computed torque controller to be placed right after the robot in order to make a linear system.

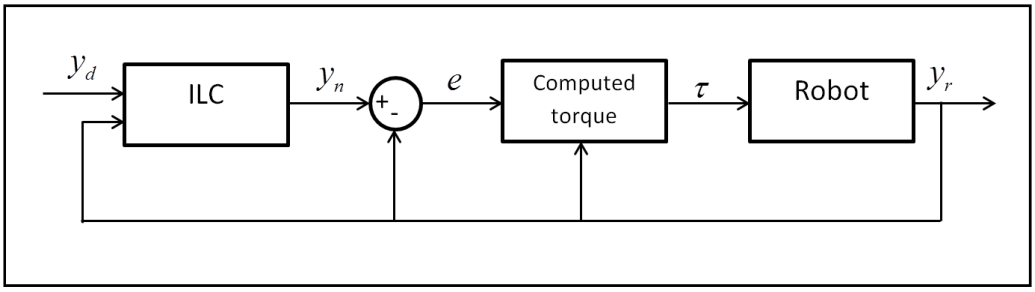


Figure 4.2 Combining ILC with computed torque

The structure in Figure 4.2 shows that the ILC modify the desired trajectory provided to the computed torque controller. This is to compensate the error done by the computed torque approach. The new trajectory given by the ILC with the error on this trajectory, caused by the computed torque, will give at the output of the robot a trajectory similar to the desired trajectory provided to the computed torque.

CHAPTER 5

SIMULATION RESULTS

5.1 Simulation in Simulink

For simulation, a model has been made in Simulink. A general schema of this model is shown in Figure 5.1.

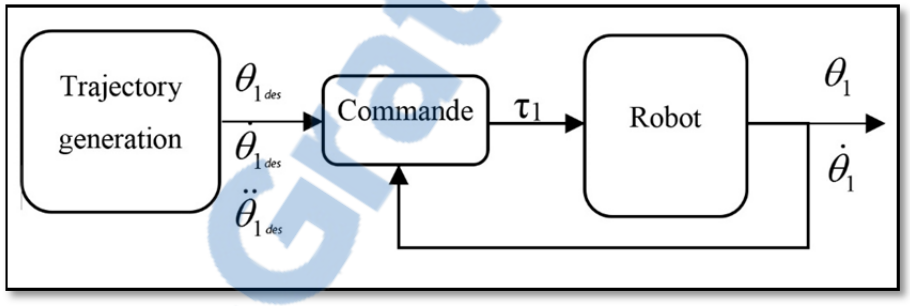


Figure 5.1 Simulation block diagram

The solver of Simulink throughout the simulation is 'ode8 (Dormand-Prince)' and the sample time is 1 ms.

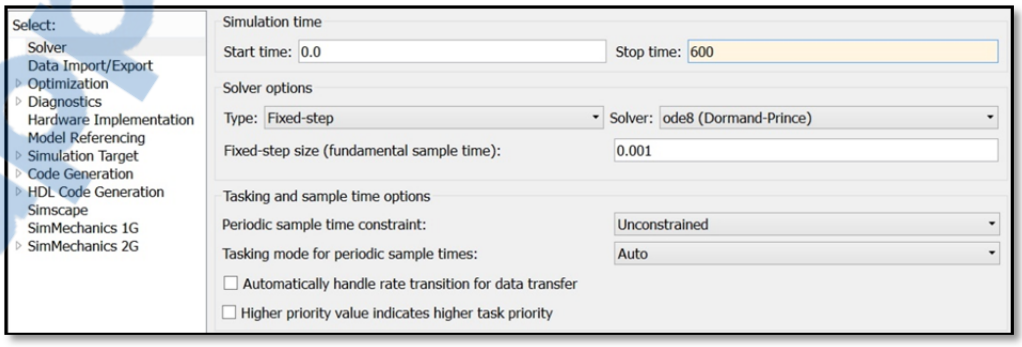


Figure 5.2 Solver configurations

5.1.1 Trajectory

The trajectory for the ILC follows these steps at each iteration:

1- The first link remains in the same position for t_d seconds. t_d is a delay time which depends on the response time at the computed torque controller. This delay time is useful because it allows the robot to stop oscillating and reach a steady state. The delay time also makes the trajectory symmetrical, which is necessary for the ILC to function (as it requires an identical trajectory to be followed at each iteration).

2- The first link goes from zero radian to π radians in 0.5 seconds, according to the seventh degree polynomial that was designed in Section 2.4 to enable smooth movement.

3- The first link remains at π radians for $2t_d$ seconds to ensure the robot is in a steady state and also to make the trajectory symmetrical.

4- The first link goes back through the same seventh degree polynomial to zero radian in 0.5 seconds.

5- The first link stays at zero radian for t_d seconds to ensure the robot is in a steady state and to make the trajectory symmetrical.

6- We begin the next cycle by going back to the first step.

Figure 5.3 shows all the steps of the trajectory of position, velocity, and acceleration, for two cycles. The position, velocity, and acceleration all start smoothly from zero.

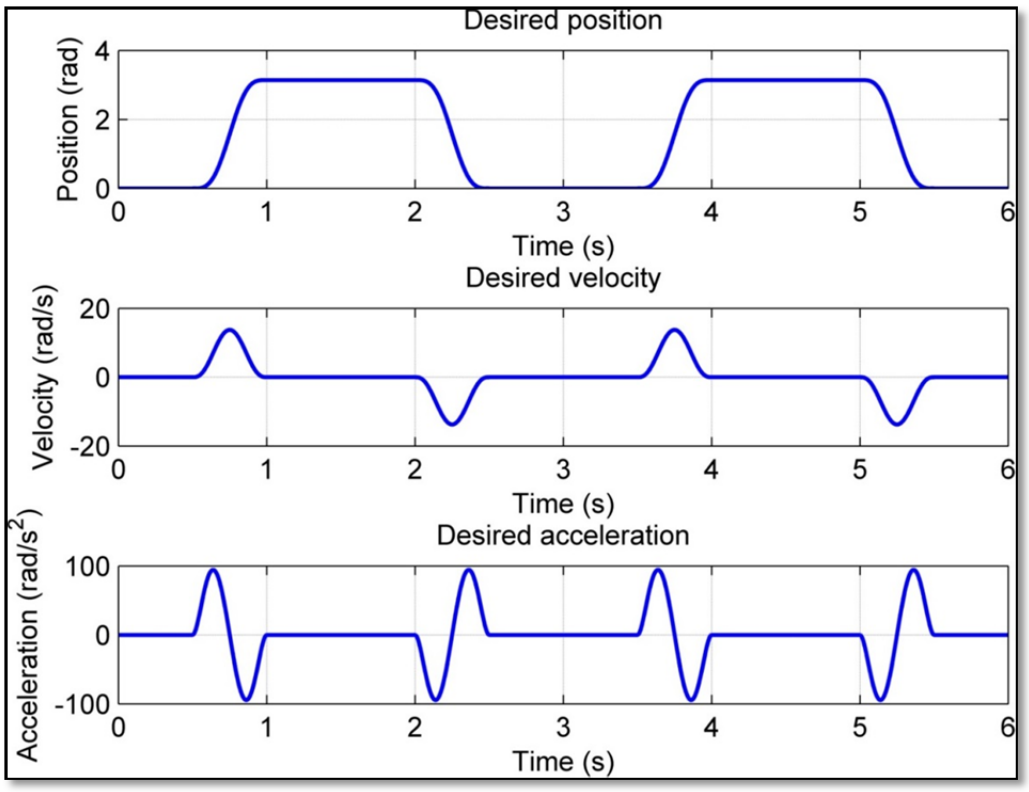


Figure 5.3 Two successive cycles of trajectory of position, velocity and acceleration

In order to test the computed torque controller in the simulation, only the first cycle of the trajectory is used.

The required torque for this trajectory is shown in Figure 5.4.

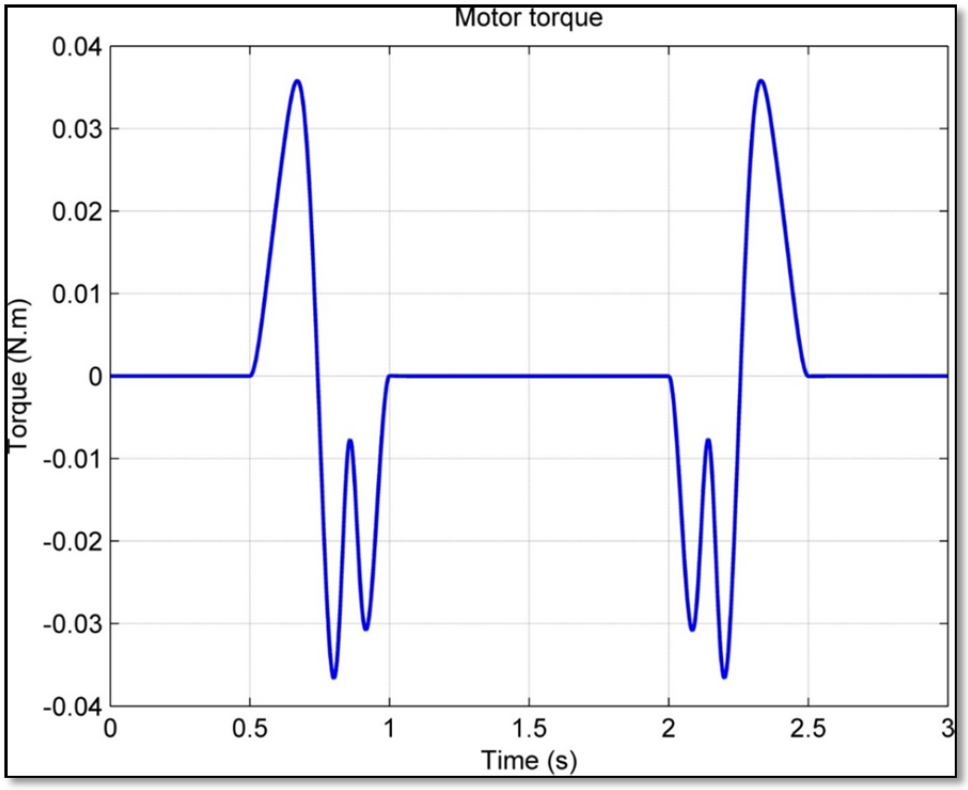


Figure 5.4 Required torque for the trajectory described in this section

Figure 5.4 shows that the maximum absolute value of the required torque is less than 0.04 N.m. The nominal torque (maximum continuous torque) for the motor on this robot is 128 mN.m, which is greater than the maximum required torque for this trajectory. So there is no problem for the robot motor to follow this trajectory. In other words, it is feasible, as required by the ILC assumption stated in Section 1.3.3. In Figure 5.4 we see that close to 0.8 s and 2.1 s the torque is oscillating a little. In these two moments the robot is close to its singularity. That is why we see some oscillation in the torque.

The mechanical parameters of the four-bars robot were presented in Table 2.1.

5.1.2 Computed torque results

In equation (2.89) we set the response time equal to 0.25 s for the PID computed torque controller. The values of $\mathbf{K}_p$, $\mathbf{K}_i$ and $\mathbf{K}_d$ are determined from **Erreur ! Source du renvoi introuvable.**, **Erreur ! Source du renvoi introuvable.** and (2.88) respectively:

$$\lambda = \frac{6.27}{T_r} = 25.08$$

$$\mathbf{K}_p = 3\lambda^2\mathbf{I} = 1887.0192$$

$$\mathbf{K}_i = \lambda^3\mathbf{I} = 15775.48051$$

$$\mathbf{K}_d = 3\lambda\mathbf{I} = 75.24$$

Figure 5.5 shows the position error of the closed loop system in radians for one cycle (or one iteration). The desired trajectory is one cycle of Figure 5.3.

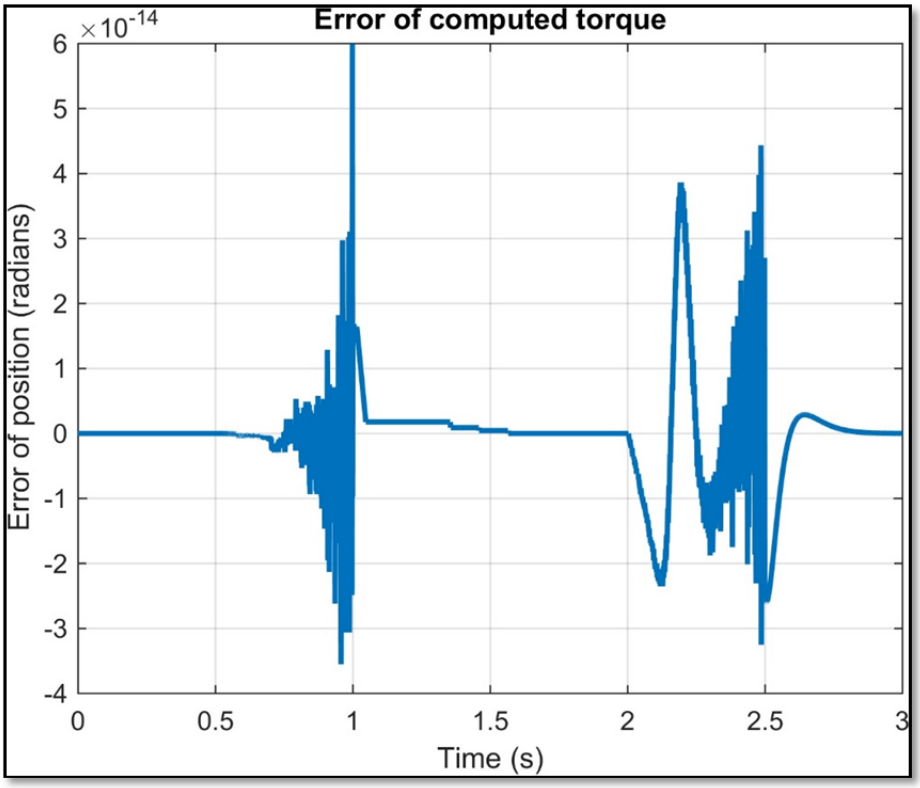


Figure 5.5 Position error for computed torque controller

In Figure 5.5 the order of error is 10^{-14}. This indicates that the model and the computed torque controller are identical and that the error probably comes from numerical integration. So the control process is complete. Eventually there would be no task remaining for the ILC to do. But we know that our model is not exactly the same as the physical robot, since we have neglected some parameters such as the motor's moment of inertia, friction of joints, and elasticity of belt. So in the computed torque controller, we add 0.001 $kg.m^2$ to the moment of inertia of the first link, as a perturbation to the model.

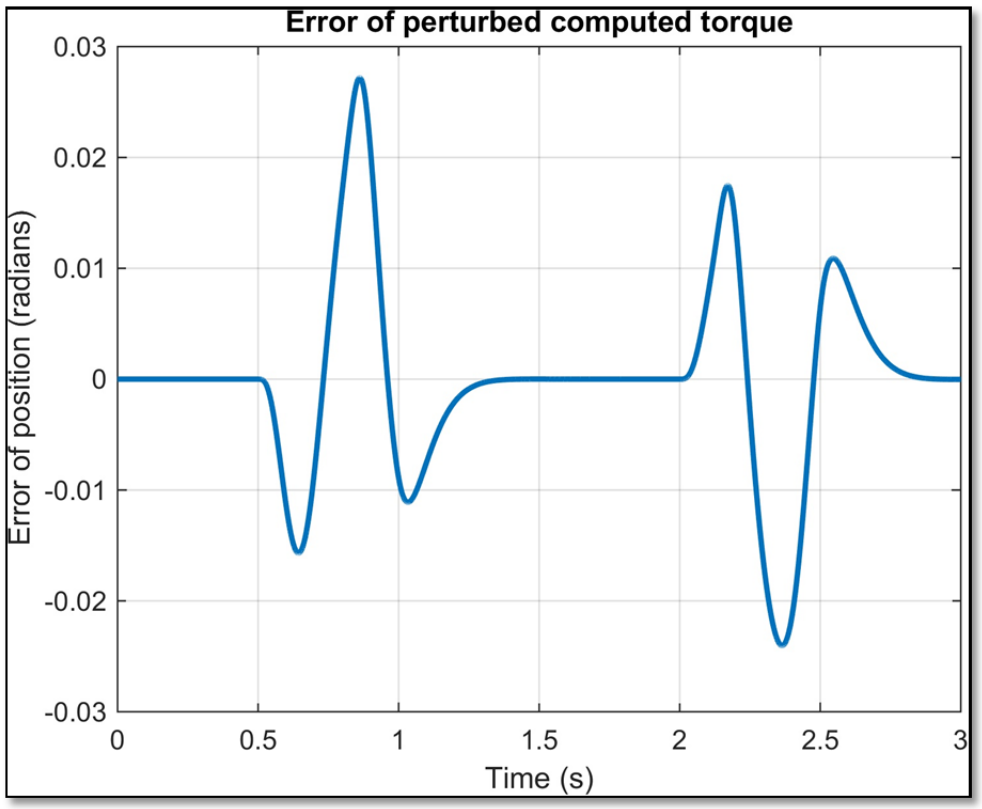


Figure 5.6 Position error for perturbed computed torque controller

Figure 5.6 depicts the error of the perturbed computed torque controller. In this case we can better verify the effect of ILC on the system. All the simulations from now on are done with the perturbed model.

5.1.3 Results of computed torque with ILC

Here the two types of ILC are combined with the computed torque. This section describes how they are tested, and presents the results. 200 iterations are performed for each test and the t_d is 0.5 seconds. In other words, the cycle time is 3 seconds. The response time of the computed torque for all the tests is 0.25 seconds.

5.1.3.1 Computed torque with P-type ILC

The proportional gain of ILC k_p is set arbitrarily to 0.5, and 0.05 to verify the effect of gain on the behavior of the closed loop system. (Please recall that the uppercase **K** denotes the computed torque gains, whereas the lowercase k denotes the ILC gains.) The trajectory is the same as Figure 5.3. The cycle duration is 3 seconds and we did 200 repetitions in 600 seconds since we had the problem of memory for iterations above 200. Figure 5.7 depicts the results of the position error for $k_p = 0.5$.

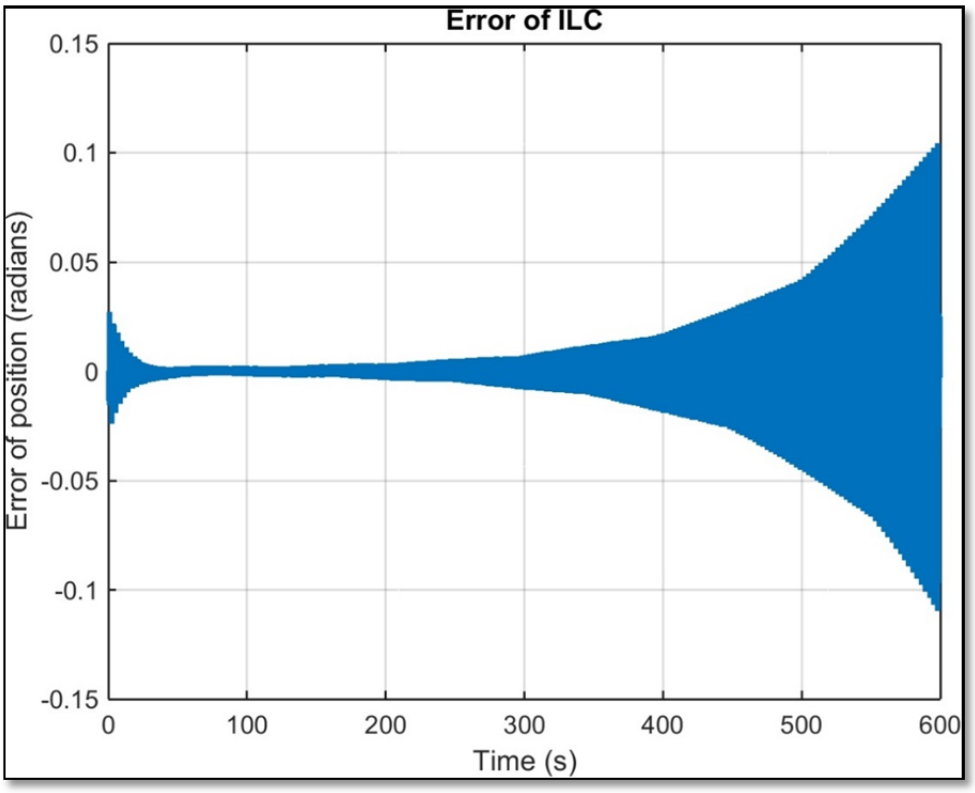


Figure 5.7 Position error for P-type ILC with $k_p = 0.5$

As can be seen from Figure 5.7, the error starts to diverge from zero after about 150 seconds. It can be because of the high value of gain, and also because of the discontinuity of trajectory that results from passing from one step to the next step.

In the next step we decrease the gain to $k_p = 0.05$ in order to see if the error converges to zero.

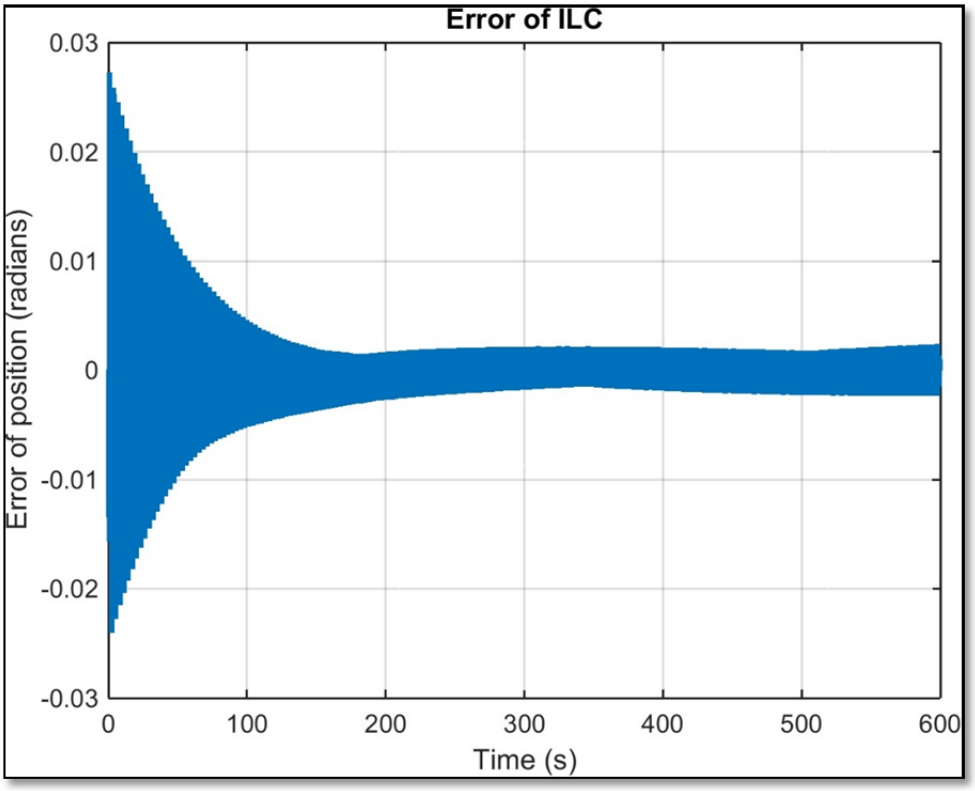


Figure 5.8 Position error for P-type ILC with $k_p = 0.05$

From Figure 5.8 one may observe a good but slow convergence of error to zero. However, we can still see that the error might diverge if we continued the procedure. It seems that $k_p = 0.05$ is satisfying for the case of P-type ILC during 600 seconds. The convergence can be better seen in Figure 5.9, which shows the RMS of error. The RMS of error is calculated using equation (5.1), as follows:

$$error_{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{5.1}$$

where $error_{RMS}$ is the root mean square (RMS) of error in one cycle, n is the total number of samples at the end of one cycle, and e_i is the error at the i *-th* sample.

As Figure 5.9 shows, the error is reduced to about 0.05 times the error of the first cycle, in which only the computed torque functioned as the controller.

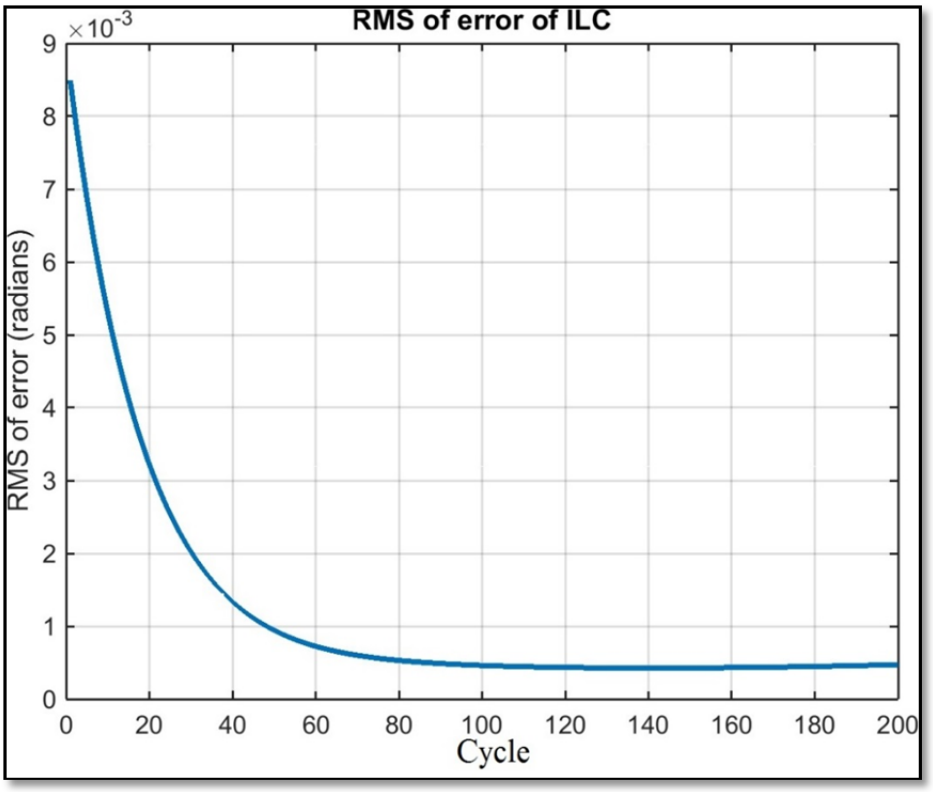


Figure 5.9 RMS position error for P-type ILC with $k_p = 0.05$

To verify the results with other criteria, the results are presented for the maximum values of the absolute error at each cycle. It is still clear from Figure 5.10 that the error does not diverge.

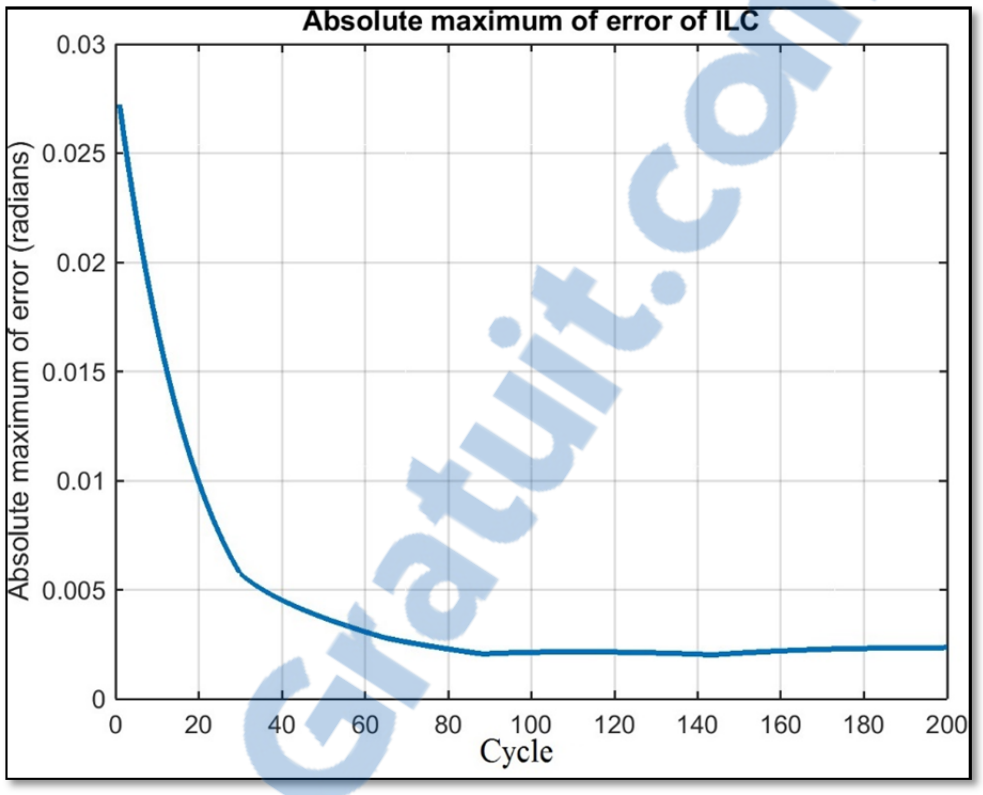


Figure 5.10 Maximum values of absolute error at each cycles

In Figure 5.11 a comparison between the first and the last cycle of position error for the first link is shown.

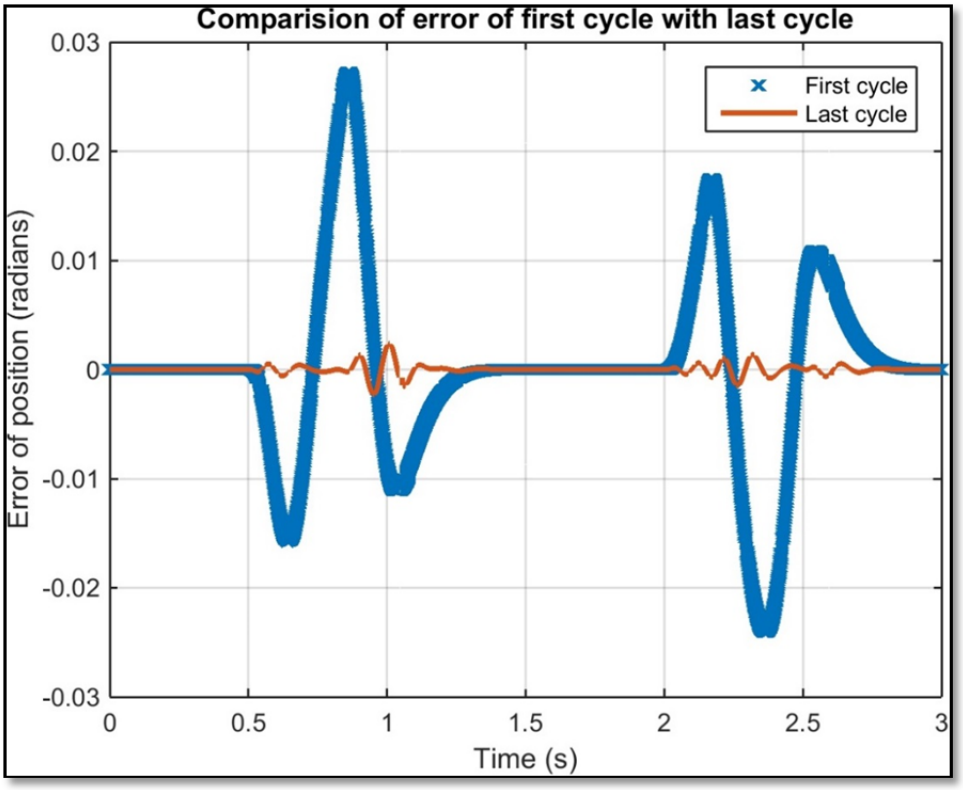


Figure 5.11 Comparison between the error of the first and the last cycles for P-type ILC

5.1.3.2 PD-type ILC

Here the results of PD-type ILC are presented. Since the error plots do not give any more information than the RMS plots, from now on we will only present the RMS error for each case.

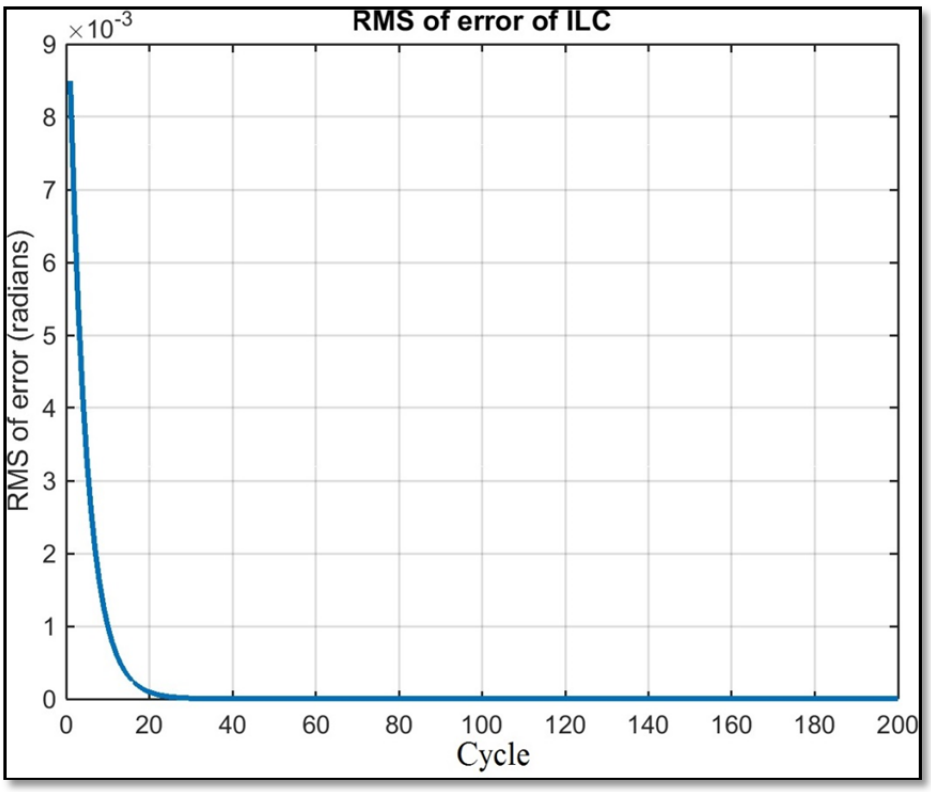


Figure 5.12 RMS position error for PD-type ILC with $k_p = 0.2$ and $k_d = 0.2$

Figure 5.12 is the case where $k_p = 0.2$ and $k_d = 0.2$. It is clear that the error has quickly and greatly decreased, and more importantly the system is stable throughout the 200 repetitions. Figure 5.13 also shows the comparison between the error of the first and the last cycles for this type of ILC.

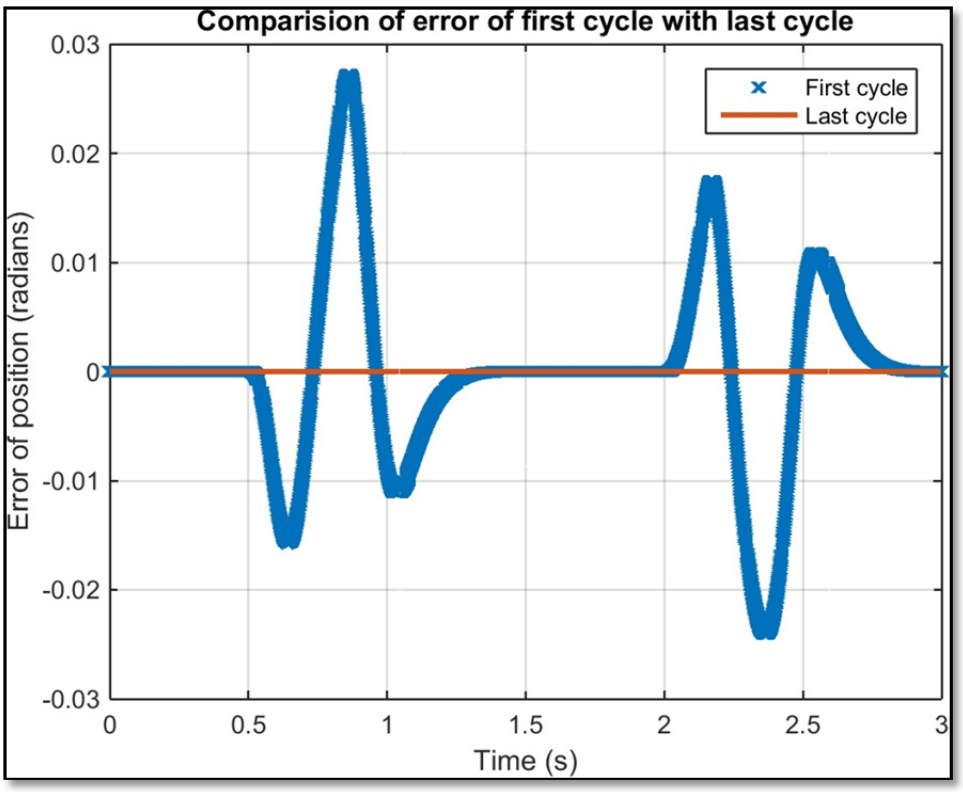


Figure 5.13 Comparison between the error of the first and the last cycles for PD-type ILC

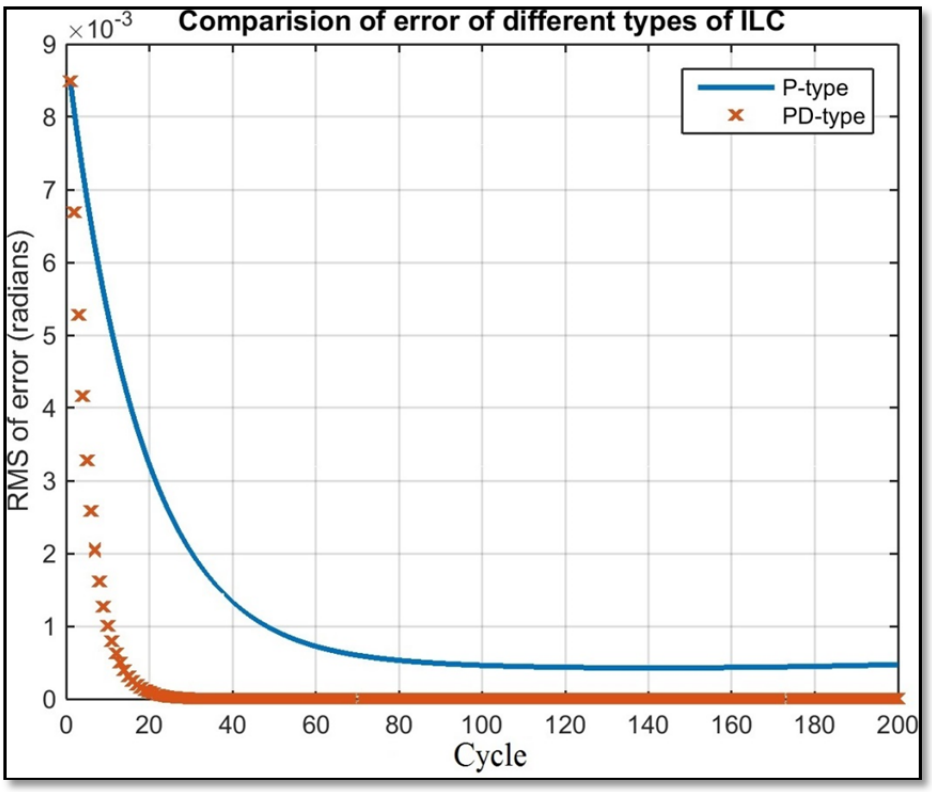


Figure 5.14 Comparison of two types of ILC according to the RMS error convergence

It can be seen from Figure 5.14, that the PD-type ILC has the best convergence. It is faster and the error is closer to zero. And the P-type ILC seems to start diverging slowly at the end of the tests. And the P-type ILC appears to begin to diverge at the end of the test.

CHAPTER 6

EXPERIMENTAL RESULTS

6.1 Real robot

In this chapter the same trajectory as in the simulation is implemented with the real robot. We use TwinCAT 3 compiler to compile exactly the same diagram-blocks of the controller in Simulink that we used in the simulation, so it would be able to upload on TwinCAT 3 software. Figure 6-5 presents the setup.

6.2 Validation

We will validate the macro-generated SimMechanics model that was presented in Chapter Three by comparing its performance during a simulation with that of the real robot during a practical experiment.

6.2.1 SimMechanics

To ensure the macro described in Chapter Three functions sufficiently, we will generate the SimMechanics file of the parallel robot from the CAD file in CATIA. By adding the actuator and sensor, we will have the dynamic model of the robot in SimMechanics. For the validation, two computed torque controllers are designed using SimMechanics and the rigid model from the previous chapter. These two controllers are used to control the position of the first link of the real robot. If the robot follows the trajectory with just a small error, then the controller is performing well and we can conclude that the SimMechanics model is a good approximation of the real robot.

Once we have the forward dynamic model in SimMechanics we can use it for inverse dynamics too. With the inverse dynamic model, we can calculate the torque that is needed for the robot to follow a specific trajectory. To obtain the inverse dynamics model, some of the

settings in the model need to be changed. This is done by double-clicking on ‘Actuator’: in the ‘Actuation’ section, the parameter ‘Actuate with’ is set to ‘Motion’ and the related units ‘Angular units,’ ‘Angular velocity units,’ and ‘Angular acceleration units’ are set respectively to rad, $\frac{rad}{s}$, and $\frac{rad}{s^2}$.

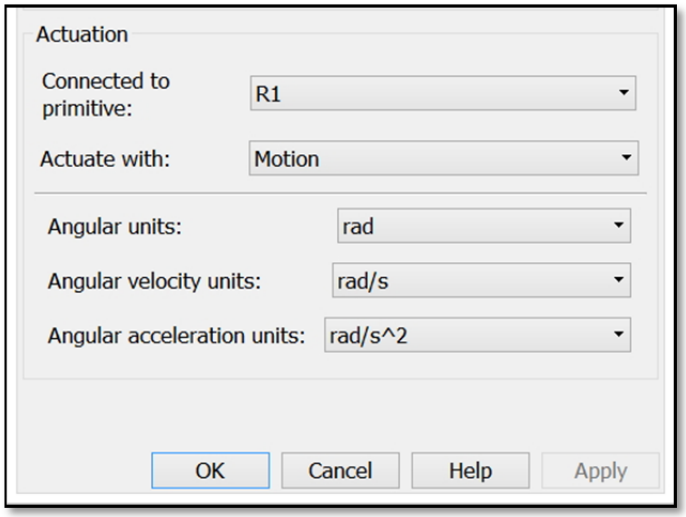


Figure 6.1 Actuator settings for obtaining the inverse dynamics model

We need a sensor to calculate the required torque for the motion that we will apply to the actuator. This is done by double-clicking on ‘Sensor’: in the ‘Measurements’ section, under ‘Primitive Outputs,’ uncheck ‘Angular velocity,’ and check ‘Computed torque.’ The ‘Units’ are set to ‘*N*m*.’

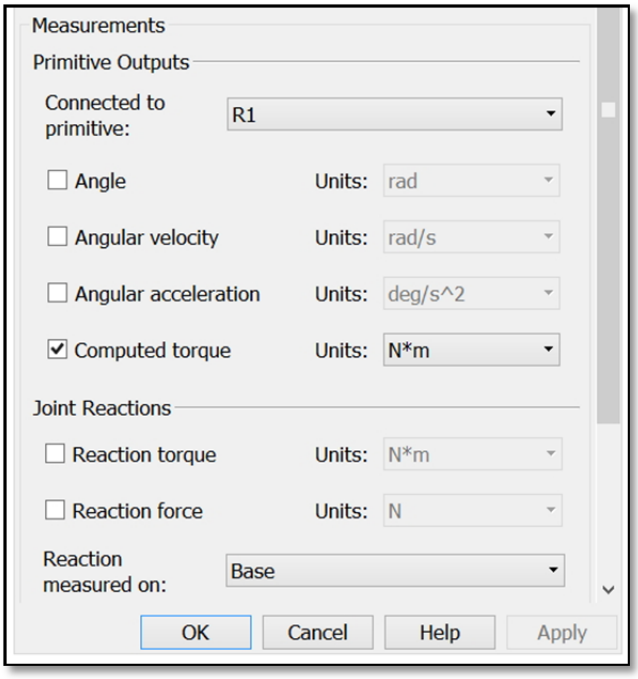


Figure 6.2 Sensor settings for obtaining the inverse dynamics model

It is possible to use this block for the computed torque. The position feedback from the real robot is directly connected to the first input of the actuator, and the robot's velocity feedback is connected to the second input. In order to use the PID computed torque, the position error is multiplied by the proportional gain $\mathbf{K_p}$, the velocity error is multiplied by the derivative gain $\mathbf{K_d}$, and the integral of the position error is multiplied by the integral gain $\mathbf{K_i}$ calculated in Section 2.5.1. The response time is 0.25 seconds. The sum of these three signals, plus the desired acceleration, produces the signal that goes directly to the acceleration input of the actuator. Figure 6.3 presents a schema of this implementation.

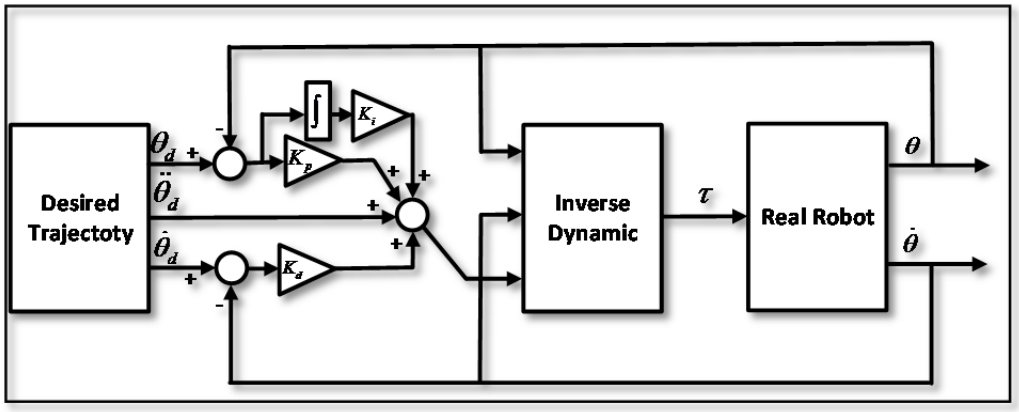


Figure 6.3 PID computed torque using inverse dynamic

After preparing the model, it must be compiled in a format that is compatible with the software that communicates with the robot. On a computer where TwinCAT 3 is already installed, we browse to find 'TwinCAT.tlc' in the following path from the SimMechanics file that we have already prepared for the validation: 'Configuration Parameters\ Code Generation.' In the 'Target selection' part, browse the 'System target file' to find 'TwinCAT.tlc.' Here we change the 'Language' to 'C' to enable compilation of the SimMechanic blocks.

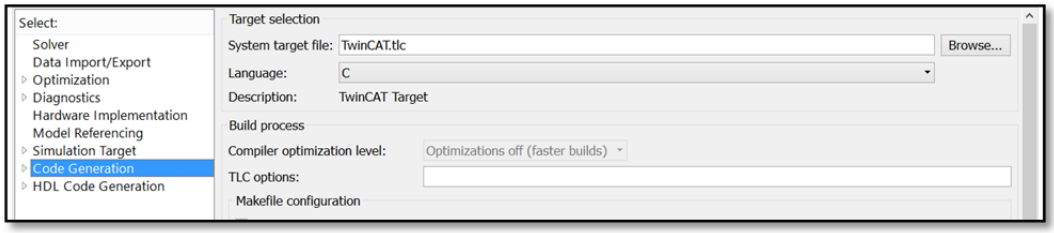


Figure 6.4 Configuring Simulink for TwinCAT file generation

After clicking on the 'Build model' button in the menu at the top of the window, Matlab will begin to compile and generate the file for TwinCAT.

6.2.2 Hardware

For the controlling task, we use an industrial PC made by Beckhoff. The model of this PC is C6920-0040. The operating system is real-time Windows 7. It makes it possible to run all the executable software on Windows directly on the PC. This PC is also equipped with USB and LAN ports.

The servo drive is from the Whistle series manufactured by Elmo Motion Control Ltd. This digital servo drive is small but powerful (3200 W peak and 1600 W continuous power). It is used for DC brush, brushless, and linear motors. The Elmo Whistle drive can operate in three different modes: position, velocity and current. It is also equipped with a LAN port.

6.2.3 Connection

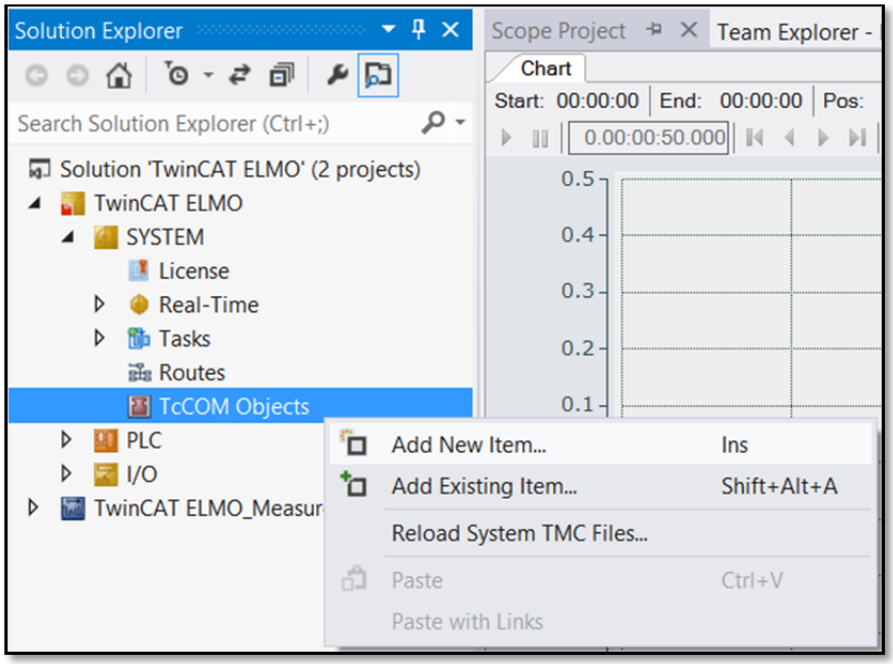
The connection between the servo drive and the industrial PC is through the LAN ports. The protocol is EtherCAT. EtherCAT is developed by Beckhoff and is a real-time industrial Ethernet. The EtherCAT protocol is suitable for hard and soft real-time requirements in automation technology, testing, and measurement.

6.2.4 Software

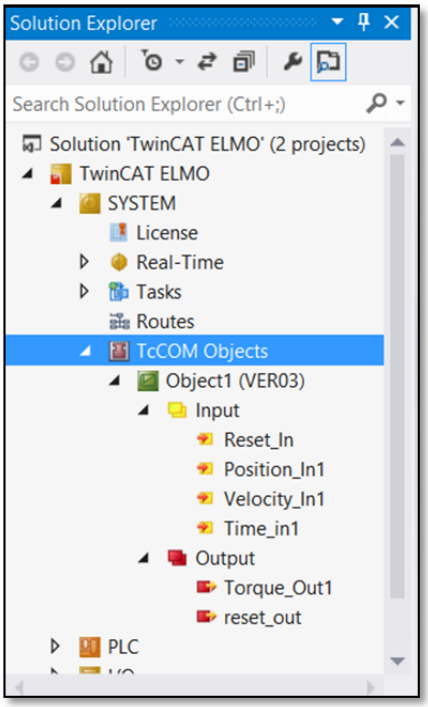
TwinCAT (The Windows Control and Automation Technology) is the center of the control system. TwinCAT 3 has many features. The most important one for our purposes is the ability to link to Matlab/Simulink. It is possible to compile a Simulink file to generate a TwinCAT file. Then we can load this file into TwinCAT 3 to run it in real time. In our case the controller of the four-bar robot can also be compiled.

After compiling the SimMechanics model with the controller, we have the TwinCAT file that will be loaded and configured in TwinCAT software. Before opening this file in TwinCAT we need to make a new project in this software in which we define the inputs, outputs,

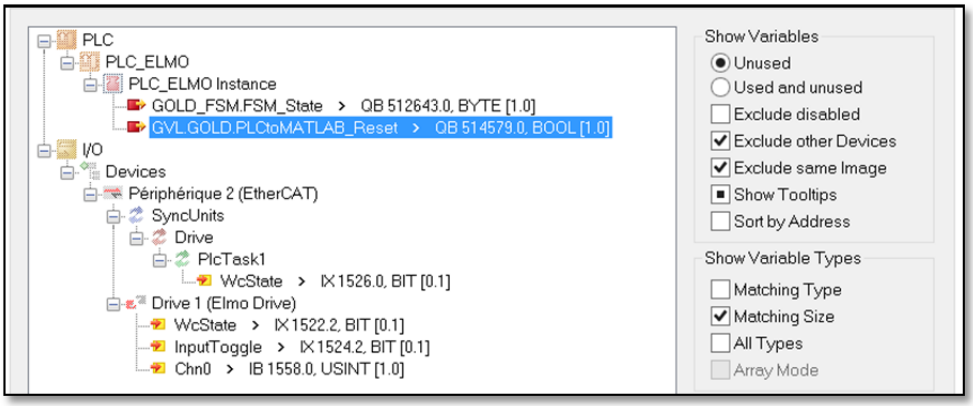
restart, and timer. This new project will be our template, in which we will load our compiled model each time. Making the template is not covered in this thesis. Once we have made it, we open the TwinCAT software and select the template that is built for our robot. In the menu on the left side of the window in ‘Solution Explorer,’ we right click on ‘TcCOM Objects \ Add New Item.’



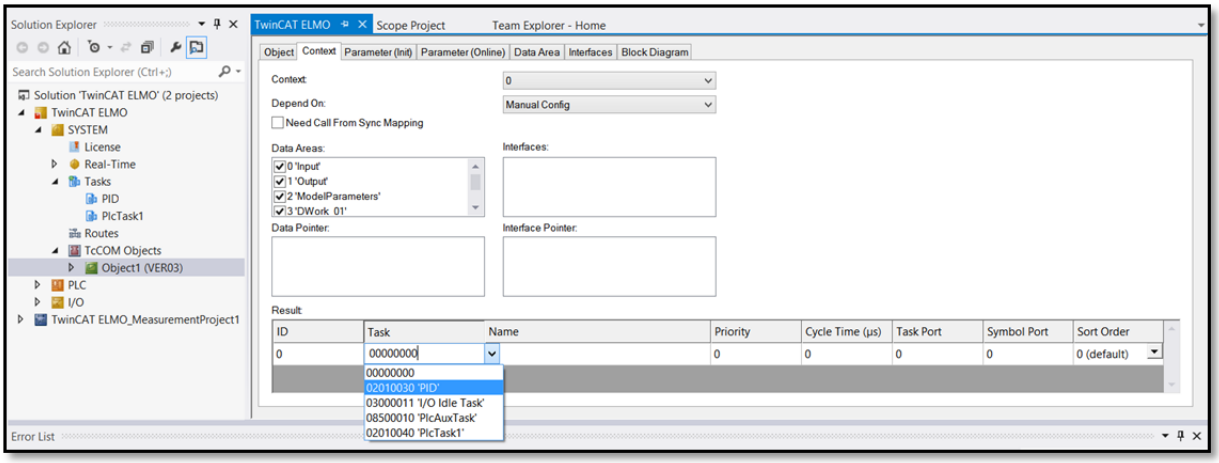
In the open window, we browse to find the compiled file that is made in SimMechanics. The inputs and outputs will appear under ‘TcCOM Objects,’ according to the names we selected while we were making the template.



We double-click on each of the inputs and outputs and attach each signal to the appropriate variable by double-clicking on it.



Then we double-click on the 'Object' under 'TcCOM Objects,' and in the second tab 'Context' we select the 'Task' that we defined while making the template. Here we have named it 'PID.'



Next we load the file on the robot. To do so, we click on ‘Activate Configuration’ at the top right of the main menu. Figure 6.5 shows the configuration of robot.

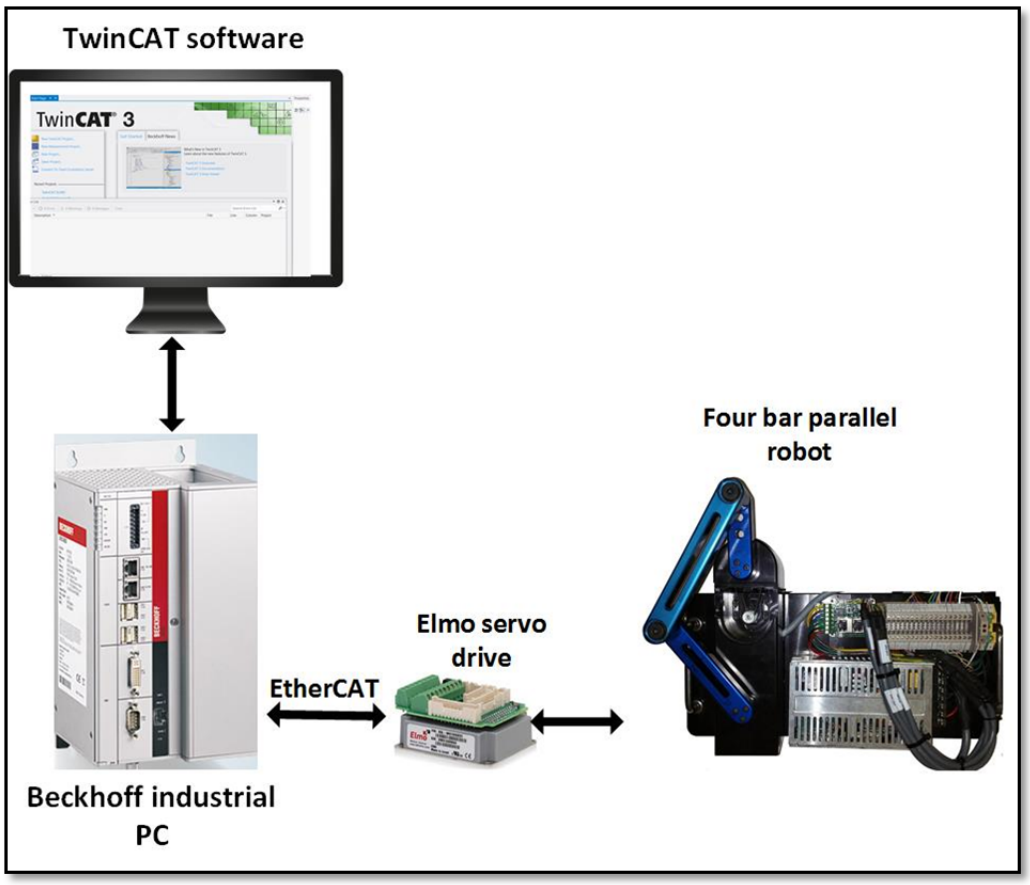


Figure 6.5 Robot setup

6.3 Experimental results

The experimental tests are done on the computed torque and the two types of ILC. This section presents the plots of error and RMS error for these tests.

6.3.1 Computed torque results

We first verify the computed torque controller alone, before moving on to verification of the ILC controllers. The desired trajectory is the same as the one we did our simulations with; it is depicted in the first cycle of Figure 5.3. All the conditions are the same as in the simulation part, including the value of the PID gains.

Figure 6.6 shows the tracking position error of the first link. The robot undergoes one cycle. This means the first link moves from 0 to π radians in 0.5 seconds, waits 0.5 seconds at π radians, and then returns to 0 in 0.5 seconds. The cycle time is 3 seconds. The error of computed torque in the experimental part is comparable to the error of perturbed computed torque in the simulation (Figure 5.6). Errors are in the same order.

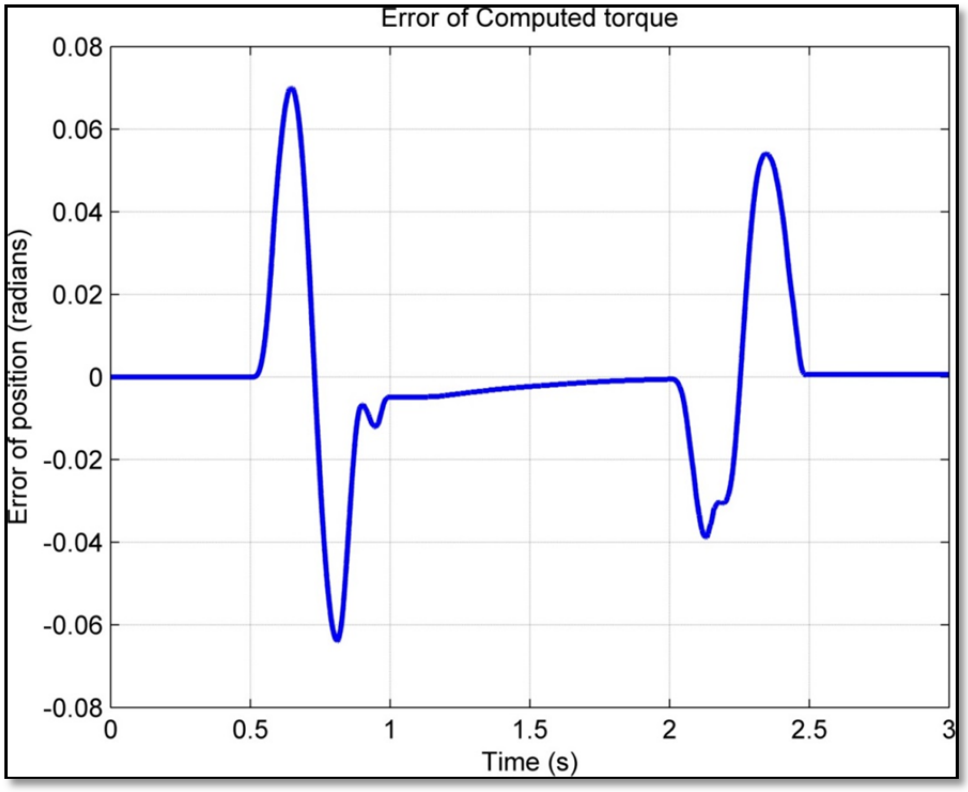


Figure 6.6 Position error for computed torque controller

6.3.2 P-type ILC

Here we test the P-type ILC that is integrated with the computed torque controller on the real robot, using the same gain as in Section 5.1.3.1.

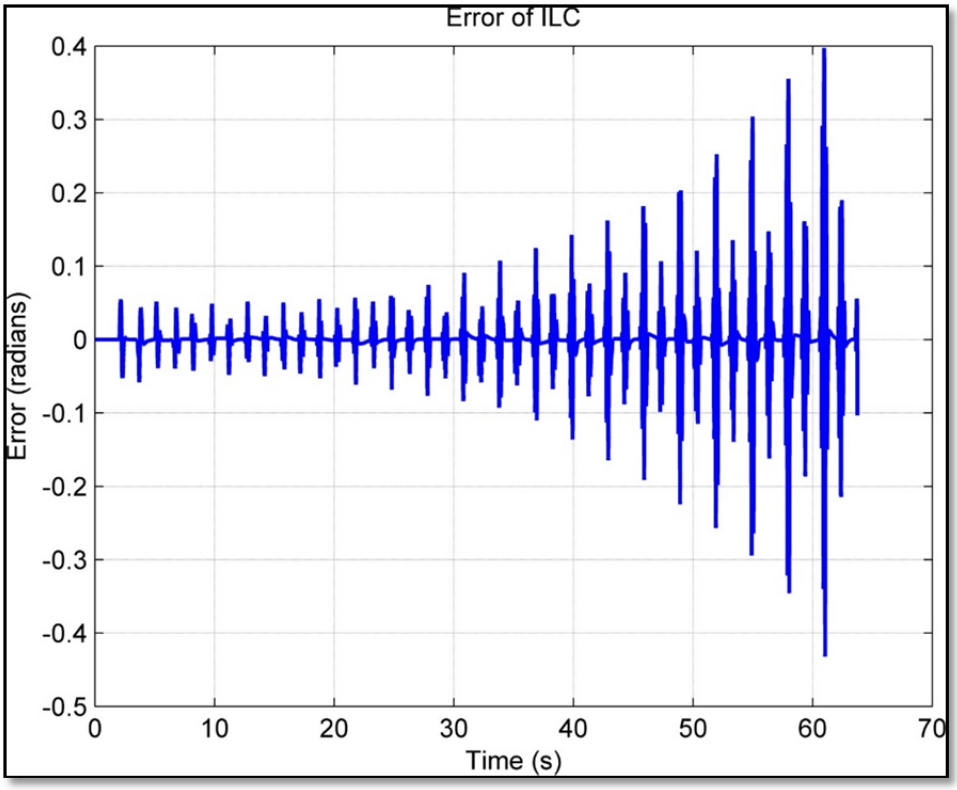


Figure 6.7 Position Error for P-type ILC with $k_p = 0.5$

As shown in Figure 6.7 and Figure 6.8, the P-type ILC does not exhibit good results for $k_p = 0.5$, since the error diverges. The same thing occurred during the simulation.

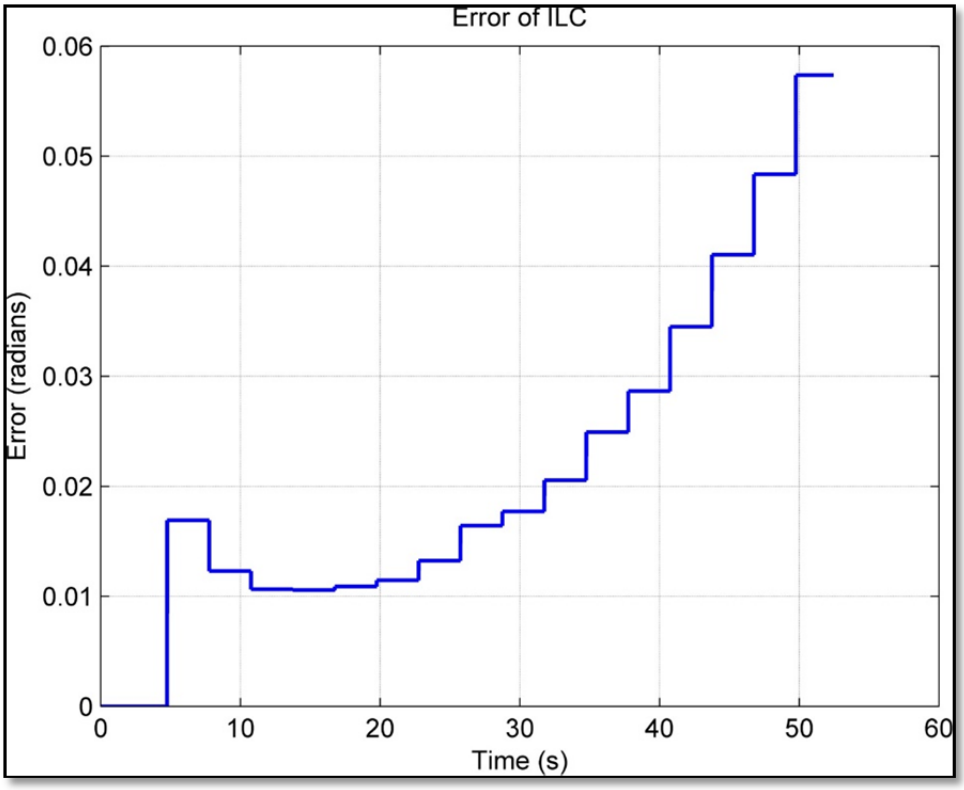


Figure 6.8 RMS position error for P-type ILC with $k_p = 0.5$

To achieve an acceptable result by trial and error, we found that $k_p = 0.01$ presents good results for the P-type ILC when considering the RMS error. But Figure 6.9 shows there are some peaks that seem to increase, so this would not be a good control approach for this application.

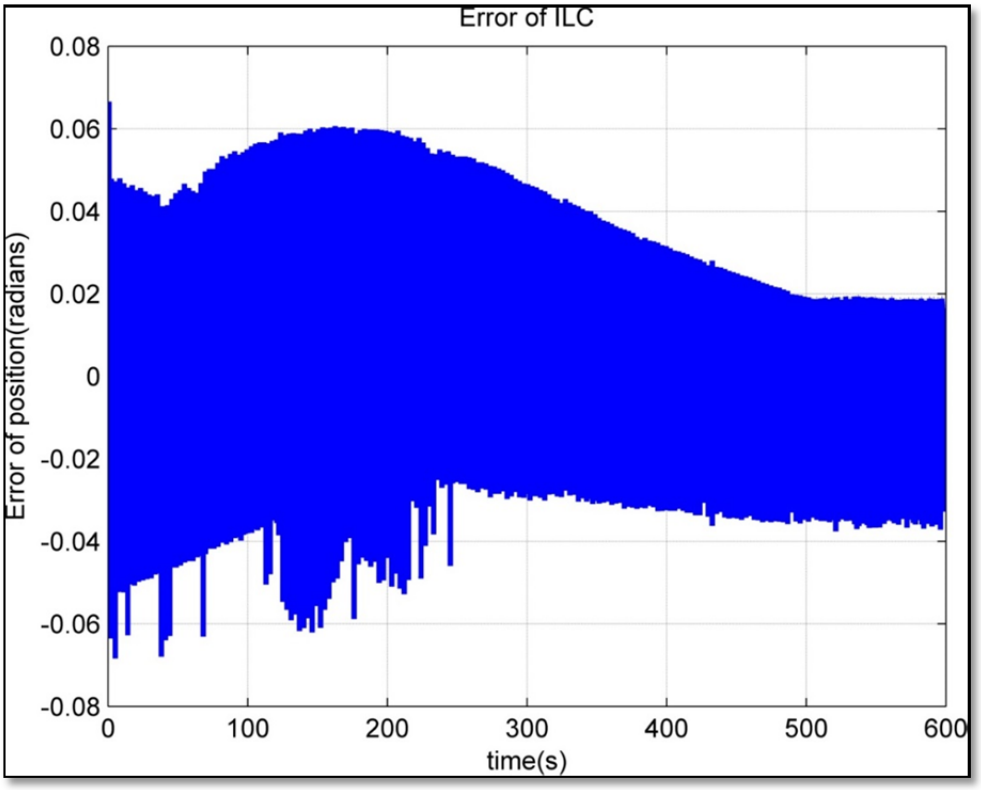


Figure 6.9 Position error for P-type ILC with $k_p = 0.01$

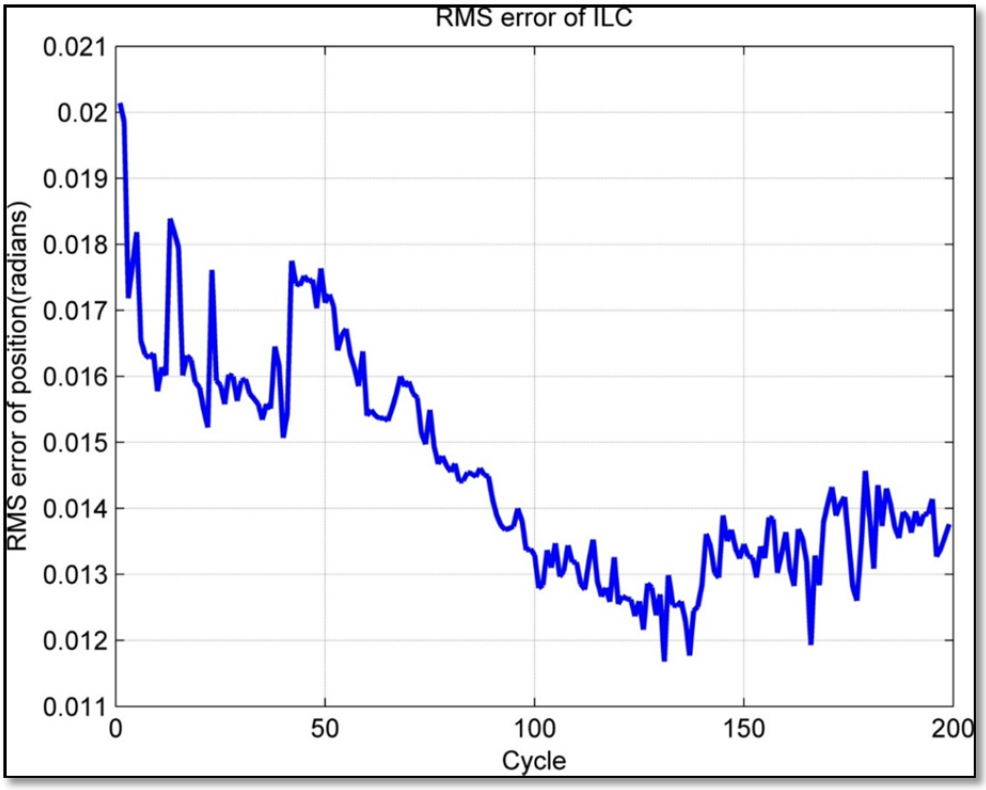


Figure 6.10 RMS position error for P-type ILC with $k_p = 0.01$

Despite the fact that the RMS of error indicates convergence, divergence can still be expected at repetitions above 200, and therefore the maximum absolute error of the last 50 cycles is expected to diverge from zero. Figure 6.11 shows the maximum absolute error, which confirms our suspicion that divergence occurs for this type of ILC.

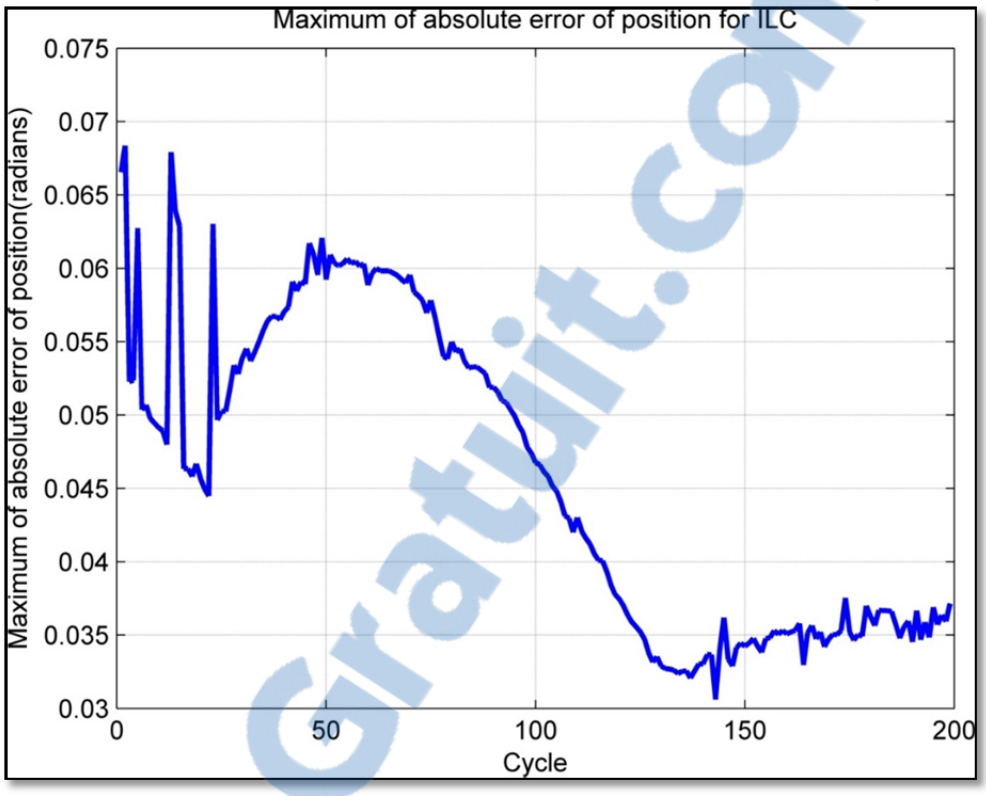


Figure 6.11 Maximum absolute error of position for P-type ILC with $k_p = 0.01$

A comparison of the error in the first and last cycles can be seen in Figure 6.12.

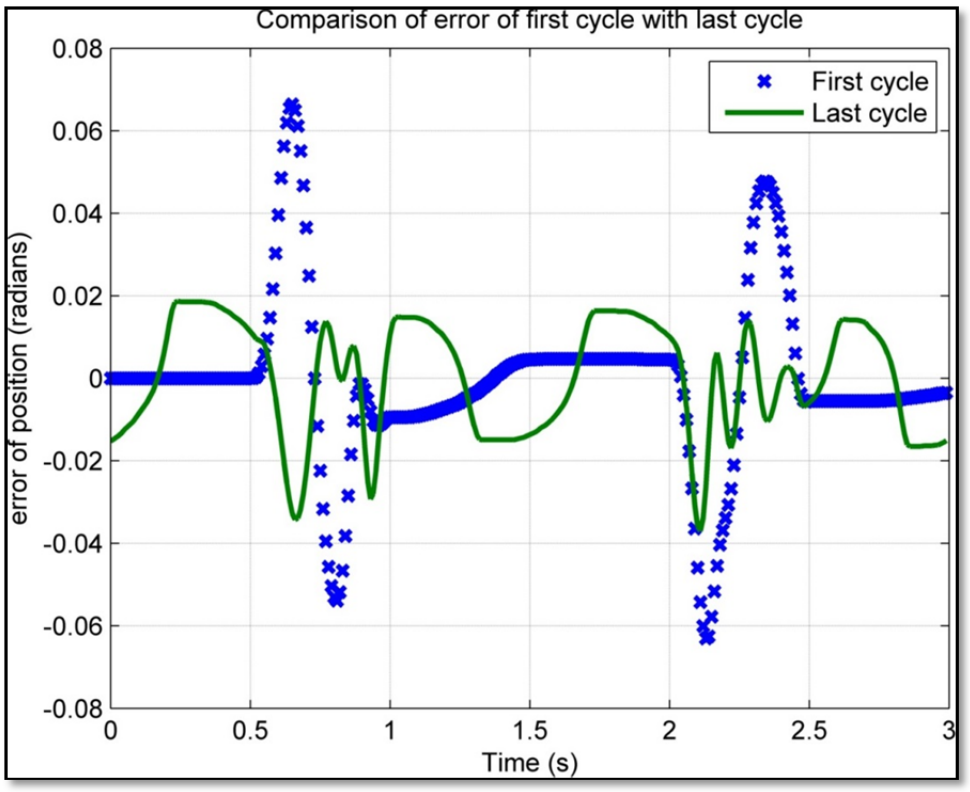


Figure 6.12 Comparison of error in first and last cycles for P-type ILC with $k_p = 0.01$

Figure 6.12 shows that the error has overall been reduced, but still is not perfect.

6.3.3 PD-type ILC

Figure 6.13 shows better convergence of the RMS error for the PD-type ILC with $k_p = 0.02$ and $k_d = 0.02$, compared to the P-type ILC. The results indicate that the PD-type ILC is able to shrink the error to approximately 0.25 times the error of the computed torque controller. There is a bump at around the 60^{th} cycle. ILC is very susceptible to change. During our experiments, we found that even a small change in conditions (e.g. friction, stiffness, small vibrations of the base of the robot, etc.) leads to a big change in the results.

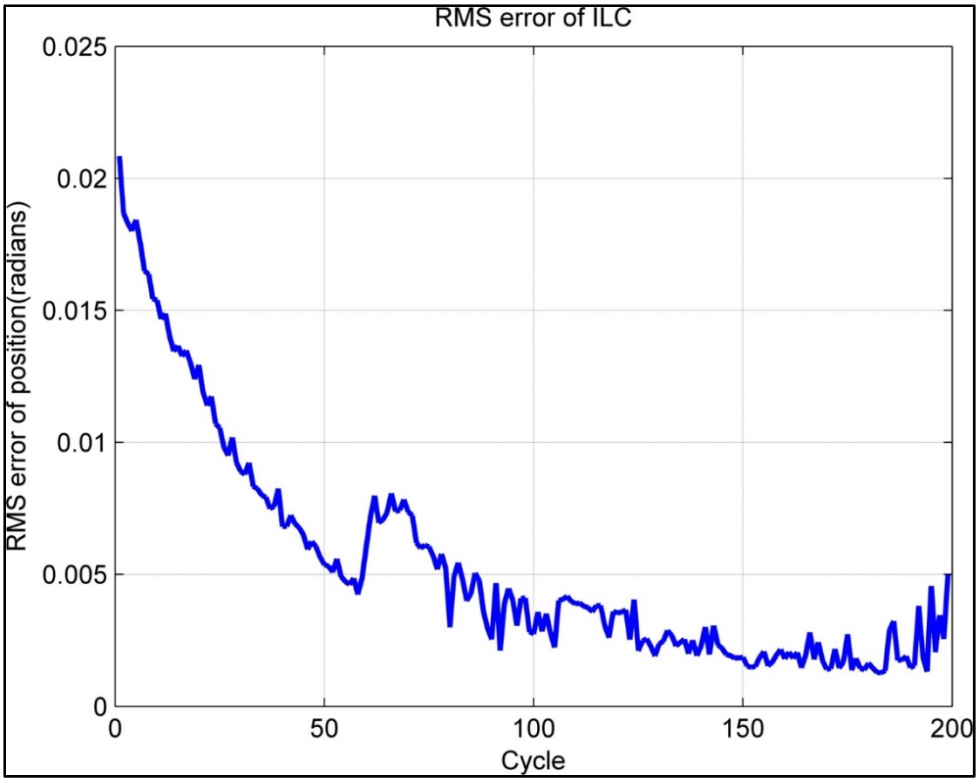


Figure 6.13 RMS position error for PD-type ILC with $k_p = 0.02$ and $k_d = 0.02$

To better understand the effect of the ILC controller, Figure 6.14 shows the maximum absolute error, and Figure 6.15 shows a comparison of the error in the first and last cycles.

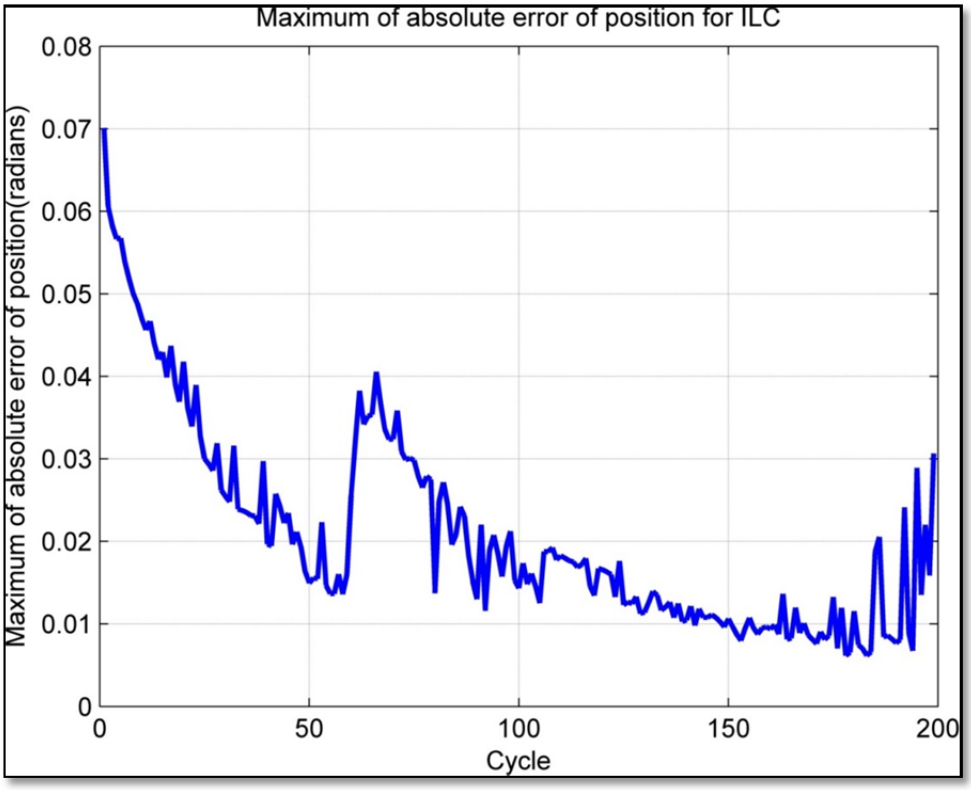


Figure 6.14 Maximum absolute position error for PD-type ILC with $k_p = 0.02$ and $k_d = 0.02$

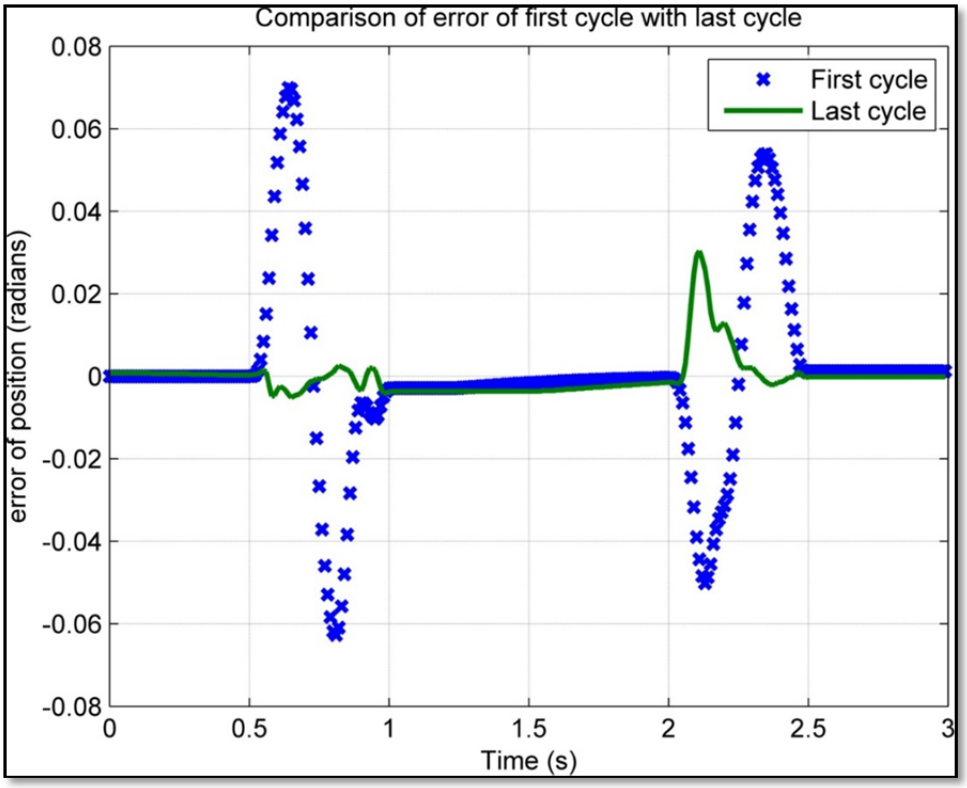


Figure 6.15 Comparison of first and last cycles' error
for PD-type ILC with $k_p = 0.02$ and $k_d = 0.02$

6.4 Comparison of two types of ILC

Here we compare the RMS error of the two ILCs that were presented in previous sections.

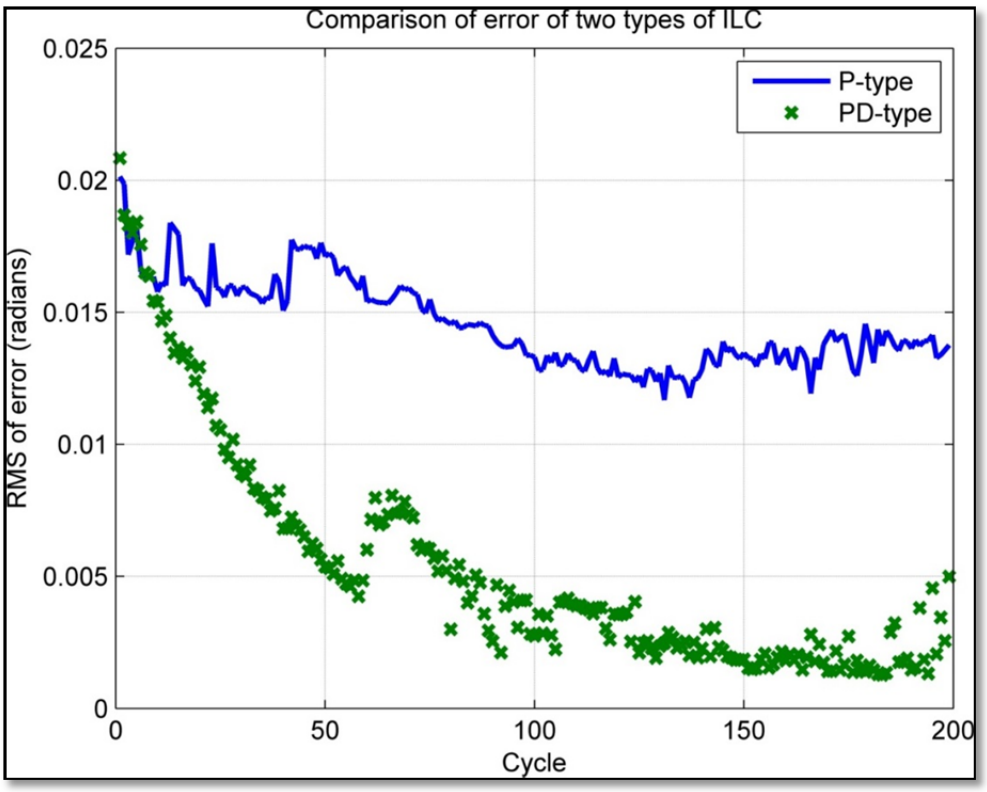


Figure 6.16 Comparison of the RMS of error convergence for the two types of ILC

As seen from Figure 6.16, the PD-type ILC has the fastest convergence. The PD-type ILC also exhibits the smallest values for error.

CONCLUSION

In this thesis we modeled a four-bar parallel robot. This robot was selected because it is easy to derive the equations of motion, and its dynamic model is non-linear. The dynamic model, which we calculated manually, using the Lagrange method, treated the robot as composed of four rigid components. After finding the equations of motion, we designed a computed torque controller. We then made a simplified model of the robot in SimMechanics in order to validate our manually-calculated (analytical) model**.** We compared the SimMechanics model to the analytical model. The results show that the analytical model is reliable.

This thesis has validated a new method of virtual prototyping. The new method is to use a macro that is added to CATIA V6 and enables the dynamic model to be generated and exported to SimMechanics. Then any sensors and actuators can be added to the model in SimMechanics, along with any other necessary modifications, to complete the dynamic model. We used two control methods to validate the functionality of the macro: computed torque and ILC. For the ILC we used two types, P-type and PD-type, which were used in serial configuration with the computed torque and the robot. With these methods, we conducted three simulations: the first used computed torque alone, the second used a combination of computed torque and P-type ILC, and the third used a combination of computed torque and PD-type ILC.

For the purpose of testing the controller combinations, a seventh degree polynomial was designed for trajectory. This is a smooth and fast trajectory. The simulation results show that the fastest convergence was done by the combination of computed torque and PD-type ILC. This combination also resulted in the smallest error value.

To compare the performance of the model with the performance of the actual robot, the actual robot was used with a Beckhoff controller for a controlling task. The results of the practical experiment show that the PD-type ILC is preferable to the P-type (both types were used in conjunction with the computed torque controller). We had to use smaller gains in the

practical experiment than in the simulation in order to prevent the error from diverging. The error was bigger in the practical experiment than in the simulation. This could be because in our virtual model we did not take into account several factors that affect the real robot, such as friction, the mass of screws and bolts, the moment of inertia of the rotor and pulleys, and the stiffness of the timing belt.

RECOMMENDATIONS

We have several recommendations for future research. Several changes should be made in hardware as well as software. First, regarding software we recommend trying other configurations of the controller. For example, the ILC controller parallel could be made with computed torque instead of the serial structure that we verified in this paper. Second, we recommend changing the trajectory to observe the effect of other trajectories. Here, the trajectory of the first link was $[0,\pi]$, but one might experiment with another trajectory such as $[0,2\pi]$ and do a complete circle. Third, there are many other types of ILCs apart from those we mentioned. For instance, we suggest testing the ILC when using the CITE approach, which uses previous and actual cycle error. Finally, during virtual prototyping we only considered three types of joint constraints: revolute, prismatic and fixed. Future researchers might improve upon our work by including other types of joint constraints such as gear or universal.

LIST OF REFERENCES

Ahn, Hyo-Sung, Kevin L. Moore and YangQuan Chen. 2007. *Iterative learning control : robustness and monotonic convergence for interval systems*. Coll. « Communications and control engineering ». London: Springer, xviii, 230 p. p.

Arimoto, S, S Kawamura and F Miyazaki. 1986. « Convergence, stability and robustness of learning control schemes for robot manipulators ». In *Proceedings of the International Symposium on Robot Manipulators on Recent trends in robotics: modeling, control and education*. (Albuquerque, New Mexico, USA), p. 307-316. 23632: Elsevier North-Holland, Inc.

Arimoto, S. 1985. « Mathematical theory of learning with applications to robot control ». *Proceedings of 4th Yale Workshop on Applications of Adaptive Systems*, p. 379-388.

Arimoto, Suguru, Sadao Kawamura and Fumio Miyazaki. 1984. « Bettering operation of Robots by learning ». *Journal of Robotic Systems,* vol. 1, n° 2, p. 123-140.

Bien, Z., and K.M. Huh. 1989. « Higher-order iterative learning control algorithm ». *IEE Proceedings D (Control Theory and Applications)*. Vol. 136, n° 3, p. 105-112. < http://digital-library.theiet.org/content/journals/10.1049/ip-d.1989.0016 >.

Bondi, P., G. Casalino and L. Gambardella. 1988. « On the iterative learning control theory for robotic manipulators ». *Robotics and Automation, IEEE Journal of,* vol. 4, n° 1, p. 14-22.

Bonev, Ilian. 2013. « Mecademic ». < www.mecademic.com >.

Craig, J. 1984. « Adaptive control of manipulators through repeated trials ». In *Proceedings of 1984 American Control Conference*. (San Diego, California), p. 1566-1572.

Craig, John J. 2005. *Introduction to robotics: mechanics and control*, 3. Pearson Prentice Hall Upper Saddle River.

Edwards, J. B. 1974. « Stability problems in the control of multipass processes ». *Electrical Engineers, Proceedings of the Institution of,* vol. 121, n° 11, p. 1425-1432.

Gauthier, Guy. 2008. « Terminal iterative learning for cycle-to-cycle control of industrial processes ». In /z-wcorg/. http://worldcat.org.

Gu, You-Liang, and NanK Loh. 1989. « Learning control in robotic systems ». *Journal of Intelligent and Robotic Systems,* vol. 2, n° 2-3, p. 297-305.

Harokopos, E. 1986. « Optimal learning control of mechanical manipulators in repetitive motions ». In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*. (Apr 1986) Vol. 3, p. 396-401.

Hauser, J. 1987. « Learning control for a class of nonlinear systems ». In *Proceedings of 26th Conference on Decision and Control*. (Los Angeles, California), p. 859-860.

Heinzinger, Greg, D. Fenwick, B. Paden and F. Miyazaki. 1989. « Robust leaning control ». In *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on*. (13-15 Dec 1989), p. 436-440 vol.1.

Hideg, L. M., and R. P. Judd. 1988. « Frequency domain analysis of learning systems ». In *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*. (7-9 Dec 1988), p. 586-591 vol.1.

Jagannathan, S. 2001. « Control of a multiple link robot arm at very high speeds for an industrial application ». In *American Control Conference, 2001. Proceedings of the 2001*. Vol. 2, p. 793-798. IEEE.

Joubair, Ahmed. 2012. « Contribution à l'amélioration de la précision absolue des robots parallèles ». École de technologie supérieure.

Kato, T. Mita and E. 1985. « Iterative control and its application to motion control of robot arm- adirect approach to servo-problems ». *Proceedings of 24th Conference on Decision and Control*, p. 1393.

M.Yamakita, K. Furuta and. 1987. « The design of a learning control system for multivariable systems ». *Proceedings of IEEE International Symoisium on Intelligent Control*, p. 371-376.

McIntyre, C. Atkeson and J. 1986. « Robot trajectory learning through practice ». *Proceedings of IEEE Conference on Robotics and Automation*.

Merlet, J. P. 2006. *Parallel robots*, 2nd. Coll. « Solid mechanics and its applications », 74. Dordrecht ; Boston, MA: Kluwer Academic Publishers, xiv, 355 p. p.

Messner, W., R. Horowitz, W. W. Kao and Michael Boals. 1991. « A new adaptive learning rule ». *Automatic Control, IEEE Transactions on,* vol. 36, n^{o} 2, p. 188-197.

Moore, Kevin L. 1993. *Iterative learning control for deterministic systems*. Coll. « Advances in industrial control ». London ; New York: Springer-Verlag, xvi, 152 p. p.

Murray, Richard M, Zexiang Li, S Shankar Sastry and S Shankara Sastry. 1994. *A mathematical introduction to robotic manipulation*. CRC press.

Sang-Rok, Oh, Bien Zeungnam and Suh Il Hong. 1988. « An iterative learning control method with application to robot manipulators ». *Robotics and Automation, IEEE Journal of,* vol. 4, n° 5, p. 508-514.

Sugie, Toshiharu, and Toshiro Ono. 1991. « An iterative learning control law for dynamical systems ». *Automatica,* vol. 27, n° 4, p. 729-732.

Tsai, Lung-Wen. 1999. *Robot analysis : the mechanics of serial and parallel manipulators*. New York: Wiley, xiii, 505 p. p.

Uchiyama, M. 1978. « Formation of high speed motion pattern of mechanical arm by trial ». *Transactions of the Society of Instrumentation and Control Engineers,* vol. 19, p. 706-712.

Yamakita, M., and K. Furuta. 1991. « Iterative Generation of Virtual Reference for a Manipulator ». *Robotica,* vol. 9, n° 01, p. 71-80.

Yamano, M. Togai and O. 1985. « Analysis and design of an optimal learning control system: a discrete system approach ». In *Proceedings of 24th Conference on Decision and Control*. (Florida), p. 1399-1404.

Yang Quan, Chen, and K. L. Moore. 2002. « An optimal design of PD-type iterative learning control with monotonic convergence ». In *Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium on*. (2002), p. 55-60.

Yeon, Je Sung, Eui Jin Kim, Sang-Hun Lee, Jong Hyeon Park and Jong-Sung Hur. 2005. « Development of Inverse Dynamic Controller for Industrial robots with HyRoHILS system ». In *Proceedings of International Conference on Control, Automation and Systems (ICCAS 2005), Korea, Kyeonggi-Do*.