BALL SCREW LINEAR ACTUATOR CONTROL AND IMPLEMENTATION BY APPLYING LUGRE FRICTION MODEL

By

Mingpo Jia, Btech

A Thesis

Submitted to the School of Graduate Studies

In Partial Fulfillment of the Requirements

For the Degree of

Master of Applied Science

McMaster University

Master of Applied Science (2018)	Master University
(Mechanical Engineering)	Hamilton, Ontario Canada

TITLE:	Ball Screw Linear Actuator Control and Implementation by Applying LuGre Friction Model
AUTHOR	Mingpo Jia
SUPERVISOR:	Dr.Fengjun Yan
NUMBER OF PAGES	182

Abstract

The linear actuator is widely used in the industrial and aerospace arenas. The application of the linear actuator varies. The ball screw type linear actuator or ball screw system is one design. The ball screw is a mechanical system that converts rotation motion into a linear motion. The ball screw linear actuator, compared with other linear actuators, has better efficiency, higher speed, less noise, and higher load capacity. Ball screw linear actuators are used in a number of areas, such as coordinated measuring machines, 3D printers, and aerospace actuators.

In this research, the industrial sponsor provided a ball screw linear actuator, and they required its accuracy to be improved. The linear actuator suffers from an accuracy problem due to various reasons. One of the major problems is nonlinear friction, which makes it difficult to estimate using the simple friction model. In this thesis, a LuGre friction model is introduced and applied to the ball screw system. The sponsor's ball screw system includes the ball screw sliding table, AC servo drive, AC servo motor, and a linear encoder sensor. The hardware control system for the ball screw system needs to be built. Therefore, this thesis describes how a custom ball screw control system was built.

The control hardware ball screw system includes a microcontroller and a custom-made digital-to-analog converter. The linear encoder position sensor's reading methods were tested and implemented in the microcontroller. A custom digital-to-analog converter was made and tested.

The control algorithms based on the LuGre friction compensator are discussed and were simulated in the Matlab Simulink environment. Then, the physical implementation of the control algorithms on ball screw system hardware were made. Finally, a new proposed control method based on the LuGre friction model performed best in terms of accuracy consistence and tracking compare to the other mentioned controllers.

Acknowledgements

I would like to special thanks and express my gratitude to my supervisor, Dr, Fengjun Yan and visiting scholar, Dr. Qi Zhang whose guidance, encouragement, and support through this project. Dr. Yan help me a lot for selecting the research topic and give me an opportunity to be a McMaster Mechanical Engineer student. I also would like thanks, Dr. Zhang who help me a lot on hardware construction, he gives me lots of suggestion and help solve several critical problems on the building the electric circuit.

I also would like to thank Mr. Cam Fisher to borrow the MARC laboratory to perform the experimental.

I also would also like to thank all my colleagues from PCL and CMHT labs for helping me support me and encourage me during this research.

Finally, I would like to thank my parent and my loved one 1111 who backup me, encouraged me and gave me love during the entire research study.

TABLE OF CONTENTS

List of Figures

1. CHAPTER INTRODUCTION

1.1 BACKGROUND

The linear actuator has been primarily used in the industries for equipment, such as coordinated measure machines, 3D printers, and aerospace actuators. Therefore, the accuracy of such a system became extremely crucial in the engineering area. One of the obstacles to increasing the accuracy performance is nonlinear friction. Therefore, this research aims to identify the occurrence of nonlinear friction and used it to improve accuracy of the prebuilding of a linear actuator system.

The linear actuator system in this research includes a ball screw system and an AC servo propulsion system. In particular, the research focuses on friction estimation, system identification, and the control of the system.

1.1.1 Ball screw system overview and application

The ball screw system is a mechanism that transfers rotation movement to translational motion. The primary purpose of such system is to transfer the motor torque to translational force in order to push or to pull objects with the goal of performing a desired task, such as position tracking or velocity tracking. There are many similar systems besides the ball system screw design that function as rotational to translational converters, such as a lead screw and a roller screw.

The lead screw is typical common translation converter. Using a center thread shaft, the lead screw system mechanism translates the force to the outer screw shaft which has a matching set of thread. The thread is in the shape of trapezoidal teeth. Due to the large contact surface on the thread of the nut and screw, the lead screw has significant friction energy loss compare to the ball screw and the roller screw. The efficiency of the lead screw ranges from 20% to 40% [1], and therefore they are not suited for carrying a large load. The more significant friction force prevents backlash. However, they required more motor torque. The cost of the lead screw is lower than that of the roller screw and ball screw [2].

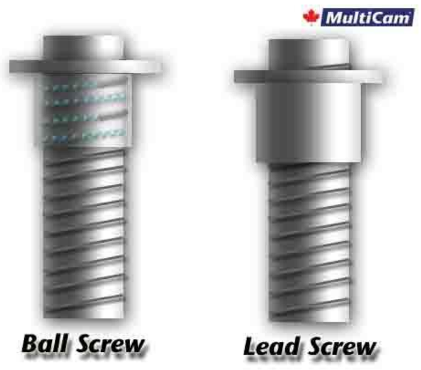


Figure 1.1.1 Ball screw vs. lead screw [2]

The roller screw—also called planetary roll screw—is a low friction precision linear screw actuator. The cost of the roller screw is usually more expensive than the lead and ball screws. They have a more complex mechanism than the ball screws systems. The roller screws are similar to the ball screw mechanism, in that it also comes with a screw shaft, a nut, and a planetary roller. The treads of the roller screw are triangular. The force transfers from rollers to the center of the screw, which has a matching set of threads. Because their contract surface is more substantial than that of the ball screw (see Figure 1.1.2), their lifetime is 10 to 15 times longer than a ball screw's lifetime [3]

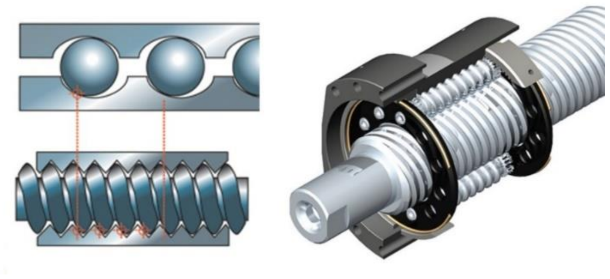


Figure 1.1.2 Roller screw vs. ball screw [3]

The ball screw is also a low friction precision linear screw actuator. The typical ball screw system contains balls, a nut, and a screw shaft. The screw shaft uses circular or ogival threads. Thread diameter allows balls to fit inside the grooves.

When the center threaded screw rotates, the balls are deflected by the deflector and force into the ball return system in the nut. The ball goes through the return system to the opposite end of the ball nut and exit from ball return grooves into the ball screw. The nut thread grooves continue recirculating in a closed circuit [4]. The ball return system allows no physical contact between the screw and the nut. Therefore, the friction only occurs in two areas, namely with the ball to the nut and the ball to the screw. The contact surface has decreased mainly due to the same contact area. The force is also distributed to many balls inside the system, the force allows a small load to each ball. By these means, a ball screw system could carry heavier weights, and it would require less motor torque compared to the lead screw system. [2]

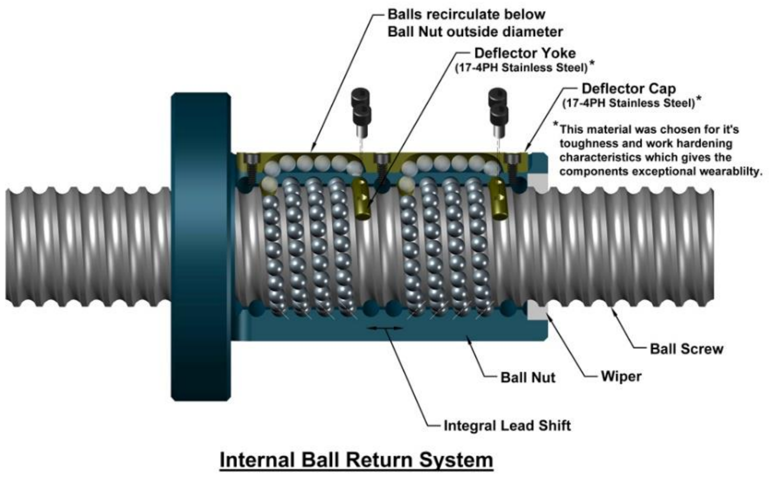

Balls recirculate below Ball Nut outside diameter
Deflector Yoke (17-4PH Stainless Steel)*
Deflector Cap (17-4PH Stainless Steel)*
*This material was chosen for it's toughness and work hardening characteristics which gives the components exceptional wearablilty.
Ball Screw
Ball Nut
Wiper
Integral Lead Shift
Internal Ball Return System

Figure 1.1.3 Internal system of a ball screw system [4]

The ball screw system typically has efficiencies that range from 70% to 85% [1]. The design of the ball screw is similar to a ball bearing, and such a design would therefore offer minor advantages, such as having a high speed and less noise. Both the ball screw and the roller screw use a rolling motion rather than a sliding movement of the lead screw.

The ball screw has a disadvantage: the motor turn briefly before the movement begins, which is an action also known as backlash. The backlash occurs due to the free axial and radial lash that is caused by low internal friction. This is an axial free motion from the ball screw to the ball nut [5]. The conventional way to eliminate the backlash is to add a preload to the balls; this preload would remove the gap between the ball screw and ball nut.

The ball screw applications have a broad range in usage. The high precision ball screw is widely used in commercial and military aircrafts. The major components that use the ball screw mechanism system include wing flaps, propeller pitch controls, engine thrust reversers, and horizontal stabilizers. Other components that use the ball screw mechanism are the main landing gear, the variable engine inlet, and the exhaust-nozzle. [6]

Aside from aerospace application, the ball screw system has also been widely used in the manufacturing industries. In the manufacturing area, the ball screw has been applied in milling-machine tables, robotics arms, and the coordinate measurement machine(CMM).

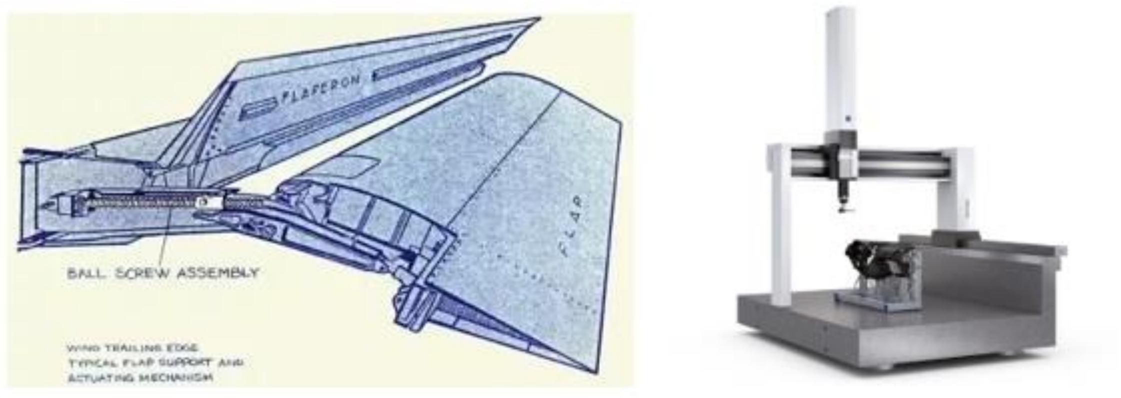

FLAPERON
FLAP
BALL SCREW ASSEMBLY
WING TRAILING EDGE TYPICAL FLAP SUPPORT AND ACTUATING MECHANISM

Figure 1.1.4 Ball-screw actuator for the flaps and the coordinate measurement machine [7] [8]

1.1.2 AC servo system overview

The servo system is the actuator part of the linear actuator system. It allows the precision control of a linear or angular position, velocity, and even sometimes the acceleration of the device. The advantage of the AC servo system is the feedback from its sensors, which allow them to become a closed loop system; in this way, this system has more accuracy than a steppe motor. The AC servo system includes three significant components—an AC servomotor, an encoder and a servo controller.

There are two types of servomotors—the AC servomotor and the DC servomotor. A DC servomotor uses a permanent magnet DC motor, and a DC servomotor is commonly used for simple applications due to it needing frequent maintenance. The AC servomotor uses AC power instead of DC power, and the AC servo motor is based on squirrel cage induction motors with two or three phases. The AC servomotor consists of a stator and a rotor. In Figure 1.1.5, the stator typically has two windings that are uniformly distributed by a 90-degree angle. One winding is supplied by the constant AC voltage AC. The other winding is the control winding, which is excited by a variable control voltage that comes from a servo amplifier. The control voltage should be 90 degrees out of phase concerning the reference winding voltage. The control winding’s flux is also 90 degrees out of phase from the reference winding’s flux. The air gap resultant flux sweeps over the rotor, which induces an electric magnetic field to the rotor. The rotor is influenced by the electric magnetic field and then produces its rotor flux. The rotating electric magnetic field interacts with the rotor flux; this interaction provides a torque on the rotor, and the motor starts rotating. The polarity of control voltage winding determines the direction of rotation. The torque and angular speed are directly related to the control voltage because of the reference voltage constant [9].

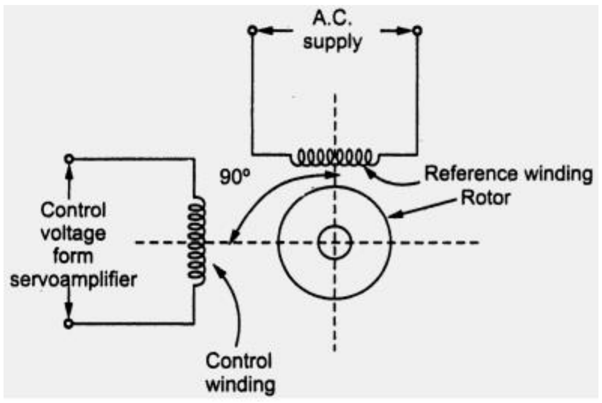


Figure 1.1.5 Stator of an AC servomotor [9]

1.1.3 Friction overview

One of the critical factors in this research is in estimating and identifying the friction that affects the ball screw system dynamic. Friction is everywhere in this world dimension; according to Newton’s law of motion, an object should either remain at rest or continue to move at a constant speed unless a force is acted upon it. By these means, if friction force does not exist, it would be impossible to walk a single step. The friction force is defined as a tangential reaction force between two surfaces in contact—this is referred to as dry friction.

Indeed, there are other types of friction, such as fluid friction, internal friction, and lubricated friction. In this research, dry friction is mainly discussed. Friction is typically categorized into the two regimes—namely static friction and kinetic friction. In the reality, rigid bodies cannot have a perfectly smooth surface; Therefore, the contact surface of two rigid bodies can be compared to two elastics bristles that in contact together. When a tangential force is applied, the bristles begin to deflect like springs and damper. When the displacement between the two bristles increases to a particular value, the bristles break. The areas before bristles break can be described as the static friction regions, while the areas after the break are the kinetic friction regions. [10]

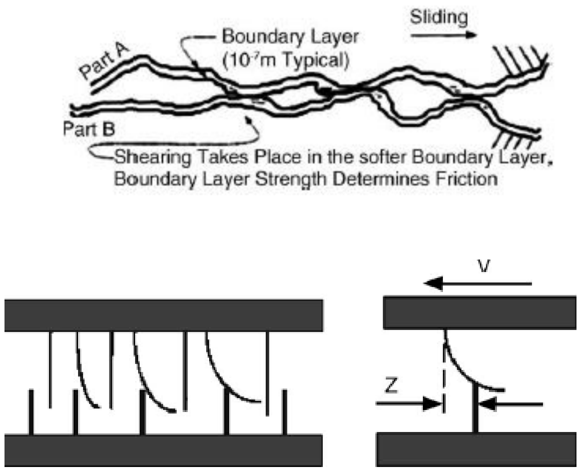

Sliding
Boundary Layer
(10⁻⁷m Typical)
Part A
Part B
Shearing Takes Place in the softer Boundary Layer,
Boundary Layer Strength Determines Friction
v
z

Figure 1.1.6 Contact surface to bristles model [10]

The most well-known friction model is the Coulomb friction model. The Coulomb friction model is a simplified frictional model with a constant magnitude force that acts in the opposite direction of the motion, as shown in Figure 1.1.7. This friction model is represented in Equation 1.1, where F_f represents the friction force, F_c is the constant coulomb friction force, F_N is the normal force, μ is the frictional factor, and v stands for velocity. Because the Coulomb friction has a constant magnitude force, and the friction estimation error of this method can result in an error up to 20%. The major issues of Coulomb friction is, that it cannot handle zero velocity. [11]

$$\begin{aligned} v(t) \neq 0 &: F_f(t) = -F_c \operatorname{sgn}(v(t)) \\ F_c &= \mu F_N \end{aligned} \tag{1.1}$$

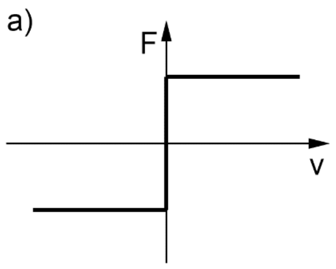


Figure 1.1.7 Coulomb friction [11]

An improvement fiction model based on Coulomb friction has been introduced, which is called the viscous friction. The viscous friction is a friction force that is proportional to the sliding velocity. Instead of a constant magnitude force, viscous friction becomes a variable magnitude force that is related to velocity. Equation (1.2) presents the viscous friction. F_v is the constant viscous friction parameter. The characteristics of vicious friction can be seen in Figure 1.1.8.

$$v(t) \neq 0 : F_f(t) = -F_v v(t) \tag{1.2}$$

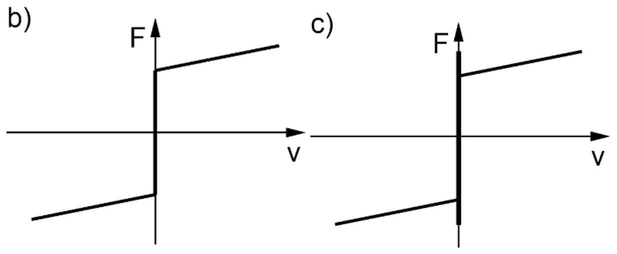


Figure 1.1.8 Viscous friction combined with Coulomb friction [11](Right)

Figure 1.1.9 Viscous friction combined with Coulomb friction and static friction [11](Left)

Both friction models do not fully capture the static part of the friction. The static friction requires an initial force needed to be more extensive than the initial viscous friction. Therefore, the static friction model combines the viscous friction and the coulomb friction present in Equation (1.3), where F_v is the viscous friction coefficients, F_s is the maximum static friction coefficient, and $u(t)$ is an external force applied to the system. The characteristic of viscous friction combined with Coulomb friction and static friction can be seen in Figure 1.1.9.

$$F_f(u(t),t) = \left\{ \begin{array}{cc} u(t) & (v(t)=0, u(t) < F_s) \\ F_s \, \mathrm{sgn}(u(t)) & (v(t) = 0, u(t) \geq F_s) \\ -F_v v(t) & (v(t) \neq 0) \end{array} \right\} \tag{1.3}$$

Based on , Figure 1.1.9, the combined model still has a discontinuity that occurs when static friction is transferred to kinetic friction. In reality, the discontinuity in Equation 1.3 is not possible.

In 1902, Stribeck experimented and discovered the phenomenon that describes the transition between the maximum static friction to Coulomb friction. The improved characteristic friction curve is called the Stribeck curve friction.

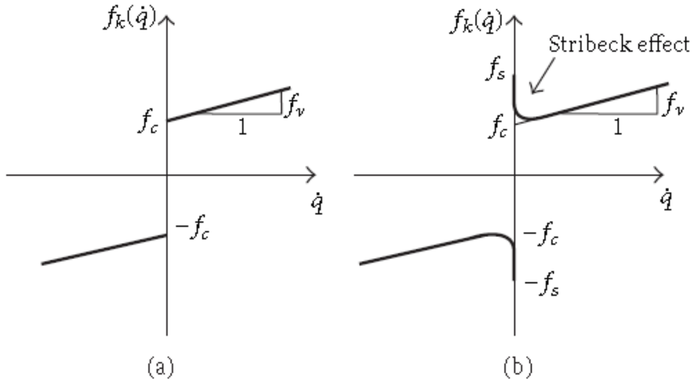


Figure 1.1.10 Comparison between (a) Coulomb friction and viscous friction with (b) Stribeck curve friction [12]

As seen in Figure 1.1.10, the discontinuity in (a) has been replaced by the exponential function known as Stribeck curve in (b). The equation that describes the Stribeck Curve is known as the Armstrong model [13], as seen in Equation (1.4). This model includes static friction, Coulomb friction, and viscous friction. It adds two extra terms—namely, the Stribeck velocity coefficient $v_{s.}$ and the fitting parameter δ_v [12].

$$F_f(v) = F_c + (F_s - F_c)e^{-(\frac{v}{v_s})^{\delta_v}} + F_v v \tag{1.4}$$

The above friction model of static friction, coulomb friction, viscous friction, and Stribeck friction are in the category of the static memoryless model. All the models above have discontinuity when crossing zero velocity. The dynamic friction model attempts to provide a solution to this issue. In the literature, two well-known dynamic friction models are the Dahl model and the LuGre model.

In 1968, Dahl introduced a new friction model which treats two rough contact surfaces of a rigid body as the bristles of two brushes (see Figure 1.1.6). Instead of using only velocity *v*, the Dhal model introduce a term *z* that also affects friction behavior. The *z* term presents the average deflection of the bristles. The equation of the Dahl model can be represented by Equation 1.5 [12]. The way that the Dahl model defines *z* variable depends on velocity *v*, and the static friction ($v = 0$) should be bound in the range of [$-F_c$, F_c]. The term σ_0 is a stiffness coefficient. The Dahl model captures the behavior of the presiding phenomenon in the static friction region, but the model does not include the Stribeck effect (see Figure 1.1.11).

$$\begin{aligned} f(v, z) &= \sigma_0 z + F_v v \\ \dot{z} &= -\frac{\sigma_0 |v|}{F_c} z + v \end{aligned} \tag{1.5}$$

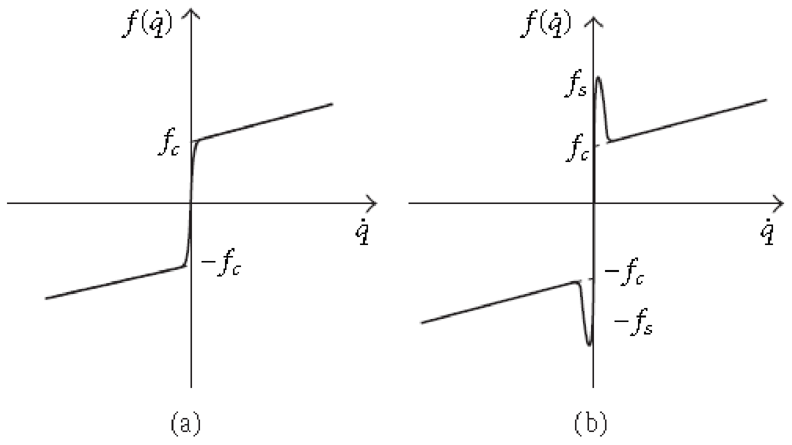


Figure 1.1.11 Friction torque versus velocity for dynamic models: (a) Dahl model; (b) LuGre model [12]

To improve the Dahl model, researchers from Sweden and France developed a new friction model in 1995 called the LuGre Model. The LuGre Model is one of the most complete friction models. The LuGre model combines all the friction models discussed above. The LuGre model includes the static friction, Coulomb friction, viscous friction, Stribeck friction and Dahl friction model. Just like the Dahl model, the LuGre model is a function of average bristle deflection *z*, and velocity *v*. Equation 1.6 shows the complete LuGre Model. The term σ_1 represents the damping coefficient, and σ_2 is the same as F_v but with different notation. The rest of the parameter is the same for the Stribeck friction and for the Dahl friction parameters.

$$F_f(v,z) = \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 v \tag{1.6}$$

$$\dot{z} = v - \frac{\sigma_0 |v|}{g(v)} z \tag{1.7}$$

$$g(v) = F_c + (F_s - F_c) e^{-|\frac{v}{v_s}|^{\delta_s}} \tag{1.8}$$

The g(v) function extends the Dahl model to describe the Stribeck effect with the same constraint as $F_c \leq g(v) \leq F_s$. In Figure 1.1.11, (b) represents the LuGre model shows the improvement of adding the Stribeck curve effect to the Dahl model. Figure 1.1.12 presents a detailed explanation on how the LuGre phenomenon occurs in a physical application. In this thesis, LuGre friction is the primary focus of the research; the remainder of the thesis also investigates how to apply the friction in a physical ball screw system.

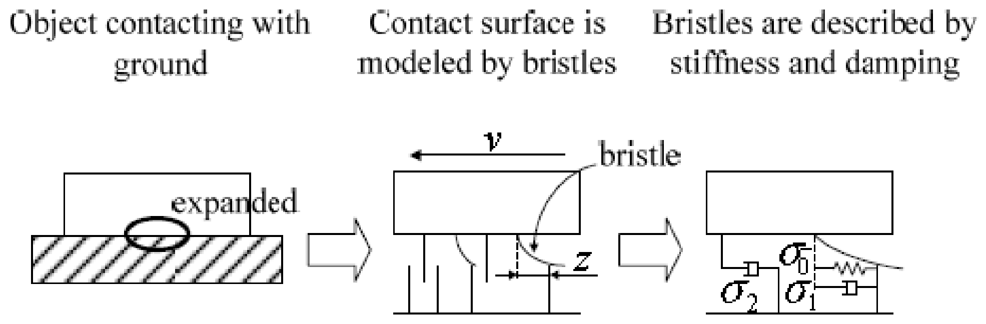

Object contacting with ground
Contact surface is modeled by bristles
Bristles are described by stiffness and damping
expanded
v
bristle
z
σ_0
σ_1
σ_2

Figure 1.1.12 Schematic explanation of the LuGre model [14]

1.2 THESIS CONTRIBUTIONS

The purpose of this thesis is to complete and improve the current prebuilding of the ball screw system from the industrial partner. The hardware of a ball screw system is prebuilt, and it includes the ball screw sliding table, an AC servo system, and a linear encoder. As contribution to the current body of research, this thesis has the following contributions have been made:

1. To design and construct a digital to analog converter circuit, then select and implement the proper microcontroller(Appendix7.5). Finally, total cost of control system hardware is more economical than other similar function product in the market.
2. To perform a LuGre friction and ball screw system identification experiment and follow by parameters identification
3. To implement a PID controller, position PD with friction estimation observer controller, and position PD with friction estimation observer with disturbance observer controller on Simulink model and compare result, then implement the above simulated controllers into an actual ball screw system
4. To implement a new multiple stage controller into physical ball screw system and compare the results to existing controllers

1.3 THESIS LAYOUT

This thesis is composed of 6 chapters. Chapter 1 provided background on the current research topic and on other linear actuator mechanism devices compared with the ball screw mechanism. The chapter also includes an introduction on the AC servo system as well as the history and background on a recent friction model.

Chapter 2 is focusing on literature review on existing LuGre friction and Ball screw system research. The literature review has been separated into 2 sections, the system modeling section and identification section and control section. In the first section, the ball screw system and LuGre friction modeling has been discussed and then following by parameter estimation method.

Chapter 3 shows how the ball screw controller hardware implemented. The encoder decoding method has been test and implemented. The design and building process of the digital to analog converter has been introduce in this chapter as well. Finally, the entire ball screw control system has been present. (The introduction on provided AC servo system and pre-building Ball screw sliding table could be found in appendix)

Chapter 4 present the system identification method and System plant model including the LuGre friction model. The system plant model and LuGre friction model has been present first. Then following system identification experiment. Finally, based on the experimental data, the parameter identification has been proceeded.

Chapter 5 present the ball screw system and LuGre friction control simulation in Matlab Simulink. Base on the LuGre friction parameter and Ball screw system parameter that have been identified in Chapter 4, variety controllers has been test in the simulation.

The second of chapter 5, applied the simulated controller into the physical ball screw system by using the Arduino code. Then base on existing controller, a novel controller has been introduced and test. Finally, the result of between the different controllers and be present and discussed.

In the Chapter 6, The conclusion has been made which sum up all the work that have been contributed. The result of new controller compared the thesis goal has been discussed. Finally, the potentiation improvement has been concluded in the future work section.

2 CHAPTER LITERATURE REVIEW

2.1 SYSTEM MODELING AND IDENTIFICATION

This section focuses on the literature that describes ball screw system and LuGre friction's mathematical model

The ball screw system can be modelled as a rotation torque translation on a linear force process. Several studies have examined the ball screw system model, and some similarities can be found in these studies. The LuGre friction used in the studies follows the fundamentals in Equations (1.6) to (1.8); the only differences are the notation.

The primary ball screw schematic is summarized in Figure 2.1.1. The servomotor produces the torque τ_2 and has the rotational angle θ_2 going through the coupling or gear, and the torque is translated to the ball screw nut. From the ball screw nut, the rotational torque is then transferred to the linear force, which moves the table and creates the displacement $x_{1.}$ The moving of the table creates a counter friction force, which is caused by the guideway of the ball screw nut interface and any other moving elements in the system; these can be summed up as the total friction F_f.

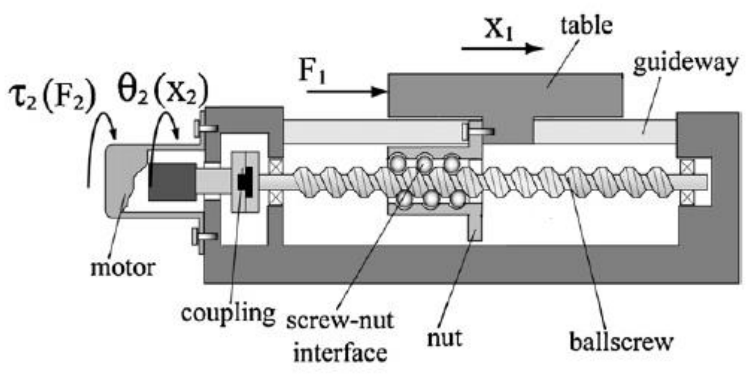


Figure 2.1.1 General ball screw system schematic [15]

2.1.1 Ball screw system and LuGre friction modeling

A study of the ball screw system [14] conducted by Tokyo Denki university presented the ball screw system as a second order system with linear and nonlinear parts. The system can be seen in Equation 2.1.

$$\mathrm{m}\ddot{\mathrm{p}} = \mathrm{K}_t \cdot K_a \cdot u - F - \sigma_2 v \tag{2.1}$$

In the equation, *m* denotes the mass of the system, *p* is the position of the mass, K_t and K_a is the torque constant of the AC servomotor and the torque/force conversion gain. *F* is a nonlinear friction force, *v* is the velocity of the mass, and σ_2 is the viscous friction coefficient. Equation 2.1 follows Newton's second law of motion of *F* = *ma*, and it separates friction as linear ($\sigma_2 v$) part and nonlinear (*F*) part, where *F* can be described the same way as found in Equation 1.6 with only the nonlinear element as follows:

$$F = \sigma_0 \mathrm{z} + \sigma_1 \dot{z} \tag{2.2}$$

The term *z* is the deformation of the bristles, and the derivation of *z* can be expressed as seen in Equation 1.7. However, the difference here is with *g*(*v*). In [14], the *g*(*v*) was expressed as

$$g(v) = F_c + (F_s - F_c)e^{-|\frac{v}{v_s}|} \tag{2.3}$$

The δ_v in Equation 2.3 has been chosen to be 1. In several studies [13] [16], there have been explanations given regarding the term δ_v, and different range settings for the term have been given by different studies. In [13], it has been suggested that $\delta_v = 2$, and in the study done by Bo and Pavelescu in [16], they suggested to have δ_v in the range from 0.5 to 1. In this case, they [14] followed [16] and selected δ_v to be 1.

In [17], the researchers proposed a similar model to [14] which can be seen in Equation 2.4 below. The main difference between [17] and [14] is [17] add the motor inertia *J* to the acceleration term. Also, in [17] instead of using *u* as voltage, the input directly becomes torqued *T*. *R* denotes the radian to displacement conversion gain. F_n is the nonlinear friction force, and *M* is the mass of the system. In Equation (2.4), torque is used for balancing Newton's second law of motion while Equation 2.1 used force. However, Equation 2.4 and Equation 2.1 have the following the same structure, which is acceleration times a constant equal to the control input minus the linear and nonlinear friction force or torque.

$$(J + R^2 M)\ddot{\theta} = T - R^2 \sigma_2 \dot{\theta} - RF_n \tag{2.4}$$

In [18], the study from Chosun University showed a more complex ball screw system model than [14] and [17]. The above studies in [14] and [17] follows Figure 2.1.1, which is a simplified model of the ball screw system; these two studies assume the ball screw nut and the table to be one single body. In the model presented by [18], the ball screw and table as considered to be two bodies with two masses. The equation presented by [18] is given as Equation (2.5) here:

$$(J_{rot} + (m_1 + m_2) l_p^{\,2})(\frac{\ddot{x}_2}{l_p}) + F(\cdot) = u(t) \tag{2.5}$$

In the equation, J_{rot} is the total rotational inertia of the ball screw and motor, while l_p is the conversation gain from the displacement to the radians. The m_1 and m_2 are the slide table mass and ball screw nut mass respectively; x_2 is sliding table moving displacement. Instead of separating the linear friction and nonlinear friction as done in [14] and [17], the researchers in [18] put them together as one single friction torque term *F*(.). The *F*(.) is the friction torque that is created by nut interfaces, motor brushes, and support bearing; however, in the later experiment, the researchers discovered that the friction torque dominates in the system. There is high stiffness in the motor torque transmission mechanism. Therefore, Equation 2.5 has been reduced and simplified to Equation 2.6:

$$J\ddot{\theta} + F(\cdot) = u(t) \tag{2.6}$$

In the equation, *J* is the inertia of the ball screw, motor, and rotational inertia of the linear masses, while $\ddot{\theta}$ is the angular acceleration of the ball screw.

The LuGre friction model used in this paper is very similar as the ones in Equations 1.6 and 1.7; the only difference is in *g*(*v*) when compared to Equation 1.8. The *g*(*v*) in [18] is given in Equation 2.11:

$$g(v) = F_c + (F_s - F_c)e^{-\left|\frac{v}{v_s}\right|^2} \tag{2.7}$$

Compare to Equation 1.8, Equation 2.7 assigned δ_v to be 2, which follows the research done by Bo and Pavelescu.

In [19] presented by Tianjin University, the researchers presented an even more complicated model than [18]. The study traded the system as a single mass servo system. In a real system, the mechanic transmission part forms deformation to an extent. The rigidity is needed for describing the elastic deformation of transmission. Therefore, the rigidity in the transmission mechanism is needed in the dynamic equation of the system. The system structure of this paper is presented in Figure 2.1.2.

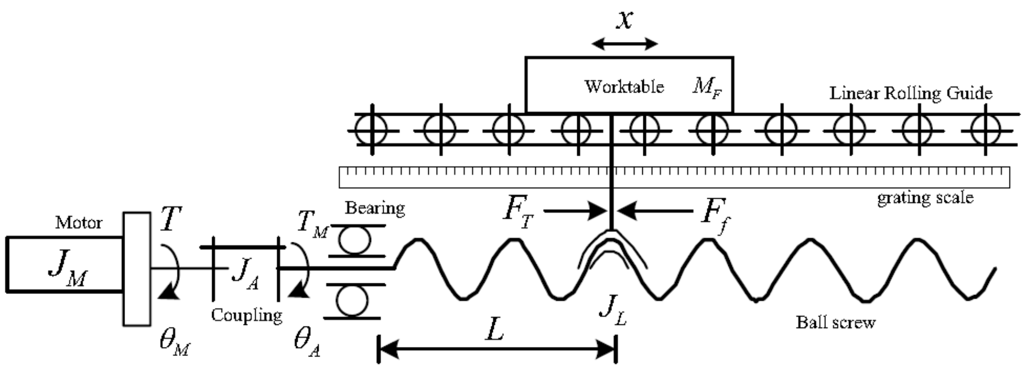


Figure 2.1.2 Ball screw system schematic with consideration of elastic deformation [19]

In the Figure 2.1.2, instead of using a sum up inertia as in [18], the system breaks the moment of inertia into three parts, namely J_m as the inertia of the motor, J_A as the inertia of coupling, and J_L as the inertia of ball screw. In addition, M_F is the working table mass, θ_M is the rotation angles of the motor, and θ_A is the rotation angles of the coupling. The x is the displacement of the worktable. T_M is the torque that is transferred from the motor to the screw through the coupling, and T is the motor torque. F_T is the axial load, and F_f is the total friction of the system. Lastly, L is the distance from the nut to the supporting end. The assumption for this paper is that the torsional deformation of the ball screw on the linear displacement of the work table is ignored. The main influences of the elastic deformation are the flexible coupling torsional rigidity K_1 and screw axial rigidity K_2. Based on the above assumption, the motion equation is expressed in the following equations from Equation 2.8 to Equation 2.12.

$$J_M \ddot{\theta}_M = T - K_1(\theta_M - \theta_A) \tag{2.8}$$

$$T_M = K_1(\theta_M - \theta_A) \tag{2.9}$$

$$(J_A + J_L)\ddot{\theta}_A = T_M - F_T \cdot \frac{K_s}{\eta} \tag{2.10}$$

$$F_T = K_2(\theta_A \cdot \frac{K_s}{\eta} - x) \tag{2.11}$$

$$M_F \ddot{x} = F_T - F_f \tag{2.12}$$

In the equations, η is the drive efficiency of the ball screw, and K_s is the radius to linear displacement gain. The LuGre friction model of F_f is used in this model, and it is the same as the one in Equations 1.6, 1.7, and 2.7.

By now, the above ball screw model is the most detailed of the dynamics of the ball screw systems compared with the studies of [19], [17], [14] and [18]. The main advantage of this system model is that it can capture the deformation of the mechanic transmission part. However, due to the parameters being increased, the system identification for Equations 2.8 to 2.12 will be a challenge if the number of feedback sensors are limited—such as if only the table displacement sensor is provided. The LuGre friction model presented in the above studies are all similar; the only difference is the selection of $\delta_{v.}$.

Aside from the classic LuGre friction model, there is also another dynamic friction model that has been investigated. In a study by Sánchez-Mazuca and Campa [12], an improved static friction part of the LuGre friction model was introduced. Both the classic LuGre model and Stribeck friction model use a constant F_s as a maximum static friction coefficient, which is also called a breakaway torque. In their paper, they suggest the maximum static friction coefficient is related to the linear increase input torque slope *m*. If the m is a smaller value, the higher of F_s and vice versa. This relationship has been seen by [20]. To add this characteristic to LuGre model, [12] present an improved LuGre Friction model as Equation 2.13

$$\begin{aligned} F_f &= \sigma_0 z + \sigma_1 z + \sigma_2 v \\ \dot{z} &= v - \sigma_0 \frac{|v|}{g(v, \dot{\tau})} z \end{aligned} \tag{2.13}$$

In this equation, $g(v, \dot{\tau})$ is the main difference found compared to the classic LuGre model in Equations (1.6) to (1.8). In $g(v, \dot{\tau})$, one more variable, the rate of change $\dot{\tau}$, was added to the applied torque, as seen in Equation 2.14.

$$g(v, \dot{\tau}) = F_c + (F_s + F_c) e^{-\left(\left|\frac{\dot{\tau}}{t_s}\right|^{\delta_v} + \left|\frac{v}{v_s}\right|^{\delta_v}\right)} \tag{2.14}$$

In this equation, t_s is the torque rate coefficient. The exponential function in Equation 2.14 added the term $\dot{\tau}/t_s$ which describes the break torque variation in relation to the applied torque rate of change.

The study of the new LuGre friction in [12] provided a more detail explanation on how the breakaway force or torque may affect the performance of the friction model. From their results, the friction estimation accuracy is increased. The [12] is a new improvement of the LuGre friction that has not been as widely used or verified by other researchers compared with the classic LuGre friction model. The study also added two more terms to the classic LuGre friction. [21] and [22] both suggested the temperature factor will affect the friction and [23] suggested the humidity will also affect the friction as well. However, this addition may create issues in the parameter estimation stage, and therefore this research focuses instead on applying the classic LuGre friction model.

2.1.2 Ball screw system and the LuGre friction parameter estimation

The ball screw system plus the LuGre friction parameter identification and estimation are essential parts of this research because they determined how accurate the simulation result is to the real physical experiment. In the LuGre model based on Equations 1.6 to 1.8, there are a total eight parameters that are needed to be identified. These are presented below in Table 2.1 The LuGre friction parameter

σ_0	Bristle stiffness coefficient
σ_1	Bristle damping coefficient
σ_3	Viscous coefficient
F_c	Coulomb friction coefficient
F_s	Maximum static friction coefficient
v_s	Stribeck velocity
δ_v	Fitting parameter

Table 2.1 The LuGre friction parameter

The LuGre friction model can be separated into three major parts—the static friction, the Stribeck friction, and viscous friction. Most of the studies used two or three experiments to identify the LuGre friction. The first experiment is used for identifying the static friction and the Coulomb friction; the second one is for defining the Stribeck curve parameter, and the third is for identifying σ_0 and σ_1

In [17], researchers first defined the maximum static friction F_s by recording the input torque when the ball screw system starts to move, and then the Coulomb friction coefficient F_c is the input torque when the table stops. The identified Stribeck velocity v_s is obtained from the Stribeck curve. However, the friction torque is not easily measurable using Newton's first law of motion at a constant velocity and input torque being equal to the friction. Therefore, the researchers built a PI speed controller to obtain a constant speed and to measure the input torque. They chose several velocities for the reference speed and they performed the PI speed experiment on each speed. The Stribeck velocity is defined as the velocity when the friction is at minimum, and σ_2 is the slope of the curve when velocity reaches a linear relationship with friction. To identify σ_0 and $\sigma_{1,}$ the researchers applied a sinusoidal wave input to the system, and they created a plot of the relation between the input to the table position which is shown in Figure 2.12. The σ_1 is the slope of the top right corner to the lower left corner of plot. Finally, σ_0 can be obtained with parameters that identified the above and contained the experiment results and simulation results.

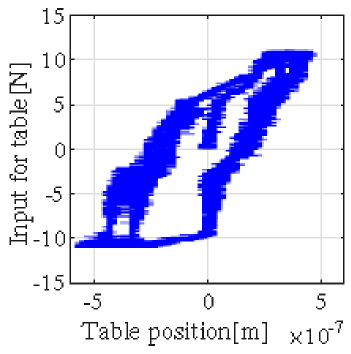


Figure 2.1.3 Sine ware output position response [17]

In [18], researchers also proposed finding the Stribeck Curve friction parameter using the friction velocity plot. To obtain the plot, a PI speed controller has been applied in the same way as [17] but at a different velocity, and the corresponding control torque was recorded. At each velocity point, the control torque has been averaged due to the control torque signal being noisy. The Stribeck curve of the friction and the velocity plot have been obtained, as seen in Figure 2.1.4.

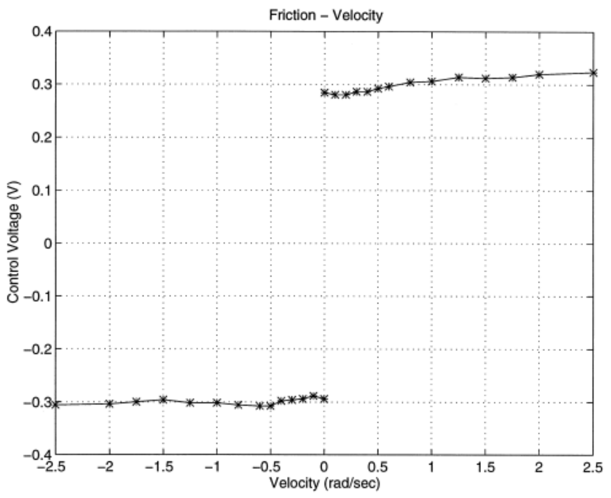


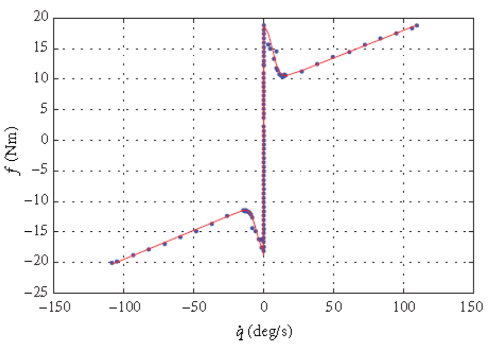
Figure 2.1.4 Stribeck friction and Viscous friction [18].

Based on Figure 2.1.4, the coulomb friction F_c dominated the static friction characteristics. The researchers assumed that the $g(v)$ in the LuGre Friction model would equal to F_c. The viscous friction coefficient σ_2 can be estimated by using Figure 2.1.4. Based on the same figure, the researchers assumed there is no motion in the stiction friction or static region. Therefore, $z \approx \theta$, and the LuGre friction in Equation 1.6 can be rewritten by replacing terms z and $\dot{z}$ into θ and $\dot{\theta}$. The plant Equation 2.6 can be rewritten to Equation 2.15. The parameter of σ_1 and σ_1 can be obtained by matching the physical system step response.

$$J\ddot{\theta} + (\sigma_1 + \sigma_2)\dot{\theta} + \sigma_0\dot{\theta} = u(t) \tag{2.15}$$

In [12], the researchers used a method similar to the one in [18] but they presented their method in a more detail way, including how to apply the hardware as well. First, the researchers intended to obtain the Stribeck curve, much like the studies in [17] and [18]. The basic principles are the same in obtaining the Stribeck curve by following Newton's second law of the motion which required the system to have a constant speed. In their work, the researchers mentioned that their servo system controller could be configured to the velocity controller mode. In this way, the input voltage is directly proportional to the reference velocity. Also, the torque output should be measured as well; the servo system could output current applied torque to the researcher. In this way, researchers do not need to build a PI speed controller as in [17] and [18]. They could directly use the prebuilt speed control mode in the servo system. In the ideal point of view, by using the speed controller, the motor should reach the reference speed. In the physical world, it is still required to measure the actual velocity since the actual velocity would have some oscillation around the reference velocity. Again, similar to studies in [17] and [18], the Stribeck curve needed to measure the input torque from a high velocity to a low velocity, and in each velocity point, it needed to average the actual velocity and average the input torque as well. The researchers obtained the Stribeck curve in Figure 2.1.5, where $\dot{q}$ is the velocity and f are friction torque.

The identification of the F_s, F_c, and σ_2 can be directly obtained from Figure 2.1.5. The way of getting the three parameters can be a reference to Figure 2.1.6. The maximum of the friction force at near zero velocity will be maximum static friction coefficient F_s. When it reaches to a high velocity and pass the Stribeck region, the slope of the curve becomes σ_2 or f_v, and when this line is extended to zero velocity, F_s is obtained. The rest of the parameter of v_s and δ_v can be estimated using known F_s, F_c, σ_2 and the Stribeck friction Equation (1.4) with the MATLAB curve fitting tool.



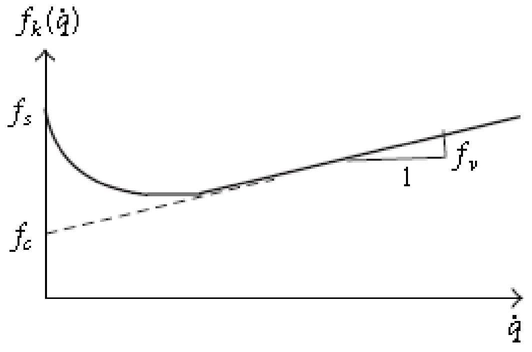


Figure 2.1.5 Stribeck friction from experiment data [12] (right)

Figure 2.1.6 Breakout of the Stribeck curve for obtaining F_c, F_s and σ_2 [12] (left)

To obtain the σ_0, a sinusoidal input was applied to the system, as done in [17]. The amplitude of the applied sine wave should be lower than F_c. In their case, the researchers chose the amplitude that is less than $F_c/2$. The displacement from the sensor that responds to this sine wave input is measured; the σ_0 is then written as Equation (2.16) where A is the input sinusoidal amplitude and θ_0 is the steady-state displacement amplitude.

$$\sigma_0 = \frac{A}{\theta_0} \tag{2.16}$$

To obtain σ_1, a step input of an amplitude lower than F_c is applied to the system, and the transient response is measured by applying the system response to the second order linear system of $s^2 + 2\xi\omega_n s + \omega_n^2$. The value of ζ can be estimated, and σ_1 is rewritten as Equation (2.17) where *J* is the moment of inertia known from the data sheet.

$$\sigma_1 = 2\xi\sqrt{J\sigma_0} - \sigma_2 \tag{2.17}$$

In [19] , researchers used a similar method for defining Stribeck curve friction as done in [17], [18] and [12]. The method involves first trying to reach a constant velocity and then measuring the input torque; afterwards, the results are averaged, and the process is repeated for a different velocity point. The researchers conducted the second experiment to define the dynamic friction parameter. The main difference is that they applied the genetic algorithm for estimating the parameter.

The genetic algorithm (GA) in [24] is based on simulating the process of a natural genetic mechanism, biological evolutionism, and a parallel random searching optimal method. The advantage of GA is that it does not need the object model and could avoid a local minimum; this method has been widely used, and its robustness has been proven.

By now, the above four studies [17], [18], [12], [19] have been reviewed to identify the LuGre Model friction parameter. The common point about these four studies is that they all apply Newton's friction law to obtain a Stribeck curve plot. Then, the researchers defined the static model parameter differently using their estimated method. To define the dynamic parameter, three of the studies [17], [18], [19] have a standard way of doing this using the presiding displacement versus input torque plot to define the σ_1 and σ_2. One of the studies used the system response directly to get step input. However, the estimation process is somewhat different. Two of the studies [17], [18] used a minimal sine wave to define the σ_1 and σ_2. Most of the studies used the curve fitting to estimate the parameter. In [19], the researchers used a more advanced way of GA to define the parameter. In the end, there is no general way to estimate the LuGre friction parameter; the identification of the LuGre friction needs to be calculated case by case because a different system has a different response. Therefore, the above studies served as a guideline to this thesis on how to choose a proper method to define the LuGre model according to the need in a real system.

2.2 CONTROL

In this section, the thesis focuses on the control of the ball screw system, which also includes the filter, the observer, and the trajectory generation. After completing the system identification process, the control of the system is the next issue to be discussed. Control system design is one of the keys to obtaining a reference target. The control section is split into two subsections that expound on the trajectory generation and the control system design.

2.2.1 Trajectory Generation

Trajectory generation is a term more often used the robot manipulator application. They often need to move in the 3D space. In this research, trajectory generation is one of the ways to help increase the accuracy of the system. Instead of using the only one final desired position such as step reference, trajectory planning could provide more detailed information to the desired final position. One way of doing this is to give the sequence of the desired *via points*. The frame is one of the terms used in the trajectory generation of trajectory planning; this can be likened to traditional motion films where each picture is a frame. Time is continuous, but on the computer, time is divided by each number. In trajectory planning, each point contains not only the information of the position and direction but also the information of time, and each point is a frame. Path points is a general name that describes the via points, such as the initial point and final point. One of the significant advantages of using trajectory planning is that users can let the system to performing smoothing.

This cannot be achieved with step reference since less information—such as step time and step position—is given. Trajectory planning could create a path ahead of motion to reach the desired step position in a very smooth way. In this way, the controller could have more information on how to reach that position. Also, trajectory planning not only creates position information, velocity, acceleration, and jerk is also desirable. This information is beneficial when the controller may need the reference velocity and acceleration in the algorithm.

In [25], trajectory planning is presented. Condition is that both the initial point and final point have zero velocity. To create a smooth motion, the initial position and final position with its time frame is defined using four kinds of data—two positions and two timeframes as follows. This could be expressed as follows:

$$\begin{aligned} \theta(0) &= \theta_0 \\ \theta(t_f) &= \theta_f \end{aligned} \tag{2.18}$$

where the θ is the position angle with time and θ_0 is the initial position. The t_f is final time, and θ_f is the final position. In addition, the condition with zero velocity for the initial and final position is fulfilled. The constraints for the first condition could be presented as follows:

$$\begin{aligned} \dot{\theta}(0) &= 0 \\ \dot{\theta}(t_f) &= 0 \end{aligned} \tag{2.19}$$

By using the cubic polynomials with the constraints (2.25) and (2.26) at a third order, a polynomial could be formed to describe the position, velocity, and acceleration:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{2.20}$$

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{2.21}$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t \tag{2.22}$$

By combining the constraints in Equations (2.25) and (2.26) into Equation (2.27) to (2.29), the following can be obtained:

$$\begin{aligned} \theta_0 &= a_0 \\ \theta_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ 0 &= a_1 \\ 0 &= a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{aligned} \tag{2.23}$$

At this point, there are four equations with four unknowns—*a_1*,*a_2*,*a_3* and *a_4*. By using the known values, the equation can be solved:

$$\begin{aligned} a_0 &= \theta_0 \\ a_1 &= 0 \\ a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0) \\ a_3 &= -\frac{2}{t_f^3}(\theta_f - \theta_0) \end{aligned} \tag{2.24}$$

By using the Equation (2.31) with Equations (2.27 to 2.29) with the constraints Equations (2.25) and (2.26), the trajectory information (i.e., position, velocity, and acceleration) in each time period could be obtained if user provide the sampling time, initial position, final position, and time for reaching the final position. The plot of using the above equation is presented in Figure 2.2.1.

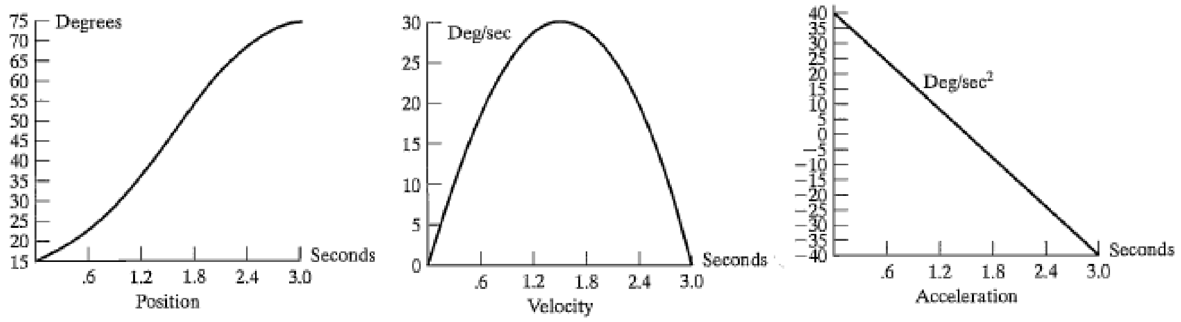


Figure 2.2.1 Trajectory planning with initial and final zero velocity [25]

Cubic polynomial trajectory generation is a very effective way to generate the trajectory planning profile. The amount of calculation is limited, which could be helpful when applying it to physical hardware. It could help step reference to be much smoother and could give the controller much more detailed information in reaching the final position.

2.2.2 Control system design

The control system design is one of the keys to having an accurate position tracking. In the literature, a variants control system has been present. A common point found that in all the studies is that they have conducted the LuGre friction estimation part. However, they may use a different method to achieve the estimation result. This subsection of the literature review will present each of the methods.

JunYa and Takayuki [17] presented a way to estimate the LuGre friction using the unscented Kalman filter. The control block diagram used in this paper is seen in Figure 2.2.3. The whole control system has two parts—the unscented Kalman filter (UKF) friction compensator, and a tracking controller. The UKF estimates the friction and provides it for tracking controller. The tracking controller based on discrete time optimal linear quadratic regulator.

The reason for using UKF instead of the Extended Kalman Filter (EKF) is because there is discontinuity LuGre friction equation. Indeed, EKF can deal with nonlinearity using the Jacobin matrix in each sampling point, and the calculation amount is less than UKF however, it cannot deal with the discontinuity. Therefore, in this case, UKF can bypass the Jacobin matrix and solve the discontinuity problem [17].

In [17] , the researchers also applied the above DLQ and UKF friction compensator into a physical ball screw table, after which they conducted an experiment. The result shows that the controller with only DLQ and no friction compensator has a considerable tracking error. By adding the UKF friction compensator, the tracking error has been mainly decreased. The lower tracking error approved the UKF friction compensator could accurately estimate LuGre friction. However, the UKF estimation process of getting a sigma point may still require significant computation power and memory as well. For less expensive microcontrollers such as Arduino, this issue may be a difficulty that needs to be overcome. Also, the observability prove for this method has not been present in this paper.

In a study done by Daiki and Norihiro [14], the researchers proposed a timer variant disturbance observer. The plant equation is based on Equation (2.4). Initially, [26] first proposed the continuous time friction observer. Later, the observer in [14] has been improved by placing a transformer into the discrete time observer. The researchers first separated Equations (2.4) into a time-invariant subsystem and time variant subsystem. The time-invariant subsystem is the plant model without the nonlinear part of the LuGre friction model, while the time-variant subsystem is the nonlinear part of the LuGre friction model. The two subsystems are both discretized by assuming a zero-order hold. Due to the high nonlinearity of the term $\frac{\left|\hat{v}_d(k)\right|}{g(\hat{v}_d(k))}$ in their LuGre friction equation [14], instant calculation of the term is required. This could create issues when being applied, the observability matrix becomes singular in certain point. By using the quadratic function that interpolate the observer gain, the observer's gain could place inside the unit circle

The study by Paul and Wonbo [18] shows another disturbance observer that is more applicable. In their work, they proposed a new friction estimation observer as the following:

$$\begin{aligned} \frac{d\hat{z}}{dt} &= v - \frac{|v|}{g(v)}\sigma_0 \hat{z} + k_z e, k_z > 0 \\ \hat{F} &= \sigma_0 \hat{z} + \sigma_1 \frac{d\hat{z}}{dt} + \sigma_2 v \end{aligned} \tag{2.25}$$

Here, v is angular velocity $\dot{\theta}$, k_z is the observer correction, and e is the position error. The position controls law combing with PD control and friction observer estimation is present below as:

$$\begin{aligned} u(t) &= \hat{J}\ddot{\theta}_d + K_p e + K_d \dot{e} + \hat{F}(\cdot) \\ e &= \theta_d - \theta \\ \dot{e} &= \dot{\theta}_d - \dot{\theta} \end{aligned} \tag{2.26}$$

The plant notation follows Equation (2.5) as discussed in the previous section. In Equation (2.32), *Kz* is the tune parameter, $\theta\ d$ is the desired the reference angle displacement, and θ is the measurement of the angle displacement by the sensor. The parameter $\hat{J}$ and another parameter in the LuGre model are identified in the friction identification. Step and sinusoidal reference are tested, and the result shows that the control law could deal with step input but would have a problem dealing with the sinusoidal reference. Tracking error is significant for sinusoidal reference. The tracking error is caused by a characteristic of PID control of poor trajectory tracking [27], and this may be due to parameter estimation being based on the friction estimation parameters. Therefore, friction estimation error could be traded as the slowly varying disturbance. Ohnishi [28] proposed that disturbance observer could cancel out the friction basis, parameter uncertainty, external disturbance, and parameter variations [29], [30]. The general disturbance observer could be presented as seen in Figure 2.2.2. The disturbance observer in Figure 2.2.2 estimated the error between the control input and the inverse of the nominal plant transfer function. The low pass filter Q(s) tries to filter out the measurement noise and high-frequency change of the estimated disturbance.

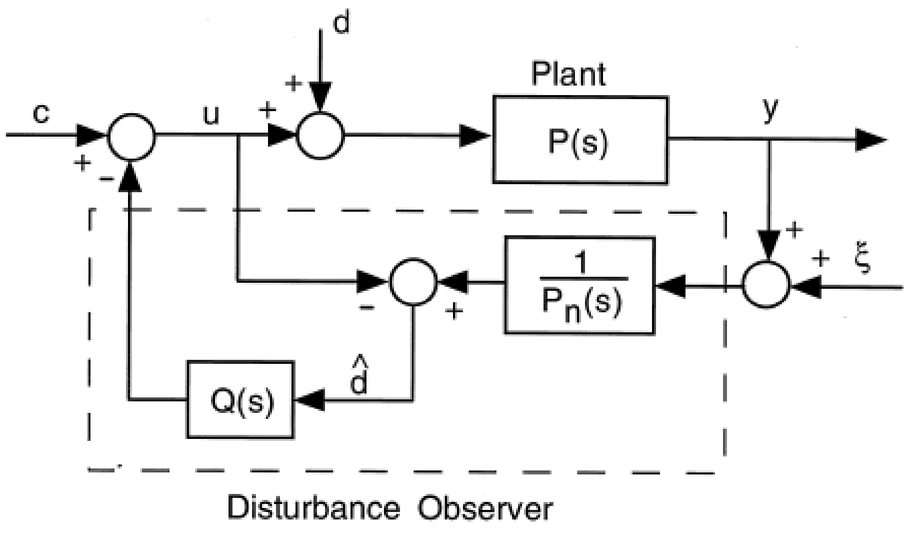


Figure 2.2.2 General disturbance observer [18]

In [18] , the researchers used the feedback signal to create a system dynamic with the friction observer and an estimated *J.* The system dynamic is based on the plant in Equation (2.6). Therefore, the disturbance observer with PD control law is presented below in block diagram format:

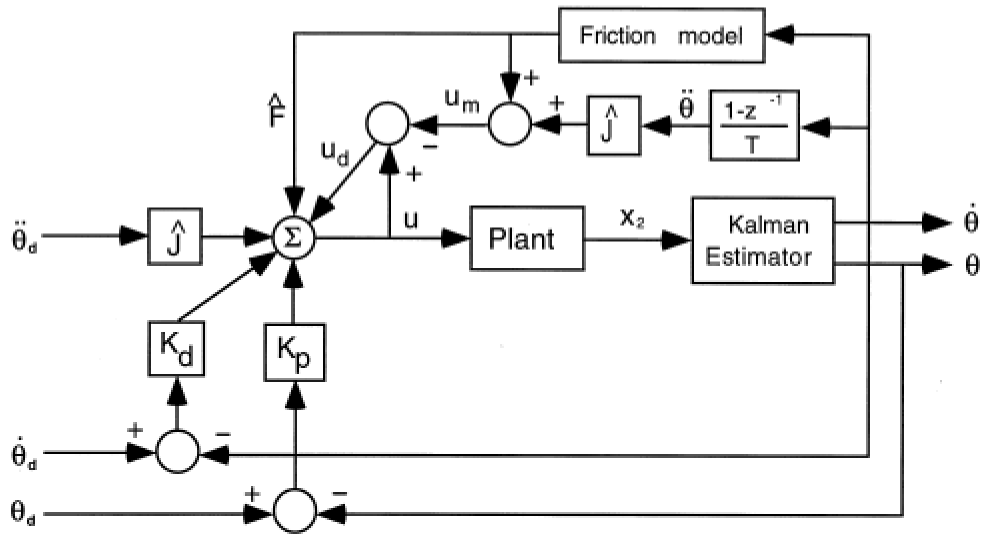


Figure 2.2.3 Proposed disturbance observer and PI control system

Figure 2.2.3 also shows the Kalman estimator, which estimates the angular velocity and angle using the motor rpm and linear displacement. A derivative of the angular velocity obtains the acceleration.

The researchers [18] also applied the above control law on a dSpace DSP controller and tested it on physical ball screw system. They found that the tracking performance for sinusoidal reference at a sub-micrometer level had a huge decrease compared to the PD control with only the friction observer.

3 CHAPTER HARDWARE IMPLEMENTATION

3.1 ELECTRICAL IMPLEMENTATION

This section discusses the electrical implementation. The mechanical hardware implementation could reference to Appendix (7.3-7.4). The section includes two subsections that investigates the encoder and the digital to analog converter (DAC). This section mainly focuses on how to implement them into the ball screw system. During the encoder testing process both Teens3.5 and Arduino Due microcontroller has been applied. The microcontroller used in this project could be reference to Appendix 7.5

3.1.1 Encoder

The encoder is one of the position sensors that measures the displacement of movement. It is an electromechanical device that measures the position or motion. In general, an encoder uses an optical sensor to produce electrical signals—such as a pulse train—which could be transferred into the motion, direction, or position. In this project, a linear incremental encoder has been attached to the ball screw system.

An incremental encoder is also called quadrature encoder. The basic working principle of the incremental encoder is to have two position sensors that scan an incremental scale, which then produces a series of square wave pulses. Figure 3.2.7 shows a breakout of a simple structure of the incremental encoder. An infrared LED light is beamed onto an incremental scale that contains the reflective and non-reflective area through the transparent read head window [31]. Normally there are two read heads on the linear encoder. Each reflective area reflects the light; the photodetector detects the light and produces the voltage pulse. Figure 3.2.8 shows how the signals are detected using the two photo detectors. Although it presents a rotatory incremental encoder, the principles are the same for the linear one. Between the incremental scale to the photodetector, there is also an index grating that averages the detected signal and effectively filters out the unmatched scale period signals. In this way, the signal stability can be achieved, which could avoid scale contamination or damage. Two photo detectors are separated by a certain distance, and this separation produces two separate square voltages which are commonly called Channel A and Channel B [32]. The number of rising edges and falling edges of the square wave is directly related to the displacement of the motion. The square wave contains information not only on position but also on direction. As shown in Figure 3.2.8, when Channel A leads Channel B, the direction of motion could be defined from left to right; when the Channel B leads Channel A, the direction of motion could be defined from right to left. In general, the direction still depends on how the hardware defines Channel A and Channel B, but the basic principle is that when Channel A leads the motion of direction, it is opposite to the direction that Channel B would lead.

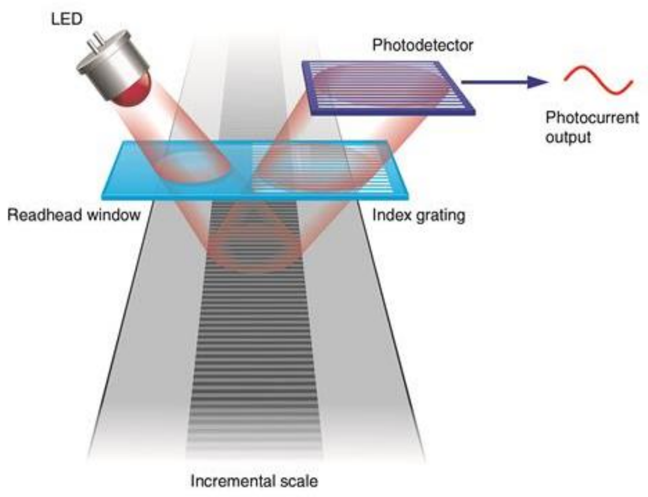


Figure 3.1.1 Incremental encoder breakout [31]

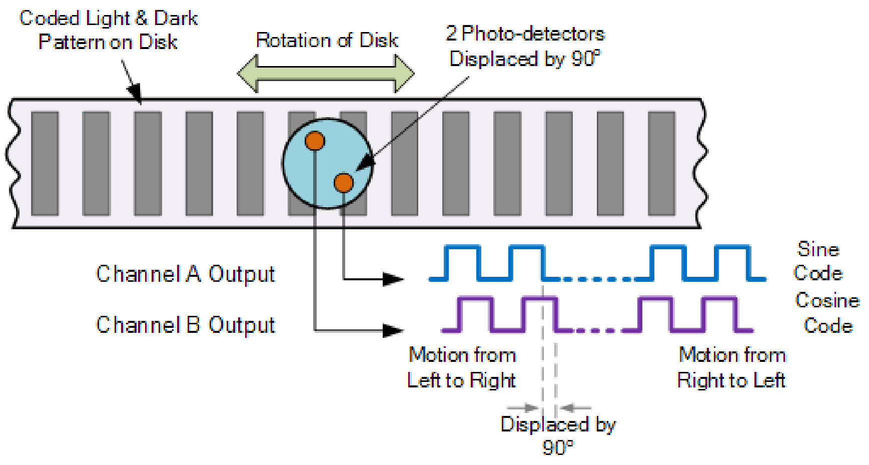

Coded Light & Dark Pattern on Disk
Rotation of Disk
2 Photo-detectors Displaced by 90°
Channel A Output
Channel B Output
Sine Code
Cosine Code
Motion from Left to Right
Motion from Right to Left
Displaced by 90°

Figure 3.1.2 Encoder signal detection [32]

In the next step, the square waves of the Channel A and Channel B were decoded and converted to the position. The edge counting of the square wave is related to the displacement. Normally there are three decoding of the resolutions X1, X2 and X4. Figures 3.2.9 to 3.2.11 shows all the decoding resolutions. Figure 3.2.9 presents the X1 resolution; this is the point when Channel A lead Channel B, and only the rising edge of Channel A is counted as an increment. When the Channel B leads Channel A, only the falling edge is counted as a decrement. Figure 3.2.10 presents X2 resolution; X2 differs from the X1 resolution in that both the rising edges and falling edges of the X2 resolution need to be detected by the counter. Therefore, each cycle (from Ch. A rising to Ch. B falling) of two increments and decrements were counted. The increment and decrement determination laws are the same for X1 resolution. Finally, X4 resolution is seen in Figure 3.2.11. In X4 resolution, both the rising edges and falling edges on Channel A and Channel B were counted, and each cycle (from Ch. A rising to Ch. B falling) resulted in four increments or decrements. The direction determination is the same for X1 resolution. Overall, X4 resolution provides the most accurate measurements due to each reflecting area on the incremental scale being detected by the counter in the microcontroller.

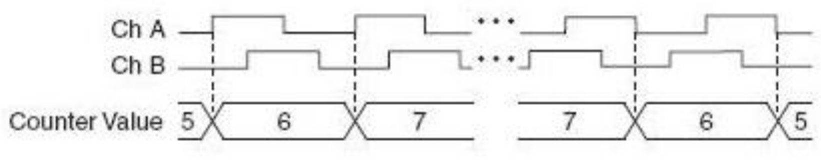


Figure 3.1.3 X1 resolution [33]

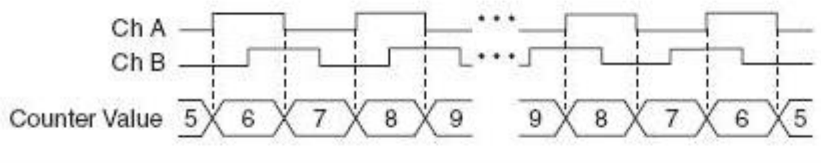


Figure 3.1.4 X2 resolution [33]

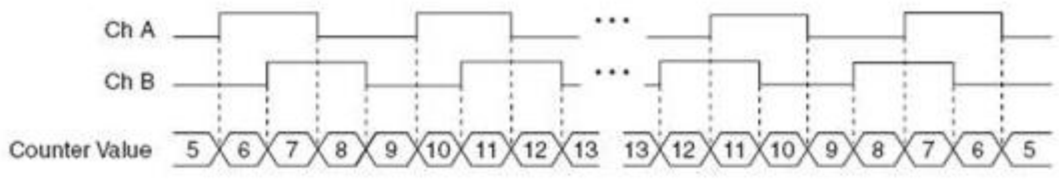


Figure 3.1.5 X4 resolution [33]

The linear incremental encoder used in the project is the JCXE5 linear encoder. The specification of JCXE5 is displayed in Table 3.1. Each resolution of the linear encoder scale presents one complete cycle of a channel—that is, from Channel A rising to Channel A rising again. If X4 resolution has been applied, a total of 5 counts would be detected in each resolution. Each edge counting is 1 µm for the JCXE5 linear encoder [34].

Model	Resolution	Max speed	Accuracy 50-500 m	Accuracy 510-1000 m	Operating voltage
JCXE5	5 µm	60 m/min	±5 µm	±8 µm	5 V±5%

Table 3.1 JCXE5 encoder specification [34]

To test the decoding method for this encoder, both the Arduino Due and Teensy 3.5 microcontrollers were used. The first problem needed to be solved is voltage conversion. Both Arduino Duo and Teensy 3.5 operate at 3.3V. The JCXE5 encoder requires the operating voltage to be at 5 V with a 5% variation using the TTL signal port [34]. The problem could be solved using a bi-directional logic level converter (BD-LLC). This device could shift 3.3 V up to 5 V or 5 V down to 3.3 V. The idea behind the BD-LLC is that the bidirectional level shifting could be achieved using a single N-channel MOSFET and a couple of pull-up resistors [35]. The hookup schematic can be referred to in Figure 3.2.12.

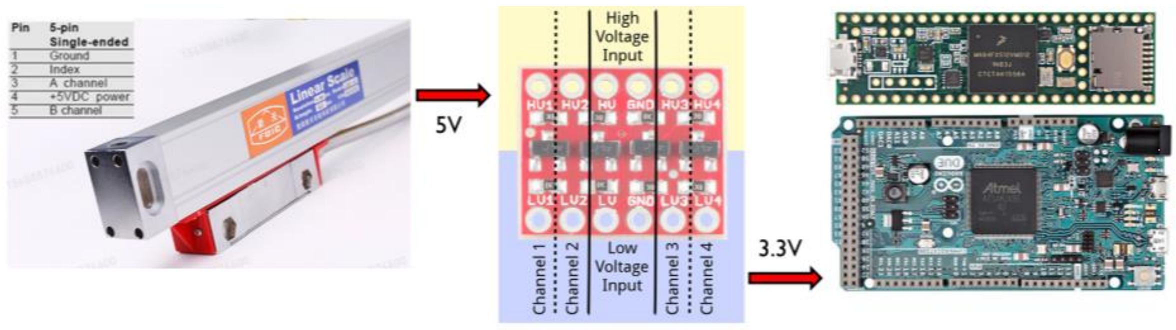


Figure 3.1.6 Bi-directional logic level converter schematic for connecting a linear encoder to Arduino Duo and Teensy 3.5

The BD-LLC has two sides—namely the high voltage input side and low voltage input side. Each BD-LLC could support a total of four channel conversions. The JCEX5 linear encoder connects the high voltage pinout (e.g., HV1) and low voltage output pinout (e.g., LV1) to the Arduino Due or Teensy 3.5. The BD-LLC also needs a reference voltage for shifting target. Teensy 3.5 can supply both a 5 V and 3.3 V constant voltage to the reference voltage pins (i.e., HV and LV pins). Therefore, the output 5 V from the linear encoder is downshifted to 3.3V; this voltage can be read by both the Arduino Duo and Teensy 3.5 from their I/O pins.

The decoding method was first tested on the Simulink Arduino support package. By using the Arduino Simulink support packing, the Simulink program could directly run the supporting Arduino platform. In order words, direct coding in Arduino is not required. The Simulink encoder reading model that was developed for this project is shown in Figure 3.2.13. The break down for Figure

3.2.13 is in Figures 3.2.14 and 3.2.15. The Simulink first reads the Channel A and Channel B signals using the Simulink digital input block. Every pulse for Channel A and Channel B results in either 1 or 0.

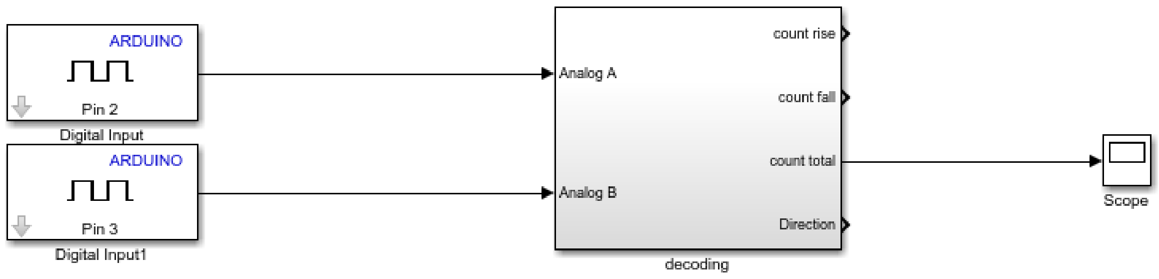


Figure 3.1.7 Simulink Arduino encoder decoding program

In the decoding block, the 0 and 1 pulse signals are counted; the detail structure can be found in Figure 3.2.14. This decoding block first tries to achieve X2 resolution by detecting the rising and falling changes of Channel A and counting number of pulses using the Channel B signal. The counting process is referenced in Figure 3.2.15. Regarding the triggered rising edge block, it is active with each variation from 0 to 1 for Channel A. Inside this block or subsection, there is a trigger function block that detects change.

The operation process of the trigger function block can be summarized in Figure 3.1.9. When the encoder moves from left to right (i.e., Ch. A leading Ch. B), Channel A first rises from 0 to 1 (i.e., rising), and the “triggered rising edge” block is enabled. The “trigger rising edge” block begins to receive a signal from Channel B; at this moment as seen in Figure 3.2.14, the arrow shows Channel B being at 0. After this, Channel B goes through the sum and adds the 0.5, and it then goes through the sign function and become 1. It then separates to two ports—one directly out of block to one direction port. The other one goes through the memory block to count 1 pulse, and then it outputs to a rising edge counter port. After that, the encoder keeps moving; when the encoder moves to a certain place that Channel A begins falling (i.e., from 1 to 0), the triggered falling edge block is enabled.

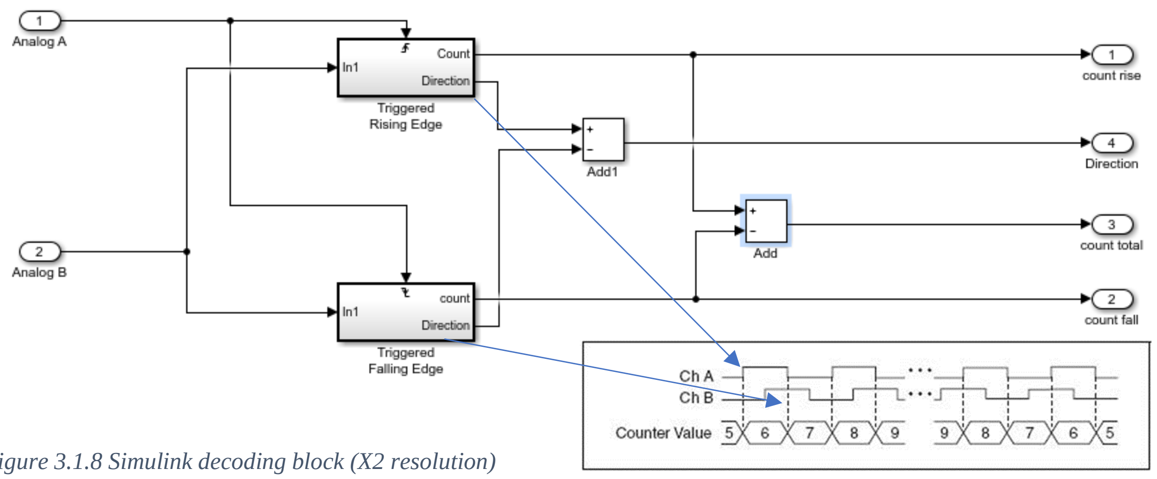


Figure 3.1.8 Simulink decoding block (X2 resolution)

The triggered falling edge block is exactly the same as the triggered rising edge block; the only difference is that the trigger function is enabled when the falling edge has been detected. In the triggered falling edge block, Channel B is currently at 1 (see the second arrow in Figure 3.2.14), then 1 subtract 0.5 became -0.5 then go through the sign function became -1 and it separated into two paths—one goes through the direction port presently at -1. The other one goes though the memory port and counts down -1 every time. At this point in the decoding block, the output count port (currently at 1) from triggered rising edge block subtract the output count port from triggered falling edge block (currently at -1) will obtain a value of 2. The process indicated that one cycle (Channel A rising to Channel A falling) is complete and total two pulses have been counted. The triggered rising edge block's direction port output subtract triggered falling edge block's direction port output will result in a value of 2, which indicates that the encoder is moving from left to right. If the encoder keeps moving from left to right, the direction port will always show a value of 2. When the encoder changes the moving direction, the above principle can be vice versa; the count total will be decreased by two pulses each cycle, and the direction port would instead show a value of -2 constantly.

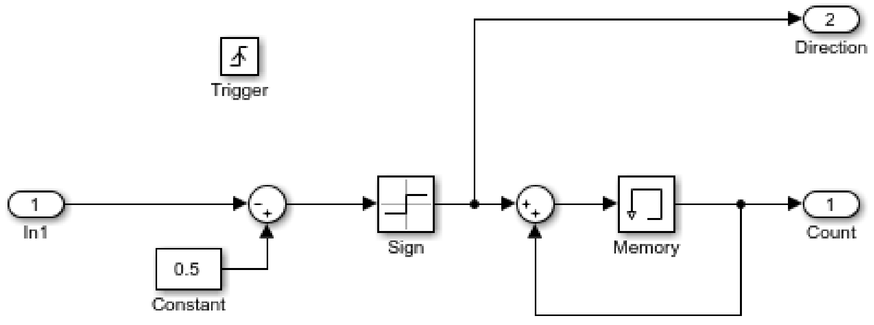


Figure 3.1.9 Triggered rising edge block

The experiment has been performed to test the above method of reading the encoder. In the experiment, the ball screw motor was not used. The movement of the encoder was done by hand manually, and the ball screw system motor coupling was rotated. The moving displacement was measured using a Mastercraft digital caliper as shown in the Figure 3.2.16. The total moving displacement was found to be 11.93 mm. The measurement from the linear encoder by Simulink plus Arduino Due is shown on Figure 3.2.17.

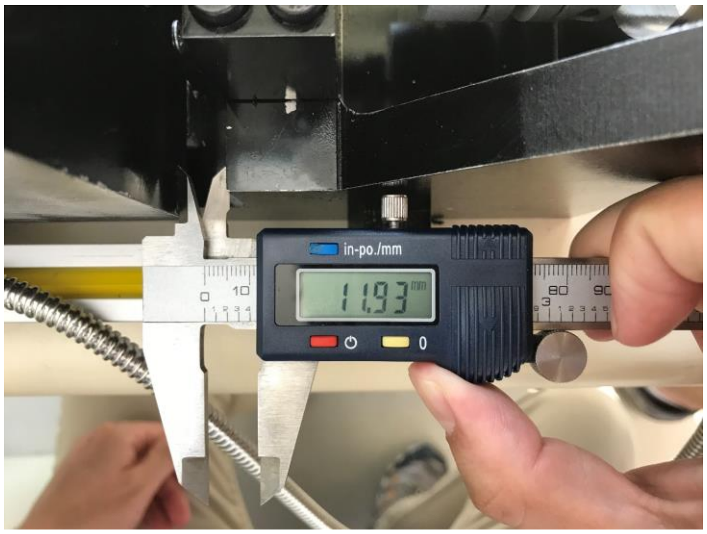


Figure 3.1.10 Caliper measured displacement

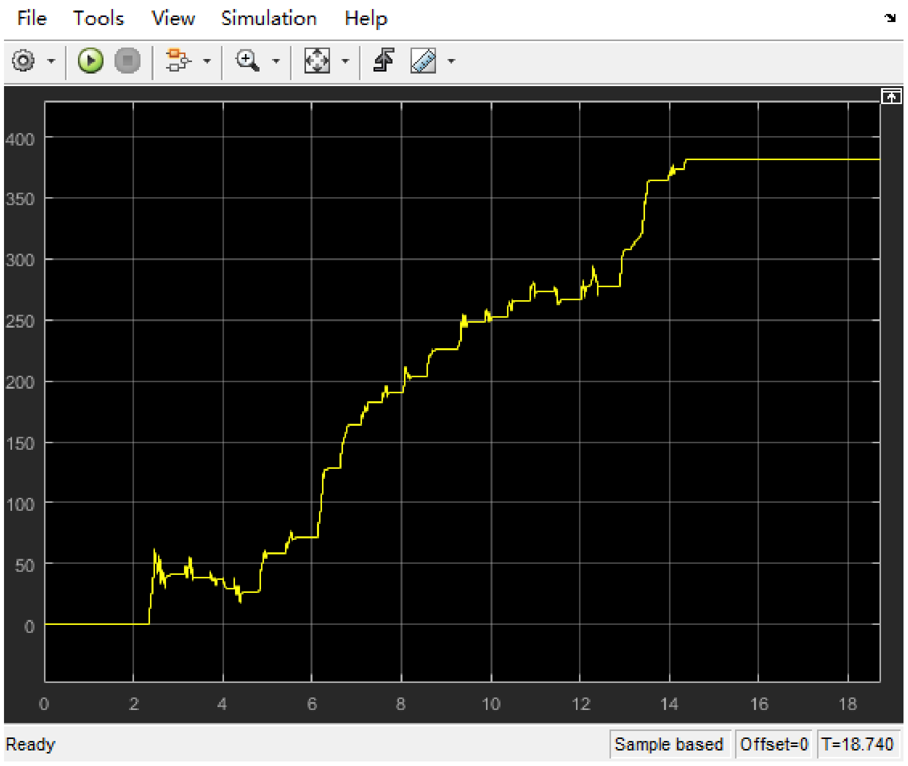


Figure 3.1.11 Encoder reading from Simulink

As the result, Simulink only obtained around 360 pulse signals. From Table 3.3, the specification of the encoder can be seen if each pulse has been captured (such as X4 resolution). The theoretical encoder should produce 11930 pulses; each pulse is 1μm. If the resolution is reduced to X2 resolution, there would be a total of 2386 cycles (i.e., 11930 pulses divided by 5 pulses or encoder cycle).[1] In X2 resolution, every encoder cycle must capture 3 pulses; therefore, a minimum of 7158 pulses should be obtained. In other words, reading the pulses using the Simulink is not sufficient. During the process, the Simulink decoding system lost many pulses. One possible reason behind the pulse loss could be that digital input with a trigger block is too slow for capturing the high-resolution pulse that is produced from the linear encoder [36].

The method of using the trigger block with a digital input block applies to software interrupt. In this case, the software interrupt cannot keep up with the pulse generation speed that is caused by the encoder. The problem is that the encoder pulse signal is sent to Simulink through the USB connection, and Simulink then makes the decision regarding the trigger or not to trigger counter. The transmission process takes time, which causes the delay of the reading pulse signal. Therefore, many pulses become lost during the process. The above method may be sufficient for an encoder that have a lower resolution.

Another way to solve the problem and obtain an accurate reading of an encoding pulse signal is to use hardware interrupt and hardware counter directly. In an Arduino environment, the user could directly control the hardware interrupts by using the ISR (Interrupt Service Routines) and enable the hardware counter register. The interrupt could be solved the timing problem in a microcontroller program; without the interrupt, it would be difficult to perform any other task in the microcontroller

[1] An encoder cycle is defined as Channel A rising to Channel A rising again based in Figure 3.2.11; a total of 5 edge changes are in one encoder cycle.

due to the program needing to check the sensor status constantly. The hardware counter register could also quickly counter each pulse signal. [37] By using the hardware interrupt which is already built into the microcontroller structure, the interrupt could save process power of the CPU and allow the CPU to perform any other task at a same time.

To use ISR, the entire system needs to be written in modified C language in Arduino IDE. Therefore, no MATLAB Simulink are involved in the process, and it would be a pure programming process. By using C language with ISR, the X4 decoding resolution could achieved. The flowchart for the reading encoder is seen in Figure 3.2.18. At this time, both Channel A and Channel B are attached to hardware interrupts (i.e., ISR_A and ISR_B). Therefore, both Channel A and Channel B edge change (rising or falling) could be detected by the microcontroller. In this way, X4 resolution could be achieved. Based on X4 resolution's principle (see Figure 3.2.11), when Channel A is detected to have an edge change, the program will enter the ISR_A. In ISR_A, if A is 1 and also B is 1 than counter increment. Therefore, Encoder moving left to right. If A is 1, but B is 0, then counter decrement which means encoder moving right to the left. If A is 0 and B is 0 then counter decrement, therefore encoder moving right to the left. The same principle applies to ISR_B. Finally, the total counter value of the pulse capture is sent to the serial port of the microcontroller. The user could obtain the data either using MATLAB's serial communication or the Arduino IDE serial monitor.

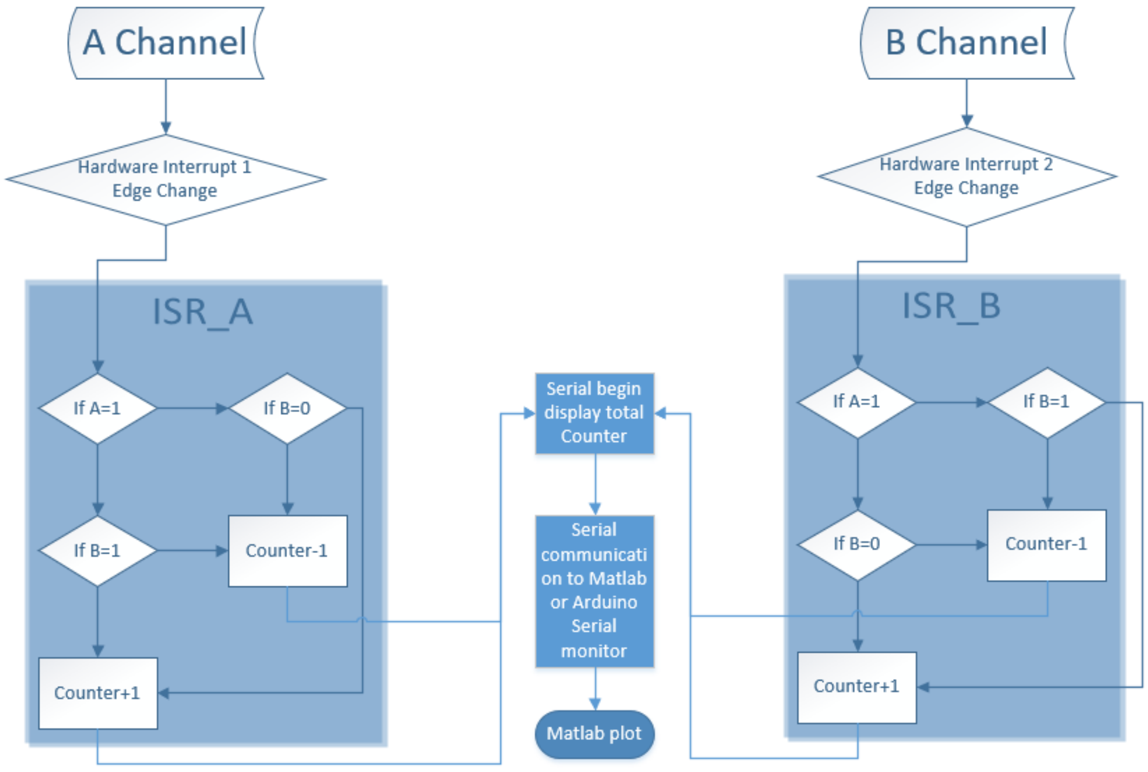


Figure 3.1.12 ISR reading encoder flowchart

Figure 3.2.18 is implemented into C code in Teensy 3.5 microcontroller using the Arduino IDE. The final result is obtained through MATLAB using serial and plot in the MATLAB, as seen in Figure 3.2.19. Just like the previous method, the encoder is moved manually by hand. The measurement result by the caliper is seen in Figure 3.2.20. As a result, from the encoder measurement, there are a total of 11870 pulses that were captured by Teensy 3.5 and displaced on MATLAB plot. The measurement result from the caliper is 11.88 mm. The result from the encoder and physical measurement is very close (11870 pulses = 11870 μm = 11.870 mm). However, in the MATLAB plot, only 2056 data points were collected. There is still a significant amount of data lost during the serial communication process from Teensy 3.5 serial port to MATLAB serial read. In the Arduino IDE serial monitor, all 11870 data points were collected.

In conclusion, the method of using the ISR with Teensy 3.5 in pure C code could sufficiently capture all the movement of the high-resolution encoder. However, if any MATLAB communication is involved in the process of capturing pulses, this will result in losing pulse count. Therefore, in a future section, all hardware implementation including linear encoder decoding process will implement in Arduino C language alone in the Arduino IDE environment using the Teensy 3.5 microcontroller.

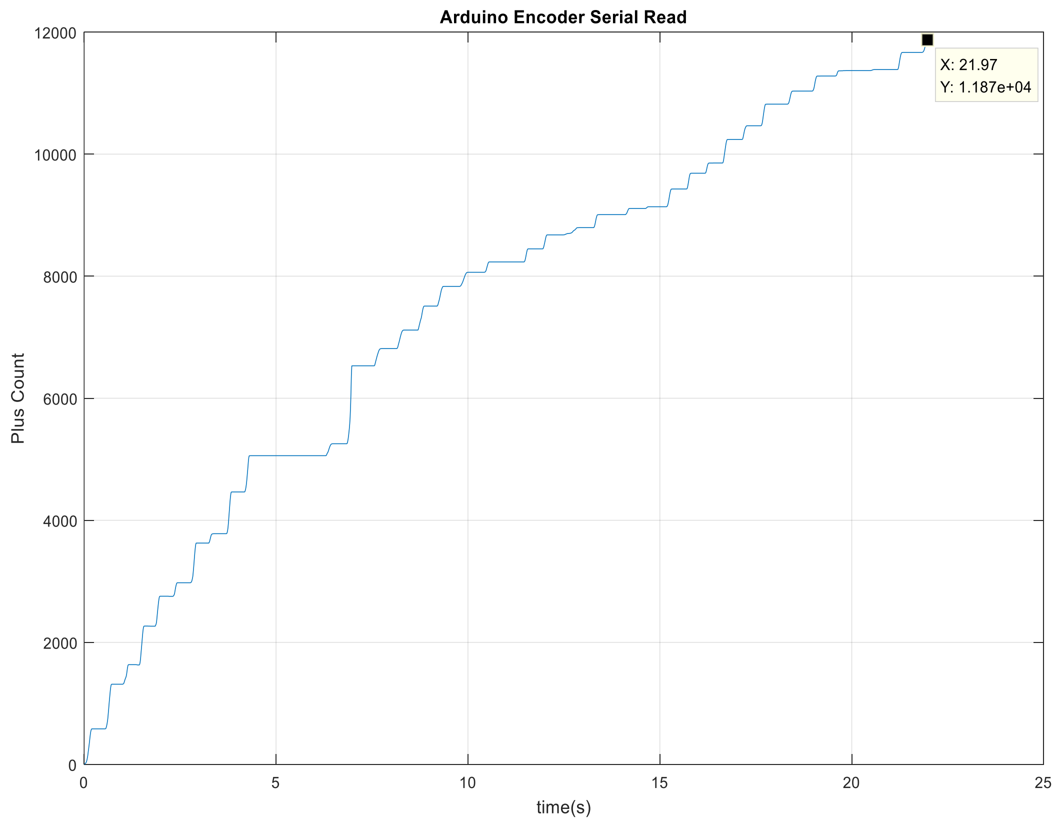


Figure 3.1.13 Encoder Teensy 3.5 MATLAB plot

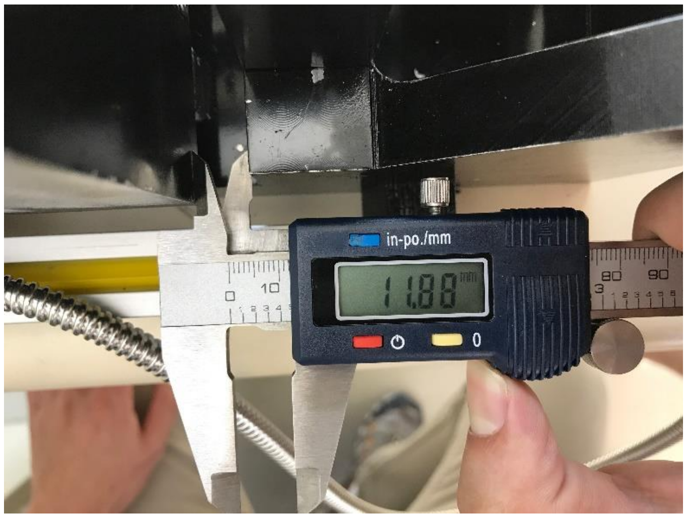


Figure 3.1.14 Measure displacement 2

3.1.2 Digital to analog converter (DAC)

For implementing the torque control model for this project, torque control power unit is required, Usually, a torque control power unit can be included in the analog output card, PLC (programmable logic controller), or a servo motion controller [38]. The cost of a PLC system for industrial application is around $500 USD [39], while the cost of an analog output card is around $250 USD [40] and the cost of a motion control card is around 600 USD [41]. Most of the old PLCs do not have an analog output, therefore an analog output card is needed; getting a new card would add up the cost of the control system. In [42], a way of using the PLC to control the servo system was presented.

Instead of using the above devices, this thesis presents a customized device that connects with Teensy 3.5 to perform the torque control. Based on the AC servo driver manual [43], the torque input reference voltage requires a ± 10 V voltage (a reference to ±1.27 Nm). The Teensy 3.5 DAC could only produce 0 V to nearly 3.3 V. Therefore, this project required a more significant range of DAC to achieve the target.

To achieve the output ± 10 V voltage, a customized DAC (CDAC) is built, as seen in Figure 3.2.21 (see Figure 3.2.22 for the schematic). The costs of the materials for customizing DAC present in in Table 3.2.

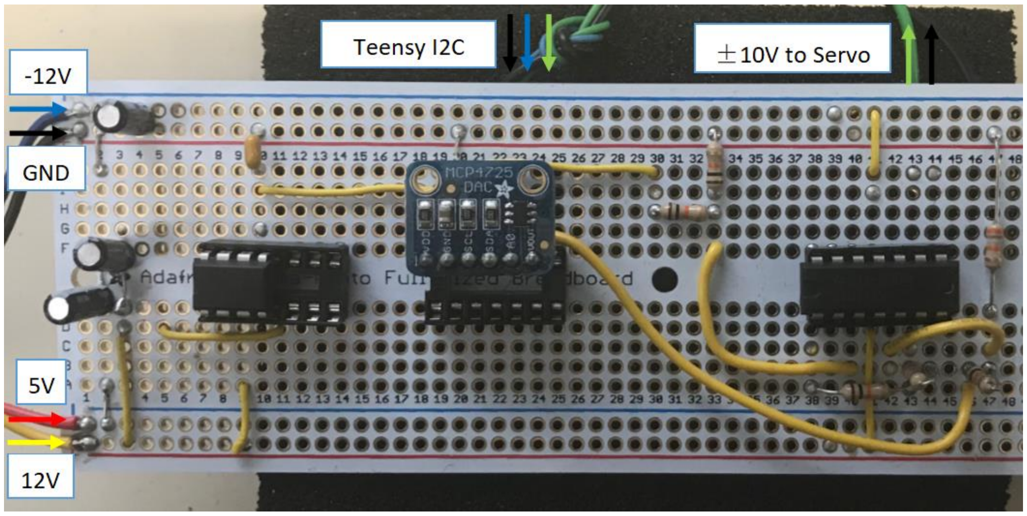
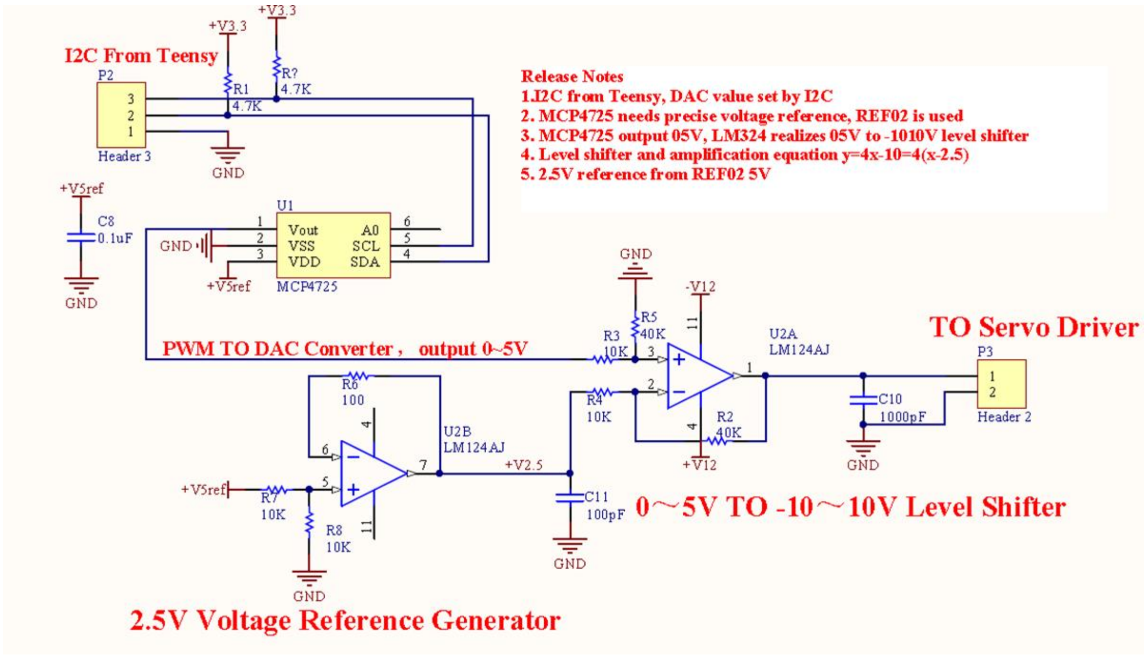


Figure 3.1.15 A customize DAC board



Release Notes
1.I2C from Teensy, DAC value set by I2C
2. MCP4725 needs precise voltage reference, REF02 is used
3. MCP4725 output 05V, LM324 realizes 05V to -1010V level shifter
4. Level shifter and amplification equation y=4x-10=4(x-2.5)
5. 2.5V reference from REF02 5V

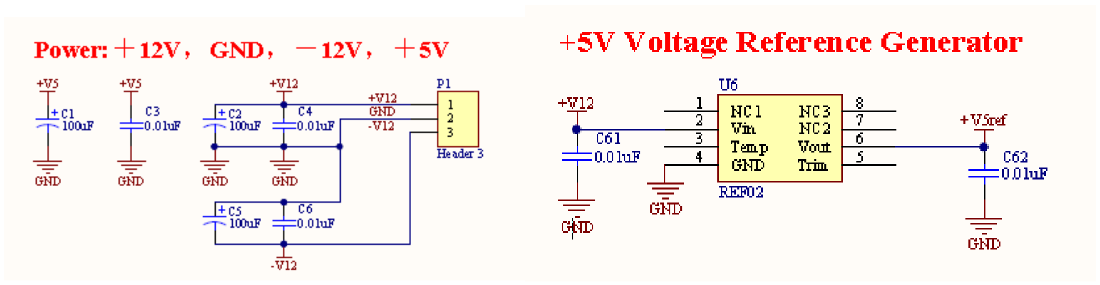


Figure 3.1.16 DAC schematic

Part	Quantity	Single Price (CAD)	Total Price (CAD)
Capacitor 100µF	3	0.09	0.27
Capacitor 100pF	1	0.3	0.3
Capacitor 1000pF	1	0.51	0.51
Capacitor 0.01µF	5	0.11	0.55
Capacitor 0.1µF	1	0.06	0.06
Resistor R4.7k	2	0.00691	0.01382
Resistor R100	1	0.00691	0.00691
Resistor R10k	4	0.00691	0.02764
Resistor R40k	3	0.00691	0.02073
MCP4725	1	6.64	6.64
LM124AJ	2	0.57	1.14
REF02	1	4.95	4.95
ATX-300GU	1	43.91	43.91
Total DAC price			58.3991

Table 3.2 DAC BOM

The main components of such a device includes the power supply ATX-300GU which supplies a ±12 V and 5 V voltage, a voltage reference generator REF02 for an accurate output of 5 V, an I2C DAC MCP4725 that produces a 0 to 5 V analog voltage, and two quadruple operational amplifiers LM124 that converts voltages—namely (0 to 5V) to (-10V to 10V).

The operating process—as seen in the schematic in Figure 3.2.22—starts from the power supply. The ATX power supply sends 5 V and ± 12 V to the CDAC. Due to the power supply, the voltage is not always keep at precisely 5V and ± 12V—that is, it will fluctuate. Therefore, it first goes through a series of capacitors that can filter the voltage fluctuation. The 12V voltage then goes through the REF02 output and exits precisely as 5 V with nearly no fluctuation; this 5 V is called 5Vref that will be used in the level shift calculation. The MCP4725 uses this 5Vref to produce an output of 0 to 5V analog voltage based on the I2C command sent from Teensy 3.5 (namely SCL, SDA pins). The MCP4725 has a range of the full 12-bit scale [44], and therefore the control range from the Teensy 3.5 side is an unsigned integer number from 0 to 4095 that represents the 0 to 5V output.

The next task is the transformation of the 0 – 5 V to ±10 V. The idea is to divide the 5 V in half, which is 2.5 v. From 0 V to 2.5 V, the -10 V to 0 V is presented as the output, and from 2.5v to 5v, 0 V to 10 V is presented as the output. For the above task, the differential amplifier method is required. The operation amplifier (op-amp) used for protection is the U2A-LM124AJ. The differential amplifier subtracts the two voltages and time to obtain the amplifier gain [45]. The basic theorem of the differential amplifier can be presented in Figure 3.2.23.

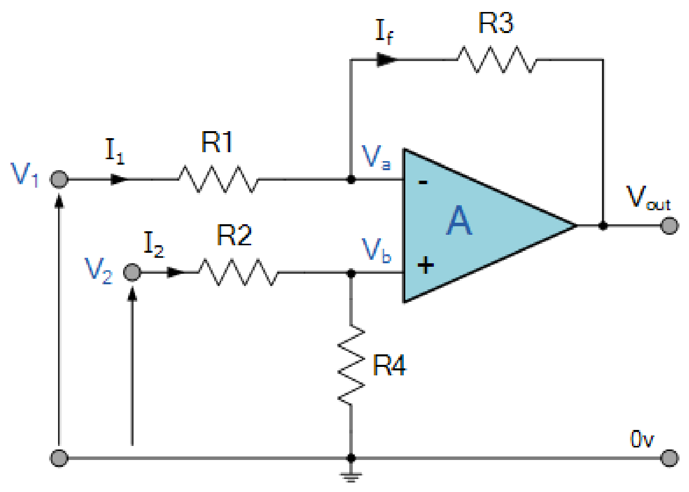


Figure 3.1.17 Differential Amplifier configuration [45]

Based on [45], when R1 = R2 and R3 = R4, the equation for Figure 3.2.23 can be expressed as below:

$$V_{out} = \frac{R3}{R1}(V2 - V1) \tag{3.1}$$

To get V_{out} = 10 V, V2 needs to be 5 V, and V1 as the benchmark is at 2.5 V. Equation (3.1) can be calculated using the following values where V_{out} = 10, V2 = 5 and V1 = 2.5. In solving R3/R1, the result is 4. The simple way is to select R3 and R1, and the two are set at certain values (i.e., R3 = 40 kΩ and R1 = 10 kΩ). The reason for choosing kilo-ohms is because the amplifier circuit use currents in the milliamp range [46]. The differential amplifier also need to have a supply voltage that performs as a saturation; the supply voltage should be higher than the output voltage, and in this case the output voltage is ±10 V. Therefore, U2A-LM124AJ uses the ±12 V that is sufficient for this task from power supply. By now, the 0 V to 5V has been differential to -10 V to +10 V. The next task is generated V1 at 2.5 V.

The above method required to have 2.5 V as the input for the U2A-LM124AJ op-amp. One way of doing that is to use two resistors that have same ohm (see Figure 3.2.22 R7 and R8) and then divide the 5Vref into two 2.5 V. After obtaining the 2.5 V, it then goes through a voltage follower op-amp circuit U2B-LM124AJ. The reason to apply another op-amp is to avoid the loading effect in the circuit. The voltage follower functions as a buffer; it does not amplify the input signal. Rather, it isolates the effect between the two circuits. This method is very useful when the input impedance is very high and output impedance is very low [47]. In other words, it helps the circuits to be kept at 2.5 V and not drop. The voltage may have dropped if the input voltage does not have enough carrying capacity to carry the load in the circuits. The basic circuit diagram of the voltage follower circle is seen in Figure 3.2.24. To avoid the short circuit in the feedback loop, a 100 Ω resistor is added to the feedback loop.

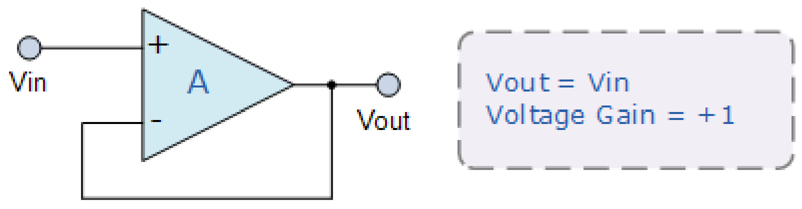


Figure 3.1.18 The voltage follower op-amp circuit [47]

Before applying the above schematic (see Figure 3.2.22) to the physical implementation, a Simulink model based on the above schematic is built to verify the CDAC design. The Simulink model is shown in Figure 3.2.25. The input of this model—which is MCP4725— generated an analog voltage of 0 V to 5 V, and the output should be ±10 V. The test has been done using the input as a sine function—namely $2.5\sin(t)+2.5$ (0 V to 5 V *sin* wave). The output and input for this Simulink model is shown in Figure 3.2.26. As the result, it proves that the current design will able to transfer the 0–5 V to ±12 V.

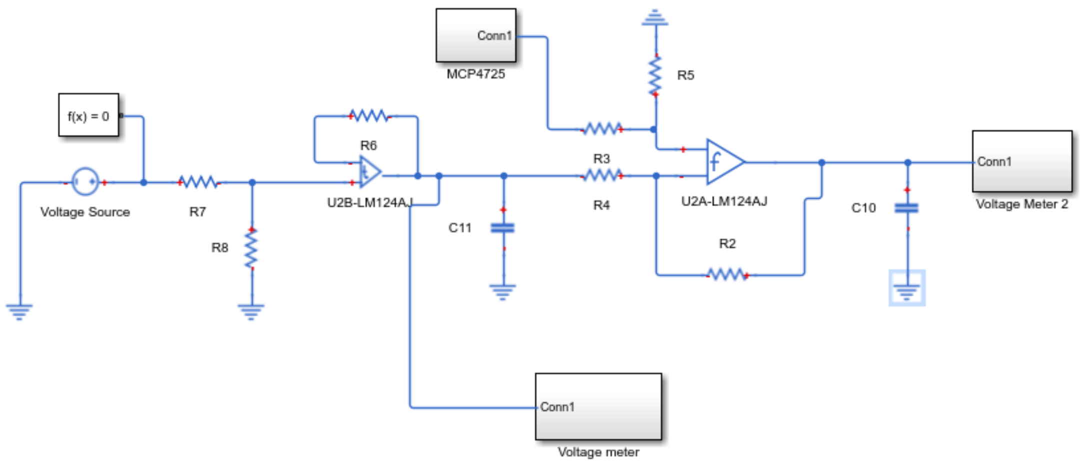


Figure 3.1.19 CDAC Simulink Model

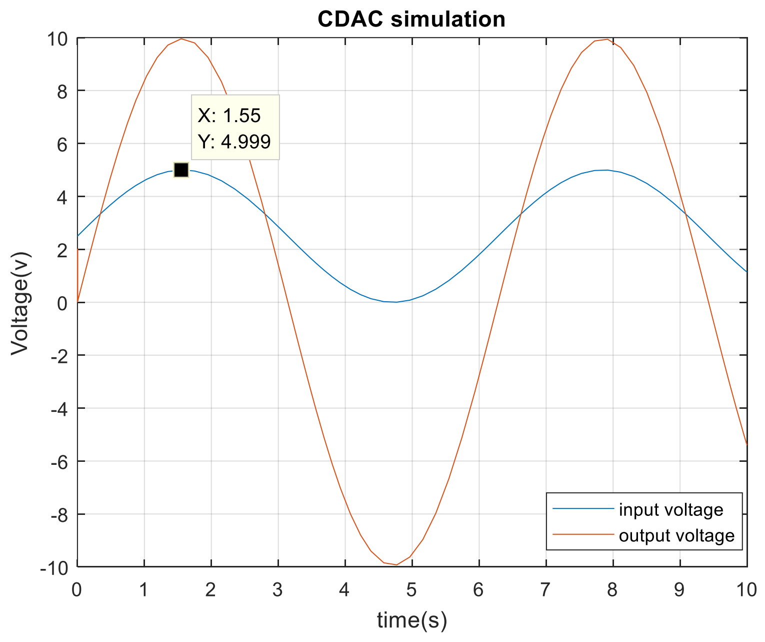


Figure 3.1.20 CDAC Simulink sin wave input and output

Based on the above design and simulation, the implemented experiment is conducted. From the side of the Teensy 3.5, a sinusoidal command with an amplitude of 2047 and a bias of 2047 was sent to the CDAC board. The CDAC board then was connected to an oscillation scope; the result is shown on Figure 3.2.24.

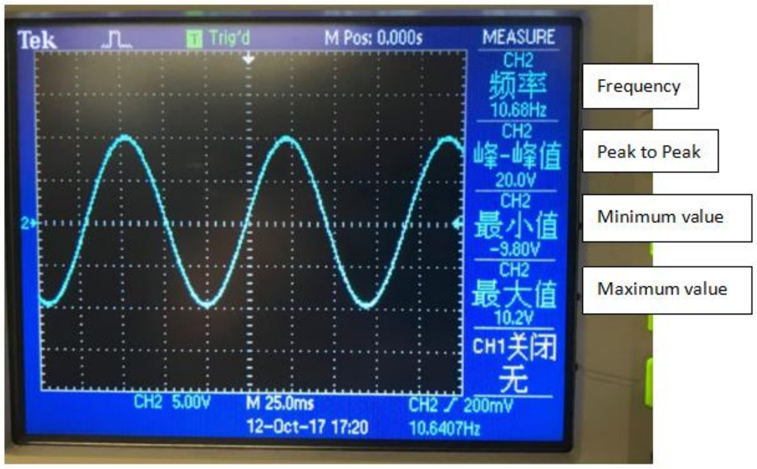


Figure 3.1.21 CDAC test oscillations scope result

The minimum voltage that the CDAC could obtain is -9.80 V, and the maximum voltage it is 10.2 V. This customizer DAC could achieve the design target using the serial I2C from Teensy 3.5 to produce an output analog voltage near the range of -10 V to +10 V. The offset of 0.2 V due to the power supply fluctuation. In conclusion, the proposed design could achieve a design target which produces ±10 V analog voltage. The cost of such a system has been lowered to $58.3991 CAD, which is less expensive than any other similar devices on the market (by my knowledge). The range from a programming point of view is 0 to 4094 integer commands to ±10v analog voltage; 4095 was not chosen because 4095 divided by 2 has a decimal point, which is difficult to divide in order to match the whole integers of -10 and +10 V. The resolution of the DAC can be summarized as 0.00488 V/int or 204.7 int/v.

3.2 Electrical and Mechanical System

The entire system—including the electrical and mechanical systems—can be constructed. The final system diagram is seen in Figure 3.3.1, and the schematic is shown in Figure 3.3.2.

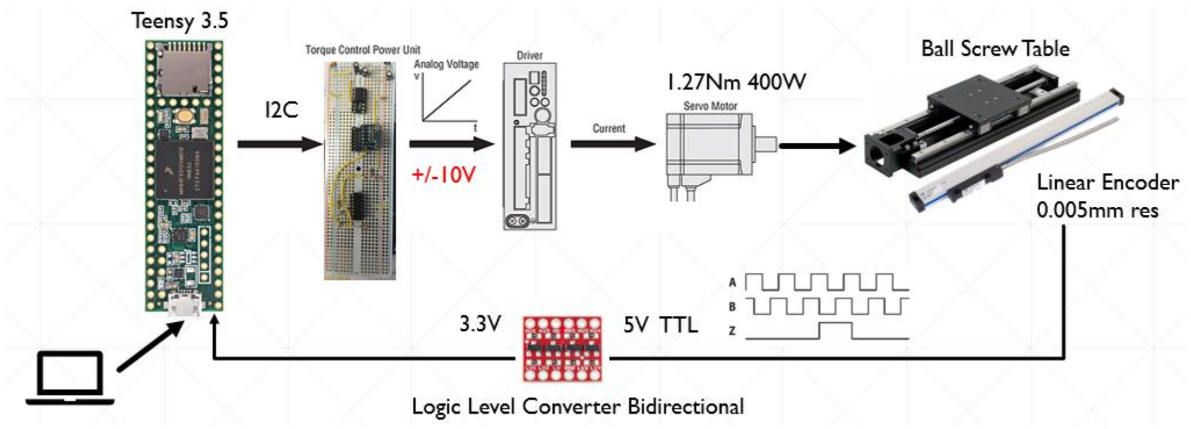


Figure 3.2.1 Ball screw system setup

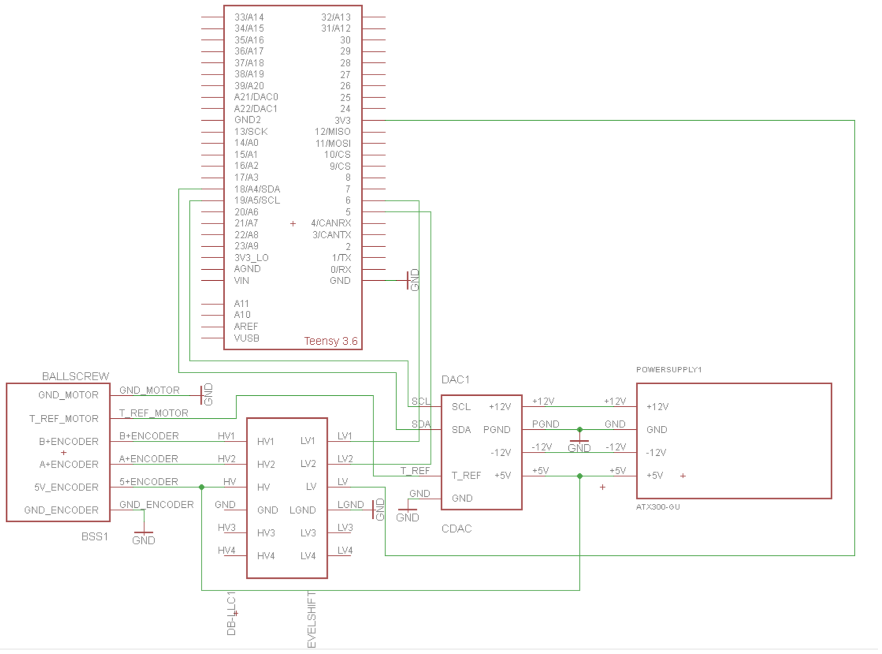


Figure 3.2.2 Ball screw system setup schematic[2]

[2] The schematic shows Teensy 3.6 because the EAGLE library only has Teensy 3.6. However, the output pins for 3.6 and 3.5 are the same.

First, the laptop sends a reference position, velocity, and acceleration command from the USB port to Teensy 3.5; the Teensy 3.5 then sends a serial I2C command through SDL and SDA pins to CDAC (i.e., the torque control power unit). Based on the I2C command (0 to 4095 integer), CDAC produces the analog voltage with a range of ±10V through T_REF pins to AC servo drive. The AC servo drive takes this reference voltage and transfers it to the corresponding current that is supplied to the AC servomotor. By receiving the current, the AC servomotor begins to rotate the ball screw coupling and move the sliding table. Movement of the sliding table drives the linear encoder. Therefore, the displacement change is detected by the linear encoder. The linear encoder sent to Channel A and Channel B is a 5 V square wave voltage. Although Teensy 3.5 could tolerate a 5 V voltage, a BD-LLC is applied to the lower linear encoder's 5 V to 3.3 V—which is the operation voltage of Teensy 3.5—in order to increase reading accuracy. The Teensy 3.5 receives the displacement feedback and decodes the square wave to physical displacement, and then the control algorithm that use displacement feedback produces controlled torque to achieve the reference position. By now, the system has closed its loop.

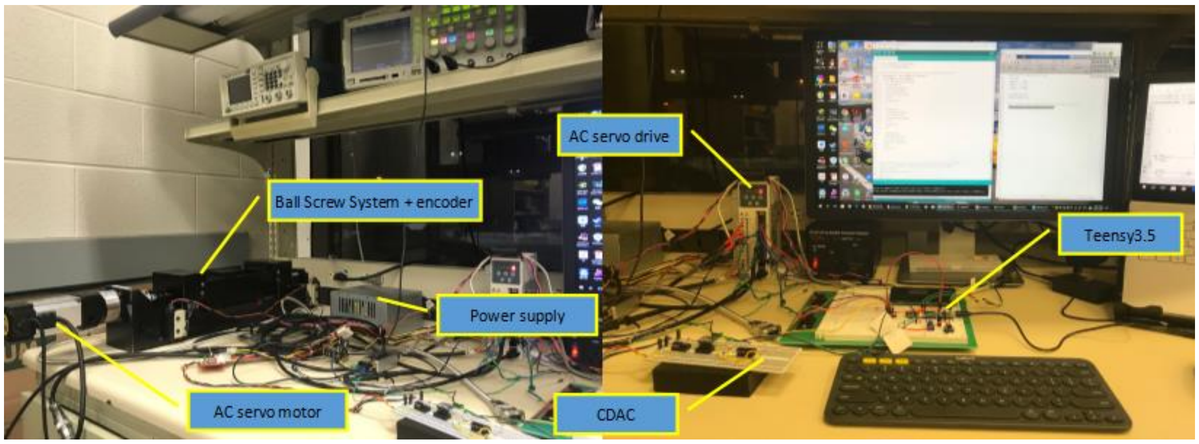


Figure 3.2.3 Ball screw system physical setup

4 CHAPTER SYSTEM MODELING AND PARAMETER IDENTIFICATION

4.1 SYSTEM MODELING

The primary objective of this section of the thesis is to introduce the ball screw system and LuGre friction model. Then, based on these, the parameter identification experiment is analyzed in the following section. Both models are used throughout the rest of this thesis.

4.1.1 LuGre friction model

The classic LuGre friction model presented in [48] [12], in equations 1.6 to 1.8, have concluded the most of characteristic of the friction such as the Coulomb friction, Viscous friction, and Stribeck curve friction. Beside above characteristics, The LuGre friction model recreates also the behavior of the presiding movement at the static friction region that Dhal model presented. The model includes the Stribeck curve phenomenon at maximum static friction Fs. Then, the system begins to display dynamic friction behavior. The old model (Equation 1.3) only has viscous friction combined with Coulomb friction, which is discontinuous when transferring from a static friction region to a dynamic friction region. The Stribeck curve solves the discontinuous problem by using an exponential function. After the Stribeck curve region, the classic LuGre friction model becomes pure viscous friction. The LuGre friction model allows the traditional function of Equation 1.3 to become a continuous function and includes more detail that describes friction behavior.

In the literature review in Section 2.1.1, several variations of the LuGre friction model were reviewed. In [14], the basic LuGre friction equation (1.6) has not changed. The main variation is the term δv in Equation 1.8. Their model [14] (Equation 2.3) assumes the δ_v is 1. Also, [17] agreed with [14], who decided that the term δ_v is 1. [18], similar to [17] and [14], changed the term δv to a constant number, and they assumed the term δ_v is 2. Therefore, for the classic LuGre friction model, the term δ_v is predefined in their [18] equation, which does not need parameter identification. The reason for this is that they follow a different principle, as suggested in [13] and [16]. In [13] suggest the term $\delta_v = 2$, and in [16] suggest the term $\delta_v = 0.5$ to 1. In the literature presented in [12], they improved the classic LuGre friction model by using the variable maximum static friction term. This new friction model [12] has not been applied in other papers yet. Based on the above conclusion, this research is based on the classic LuGre friction model:

$$F_f = \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 v \tag{4.1}$$

$$\dot{z} = v - \frac{\sigma_0 |v|}{g(v)} z \tag{4.2}$$

$$g(v) = F_c + (F_s - F_c) e^{-|\frac{v}{v_s}|^{\delta_s}} \tag{4.3}$$

The classic LuGre friction model is largely used in the literature, which proves it accurately represents friction behavior. Also, the selection of the term δ_v is not agreed upon in the literature. Therefore, in this research, the term δ_v is identified in the parameter identification process.

4.1.2 Ball screw system model

In the literature, [14], [17], [18], and [19] all discuss the ball screw system model. In [14], they present a ball screw system model based on Newton's second law of motion (*F=ma*), illustrated in Equation 2.1. In the equation, they used the force to balance each side of the motion. However, in [14]'s model, the inertia of ball screw, motor, and sliding table is not taken into consideration. In [18], they simplified Equation 2.5 and Equation 2.6, in which only the total inertia system is considered. The mass of the moving table is ignored. Also, [18] used torque to balance the input and output.

[17] and [19] present a more complex model than [14] and [18], in which they have considered the effect of both system inertia and system mass. In [17], they consider elastic deformation from the motor to ball screw in Equation 4.4, assume zero delays of torsion ($R\theta = x_p$), and apply Newton's second law (Equation 4.5). The term *K* is spring stiffness. *T* is input torque. Finally, [17] presents the ball screw system model as Equation 2.4.

$$J\ddot{\theta} = T - RK\left(R\theta - x_p\right) \tag{4.4}$$

$$M_t\ddot{x}_p = K\left(R\theta - x_p\right) - F_n - \sigma_2\dot{x}_p \tag{4.5}$$

In [19], they consider the entire deformation in the mechanic transmission part and include the delay on torsion. The Equations 2.8 and 2.9 from [19] present elastic deformation from motor to motor coupling. Then, Equations 2.10 and 2.11 present elastic deformation from motor coupling to ball screw. Comparing Equation 2.12 with Equation 2.4, they both use the torque that acts on the table to subtract to the friction torque, that let it be equal to the system mass times acceleration

As stated in the literature review, Equations 2.8 to 2.12 have too many parameters. Furthermore, Equations 2.8 to 2.12 do not provide a method to identify those parameters. In Equation 2.6 and Equation 2.1, they have ignored the mass and inertial that may affect system model accuracy. To sum up, this thesis deploys the system model based on Equations 4.4, 4.5, and 2.4. Then, the system model is proved using Equations 2.8 and 2.9 with some assumptions.

In Equation 2.4, they [17] use a rotational encoder as feedback. Therefore, they used θ to represent the position, but in this project the rotational encoder could not access the servo drive.

Therefore, the linear encoder displacement *x* replaces the *θ* in the system model (Equation 2.4). The relationship between *θ* and *x* and their derivative are displayed in Equation 4.6. The term *L* is the same as *R* in Equation 2.4, which is radians to displacement conversion gain.

$$\begin{aligned} \frac{x}{L} &= \theta \\ \frac{\dot{x}}{L} &= \dot{\theta} \\ \frac{\ddot{x}}{L} &= \ddot{\theta} \end{aligned} \tag{4.6}$$

Rewrite Equation 2.4 and replace *R* with *L*:

$$(J + L^2 M)\ddot{\theta} = T - L \cdot F_f \tag{4.7}$$

The term F_f is friction force, which is the sum of the friction of $(R\sigma_2\dot{\theta} + F_n)$ in Equation 2.4. Submit Equation 4.6 to Equation 4.7 and change notation *T* to T_m motor input torque:

$$\left(\frac{J}{L} + L \cdot M\right)\ddot{x} = T_m - L \cdot F_f \tag{4.8}$$

Equation 4.8 can also be written in fiction torque T_f :

$$\left(\frac{J}{L} + L \cdot M\right)\ddot{x} = T_m - T_f \tag{4.9}$$

$$T_f = L \cdot F_f \tag{4.10}$$

in which T_f is represented in classic LuGre friction by Equations 4.1 to 4.3 by replacing the *v* with $\dot{x}$ as follows:

$$T_f = \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 \dot{x} \tag{4.11}$$

$$\dot{z} = \dot{x} - \frac{\sigma_0 |\dot{x}|}{g(\dot{x})} z \tag{4.12}$$

$$g(\dot{x}) = F_c + (F_s - F_c) e^{-|\frac{\dot{x}}{v_s}|^{\delta_s}} \tag{4.13}$$

The entire ball screw system or plant, including the nonlinear LuGre friction, has been summarized in Equations 4.8 to 4.13. The parameters and variables for Equations 4.8 to 4.13 are in the Appendix Terminology section.

The Equations 2.8 to 2.12 can be used to prove the above system model. The following assumption is made, ***the motor and motor coupling will be a rigid body, ignoring the ball screw driving efficiency, and the ball screw does not have torsion delay***. Therefore, for Equations 2.8 to 2.12, the following condition is added: $T_m = T, \theta_m = \theta_A, \eta = 100\%, J_A = J_M, K_1=0, K_2=0$ and:

$$\frac{x}{K_s} = \theta_m \tag{4.14}$$

Then, Equations 2.8, 2.9, and 2.11 are canceled. The rest of the equation is as follows:

$$(J_A + J_L)\ddot{\theta}_m = T_M - F_T \cdot K_s \tag{4.15}$$

$$M_F \ddot{x} = F_T - F_f \tag{4.16}$$

Rewrite Equation 4.15 and let $J_A+J_L=J$, then:

$$\frac{T_M - J\ddot{\theta}}{K_s} = F_T \tag{4.17}$$

Submit Equation 4.17 into Equation 4.16 and get:

$$M_F \ddot{x} = \frac{T_M - J\ddot{\theta}_M}{K_s} - F_f \tag{4.18}$$

Rewrite Equation 4.18 as follows:

$$K_s M_F \ddot{x} + J\ddot{\theta} = T_M - K_s F_f \tag{4.19}$$

Apply Equation 4.14 to Equation 4.19 and rearrange as follows:

$$\left(\frac{J}{K_s} + K_s M_F\right)\ddot{x} = T_M - K_s F_f \tag{4.20}$$

$$\left(\frac{J}{K_s} + K_s M_F\right)\ddot{x} = T_M - T_f \tag{4.21}$$

Comparing Equation 4.21 and Equation 4.9, the only difference is notation. Therefore, by using Equations 2.8 to 2.12 in [19] proves the ball screw system model equation is feasible.

4.2 Experiment and Identification

In this section, based on the system model and the LuGre model introduced in the previous section, the experimental on the measure the LuGre friction and the experimental on system identification will be introduced and conduct in this section This section will present two subsections, in the first section, the LuGre friction experimental will be introduced and following the parameter identification process. The second subsection, the ball screw model system identification experimental will proceed and follow by ball screw model parameter identification.

4.2.1 LuGre friction identification experiment and parameter identification

To identify the LuGre friction model, a LuGre friction speed curve needs to be obtained, as illustrated in Figure 1.1.11 b. [17], [18] , [19], and [12] all first tried to obtain the Stribeck curve

(Figure 1.1.10 b). [17], [18], and [19] built a speed PI controller to obtain a constant speed. Based on Newton's first law, at a constant speed, input torque is equal to friction force. In [12], they used the same principle, but instead of using the customized PI speed controller, they directly accessed the speed control in the servo drive.

In this project, the servo drive has a speed controller function, but it is only a prototype from the sponsor, and therefore this function has not been fully developed yet, meaning the torque input to the servo motor cannot be obtained from the servo drive (i.e. it cannot monitor the servo drive torque output). Therefore, a customized PID speed controller was developed using the hardware introduced in the chapter Hardware Implementation. The total number of parameters that need to estimate in the LuGre model are illustrated in Table 2.1.

The customized PID the speed controller uses speed as the feedback, and motor torque as the control input. The control input for this setup used voltage instead of torque. The voltage has a direct relationship to torque based on the servo manual [43]. In this setup, the speed cannot be directly obtained, but the position can. Therefore, the speed $\dot{x}$ is obtained via the derivative from the position input *x* as Equation 4.22. The term Δt represents the sampling time. The PID speed controller is presented in Equation 4.23. The term $\dot{x}_{ref}$ represents the reference speed.

$$\dot{x}(t) = \frac{x(t) - x(t - \Delta t)}{\Delta t} \tag{4.22}$$

$$\begin{aligned} T_m &= K_p e(t) + K_i \int e(t)dt + K_d \frac{d}{dt} e(t) \\ e(t) &= \dot{x}_{ref} - \dot{x} \end{aligned} \tag{4.23}$$

Then, equations 4.22 and 4.23 are applied to Teensy 3.5. After this application, the manual tuning of the PID parameters (K_p, K_i, and K_d) can proceed. The tuning reference is constant speed, which is 1mm/s. The PID manual tuning method was first applied in the Ziegler-Nichols method [49]. The K_p is first increased to a certain valve that output starts to same oscillation amplitude. then record K_p as K_u and record the oscillation period as P_u.Then, apply Equation 4.24 [50], below, to obtain the PID parameter. The PID parameter from the Ziegler-Nichols also has been manual adjust to have stable output.

$$K_p = 0.6K_u \qquad K_i = \frac{2}{P_u} \qquad K_d = \frac{P_u}{8} \tag{4.24}$$

The results are displayed in Figure 4.2.1, with $K_p=20$, $Ki=50$, and $K_d=1$. The sampling time Δt = 0.01(s) for Teensy 3.5.

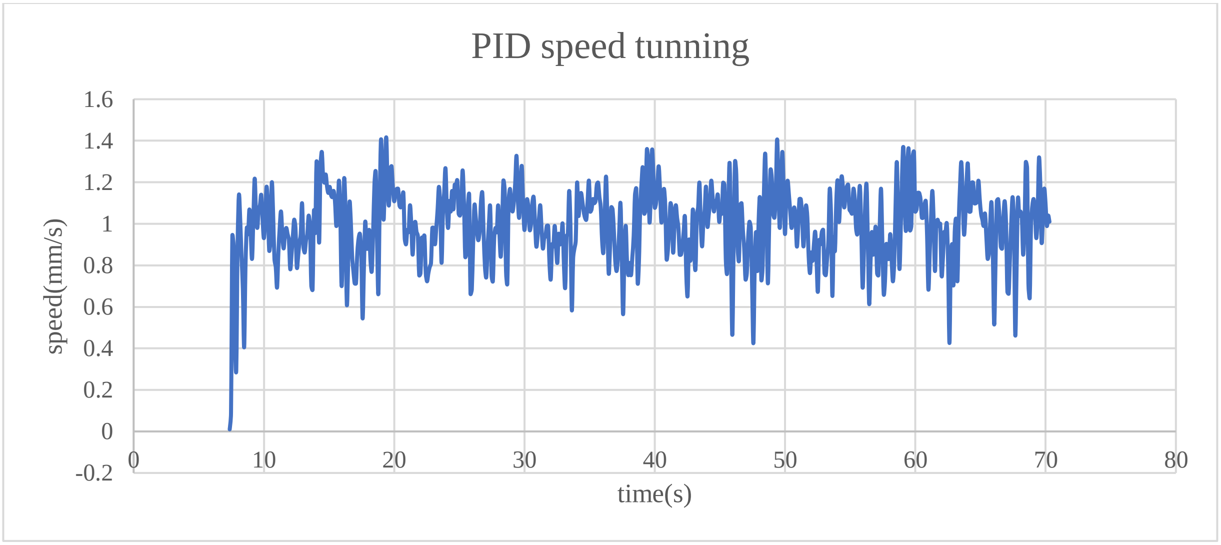


Figure 4.2.1PID speed tuning

In Figure 4.2.1, the controlled speed continues to display much oscillation. This oscillation is due to the speed derivate from the position encoder, which creates noise and in the current amplifier in the servo drive has a torque ripple [18]. Therefore, to obtain friction velocity data (LuGre friction curve), it is necessary to average the measured velocity and input controlled voltage. For example, in Figure 4.2.1, the average speed is 0.99mm/s and the average torque input is 0.77 V.

Applying the PID parameters, each speed constant has chosen from speed range 0.01 to 1. The experiment was conducted three times in three days at a constant temperature room. The results are displayed in Figure 4.2.2.

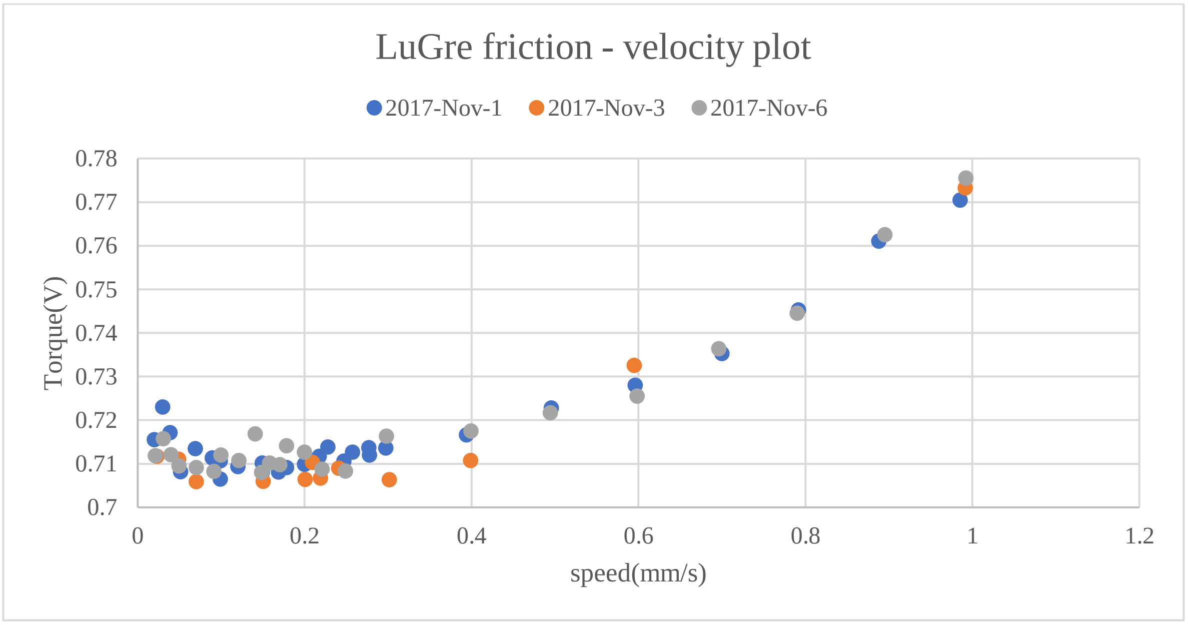


Figure 4.2.2 Experiment: LuGre friction – velocity plot

Figure 4.2.2 helps to establish the Stribeck curve parameters as F_c, F_s, v_s, δ_v, and σ_2. Th figure could also help to validate σ_0 and σ_1 due to its general shape being similar to the LuGre friction character. The next step is based on the data from Figure 4.2.2 as a curve fitting. The curve fitting method uses the Excel polynomial trendline fitting method and a selection order of six (the maximum order number that Excel could support). The results are displayed in Figure 4.2.3.

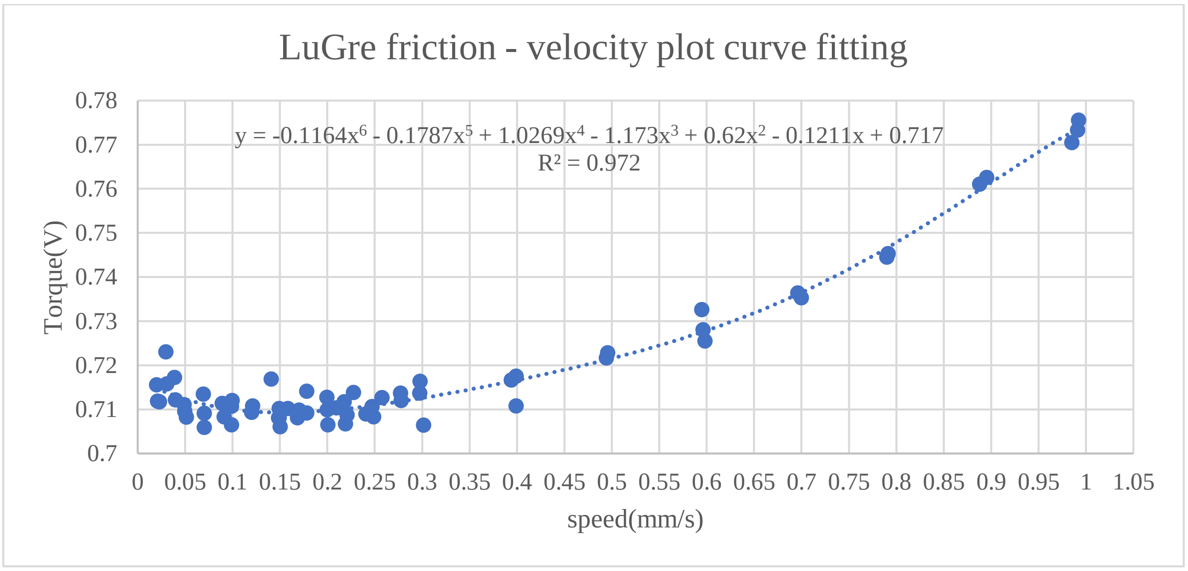


Figure 4.2.3 LuGre friction – velocity curve fitting

The curve fitting polynomial equation (4.25) is displayed in Figure 4.2.3. The *R*-square is 97.2%, which proves the fitted regression is similar to the raw data. By using this fitted regression, applying the Stribeck friction equation (1.4) should result in the parameters Fc, Fs, v_s, δv, and $\sigma 2$. [19] suggest that the equation (1.4) could be used to estimate the parameters F_c, F_s, v_s, δv, and $\sigma 2$ for Equation 4.14 because they all represent the Stribeck curve phenomenon.

$$y = -0.1164x^6 - 0.1787x^5 + 1.0269x^4 - 1.173x^3 + 0.62x^2 - 0.1211x + 0.717 \quad (4.25)$$

By using Equation 4.25 and the data from Figure 4.2.3 as reference, the parameter estimation was made using the Matlab Simulink parameter estimation toolbox. To do this, a Simulink model for the Stribeck curve equation (1.4) was built (see Figure 4.24).

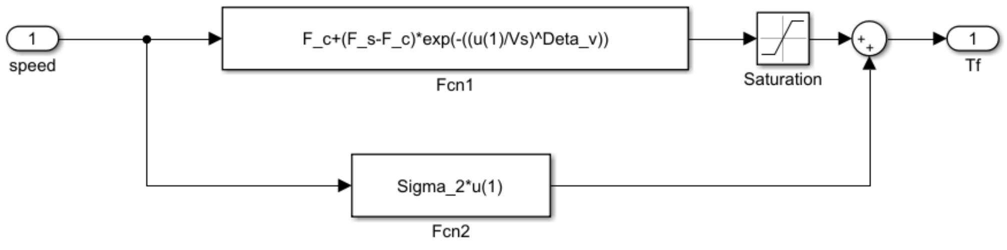


Figure 4.2.4 Stribeck curve Simulink model

The input port is the speed, and the output port is *Tf*, which is the torque in voltage unit. There is also a constraint based on the [12], and equations 1.8 or 4.14 should be at a range of $Fc \leq g(v) \leq F_{s.}$ Therefore, a saturation block was added just after the *Fcn1* block to represent this constraint. To use the parameter estimation toolbox, the initial guesses for F_c, F_s, v_s, δv, and σ_2 are needed. Based on Figure 1.1.10 and [12], the initial guesses for F_c, $F_{s,}$ and σ_2 could be obtained directly from the friction velocity plot, as displayed in Figure 4.2.5.

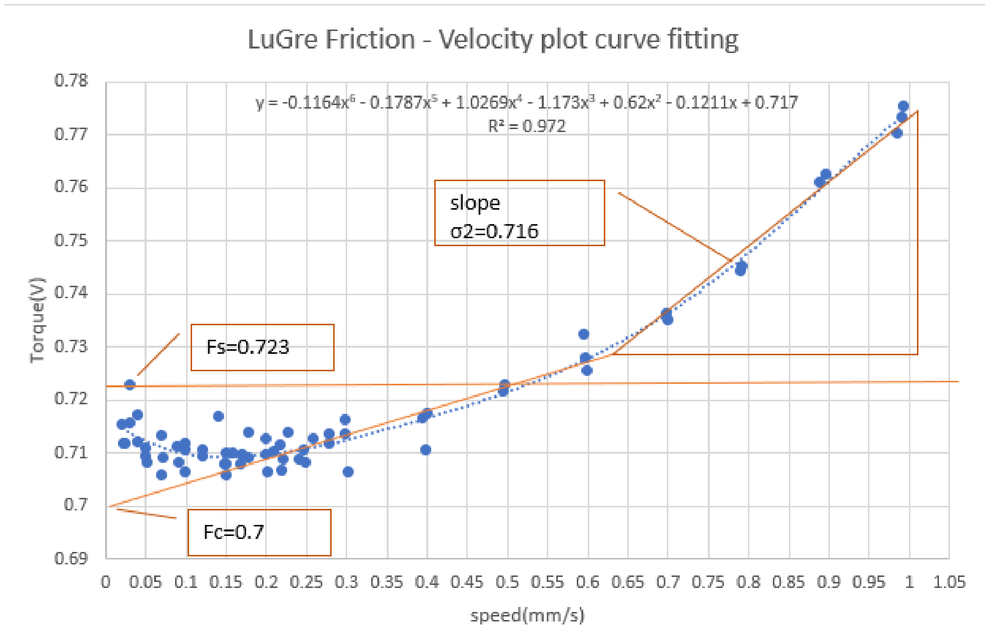
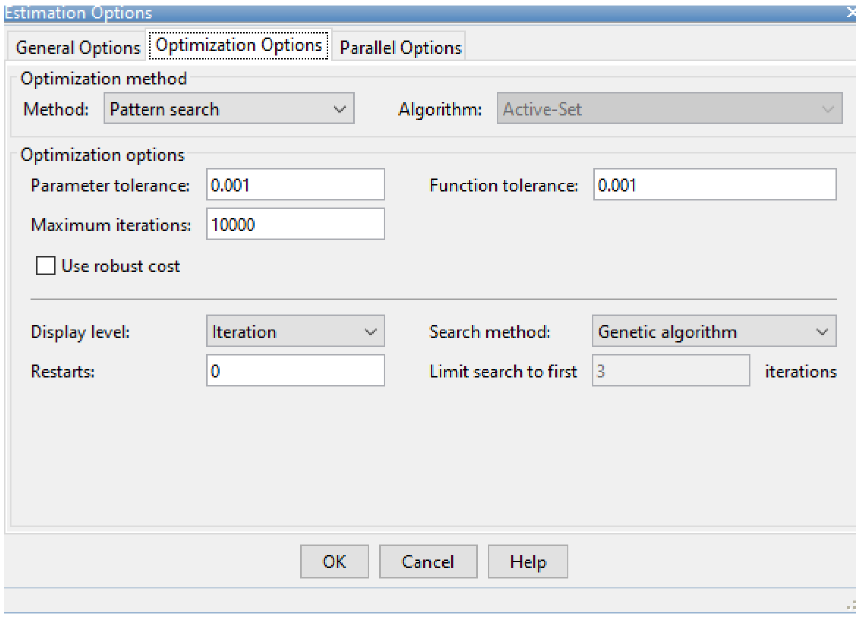


Figure 4.2.5 Initial guesses for parameter estimation

For the parameters v_s and σ_2, the initial guesses start from a minimal number, such as 0.01. The estimation method is displayed in Figure 4.2.6.



Optimization method
Method: Pattern search Algorithm: Active-Set

Optimization options
Parameter tolerance: 0.001 Function tolerance: 0.001
Maximum iterations: 10000
☐ Use robust cost

Display level: Iteration Search method: Genetic algorithm
Restarts: 0 Limit search to first 3 iterations

Figure 4.2.6 Estimation method

The pattern search method was chosen for performance optimization. The pattern search algorithm was selected to the genetic algorithm (GA), as suggested in [19]. A detailed explanation of the GA is [24]. The pattern search method presents the estimation as the tracking problem. The Matlab first computes the scale error by using the linear interpolation [51], as displayed in Equation 4.26.

$$e(t) = \frac{y_{sim}(t) - y_{ref}(t)}{\max\left|y_{ref}\right|} \tag{4.26}$$

Then, the Matlab computes the sum square error as Equation 4.27, including the weight term *w(t)* [51] (by default, the *w(t)* =1).

$$f = \int w(t)e(t)^2\,dt \tag{4.27}$$

Then, the GA attempts to minimize *f*, the cost object function. The parameter boundary conditions for the GA are necessary. Therefore, the boundary conditions for the parameters are displayed in Table 4.2.1. The boundary conditions are selected based on the initial guess for F_c, F_s, and σ_2. Also, multiple runs of the parameter estimation toolbox were used to adjust the boundary conditions to achieve the best result.

	δ_v (unit less)	F_c (V)	F_s (V)	σ_2 (V.s/mm)	v_s (mm/s)
Min	0.5	0.65	0.71	0	0.14
Max	1	0.71	0.73	0.8	0.2

Table 4.2.1 Genetic algorithm boundary condition

Following multiple adjustments, the final estimated results are displayed in Figure 4.2.7. Due to the raw data being too noisy for the estimation, the curve fitted data are used to perform the estimation. In Figure 4.2.7, the top plot is the measured (curve fitted) vs. simulated (parameter estimated) graph with the x-axis being speed(mm/s) and the y-axis being torque (V). The bottom plot is the linear speed with a slope of 1 (the x-axis is time(s), and the y-axis is speed (mm/s)).

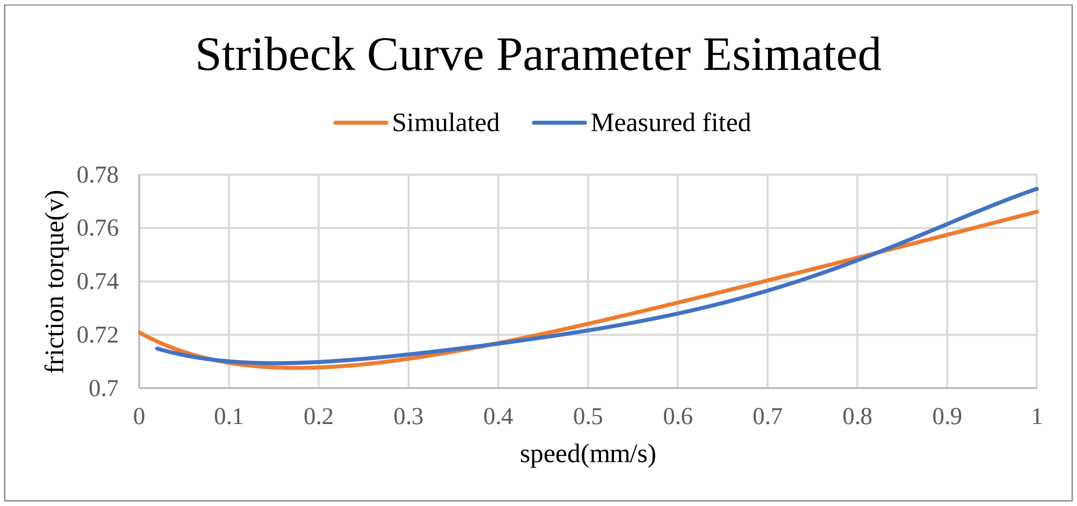


Figure 4.2.7 Stribeck curve parameter estimated result based on curve fitting

After 163 iterations (see Figure 4.2.8), the cost function was minimized to 0.0017. The parameter-estimated process is displayed in Figure 4.2.9. As Figure 4.2.7 illustrates, some inconsistencies remain. These inconsistencies are caused by the noisy speed data, and torque data which obtained from the average raw data. However, the parameters estimated friction velocity curve could still best represent the character of the raw experimental friction velocity curve, such as the character of maximum static friction, Coulomb friction, and viscous friction. In Figure 4.2.9, the initial guess is very similar to the final valve for F_c, $F_{s,}$ and σ_2. This proves the guess method suggested in [12] is correct. Finally, the estimated parameters are displayed in Table 4.2.2. The unit derive is in the Appendix 'unit conversion'.

δ_v (unit less)	F_c (V)	F_s (V)	σ_2 (V.s/mm)	v_s (mm/s)
0.9998	0.67893	0.72088	0.0649	0.15313

Table 4.2.2 Stribeck curve estimated parameter

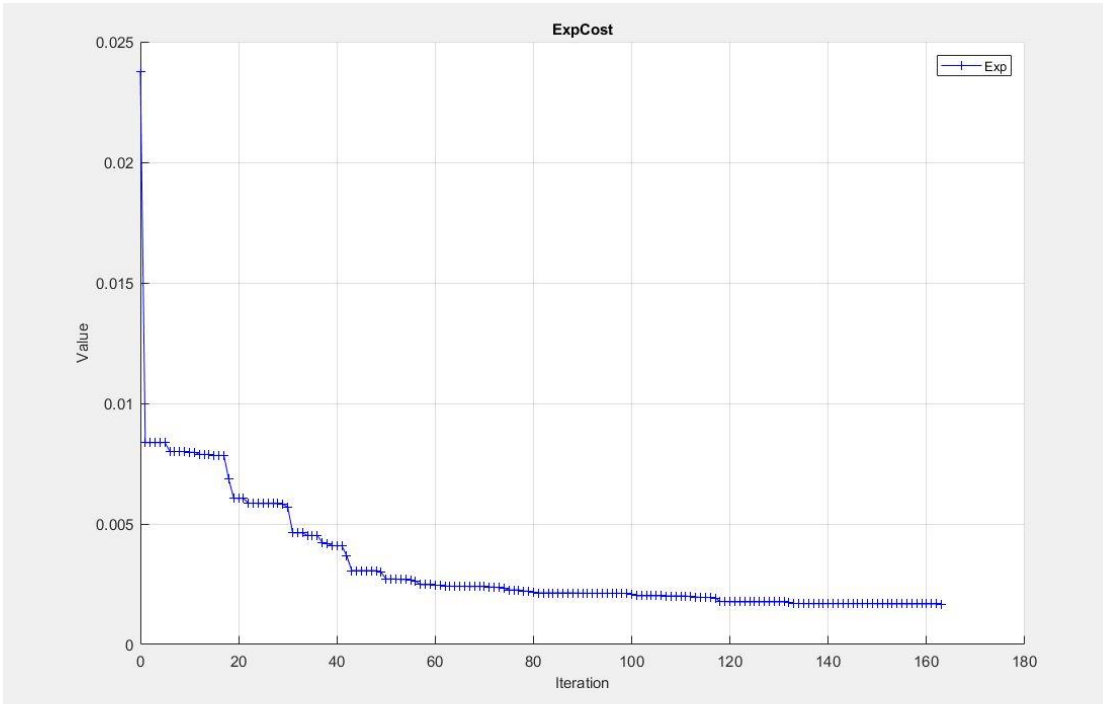


Figure 4.2.8 Minimized cost function plot

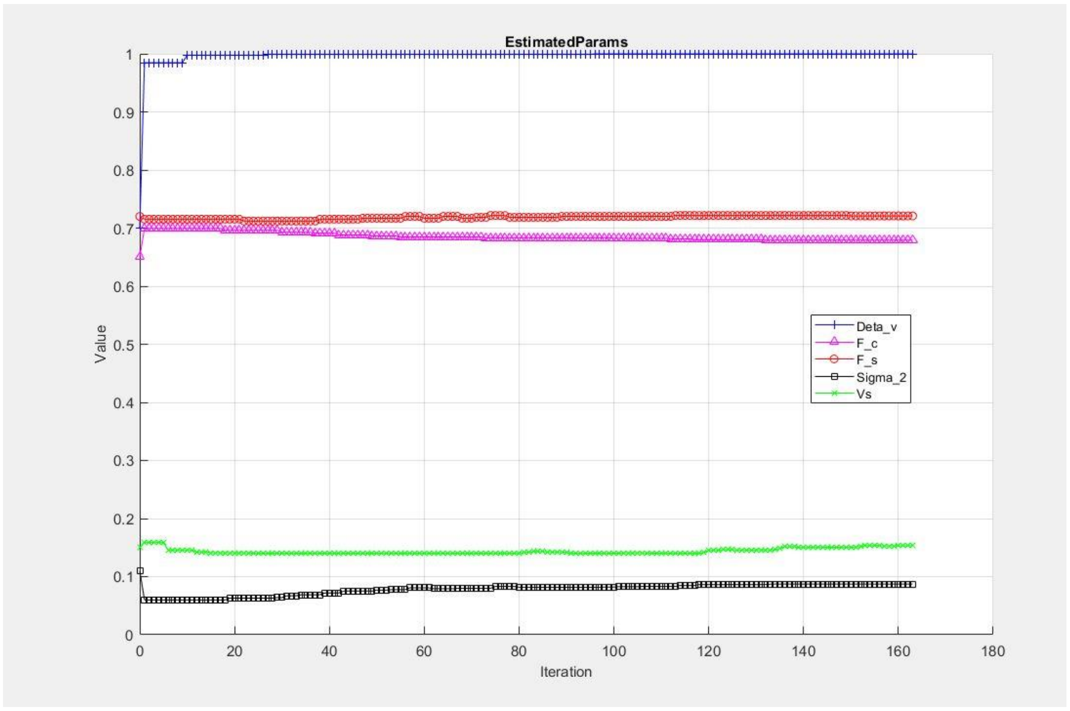


Figure 4.2.9 Estimated parameters

To obtain the σ_0 and $\sigma_{1,}$ a sine wave input is needed, as suggested in [12] and [17]. However, [17] did not provide much detail on how to select the sine wave input. In [12], they suggest the amplitude A should be less than F_c. For this project, the estimated $F_c = 0.67893$. Therefore, A should be less than that number. As [12] suggest, $A=F_c/2$. Therefore, this amplitude was applied to the ball screw system. However, during the test, the input torque-displacement plot cannot have steady-state micro displacement as they suggest in Equation 2.16. The displacement change is minimal, such as 0.000 to 0.001mm, and unstable. The reason for this is the linear encoder, which cannot detect a change that small as it is beyond its resolution. Therefore, in the following few tests, Amplitude A was increased gradually to obtain a torque and displacement plot with a pattern similar to the one in Figure 2.1.3 in [17]. Finally, at amplitude A = 0.63, some tetragon patterns emerged. Figure 4.2.10 displays the result of the sine wave input, in which $A= 0.63$.

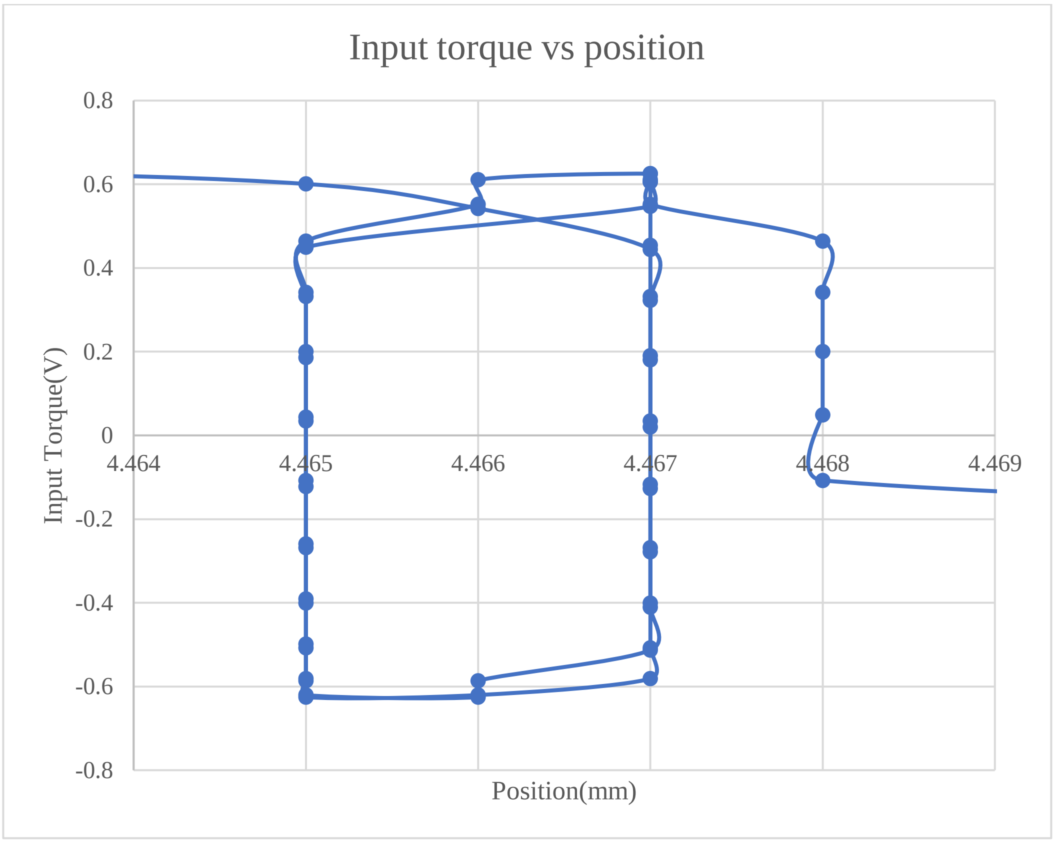


Figure 4.2.10 Sine wave input response

Following [17], the σ_0 is the slope from the top right corner to the lower left corner of the plot. Therefore, in this case, the top right corner is 4.467, 0.625, and the lower left corner is 4.465, -0.625. Therefore, the slope is 625, which led to an initial guess of σ_0 being 625. Also, based on [17], σ_1 could obtained from experimental friction velocity and experimental system response by using know parameter as F_c, F_s, v_s, δ_v, and σ_2.

Based on an initial guess of σ_0, and already knowing parameters F_c, F_s, v_s, δ_v, and σ_2, the estimation process for σ_0 and σ_1 can proceed. To proceed with the parameter estimation for LuGre friction, the Simulink model based on Equations 4.11 to 4.13 can be constructed as follows:

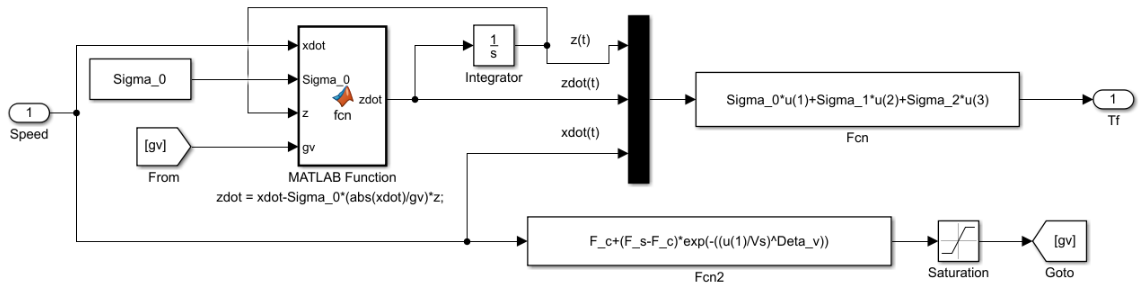


Figure 4.2.11 LuGre Simulink model

The input and output for this model are the curve fitting data from Figure 4.2.3. Comparing Figure 4.2.3 with Figure 1.1.11b, the only missing element is the presiding part. Due to the resolution of the linear encoder, the presiding part cannot be detected. However, the general shape of the experimental friction velocity plot still agrees with the LuGre friction curve.

To obtain the σ_0 and σ_1, the parameter estimation toolbox was applied to the new LuGre friction model using the known parameters displayed in Table 4.2.2. Then, the GA pattern search optimization method with cost function as in equations 4.26 and 4.27 was applied. The initial guess of σ_0 was 625 from the sine wave input. The initial guess for σ_1 was a random number. After the multiple runs of the parameter estimation toolbox, the boundary conditions for GA were adjusted (the boundary conditions were narrowed to σ_0 minimum 600 with maximum 15,000, and σ_1 minimum 7 with maximum 15). Finally, the estimation plot is displayed in Figure 4.2.12.

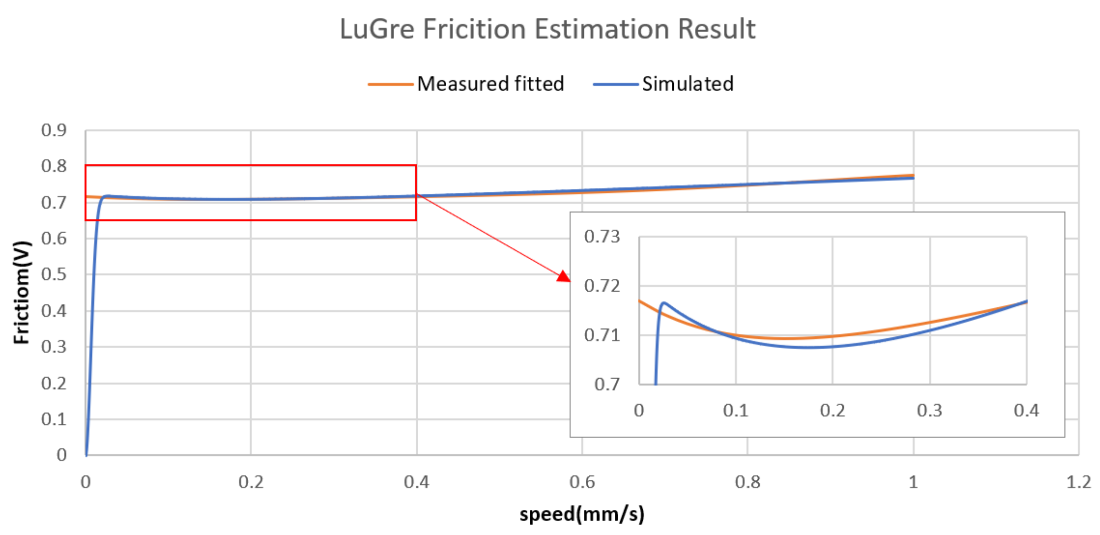


Figure 4.2.12 LuGre friction parameter estimated result based on friction velocity plot curve fitting

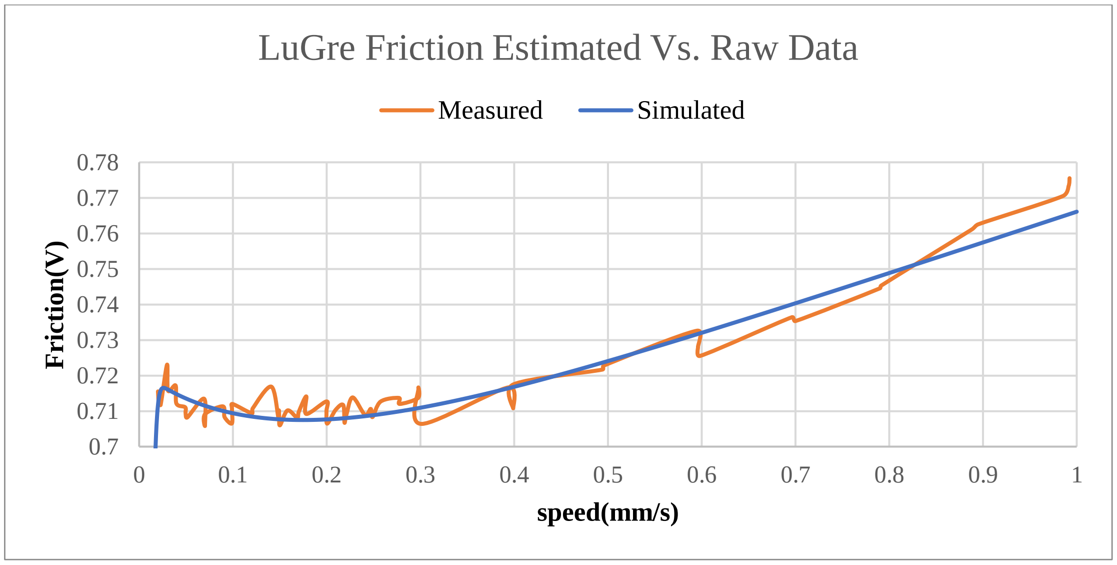


Figure 4.2.13 LuGre friction parameter estimated result base on raw friction velocity data

Furthermore, the raw friction velocity data (measured in 3 days) compared with the LuGre friction parameter fitted data (simulated) is displayed in Figure 4.2.13. The result of the simulated data matched most of the raw experimental data and the fitted curve data. The cost function finally was minimized to 0.0017.

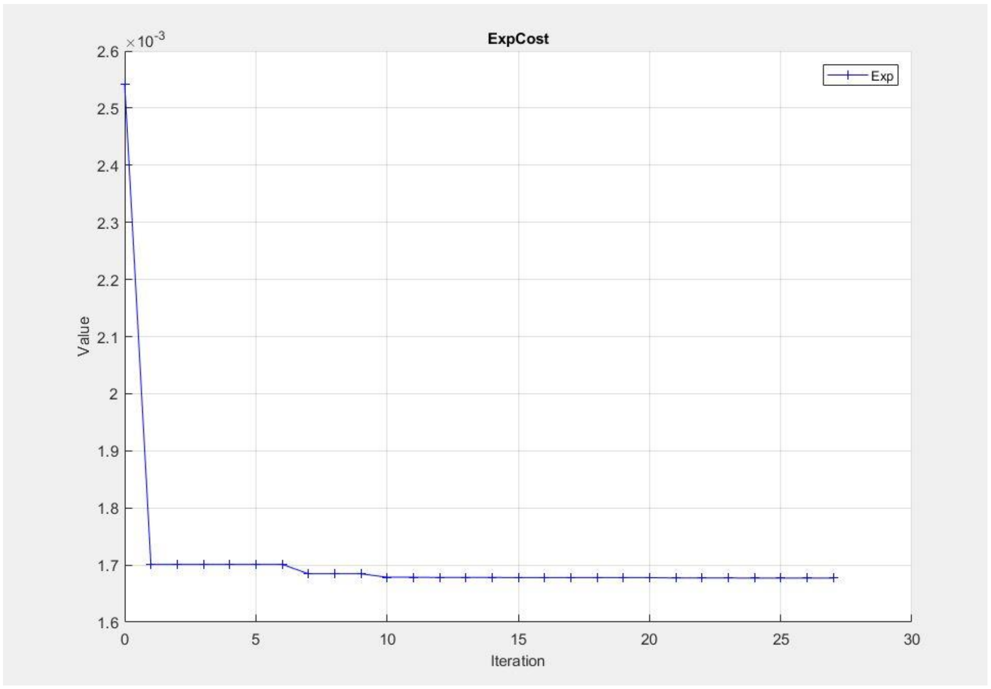


Figure 4.2.14 Minimized cost function plot for the LuGre friction model

Finally, the estimated σ_0 and σ_1 together with the previous parameter estimation (Table 4.2.2) are displayed in Table 4.2.3, below.

δ_v (unitless)	F_c (V)	F_s (V)	σ_0(V/mm)	σ_1(V.s/mm)	σ_2(V.s/mm)	v_s (mm/s)
0.9998	0.67893	0.72088	14998	14.998	0.0649	0.15313

Table 4.2.3 LuGre friction estimated parameter

The final result not only agreed with the experimental data, but also predicated the static presiding region. The final result also prove its repeatability.The estimated parameter has been catching the critical features of the LuGre friction model displayed in Figure 1.1.11b. Based on Figure 4.2.12 and Figure 1.1.11b, the characters for the LuGre model based on the estimated data are displayed in Figure 4.2.15. The right plot illustrates the entire LuGre friction velocity plot. The LuGre friction differs from the Stribeck curve because the friction begins as zero torque in the static presiding region and increases as a parabola to the maximum static friction F_s and enters the Kinetic Stribeck curve region. Finally, the friction torque moves to the viscous friction region.

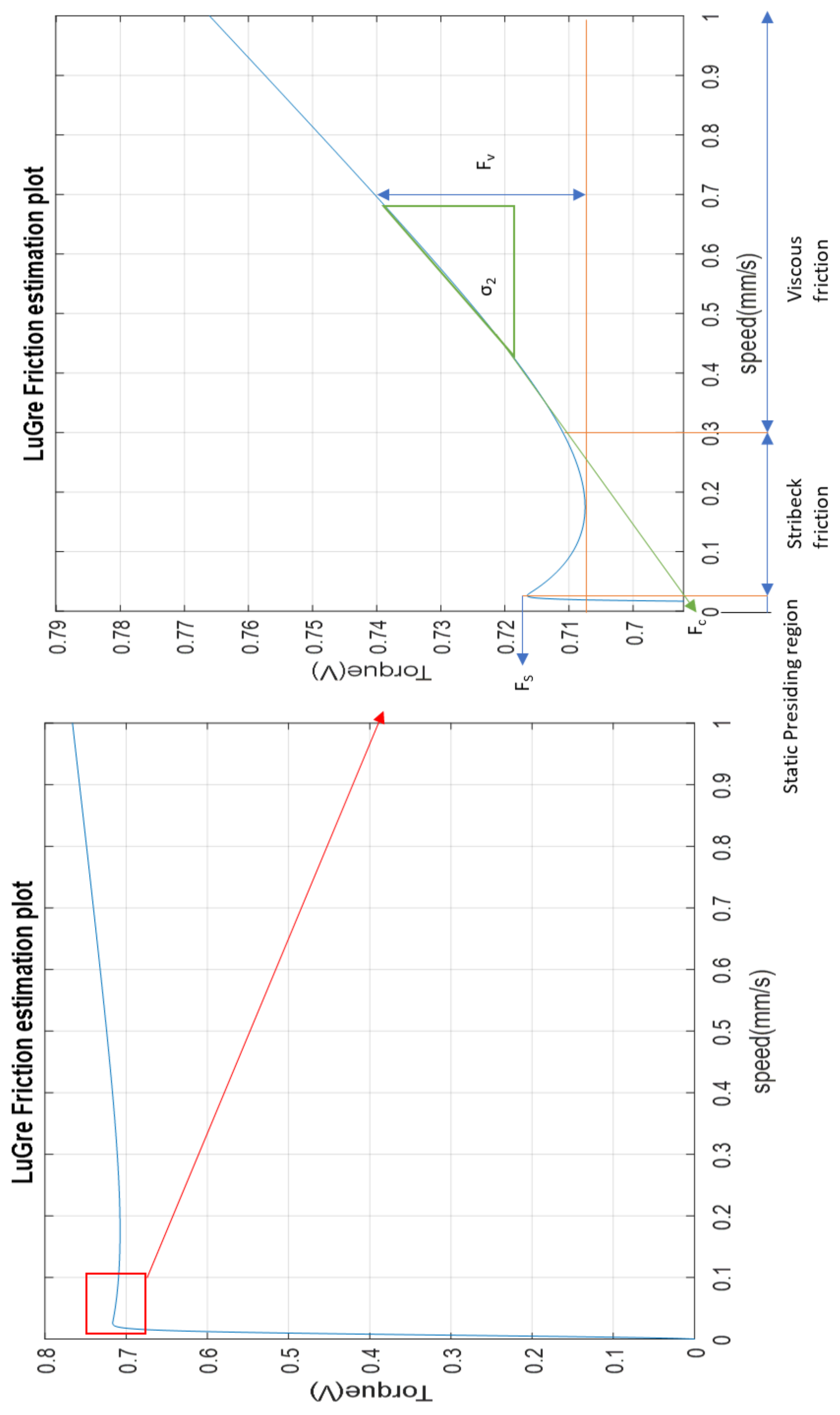


Figure 4.2.15 LuGre friction plot and characteristic identification

4.2.2 Ball screw system identification experiment and parameter identification

After identifying the LuGre friction term T_f, the next objective is to identify the ball screw model parameters. Based on the system model Equation 4.9, only the first part of the equation $\left(\frac{J}{L} + L \cdot M\right)$ needs to be identified, and this leads to the parameters needed, which are J – the combined moment of inertia of ball screw and motor, and L – the radians to displacement conversion gain, and M – the mass of the sliding table.

The first parameter is L. This is obtained by directly using the manual 'JOG' mode on the servo drive. This 'JOG' mode is a function that allows users to manually setup the motor RPM and lets the motor move counter-clockwise or clockwise by pressing the up and down button on the servo drive. This JOG mode is different from the speed control mode in which the speed command is given from an analog voltage signal. In JOG mode, servo drive could not output or obtain data from microcontroller, and only manual adjustments can be made by using the up and down button on the servo drive. However, the RPM of the motor is visible on the small screen on the servo drive. The primary usage of the JOG mode is to allow the user to manually adjust the motor, which lets the user manually adjust the position of the ball screw sliding table without using a PC, microcontroller, or PLC.

Therefore, by using the JOG mode, the constant RPM of the motor can be obtained from the drive screen, and the related linear speed obtained by using the linear encoder via the microcontroller. The linear encoder can only represent the displacement, however; therefore, the speed is derived from the displacement data.

The experiment was conducted by choosing 15 RPM data points and recording the steady-state linear encoder speed. The RPM unit was converted to rad/s, and the rotational angular speed vs. linear speed plot obtained is displayed in Figure 4.2.16.

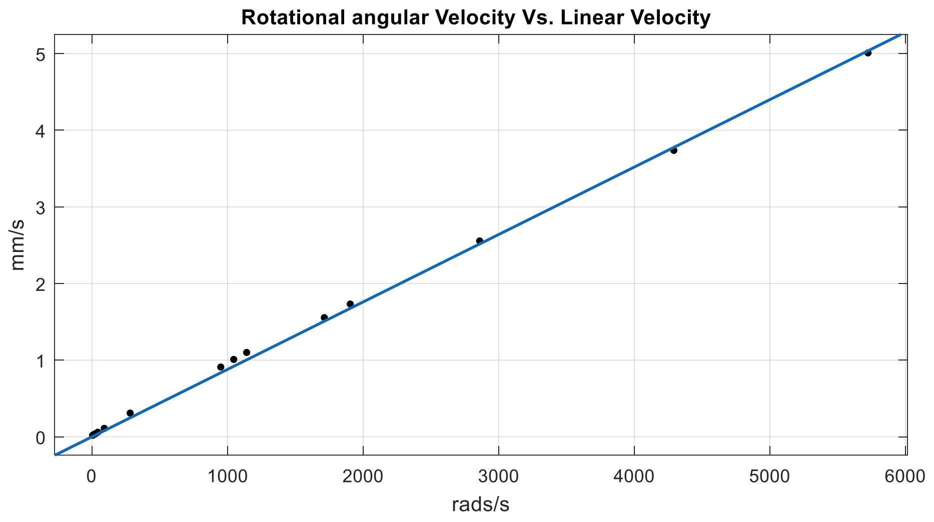


Figure 4.2.16 rad/s Vs mm/s

The data in Figure 4.2.16 is represented by a very straight line. Therefore, the assumption of zero delay of torsion is proved. The relationship between θ and x as Equation 4.6 is proved also. By using the curve fitting toolbox from the Matlab, based on Equation 4.6, the L can be estimated as

$$L = 0.0008797 \frac{mm}{rad} \tag{4.28}$$

The R-square for the estimation is 99.91%. Therefore, the estimation is accurate. Based on Equation 4.8, the term $\left(\frac{J}{L} + L \cdot M\right)$, which combines all three parameters can be assumed to be, in one term, $\left(\frac{J}{L} + L \cdot M\right) = JLM$. Based on Equation 4.8, the plant model is expressed as Equation 4.29, and the Simulink based on Equation 4.29 is displayed in Figure 4.2.17.

$$JLM\ddot{x} = T_m - T_f \tag{4.29}$$

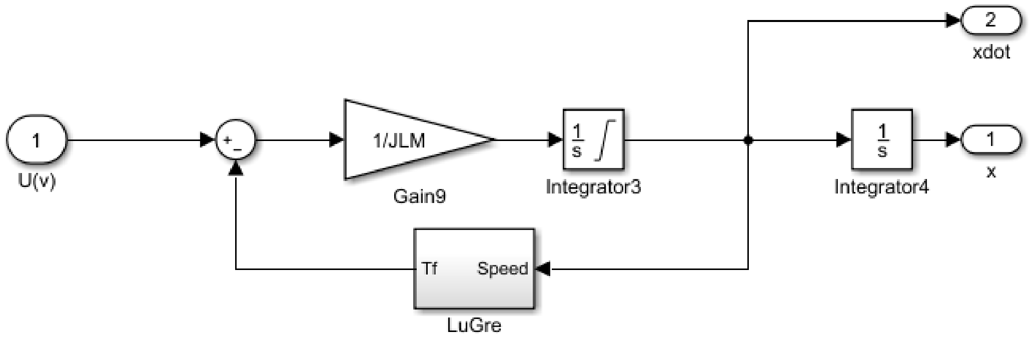


Figure 4.2.17 Plant model

Figure 4.2.17 illustrates that the input of the system is the torque in voltage. The LuGre friction model block as a Figure (4.2.11). The input torque subtracts LuGre friction torque through the inverse of the *JLM* term, and then it becomes acceleration. The speed is obtained from the acceleration integration. However, the integration has a saturation limit due to the AC servo motor having a rated speed of 3000 RPM, as illustrated in Table 3.1.1. The AC servo drive does not allow the AC servo motor to go faster than 3000 rpm. Therefore, converting the 3000RPM to mm/s by using Equation 4.28 revealed the upper and lower saturation values as $\pm$ 25.2mm/s. After determining the speed $\dot{x}$, this is then the input to the LuGre friction model to produce the friction. Finally, speed $\dot{x}$ is integrated again to determine the position *x*.

After building the Simulink model, three ramp torque input tests were conducted on the ball screw system, and the corresponding position and speed were recorded. The ramp function inputs were selected using the following equations:

$$\begin{aligned} &exp1: T_m = 0.027t + 0.0049 \\ &exp2: T_m = 0.00195t + 0.0049 \\ &exp3: T_m = 0.0139t + 0.0049 \end{aligned} \tag{4.30}$$

The three ramp function experiments are called exp1, exp2, and exp3. Again, the parameter estimation tool in Simulink was used to estimate the *JLM*. The input for the system is torque in voltage, and the output for the system is position x and speed xdot ($\dot{x}$). The optimization method for the parameter estimation is the simplex search method. The reason this method was chosen is that the simplex search does not need to specify the parameter boundaries. This method is useful when the information of the estimation term is limited.

Several attempts on the estimation of the *JLM* term has been performance. However, the parameter estimation tool could not successfully estimate the *JLM* based on the exp1, exp2, and exp3 data. The reason for this is due to the inaccurate estimation of the LuGre friction parameters and especially the σ_0 and σ_1. The terms σ_0 and σ_1 were obtained from the prediction of the unable measured presiding stage(Figure 4.2.15 static presiding region), therefore the inaccurate σ_0 and σ_1 caused estimation failure at the ball screw system level. Also, [17] suggest that adjusting the parameter of σ_0 and σ_1 is necessary at the system level. Therefore, the σ_0, σ_1, and *JLM* are needed for the simplex search parameter estimation process. The previously estimated σ_0 and σ_1 are used as an initial guess for the simplex search method. The initial guess for *JLM* was selected as default. In the simplex search, the cost function was selected as the sum square error in Equation 4.31 [52].

$$\begin{aligned} &e(t) = y_{ref}(t) - y_{sim}(t) \\ &F(x) = \sum_{t=0}^{t_N} e(t)^2 \end{aligned} \tag{4.31}$$

The parameter estimation was processed again by using the three experiments' data at the same time. The results are displayed in Figure 4.2.18 to Figure 4.2.20 , and the minimized cost function is presented in Figure 4.2.21. In Figure 4.2.21, total 6 cost functions have been minimized, the top three of which are exp1, 2, and 3 speed data, and the bottom three are position data.

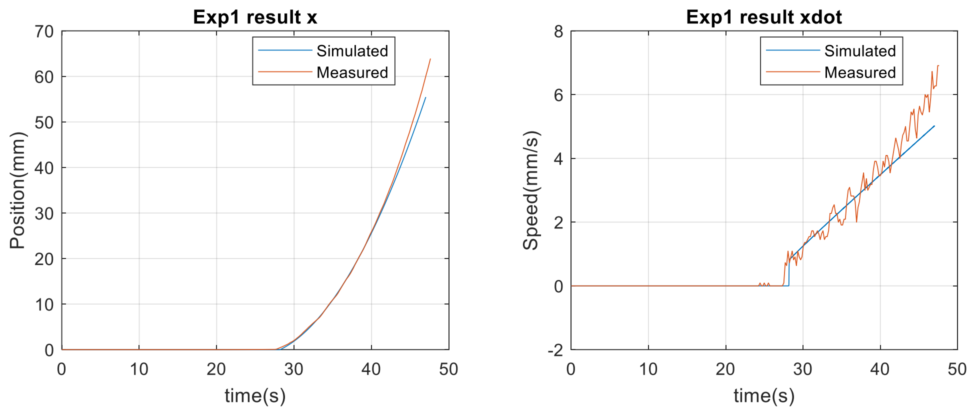


Figure 4.2.18 Exp1 estimation result

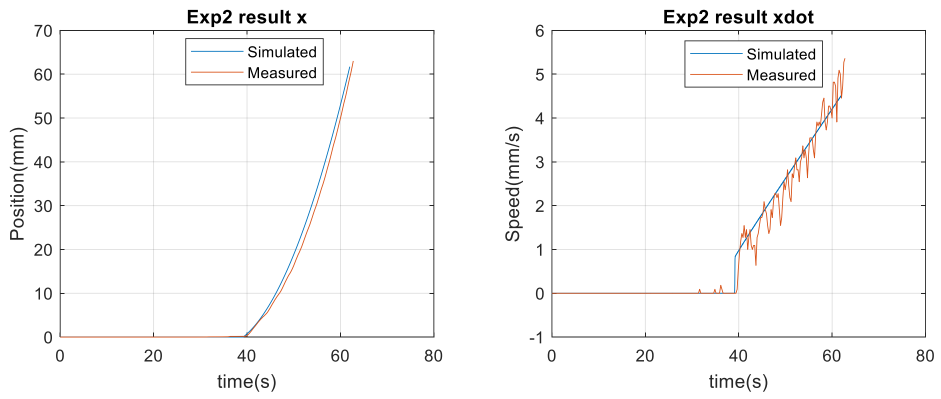


Figure 4.2.19 Exp2 estimation result

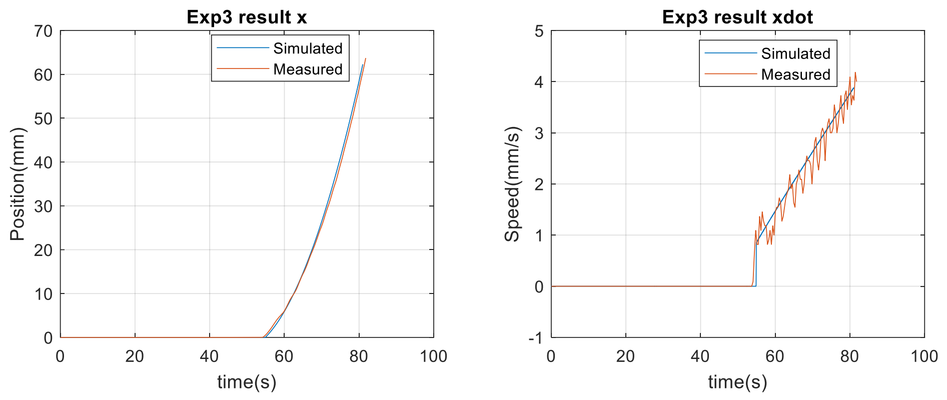


Figure 4.2.20 Exp3 estimation result

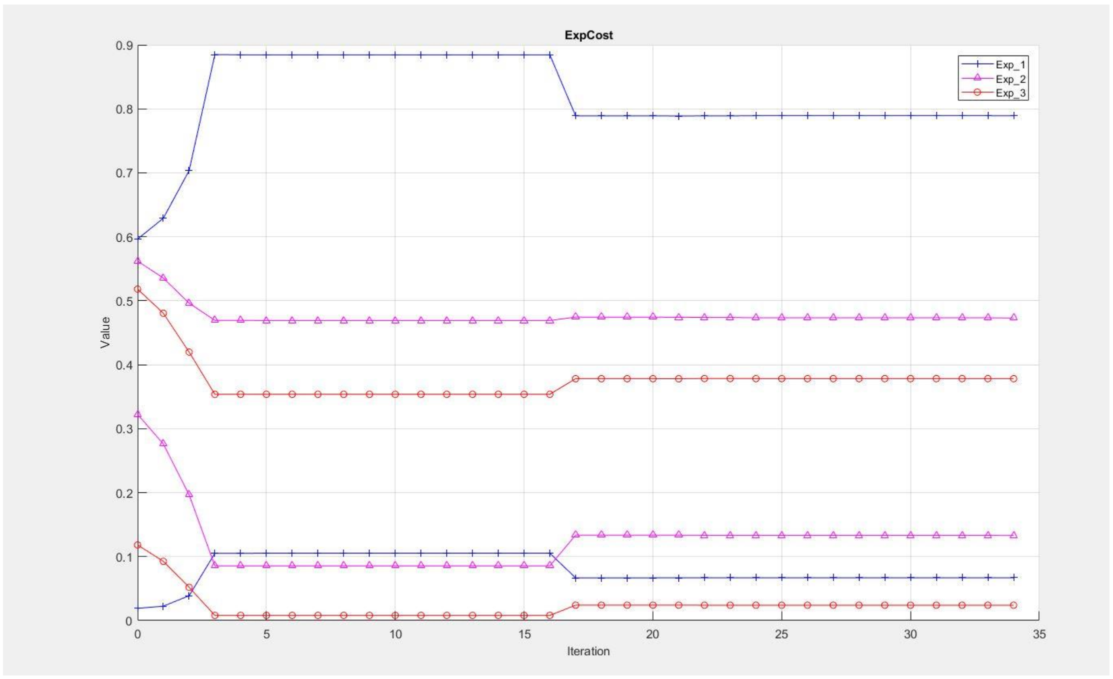


Figure 4.2.21 Minimize cost function for both speed and position

As Figure 4.2.21 illustrates, after 34 iterations, the cost functions have been minimized to less than function tolerance. The final estimated results are in Table 4.2.4.

σ_0(V/mm)	σ_1(V.s/mm)	JLM (V. s^2 /mm)
13882	8.6776	0.007046

Table 4.2.4 Parameter estimation for ball screw system experiment

In Figure 4.2.18 to Figure 4.2.20, the estimation is acceptable. The estimation model could accurately represent the characteristic phenomenon of the static friction to kinetic friction transformation. The speed outputs from the simulation are similar to the actual experimental data. Therefore, the plot proves the accuracy of the parameter estimation.

To obtain the actual number of *J* and *M* is difficult based on the following equation:

$$\left(\frac{J}{L} + L \cdot M\right) = JLM = 0.00704\,\frac{V \cdot s^2}{mm} \tag{4.32}$$

Although, the term *L* has been determined previously, in Equation 4.32, it has two unknowns, which are difficult to solve.

Therefore, combining tables Table 4.2.4 and Table 4.2.3, the entire ball screw system with LuGre friction parameter is summarized in Table 4.2.5. Now, all the parameters are identified, and the system model is validated using exp1, 2, and 3. Finally, cost result is illustrated in Figure 4.2.21.

Ball screw system overall parameters	
δ_v (unitless)	0.9998
F_c (V)	0.67893
F_s (V)	0.72088
v_s (mm/s)	0.15313
σ_0(V/mm)	13,882
σ_1(V.s/mm)	8.6776
σ_2(V.s/mm)	0.0649
JLM(V.s^2/mm)	0.007046
L(mm/rad)	0.0008797

Table 4.2.5 Ball screw system entire parameters

5 CHAPTER CONTROL SYSTEM AND CONTROL EXPERIMENTAL TESTING

5.1 CONTROL SYSTEM SIMULATION

In this section, the developed control system is discussed and implemented as a simulated ball screw system. Based on the system parameters and system model discussed in the Chapter 4, the simulated ball screw system was built in Matlab Simulink. The control algorithm was created based on the literature review in Chapter 2. Three controllers were simulated for this project. Finally, the performance of three controllers is discussed.

5.1.1 PID controller

The PID controller is used as a benchmark for the project. The plant mode for the PID controller is illustrated in Figure 4.2.17 and discussed in Section 4.2.2. The PID controller is a well-known feedback controller that uses the error between the reference signal and the actual signal. The control output is based on the present error (K_p), the past accumulated error (K_i), and the predicted future error (K_d). For this project, the PID controller was implemented as illustrated in Figure 5.1.1.

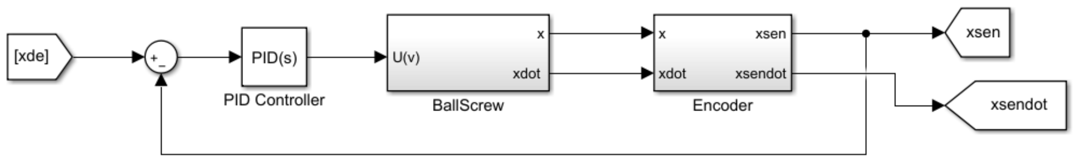


Figure 5.1.1 Ball screw system PID controller block diagram

The input of the system is xde – the desired position. The output is xsen – the simulated encoder measures the position, and xsendot – the simulated encoder measures the speed. The ball screw block is exactly represented in the block diagram in Figure 4.2.17, and it outputs the actual x position and xdot speed. The actual position and speed then go through the encoder block to simulate the accuracy of ±5μm of the linear encoder (Table 3.1 JCXE5 encoder specification). Then, the PID controller takes the error as the input. The errors are between xde, to the sensing position xsen. The output for controller is the torque U.

In the encoder block illustrated in Figure 5.1.2, a uniform random number with a minimum and maximum range of ±0.005mm simulated the inaccuracy of the linear encoder. Also, to simulate the reading of the encoder, a round function was added to make sure the speed and position reading is within the range of 1um (0.001mm). The tuning parameters for this control are $K_{p,}$ $K_{i,}$ and K_d from the PID controller block.

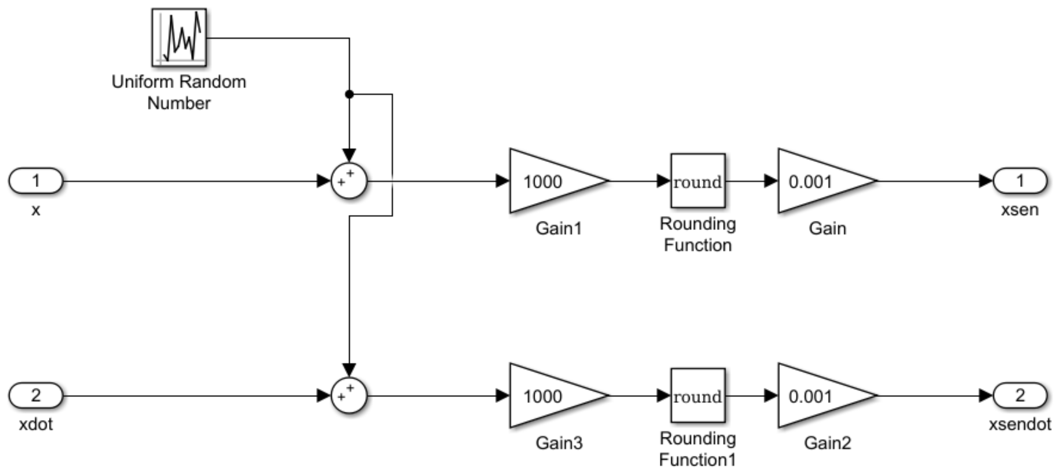


Figure 5.1.2 Encoder block

5.1.2 Positioning PD controller with friction estimation observer

The Positioning PD controller with friction estimation observer (PD-F) [18] was first introduced in the literature review (Section 2.2.2). In their [18] work, they propose a control law that combines the friction estimation, the PD controller, and the system inertia term. The primary purpose of this controller is to compensate for the nonlinear friction that may affect the overall system response due to the friction parameters estimated in the previous chapter being proximate and the term *z(t)*-bristle displacement being unmeasurable. Therefore, a friction observer is needed to overcome the inaccurate friction parameter estimation. The control law, including the friction observer, is represented in Equations 5.1 and 5.2.

$$\begin{aligned} \frac{d\hat{z}}{dt} &= \dot{x} - \frac{|\dot{x}|}{g(\dot{x})}\sigma_0 \hat{z} + k_z e, k_z > 0 \\ \hat{T}_f &= \sigma_0 \hat{z} + \sigma_1 \frac{d\hat{z}}{dt} + \sigma_2 \dot{x} \end{aligned} \qquad (5.1)$$

$$\begin{aligned} u(t) &= JLM \cdot \ddot{x}_d + K_p e + K_d \dot{e} + \hat{T}_f \\ e &= x_d - x \\ \dot{e} &= \dot{x}_d - \dot{x} \end{aligned} \qquad (5.2)$$

Equation 5.1 represents a nonlinear friction observer. The term $K_z e$ is the observer correction term based on the errors from the position. The function $g(\dot{x})$ refers to Equation 4.13. The error is based on the difference between the desired position x_d and the encoder output position x. The control Equation 5.1 begins with the *JLM* term with desired acceleration. The term *JLM* replaces the original *J* term in Equation 2.64 to match the plant equation. Then, the PD controller term corresponds with the PID controller. Finally, the friction observer corrects and compensates the nonlinear fiction. Based on Equations 5.1 and 5.2, a Simulink model was made, and is illustrated in Figure 5.1.3 and Figure 5.1.4.

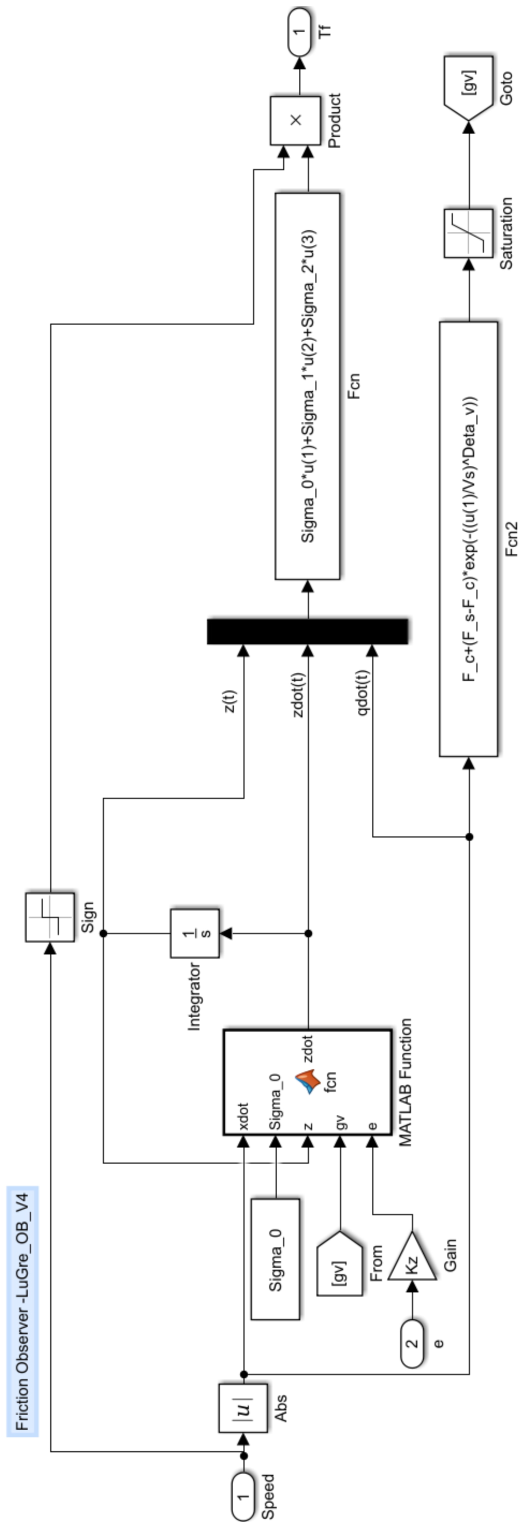


Figure 5.1.3 LuGre friction observer

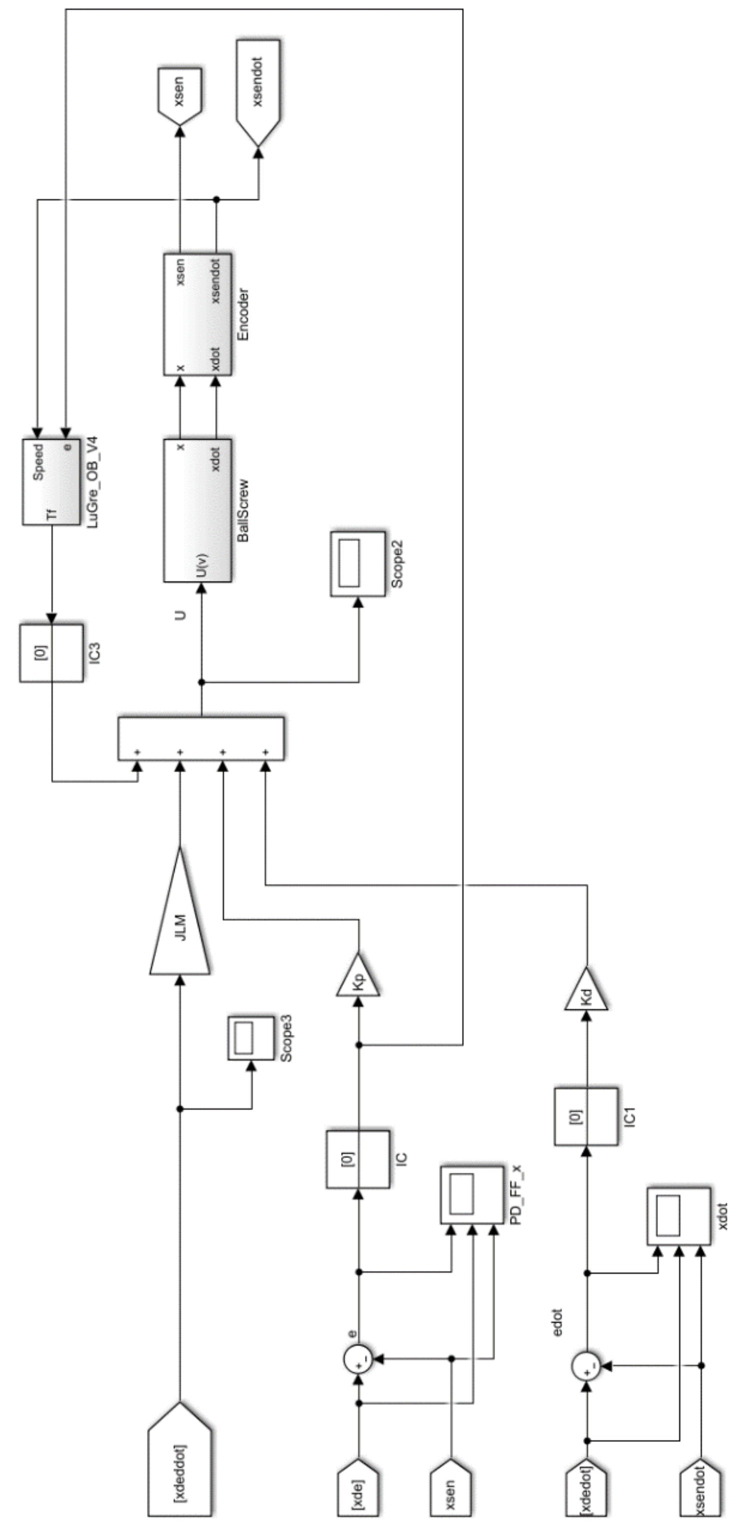


Figure 5.1.4 PD control with friction observer block diagram

Figure 5.1.3 presents the friction observer block LuGre_OB_V4. The LuGre friction observer block is based on Figure 4.2.11 with serval modifications. The first modification adds the observer correct term $K_z e$ into the zdot function. The second modification addresses the sign problem in the friction. In this LuGre friction, the assumption is made that the LuGre friction model is symmetric to the speed data. Therefore, as the speed is negative, the friction is negative also, which follows Figure 1.1.11b. In Figure 5.1.3, the absolute term ensures the speed is always positive, and signs of the friction are determined by the speed sign by using the sign block. In Figure 5.1.4, xdeddot and xdedot illustrate the desired acceleration and desired speed from the trajectory generation. The friction observer is based on the feedback encoder speed and error between the desired and actual position. The tuned parameters are K_p and K_d. The sampling time selected for this Simulink model was 0.0001s. A faster sampling time was tested also. However, the faster sampling time caused errors on the ball screw integration part. The encoder block is the same block as that used in the PID controller.

5.1.3 Positioning PD controller with friction estimation observer and a disturbance observer

The Positioning PD controller with friction estimation observer and a disturbance observer (PD-DOB) originates from the same literature that introduced the Positioning PD controller with friction estimation [18]. In their work, they state that the above method has difficulties dealing with sine wave input. The problems mainly occur during the step command, as acceleration is not changed. However, in the sine wave case, the acceleration is always changing. Therefore, if there is an inaccurate measure of the system parameters, the parameter associated with acceleration affects the performance. Based on [18], the cause of a primary tracking error can be derived from following.

Rewrite control law (5.2), replace *JLM* term with $J L \hat{M}$, which represents the identified parameters and sub them into the plant equation (4.29). Assume that:

$$\begin{aligned} JLM &= JL\hat{M} + \Delta JLM \\ \tilde{T}_f &= T_f - \hat{T}_f \\ e &= \ddot{x} - \ddot{x}_d \end{aligned} \tag{5.3}$$

Then:

$$\begin{aligned} \ddot{x} &= \frac{JL\hat{M} \cdot \ddot{x}_d + K_p e + K_d \dot{e} + \hat{T}_f}{JLM} - T_f \\ JLM\ddot{x} + T_f &= JL\hat{M} \cdot \ddot{x}_d + K_p e + K_d \dot{e} + \hat{T}_f \\ JL\hat{M}\ddot{x} + \Delta JLM\ddot{x} + T_f - JL\hat{M}\ddot{x}_d + \hat{T}_f &= K_p e + K_d \dot{e} \end{aligned} \tag{5.4}$$

Rearrange Equation 5.4 with the assumption (5.3) as Equation 5.5:

$$\tilde{T}_f + \Delta JLM \cdot \ddot{x} = K_p e + K_d \dot{e} + JL\hat{M} \cdot \ddot{e} \tag{5.5}$$

In Equation 5.5, the primary tracking errors stem from the friction estimation error and mass inertia parameter error. Therefore, a disturbance observer is needed, not only to cancel out the error from the friction estimation and mass inertia bias but also to cancel out the external disturbance [28]. The fundamental structure disturbance observer is presented in Figure 2.2.4. By using the system input minus the inverse of the plant model nonlinear transfer function, the estimated disturbance $\hat{d}$ is obtained. Next, the control command *c* adds $\hat{d}$ to become the system input *u*. In this case, they [18] implement this disturbance observer (DOB) with their friction observer as in Figure 2.2.3. In this study, the plant mode is very similar to [18]'s plant model. Therefore, according to [18], the control law based on Figure 2.2.5 is represented as below:

$$u(t) = JL\hat{M} \cdot \ddot{x}_d + K_p e + K_d \dot{e} + \hat{T}_f + u_d(t) \tag{5.6}$$

Where $u_d(t)$ estimates the disturbance as:

$$u_d(t) = u(t-T) - u_m(t-T) \tag{5.7}$$

The term T is the sampling time, the delays for u and u_m are needed because the u_d cannot compute instantaneously. The term u_m as the inverse of plant is presented as follows:

$$u_m = JL\hat{M} \cdot \ddot{x} + \hat{T}_f \tag{5.8}$$

If using plant model (4.29) minus Equation 5.8 and applying assumption (5.3), and replacing T_m to u, then:

$$\begin{aligned} \left(JLM - JL\hat{M}\right)\ddot{x} + (T_f - \hat{T}_f) &= u - u_m \\ \Delta JLM \cdot \ddot{x} + \tilde{T}_f &= u - u_m = u_d \end{aligned} \tag{5.9}$$

In Equation 5.9, u_d is the tracking error present in Equation 5.5. Therefore, the overall control u is considered in the tracking error as a compensator to overcome the inaccurate estimation of the friction force and mass inertia *JLM* term. The term $\hat{T}_f$ is the friction observer present in Equation 5.1, and $\ddot{x}$ is obtained from the derivation of the speed $\dot{x}$.

The Simulink model was built as illustrated in Figure 5.1.5. The input of the control system is desired acceleration (xdeddot), speed (xdedot), and position (xdedot). In the test, the Simulink model Figure 5.1.5 had issues with U_d. The U_d became unstable and kept oscillating with a high amplitude and frequency. The solution is to uses a lowpass filter [18]. Therefore, in Figure 5.1.5, a Butterworth low pass filter was added to U_d to filter out the noise and make system realizable.

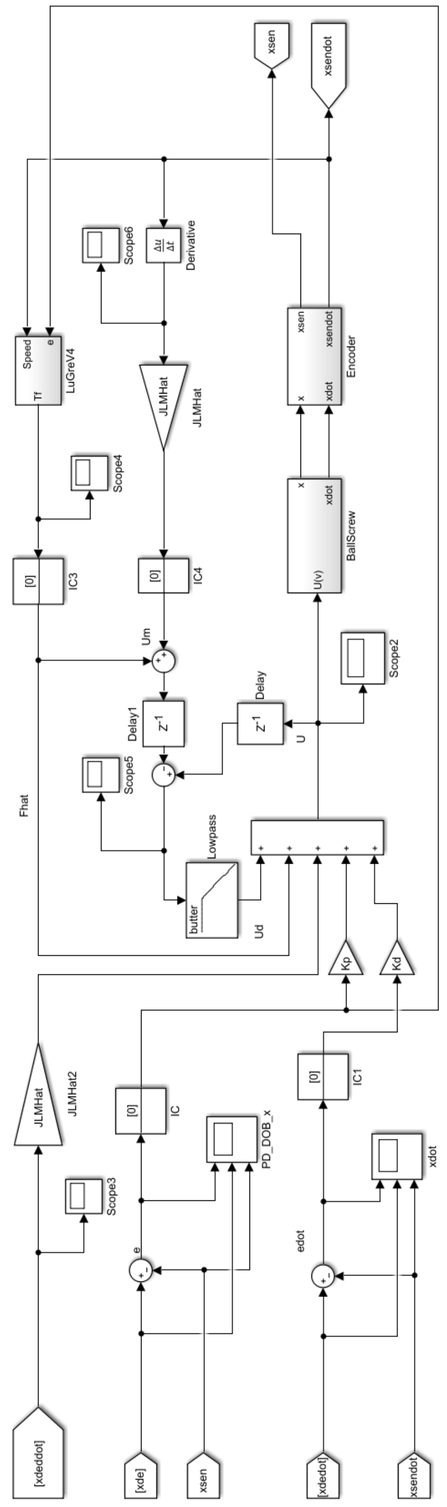


Figure 5.1.5 PD control with friction estimation and a disturbance observer block diagram

5.1.4 Trajectory generation

Instead of using the single step reference, the generated travel path was used in this project. In the machine or 3D printer industry, trajectory generation is critical. Instead of using the single position command, a series of position commands are given to the motor, which makes the sliding table follow the series of reference positions to the final position.

The smooth trajectory generation method used was followed by the trajectory generation subsection, which was discussed in the literature review. The trajectory generation follows Equations 2.27 to 2.29 and 2.31. The trajectory generation in these equations attempts to simulate the scenario of moving an object from a stop position (zero speed) to following a smooth path to reach the desired position to zero speed once more. Therefore, based on the assumption of Equations 2.25 and 2.26, the assumptions for smooth trajectory generation in this project are as follows:

Condition 1: The initial position x_0 and final position x_f and time of travel t_f are given as follows:

$$\begin{aligned} x(0) &= x_0 \\ x(t_f) &= x_f \end{aligned} \tag{5.10}$$

Condition 2: The initial speed and final speed are 0:

$$\begin{aligned} \dot{x}(0) &= 0 \\ \dot{x}(t_f) &= 0 \end{aligned} \tag{5.11}$$

The polynomial based on equations 2.27 to 2.29 and 2.31 to fulfil the desired trajectory position x, speed $\dot{x}$, and acceleration $\ddot{x}$ are as follows:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{5.12}$$

$$\dot{x}(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{5.13}$$

$$\ddot{x}(t) = 2a_2 + 6a_3 t \tag{5.14}$$

The terms a_1, a_2, and a_3 are presented, below:

$$\begin{aligned} a_0 &= x_0 \\ a_1 &= 0 \\ a_2 &= \frac{3}{t_f^2}(x_f - x_0) \\ a_3 &= -\frac{2}{t_f^3}(x_f - x_0) \end{aligned} \tag{5.15}$$

In addition to the smooth trajectory generation, the sinusoidal trajectory generation was also implemented in the simulation. The position x is generated by sinusoidal function, then speed $\dot{x}$ and acceleration $\ddot{x}$ were obtained from the derivative of x as follows:

$$\begin{aligned} x(t) &= A\sin(P\pi t) \\ \dot{x}(t) &= A\cdot\pi\cdot P\cos(P\pi t) \\ \ddot{x}(t) &= \pi^2\cdot(-A)\cdot P^2\sin(P\pi t) \end{aligned} \tag{5.16}$$

in which A is amplitude and P is period.

5.1.5 Simulation result

The tracking position reference x_d and its derivate are based on the smooth trajectory generation and sinusoidal trajectory generation. Therefore, two trajectory tracking performances were compared between the three controllers.

For the smooth trajectory generation, the final position x_f was selected as 30mm, and travel time t_f as 2s. The travel time was selected as 2s, which, due to this trajectory, the speed limit was below 25mm/s (maximum speed of the motor). The maximum speed this trajectory could reach was 22.5mm/s. The overall reference smooth trajectory plot is as follows:

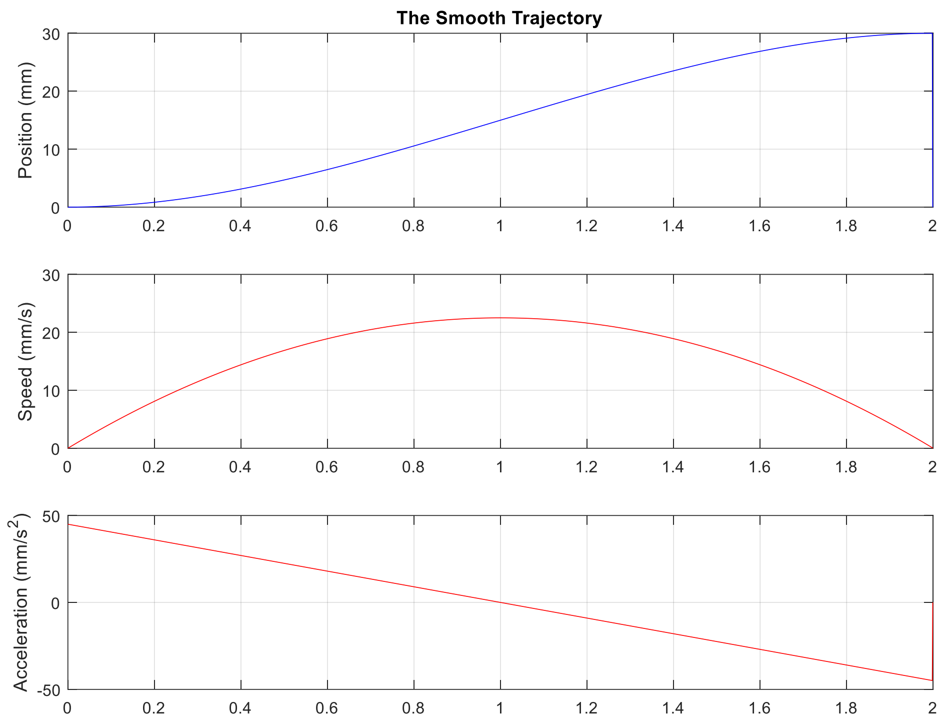


Figure 5.1.6 Smooth trajectory generation

For the sinusoidal trajectory, the amplitude A was set at 10mm and P at 0.5. Based on those parameters, the trajectory had an amplitude of 10mm with a period of a 4s sine wave, as illustrated in Figure 5.1.7. The total running time was 8s, which means two cycles.

Each controller performed two conditions on the two trajectories. The first condition simulated the ideal condition, in which the estimation of the parameters was correct and there was no disturbance added to the system. The encoder had zero errors. The second condition simulated the real condition, in which the parameters were not perfect and there were random disturbances added to the LuGre friction. The encoder also included random noise between ±0.005mm.

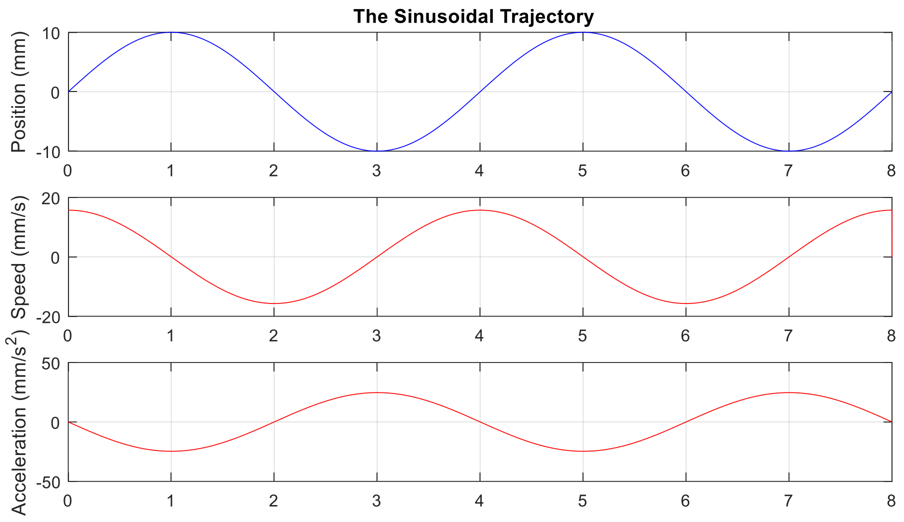


Figure 5.1.7 Sinusoidal trajectory generation

To compare both the ideal condition and the real condition, the noise and disturbance level are summarized in Table 5.1.1.

	JLM estimated	Friction disturbance	Encoder accuracy
Ideal condition	JLMHat = JLM	No disturbance	Prefect accuracy
Real condition	JLMHat = 30% JLM	Band-limited white noise (sampling time 0.001, noise power 0.0001)	Uniform random ± 0.005mm

Table 5.1.1 Ideal condition and real condition setup

The simulation began with the smooth trajectory tracking, then the ideal condition was applied to the PID controller, the PD-F controller, and the PD-DOB controller. The sampling time for all the controllers was 0.0001s. The small sampling time is due to the limitation of the PD-DOB controller. During the test run for the PD-DOB controller, the shorter sampling time caused integration errors. To ensure consistency of simulation, all the controllers ran with a 0.0001s sampling time. The PID controller speed and position tracking simulation were based on the Simulink model in Figure 5.1.1. The PID controller parameters were manually tuned to $K_p=20$, $K_i=2.5$, and $K_d=0$. The results for the PID controller are presented in Figure 5.1.8.

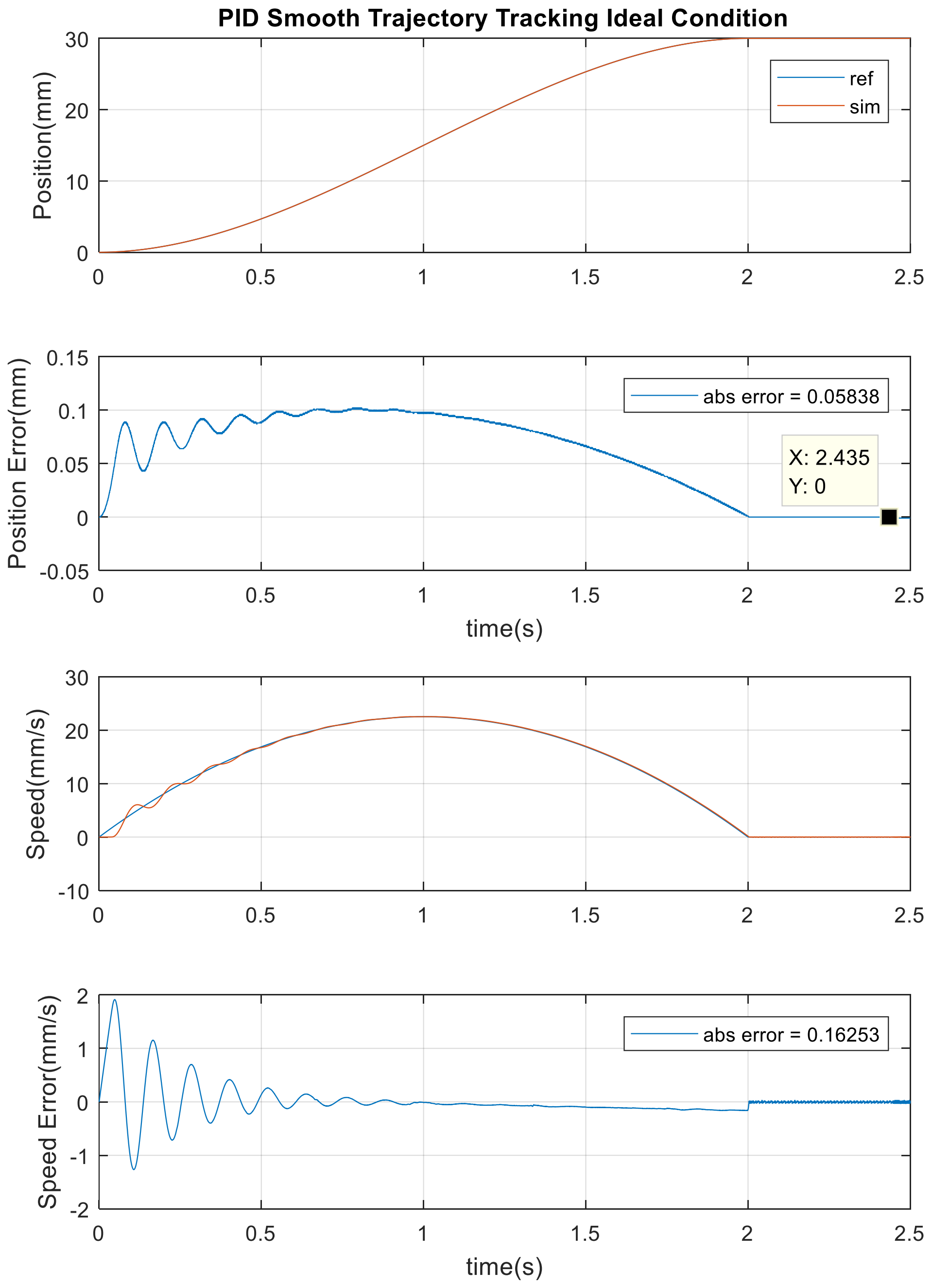


Figure 5.1.8 PID smooth trajectory tracking ideal condition

Based on Figure 5.1.8, the absolute position mean tracking error is 0.05838. From the tracking result, the starting stage has an enormous tracking error. This tracking error is due to the static friction at the beginning of the motion. After one second, the tracking error begins to decrease. Eventually, it reaches a zero steady-state error. From the same figure, the speed tracking absolute mean error is 0.16253. The speed tracking error is similar to the position tracking error, in that they both a very large tracking error at the beginning stage. Then, after 0.5 seconds, the speed tracking begins to converge to the desired speed.

Regarding the PD-F controller results, the speed and position tracking simulation are based on the Simulink model in Figure 5.1.4. The parameters for the PD-F were selected as $K_p=20, Kd=0.1$, and the friction observer correct term $K_z=0.5$. The results for the PD-F controller are displayed in Figure 5.1.9. In this figure, the tracking performance has increased greatly from the tracking error of 0.1 to 10^{-3}, which reached the maximum reading of the linear encoder. The impulse error signal is caused by the quantizer encoder reading of 0. 001mm.The position tracking absolute mean error is reduced to 1.3039×10^{-5} (Figure 5.1.9 displays 1×10^{-5} due to digit limitation). The steady-state error is also zero. The speed tracking performance increased. The speed tracking error decreased compared with the PID controller.

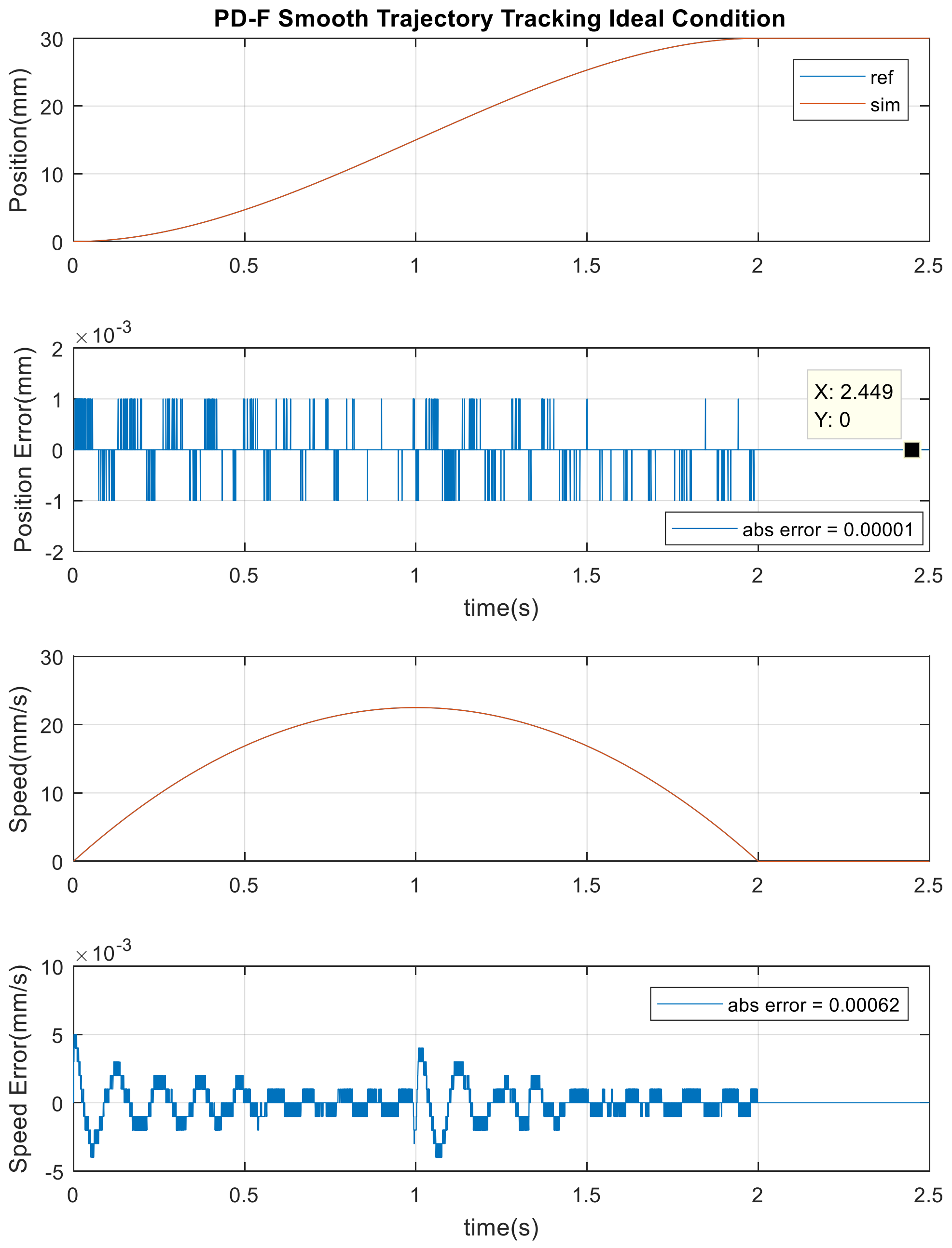


Figure 5.1.9 PD-F smooth trajectory tracking ideal condition

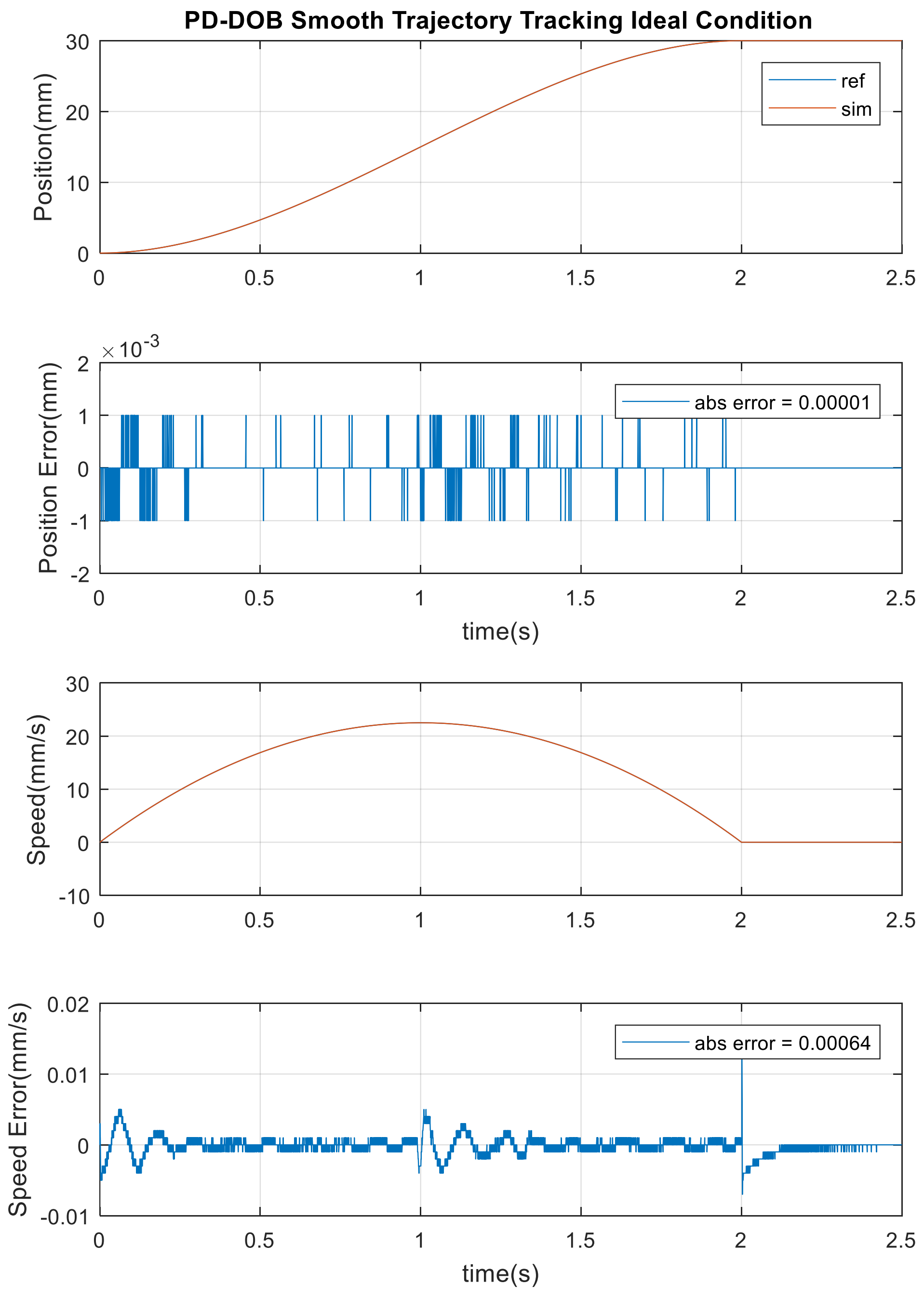


Figure 5.1.10 PD-DOB smooth trajectory tracking ideal condition

The final controller is the PD-DOB controller. The speed and position tracking simulation are based on the Simulink model in Figure 5.1.5. The simulation result is presented in Figure 5.1.10. Based on the simulation result, the tracking error was very similar to the PD-F controller result. The tracking error range also decreased to the limit of the reading digit of the linear encoder. The absolute mean tracking error reduced to 1.0480e-05, which is very similar to the result of the PD-F controller. The steady-state error was also zero. However, the speed tracking error was slightly larger than the PD-F controller tracking error due to a small oscillation at zero speed. The DOB had some oscillation when trying to reach zero speed. This problem is due to the integration error with the fix sampling time.

The next step was testing the real condition of the three controllers by applying the smooth trajectory projection. Figure 5.1.11 presents the PID controller results. Figure 5.1.12 presents the PD-F controller results, and Figure 5.1.13 presents the PD-DOB controller results. Comparing the three controllers, the PID controller had the worst tracking error (0.059). The tracking error of the PD-F controller was 0.00659, which is 10 times less than the PID controller. The PD-DOB controller had the best results in terms of tracking error (0.00259). The steady-state error means it is difficult to compare the three controllers due to the encoder noise. If we focus on the steady-state section of the three controller tracking errors (see Figure 5.1.14), the PID controller had the most stable steady state. However, the PD-F and the PD-DOB controllers displayed some oscillation in the stable steady state. Regarding speed tracking, the PD-DOB controller had the best result in terms of tracking error (0.08919). Next came the PD-F controller (0.20793), and the worst was the PID controller (0.26769).

In this project, the position control is the primary objective. In terms of position control, comparing the results for the real condition and the ideal condition, the PD-DOB controller had the best trajectory tracking and robustness result. The PD-F controller came next, and the PID controller had the worst tracking and robustness results. However, in terms of steady-state holding, the PID controller had the best performance. Both the PD-F controller and the PD-DOB controller had trouble dealing with the constant position when it reaches a steady state.

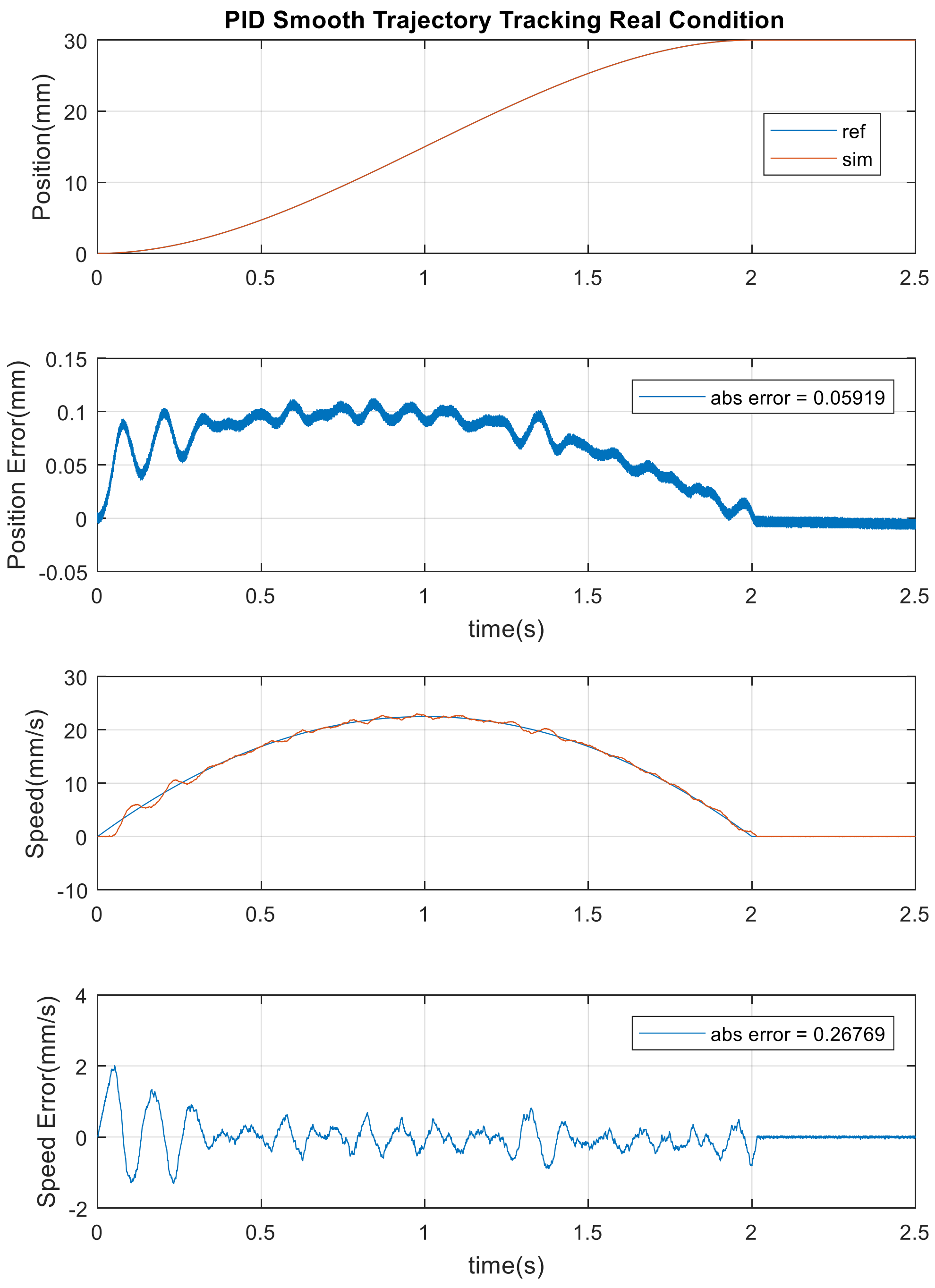


Figure 5.1.11 PID smooth trajectory tracking real condition

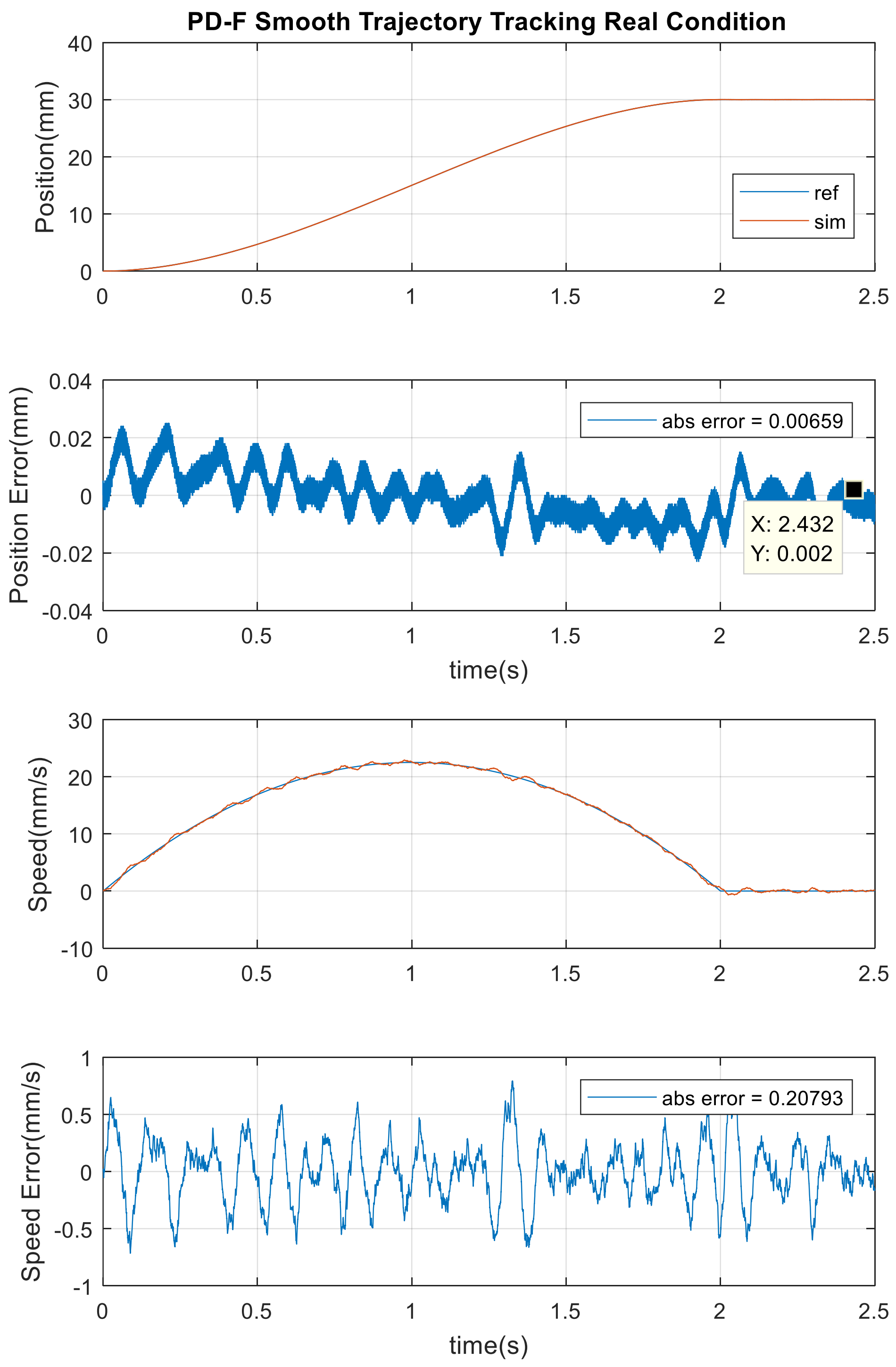


Figure 5.1.12 PD-F smooth trajectory tracking real condition

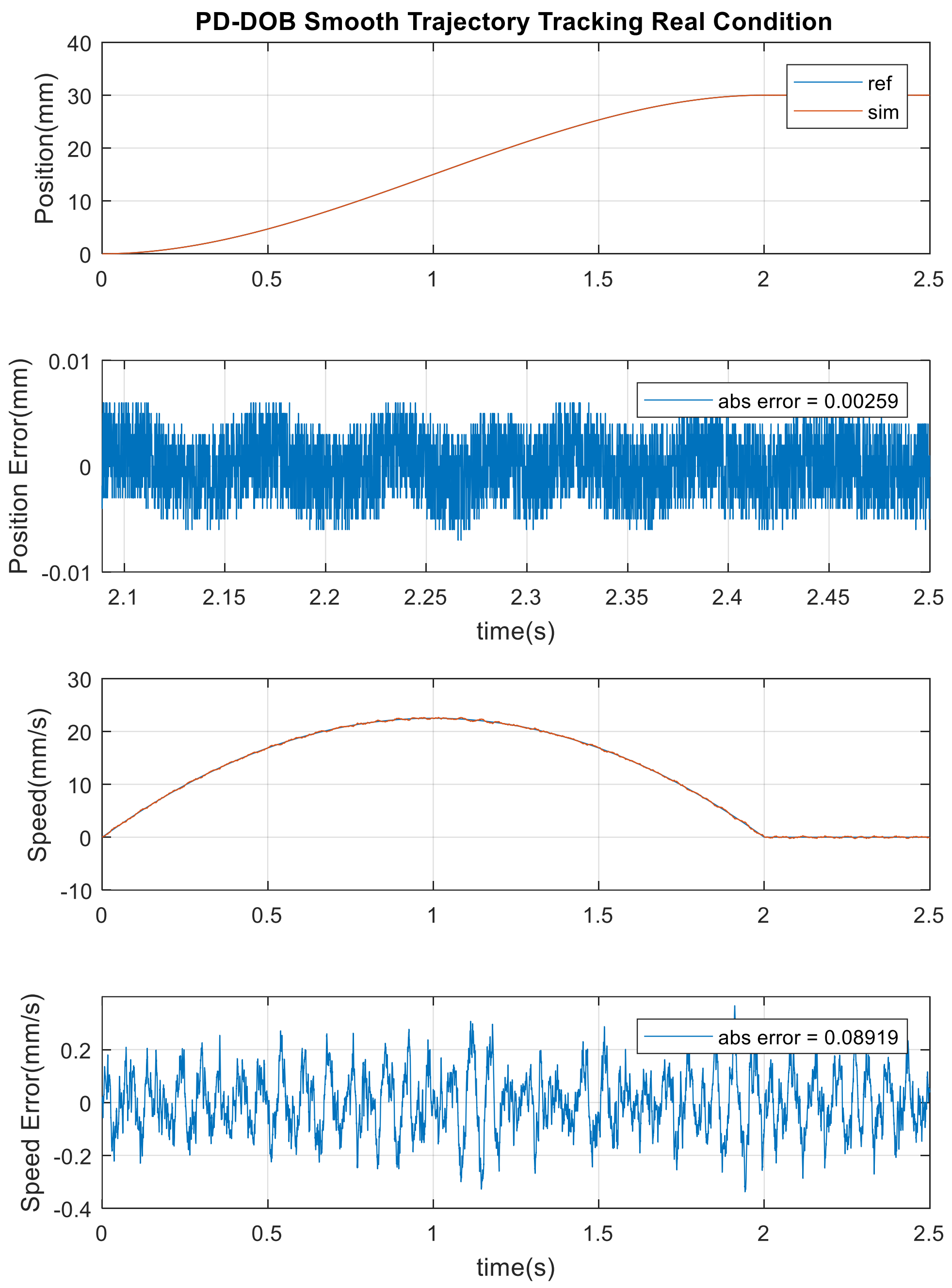


Figure 5.1.13 PD-DOB smooth trajectory tracking real condition

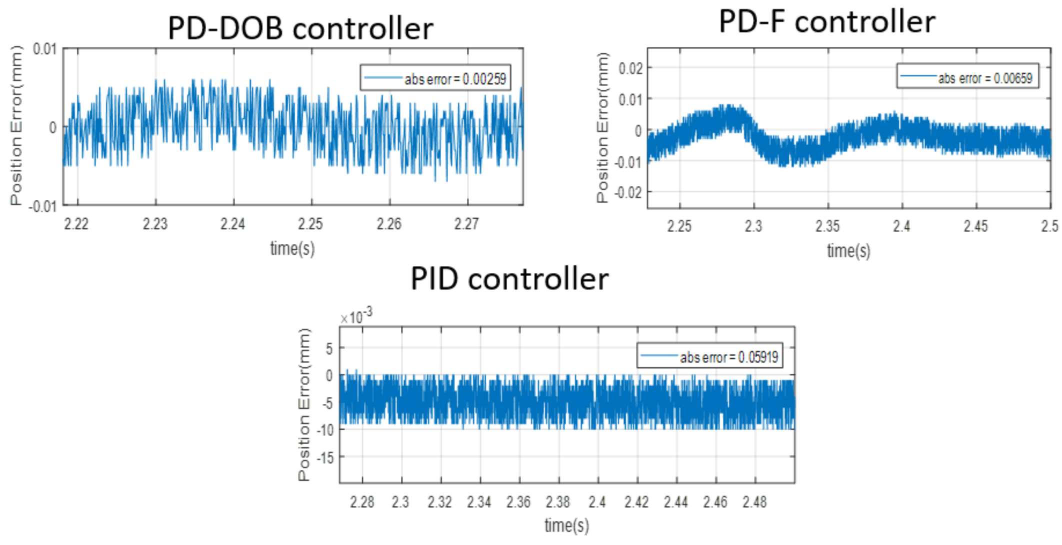


Figure 5.1.14 Steady-state focus on real condition

The next simulation applied the sinusoidal trajectory tracking based on Figure 5.1.7. In the first test, the ideal condition was applied. The parameter settings for all three controllers were the same as for the smooth tracking case. The result of the three controllers are presented below.

In the sinusoidal case, based on the tracking error comparison in Figure 5.1.18, the tracking performance for the PD-F and the PD-DOB controllers improved greatly compared with the PID controller. The tracking performance for the PD-F and the PD-DOB controllers were very similar also. The mean absolute error of the position tracking for the PID controller is 0.06658; for the PD-F controller, 0.00328; and for the PD-DOB controller, 0.00358. In the ideal case, the tracking performance of the PD-F controller is slightly better than the PD-DOB controller.

Based on Figure 5.1.15, the PID controller displayed some oscillation along the path. Both the PD-F and the PD-DOB controllers had some oscillation at the start but quickly converged. The reason for this may be due to the initial desired velocity not being zero; therefore, both controllers tried to reach the desired speed from 0 velocity.

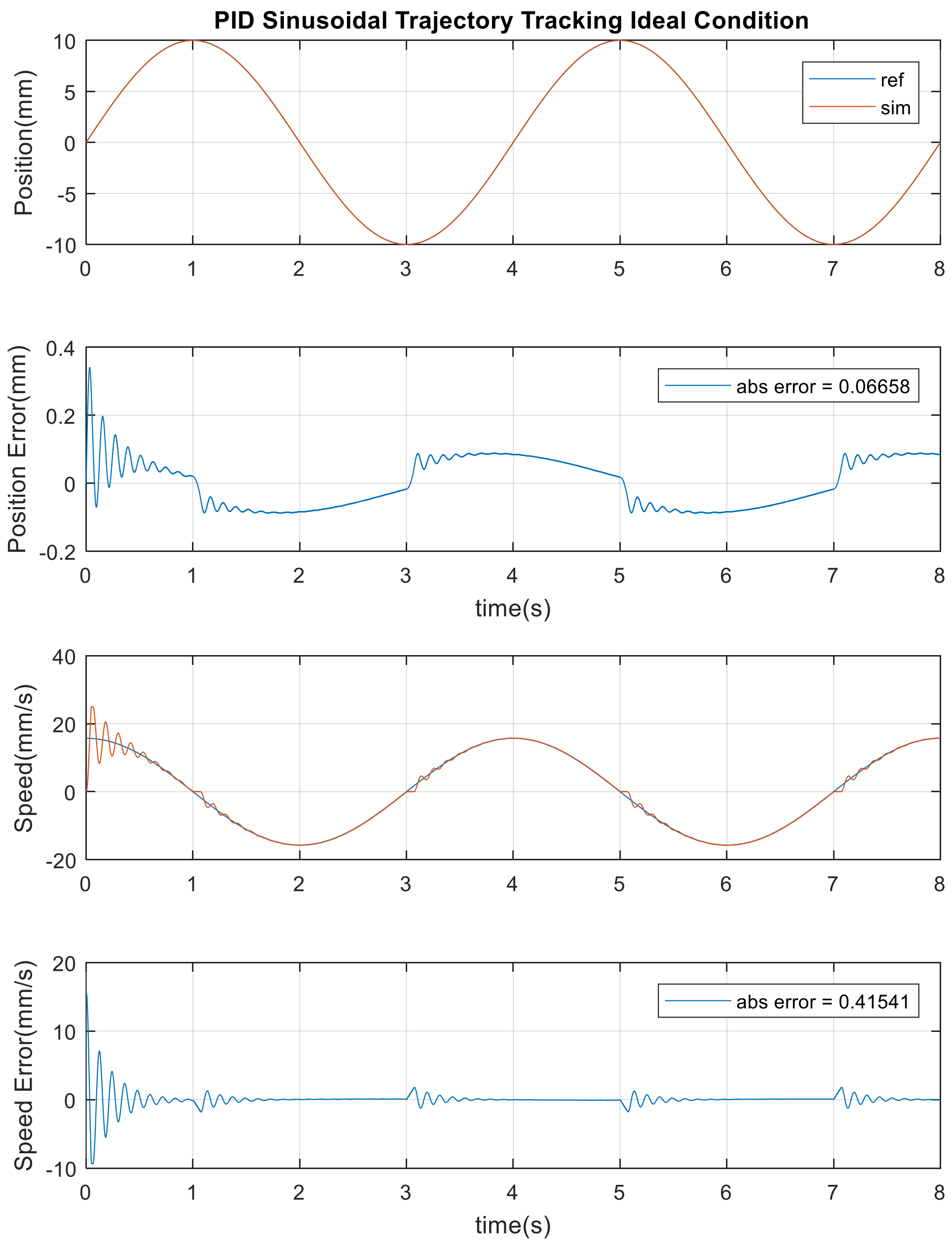


Figure 5.1.15 PID sinusoidal trajectory tracking ideal condition

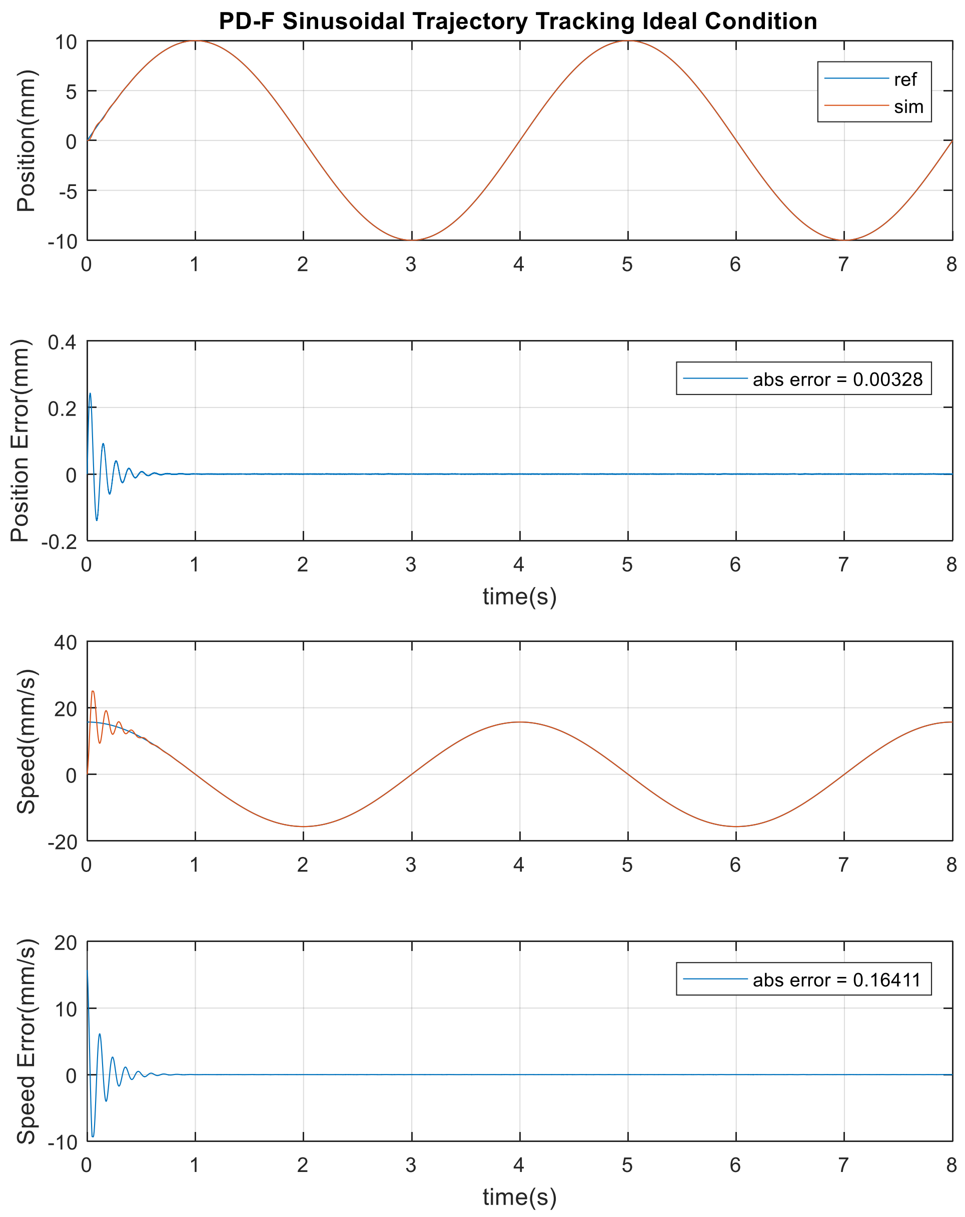


Figure 5.1.16 PD-F sinusoidal trajectory tracking ideal condition

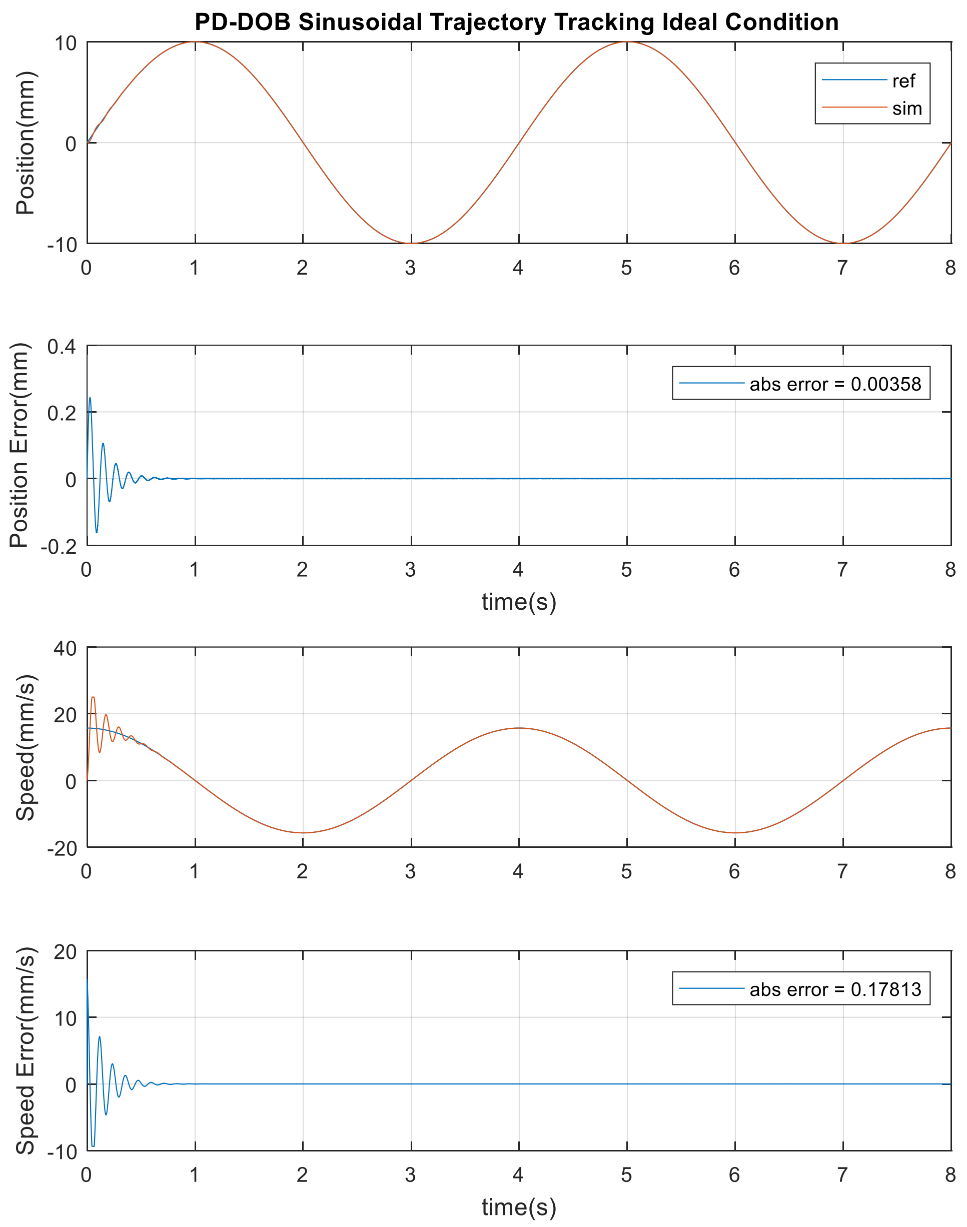


Figure 5.1.17 PD-DOB sinusoidal trajectory tracking ideal condition

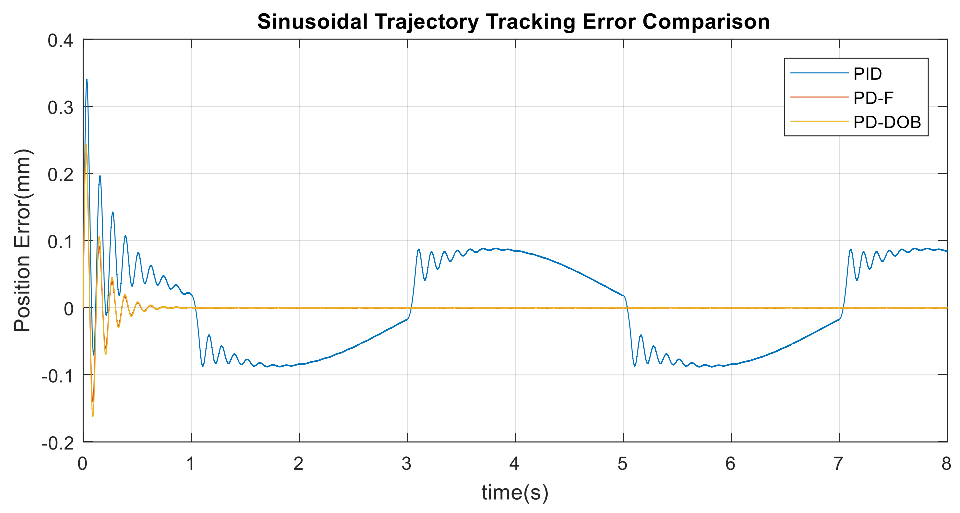


Figure 5.1.18 Sinusoidal trajectory tracking error comparison

The real condition simulation was conducted next. Figure 5.1.19 to 5.1.21 present the results. Again, the PID controller had the worst tracking result. The PD-F and the PD-DOB controllers both displayed an increase in tracking performance. The tracking mean absolute error of the PID controller is 0.06670; the tracking mean absolute error of the PD-F controller is 0.00823; and the tracking mean absolute error for the PD-DOB controller is 0.00371. From the error comparison in Figure 5.1.22, it is apparent that the tracking performance for the PD-DOB controller is better than the PD-F controller when dealing with disturbance and uncertain parameters.

Based on the tracking path, the PID controller performance is similar for the ideal condition; there is much oscillation along the tracking path. Due to the disturbance and uncertain parameters added to the system, the PD-F controller also displayed a small level of oscillation along the path. For the PD-DOB controller, the distance and uncertain parameters are considered in the disturbance observer; there is almost no oscillation along the tracking path. Thus, the PD-DOB controller had the best performance when dealing with disturbance and uncertain parameters.

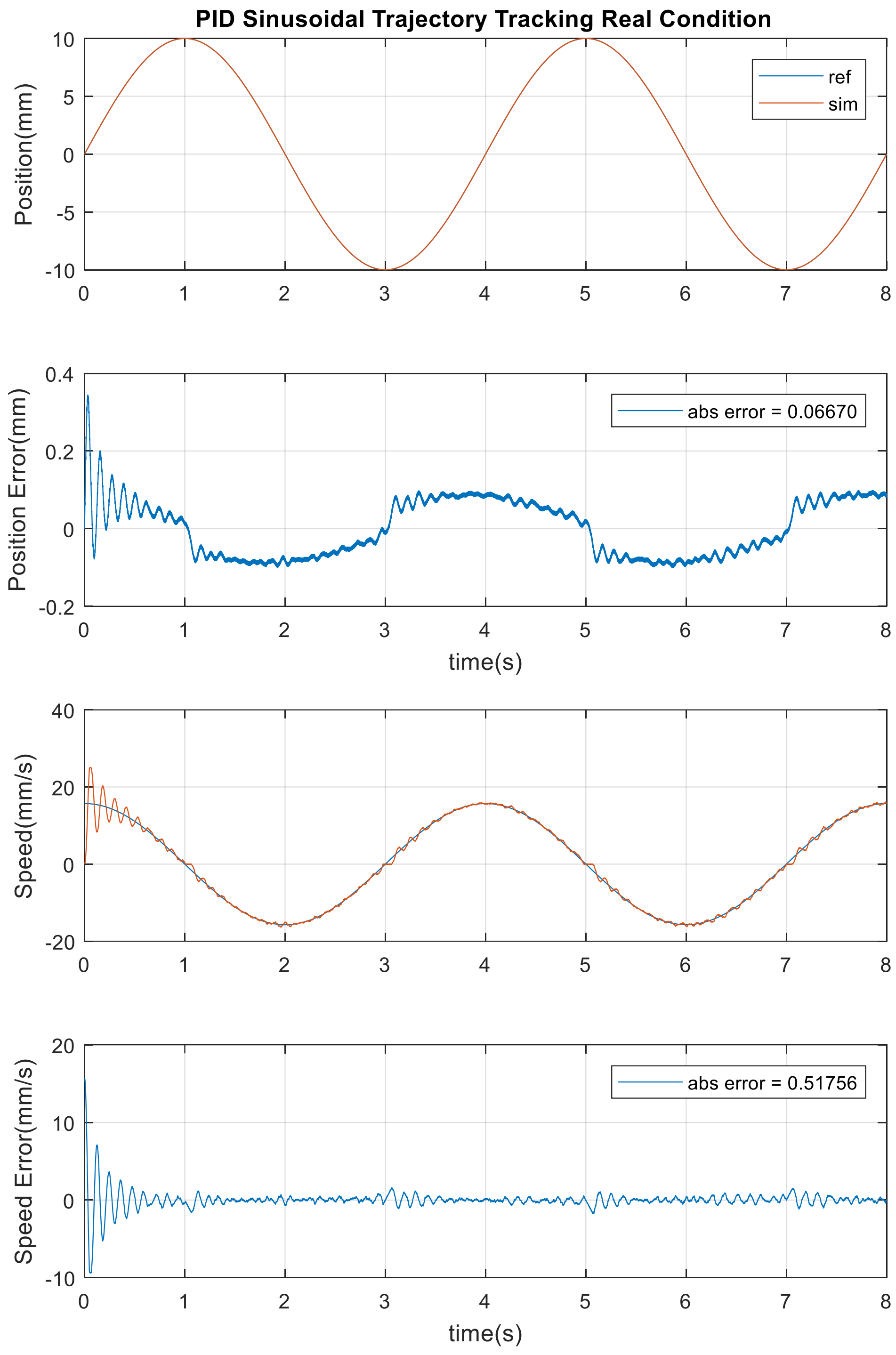


Figure 5.1.19 PID sinusoidal trajectory tracking real condition

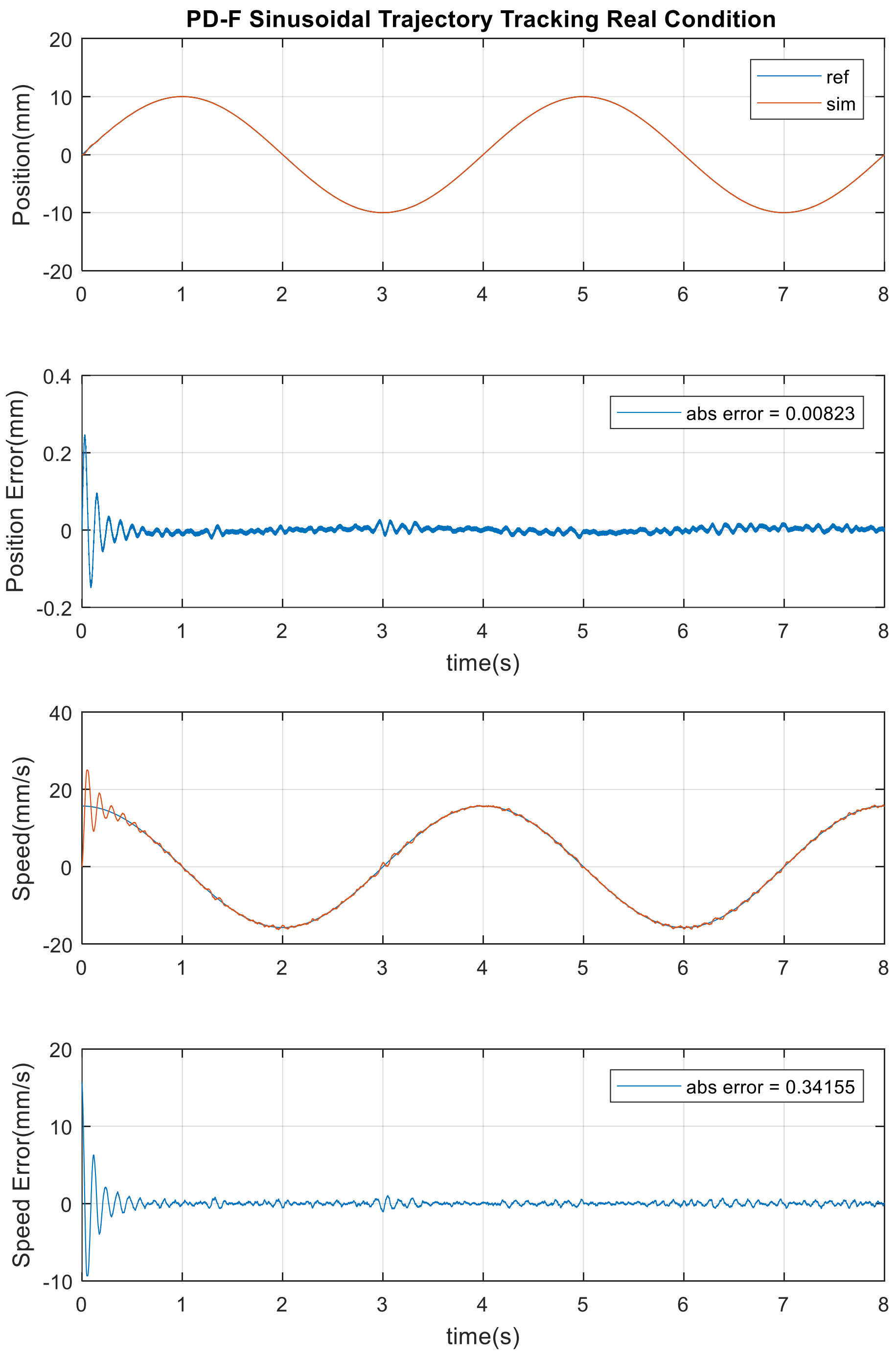


Figure 5.1.20 PD-F sinusoidal trajectory tracking real condition

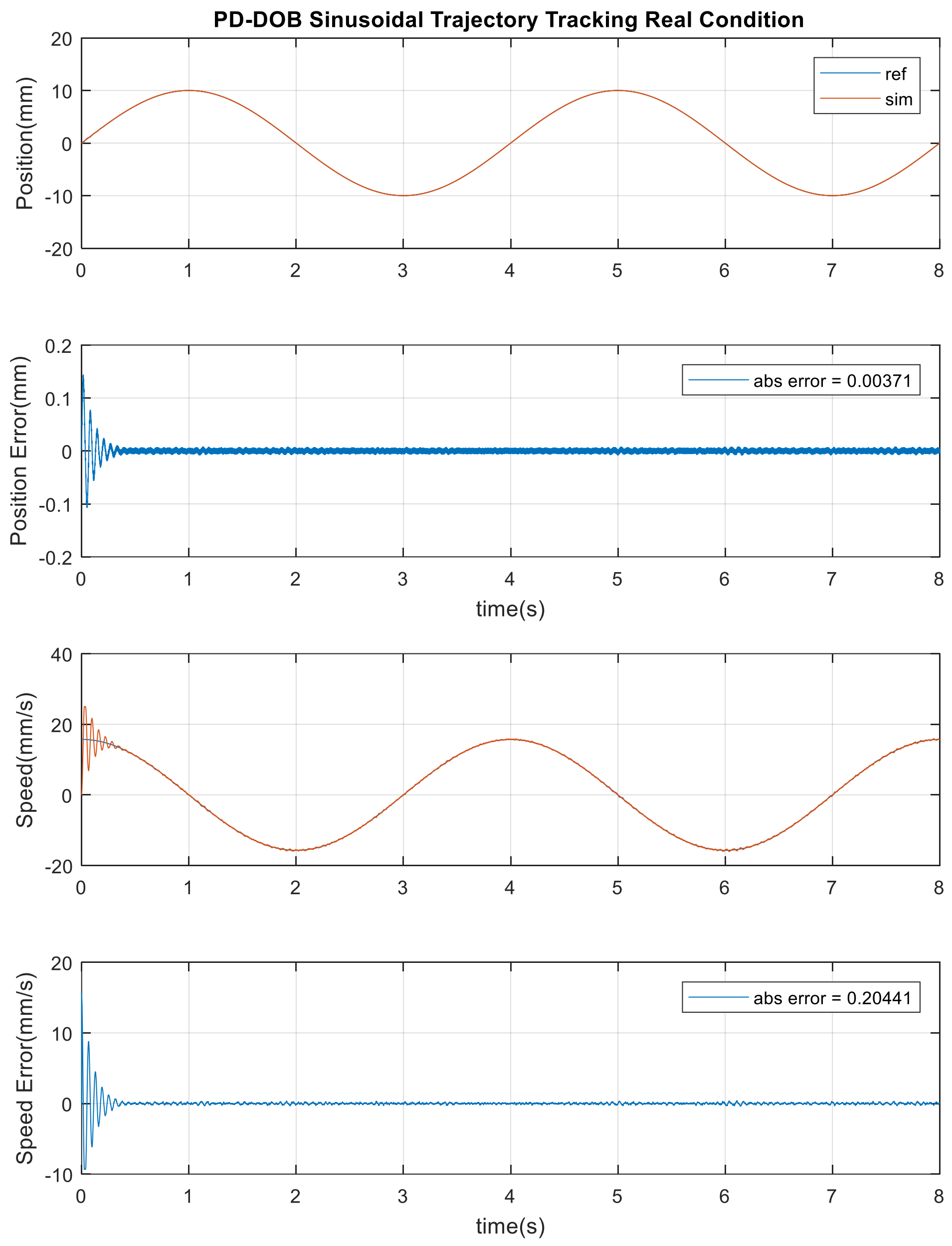


Figure 5.1.21 PD-DOB sinusoidal trajectory tracking real condition

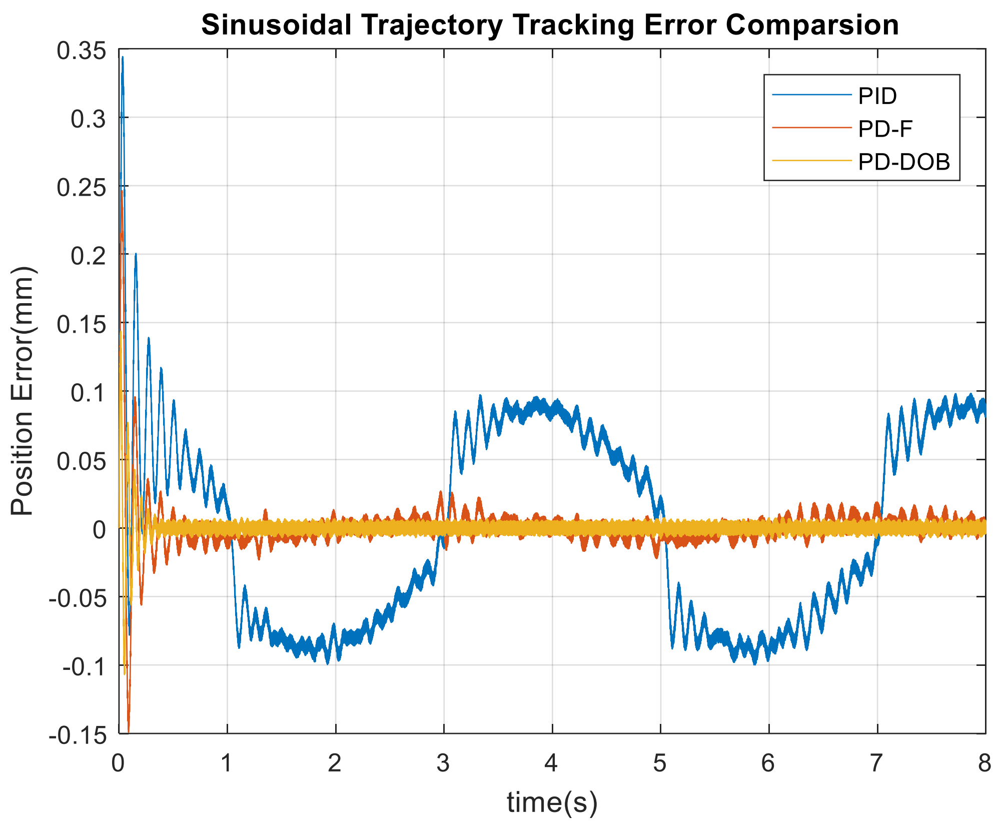


Figure 5.1.22 Sinusoidal trajectory tracking error comparison for real condition

Smooth trajectory tracking		
Controller	Ideal condition MAE[3]	Real condition MAE
PID	0.05838	0.05919
PD-F	1.3039e-5	0.00659
PD-DOB	1.0480e-5	0.00259

Table 5.1.2 Smooth trajectory tracking summary

Sinusoidal trajectory tracking		
Controller	Ideal condition MAE	Real condition MAE
PID	0.06658	0.06670
PD-F	0.00328	0.00823
PD-DOB	0.00358	0.00371

Table 5.1.3 Sinusoidal trajectory tracking summary

The simulation summary begins with the smooth trajectory case in the ideal condition. The PD-DOB controller and the PD-F controller had a similar performance in trajectory tracking. Their

[3] MAE = Mean Absolute Error

tracking performances are better than the traditional PID controller with no steady-state error. In the real condition, the PD-DOB controller displayed the best tracking performance due to its disturbance observer and friction observer. However, the PD-DOB controller and the PD-F controller had some problems dealing with zero velocity. The PD-DOB controller and the PD-F controller displayed some oscillation in the steady-state area. In the sinusoidal trajectory case with the ideal condition, the PD-F and the PD-DOB controllers had very similar performances regarding tracking performance, and their MAE are very similar also (see Table 5.1.3). Again, the PID controller has a large MAE compared with the PD-F and the PD-DOB controllers. Furthermore, the PID controller displayed some oscillation along the sine wave tracking path. At the beginning stage of the sine wave, all the controllers displayed significant oscillation, which was mainly due to the non-zero velocity at the start. Therefore, all the controllers tried to reach this initial speed at the beginning. After that, the PD-F and the PD-DOB controllers began to converge to a small error. In the real condition, the PD-DOB controller dominated tracking performance by having the smallest MAE. This again proved the disturbance rejection ability of the PD-DOB controller.

To conclude, the friction estimation could help to increase tracking performance in terms of the trajectory tracking error. The traditional PD controller combined with friction estimation could decrease the tracking MAE compared with the results of the traditional PID controller (or PI controller in the simulation). If disturbance and uncertain parameters are added to the system, the PD controller with friction estimation could deal with that also. However, with the help of the disturbance observer, the tracking performance is dramatically increased compared with the traditional PID controller. Nevertheless, the PD-F and the PD-DOB controllers may have difficulty dealing with the constant reference position with disturbance added. In the sinusoidal tracking case, in which there is no constant position, the PD-F and the PD-DOB controllers dominated performance in terms of MAE. Therefore, the traditional PD controller with the help of friction estimation and disturbance observer displayed increased tracking performance and robustness.

5.2 EXPERIMENT ON CONTROL SYSTEM

This section discusses the results of when the control algorithm based on the previous section was modified and implemented. In addition, the new improved control algorithms were tested and implemented and are discussed in this section also. The built-in control algorithm in the servo drive was also implemented and tested in the physical ball screw system. Finally, the results for the different controllers and different conditions are compared and discussed

5.2.1 Experiment on PWM control

The first controller implemented was the servo drive with a built-in control algorithm. The built-in control algorithm is based on the position control mode presented in Appendix Figure 7.3.3. Ac Servo System Hardware. Based on Figure 7.3.2, the Teensy 3.5 sends a PWM pulse signal to the servo drive. The PWM pulse number is directly related to the rotation angle of the AC servo motor. Based on [43], each of the PWM pulses is one motor rotational encoder unit. Based on the AC servo drive setting, each revolution can take 2048 pulses.

The PWM signal comprises a group of fixed amplitude voltage pulses; their duty cycle and period of the pulse can vary. Figure 5.2.1 presents the relationship between period, frequency, duty cycle, and amplitude.

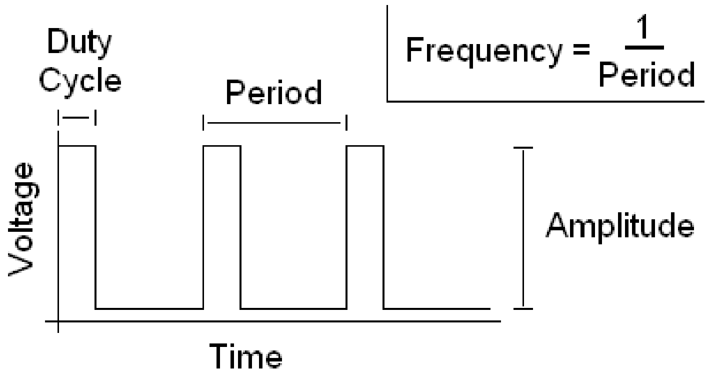


Figure 5.2.1 PWM signals [53]

The Teensy 3.5 can use the PWM module to send fixed amplitude, variable duty cycle, and period PWM signals. The PWM signal output pins connect to an interrupt pin on Teensy 3.5. The PWM module in the microcontroller does not have a counter function; therefore, an interrupt pin is needed to count the number of PWM pulses sent.

To test the accuracy of the onboard control method, two experiments were conducted. In each experiment, the same PWM signal was sent to servo drive. The readings of the linear encoder for each experiment were compared. In this way, the consistency of the onboard control method is presented.

In both experiments, a total of 2×10^5 pulses with fixed 50% duty cycle were sent to the servo motor. The PWM pulse frequency for both experiments were set to 2×10^4 Hz. The data sampling rate was 0.05 seconds. The results of both experiments are illustrated in Figure 5.2.2.

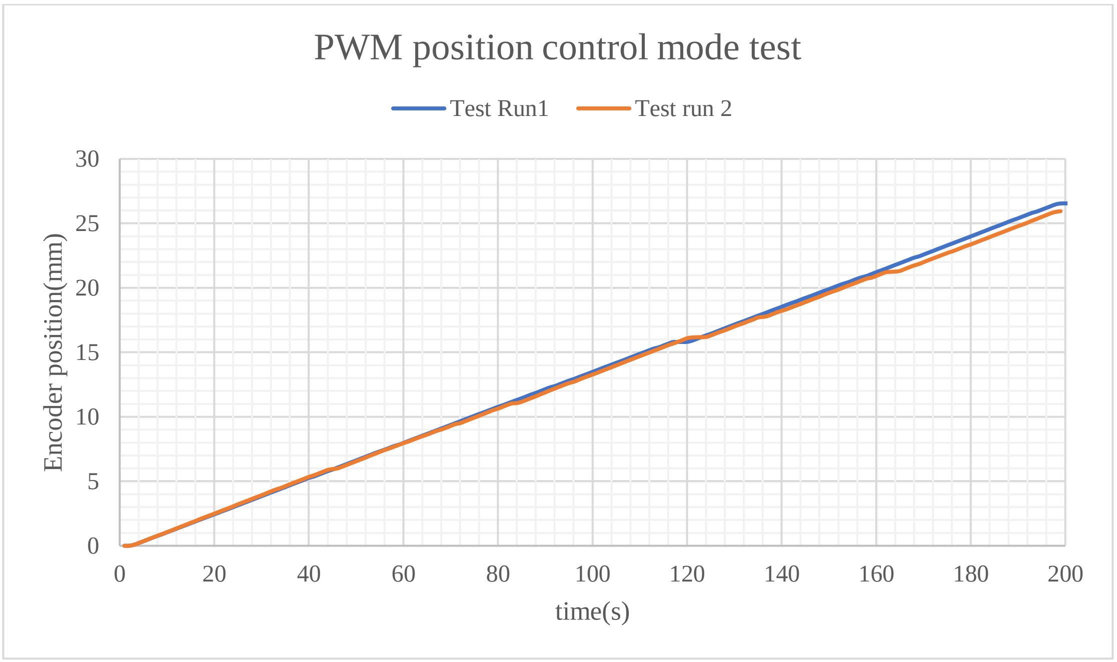


Figure 5.2.2 PWM position control mode test

In Test Run 1, the linear encoder stopped at 25.979mm. In Test Run 2, the linear encoder stopped at 26.113mm. The difference between the two tests is 0.134mm. Therefore, the onboard control algorithm cannot maintain a proper consistency, which affects the accuracy of the system. The rest of this section is based on the torque control mode by applied Teensy 3.5.

5.2.2 Experiment Trajectory generation

The trajectory generation was applied in the physical experiment. The implemented trajectory was based on the smooth trajectory discussed and applied in the simulation in Section 5.1.4. The position, velocity, and acceleration profile are based on equations 5.12 to 5.14. The testing trajectory was chosen to be similar to the simulation trajectory in Figure 5.1.6. Based on Figure 5.1.6, the implement trajectory underwent some modifications. In Figure 5.1.6, the moving direction is single (from left to right only). In the physical implementation, the initial trajectory is the same as Figure 5.1.6. After sliding table reaches the desired final position (30 mm), it then waited for 2 seconds and returned to the initial position with the same path.

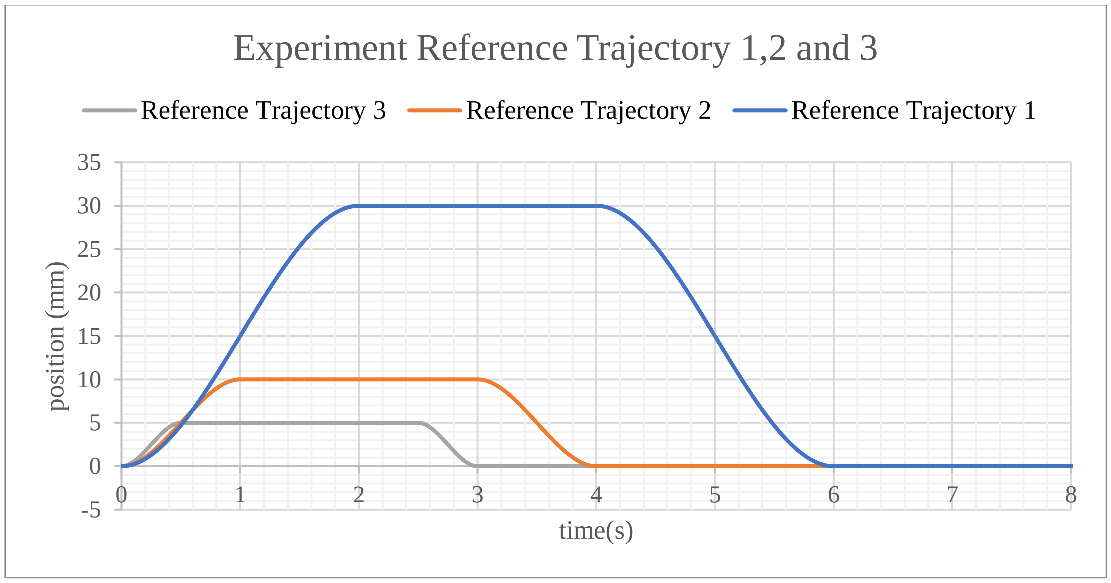


Figure 5.2.3 Experiment: Reference trajectory 1,2 and 3

Two other reference trajectories were included in the experiment. The second trajectory was set to a desired final position at 10mm, taking 1s to reach there and waiting for 2s, as illustrated in Figure 5.2.3 (Reference Trajectory 2). The third trajectory was set to a desired final position at 5mm, taking 1s to reach there and waiting for 2s, as illustrated in Figure 5.2.3 (Reference Trajectory 3).

The reference speed and acceleration profile for the three trajectories are as follows:

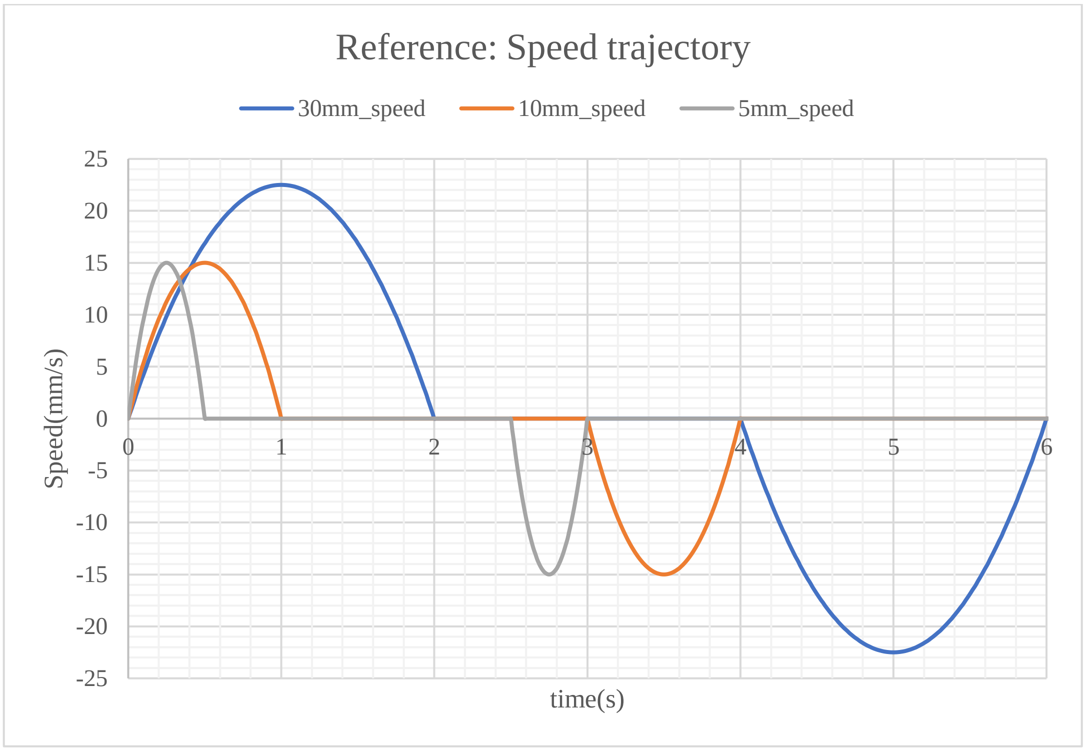


Figure 5.2.4 Reference: Speed trajectory

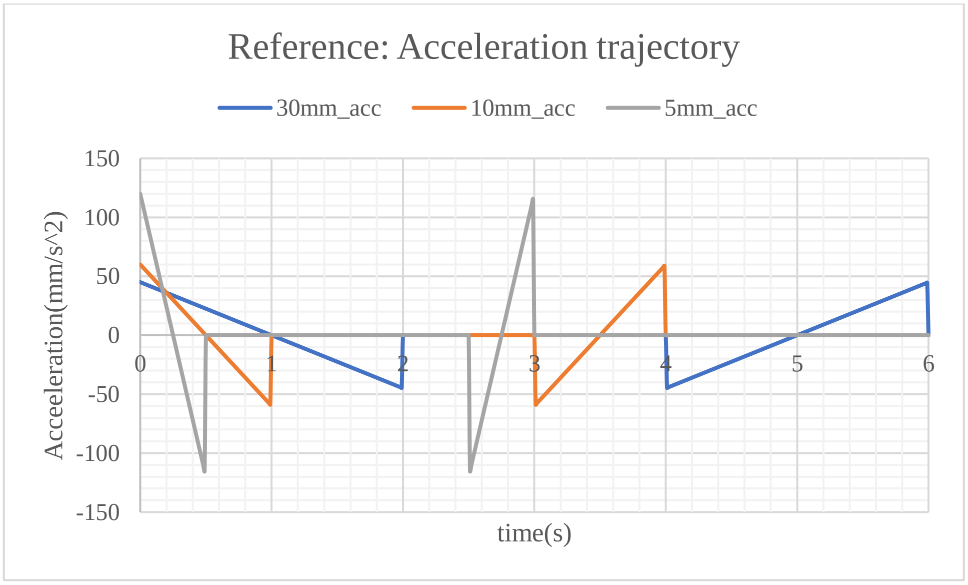


Figure 5.2.5 Reference: Acceleration trajectory

The three different trajectories present three different cases. In general, this project attempts to simulate a scenario that is moving an object to particular position along a smooth path. After the specific process on the object was completed (such as the machining on the object), the object returned to its starting position. Trajectory 1 simulated a long-distance move that required significant speed. Trajectory 2 simulated a middle-distance move in which the speed and acceleration are between those of Trajectory 1 and Trajectory 3. Trajectory 3 simulated a quick move over a short distance but with high acceleration.

5.2.3 Experiment on PID controller

The traditional PID controller was the first controller to be implemented into the physical system. The PID controller control law was referenced using the equations 4.22 and 4.23, and error *e(t)* used the desired position minus the encoder position.

After manually tuning the PID parameters, $K_p=0.6$, the $K_i=0.001$, and $K_d=0$ provided the best results in terms of steady-state error and tracking MAE. The sampling time on Teensy 3.5 was 0.01s. The total running time was 10s. The results for the PID controller with the three trajectories are presented below:

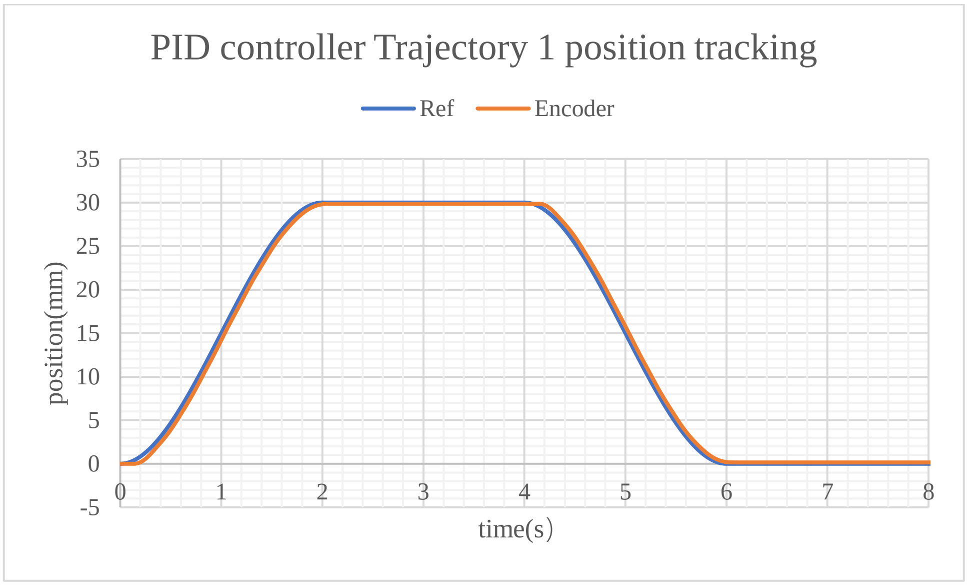


Figure 5.2.6 PID controller Trajectory 1 position tracking

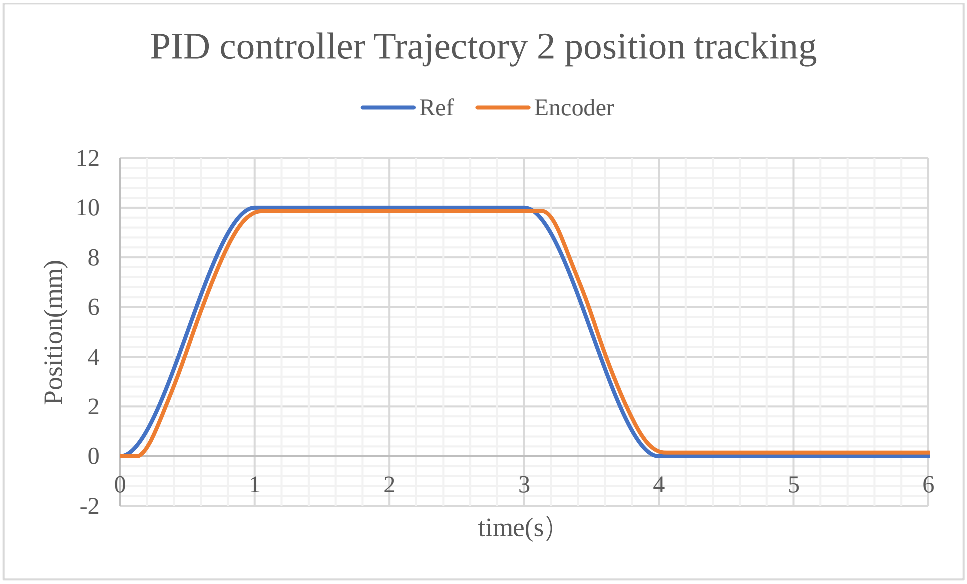


Figure 5.2.7 PID controller Trajectory 2 position tracking

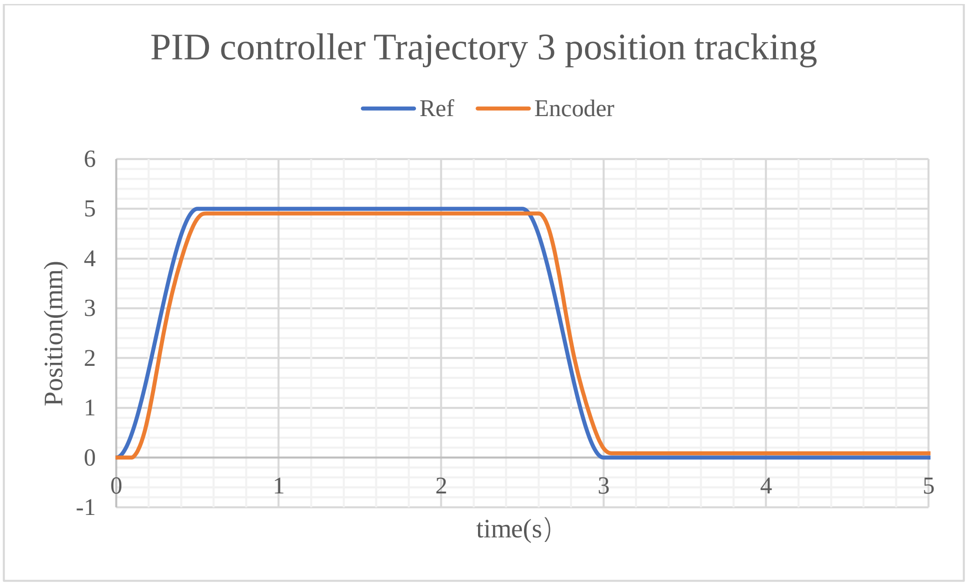


Figure 5.2.8 PID controller Trajectory 3 position tracking

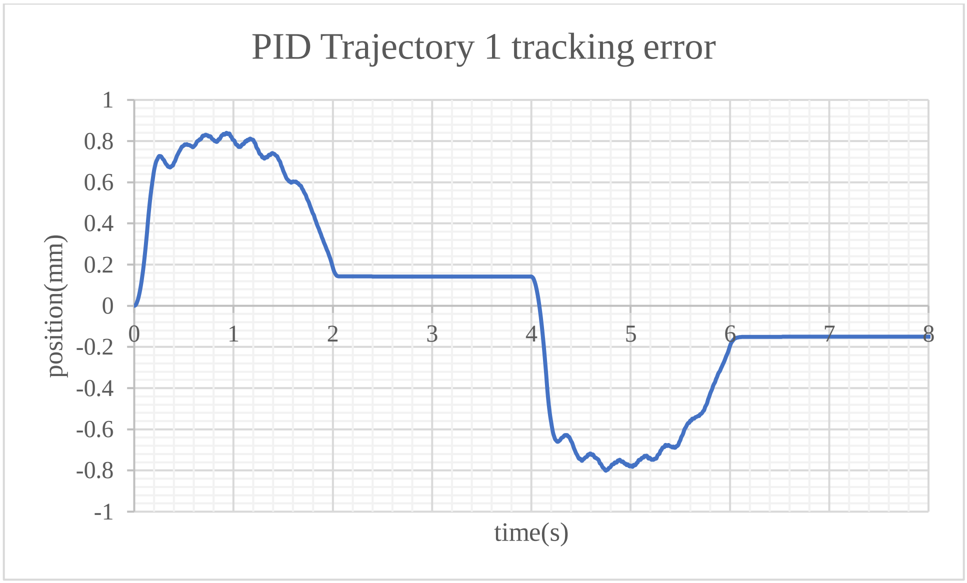


Figure 5.2.9 PID controller Trajectory 1 tracking error

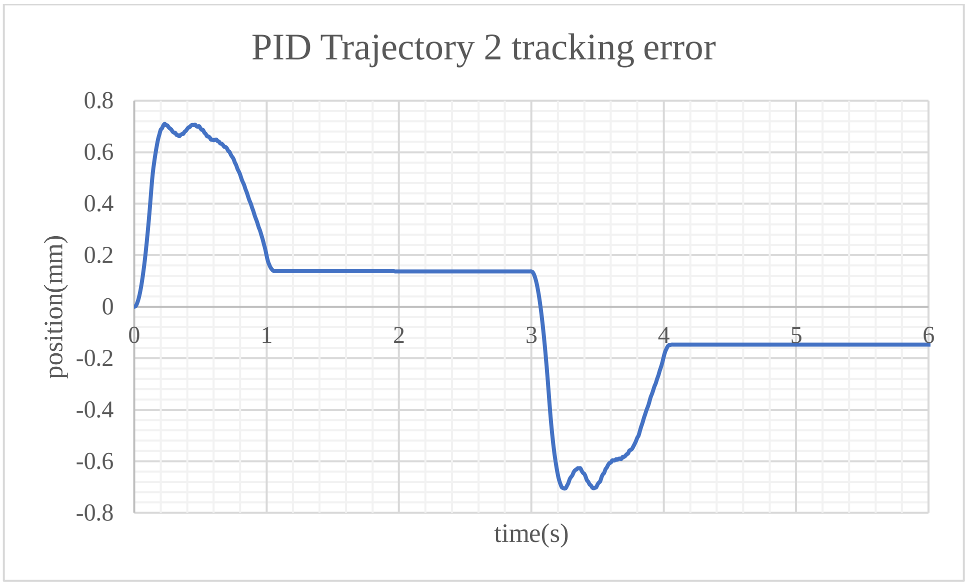


Figure 5.2.10 PID controller Trajectory 2 tracking error

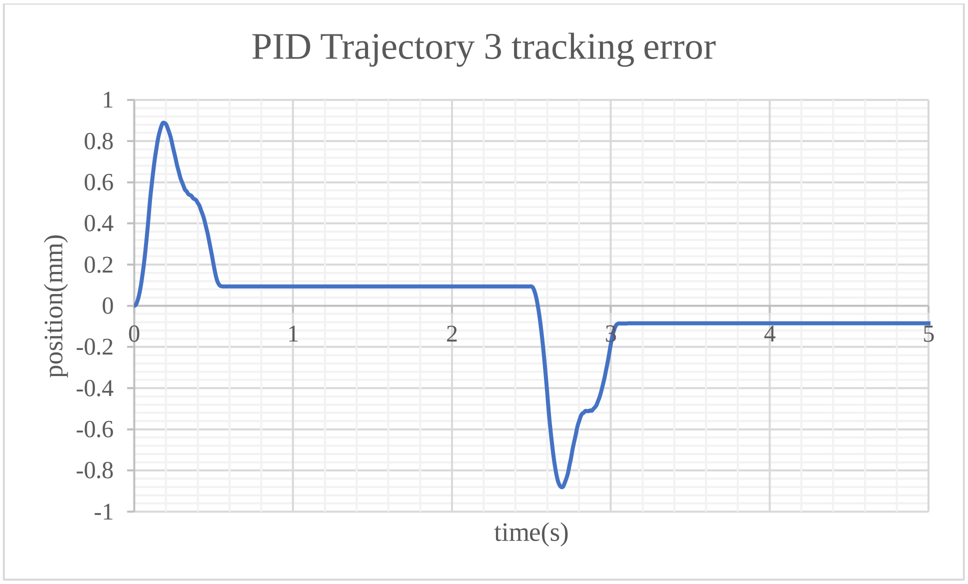


Figure 5.2.11 PID controller Trajectory 3 tracking error

PID controller tracking error table			
	SSE forward (mm)	SSE backward (mm)	MAE (mm)
Trajectory 1 (30mm)	0.142	-0.150	0.384
Trajectory 2 (10mm)	0.137	-0.147	0.267
Trajectory 3 (5mm)	0.094	-0.085	0.174

Table 5.2.1 PID controller tracking error table

Figure 5.2.6 to Figure 5.2.8 present the tracking results compared with the reference position. The tracking errors are presented in Figure 5.2.9 to Figure 5.2.11. The error summary results are presented in Table 5.2.1. To have a fair measure of MAE, the time range selected for the trajectory MAE measurement is different. To have a fair amount of waiting time for both the forward and backward steady state, the 2s waiting time for the desired position was also added to the calculation when the system returned to its initial position. Therefore, the range of MAE for Trajectory 1 was 8s; for Trajectory 2 it was 6s; and for Trajectory 3 it was 5s. The MAE calculation range was applied to the rest of the experiment also. Based on the PID tracking results above, the steady-state error (SSE) remains very large in all three trajectories. From Trajectory 1 to Trajectory 3, the SSE increased with total length of travel. This is due to the "I" term is too small, the time that require to eliminated to SSE is long. In current trajectory, the waiting time that related to the time required is small. In other word, the trajectory didn't give enough time to let PID controller eliminated the SSE. The MAE illustrates that the travel distance increased also, as did the MAE.

5.2.4 Experiment on positioning the PD controller with friction estimation observer

This project next applied the PD controller with friction estimation (PD-F). The PD-F controller was already tested in the simulation in Section 5.1.2. The control law is based on equations 5.1 and 5.2. However, to apply Equation 5.1 to the microcontroller, discretization is needed. The discretization process is presented below, assuming the zero-order holds:

$$\begin{aligned}
&\hat{z}(t+\Delta t) = A_d \hat{z}(t) + B_d(\dot{x} + K_z e(k)) \\
&A_d = e^{A\cdot\Delta t} \\
&A = -\frac{|\dot{x}|}{g(\dot{x})}\sigma_0 \\
&B_d = A^{-1}(A_d - 1)
\end{aligned} \tag{5.17}$$

In Equation 5.17, A^{-1} cannot have a zero denominator; therefore, when $\dot{x}$ =0, A^{-1}=0.

$$\hat{T}_f(\dot{x}) = \sigma_0 \hat{z}(t) + \sigma_1 \frac{z(t+\Delta t) - z(t)}{\Delta t} + \sigma_2 \dot{x} \tag{5.18}$$

Thus, Equation 5.1 becomes Equation 5.17 and Equation 5.18. The term *Δt* is the sampling time. The term *t* is the current sampling time. The control law in Equation 5.2 replaced Equation 5.1 with equations 5.17 and 5.18. The friction, mass, and initial parameters are based on Table 4.2.5. After applying the control law Equation 5.2 to the Arduino code and commencing the test run with a sampling time of 0.01s, the system began to oscillate and go out of control. The reason for this is that the friction estimation became very noisy. The friction estimation Equation 5.18 depends on the speed measurement; however, the speed is obtained from the derivative of the position. The noisy speed leads to a noisy friction estimation. A filter is needed to filter out the noisy speed data.

The speed filter chosen for this project is the linear recursive exponential (LRE) filter because the memory space requirement is less than other filters [54]. The [54] provides an LRE library that can guide implementation into the Arduino code. The LRE equation based on [54] is presented below:

$$y(t) = w \cdot x(t) + (1 - w)\, y(t - \Delta t) \tag{5.19}$$

The *x* is the input valve, and *y* is the output valve. The *w* is the weighting faction from 0 to 100 range. The higher *w* valve will trend filtered output more to input valve and responds quicker but is not as smooth. The lower *w* valve will trend filtered output more to output valve and responds slower, but the output is much smoother. Applying the LRE filter to the system made it much more stable. The raw speed data and filtered speed data are compared in Figure 5.2.12.

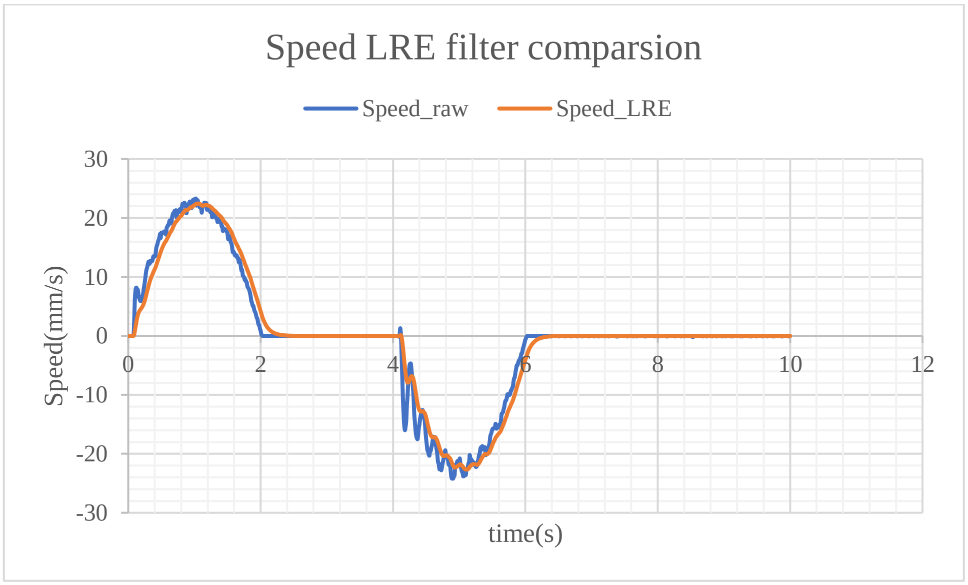


Figure 5.2.12 Speed LRE filter results

After applying the LRE filter, then manually selecting *Kp=5.5*, *Kd=0.1*, and *Kz=0.025*, the results for trajectories 1, 2, and 3 are presented below:

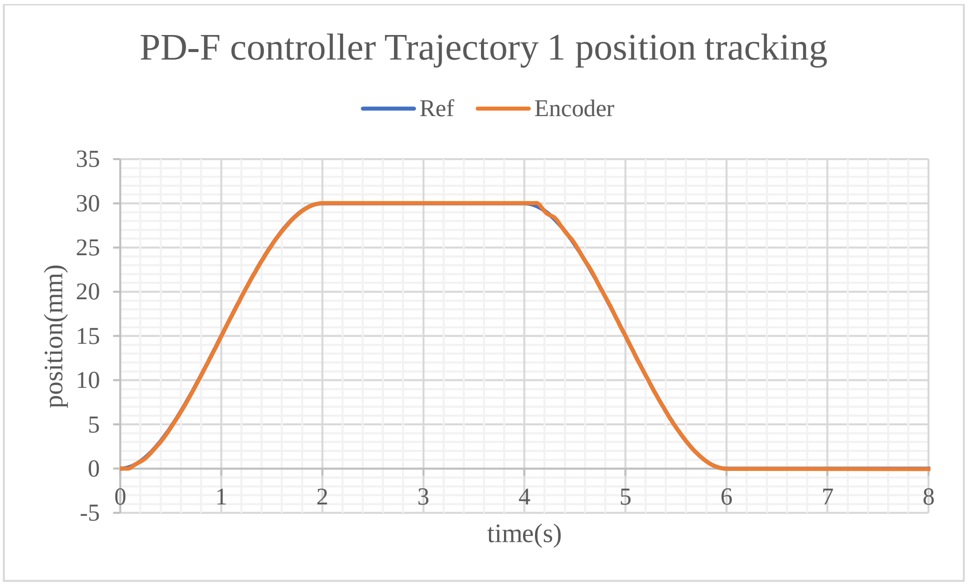


Figure 5.2.13 PD-F controller Trajectory 1 position tracking

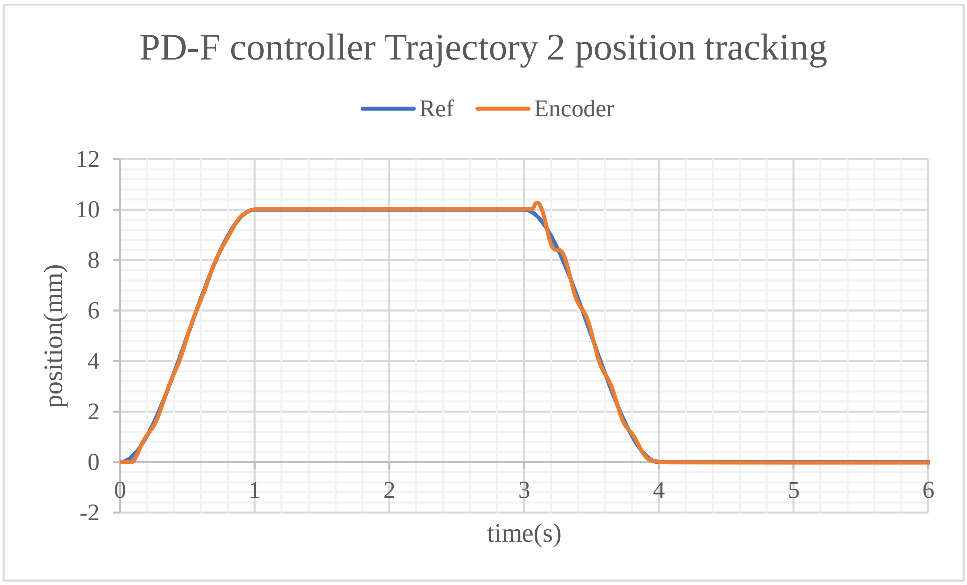


Figure 5.2.14 PD-F controller Trajectory 2 position tracking

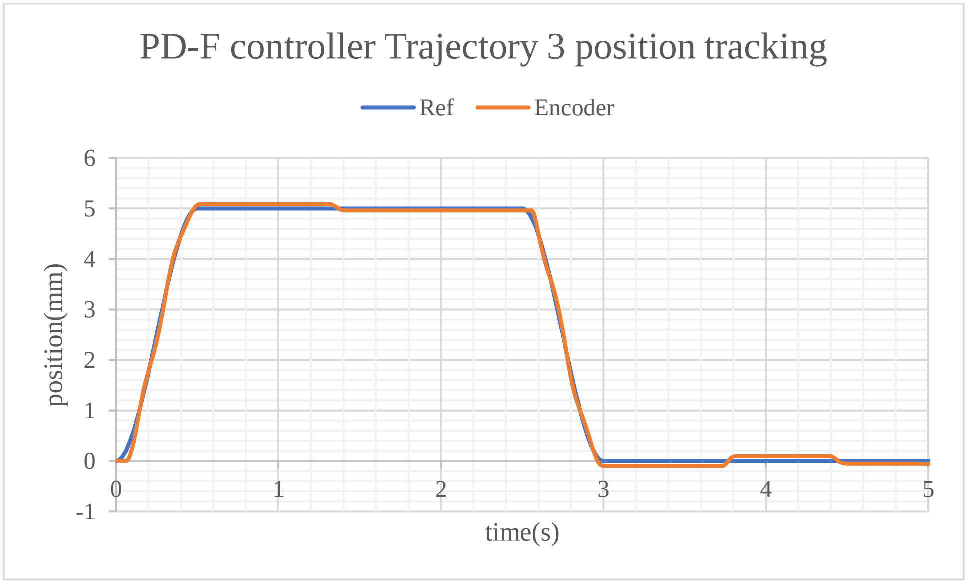


Figure 5.2.15 PD-F controller Trajectory 3 position tracking

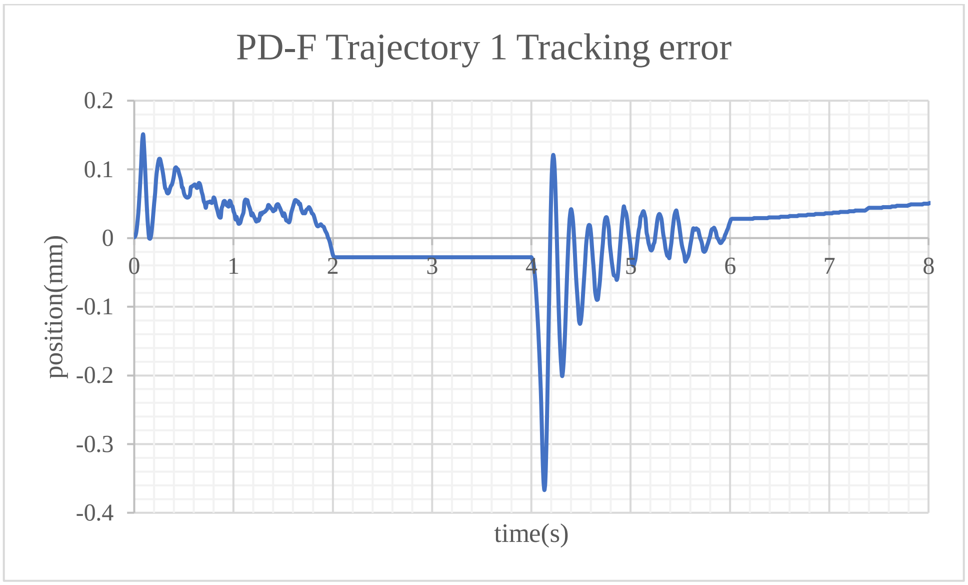


Figure 5.2.16 PD-F controller Trajectory 1 tracking error

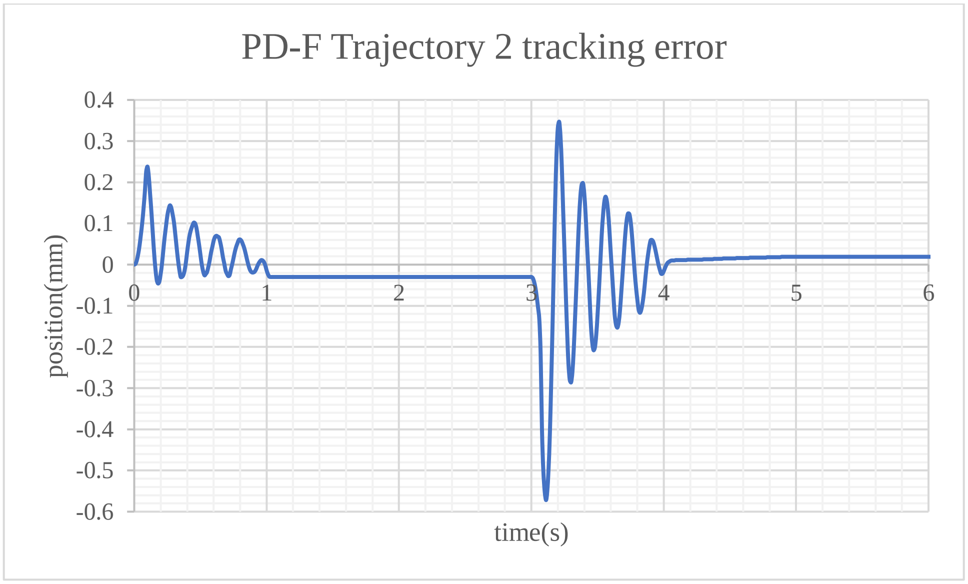


Figure 5.2.17 PD-F controller Trajectory 2 tracking error

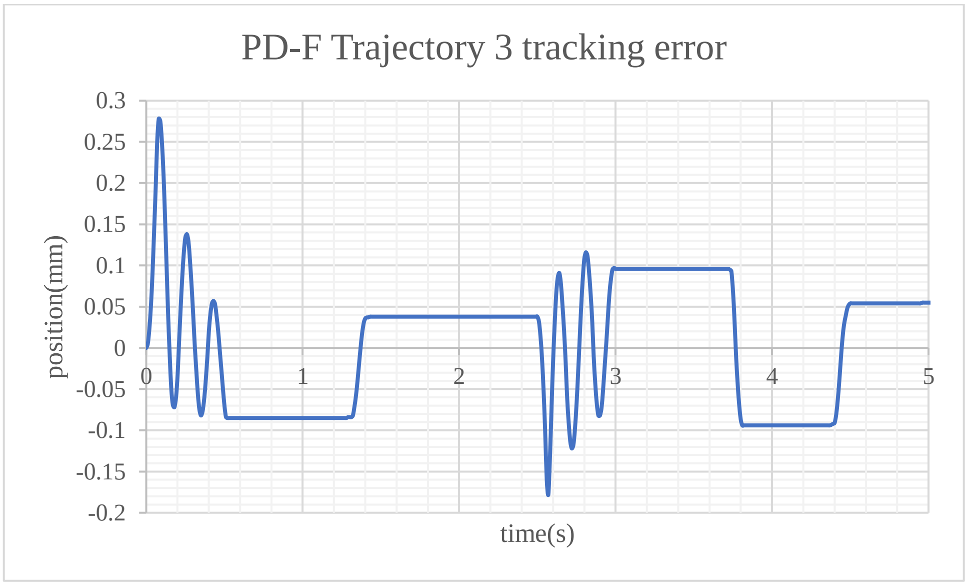


Figure 5.2.18 PD-F controller Trajectory 3 tracking error

PD-F controller tracking error table			
	SSE forward (mm)	SSE backward (mm)	MAE (mm)
Trajectory 1 (30mm)	-0.028	Unstable	0.041
Trajectory 2 (10mm)	-0.03	-0.019	0.047
Trajectory 3 (5mm)	-0.058, 0.038	0.096, -0.094, 0.054	0.070

Table 5.2.2 PD-F controller tracking error table

Figure 5.2.13 to Figure 5.2.18 and Table 5.2.2 summarize the results of the PD-F controller response to three different trajectories. Compared with the results of the PID controller in Table 5.2.1, the SSE for both forward and backward decreased in all three cases. The MAE is ten times smaller than the PID controller result. However, the PD-F controller did not provide a stable control when the reference position was constant. Figure 5.2.13 displays an unstable steady-state tracking for Trajectory 1; the error keeps increasing. Also, for trajectories 2 and 3, the system tended to oscillate around the fixed reference position. In Table 5.2.2, the SSE for the forward and backward of trajectories 2 and 3 displays a jumping steady-state error. The simulation for the PD-F controller for the smooth trajectory case in Figure 5.1.13 also proved the PD-F controller had trouble when dealing with the constant position situation. In Figure 5.2.14, when the system begins moving back to its initial position, some oscillation occurred along the travel path.

5.2.5 Experiment on the Positioning PD controller with friction estimation and a disturbance observer

The next implemented controller was the Positioning PD controller with friction estimation and a disturbance observer (PD-DOB). This control system was simulated and discussed in Section 5.1.3 The applied control law was presented in equations 5.6 to 5.8. The friction estimation observer is same as that in Section 5.2.4. The PD-DOB controller required not only the position feedback, but speed and acceleration feedback as well. Therefore, the speed and acceleration were obtained from the derivative position and then filtered using the LRE filter to ensure smoothness. Also, a lowpass filter was added to the u_d output to filter out the noisy estimation. The lowpass filter is one pole RC-type filter based on [55].

Applying the PD-DOB controller to the Arduino code, the Trajectory 1 test was run with the same K_p, $K_{d,}$ and Kz parameters used in the PD-F controller. The results were not good. Figure 5.2.19 presents the tracking results. The system became very oscillated and unstable. One reason for this is the sampling time. In the simulation, to have the PD-DOB controller running required a minimal sampling time of 0.0001s. The DOB required a smaller sampling time to measure the disturbance. However, in Teensy 3.5, the smallest sampling time that can be applied is 0.01s, any sampling time faster than this causes bugs.

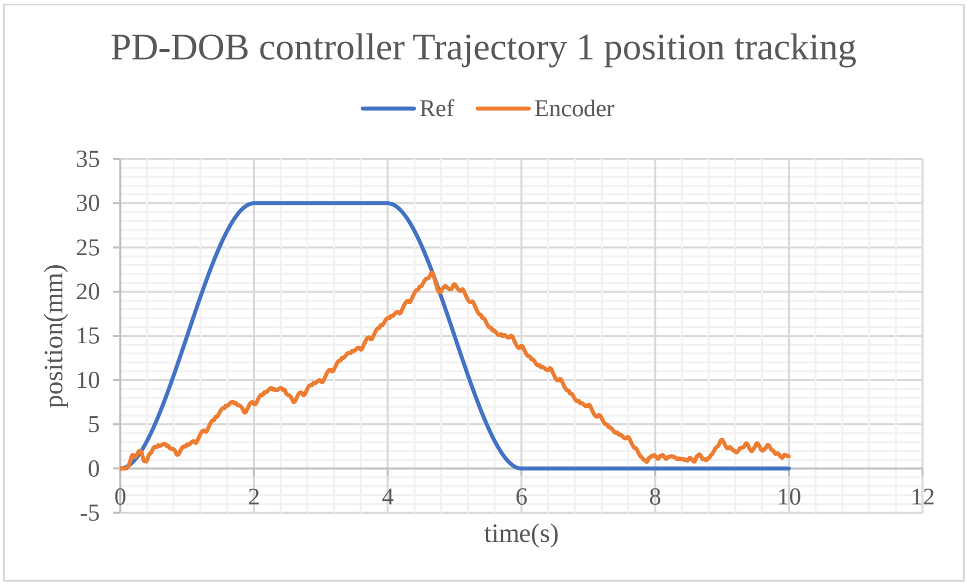


Figure 5.2.19 PD-DOB controller Trajectory 1 position tracking

5.2.6 Experiment on the Positioning PD controller with friction feedforward estimation

Based on the PD-F controller, this thesis proposes using a different position PD controller with friction estimation. In the PD-F controller, the friction is estimated by the feedback speed $\dot{x}$, with the observer correction term $K_z e$. In this controller, instead of using the feedback speed from the encoder, an estimated feedforward speed from the desired trajectory is applied. The feedforward speed is the reference trajectory speed $\dot{x}_d$ generated from the trajectory estimation in Figure 5.2.4. The trajectory generation equations are still based on equations 5.12 to 5.14, meaning the controller uses the estimated feedforward speed from trajectory generation, and therefore there is no need to apply the LRE speed filter. This controller has been called the Positioning PD controller with friction feedforward estimation (PD-FF).

The friction estimations (5.17) and (5.18) are replaced with $\dot{x}$ to $\dot{x}_d$ and the observer term $K_z e$ is removed as follows:

$$\begin{aligned}
&\hat{z}(t+\Delta t) = A_d \hat{z}(t) + B_d(\dot{x}_d) \\
&A_d = e^{A\cdot\Delta t} \\
&A = -\frac{|\dot{x}_d|}{g(\dot{x}_d)}\sigma_0 \\
&B_d = A^{-1}(A_d - 1)
\end{aligned} \tag{5.20}$$

$$\hat{T}_f(\dot{x}_d) = \sigma_0 \hat{z}(t) + \sigma_1 \frac{z(t+\Delta t) - z(t)}{\Delta t} + \sigma_2 \dot{x}_d \tag{5.21}$$

The controller law is same as Equation 5.2. The *Kp* and *Kd* parameters are the same as used in the PD-F controller. The sampling time is again 0.01s, and the results for trajectories 1,2, and 3 are presented below:

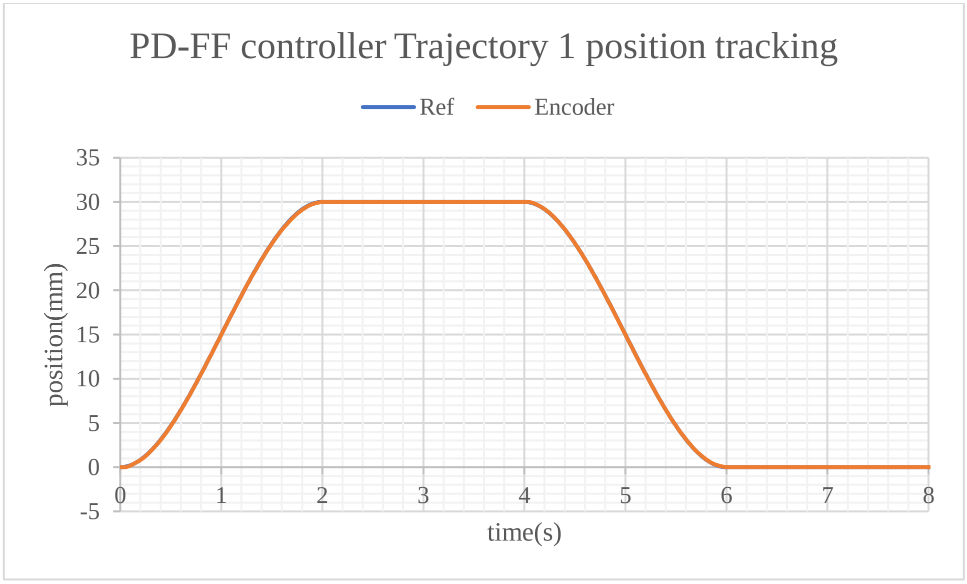


Figure 5.2.20 PD-FF controller Trajectory 1 position tracking

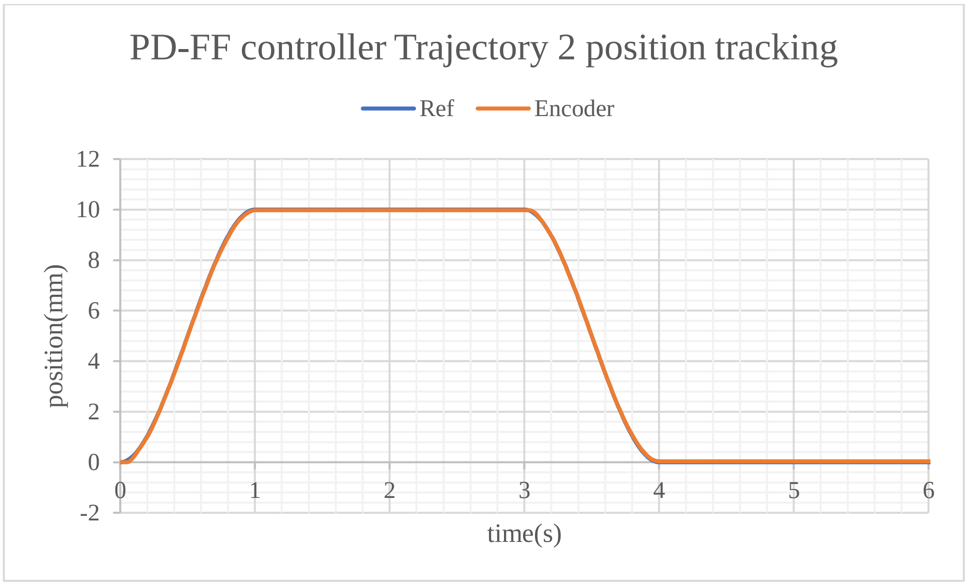


Figure 5.2.21 PD-FF controller Trajectory 2 position tracking

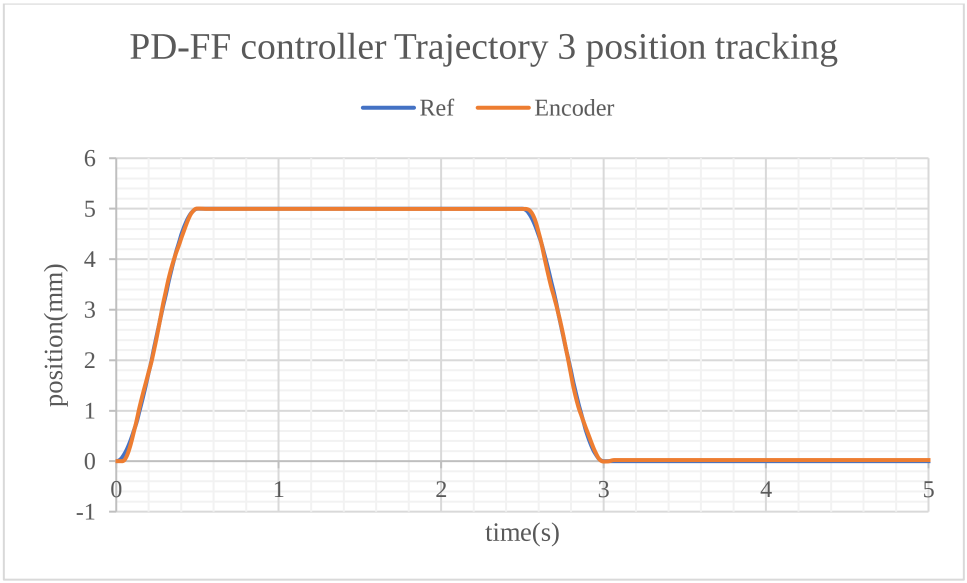


Figure 5.2.22 PD-FF controller Trajectory 3 position tracking

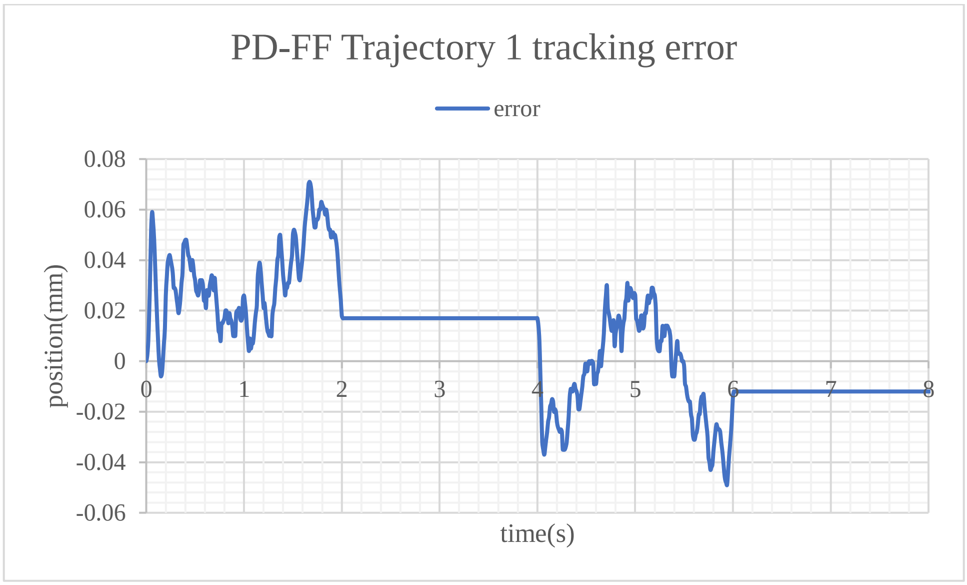


Figure 5.2.23 PD-FF controller Trajectory 1 tracking error

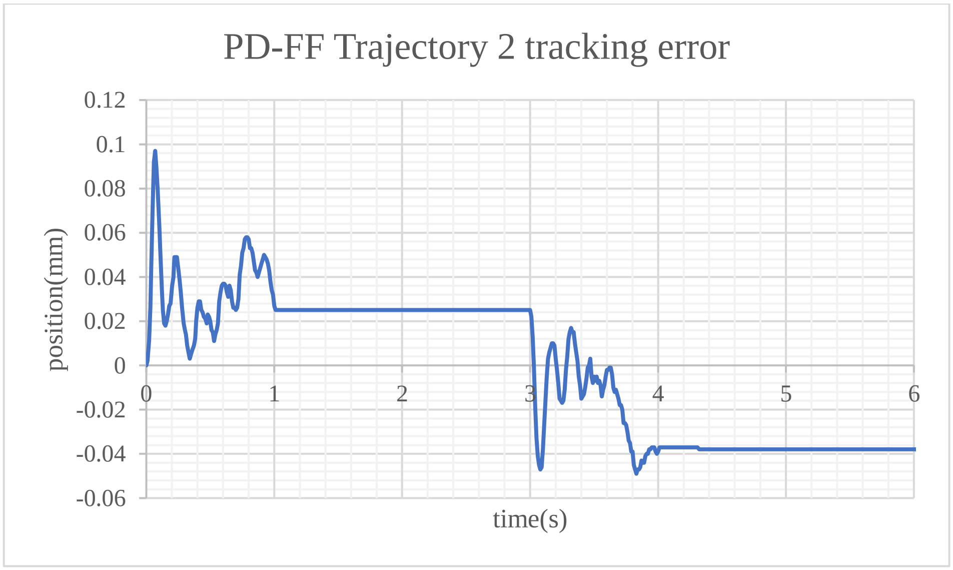


Figure 5.2.24 PD-FF controller Trajectory 2 tracking error

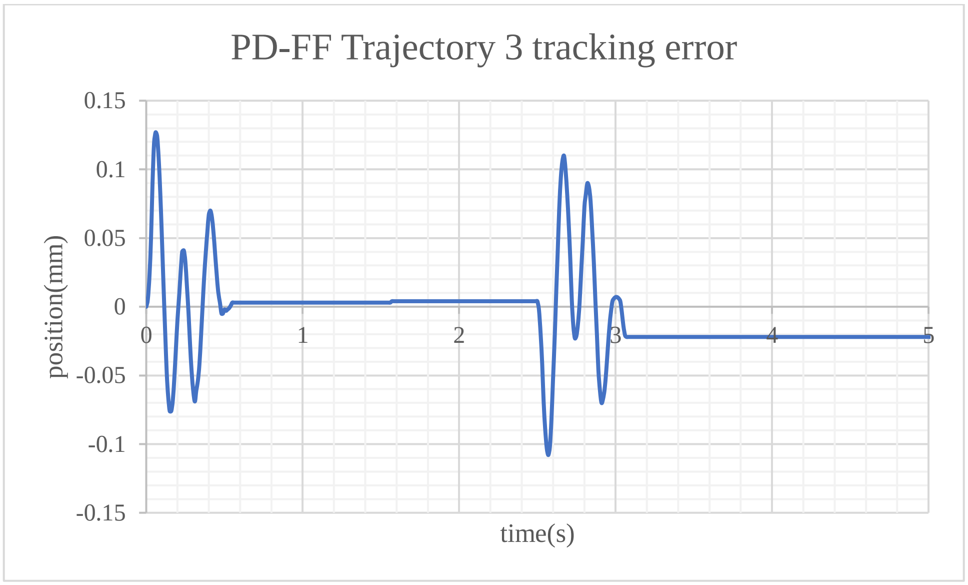


Figure 5.2.25 PD-FF controller Trajectory 3 tracking error

PD-FF control tracking error table			
	SSE forward (mm)	SSE backward (mm)	MAE (mm)
Trajectory 1 (30mm)	0.017	-0.012	0.020
Trajectory 2 (10mm)	0.025	-0.038	0.030
Trajectory 3 (5mm)	0.004	-0.022	0.019

Table 5.2.3 PD-FF control tracking error table

The results for the PD-FF controller are presented in Figure 5.2.20 to Figure 5.2.25, and Table 5.2.3 summarizes the error. Compared with the results of the PD-F controller, the MAE decreased. The SSE for both forward and backward paths are similar to the PD-F controller. The stability issues for the PD-F controller have been eliminated; the controller is stable when reaching the fixed reference position. Also, the oscillation displayed in Figure 5.2.14 has been eliminated. However, both the PD-F and the PD-FF controllers continue to display some SSEs that need to be considered.

5.2.7 Experiment on the Positioning PID controller with static friction cutoff compensator

As stated above, both the PD-F and the PD-FF controllers continue to exhibit some SSEs. Using the feedback friction estimation from the PD-F case, when the system is very near the desired final position, the speed is much greater than the desired speed. Therefore, the friction estimation is large, which leads to the control output being enlarged also. This increased control output causes the system to overshoot the desired final position. This, in turn, leads to the PD controller trying to output negative torque to stop the system. Once the system stops, the PD controller continues trying to output negative torque back to the desired final position; however, due to the system having already stopped, there is no speed nor desired acceleration. Therefore, there is no compensator to help the PD controller return to the desired final position, and it becomes stuck at the current position. Thus, the PD-F controller always has an overshoot and becomes stuck, which causes the SSE displayed in Table 5.2.2

The PD-FF controller is the opposite of that described above. Based on the reference trajectory speed, when the system is near the desired final position, the reference trajectory speed is already zero, as is the reference acceleration. Again, there is no compensator to help the PD controller overcome the static friction and then reach the desired final position. The system then becomes stuck at the current position. Therefore, there is always some undershoot at the final desired position, which causes the SSE in Table 5.2.3

To solve the SSE problems, a compensator needs to be applied when the system is near the desired final position. The compensator was tested by applying static friction F_s to the control output. When the system is within the resolution range of the linear encoder (0.005mm), the control output is set to zero as a cutoff to stop the motion to avoid oscillation.

The control law used in this test is the simple PID controller that was applied in Section 5.2.3. The control law *u(t)* with static friction compensator and cutoff function (PID-SF) is presented below:

$$
\begin{aligned}
&e_f(t) = x_f - x(t) \\
&u_{PID}(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{d}{dt} e(t) \\
&u(t) = \begin{cases} u_{PID}(t) + \mathrm{sgn}(u_{PID}) F_s & \\ 0 & (-0.006 \le e_f(t) \le 0.006) \end{cases}
\end{aligned} \tag{5.22}
$$

The term x_f is from Equation 5.10 – the desired final position.

The sign function used in Equation 5.22 ensures the static friction compensator has the same sign for the PID control output u_{PID}. The cutoff range is between -0.006mm and +0.006mm. The reason for this range, instead of using ±0.005mm, is to avoid the problem with the encoder. Since the encoder resolution is also 0.005mm, during the encoder test, at the location of 0.005mm, the encoder tended to jump between 0.005 and 0.006mm. Also, the encoder reading below and equal to 0.005mm trend to jump around. To avoid this problem, this control law employed 0.006mm as the cutoff range.

Equation 5.22 was then applied to the Arduino code and the sampling time was selected as 0.01s. The parameters for the PID controller are the same as in Section 5.2.3. The results of the tracking are presented below:

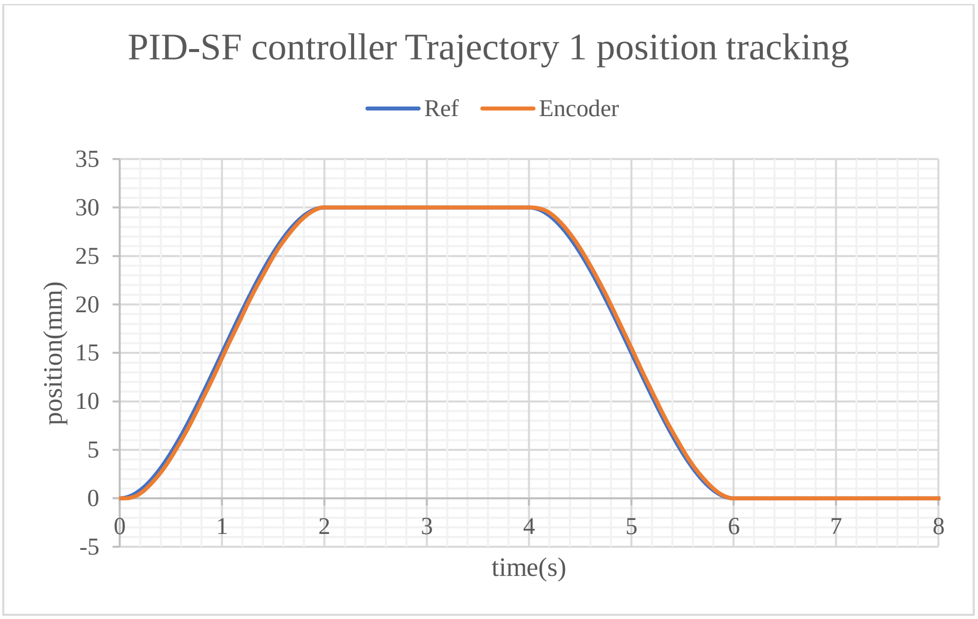


Figure 5.2.26 PID-SF controller Trajectory 1 position tracking

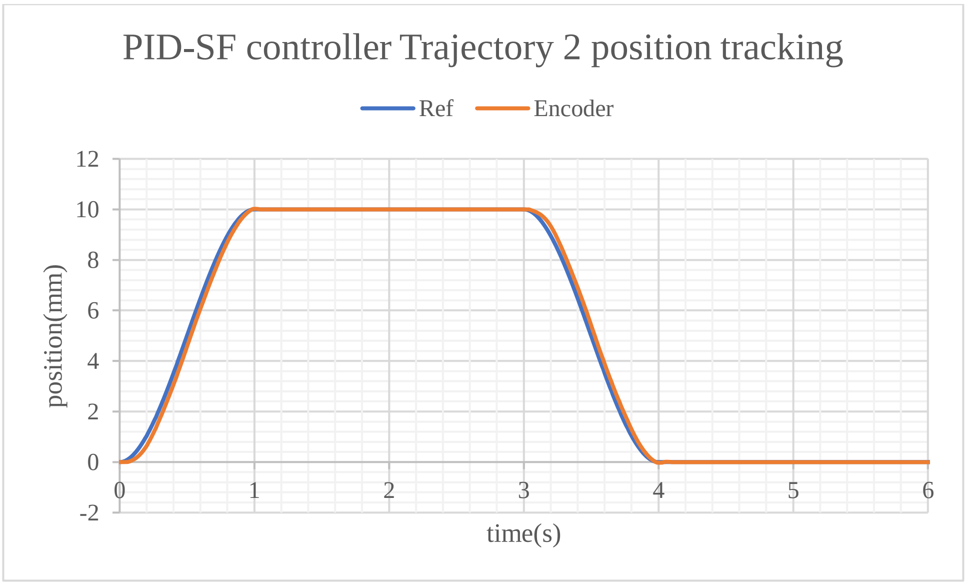


Figure 5.2.27 PID-SF controller Trajectory 2 position tracking

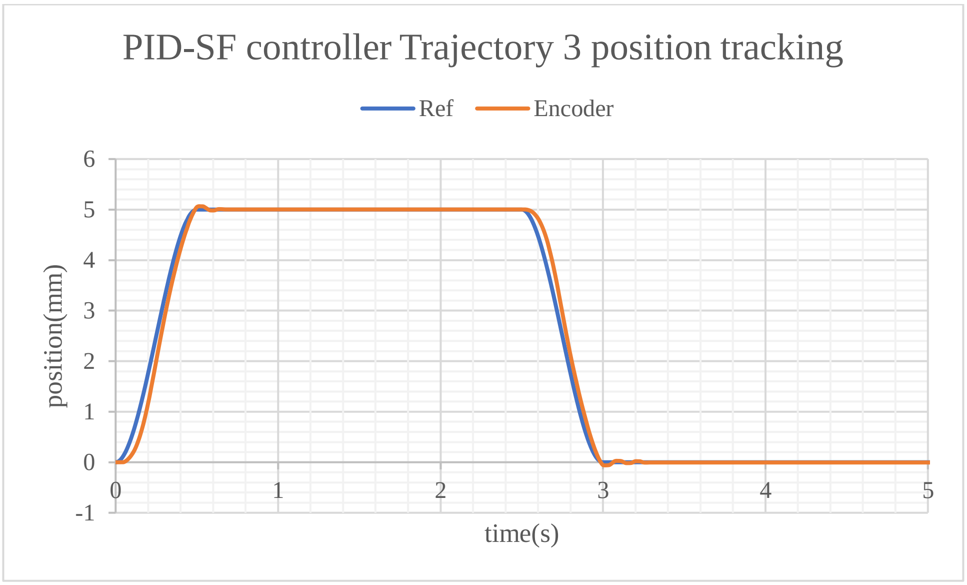


Figure 5.2.28 PID-SF controller Trajectory 3 position tracking

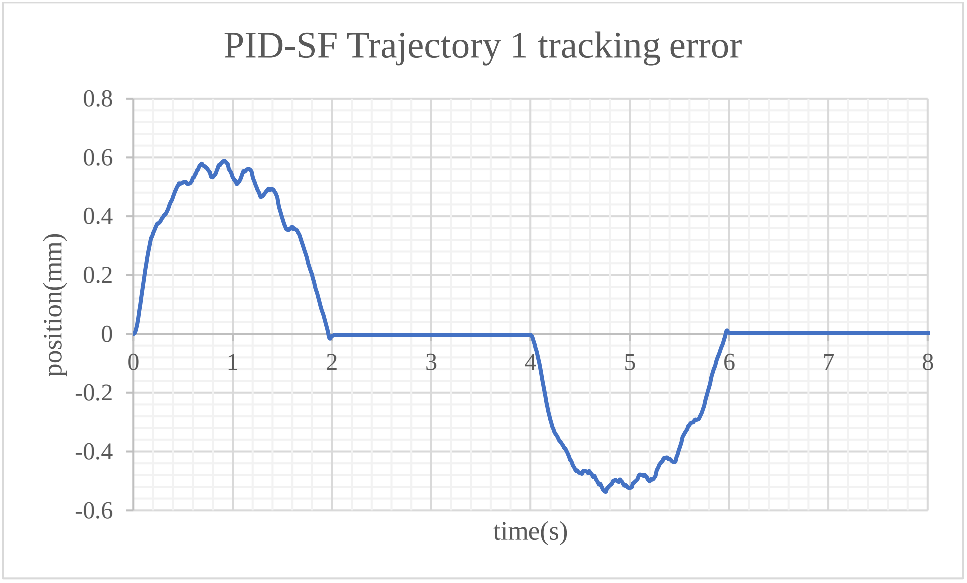


Figure 5.2.29 PID-SF controller Trajectory 1 tracking error

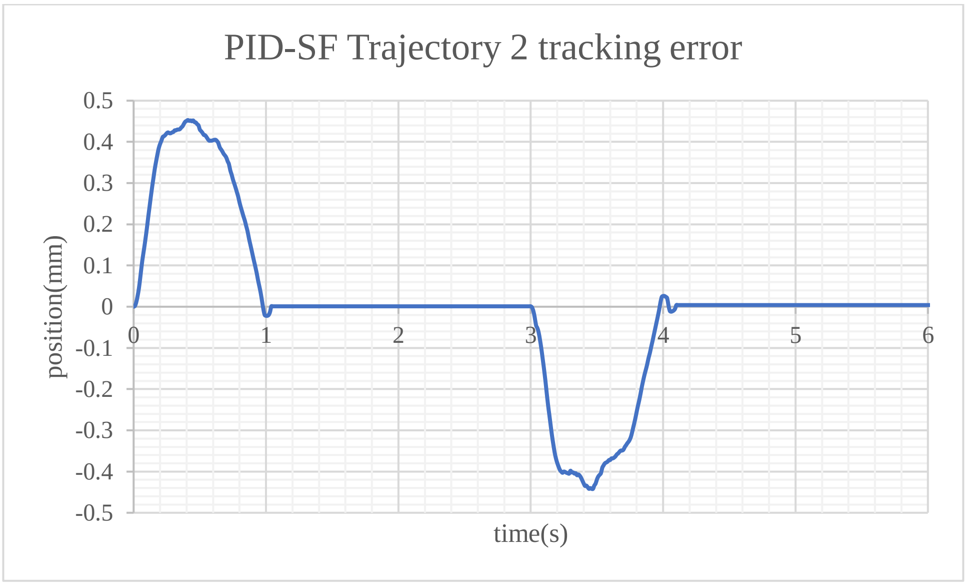


Figure 5.2.30 PID-SF controller Trajectory 2 tracking error

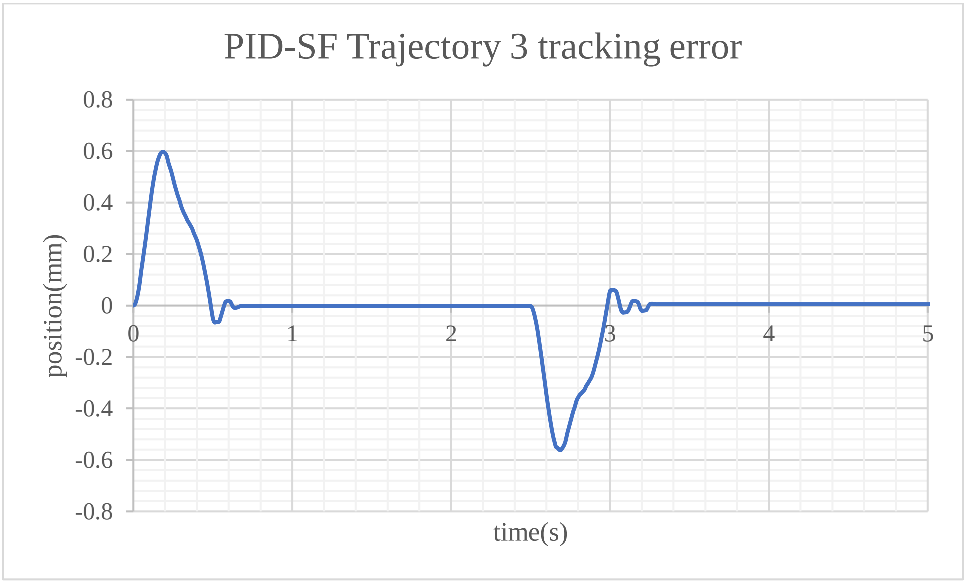


Figure 5.2.31 PID-SF controller Trajectory 3 tracking error

PID-SF control tracking error table			
	SSE forward (mm)	SSE backward (mm)	MAE (mm)
Trajectory 1 (30mm)	-0.003	0.004	0.193
Trajectory 2 (10mm)	-0.001	0.004	0.100
Trajectory 3 (5mm)	-0.002	0.005	0.067

Table 5.2.4 PID-SF control tracking error table

Figure 5.2.26 to Figure 5.2.31 present the tracking results of the PID-SF controller, and Table 5.2.4 presents the tracking error summary. The SSE of the PID-SF controller has greatly decreased in both directions compared with the PID, the PD-F, and the PD-FF controllers. The SSE is within the range of encoder resolution also, although the MAE is larger than the PD-F and the PD-FF controllers. However, the results prove that the above control law can be used to decrease the SSE to within the resolution limitations of the encoder.

5.2.8 Multiple stage control

Now that all the controllers have been tested, and based on the results, a multiple stage controller (MSC) is proposed. The PD-FF controller has a smaller MAE than the PD-F controller; however, both controllers suffered substantial SSE issues. The PID-SF controller has a very small SSE compared with the PD-FF and the PD-F controllers, but its MAE is large.

Therefore, the new MSC controller takes advantage of the smaller MAE of the PD-FF controller and the smaller SSE of the PID-SF controller to form a new controller.

The new MSC controller applies different control strategies for different stages of the reference position. The PD-FF controller has excellent tracking during the beginning and middle stages. The PID-SF controller has a smaller SSE during the final stage of the reference position. Therefore, the control law for MSC is as follows:

$$
\begin{aligned}
& e_f(t) = x_f - x(t) \\
& u_{PD} = K_p e(t) + K_d \dot{e}(t) \\
& u(t) = \begin{cases} JLM \cdot \ddot{x}_d(t) + u_{PD} + \hat{T}_f(\dot{x}_d(t)), & (e_f(t) < -e_{stage} \,\|\, e_f(t) > e_{stage}) \\ u_{PD} + \operatorname{sgn}(u_{PD}) \cdot F_s & ,(-e_{stage} \le e_f(t) \le e_{stage}) \\ 0 & ,(-0.006 \le e_f(t) \le 0.006) \end{cases}
\end{aligned}
\tag{5.23}
$$

The function $\hat{T}_f(\dot{x}_d(t))$ can be referenced to equations 5.20 and 5.21. The term x_f is the final desired position. The term e_{stage} is a constant to determine when the control law changes the control stage. As presented in Equation 5.23, the initial stage is the PD-FF controller, when the desired final error e_f reaches a specific valve as e_{stage} the control method changes to the PD controller with static friction compensator control (PD-SF). Finally, when the system moves within the range of the encoder resolution limitation, the controller cuts off and the output is zero to stop motion.

During the multiple tests, e_{stage} was chosen as 0.1mm for the best results. The K_p and K_i parameters are the same as used in the PD-F and the PD-FF controllers. The friction parameters and system parameters are in Table 4.2.5.

For example, the MSC control law for Trajectory 1 is as follows. Initially, the Stage 1 control method was the PD-FF controller, but when the system reached 29.900 mm, Stage 2 began, in which the control method changed to the PD-SF controller, and then finally, when it reached to 29.994 mm, the system output changed to zero voltage. The backward movement is vice versa. During the multiple tests, one of the advantages of the MSC controller is that no matter where the system is in Stage 2, the PD-SF with cutoff control can guarantee the system reaches in the range of ±0.006mm SSE.

The tracking results for the MSC controller for trajectories 1, 2, and 3 are presented below:

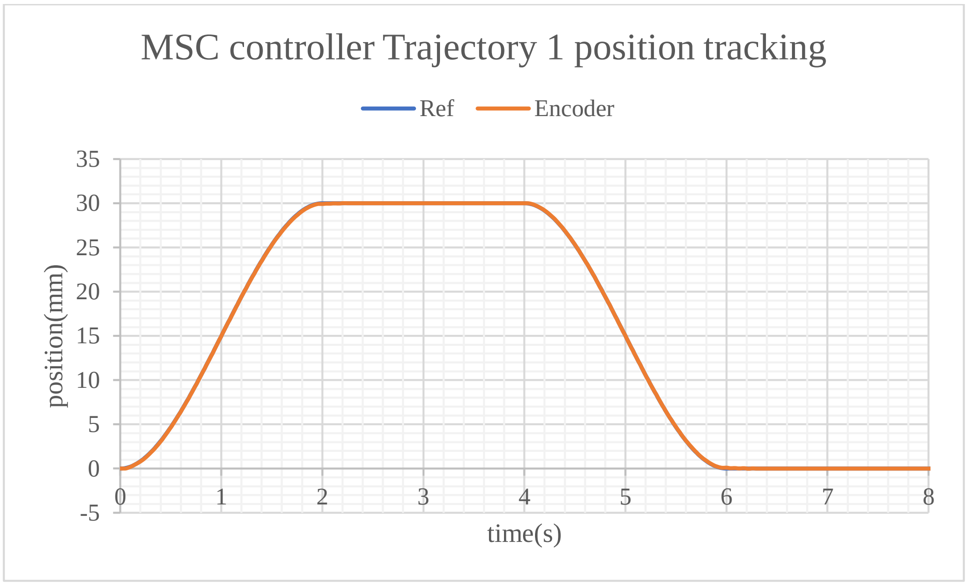


Figure 5.2.32 MSC controller Trajectory 1 position tracking

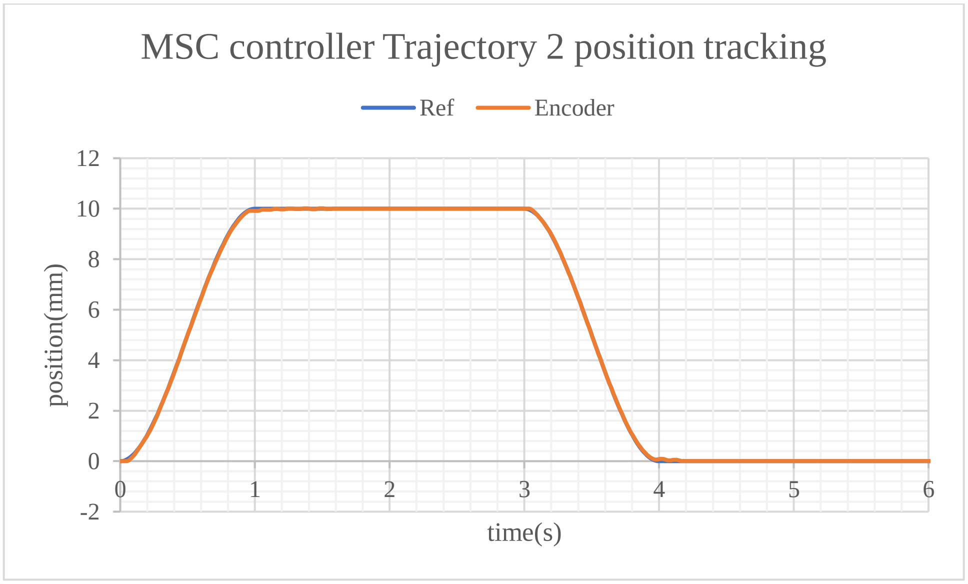


Figure 5.2.33 MSC controller Trajectory 2 position tracking

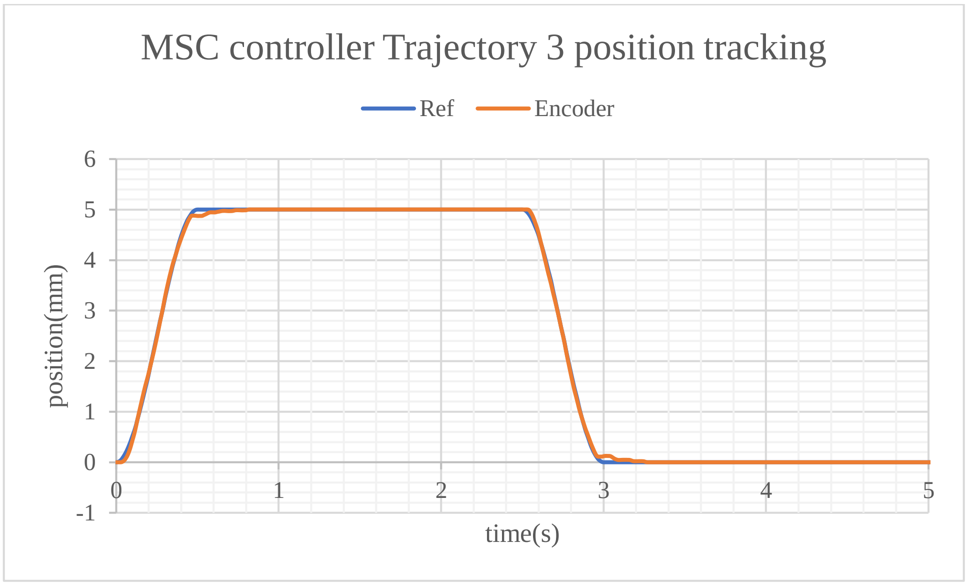


Figure 5.2.34 MSC controller Trajectory 3 position tracking

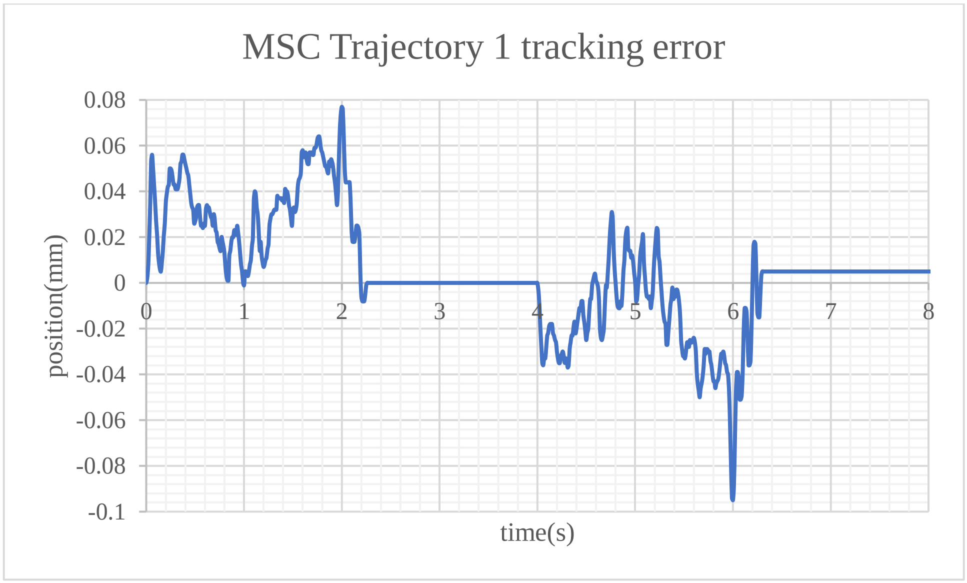


Figure 5.2.35 MSC controller Trajectory 1 tracking error

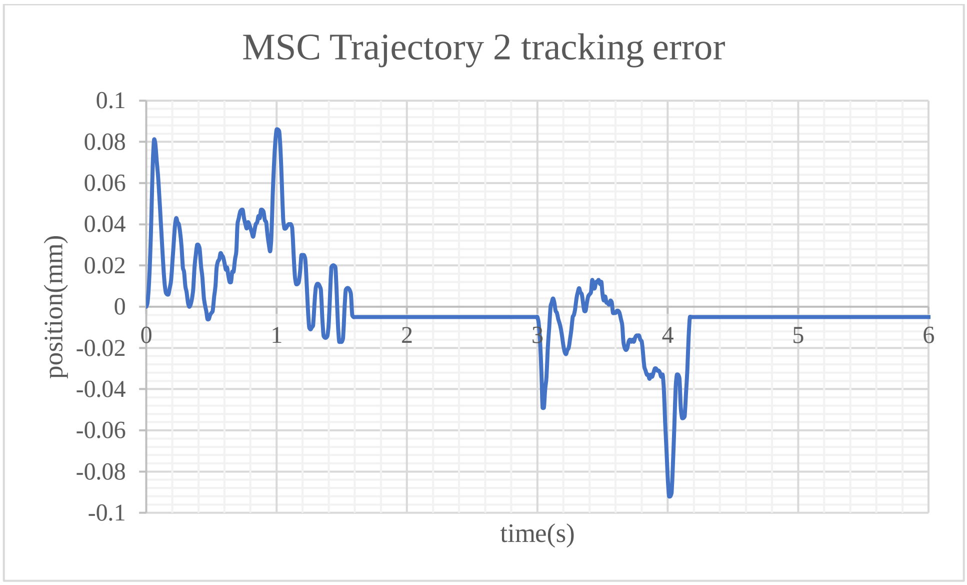


Figure 5.2.36 MSC controller Trajectory 2 tracking error

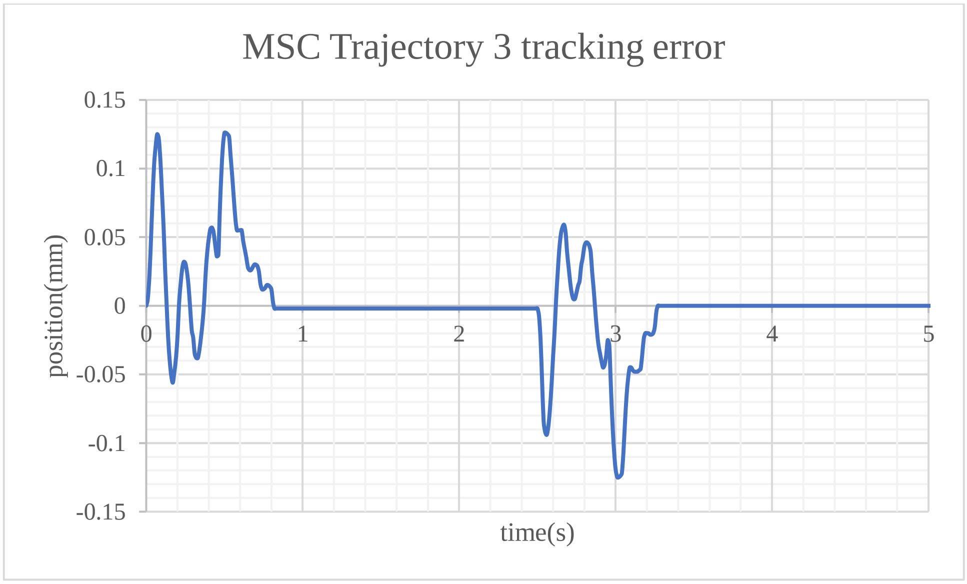


Figure 5.2.37 MSC controller Trajectory 3 tracking error

MSC controller tracking error table			
	SSE forward (mm)	SSE backward (mm)	MAE (mm)
Trajectory 1 (30mm)	0	-0.005	0.017
Trajectory 2 (10mm)	-0.005	-0.005	0.014
Trajectory 3 (5mm)	-0.002	0	0.015

Table 5.2.5 MSC controller tracking error table

The results of the MSC controller are presented in Figure 5.2.32 to Figure 5.2.37. Table 5.2.5 summarizes the tracking error. Comparing Table 5.2.1 to 5.2.4, the MSC controller provides the best results in term of SSE and MAE. The MSC controller not only has as small an MAE as the PD-FF controller, but also as minimal SSE error as the PID-SF controller. Furthermore, the MSC controller overcomes the oscillation issues of the PD-F controller.

5.2.9 Result comparison between the controllers and robustness test

In this section, the horizonal comparison between the controllers for the different cases is presented then discussed. The traditional PID controller, the PD-F controller, and the MSC controller are compared. The PID-SF controller and the PD-FF controller were integrated into the MSC controller; therefore, they are included in the horizon comparison.

For all three trajectory cases, the tracking error for the three controllers is presented below:

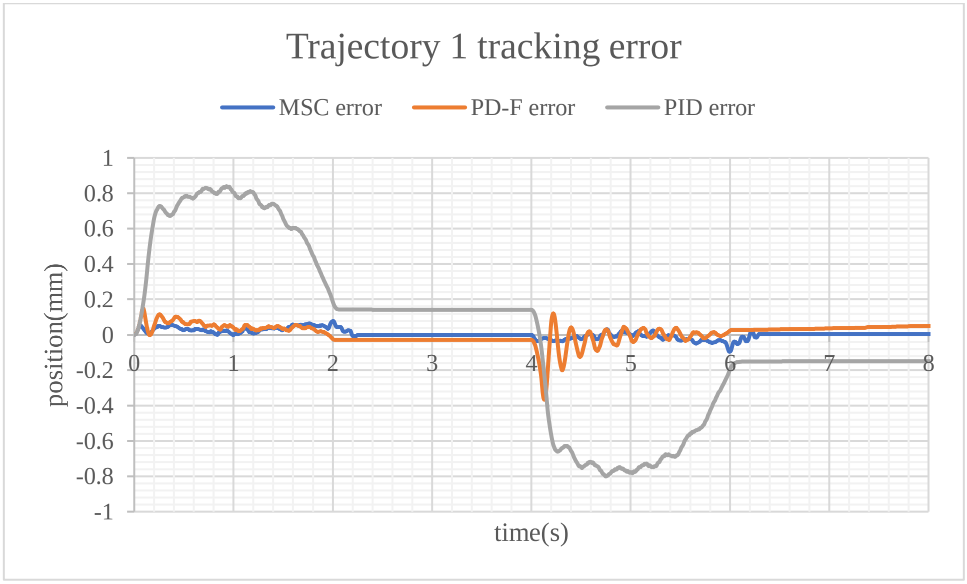


Figure 5.2.38 Trajectory 1 tracking error

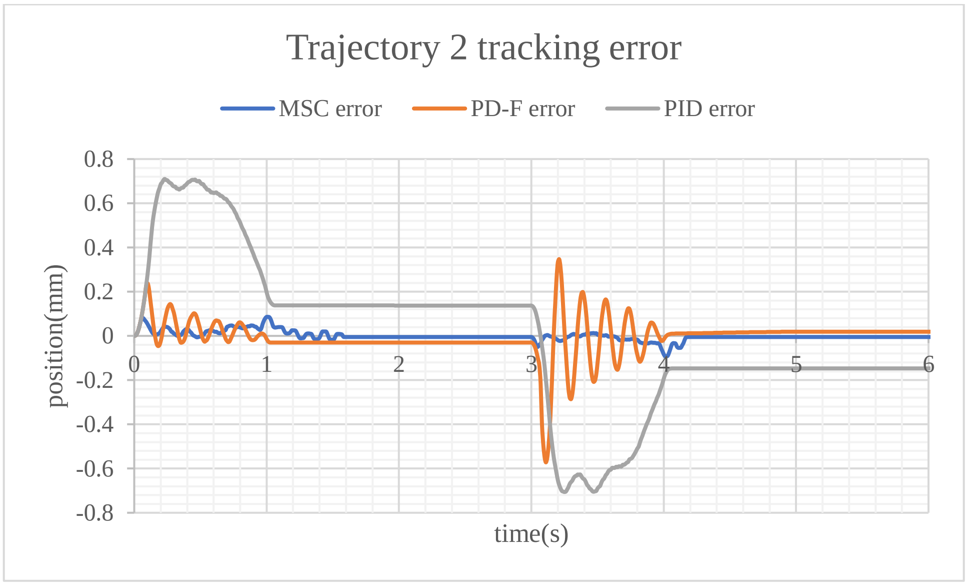


Figure 5.2.39 Trajectory 2 tracking error

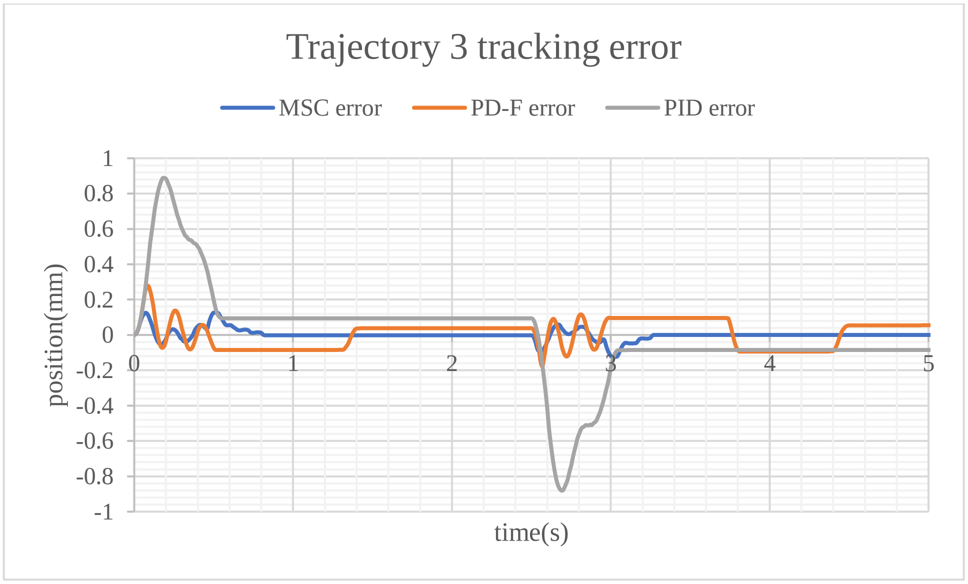


Figure 5.2.40 Trajectory 3 tracking error

Tracking error table			
Trajectory 1 (30mm)	SSE forward (mm)	SSE backward (mm)	MAE (mm)
PID	0.142	-0.150	0.384
PD-F	-0.028	Unstable	0.041
MSC	0	-0.005	0.017
Trajectory 2 (10mm)	SSE forward (mm)	SSE backward (mm)	MAE (mm)
PID	0.137	-0.147	0.267
PD-F	-0.03	-0.019	0.047
MSC	-0.005	-0.005	0.014
Trajectory 3 (10mm)	SSE forward (mm)	SSE backward (mm)	MAE (mm)
PID	0.094	-0.085	0.174
PD-F	-0.058, 0.038	0.096, -0.094, 0.054	0.070
MSC	-0.002	0	0.015

Table 5.2.6 Trajectory tracking error table

From the results of Figure 5.2.38 to Figure 5.2.40 and Table 5.2.6, the MSC controller is best in both SSE and MAE. For all three cases, the PID controller has the worst tracking results. The MSC controller has the best tracking performance.

The MSC controller always provides a stable output at the constant reference position, whereas the PD-F controller has some oscillation. For all three cases, the PD-F controller has a more significant tracking error at the initial stage compared with the MSC controller. For example, for Trajectory 1, from time 0 to 1s, the tracking error of the PD-F controller is more significant than the MSC controller. Also, for the same trajectory, from time 4 to 5s, which is the initial stage of moving backward, the PD-F controller displays a considerable tracking error compared with the MSC controller. A similar phenomenon occurs for Trajectory 2 (Figure 5.2.39).

The reason for this phenomenon is the feedback speed for the PD-F controller. In the MSC controller, the friction compensator is based on reference speed. However, in the PD-F controller, the friction compensator is based on feedback speed. In Figure 5.2.41, it is evident that the feedback speed (Speed_LRE) has some delay compared with the reference speed. The slower response speed means the friction observer cannot estimate enough friction to compensate real friction. Therefore, the control input is not enough, which causes a more significant tracking error during the beginning stage. One of the reasons for the speed delay is the LRE speed filter. This filter causes some time delay. The second reason is that the real feedback speed is not fast enough to match the reference speed. Both these reasons cause the speed time delay. In the MSC controller, the friction compensator is based on the reference speed; therefore, the estimated friction force is more accurate and faster than the feedback friction observer. Thus, the MSC tracking error is much smaller than the PD-F tracking error at the initial stage.

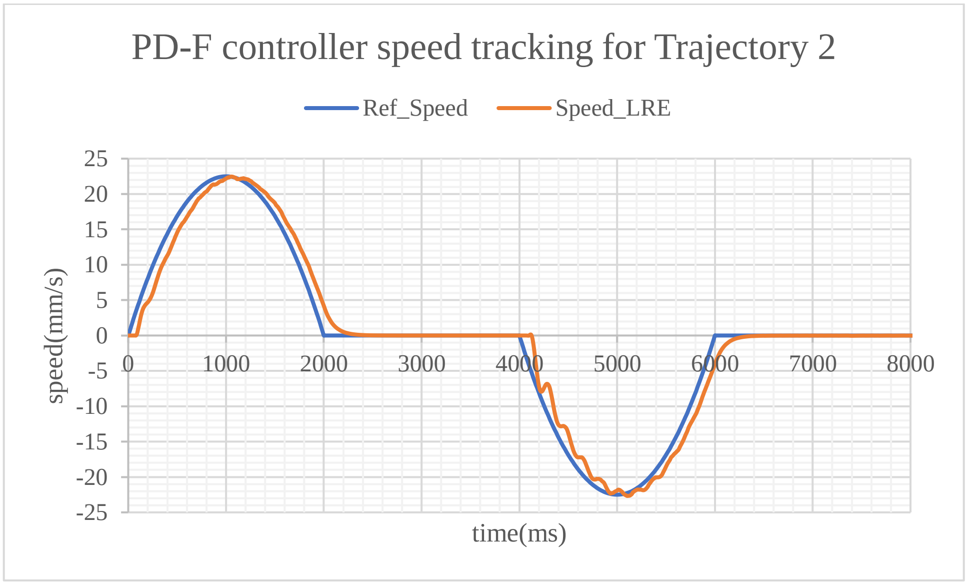


Figure 5.2.41 PD-F controller speed tracking for Trajectory 2

So, the MSC controller provides the most accurate results for reference tracking. Next, the robustness test was performed on the PD-F controller and the MSC controller. The PID controller already recorded the worst results in the above test; therefore, there is no need to repeat the test for this controller.

The robustness test for the controllers requires a disturbance. Therefore, an extra weight was added to the ball screw sliding table. As illustrated in Figure 5.2.42, the extra weight was a robotic car chassis. The weight of the car chassis was 16.7kg. Trajectory 1 was used for the robustness test. The tracking results are presented below:

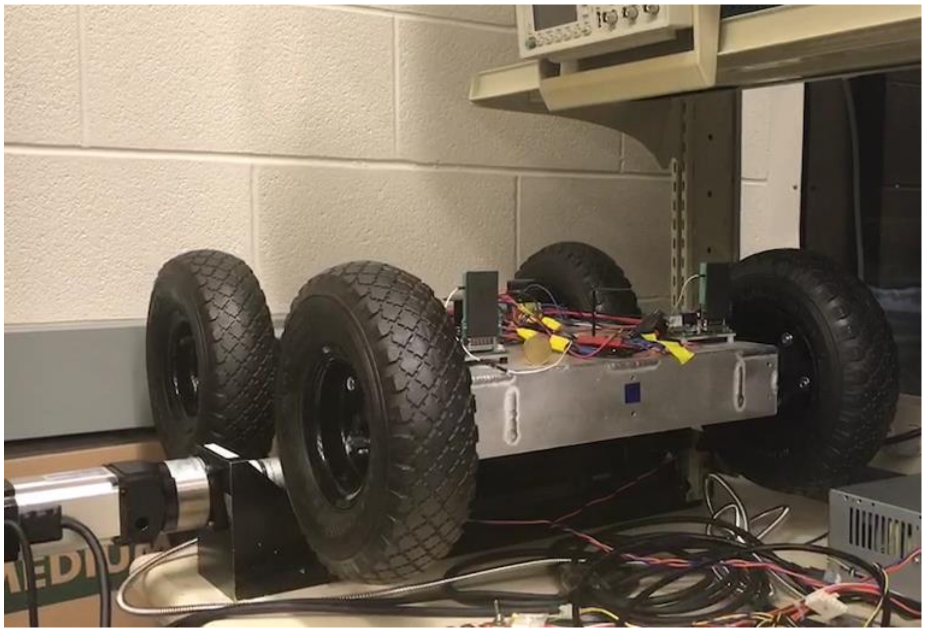


Figure 5.2.42 Robustness test setup

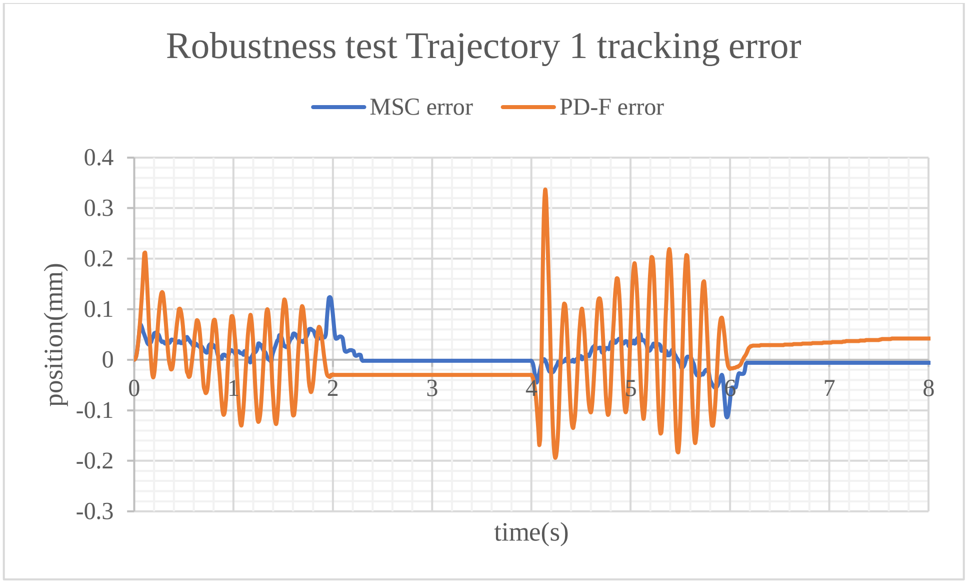


Figure 5.2.43 Robustness tracking error MSC vs PD-F

Robustness tracking error table – Trajectory 1			
	SSE forward (mm)	SSE backward (mm)	MAE (mm)
MSC	-0.002	-0.006	0.018
PD-F	-0.03	Start at 0.029	0.056

Table 5.2.7 Robustness tracking error table

Figure 5.2.43 and Table 5.2.7 present the results of the robustness test tracking error for both controllers. The MSC controller maintained a small SSE and MAE. The MSC controller performance is still better than the PD-F controller. Comparing the results in Table 5.2.6 with Table 5.2.7, the MSC controller still has a small SSE error in the resolution range of the linear encoder. When the weight was added, both controllers displayed increased MAE. Therefore, the extra weight affected the system response. Comparing the MAE before and after the weight was added, the difference is not significant for the MSC controller (only 0.001mm). The MAE result for the PD-F controller before and after the weight was added was more significant (0.009mm). Thus, the MSC controller is more robust than the PD-F controller.

The final test was the consistency test for the MSC controller. Consistency is critical to the linear actuator system as it ensures that in each run the sliding table reaches the same location. The consistency test was conducted using Trajectory 1. Four runs were performed using the MSC controller. The tracking error results are presented below:

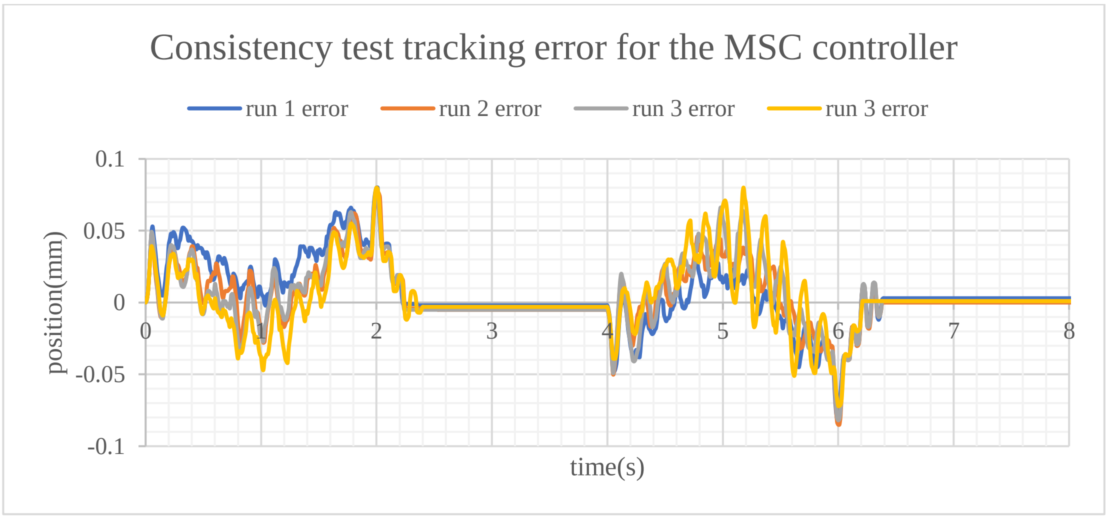


Figure 5.2.44 Consistency test tracking error for the MSC controller

Consistency tracking error table – MSC controller			
	SSE forward (mm)	SSE backward (mm)	MAE (mm)
Run 1	-0.002	0.003	0.016
Run 2	-0.005	0	0.014
Run 3	-0.005	0.001	0.015
Run 4	-0.003	0.001	0.015

Table 5.2.8 Consistency tracking error table – MSC controller

As Figure 5.2.44 and Table 5.2.8 illustrate, the same trajectory was run four times. Each SSE is in the range of the encoder resolution (±0.005mm), and the MAE has a consistency range between 0.016 and 0.015. Comparing the results with the built-in PWM control (Figure 5.2.2), the consistency of the MSC controller is much better than the built-in control algorithm, and the MSC controller provides higher tracking accuracy also.

5.3 Discussion of the Results of the Experiment and the Simulation

In the simulation, the PID controller, the PD-F controller, and the PD-DOB controller were used. During the simulation, two trajectories were used: the smooth trajectory and the sinusoidal trajectory. Also, two conditions – the ideal condition and the real condition – were simulated to test the robustness of the controllers.

For the smooth trajectory, the PD-DOB controller had the best tracking performance in both conditions. The PD-F controller came second. However, for the real condition, both controllers had stability issues when the trajectory reached the constant position. However, although the PID controller had poor tracking results, it did not have stability issues when dealing with the constant position.

For the sinusoidal trajectory, the PD-DOB controller dominated the sinusoidal trajectory in terms of tracking performance. The PD-DOB controller had the smallest MAE for both the ideal condition and the real condition. The sinusoidal trajectory proved the disturbance rejection ability of the PD-DOB controller. Due its disturbance observer, PD-DOB controller could observe any disturbance added to the system and then compensate. Thus, the uncertainty in control was eliminated.

In the experiment stage, only the smooth trajectory was applied. In most industrial applications, after a smooth path, a stop is required, which allows another system to finish a different task. The smooth trajectory applied in the experiment is slightly different from its use in the simulation. In the experiment, the smooth trajectory moves backward and forward. In the experiment, instead of using one smooth trajectory path, three trajectories were added to the system. Each trajectory presented different cases: A long travel distance with high speed (Trajectory 1); a middle travel distance with middle speed (Trajectory 2); and a short travel distance with high acceleration (Trajectory 2). In the experiment stage, the PID controller, the PD-F controller, and the PD-DOB controller were all tested. However, the PD-DOB controller did not work in the microcontroller, perhaps because of the slow sampling time in the microcontroller.

In addition to the two working controllers, an onboard PWM controller was tested also, although the input was a step motor position. A Position PD controller with friction feedforward estimation (PD-FF) and a PID controller with a static friction cutoff compensator (PID-FF) were proposed and implemented also. Finally, an MSC controller, which combines the advantages of the PID-FF and the PD-FF controllers was implemented and tested.

Regarding the results, the PD-F controller was in line with the simulation result, which decreased the tracking error compared with the PID controller. Moreover, as in the simulation, the PD-F controller had stability issues at the constant position stage. The system oscillated at the final desired position. The MSC controller provided the best results in terms of tracking error MAE and SSE at the desired final position. The MSC controller's SSE for all three trajectories were within the limited resolution range of the linear encoder. Moreover, the MAE of the MSC controller was smaller than either the PD-F or the PID controllers.

The MSC controller then underwent a robustness test and a consistency test. The results of the robustness test demonstrate that if extra weight is added to the system, the MSC controller SSE could continue to reach the limitation resolution of the linear encoder with a smaller MAE. This result proves the robustness of the MSC controller. In the consistency test, all four runs displayed very good consistency, unlike the servo drive onboard the PWM controller, which has major consistency issues. Thus, the MSC controller helps the ball screw system improve not only position accuracy, but also trajectory tracking ability.

6 CHAPTER CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The ball screw linear actuator is widely used in different areas, such as in milling machines, 3D printers, and coordinate measuring machines. The ball screw mechanic system has the advantages of high speed, less noise, and high efficiency. In the ball screw system, the AC servo motor combines with the AC servo drive is the proposition system that drives the ball screw mechanic. In the system, nonlinear friction is one of the critical factors that affects accuracy. The primary object of this project is to use nonlinear friction to improve the accuracy of the ball screw linear system.

There are different types of friction models that describe the friction phenomena. In addition to the common static friction model (coulomb friction, static friction, and viscous friction model), a dynamic LuGre friction model has been widely researched. This model includes the static friction region presiding phenomena, the friction memory, and the Stribeck friction curve phenomena. Therefore, the LuGre friction model is one of the most accurate friction models that describe friction phenomena.

A computing system is needed to apply the control algorithm. To include the advantage of clock speed, real-time control, and interrupt ability, the Teensy 3.5 microcontrollers were chosen for this project.

Position feedback is another important aspect of this project. A decoding method for the linear encoder is needed to obtain an accurate position. In this project, different methods to read the linear encoder signal were tested. Finally, a rule-based method for decoding was chosen and applied.

To apply the friction base control, a torque control power unit for the AC servo drive was needed. In this project, a custom CDAC was made. The CDAC could output analog voltage $\pm 10V$, which is the maximum torque that the AC motor could produce. In the test, the CDAC met the design target, and the cost of the CDAC is much lower than the existing product on the market.

Then, the ball screw system model and LuGre friction model were constructed then followed the system identification process. System identification was divided into two sections: the LuGre friction identification and the ball screw system identification. The LuGre friction model parameters were first identified by conducting experiments based on Newton's first law of motion. Then, by applying torque ramp inputs to the ball screw system, the ball screw plant model parameters were identified, and the LuGre friction parameters were adjusted based on the ramp experiment output.

Based on the parameters identified, a simulation using the Matlab Simulink was performed. Instead of using the standard step input, a smooth trajectory input and a sinusoidal trajectory input were applied in the simulation. The control methods were selected for the PID controller, the PD-F controller, and the PD-DOB controller. The results demonstrate that the PD-DOB controller performed best in terms of tracking, with the PD-F controller second. Both the PD-DOB controller and the PD-F controller had excellent tracking performance compared with the conventional PID controller. However, both the PD-F controller and the PD-DOB controller displayed stability issues when at the fixed reference position area.

The controller experiment test was performed next. The onboard control method using the PWM position control was tested first. The following proposed controller test was based on the torque control mode. The smooth trajectory was chosen as the reference position. The PID controller, the PD-F controller, and the PD-DOB controller were first applied to the microcontroller. The parameters were based on the system-identified parameters. In the test, the PD-F controller had more accurate tracking results compared with the PID controller. However, the PD-F controller continued to suffer from constant position stability issues, as it did in the simulation. The PD-DOB controller did not work in the implementation process; the controller lost control and became unstable due to the lower sampling time. A new proposed MSC controller was introduced. The new MSC controller displayed improved initial stage tracking performance compared with the PD-F controller and decreased the final position SSE.

Thus, the MSC controller with LuGre friction compensator could achieve an SSE within the limitation resolution range of the linear encoder. The tracking error is lower than either the PD-F or the PID controllers. The consistency of the MSC controller is much better than the servo drives onboard control algorithm. Furthermore, the MSC controller displayed a better robustness when the system weight changed.

Thus, this thesis proves that using the proposed MSC controller with LuGre friction could help the existing ball screw linear actuator system improve motion accuracy and consistence at a reasonable cost.

6.2 FUTURE WORKS

For the future works, several types of future research could be implemented.

1. **Implementing the PD-DOB controller in the ball screw system**. From the simulation, the PD-DOB shows dramatic improvement on disturbance rejection. Implement the PD-DOB controller into physical ball screw system could possible increase control accuracy. However, the current microcontroller has a limited sampling time, which causes unstable issues in the PD-DOB controller. A faster microcontroller could solve this problem.
2. **Friction observer based on UKF could be applied.** In the literature, the UKF could be applied as the friction compensator. The current MSC controller's friction compensator is based on feedforward reference speed; there is no feedback position to correct friction estimation. The UKF could using the feedback speed to correct LuGre friction that could increase tracking accuracy. The UKF also requires a faster microcontroller due to its computational complexity.
3. **Higher resolution and accuracy linear encoder.** The current linear encoder supplied by the sponsor only has a one cycle resolution of 0.005mm. Therefore, the current control system could only achieve SSE with in the $\pm$0.006mm. The higher resolution linear encoder could improve the system position tracking accuracy

7 APPENDIX

7.1 TERMINOLOGY

x = position (mm)

x_d = desired position (mm)

x_{sen} = encoder position (mm)

V stands for voltage

T_m = Motor input torque (V)

J = moment of inertia of ball screw and motor (V.s^2/mm)

L = radians to displacement conversion gain(mm/rad)

M = mass of the sliding table (kg)

x = displacement of sliding table(mm)

θ = angler displacement of motor (rad)

F_f = LuGre friction force (V)

T_f = LuGre friction torque (V.L)

σ_0 = Bristle stiffness coefficient (V/mm)

σ_1 = Bristle damping coefficient(V.s/mm)

σ_2 = viscous friction coefficient(V.s/mm)

z = deformation of bristles (mm)

F_c = Coulomb friction (V)

F_s = Maximum static friction(V)

v_s = Stribeck velocity (mm/s)

δ_v = Stribeck fitting parameter. (unit less)

K_p = PID proportional gain

K_i = PID integration gain

K_d = PID derivative gain

K_z = friction observer correction term

7.2 UNIT CONVERSION FOR LUGRE FRICTION

$$\dot{z} = \dot{x} - \frac{\sigma_0 |\dot{x}|}{g(\dot{x})} z$$

$$\frac{mm}{s} = \frac{mm}{s} - \frac{\frac{V}{mm} \cdot \frac{mm}{s}}{V} \cdot mm$$

$$g(\dot{x}) = F_c + (F_s - F_c) e^{-\left|\frac{\dot{x}}{v_s}\right|^{\delta_s}}$$

$$V = V + \left(V - V\right) e^{-\left|\frac{mm/s}{mm/s}\right|^{1}}$$

7.3 AC SERVO SYSTEM

The servo system is normally a closed loop system because a servomotor typically has a built-in encoder. The encoder signal generates the position and speed signal back to servo driver. The servo driver has its control algorithm that minimizes the feedback error. Compared to the steppe motor system, there is no feedback signal given to the stepper driver in the servo system, and therefore this system could have a higher accurate position operation. Normally, a servo system has three control methods: a position control using a pulse signal (PMW), a speed control using the analog voltage, and a torque control using analog voltage [56].

The AC servo system for this research is also provided by the sponsor. The AC servo system used in this project is the JMC-AC servomotor (60JASM504230K-17Z) and JMC-AC Servo Driver(JASD4002-17Z) shown in Figure 7.3.1. The JMC-AC servomotor has 400W of power, and it can produce a 1.27 N/m max torque with a 17-bit absolute encoder that is built into the motor. The supply voltage required is up to 220 V. The AC servo driver comes with the system that could provide 400 W of power, and the supply voltage is also required to be 220 V. It also could support a 17-bit absolute encoder. The detail specification of the motor in Table 7.1.

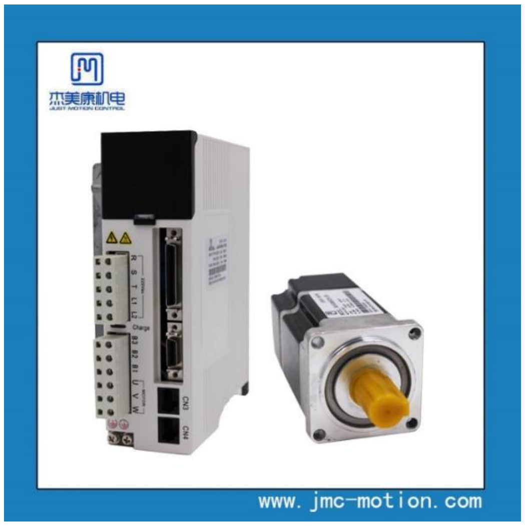


Figure 7.3.1 JMC AC servomotor and the servo driver [57]

Type	Supply Voltage	Power rating (W)	Rated torque (Nm)	Peak torque (Nm)	Rated current (A)

60JASM 504230K-17Z	220 V	400	1.27	4.45	2.75
	Max current(A)	Rotor Inertia (10^{-4}kgm^2)	Rated RPM	Max RPM	
	8.25	0.407	3000	5000	

Table 7.1 JMC AC servomotor specifications [58]

In the position control mode, the programmable controller produces a pulse width modulation (PWM) signal—this modulation creates an analog impulse voltage using digital means [59]. The PWM signal is sent to the servo driver, and the driver then uses this PWM signal as the reference set point. The PWM signal is a high voltage and low voltage signal that displays 0 and 1. The servo driver counts how many pulses have been sent, and it also measures the frequency between the pulses. The number of pulses is related to the reference of the angle. The frequency of the pulses determines the reference angular speed needed. Therefore, PWM signal needs to contain two pieces of information—namely the reference angle and reference speed. The servo driver decodes the PMW signal as a reference position and reference speed. By using this information, the servo driver can produce the controlled current to the servomotor. The built-in encoder then sends the encoder signal back to the driver, and the driver decodes the encoder position and speed signal. After this, the driver control algorithm produces the current to obtain the reference position and speed. The schematic of the PWM position control structure is seen in Figure 7.3.2.

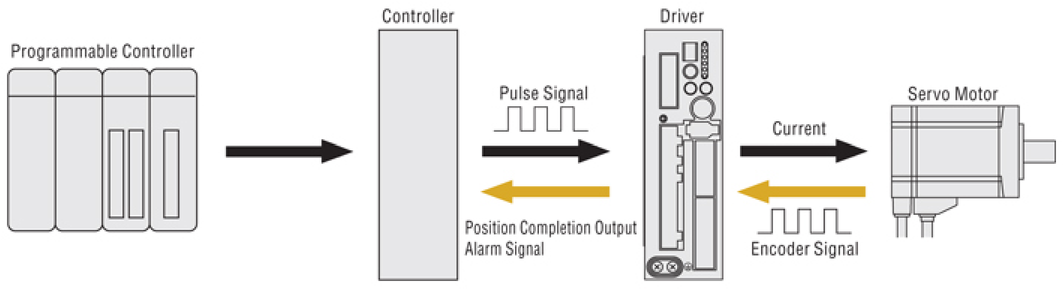


Figure 7.3.2 Position control using a pulse signal [56]

The control algorithm used in the JMC servo driver is in Figure 7.3.3, which is provided from the user manual. The basic idea for position control is to use the feedforward PI control. The position reference is the PWM signal, given that the signal contains the speed and position reference. The position signal first goes through the position proposition gain and the speed feedforward block. The speed feedforward block calculates the needed speed using the reference position. The position reference is subtracted from the position feedback to create position tracking error. The speed feedback signal is added to the position proposition gain and speed feedforward to create the speed tracking error. After this, the signal goes through the speed control loop. In the speed control loop, the signal passes through speed PI control and the torque feedforward is added. The torque feedforward calculation is done using the speed reference. The control signal is then transferred to the current, from the current inverter to the motor.

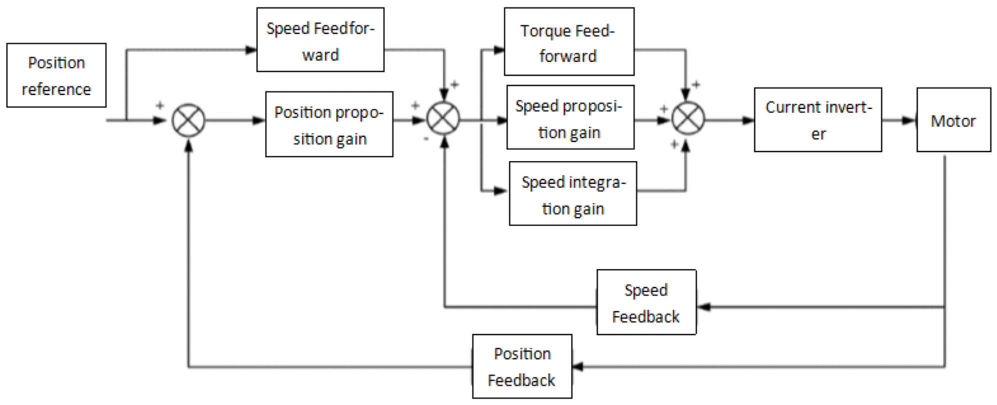


Figure 7.3.3 PWM position in the control block diagram [60]

In speed control mode, the speed reference signal is produced from the microcontroller and it goes through the speed control power unit, which then transfers the analog voltage. The servo driver takes the analog voltage as the reference speed target. The analog voltage directs a proportion to the speed reference. For the JMC servo driver and motor, the rated speed is 3000 rpm with a max input voltage of 10 V [43]. Therefore, the analog voltage to speed constant will be 0.003 rpm/V. The servo driver takes this reference voltage and then uses the build-in control algorithm to produce the corresponding current to the servomotor. The motor encoder provides speed feedback to the servo driver. By using this speed feedback, the control law in the servo driver corrects the out current to the motor which allows the motor to reach the reference speed. Figure 7.3.4 shows how speed control is done using an analog voltage structure.

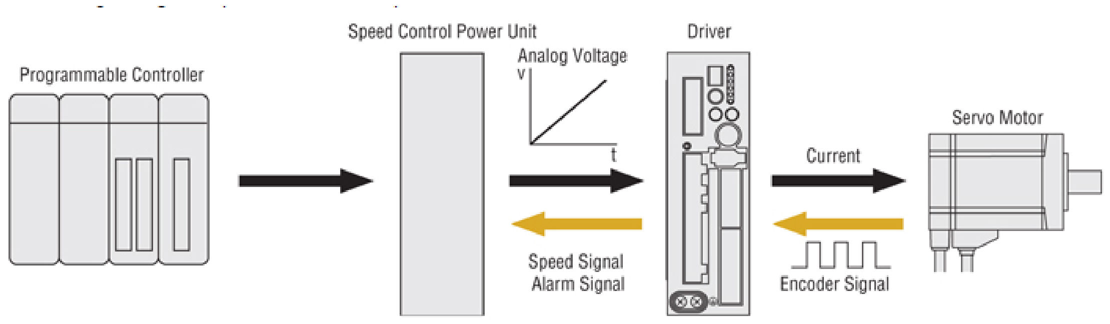


Figure 7.3.4 Speed control using analog voltage [56]

Figure 7.3.5 can be used as a reference for the control law behind the speed control. The server drive manual did not provide the complete control block diagram for speed control. Therefore, this thesis can only take the general servo driver speed control block diagram as seen in Figure 7.3.5. The speed command goes through the speed adjuster, and the speed adjuster will also take the encoder speed and feedback signal. The speed adjuster tries to minimize the error between the reference speed signal and speed feedback signal by producing the corresponding current command. After this, the current command is sent to the current adjuster. The current adjuster also takes the current feedback to make sure the command current is achieved. The current adjuster then sends a signal to the inverter to produce an actual current to the servomotor.

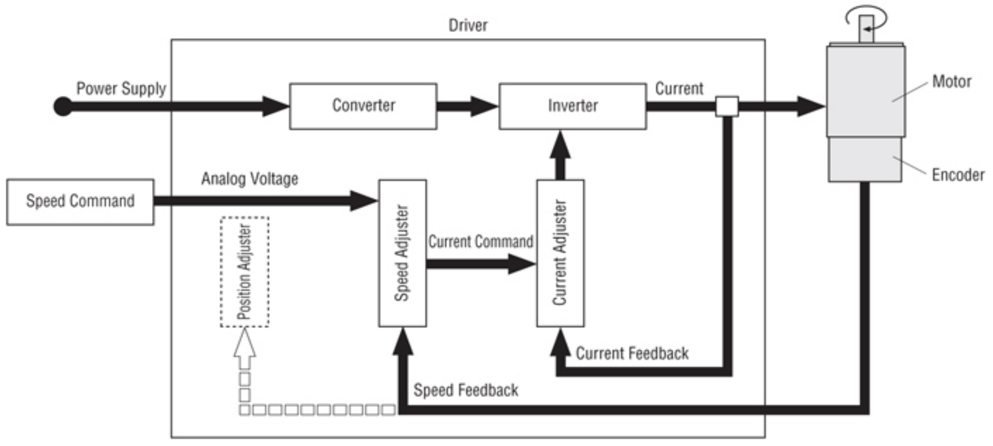


Figure 7.3.5 Analog voltage speed control block diagram

In torque control mode, the torque reference signal is produced from the microcontroller as well. Similar to the speed control, there is a torque control power unit that transfers the microcontroller signal to an analog voltage. The servo driver then takes this analog voltage as the torque reference, and this analog voltage directs a proportion to the torque reference. For the JMC servo driver, the rated torque is 1.27 Nm with a max input voltage of 10V. Therefore, the voltage to torque linear relation will be 1.27 Nm / 10 V. The servo driver takes the reference voltage, and then it produces the controlled current to the servomotor. Due to there being no torque sensor feedback, the motor torque is directly related to the input current. Figure 7.3.6 shows the structure of torque control done by an analog voltage.

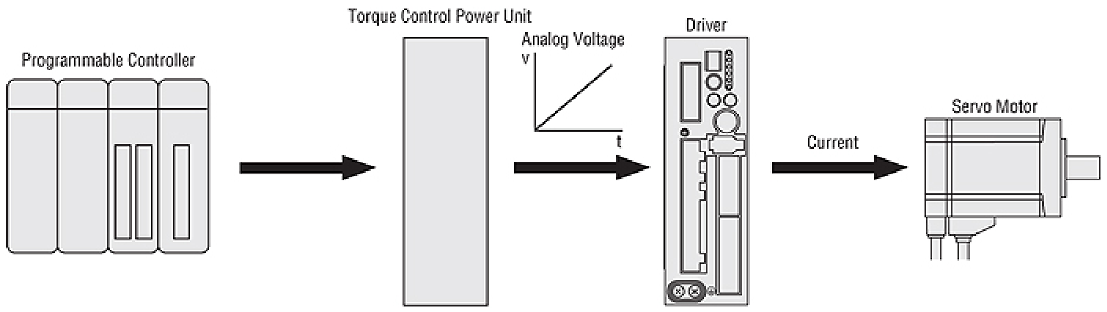


Figure 7.3.6 Torque control using analog voltage [56]

Same with speed control, the user manual did not provide the control block diagram for torque control; the general torque control block diagram is in Figure 7.3.7. Due to the servomotor not having the torque sensor, the motor torque is directly related to the supply the current. Therefore, the current is considered as an torque feedback. The current adjuster takes the reference analog voltage that is transferred to the related current. The current adjuster tries to minimize the error between the current reference command (i.e., torque command) to the current feedback (i.e. torque feedback). The control current signal is then sent to the inverter, and the physical current is sent to the motor.

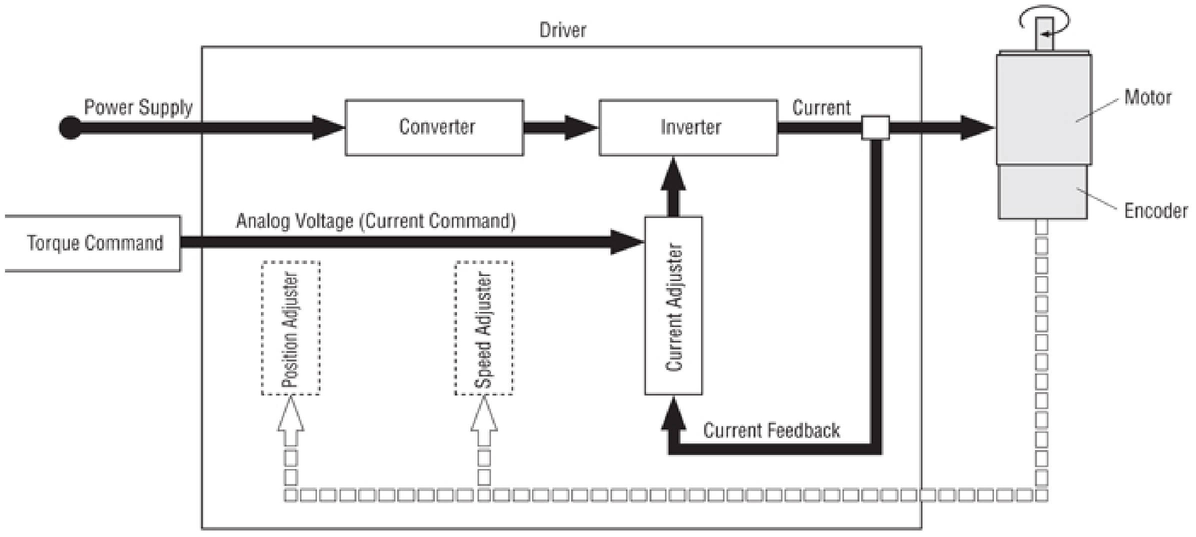


Figure 7.3.7 Analog voltage torque control block diagram

In this research, the built-in position control sets the benchmark. The research proposes that the control law uses the torque control. The reason for choosing analog voltage torque control is because in the torque control mode a full adjustment of the torque could be achieved. Also, the LuGre friction model is based on Newton’s law of motion by balancing the torque or force. In the torque control mode, the position reference and speed reference could be achieved by controlling the torque input to the system. A number of studies have also suggested that the torque control input is sufficient [14], [61], [19], [10], [62], [17].

7.4 BALL SCREW SYSTEM HARDWARE

The ball screw system used in this research was pre-assembled, and it was obtained from the sponsor for this project. The primary ball screw component can include the sliding table, ball screw assembly, base, linear guide, and motor bracket. Figure 7.4.1 presents the general assembly of the ball screw system.

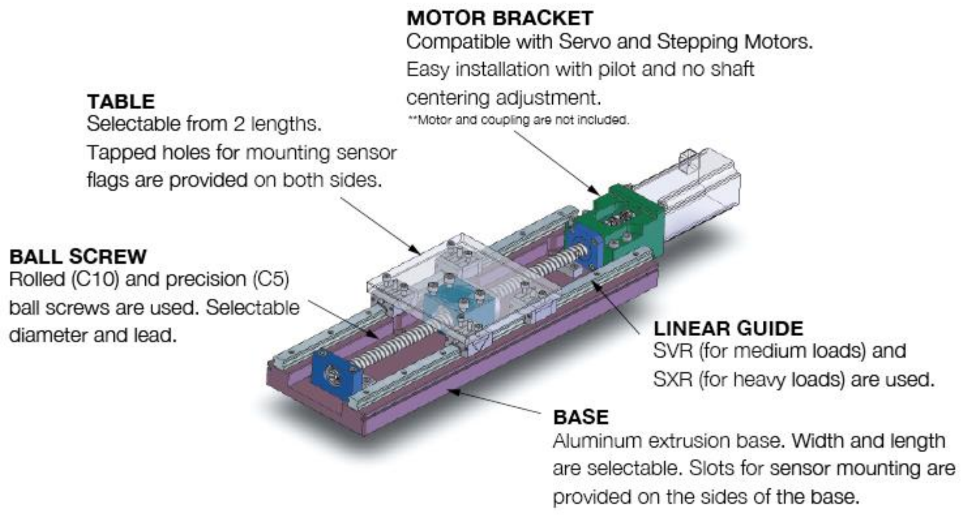

MOTOR BRACKET
Compatible with Servo and Stepping Motors. Easy installation with pilot and no shaft centering adjustment.
**Motor and coupling are not included.
TABLE
Selectable from 2 lengths. Tapped holes for mounting sensor flags are provided on both sides.
BALL SCREW
Rolled (C10) and precision (C5) ball screws are used. Selectable diameter and lead.
LINEAR GUIDE
SVR (for medium loads) and SXR (for heavy loads) are used.
BASE
Aluminum extrusion base. Width and length are selectable. Slots for sensor mounting are provided on the sides of the base.

Figure 7.4.1 General ball screw system mechanical setup [63]

The base is the main component that can be considered as the chassis of the entire assembly. The linear guide, the sensor slot, the ball screw bracket, and the motor bracket all sits on the base. At the two sides of the base, there is a linear guide which keeps the sliding table running in a straight line. In the middle of the base is the ball screw; the ball and the screw nut are under the sliding table, and they pass through the ball screw. At the end of the base, there is a motor bracket which can be mounted to a different type of the motor, such as a stepper motor or servomotor.

The ball screw system that has been provided in this research is shown in Figure 7.4.2. It has the same design as the one in Figure 7.4.1, which includes all the major components such as the sliding table, ball screw assembly, base, linear guide, and motor bracket. It also includes the kill switches in the assembly. In this case, the kill switches act as a safety feature. When the sliding table reaches a particular position, it will trigger the kill switch, and the kill switch will stop the motor from rotating. The total available running the length of the sliding table is 73.55 mm + -5 mm; this is measured from the right kill switch to the left kill switch.

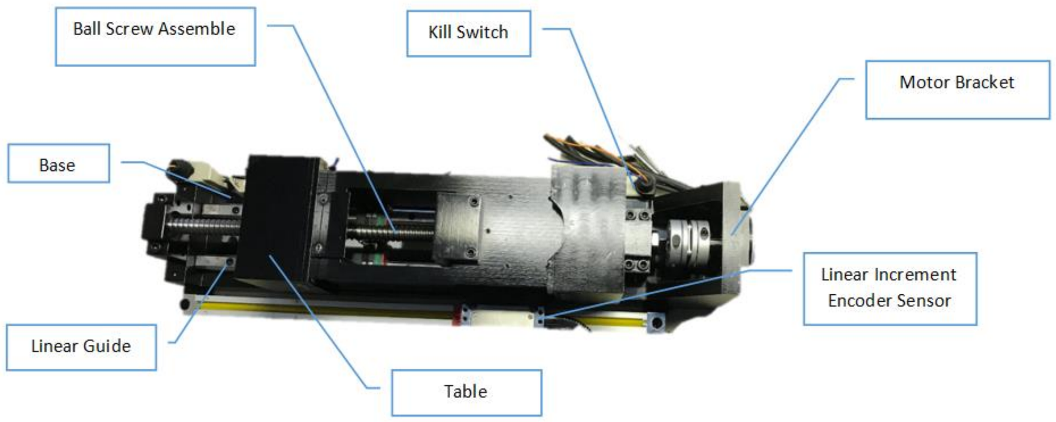


Figure 7.4.2 Ball screw system assembly

7.5 MICROCONTROLLER SELECTION PROCESS

The microcontroller—also called the microcontroller unit (MCU)—is an integrated circuit small computer that has been applied mainly to the embedded system. The MCU is an integrated circuit that contains the processor core, RAM, ROM and input/output peripherals (I/O pins) [64]. The MCU is the device that allows users to have direct control in the project or program if needed. The microcontroller could integrate anything that users need in the computing process and input-output data, and it does not require external circuits. The examples of the microcontrollers are the 8051, AVR, and the PIC series microcontroller chip.

The other computing device similar to the microcontroller is the microprocessor. The microprocessor has one process unit (CPU), and it has fewer integrated circuits than a microcontroller. [65] A microprocessor needs another external circuit to perform peripheral tasks, and it does not usually have any RAM, ROM, and other peripherals. The microprocessor is not only made for doing a specific task, but it also has a broader range of applications due to its higher processing speed and higher memory. It is suited for tasks that are required to have many inputs and outputs, and it also more suit for running the complex algorithm tasks. Some examples include the Intel I3, I5, and I7 CPU that are in the PC. As mentioned, permanence of a task requires additional components. In other words, a single I3 CPU cannot do anything; it needs to have a motherboard and another circuit to perform complex tasks. Figure 7.5.2 shows how the chips differ in the microprocessor and the microcontroller.

From an application viewpoint, the microprocessor is similar to a PC in that it normally has an operating system with few I/O ports, and it has a USB or Ethernet port. One of the examples of the microprocessor board is the Raspberry Pi, which can run a full Linux operating system. By contrast, the microcontroller has less computing power but due to its integration, it is more suitable for specific tasks. It normally has direct control of the I/O pins, such as timers, counters and interrupts. The microcontroller is a computer on the chip; it can run a single program that is sent from the PC and perform the dedicated tasks. One of the examples of the microcontroller board is Arduino UNO series. Figure 7.5.1 presents the structural differences between the microcontroller and the microprocessor. The microprocessor contains an arithmetic and logic unit (ALU), a control unit, and a register. The microcontroller contains CPU, RAM, ROM, I/P ports, counters, and timers which are in one chip. Due to the difference in structure, microcontrollers can handle real-time tasks, but microprocessors are not real-time systems, and they depend on other components on the board to get the job done. In this research, three of the microcontroller circuit boards (i.e., Arduino MEGA 2560, Arduino Duo, Teensy 3.5) and one microprocessor board (i.e., Raspberry Pi 3 Model B) are compared and discussed. Finally, one of them is chosen to use into ball screw control system.

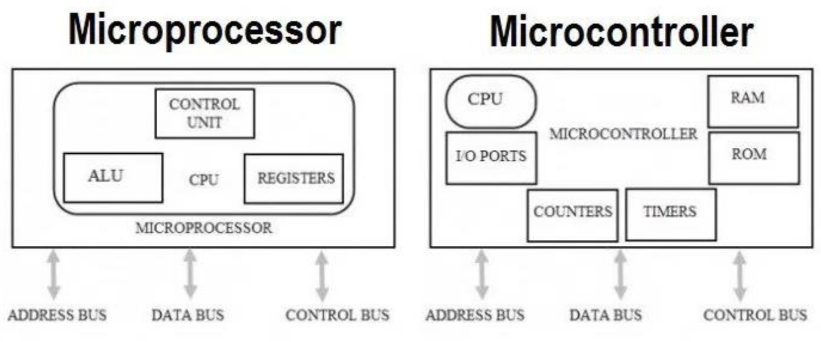


Figure 7.5.1 Comparison of a microcontroller with a microprocessor [66]

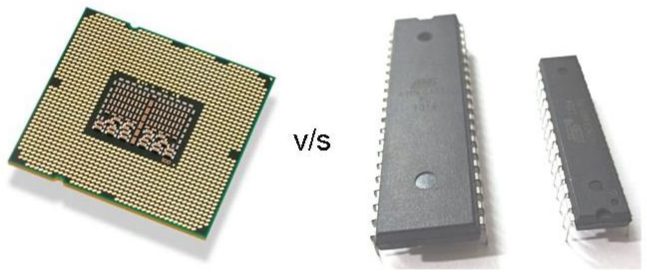


Figure 7.5.2 Chip differences between a microprocessor and a microcontroller [64]

Arduino Mega 2560 is a microcontroller (see Figure 7.5.3) that is based on the well-known microcontroller Arduino Uno. Both have same clock speed at 16 Mhz, and they both run on the ATmega microchip. However, Arduino Mega 2560 has more sketch memory and more RAM [67]. It has a total of 54 I/O pins—the Arduino Uno only has 14 I/O pins—and it has a 15 PWM[4] output

[4] PWM stand for pulse width modulation which creates an analog impulse voltage with digital means [59].

[68]. It also has a flash memory of 32 KB, whereas the Arduino Uno has only a 14 KB flash memory. Moreover, its operating voltage is 5 V, and it also supports serial ports (UART or USART) which are used for serial communication protocols such as SPI, I2C, and CAN bus.

Figure 7.5.3 Arduino Mega 2560 [68]

The Arduino Due is a microcontroller running on a 32-bit ARM core (i.e., Atmel SAM3X8E ARM Cortex-M3 CPU), as seen in Figure 7.5.4. It is the first Arduino board that runs on a 32-bit ARM structure. The clock speed is 84 Mhz, which is far higher than the Arduino Mega 2560's 16 Mhz. Its appearance is very similar to Arduino Mega 2560, and both have the same number of I/O pins (i.e., 54 pins). In terms of I/Os, the difference is that the Arduino Mega has 15 PMW outputs, and the Arduino Due has only 12 PWM outputs; also, the Arduino Due has only true 2 DAC ports, whereas the Arduino Mega 2560 only has the DAC software by using the PWM. The flash memory in the Arduino Due is 512 KB, which is one time higher than the Arduino Mega 2560. The Arduino Due is similar to the Mega 2560, which supports serial communication. In short, the Arduino Duo is more powerful than the Arduino Mega 2560 with more flash memory and much higher clock speed. The major drawback of the Arduino Due is that it runs at 3.3 V, unlike the other Arduino boards which all run at 5 V [69]. In this project, most of the devices run at 5 V, such as the encoder and the PMW position input port. However, this problem could solve using the external IC board, which could shift from 5 V to 3.3 V; the detailed solution related to this problem is explained in the encoder subsection.

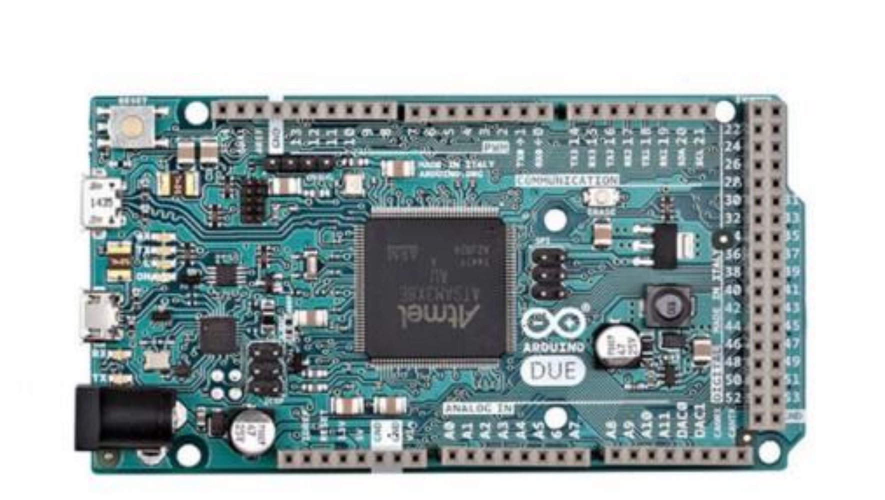


Figure 7.5.4 Arduino Due [69]

The third microcontroller discussed is the Teensy 3.5. Developed by the company PJRC, the Teensy microcontroller can be considered to be more an enhanced version of the Arduino Duo. Therefore, this microcontroller is compatible with the Arduino program IDE be done by applying the Teensyduino software add-on. Most of the sketch writing in the Arduino environment can run using the Teensy board as well. Also, the most of Arduino library can be applied to Teensy. From the hardware point of view, Teensy 3.5 offers a 120 MHz ARM Cortex-M4 with a floating-point unit CPU (MK64FX512VMD12) which is much faster than an Arduino Due [70]. Teensy 3.5 has the 32-bit ARM structure, which is also what the Arduino Due uses.

Figure 7.5.5 Teensy 3.5 [71]

Teensy 3.5 has a total of 62 I/O pins with 2 ADC (analog to digital) ports and 2 DAC ports. The USB serial could run at a full speed at 12 Mb/sec. A higher USB speed is very useful for collecting data at the PC terminal when the real time is sufficient. This feature is especially important in this project; the encoder sensor data is in real time, and a faster data output feature is needed in order not to lose any encoder reading. Teensy also could support 20 PWM outputs and has total 6 hardware serial ports that support I2C and SPI. From the I/O point of view, Teensy 3.5 offers more I/O pins than the Arduino Due and the Arduino Mega 2560. Teensy 3.5 also operates at a 3.3 V like the Arduino Due, but it has a 5 V tolerance on all digital I/O pins. If the user supplies more than 3.3 V on the Arduino Due the board could be burned out; however, this cannot happen on the Teensy 3.5. Teensy 3.5 also offers a micro SD card slot that allows the user to pull or push data to the SD card; this is useful when recording data offline is needed. The size of a Teensy 3.5 is much smaller than the Arduino Due and Arduino Mega 2560. The cost of a Teensy 3.5 (priced at $35.86 CAD at the time of writing) [72] is also less expensive compared to the Arduino Due ($51.08 CAD) [73] and Arduino Mega 2560 ($53.45 CAD) [74].

A comparison between all three microcontrollers has been summarized in Table 7.2. From the table, it can be seen that Teensy 3.5 dominates the group for microcontrollers in this study. The Teensy has a faster CPU with more memory, more RAM, and more I/O pins. It is also smaller and less expensive than the other two controllers. The main drawback for Teensy 3.5 is that it still runs on 3.3 V even though it has a 5 V tolerance. Arduino Mega 2560 has a 5 V operation voltage, but the CPU clock speed may not be sufficient for running a sophisticated control algorithm. In this project, most of the devices require a 5V output (see the section "Electrical and Mechanical System" for details on the devices). Therefore, there still needs to be an IC for performing a level shift between 5 V and 3.3 V. In conclusion, in the microcontroller area, Teensy 3.5 is more suitable for this project in terms of clock speed, I/O ability, and cost performance.

	Arduino Mega 2560	Arduino Due	Teensy 3.5
CPU	ATmega2560	AT91SAM3X8E	MK64FX512VMD12
Structure	AVR	ARM Cortex	ARM Cortex
Operation Voltage	5 V	3.3 V	3.3 V (5 V tolerance)
Digital I/O Pins	54 (15 PWM)	54 (12 PWM)	62 (20 PWM)
Analog Input Pins	16	12	25 (2 ADC)
Analog Output Pins	NA	2 DAC	2 DAC
Flash Memory	256 KB	512 KB	512 KB
SRAM	8 KB	96 KB	192 KB
Clock Speed	16 MHz	84 MHz	120 MHz
Length, Width	101.52 mm*53.3 mm	101.52 mm*53.3 mm	62.3 mm*18.0 mm
Cost	$53.448 CAD	$51.08 CAD	$35.86 CAD

Table 7.2 Comparison summary of microcontrollers

Aside from using microcontrollers as the main computing power, microprocessors are also under consideration. One of the more well-known microprocessors is the Raspberry Pi series developing board. The latest version of the Raspberry Pi series is the Raspberry Pi 3 computer board (see Figure 7.5.6). It has a Broadcom BCM2837 64-bit ARM Cortex-A53 Quad Core Processor running at 1.5 Ghz, which provide a lot more speed than the microcontroller. It also has 1 GB of RAM with 4 USB ports. It has an operating system (OS) much like a PC, and its runs on a modified version of Linux, namely the Raspbian. This mean that the users do not need to send their program from a PC terminal. By using its HDMI port and USB ports, the monitor, mouse, and keyboard could be directly used on the Raspberry Pi system. The storage space is for the OS and for other programs stored in the micro SD card. Like a PC laptop, it also has onboard WiFi and Bluetooth module. From I/O point of view, it offers 27 I/O pins that run on 3.3V with the support of I2C and SPI.

However, Raspberry Pi does not have direct control of the I/O ports; it has a lack of functions such as timers, counters and interrupts.

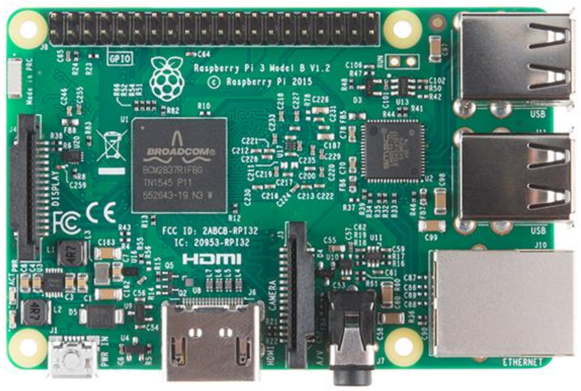


Figure 7.5.6 Raspberry Pi 3 [75]

In comparing the Raspberry Pi 3 with the Teensy 3.5, Raspberry Pi as a microprocessor offers more computer power, but it lacks I/O ability. In Linux, the user is not able to directly handle hardware interrupts; these should be handled by the kernels [76]. In this project, an interrupt or interrupt service routine (ISR) is critical for reading the encoder signal. In Teensy 3.5, the user could directly setup the ISR to read the encoder signal in real time. However, in the Raspberry Pi 3—which has its operating system—the entire system may be too bulky to handle the time-sensitive jobs. Raspberry Pi 3 is more suitable for the Internet-of-Things (IOT) type of jobs, where massive computing power and internet connectivity is required. For a specific job—such as the ball screw system dynamic control which depends on real-time hardware control—a microcontroller Teensy 3.5 is more suitable.

8 REFERENCES

[1] T. Igor Gilkin, "A Critical Look at Acme, Ball, and Roller Screws for Linear Motion," 06 May 2009. [Online]. Available: http://www.machinedesign.com/motion-control/critical-look-acme-ball-and-roller-screws-linear-motion.

[2] MultiCam.ca, "The Ball Screw Vs. Lead Screw," 4 April 2012. [Online]. Available: http://multicam.ca/the-ball-screw-vs-lead-screw/.

[3] M. v. d. Steenhoven, "6 advantages of the unique Exlar roller screw servo actuator," 27 Oct 2016. [Online]. Available: https://atbautomation.eu/en/blog/19-advantages-exlar-roller-screw-servo-actuator.html.

[4] B. I. e. department, "HOW A BALL SCREW WORKS," [Online]. Available: http://www.barnesballscrew.com/how-a-ball-screw-works/.

[5] nookindustries.com, "BALL SCREW TORQUE CALCULATIONS," [Online]. Available: http://www.nookindustries.com/LinearLibraryItem/Ballscrew_Torque_Calculations.

[6] J. DEGENOVA, "What You Ought To Know About Ball Screws," 22 July 1989. [Online]. Available: http://www.machinedesign.com/mechanical-drives/what-you-ought-know-about-ball-screws.

[7] J. Kes, "What is the drilling sound heard under an airplane before takeoff as well as at the gate upon landing?," 2 Sep 2014. [Online]. Available: https://www.quora.com/What-is-the-drilling-sound-heard-under-an-airplane-before-takeoff-as-well-as-at-the-gate-upon-landing.

[8] Zeiss, "ZEISS ACCURA," [Online]. Available: https://www.zeiss.ca/metrology/products/systems/bridge-type-cmms/accura.html.

[9] HAMADA, "A.C. Servomotor," 17 Jan 2013. [Online]. Available: http://www.yourelectrichome.com/2013/01/ac-servomotor.html.

[10] M. J. b. K. L. C.L. Chena, "Modeling and high-precision control of a ball-screw-driven stage," *Precision Engineering*, vol. 28, no. (2004) 483–495, p. 483, 2004.

[11] H. J. D. S. K. B. A. D. T. P. Dr. Korondi Péter, "8. Models of Friction," in *Robot Applications*, BME MOGI, 2014.

[12] R. C. Sergio Sánchez-Mazuca, "An Improvement Proposal to the Static Friction Model," *Mathematical Problems in Engineering*, vol. 2013, p. 8, 2013.

[13] B. Armstrong-Helouvry, Control of Machines With Friction, Kluwer Academic Publishers, 1991.

[14] N. K. I. Daiki Hoshino, "Friction Compensation Using Time Variant Disturbance Observer Based on the LuGre Model," in *The 12th IEEE International Workshop on Advanced Motion Control*, Sarajevo, Bosnia and Herzegovina, 2012.

[15] A. V. ,. C. B. ,. L. U. ,. G. P. Y. Altintas, "Machine tool feed drives," *CIRP Annals - Manufacturing Technology,* vol. 60, pp. 779-796, 2011.

[16] D. P. Li Chun Bo, The friction-speed relation and its influence on the critical velocity of stick-slip motion, Elsevier, 1982.

[17] T. Y. G. C. T. Jun'ya FUKUI, "Real-time Identification and Compensation of Asymmetric Friction Using Unscented Kalman Filter," in *IEEE Conference on Control Technology and Applications*, Hawai'i, 2017.

[18] ,. W. S. S. J. Paul I. Roa, "Robust friction compensation for submicrometer positioning and tracking for a ball-screw-driven slide system," *Precision Engineering,* vol. 24, pp. 160-173, 2000.

[19] Z. Q. X. L. Hongbiao Xiang, "Simulation and Experimental Research of Non-linear Friction Compensation for High-Precision Ball Screw Drive System," in *The Ninth International Conference on Electronic Measurement & Instruments*, 2009.

[20] M. A. G. a. C. A. B. V. I. Johannes, "The role of the rate of application of the tangential force in determining the static friction coefficient," *Wear,* vol. 24, no. 3, pp. 381-385, 1973.

[21] E. V. M. B. A. M. P. L. M. T. G. L. A. V. Luca Simoni, "On the use of a temperature based friction model for a virtual force sensor in industrial robot manipulators," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Limassol, Cyprus, 2017.

[22] Z. Wang, F. Guo, S. Liu, B. Baatar, Y. Wang and H. Liang, "Temperature characteristics of sliding friction pair under high-speed and strong-current conditions," in *2017 IEEE Holm Conference on Electrical Contacts*, Denver, CO, 2017.

[23] YasuhisaAndo, "The effect of relative humidity on friction and pull-off forces measured on submicron-size asperity arrays," *Wear,* vol. 238, no. 1, pp. 12-19, 2000,March.

[24] Z. Wenjing, "Parameter Identification of LuGre Friction Model in Servo System Based on Improved article Swarm Optimization Algorithm.," 2007.

[25] J. J.Craig, "Trajectory generation," in *Introduction to Robotics*, 2005, pp. 204-205.

[26] D. M. N. J.Ishikawa and S.Tei, "Friction Compensation Based on the LuGre Friction Model," in *SICE Annual Conference 2010*, Taiwan, 2010.

[27] C. X. Li W, "Adaptive high-precision control of positioning tables Theory and experiments," *IEEE Trans Contr Syst Technol,* vol. 2, no. 3, pp. 265-270, 1994.

[28] O. K., "A new servo method in mechatronics," *Trans Japan Soc Elect Eng,* vol. D, no. 83-86, p. 107, 1987.

[29] K. S. Kempf C, "Discrete-time disturbance observer design for systems with time delay," *Int Workshop Advan Motion Contr-AMC,* vol. 1, pp. 332-337, 1996.

[30] T. M. Fischer M, "Application and compensation of alternative position sensors in high-accuracy control of an X-Y table," *Int Workshop Workshop,* vol. 1, pp. 494-499, 1996.

[31] renishaw Inc, "RGH20 incremental encoder system with RSLR linear scale," [Online]. Available: http://www.renishaw.com/en/rgh20-incremental-encoder-system-with-rslr-linear-scale--6441. [Accessed 16 Jan 2018].

[32] Electronics Tutorials Inc, "Position Sensors," [Online]. Available: http://www.electronics-tutorials.ws/io/io_2.html. [Accessed 16 Jan 2018].

[33] National Instruments, "Encoder Measurements: How-To Guide," 3 Mar 2017. [Online]. Available: http://www.ni.com/tutorial/7109/en/. [Accessed 16 Jan 2018].

[34] FOIC , "Linear Scale and DRO system," [Online]. Available: http://www.chfoic.cn/proshow_72.html. [Accessed 16 Jan 2016].

[35] JIMB, "Bi-Directional Logic Level Converter Hookup Guide," [Online]. Available: https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide. [Accessed 16 Jan 2018].

[36] Erik, "Running encoder on Arduino Mega using Simulink," stackoverflow.com, [Online]. Available: https://stackoverflow.com/questions/26649706/running-encoder-on-arduino-mega-using-simulink. [Accessed 17 Jan 2018].

[37] arduino.cc, "attachInterrupt()," [Online]. Available: https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/. [Accessed 17 Jan 2018].

[38] automationdirect.com, "Motion Control Explained," [Online]. Available: https://library.automationdirect.com/motion-control-explained/. [Accessed 17 Jan 2018].

[39] alliedelec.com, "Siemens 6ES72141BG400XB0," Siemens, [Online]. Available: https://www.alliedelec.com/siemens-6es72141bg400xb0/70781107/. [Accessed 17 Jan 2018].

[40] circuitspecialists.com, "8-CH 16-Bit PCI Express Analog Output Card," circuitspecialists.com, [Online]. Available: https://www.circuitspecialists.com/pcie-6208v-glg.html. [Accessed 17 Jan 2018].

[41] vitalsystem.com, "4-Axis Motion Control & Data Acquisition PCI Board," vitalsystem.com, [Online]. Available: http://www.vitalsystem.com/web/motion/motionLite.php. [Accessed 17 Jan 2017].

[42] automationdirect.com, "Application Note AN-SERV-002," automationdirect.com, 2006.

[43] JUST MOTION CONTROL Inc, JASD series AC servo driver manual, Shenzhen: JUST MOTION CONTROL Inc.

[44] adafruit, "Adafruit_MCP4725," github, 24 Jul 2012. [Online]. Available: https://github.com/adafruit/Adafruit_MCP4725. [Accessed 17 Jan 2018].

[45] Electronics Tutorials Inc, "The Differential Amplifier," [Online]. Available: http://www.electronics-tutorials.ws/opamp/opamp_5.html. [Accessed 18 Jan 2018].

[46] Texas Insturments Inc, LMx24, LMx24x, LMx24xx, LM2902, LM2902x, LM2902xx, LM2902xxx Quadruple Operational Amplifiers, Dallas: Texas Instruments Incorporated, 2015.

[47] Electronics Tutorials Inc, "Operational Amplifier Building Blocks," [Online]. Available: http://www.electronics-tutorials.ws/opamp/op-amp-building-blocks.html. [Accessed 18 Jan 2018].

[48] C. C.-D.-W. KARL JOHAN ÅSTRÖM, "Revisiting the LuGre Friction Model," *IEEE CONTROL SYSTEMS MAGAZINE,* vol. 1066, no. 033, pp. 101-114, 2008.

[49] J. N. N. B. Ziegler, "Optimum Settings for Automatic Controllers," *Transactions of the ASME,* vol. 64, pp. 759-768, 1942.

[50] D. Bone, "4.2 Discrete PID control and Zeigler-Nichols Tunning Method," in *MECH ENG 751: Advanced Mechanical Egineering Control System,* Hamilton, Dr.Gary Bone, 2016, p. 33.

[51] Matlab, "How the Optimization Algorithm Formulates Minimization Problems," Matlab 2017a help manual , 2017.

[52] Matlab , "How the Software Formulates Parameter Estimation as an Optimization Problem," Matlab, 2017.

[53] pyroelectro.com, "pyroelectro.com," pyroelectro.com, [Online]. Available: http://www.pyroelectro.com/tutorials/fading_led_pwm/theory.html. [Accessed 31 Jan 2018].

[54] megunolink.com, "EXPONENTIAL FILTER," [Online]. Available: https://www.megunolink.com/documentation/arduino-libraries/exponential-filter/. [Accessed 1 Jan 2018].

[55] JonHub, "A realtime digital signal processing (DSP) library for Arduino.," [Online]. Available: https://playground.arduino.cc/Code/Filters. [Accessed 1 Feb 2018].

[56] OrientalMotor, "Servo Motor Features Overview," [Online]. Available: http://www.orientalmotor.com/servo-motors/technology/servo-motor-features.html.

[57] JMC motion, "ASD 400W AC servo drive JASD 4002-20B," [Online]. Available: http://www.jmc-motor.com/products.php?cid=251&id=194.

[58] JUST MOTION CONTROL, JASD series AC servo driver selection manual, JUST MOTION CONTROL.

[59] arduino.cc, "PWM," [Online]. Available: https://www.arduino.cc/en/Tutorial/PWM.

[60] JUST MOITION CONTROL Inc, "Manual gain tuning," in *JASD series AC servo driver user manual* , Shenzhen, JUST MOITION CONTROL Inc, pp. 49-51.

[61] R. E. F. Vargas, "Identification and Friction Compensation for an Industrial Robot Using Two Degrees of Freedom Controllers," in *Control Automation Robotics and Vision*, 2004.

[62] F. S. Jinkun Liu, "Fuzzy Global Sliding Mode Control for a Servo System with Lugre Friction Model," in *Proceedings of the 6th World Congress on Intelligent Control*, Dailan, 2006.

[63] C. Straetz, "Designing with Single Axis and Multi-Axis Actuators," 29 July 2014. [Online]. Available: http://blog.misumiusa.com/choosing-the-right-linear-actuator/.

[64] Apoorve, "What is the difference between microprocessor and microcontroller?," [Online]. Available: https://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller.

[65] "Difference Between Microprocessor and Microcontroller," 29 May 2015. [Online]. Available: https://www.electronicshub.org/difference-between-microprocessor-and-microcontroller/.

[66] EEE PROJECTS, "Difference between Microprocessor and Microcontroller," [Online]. Available: https://eeeproject.com/microprocessor-vs-microcontroller/.

[67] arduino.cc, "Getting Started with Arduino and Genuino MEGA2560," arduino.cc, [Online]. Available: https://www.arduino.cc/en/Guide/ArduinoMega2560.

[68] arduino.cc, "ARDUINO MEGA 2560 REV3," [Online]. Available: https://store.arduino.cc/usa/arduino-mega-2560-rev3.

[69] arduino.cc, "ARDUINO DUE," arduino.cc, [Online]. Available: https://store.arduino.cc/usa/arduino-due.

[70] sparkfun.com, "Teensy 3.5," sparkfun.com, [Online]. Available: https://www.sparkfun.com/products/14055#description-tab.

[71] pjrc.com, "Teensy USB Development Board Version 3.5," pjrc.com, [Online]. Available: https://www.pjrc.com/store/teensy35.html.

[72] DigiKey.ca, "DEV-14055," DigiKey.ca, [Online]. Available: https://www.digikey.ca/products/en?keywords=DEV-14055.

[73] DigiKey.ca, "Arduino A000062," DigiKey.ca, [Online]. Available: https://www.digikey.ca/products/en/development-boards-kits-programmers/evaluation-boards-embedded-mcu-dsp/786?k=Arduino%20Due.

[74] DigiKey.ca, "Arduino GBX00067," DigiKey.ca, [Online]. Available: https://www.digikey.ca/product-detail/en/arduino/GBX00067/1660-1001-ND/5980469.

[75] sparkfun.com, "Raspberry Pi 3 DEV-13825," sparkfun.com, [Online]. Available: https://www.sparkfun.com/products/13825.

[76] gordon@drogon.net, "Interrupt handling," raspberrypi.org, [Online]. Available: https://www.raspberrypi.org/forums/viewtopic.php?t=9207.

[77] R. C. Sergio Sánchez-Mazuca, "An Improvement Proposal to the Static Friction Model," *Mathematical Problems in Engineering,* vol. 2013, no. Article ID 946526, 2013.