PREDICTION-BASED HAPTIC INTERFACES TO IMPROVE TRANSPARENCY FOR COMPLEX VIRTUAL ENVIRONMENTS

by

Shane Forbrigger

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
June 2017

Table of Contents

List of Tables

List of Figures

Abstract

Virtual surgical training simulators are an exciting opportunity to improve surgical training, especially in challenging fields like minimally invasive surgery. Virtual surgical training requires highly accurate haptic feedback to provide effective training, however, accurate human tissue models are computationally intensive and update too slowly. This work investigates a method of improving the haptic feedback realism (or transparency) from slowly updating virtual environments by designing a control structure that takes advantage of higher update rates outside of the virtual environment. The current state of the art in surgical training and controls research is identified through a review of the relevant literature. The contributions of this work are as follows. A predictor is designed for an unknown linear-time invariant system using Lyapunov-based methods to provide an estimate of the ideal virtual environment output at a higher output rate. The predictor design is extended to a gain-scheduled predictor using linear parameter-varying systems analysis. The resulting haptic system is tested both in simulation and experiment. For linear-time invariant systems the predictor provides excellent performance, leading to experimental improvements in transparency of up to 40%. For nonlinear systems the predictor provides mixed results, ranging from negligible results to improvements of approximately 20%.

List of Abbreviations and Symbols Used

B	Haptic device viscous friction torque matrix.
b_e	Virtual environment damping or viscous friction.
b_t	Virtual tool damping or viscous friction.
C	Haptic device Coriolis and centrifugal torque matrix.
c_e	Virtual environment force offset.
Δ_T	Disturbance due to sampling.
Δ_θ	Disturbance due to parameter estimation error.
DOF	Degree(s) of freedom.
$\boldsymbol{e}$	Prediction error, i.e. the difference between the predicted virtual tool trajectory and the ideal virtual tool trajectory.
$\boldsymbol{e}_\Delta$	Sampling and delay error, i.e. the difference between the sampled and delayed virtual tool trajectory and the ideal virtual tool trajectory.
$\boldsymbol{e}_{tot}$	Total error, i.e. the difference between the actual end-effector trajectory and the ideal virtual tool trajectory.
$\boldsymbol{F}_e$	Virtual environment force.
$\hat{\boldsymbol{F}}_e$	Estimated virtual environment force.
$\boldsymbol{F}_h$	Operator input force.
γ	Disturbance attenuation factor.
Γ	Haptic controller adaptation gain matrix.
Γ_e	Predictor adaptation gain matrix.
H	Haptic device inertia matrix.
J	Jacobian matrix relating the end effector velocity to the joint angular velocities.
K_d	Haptic controller control gain matrix.
k_e	Virtual environment stiffness.
k_t	Virtual tool stiffness.
L	Predictor feedback gain.

Λ	Haptic controller reference signal gain matrix.
LMI	Linear matrix inequality.
LPV	Linear parameter-varying.
LTI	Linear time-invariant.
MIS	Minimally invasive surgery.
m_t	Virtual tool mass.
Ω_{θ_e}	Set of all possible virtual environment parameter values.
PC	Personal computer.
$\boldsymbol{\varphi}_e$	Parameterized virtual environment state matrix.
$\hat{\boldsymbol{\varphi}}_e$	Parameterized estimated virtual environment state matrix.
PID	Proportional-integral-derivative control.
$\boldsymbol{q}$	Joint variable vector, specifically the joint angle vector.
$\Re$	The set of real numbers. $\Re^{n \times m}$ signifies the set of real matrices with n rows and m columns.
$\boldsymbol{\rho}$	Scheduling variable vector.
$\boldsymbol{\tau}_c$	Haptic device control torque vector.
$\boldsymbol{\tau}_g$	Haptic device gravitational torque vector.
T_d	Computation time or delay, i.e. the time between samples.
$\boldsymbol{\theta}$	Haptic device dynamic parameter vector.
$\boldsymbol{\theta}_e$	Virtual environment parameter vector.
$\hat{\boldsymbol{\theta}}_e$	Estimated virtual environment parameter vector.
$\hat{\boldsymbol{\theta}}$	Estimated haptic device dynamic parameter vector.
VE	Virtual environment.
$\boldsymbol{x}$	State vector, specifically the position of the end-effector.
$\boldsymbol{x}_e$	Position of the virtual tool.
$\hat{\boldsymbol{x}}_e$	Predicted position of the virtual tool.

Acknowledgements

I would like to thank my supervisor, Dr. Ya-Jun Pan, for all of her support and guidance. I would also like to thank all of the members of my Supervisory Committee for their feedback and advice. Finally, I would like to thank my family for all of their love and support.

Many thanks to NSERC and the Nova Scotian Government for helping to fund this research.

Chapter 1

Introduction

Haptic interfaces are devices that produce force or touch sensations, creating the illusion that the operator is physically interacting with an object or environment. They may be as simple as the vibration feedback from a smartphone keyboard or as complex as a glove that allows the operator to touch a remote or virtual object [1]. Haptic interfaces are being used in an increasing number of applications, including entertainment, remote control of robots (or teleoperation), and training simulators for aerospace and surgery [2]. Training simulators are an area of particular interest, as haptic interfaces open the possibility of virtual reality that not only looks real, but *feels* real.

Virtual surgical training simulators allow surgeons to learn and practice a variety of skills, but they need to be very realistic to provide effective training. One major limitation to creating realistic virtual training simulators is the computational difficulty of simulating a realistic virtual environment. Virtual environments in surgical training simulators can be quite complex, involving multiple bodies that must be able to deform and be cut. Such complex models often use some form of finite element model in order to simulate their properties realistically [3]. Expensive, specialized hardware is needed to simulate these models at high update rates. Haptic feedback requires high update rates in order to provide realistic feedback. Low update rates can result in noticeable delay in the feedback, undesirable vibrations or 'buzzing', and/or instability. Is there a way to provide higher-quality feedback without having to purchase expensive computer hardware?

1.1 Problem Definition

Consider the scenario shown in Fig. 1.1. The operator and haptic device operate in continuous time, and ideally the operator would perceive a continuous-time virtual environment through the haptic device. A controller, running at a high enough update

rate that it is approximately continuous, mediates the interaction between the haptic device and the virtual environment running at a slow update rate. The key concept is that the controller and virtual environment operate at different update rates. There are many situations where this scenario could exist. For example, a single Personal computer (PC) runs the controller and virtual environment on separate processors, or a dedicated control board runs the control software while a PC simulates the virtual environment. Given the rise in popularity of cloud-based computing, another possibility would be a local PC running the control software while a server simulates the virtual environment.

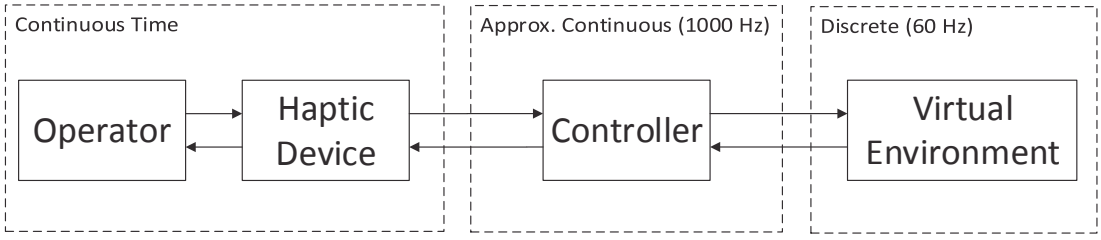


Figure 1.1: Main concept

The objective is to design the controller to take advantage of the increased update rate to provide higher-quality haptic feedback from the slowly updating environment.

1.2 Literature Review

This section presents an overview of the literature relevant to this work.

1.2.1 Virtual Surgical Training

Minimally invasive surgery (MIS) involves the insertion of tools and a video camera through small incisions in a patient's body, allowing the surgeon to access and view the surgery from outside. The tools are placed at the end of long slender rods and actuated via grips and levers outside of the patient's body. The ability of the surgeon to perceive forces from tissue is greatly diminished compared to open surgery, therefore MIS is very challenging and requires extensive training [4]. This training has been performed using cadavers, mannequins and physical simulators, but in the past couple decades virtual simulators have become increasingly common [5].

Virtual training simulators allow the trainee to practice procedures on a virtual patient. The simulator may provide visual feedback to the operator in the form of a television screen or virtual reality goggles, and haptic feedback in the form of mock surgical tools connected to a robotic actuator [4]. Virtual surgical training simulators provide the advantage of allowing surgeons to train without the potential ethical implications of practising on humans or animals [5]. They are also capable of providing useful feedback to the trainee and instructor. For example, the forces a trainee applies during surgery are indicative of their skill level, with more skilled surgeons applying smaller, more accurate forces [6]. A virtual surgical training simulator can easily record a trainee's motions, forces, and actions to provide a detailed report on their performance and help the instructor determine areas for improvement. Virtual surgical training simulators also provide novel approaches to training, such as a "shared-control" approach where an instructor and trainee operate simultaneously and the instructor can help guide the trainee's actions [7]. Virtual training simulators can also easily present a variety of scenarios, from standard procedures to complex problem solving or high intensity situations, such as a mistakenly severed artery. They also allow trainees to be exposed to variations in patient anatomy, which is difficult for traditional methods to do [8].

Although there are many potential benefits to using virtual surgical training simulators, there are still concerns about the effectiveness of this type of training. The primary concern in the medical community is the ability of the virtual simulators to accurately reproduce the surgical environment [9]. There is a great need for highly realistic virtual training simulators, as well as detailed studies regarding their effectiveness at transferring useful skills to trainees. Although there have been many small-scale studies of the effectiveness of virtual surgical training simulators that suggest they are effective, there have not been any large-scale randomized controlled trials that confirm that this type of training results in improved performance in the operating room [10].

1.2.2 Motivation

Virtual surgical training simulators require accurate haptic feedback to provide effective training [11], particularly for forces perpendicular to the axis of the virtual

tool [12]. The ability to simulate realistic virtual tissue models (or virtual environments) that are capable of deformation and cutting is crucial to providing accurate feedback. Realistic deformable and cut-able tissues are difficult to model explicitly, therefore many models are based on finite element methods [3]. These models are computationally-intensive and require expensive, high-end computer hardware such as a dedicated graphics processing unit [13, 14]. Even using high-end computer hardware can result in slow update rates, especially for virtual environments with a large number of deformable bodies. A simulation involving a single deformable but not cut-able body interacting with several rigid bodies could run at 1700 Hz on specialized hardware [14]. For lower-end hardware and more complex virtual environments the update rate could drop below the ideal update rate of 1000 Hz for haptic feedback.

1.2.3 Control of Haptic Interfaces

There are two primary concerns when considering the control of haptic interfaces: stability and transparency. A haptic system is stable if its error is bounded, and it is transparent if its feedback forces are accurate. The stability of a haptic interface interacting with a linear time-invariant (LTI) virtual environment has been investigated extensively in literature. Colgate [15] showed that the stability of a discrete-time LTI virtual environment interacting with an LTI haptic device depends on the sampling rate and the damping of the haptic device. Increasing the stiffness or damping, or decreasing the sampling rate, leads to a less stable system. This stability criterion was extended to consider the effects of quantization and Coulomb friction in the haptic device dynamics [16]. The stability criterion guarantees a stable system, but it does not consider the transparency of the system.

The virtual coupler method is one approach to design a haptic interface that is both stable and transparent. A virtual coupler introduces artificial dynamics between the haptic display and the virtual environment, allowing the haptic interface to represent a wider range of environment parameters and sampling rates [17]. The design of the virtual coupler can incorporate a transparency criterion to maximize the transparency over the stable range of the haptic interface [18]. The concept of virtual couplers was extended to maintain stability and transparency for virtual environments with multiple materials of different stiffness [19]. A major limitation of virtual couplers is

that their design process is only valid for LTI virtual environments and haptic device dynamics. Many haptic devices have nonlinear dynamics, and many tissue models are nonlinear as well [20, 21] They also do not consider the effect of sampling on the transparency of the system.

Another method of ensuring the stability of a haptic interface while maintaining transparency is to use a time-domain passivity control approach. The passivity control approach ignores the dynamics of the systems being controlled and instead considers the energy content of the input and output ports of the system. If the energy in the system is increasing in an unstable fashion an artificial damping is applied to dissipate the energy. Transparency is maintained by only applying the damping when it is necessary to ensure stability [22]. The advantage of this method is that it is independent of the system dynamics. The disadvantage is that it may be overly conservative by not taking advantage of inherent dissipation in the system.

Another approach to designing haptic interfaces is to treat the haptic system as a tracking problem using a coupling controller. The virtual environment is treated as the desired system dynamics and generates a trajectory according to the operator input force. The coupling controller tracks the output trajectory of the virtual environment, resulting in the desired dynamics being displayed to the operator [23]. The advantage to this approach is that the design of the virtual environment and the coupling controller are separated: as long as the virtual environment outputs a stable trajectory, and the coupling controller can stably follow that stable trajectory, then the whole system is stable. The disadvantage to this approach is that it requires accurate operator force measurements both to generate the correct virtual environment trajectory and to allow the coupling controller to track that trajectory.

The controller selection for the haptic device depends on the device dynamics. For a linear device methods such as PID or pole-placement may be appropriate. However, most haptic devices have nonlinear dynamics. In that case a passivity-based controller may be used to ensure stability and desired reference tracking. For haptic devices with unknown parameters an adaptive control law may be added to improve tracking [24].

1.2.4 Control of Time-Delay Systems

The control of time-delay systems is a problem that has been covered extensively for networked control systems and robotic teleoperation. The stability of LTI time-delay systems may be analysed using Lyapunov-based methods. Lyapunov-based analysis of time-delay systems leads to a large, non-convex matrix inequality. The non-convex matrix inequality can be linearized to form a linear matrix inequality (LMI) which can be solved numerically [25]. Lyapunov-based methods can also be applied to linear parameter-varying (LPV) systems, however this results in an infinite number of potentially feasible solutions to the LMI. Therefore the LMI is relaxed by imposing a structure on the Lyapunov function's parameter dependence, or for a less conservative but more complex solution, expressing the LMI in terms of B-splines [26, 27].

Providing transparent haptic feedback in a delayed system is challenging. One method of improving haptic feedback in a time-delay system is through the use of a predictor. A predictor is similar to an observer or a Kalman filter, except that it predicts a future value of a delayed signal rather than just providing an estimation of an unmeasurable state. The error between the predicted signal and the delayed signal can be used to improve the prediction. Predictors are used in teleoperation to improve performance [28].

There are two primary prediction methods: signal-based prediction and model-based prediction. Signal-based prediction uses past outputs, rather than a model of the system, to estimate future outputs of the system. Signal-based predictors use extrapolation and interpolation [29], autoregressive models [30], or Smith predictors [31] to generate predictions. Model-based prediction uses the known dynamic model of the system combined with a measurement of the system input to predict future outputs. Model-based predictors are used in LTI systems to reduce the effect of delay in the control input [32, 33], and to handle long sampling delay in feedback signals [34]. There are also "hybrid predictors" which generate predictions using several signal-based and model-based predictors simultaneously, and switch between predictors to minimize the error [35]. The advantage of signal-based prediction is that it can be used to predict the output of any system. The advantage of model-based prediction is that, if the model is accurate, the output of the predictor is more accurate. Model-based predictors have not been used to predict the output of LPV or nonlinear systems.

Accurate models are necessary for model-based prediction. For systems where the model parameters are unknown a parameter adaptation law can be used to estimate the parameters online. For LTI systems a least-squares adaptation law can be used to estimate the parameters [36]. For systems with bounded, slowly-varying, unknown parameters a projection-type least-squares adaptation law can be used to provide an estimate of the parameters [37].

1.3 Thesis Contributions

This work contains the following contributions:

1. A model-based predictor is applied to improve the transparency of a slowly updating virtual environment with unknown nonlinear dynamics. Both a constant-gain and gain-scheduled (LPV) predictor are designed using Lyapunov-based methods.

2. The stability problem of the predictor interacting with the virtual environment is formulated as a linear matrix inequality using Lyapunov-based methods.

3. The resulting haptic system is simulated and its performance with and without the predictor is quantified and analysed.

4. The resulting haptic system is tested experimentally and its performance with and without the predictor is quantified and analysed.

Other works have either assumed the virtual environment dynamics are known and LTI, or have used signal-based prediction strategies. The model-based predictor approach has been applied to teleoperation, but the virtual environment case presents a different set of challenges. For example in teleoperation the predicted signal and delayed signals are usually updated at close to the same rate. Unlike in the teleoperation case the predictor in the virtual environment case can operate at a much higher sampling rate than the dynamics that are being predicted.

1.4 Thesis Organization

The rest of this work is organized as follows. Chapter 2 describes the robotics and controls terminology and theory used in this work. Chapter 3 gives a detailed overview of the problem scope and describes the mathematical models used in this work. Chapter 4 gives a detailed description of the control design for both the haptic device controller and the predictor for increasingly complex cases. Chapter 5 describes the control hardware and software used to perform the simulations and experimental work. It also describes the calibration of the control hardware. Chapter 6 describes the results for simulations of the haptic system and discusses their implications. Chapter 7 describes the experimental performance of the haptic system, and discusses the advantages and limitations of using the predictor. Chapter 8 summarizes the conclusions of this work and suggests areas for future research.

Chapter 2

Background Theory

This section describes the terminology and theoretical concepts used in this work. The main focus is to explain the essential concepts, while specific derivations are introduced in Chapter 3.

2.1 Robotics Theory

This section explains the essential information needed to understand the derivations of kinematic and dynamic models used in this work.

2.1.1 Robotics Terminology

A manipulator is a type of robot commonly referred to as a 'robot arm.' It generally consists of a set of rigid bodies, called 'links,' joined together by a set of connections, called 'joints.' The part of the robot connected to the ground is called the 'base', and the free end is called the 'end-effector.' The end effector may hold a gripper for lifting objects, a tool for performing some work, or a handle for interacting with an operator [38]. Fig. 2.1 shows an example schematic of a manipulator with each part labelled.

The joints constrain the motion of the links. Revolute joints allow the links to rotate about the center of the joint. The state of the joints is described by the 'joint variable' vector, denoted $\boldsymbol{q}$. The joint variable for a revolute joint is its joint angle.

A serial chain manipulator is a manipulator that does not contain any parallel connections of links, i.e. there is only one path along the links from the base to the end-effector.

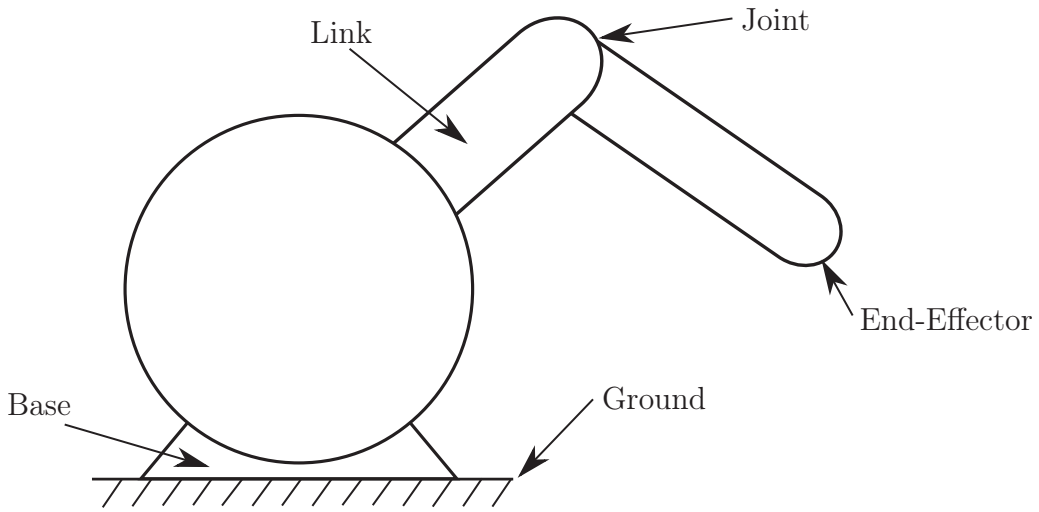


Figure 2.1: A schematic of a robot manipulator illustrating the robotics terminology.

2.1.2 Robot Kinematics

Forward Kinematics

The forward kinematic equations express the pose of a point on the manipulator, normally the end-effector, as a function of the joint variables of the manipulator. The forward kinematics can be systematically derived by expressing the manipulator as a series of homogeneous transformations using the Denavit-Hartenberg convention. For a manipulator with revolute joints, the resulting equations are nonlinear [39].

Inverse Kinematics

The inverse kinematic equations express the joint variables as a function of the pose of the end-effector. Unlike the forward kinematics, the inverse kinematics cannot be derived through a specific algorithm. This is because there may be no solutions, multiple solutions, or even infinite solutions to the inverse kinematics problem. For simple systems with a small number of degrees of freedom it is often possible to derive a closed form solution using geometry by applying additional constraints to remove multiple solutions. More complex systems may use numerical methods to solve the inverse kinematics [39].

Differential Kinematics

The differential kinematic equations relate the velocity and acceleration of the end-effector to the rate of change of the joint variables. The forward differential kinematics express the velocity of the end-effector as a linear matrix function of the joint rates of change. Therefore the inverse differential kinematics are easily derived from the forward kinematics [39].

2.1.3 Robot Dynamics

The dynamics relate the acceleration of the robot to the applied forces and torques. The dynamics of manipulators are normally expressed in joint space, i.e. relating the joint angular acceleration to the motor torque and other applied loads. The dynamics of a robot may be derived systematically by deriving the total energy of the robotic system and applying Lagrange's Equation [40].

2.2 Control Theory

This section explains the essential control-theory concepts.

2.2.1 Lyapunov-Based Methods

Lyapunov-based methods are used to develop sufficient (but not necessary) conditions for the stability of a system. This means that Lyapunov-based methods are inherently conservative: a system may be stable even if it does not meet the criteria specified by the Lyapunov-based methods. Lyapunov-based methods are based on Lyapunov's Stability Theorem, which is paraphrased as follows. Define a continuously differentiable function $V(x)$ known as a 'Lyapunov Function.' If it can be demonstrated that $V(0) = 0$, $V(x) > 0 \quad \forall x \neq 0$, and $\dot{V}(x) \leq 0$ (where $\dot{V}(x)$ is the time derivative of $V(x)$), then $x = 0$ is a stable equilibrium point, i.e. x is bounded. If it can additionally be shown that $\dot{V}(x) < 0 \quad \forall x \neq 0$, then $x = 0$ is an asymptotically stable equilibrium point, i.e. $\lim_{t\to\infty} x = 0$ [41].

The idea behind Lyapunov-based methods is to define a Lyapunov function for the error which proves that the error converges towards zero. If a function cannot be

found to prove that the error converges towards zero, at the very least it can be shown that the error is bounded and that the bound is sufficiently small.
For example, the simplest case would be an LTI system described by

$$\dot{\boldsymbol{x}} = A\boldsymbol{x},$$

where $\boldsymbol{x} \in \Re^{n\times 1}$ is the state vector and $A \in \Re^{n\times n}$ is a matrix. The Lyapunov function may be defined as

$$V(x) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x},$$

which satisfies $V(0) = 0$ and $V(x) > 0 \quad \forall x \neq 0$. Taking the time derivative results in

$$\begin{aligned}
\dot{V}(x) &= \frac{1}{2}\left(\dot{\boldsymbol{x}}^T\boldsymbol{x} + \boldsymbol{x}^T\dot{\boldsymbol{x}}\right), \\
\dot{V}(x) &= \frac{1}{2}\left(\boldsymbol{x}^T A^T\boldsymbol{x} + \boldsymbol{x}^T A\boldsymbol{x}\right), \\
\dot{V}(x) &= \boldsymbol{x}^T A\boldsymbol{x}.
\end{aligned}$$

For the system to be asymptotically stable $\dot{V}(x) < 0 \quad \forall x \neq 0$, therefore the matrix A must be negative definite. Not coincidentally, this is the same stability condition that one would arrive at from linear systems theory.

Barbalat's Lemma

For nonautonomous systems, i.e. in cases where the Lyapunov function is not only dependent on the error, but also on time, Lyapunov's stability theorem is insufficient to prove stability. Another property, known as Barbalat's Lemma, is needed.

Lemma 1 (Barbalat's Lemma)**.** [41] *Let ϕ be a uniformly continuous function on $[0,\infty)$, whose inputs and outputs are real numbers. Suppose that $\lim_{t\to\infty}\int_0^t \phi(\tau)d\tau$ exists and is finite. Then,*

$$\phi(t) \to 0 \text{ as } t \to \infty.$$

This is necessary to prove that not only does a stable equilibrium point exist, but that the system approaches that equilibrium over time.

2.2.2 Linear Matrix Inequalities

For systems more complicated than the example given in Section 2.2.1, verifying Lyapunov stability may require finding values for variables that satisfy a large set of simultaneous equations, rather than simply checking that a single matrix is negative definite. The variables that are being solved for are called the decision variables. In those cases it is convenient to express the set of equations as a single symmetric matrix that must meet a criterion such as negative definiteness. If the matrix equations can be made linear in terms of the decision variables the whole system can be expressed as a linear matrix inequality (LMI). The LMI may be solved numerically as a convex optimization problem. Fortunately MATLAB has LMI solvers that can be used to solve these problems [42].

The Schur Complement

The Schur Complement formula is often used to transform a nonconvex matrix inequality into a convex matrix inequality. The Schur Complement formula states that the following matrix inequality,

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} < 0, \tag{2.1}$$

is true if and only if $C < 0$ and $A - BC^{-1}B^T < 0$. This is useful because if B is a matrix of decision variables, $A - BC^{-1}B^T < 0$ is nonconvex, but (2.1) is convex [42].

Chapter 3

Problem Formulation

3.1 Problem Scope

The problem of accurate haptic feedback for virtual surgical training simulators is quite broad. A surgeon performs many different tasks during an operation that each provide their own challenges. In order to constrain the scope to a more manageable scale, this work will only consider the stability and transparency of a haptic system for a palpation task during a MIS procedure.

There are many advantages to constraining the problem scope to a MIS palpation task. First, high-quality force feedback is essential to a palpation task because the surgeon is investigating subtle changes in the tissue properties to identify a region of interest. Second, all interactions between the surgeon and the patient are performed through the surgical tools. Third, palpation does not involve any cutting or permanent deformations of the tissue, therefore the environment can be simulated using a static model. Fourth, the applied force, velocity, and acceleration involved with a palpation task are low compared to free motion or other tasks.

The virtual environment will be treated as an unknown nonlinear function that accepts an operator force input and outputs the virtual environment force and the virtual tool position and velocity.

The scope of the problem may be expressed by the following set of constraints:

1. The geometry of the virtual environment model is static. The position and orientation of the virtual environment surface does not change, nor does it permanently deform.

2. Contact between the virtual surgical tool and environment is uninterrupted. Therefore the energy effects of the initial contact/collision with the environment will not be considered.

3. The virtual interaction forces and virtual tool position, velocity and acceleration are all bounded and the bounds are known. The bounds may be introduced by hardware limitations, or artificially prescribed in software.

4. The bounds on the parameters used to approximate the virtual environment are known or may be accurately estimated.

3.2 System Overview

The system that will be considered is shown in Fig. 3.1. The operator applies a force $\boldsymbol{F}_h(t)$ that is measured by a force sensor. The measured operator force is used to calculate the ideal virtual tool trajectory $\boldsymbol{x}_e(t)$ and the ideal virtual environment force $\boldsymbol{F}_e(t)$. The complex virtual environment (VE) running in discrete time is approximated as a continuous-time system that is being sampled, delayed by one sampling time (T_d), and then run through a zero-order hold, resulting in $\boldsymbol{x}_{e,z}(t-T_d)$ and $\boldsymbol{F}_{e,z}(t-T_d)$. This approximation assumes that the VE produces a stable output (such as by running a batch calculation based on a buffered $\boldsymbol{F}_h$). The predictor uses the operator force, sampled position and velocity, and sampled environment force to generate a predicted trajectory $\hat{\boldsymbol{x}}_e$. The predicted trajectory is tracked by the controller, which uses the operator force and haptic device joint position and velocity $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ to generate the control torque. The haptic device applies the control torque and interacts with the operator.

3.3 System Modelling

This section describes the kinematic and dynamic models used in the haptic system. For ease of representation the following substitutions are used:

$$\begin{aligned} c_i &= \cos(q_i) \qquad & s_i &= \sin(q_i) \\ c_{ij} &= \cos(q_i + q_j) \qquad & s_{ij} &= \sin(q_i + q_j) \end{aligned}$$

The Phantom Omni® Haptic Device will be used for experimental testing (see 5.2). The Phantom Omni® is a 3-DOF serial-link manipulator, however the kinematic and dynamic models could be extended for an n-DOF serial-link manipulator.

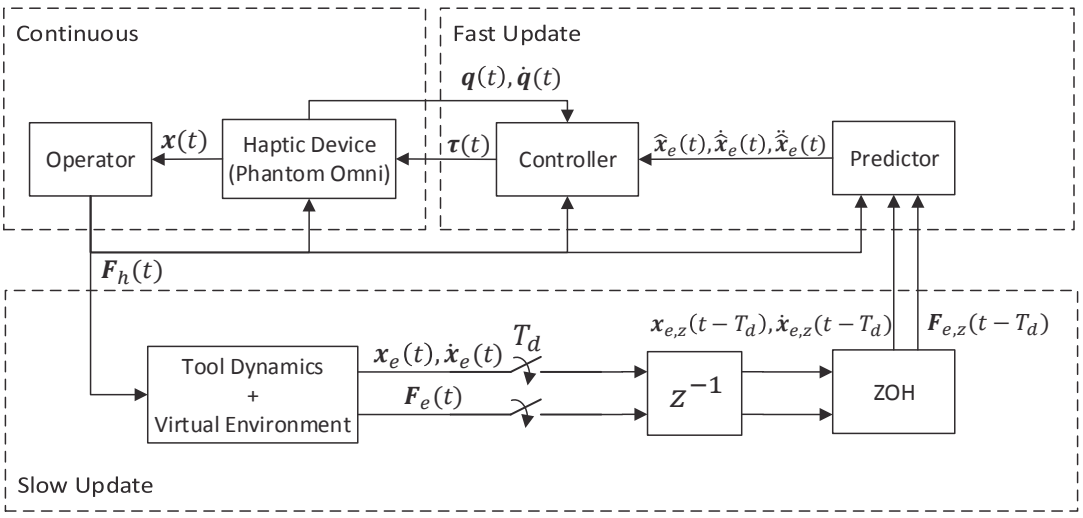


Figure 3.1: Haptic surgical simulator control system diagram

3.3.1 Phantom Omni® Kinematic Model

The forward, inverse, and differential kinematics of the Phantom Omni® can be derived explicitly. The vector $\boldsymbol{q} \in \Re^{3\times 1}$ represents the joint variables, specifically the joint angles, and the vector $\boldsymbol{x} \in \Re^{3\times 1}$ represents the position of the end-effector in Cartesian coordinates.

Fig. 3.2 shows a schematic of the reference frames used in the derivation of the kinematic model. Four reference frames, numbered from 0 to 3, are shown. The base frame (Frame 0) is selected to be at the joint between the first and second links, with the z_0-axis vertical and the x_0-axis pointing toward the front of the device. Frame 3 is located at the joint between the stylus and the third link, with the x_3-axis along the length of the stylus toward the operator.

Forward Kinematics

A vector in Frame 3 can be expressed in the base frame using a linear transformation as follows,

$$ {}^0\boldsymbol{x}' = {}^0_3T(\boldsymbol{q})\,{}^3\boldsymbol{x}'. \tag{3.1}$$

The transformation matrix is derived systematically from each reference frame to the next using the standardized Denavit-Hartenberg (DH) convention. Table 3.1 shows

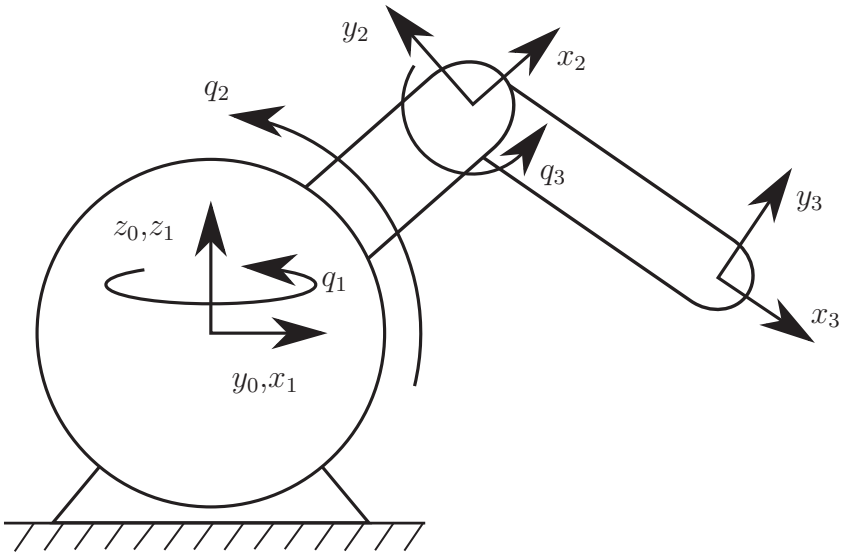


Figure 3.2: Schematic of the Phantom Omni device, establishing the nomenclature and reference frames used in the derivation of the device kinematics.

Table 3.1: Denavit-Hartenberg parameters of the Phantom Omni device.

Link	d_i	ϑ_i	a_i	α_i
1	l_1	q_1	0	$\pi/2$
2	0	q_2	l_2	0
3	0	q_3	l_3	0

the DH parameters for the Phantom Omni. Since $l_1 = 0$, it is omitted in the following derivation:

$$ {}^0_1T = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^1_2T = \begin{bmatrix} c_2 & -s_2 & 0 & l_2c_2 \\ s_2 & c_2 & 0 & l_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad {}^2_3T = \begin{bmatrix} c_3 & -s_3 & 0 & l_3c_3 \\ s_3 & c_3 & 0 & l_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

where, as previously mentioned, $c_i = \cos q_i$ and $s_i = \sin q_i$.

$$ {}^0_3T = {}^0_1T\,{}^1_2T\,{}^2_3T = \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & (l_2c_2 + l_3c_{23})c_1 \\ s_1c_{23} & -s_1s_{23} & -c_1 & (l_2c_2 + l_3c_{23})s_1 \\ s_{23} & c_{23} & 0 & l_2s_2 + l_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

The position of the end-effector can be extracted from the fourth column of ${}^0_3T$, resulting in (3.2).

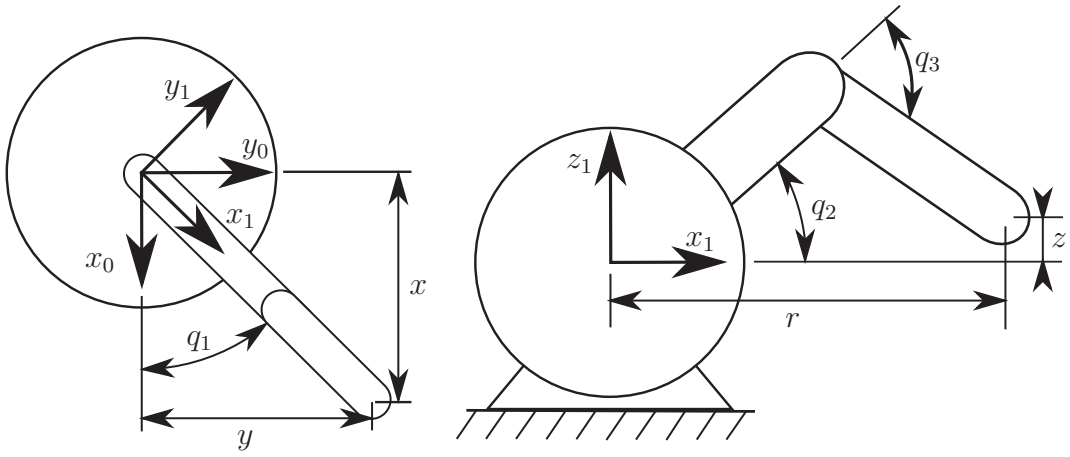


Figure 3.3: A schematic illustrating the geometry used in the inverse kinematics.

$$\boldsymbol{x} = {}^0\boldsymbol{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (l_2 \cos q_2 + l_3 \cos(q_2 + q_3)) \cos q_1 \\ (l_2 \cos q_2 + l_3 \cos(q_2 + q_3)) \sin q_1 \\ l_2 \sin q_2 + l_3 \sin(q_2 + q_3) \end{bmatrix}. \tag{3.2}$$

Inverse Kinematics

The joint variables can be derived geometrically for a given end-effector position. Fig. 3.3 shows the geometry used to solve the inverse kinematics of the Phantom Omni®. The position of the first joint, q_1, can be determined from the x-y position of the end-effector using (3.3),

$$q_1 = \tan^{-1}\left(\frac{y}{x}\right). \tag{3.3}$$

To solve for the positions of the second and third joints, q_2 and q_3, it is easier to consider the geometry in Frame 1. The distance along the x_1 axis, denoted by r in Fig. 3.3, can be calculated using (3.4). Using the cosine law and solving for q_3 leads to (3.5). Note that the cosine law would result in two possible solutions for q_3, one positive and one negative. However, due to the physical constraints of the Phantom Omni®, only the negative solution is feasible.

$$r = \sqrt{x^2 + y^2}, \tag{3.4}$$

$$q_3 = \cos^{-1}\left(\frac{l_2^2 + l_3^2 - r^2 - z^2}{2l_2 l_3}\right) - \pi. \tag{3.5}$$

Finally, the position of the second joint can be solved based on the known angles in the triangle formed by Link 2, Link 3, and r, resulting in (3.6).

$$q_2 = \tan^{-1}\left(\frac{z}{r}\right) + \tan^{-1}\left(\frac{-l_3 \sin q_3}{l_2 + l_3 \cos q_3}\right). \tag{3.6}$$

Differential Kinematics

The velocity of the end-effector $\dot{\boldsymbol{x}}$ is linearly related to the joint angular velocities $\dot{\boldsymbol{q}}$ by the Jacobian matrix $J \in \Re^{3\times3}$, as shown in (3.7). The Jacobian can be calculated using (3.8), where $\hat{\boldsymbol{z}}_i \in \Re^{3\times1}$ is the unit vector along the axis of joint i, and $\boldsymbol{p}_i \in \Re^{3\times1}$ is the vector from the base frame to the position of joint i. The resulting matrix is given in (3.9).

$$\dot{\boldsymbol{x}} = J(\boldsymbol{x}, \boldsymbol{q})\dot{\boldsymbol{q}}, \tag{3.7}$$

$$J = \begin{bmatrix} \hat{\boldsymbol{z}}_1 \times (\boldsymbol{x} - \boldsymbol{p}_1) & \hat{\boldsymbol{z}}_2 \times (\boldsymbol{x} - \boldsymbol{p}_2) & \hat{\boldsymbol{z}}_3 \times (\boldsymbol{x} - \boldsymbol{p}_3) \end{bmatrix}, \tag{3.8}$$

$$J = \begin{bmatrix} -rs_1 & -zc_1 & -l_3 c_1 s_{23} \\ rc_1 & -zs_1 & -l_3 s_1 s_{23} \\ 0 & r & l_3 c_{23} \end{bmatrix}. \tag{3.9}$$

The acceleration of the end-effector $\ddot{\boldsymbol{x}}$ is influenced by both the joint angular velocity $\dot{\boldsymbol{q}}$ (the centripetal component) as well as the joint angular acceleration $\ddot{\boldsymbol{q}}$, as shown in (3.10).

$$\ddot{\boldsymbol{x}} = \dot{J}\dot{\boldsymbol{q}} + J\ddot{\boldsymbol{q}}. \tag{3.10}$$

The derivative of the Jacobian is given in (3.11). The bold zeros represent appropriately sized zero vectors.

$$\dot{J} = \begin{bmatrix} \dot{\boldsymbol{q}}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\dot{\boldsymbol{q}}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\dot{\boldsymbol{q}}^T \end{bmatrix} \begin{bmatrix} -rc_1 & zs_1 & l_3 s_1 s_{23} \\ zs_1 & -rc_1 & -l_3 c_1 c_{23} \\ l_3 s_1 s_{23} & -l_3 c_1 c_{23} & -l_3 c_1 c_{23} \\ rs_1 & zc_1 & l_3 c_1 s_{23} \\ zc_1 & rs_1 & l_3 s_1 c_{23} \\ l_3 c_1 s_{23} & l_3 s_1 c_{23} & l_3 s_1 c_{23} \\ 0 & 0 & 0 \\ 0 & z & l_3 s_{23} \\ 0 & l_3 s_{23} & l_3 s_{23} \end{bmatrix}. \tag{3.11}$$

Inverse Differential Kinematics

Since the relationship between $\dot{\boldsymbol{x}}$ and $\dot{\boldsymbol{q}}$ is linear it is straightforward to derive the inverse equation given in (3.12) by rearranging (3.7). The analytical expression of the inverse Jacobian is given in (3.13).

$$\dot{\boldsymbol{q}} = J^{-1}\dot{\boldsymbol{x}}, \tag{3.12}$$

$$J^{-1} = \begin{bmatrix} -\frac{s_1}{r} & \frac{c_1}{r} & 0 \\ \frac{c_1 c_{23}}{l_2 s_3} & \frac{s_1 c_{23}}{l_2 s_3} & \frac{s_{23}}{l_2 s_3} \\ -\frac{rc_1}{l_2 l_3 s_3} & -\frac{rs_1}{l_2 l_3 s_3} & -\frac{z}{l_2 l_3 s_3} \end{bmatrix}. \tag{3.13}$$

Similarly, the equation for angular joint acceleration can be derived by rearranging (3.10), resulting in (3.14).

$$\ddot{\boldsymbol{q}} = J^{-1}\left(\ddot{\boldsymbol{x}} - \dot{J}\dot{\boldsymbol{q}}\right). \tag{3.14}$$

3.3.2 Phantom Omni® Dynamic Model

The generalized joint space dynamic model of a serial-link manipulator is shown in (3.15).

$$H(\boldsymbol{q})\ddot{\boldsymbol{q}} + (C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + B)\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}) = \boldsymbol{\tau}_c + J^T \boldsymbol{F}_h, \tag{3.15}$$

where $H \in \Re^{3\times3}$ is the inertia matrix, $C \in \Re^{3\times3}$ is the Coriolis and centrifugal torque matrix, $B \in \Re^{3\times3}$ is the viscous friction matrix, $\boldsymbol{\tau}_g \in \Re^{3\times1}$ is the gravity torque vector, $\boldsymbol{\tau}_c \in \Re^{3\times1}$ is the control input torque, and $\boldsymbol{F}_h \in \Re^{3\times1}$ is the human operator force applied to the haptic device.

Dynamic Properties

The dynamic model of the haptic device has several useful properties that will be used in the design of the control algorithm.

Property 1. *The inertia matrix H is symmetric and positive-definite.*

$$H = H^T > 0.$$

Property 2. *The inertia and Coriolis force matrices have the following skew-symmetry property:*

$$\boldsymbol{\xi}^T \left(\dot{H} - 2C \right) \boldsymbol{\xi} = 0,$$

for any vector $\boldsymbol{\xi}$ of appropriate dimensions.

Property 3. *The dynamic model can be represented in a linear parametric form in terms of the constant parameter vector $\boldsymbol{\theta}$. This form is given in* (3.16).

$$\boldsymbol{\varphi}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\boldsymbol{\theta} = H(\boldsymbol{q})\ddot{\boldsymbol{q}} + (C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + B)\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}), \tag{3.16}$$

where $\boldsymbol{\varphi}$ is a matrix function containing the states and any known parameters.

Derivation

The matrices H and C, and the vector $\boldsymbol{\tau}_g$ are derived from the Lagrangian $\mathbb{L} \in \Re$ of the haptic device [40], given in (3.17), where $\mathbb{T} \in \Re$ and $\mathbb{U} \in \Re$ are the total kinetic and potential energy of the haptic device, respectively,

$$\mathbb{L} = \mathbb{T} - \mathbb{U}. \tag{3.17}$$

The total kinetic and potential energy is derived by considering each link as a rod with mass m_i, a center of mass located at $l_{c,i}$ from the base of the link, and a moment of inertia $\mathcal{I}_i$ about the center of mass. Therefore, the total kinetic and potential energy are expressed by (3.18) and (3.19).

$$\mathbb{T} = \sum_{i=1}^{3} \mathbb{T}_i = \sum_{i=1}^{3} \left(\frac{1}{2} m_i v_{c,i}^2 + \frac{1}{2} \mathcal{I}_i \omega_i^2 \right), \tag{3.18}$$

$$\mathbb{U} = \sum_{i=1}^{3} \mathbb{U}_i = \sum_{i=1}^{3} m_i g z_{c,i}, \tag{3.19}$$

where g is the acceleration due to gravity, and $v_{c,i}$, ω_i, and $z_{c,i}$ are the center of mass velocity, angular velocity, and center of mass z position, of link i, respectively. Those variables can be expressed in terms of $\boldsymbol{q}$ using forward kinematics.

The dynamic equations are derived from Lagrange's Equation, as

$$\frac{d}{dt}\boldsymbol{\nabla}_{\dot{\boldsymbol{q}}}\mathbb{L} - \boldsymbol{\nabla}_{\boldsymbol{q}}\mathbb{L} = H(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}), \tag{3.20}$$

where $\boldsymbol{\nabla}_{\dot{\boldsymbol{q}}} = \begin{bmatrix} \frac{d}{d\dot{q}_1} & \frac{d}{d\dot{q}_2} & \frac{d}{d\dot{q}_3} \end{bmatrix}^T$ and $\boldsymbol{\nabla}_{\boldsymbol{q}} = \begin{bmatrix} \frac{d}{dq_1} & \frac{d}{dq_2} & \frac{d}{dq_3} \end{bmatrix}^T$. Lagrange's equation does not result in a unique expression of C, therefore C is selected so that it meets Property 2. The resulting H and C matrices, and $\boldsymbol{\tau}_g$ vector, are given in (3.21), (3.22), and (3.23).

$$H = \begin{bmatrix} h_{11} & 0 & 0 \\ 0 & h_{22} & h_{23} \\ 0 & h_{32} & h_{33} \end{bmatrix}, \tag{3.21}$$

$$h_{11} = (m_2 1_{c2}^2 + \mathcal{I}_2 + m_3 l_2^2)c_2^2 + (m_3 l_{c3}^2 + \mathcal{I}_3)c_{23}^2 + 2m_3 l_2 l_{c3} c_2 c_{23} + \mathcal{I}_1,$$

$$h_{22} = m_2 1_{c2}^2 + \mathcal{I}_2 + m_3 l_{c3}^2 + \mathcal{I}_3 + m_3 l_2^2 + 2m_3 l_2 l_{c3} c_3,$$

$$h_{23} = m_3 l_{c3}^2 + \mathcal{I}_3 + m_3 l_2 l_{c3} c_3,$$

$$h_{32} = h_{23},$$

$$h_{33} = m_3 l_{c3}^2 + \mathcal{I}_3,$$

$$C = \begin{bmatrix} -(a_1\dot{q}_2 + a_2\dot{q}_3) & -a_1\dot{q}_1 & -a_2\dot{q}_1 \\ a_1\dot{q}_1 & -a_3\dot{q}_3 & -a_3(\dot{q}_2 + \dot{q}_3) \\ a_2\dot{q}_1 & a_3\dot{q}_2 & 0 \end{bmatrix}, \tag{3.22}$$

$$a_1 = (m_2 l_{c2}^2 + \mathcal{I}_2 + m_3 l_2^2)c_2 s_2 + m_3 l_2 l_{c3} \sin(2q_2 + q_3) + (m_3 l_{c3}^2 + \mathcal{I}_3)c_{23} s_{23},$$

$$a_2 = m_3 l_2 l_{c3} c_2 s_{23} + (m_3 l_{c3}^2 + \mathcal{I}_3)c_{23} s_{23},$$

$$a_3 = m_3 l_2 l_{c3} s_3,$$

$$\boldsymbol{\tau}_g = g \begin{bmatrix} 0 \\ (m_2 l_{c2} + m_3 l_2)c_2 + m_3 l_{c3} c_{23} \\ m_3 l_{c3} c_{23} \end{bmatrix}. \tag{3.23}$$

The viscous friction matrix is selected so that friction acts independently on each joint, as shown in (3.24), where $b_i > 0$ is the friction associated with joint i,

$$B = \begin{bmatrix} b_1 & 0 & 0 \\ 0 & b_2 & 0 \\ 0 & 0 & b_3 \end{bmatrix}. \tag{3.24}$$

Parameterized Dynamics

The dynamics of the haptic device must be represented in a form that separates the dynamic parameters from the state variables to develop an adaptation law for the adaptive control of the Phantom Omni. The parameterized dynamics are given in (3.16). There are nine independent unknown parameters in the dynamic model. The parameter vector $\boldsymbol{\theta} \in \Re^{9\times 1}$ is given by (3.25).

$$\boldsymbol{\theta} = \begin{bmatrix} \mathcal{I}_1 & m_2 l_{c2}^2 + \mathcal{I}_2 + m_3 l_2^2 & m_3 l_{c3}^2 + \mathcal{I}_3 & m_3 l_2 l_{c3} & m_2 l_{c2} + m_3 l_2 & m_3 l_{c3} & b_1 & b_2 & b_3 \end{bmatrix}^T. \tag{3.25}$$

Expressing the elements of the dynamic matrices in terms of $\boldsymbol{\theta}$ results in:

$$\begin{aligned} h_{11} &= \theta_1 + \theta_2 c_2^2 + \theta_3 c_{23}^2 + 2\theta_4 c_2 c_{23}, \\ h_{22} &= \theta_2 + \theta_3 + 2\theta_4 c_3, \\ h_{23} &= \theta_3 + \theta_4 c_3, \\ h_{33} &= \theta_3, \\ a_1 &= \theta_2 c_2 s_2 + \theta_3 c_{23} s_{23} + \theta_4 s_{223}, \\ a_2 &= \theta_3 c_{23} s_{23} + \theta_4 c_2 s_{23}, \\ a_3 &= \theta_4 s_3, \\ \boldsymbol{\tau}_g &= \begin{bmatrix} 0 \\ \theta_5 g c_2 + \theta_6 g c_{23} \\ \theta_6 g c_{23} \end{bmatrix}. \end{aligned}$$

Therefore, $\boldsymbol{\varphi} \in \Re^{3\times 9}$ is given by (3.26).

$$\boldsymbol{\varphi} = \begin{bmatrix} \ddot{q}_1 & c_2^2\ddot{q}_1 - 2c_2 s_2 \dot{q}_1 \dot{q}_2 & c_{23}^2\ddot{q}_1 - c_{23}s_{23}(2\dot{q}_2 + 2\dot{q}_3)\dot{q}_1 & \cdots \\ 0 & \ddot{q}_2 + c_2 s_2 \dot{q}_1^2 & \ddot{q}_2 + \ddot{q}_3 + c_{23}s_{23}\dot{q}_1^2 & \cdots \\ 0 & 0 & \ddot{q}_2 + \ddot{q}_3 + c_{23}s_{23}\dot{q}_1^2 & \cdots \end{bmatrix}$$

$$\begin{bmatrix} \cdots & 2c_2c_{23} - 2(s_{223}\dot{q}_2 + c_2 s_{23}\dot{q}_3)\dot{q}_1 & 0 & 0 & \dot{q}_1 & 0 & 0 \\ \cdots & 2c_3\ddot{q}_2 + c_3\ddot{q}_3 + s_{223}\dot{q}_1^2 - s_3(2\dot{q}_2 + \dot{q}_3)\dot{q}_3 & gc_2 & gc_{23} & 0 & \dot{q}_2 & 0 \\ \cdots & c_3\ddot{q}_2 + c_2 s_{23}\dot{q}_1^2 + s_3\dot{q}_2^2 & 0 & gc_{23} & 0 & 0 & \dot{q}_3 \end{bmatrix} \tag{3.26}$$

Controller Parameterized Dynamics

The acceleration of the haptic device is not measurable, therefore another parameterization of the dynamics will be necessary for the design of the nonlinear controller. Let $\dot{\boldsymbol{q}}_r$ and $\ddot{\boldsymbol{q}}_r$ be the velocity and acceleration of a reference signal. The dynamics of the haptic device can be expressed in terms of the reference signal, as shown in (3.27),

$$\boldsymbol{\varphi}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}_r, \ddot{\boldsymbol{q}}_r)\boldsymbol{\theta} = H(\boldsymbol{q})\ddot{\boldsymbol{q}}_r + (C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + B)\dot{\boldsymbol{q}}_r + \boldsymbol{\tau}_g(\boldsymbol{q}). \tag{3.27}$$

The resulting expression for $\boldsymbol{\varphi}_r \in \Re^{3\times 9}$ is,

$$\boldsymbol{\varphi}_r = \begin{bmatrix} \ddot{q}_{r1} & c_2^2\ddot{q}_{r1} - c_2 s_2(\dot{q}_2\dot{q}_{r1} + \dot{q}_1\dot{q}_{r2}) & c_{23}^2\ddot{q}_{r1} - c_{23}s_{23}\left(\dot{q}_{r1}(\dot{q}_2 + \dot{q}_3) + \dot{q}_1(\dot{q}_{r2} + \dot{q}_{r3})\right) & \cdots \\ 0 & \ddot{q}_{r2} + c_2 s_2 \dot{q}_1\dot{q}_{r1} & \ddot{q}_{r2} + \ddot{q}_{r3} + c_{23}s_{23}\dot{q}_1\dot{q}_{r1} & \cdots \\ 0 & 0 & \ddot{q}_{r2} + \ddot{q}_{r3} + c_{23}s_{23}\dot{q}_1\dot{q}_{r1} & \cdots \end{bmatrix}$$

$$\begin{bmatrix} \cdots & 2c_2c_{23}\ddot{q}_{r1} - s_{223}(\dot{q}_{r1}\dot{q}_2 + \dot{q}_1\dot{q}_{r2}) - c_2 s_{23}(\dot{q}_{r1}\dot{q}_3 + \dot{q}_1\dot{q}_{r3}) & 0 & 0 & \cdots \\ \cdots & 2c_3\ddot{q}_{r2} + c_3\ddot{q}_{r3} + s_{223}\dot{q}_1\dot{q}_{r1} - s_3((\dot{q}_{r2} + \dot{q}_{r3})\dot{q}_3 + \dot{q}_2\dot{q}_{r3}) & gc_2 & gc_{23} & \cdots \\ \cdots & c_3\ddot{q}_{r2} + c_2 s_{23}\dot{q}_1\dot{q}_{r1} + s_3\dot{q}_2\dot{q}_{r2} & 0 & gc_{23} & \cdots \end{bmatrix}$$

$$\begin{bmatrix} \cdots & \dot{q}_{r1} & 0 & 0 \\ \cdots & 0 & \dot{q}_{r2} & 0 \\ \cdots & 0 & 0 & \dot{q}_{r3} \end{bmatrix} \tag{3.28}$$

3.3.3 Tool Dynamics

The general dynamics for the tool are given in (3.29),

$$m_t\ddot{\boldsymbol{x}}_e(t) + b_t\dot{\boldsymbol{x}}_e(t) + k_t\boldsymbol{x}_e(t) = \boldsymbol{F}_h(t) - \boldsymbol{F}_e(t). \tag{3.29}$$

where $\boldsymbol{x}_e \in \Re^{3\times 1}$ and $\dot{\boldsymbol{x}}_e$ are the position and velocity of the virtual tool, and $m_t \in \Re$, $b_t \in \Re$, and $k_t \in \Re$ are the mass, damping, and stiffness of the tool, respectively, $\boldsymbol{F}_h \in \Re^{3\times 1}$ is the operator input force, and $\boldsymbol{F}_e \in \Re^{3\times 1}$ is the virtual environment force. The mass, stiffness, and damping of the tool are known and constant since the tool itself does not change during a task.

3.3.4 Virtual Environment Dynamics

The virtual environment is a nonlinear system with unknown dynamics, however it is assumed that the unknown dynamics can be expressed in a pseudo-linear parameter

varying form. The environment dynamics are given in (3.30),

$$\boldsymbol{F}_e = B_e(t, \boldsymbol{x}_e, \dot{\boldsymbol{x}}_e)\dot{\boldsymbol{x}}_e(t) + K_e(t, \boldsymbol{x}_e, \dot{\boldsymbol{x}}_e)\boldsymbol{x}_e(t), \tag{3.30}$$

where $B_e \in \Re^{3\times3}$ and $K_e \in \Re^{3\times3}$ are unknown nonlinear functions representing the damping and stiffness, respectively, associated with the environment.
Assuming that the damping and stiffness forces are independent for each degree of freedom, the environment force can be considered in a single degree of freedom as

$$F_e = b_e(t, x_e, \dot{x}_e)\dot{x}_e(t) + k_e(t, x_e, \dot{x}_e)x_e(t), \tag{3.31}$$

where $F_e \in \Re$, $x_e \in \Re$, $k_e \in \Re$, and $b_e \in \Re$ are the virtual environment force, virtual tool position, virtual environment stiffness, and virtual environment damping, respectively, expressed in 1-DOF.
The 1-DOF virtual environment dynamics may be expressed in a parameterized form as,

$$F_e = \boldsymbol{\varphi}_e^T \boldsymbol{\theta}_e \tag{3.32}$$

where $\boldsymbol{\varphi}_e = [x_e \quad \dot{x}_e]^T$ and $\boldsymbol{\theta}_e = [k_e \quad b_e]^T$.

Alternative Virtual Environment Model

For some virtual environments it may be more accurate to include a force 'offset' term, such as

$$F_e = c_e(t, x_e, \dot{x}_e) + k_e(t, x_e, \dot{x}_e)x_e + b_e(t, x_e, \dot{x}_e)\dot{x}_e, \tag{3.33}$$

where $c_e \in \Re$ is the force offset. This model expresses the dynamics as a tangent line to the actual nonlinear force function.
This virtual environment force model may also be parameterized as in (3.32), where $\boldsymbol{\varphi}_e = [x_e \quad \dot{x}_e \quad 1]^T$ and $\boldsymbol{\theta}_e = [k_e \quad b_e \quad c_e]^T$.

Continuous Approximation of Piecewise Stiffness

Human soft tissue exhibits a nonlinear stiffness, so that in some cases the restoring force can be described by a piecewise linear function of penetration depth [43]. For the virtual environment it would be convenient to have a continuous and continuously

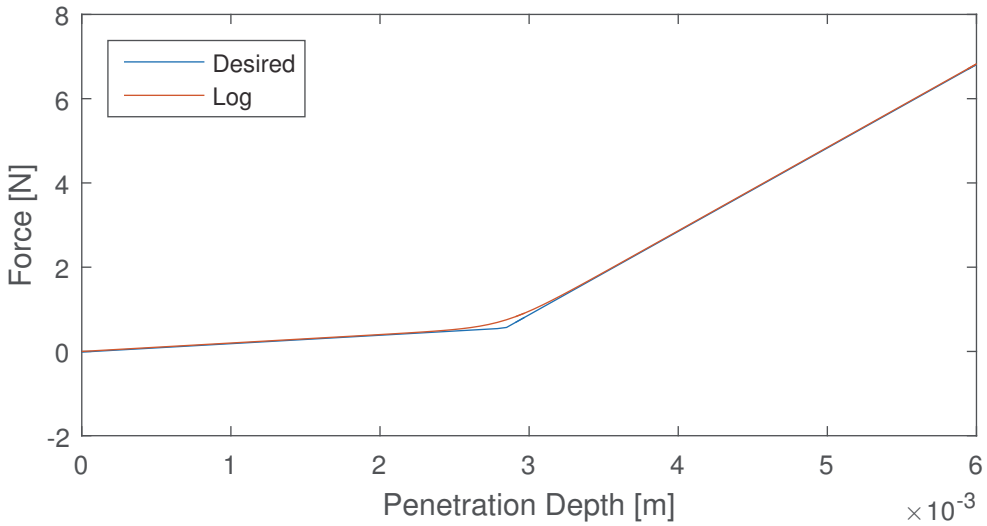


Figure 3.4: Comparison of the piecewise linear and continuous approximation of the virtual environment force.

differentiable function that approximates this behaviour. This function will be used to generate the virtual environment force, it will not be used in the predictor.
The desired behaviour is to have an initial stiffness (slope) $k_{e,1}$, that at a penetration depth x_s transitions to a higher stiffness (slope) $k_{e,2}$. The designed function is,

$$F_k(x) = k_{e,1}x + (k_{e,2} - k_{e,1})\frac{x_s}{R}\ln\left(1 + e^{R\left(\frac{x}{x_s}-1\right)}\right) \tag{3.34}$$

where x is the penetration depth and R affects the rate of transition from one slope to the next, with larger R values resulting in a sharper transition. Fig. 3.4 shows a comparison of the force generated by (3.34) compared to the piecewise linear equivalent for $k_{e,1} = 200$ N/m, $k_{e,1} = 2000$ N/m, $x_s = 0.00284$ m, and $R = 20$.

3.4 Performance Metrics

The performance of a haptic system is described by transparency. Since in the ideal case the virtual environment would output a trajectory that would be perfectly tracked by the haptic device, the transparency of the haptic system will be measured by comparing the actual trajectory tracked by the haptic device to the ideal trajectory. In a real system the ideal trajectory would not be available, but it is available in the test system.

The total error is defined by (3.35), where $\boldsymbol{x}$ is the position of the end-effector of the haptic device and $\boldsymbol{x}_e$ is the ideal position of the virtual tool.

$$\boldsymbol{e}_{tot} = \begin{bmatrix} \boldsymbol{x} - \boldsymbol{x}_e \\ \dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_e \end{bmatrix}. \tag{3.35}$$

An increase in the total error corresponds to a decrease in transparency, and vice-versa.

The effect of sampling appears as a $1/T_d$ Hz noise in the error signal making it difficult to visually compare two error signals. Therefore two scalar measurements of the total error are used to describe the performance of the system. In order to determine the effective bound on the error, the first performance metric is the maximum of the 2-norm of the total error, given in (3.36),

$$\|\boldsymbol{e}_{tot}\|_{max} = \max\left(\sqrt{\boldsymbol{e}_{tot}^T \boldsymbol{e}_{tot}}\right). \tag{3.36}$$

In order to measure the average trend of the error, the $\mathcal{L}_2$ norm of the error will be considered, as defined by (3.37),

$$\|\boldsymbol{e}_{tot}\|_{\mathcal{L}_2} = \sqrt{\int_0^\infty \boldsymbol{e}_{tot}^T \boldsymbol{e}_{tot} dt}. \tag{3.37}$$

In particular we are interested in the change in error between a system with the predictor and a system without the predictor. Therefore in some cases the relative error change, given by (3.38), will be considered.

$$\text{Relative Error Change} = \frac{\text{Error with prediction} - \text{Error without prediction}}{\text{Error without prediction}} \tag{3.38}$$

3.5 Control Objectives

The control objective is to design the haptic controller and predictor such that the following requirements are met:

1. *Stability*: The stability of the closed loop system must be maintained for a passive operator, bounded virtual environment parameters, and in the presence of computational delay and sampling in the feedback.

2. *Transparency*: The haptic device must provide the human operator with impedance behaviour similar to the actual feeling of interacting with the virtual environment via a surgical tool. In other words, the controller will cancel the haptic device dynamics and display the virtual environment dynamics instead.

The transparency of the haptic system can be guaranteed if the end-effector of the haptic device asymptotically tracks the trajectory of the simulated surgical instrument. The stability of the system can be guaranteed if the stability of the virtual environment is guaranteed for any operator force input. Therefore this project addresses two problems: asymptotic tracking of a reference signal generated by the virtual environment and stability of the virtual environment in the presence of delays.
The transparency condition is considered met if either $\|\boldsymbol{e}_{tot}\|_{max}$ or $\|\boldsymbol{e}_{tot}\|_{\mathcal{L}_2}$ is lower for a system with the predictor than for a system without the predictor. The performance improvement is considered significant if (3.38) exceeds 7%, i.e. the Weber fraction for force detection [43].

Chapter 4

Controller and Predictor Design

4.1 Nonlinear PD Controller Design for Known Parameters

The haptic device is controlled by a nonlinear controller, designed so that the haptic device end-effector tracks a desired trajectory. The controller will be designed in the joint space since the Phantom Omni® uses joint torque as its control input. The Phantom Omni® measures the joint positions through encoders; the velocity can be calculated through a numerical difference method. The acceleration is unmeasurable since the numerical difference method amplifies the signal noise. Therefore the reference signal is chosen to avoid the use of $\ddot{\boldsymbol{q}}$. The selected reference signal is defined as

$$\dot{\boldsymbol{q}}_r = \dot{\boldsymbol{q}}_e - \Lambda(\boldsymbol{q} - \boldsymbol{q}_e), \tag{4.1}$$

where $\boldsymbol{q}_e$ and $\dot{\boldsymbol{q}}_e$ are the desired joint position and velocity calculated from the VE trajectory using inverse kinematics, and $\Lambda > 0$ is a control gain matrix.
The controller is designed as

$$\boldsymbol{\tau}_c = \boldsymbol{\varphi}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}_r, \ddot{\boldsymbol{q}}_r)\boldsymbol{\theta} - K_d\boldsymbol{\delta} - J^T\boldsymbol{F}_h, \tag{4.2}$$

where $K_d > 0$ is a control gain matrix and $\boldsymbol{\delta}$ is the controller error defined as

$$\boldsymbol{\delta} = \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_r. \tag{4.3}$$

The closed-loop dynamics given in (4.4) can be derived by inserting (4.2) into (3.15) and using (3.27).

$$\begin{aligned} H(\boldsymbol{q})\ddot{\boldsymbol{q}} + (C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + B)\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}) &= \boldsymbol{\varphi}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}_r, \ddot{\boldsymbol{q}}_r)\boldsymbol{\theta} - K_d\boldsymbol{\delta} - J^T\boldsymbol{F}_h + J^T\boldsymbol{F}_h, \\ H(\boldsymbol{q})\ddot{\boldsymbol{q}} + (C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + B)\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}) &= H(\boldsymbol{q})\ddot{\boldsymbol{q}}_r + (C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + B)\dot{\boldsymbol{q}}_r + \boldsymbol{\tau}_g(\boldsymbol{q}) - K_d\boldsymbol{\delta}, \\ H(\boldsymbol{q})\dot{\boldsymbol{\delta}} + (C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + B + K_d)\,\boldsymbol{\delta} &= 0. \end{aligned} \tag{4.4}$$

Stability Analysis

The stability of the closed loop system can be shown by defining the following Lyapunov function:

$$V = \frac{1}{2}\boldsymbol{\delta}^T H \boldsymbol{\delta} > 0 \qquad \forall \boldsymbol{\delta} \neq 0. \tag{4.5}$$

Taking the derivative of (4.5) and using the symmetry of H leads to

$$\begin{aligned}\dot{V} &= \frac{1}{2}\left(\boldsymbol{\delta}^T H \dot{\boldsymbol{\delta}} + \dot{\boldsymbol{\delta}}^T H \boldsymbol{\delta} + \boldsymbol{\delta}^T \dot{H} \boldsymbol{\delta}\right), \\ &= \boldsymbol{\delta}^T H \dot{\boldsymbol{\delta}} + \frac{1}{2}\boldsymbol{\delta}^T \dot{H} \boldsymbol{\delta}.\end{aligned}$$

Rearranging (4.4) for $H\dot{\boldsymbol{\delta}}$ and substituting into the previous equation,

$$\begin{aligned}\dot{V} &= -\boldsymbol{\delta}^T \left(C + B + K_d\right) \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T \dot{H} \boldsymbol{\delta}, \\ &= \frac{1}{2}\boldsymbol{\delta}^T \left(\dot{H} - 2C\right) \boldsymbol{\delta} - \boldsymbol{\delta}^T (K_d + B)\boldsymbol{\delta}.\end{aligned}$$

Finally, using Property 2, and given that K_d and B are positive definite results in

$$\dot{V} = -\boldsymbol{\delta}^T (K_d + B)\boldsymbol{\delta} < 0, \qquad \forall \boldsymbol{\delta} \neq 0. \tag{4.6}$$

Considering (4.5) and (4.6) and applying Barbalat's Lemma [41], it is shown that the closed-loop system regulates the controller error $\boldsymbol{\delta}$ to zero and the control gain K_d affects the rate of convergence.

4.2 Nonlinear Adaptive Controller Design

During experimental testing it was found that the offline measurements of the Phantom Omni® dynamic parameters are not sufficiently accurate to provide a stable controller. An adaptive controller was designed to estimate the parameters online. The adaptive controller has a similar structure to the controller in (4.2), except that the estimated parameter vector, $\hat{\boldsymbol{\theta}}$, is used. The adaptive controller is given by (4.7).

$$\boldsymbol{\tau}_c = \boldsymbol{\varphi}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}_r, \ddot{\boldsymbol{q}}_r)\hat{\boldsymbol{\theta}} - K_d\boldsymbol{\delta} - J^T \boldsymbol{F}_h. \tag{4.7}$$

The adaptive controller uses an adaptation law to provide an online estimate of the haptic device's dynamic parameters. The adaptation law is given in (4.8), where $\Gamma > 0$ is the symmetric adaptation gain matrix,

$$\dot{\hat{\boldsymbol{\theta}}} = -\Gamma \boldsymbol{\varphi}_r^T \boldsymbol{\delta}. \tag{4.8}$$

The closed-loop dynamics change accordingly, as

$$H(\boldsymbol{q})\dot{\boldsymbol{\delta}} + (C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + K_d)\,\boldsymbol{\delta} = \boldsymbol{\varphi}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}, \dot{\boldsymbol{q}}_r, \ddot{\boldsymbol{q}}_r)\tilde{\boldsymbol{\theta}}, \tag{4.9}$$

where $\tilde{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}$ is the parameter adaptation error.

Stability Analysis

The stability of the closed loop system can be shown by defining the following Lyapunov function, similar to (4.5),

$$V = \frac{1}{2}\boldsymbol{\delta}^T H \boldsymbol{\delta} + \frac{1}{2}\tilde{\boldsymbol{\theta}}^T \Gamma^{-1} \tilde{\boldsymbol{\theta}} > 0, \qquad \forall \boldsymbol{\delta} \neq 0,\ \tilde{\boldsymbol{\theta}} \neq 0. \tag{4.10}$$

Taking the derivative of (4.10) leads to

$$\dot{V} = \boldsymbol{\delta}^T H \dot{\boldsymbol{\delta}} + \frac{1}{2}\boldsymbol{\delta}^T \dot{H} \boldsymbol{\delta} + \frac{1}{2}(\dot{\tilde{\boldsymbol{\theta}}}^T \Gamma^{-1} \tilde{\boldsymbol{\theta}} + \tilde{\boldsymbol{\theta}}^T \Gamma^{-1} \dot{\tilde{\boldsymbol{\theta}}}).$$

Using the property that Γ, and therefore Γ^{-1}, is symmetric,

$$\dot{V} = \boldsymbol{\delta}^T H \dot{\boldsymbol{\delta}} + \frac{1}{2}\boldsymbol{\delta}^T \dot{H} \boldsymbol{\delta} + \tilde{\boldsymbol{\theta}}^T \Gamma^{-1} \dot{\tilde{\boldsymbol{\theta}}}.$$

Rearranging (4.9) and substituting into the previous equation, and noting that since $\boldsymbol{\theta}$ is constant, $\dot{\tilde{\boldsymbol{\theta}}} = \dot{\hat{\boldsymbol{\theta}}}$,

$$\dot{V} = -\boldsymbol{\delta}^T (C + K_d)\,\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T \dot{H} \boldsymbol{\delta} + \boldsymbol{\delta}^T \boldsymbol{\varphi}_r \tilde{\boldsymbol{\theta}} + \tilde{\boldsymbol{\theta}}^T \Gamma^{-1} \dot{\hat{\boldsymbol{\theta}}}.$$

Substituting (4.8) into the previous equation, as well as rearranging and applying Property 2 as before,

$$\begin{aligned} \dot{V} &= -\boldsymbol{\delta}^T K_d \boldsymbol{\delta} + \boldsymbol{\delta}^T \boldsymbol{\varphi}_r \tilde{\boldsymbol{\theta}} - \tilde{\boldsymbol{\theta}}^T \boldsymbol{\varphi}_r^T \boldsymbol{\delta}, \\ \dot{V} &= -\boldsymbol{\delta}^T K_d \boldsymbol{\delta} < 0, \qquad \forall \boldsymbol{\delta} \neq 0. \end{aligned} \tag{4.11}$$

The result is the same as in the previous section: considering (4.10) and (4.11), and applying Barbalat's Lemma [41], it is shown that the controller error converges to zero.

Table 4.1: Cases considered in predictor design

Case	Environment Type	Sampled?	Known Parameters?
1	Linear	No	Yes
2	Linear	Yes	Yes
3	Linear	Yes	No
4	Nonlinear	Yes	No

4.3 Virtual Environment Predictor Design

A predictor is proposed in order to compensate for the computational delay and sampling of the environment. The predictor will allow the haptic device to track a continuous trajectory approximating the true trajectory of the environment without delay or sampling. The predictor will be designed for four cases, outlined in Table 4.1. All four cases involve time delay, while only the last three involve the sampling of the feedback signal. The first two cases are intended to illustrate the design of the predictor, while the last two cases demonstrate the robustness of the predictor to handle unknown parameters and nonlinearities.

4.3.1 Known Linear Virtual Environment with Delay

From this point onward all signals lacking (t) can be assumed to occur at time t, such as $x_e = x_e(t)$. The subscript d will be used to indicate a signal delayed by T_d, such as $x_{e,d} = x_e(t - T_d)$. Let $\hat{x}_e$ and $\tilde{x}_e = \hat{x}_e - x_e$ represent the predicted tool position and the difference between the predicted and actual tool position, respectively. The prediction error is defined as $\boldsymbol{e} = [\tilde{x}_e \quad \dot{\tilde{x}}_e]^T$.
The predictor is defined in (4.12) and (4.13), with predictor gain $L \in \Re^{1\times 2}$.

$$m_t\ddot{\hat{x}}_e + b_t\dot{\hat{x}}_e + k_t\hat{x}_e = F_h - \hat{F}_e + L\boldsymbol{e}_d, \tag{4.12}$$

$$\hat{F}_e = b_e\dot{\hat{x}}_e + k_e\hat{x}_e. \tag{4.13}$$

Subtracting a single degree of freedom expression of (3.29) from (4.12) results in the error dynamics as

$$m_t\ddot{\tilde{x}}_e + b_t\dot{\tilde{x}}_e + k_t\tilde{x}_e = F_e - \hat{F}_e + L\boldsymbol{e}_d. \tag{4.14}$$

Substituting (3.31) and (4.13) into (4.14) and rearranging, we have

$$m_t\ddot{\tilde{x}}_e + (b_t + b_e)\dot{\tilde{x}}_e + (k_t + k_e)\tilde{x}_e = L\boldsymbol{e}_d. \tag{4.15}$$

The state space representation of the error dynamics is given in (4.16).

$$\dot{\boldsymbol{e}} = \begin{bmatrix} 0 & 1 \\ -\frac{k_t+k_e}{m_t} & -\frac{b+b_e}{m_t} \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} 0 \\ \frac{1}{m_t} \end{bmatrix} L\boldsymbol{e}_d = A\boldsymbol{e} + BL\boldsymbol{e}_d. \tag{4.16}$$

Theorem 2. *For the given nonzero scalars a_2, a_3, and T_d, if there exists symmetric positive definite matrices $\bar{P}$, $\bar{Q}$, and $\bar{R}$, as well as matrices $\bar{N}_1$, $\bar{N}_2$, $\bar{N}_3$, and Y, and a non-singular matrix X such that*

$$\begin{bmatrix} \Phi_{11} & \Phi_{21}^T & \Phi_{31}^T & T_d\bar{N}_1 \\ \Phi_{21} & \Phi_{22} & \Phi_{32}^T & T_d\bar{N}_2 \\ \Phi_{31} & \Phi_{32} & \Phi_{33} & T_d\bar{N}_3 \\ T_d\bar{N}_1^T & T_d\bar{N}_2^T & T_d\bar{N}_3^T & -T_d\bar{R} \end{bmatrix} < 0, \tag{4.17}$$

where

$$\begin{aligned} \Phi_{11} &= \bar{Q} + \bar{N}_1 + \bar{N}_1^T - AX^T - XA^T, \\ \Phi_{21} &= \bar{N}_2 - a_2AX^T - \bar{N}_1^T - Y^TB^T, \\ \Phi_{22} &= -\bar{Q} - \bar{N}_2 - \bar{N}_2^T - a_2BY - a_2Y^TB^T, \\ \Phi_{31} &= \bar{P} + \bar{N}_3 - a_3AX^T + X, \\ \Phi_{32} &= -\bar{N}_3 - a_3BY + a_2X, \\ \Phi_{33} &= a_3X^T + a_3X + T_d\bar{R}, \end{aligned}$$

then the system given in (4.16) *is stable for $L = Y(X^T)^{-1}$ and the error, $\boldsymbol{e}$, is regulated to zero.*

Proof. A proof of stability for a similar system is given in [25]. Selecting the Lyapunov function given in (4.18), where $P, R, Q \in \Re^{2\times 2}$ are symmetric positive definite matrices.

$$V = \boldsymbol{e}^TP\boldsymbol{e} + \int_{t-T_d}^{t} \boldsymbol{e}^T(s)Q\boldsymbol{e}(s)ds + \int_{-T_d}^{0}\int_{t+\delta}^{t} \dot{\boldsymbol{e}}^T(s)R\dot{\boldsymbol{e}}(s)dsd\delta \geq 0. \tag{4.18}$$

Taking the time derivative of (4.18) results in (4.19).

$$\dot{V} = \dot{\boldsymbol{e}}^TP\boldsymbol{e} + \boldsymbol{e}^TP\dot{\boldsymbol{e}} + \boldsymbol{e}^TQ\boldsymbol{e} - \boldsymbol{e}_d^TQ\boldsymbol{e}_d + T_d\dot{\boldsymbol{e}}^TR\dot{\boldsymbol{e}} - \int_{t-T_d}^{t} \dot{\boldsymbol{e}}^T(s)R\dot{\boldsymbol{e}}(s)ds. \tag{4.19}$$

Since the control gain does not appear in (4.19), it will be included using the zero equations (4.20) and (4.21), where $M_i, N_i \in \Re^{2\times 2}$.

$$\psi_0 = 2\left[\boldsymbol{e}^T N_1 + \boldsymbol{e}_d^T N_2 + \dot{\boldsymbol{e}}^T N_3\right]\left[\boldsymbol{e} - \boldsymbol{e}_d - \int_{t-T_d}^{t} \dot{\boldsymbol{e}}(s)ds\right], \tag{4.20}$$

$$\psi_1 = 2\left[\boldsymbol{e}^T M_1 + \boldsymbol{e}_d^T M_2 + \dot{\boldsymbol{e}}^T M_3\right]\left[\dot{\boldsymbol{e}} - A\boldsymbol{e} - BL\boldsymbol{e}_d\right]. \tag{4.21}$$

To simplify (4.20) and (4.21), defining the augmented state variable $\boldsymbol{z} = \begin{bmatrix}\boldsymbol{e} & \boldsymbol{e}_d & \dot{\boldsymbol{e}}\end{bmatrix}^T$ and the following matrices:

$$N = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix}, \qquad M = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix}.$$

Now adding (4.20) and (4.21) to (4.19).

$$\dot{V} = \dot{\boldsymbol{e}}^T P\boldsymbol{e} + \boldsymbol{e}^T P\dot{\boldsymbol{e}} + \boldsymbol{e}^T Q\boldsymbol{e} - \boldsymbol{e}_d^T Q\boldsymbol{e}_d + T_d\dot{\boldsymbol{e}}^T R\dot{\boldsymbol{e}} - \int_{t-T_d}^{t} \dot{\boldsymbol{e}}^T(s)R\dot{\boldsymbol{e}}(s)ds$$
$$+ 2\boldsymbol{z}^T N\left[\boldsymbol{e} - \boldsymbol{e}_d - \int_{t-T_d}^{t} \dot{\boldsymbol{e}}(s)ds\right] + 2\boldsymbol{z}^T M\left[\dot{\boldsymbol{e}} - A\boldsymbol{e} - BL\boldsymbol{e}_d\right]$$

The inequality (4.22) can be used to cancel the integral term.

$$-2\boldsymbol{z}N\int_{t-T_d}^{t} \dot{\boldsymbol{e}}(s)ds \le T_d\boldsymbol{z}^T NR^{-1}N^T\boldsymbol{z} + \int_{t-T_d}^{t} \dot{\boldsymbol{e}}^T(s)R\dot{\boldsymbol{e}}(s)ds. \tag{4.22}$$

This results in the inequality given in (4.23).

$$\dot{V} \le \boldsymbol{z}^T\left(\begin{bmatrix}\Xi_{11} & \Xi_{21}^T & \Xi_{31}^T \\ \Xi_{21} & \Xi_{22} & \Xi_{32}^T \\ \Xi_{31} & \Xi_{32} & \Xi_{33}\end{bmatrix} + T_d N R^{-1} N^T\right)\boldsymbol{z} = \boldsymbol{z}^T\Psi\boldsymbol{z} \tag{4.23}$$

$$\Xi_{11} = Q + N_1 + N_1^T - M_1A - A^T M_1^T$$
$$\Xi_{21} = N_2 - M_2A - N_1^T - L^T B^T M_1^T$$
$$\Xi_{22} = -Q - N_2 - N_2^T - M_2BL - L^T B^T M_2^T$$
$$\Xi_{31} = P + N_3 - M_3A + M_1^T$$
$$\Xi_{32} = -N_3 - M_3BL + M_2^T$$
$$\Xi_{33} = M_3 + M_3^T + T_dR$$

If $\Psi < 0$ then $\dot{V} < 0$, $\forall \boldsymbol{z} \neq 0$. Using the Schur complement, $\Psi < 0$ is equivalent to the following matrix inequality,

$$\begin{bmatrix} \Xi_{11} & \Xi_{21}^T & \Xi_{31}^T & T_d N_1 \\ \Xi_{21} & \Xi_{22} & \Xi_{32}^T & T_d N_2 \\ \Xi_{31} & \Xi_{32} & \Xi_{33} & T_d N_3 \\ T_d N_1^T & T_d N_2^T & T_d N_3^T & -T_d R \end{bmatrix} < 0 \tag{4.24}$$

The matrix inequality given in (4.24) is nonconvex with respect to the decision variables M_1, M_2, M_3, and L, therefore it must be linearized to solve using LMI techniques. First M_2 and M_3 are expressed as scalar multiples of M_1 (i.e. $M_2 = a_2 M_1$ and $M_3 = a_3 M_1$, where $a_i \in \Re$). Second, (4.24) is pre- and post-multiplied by $W = \text{diag}(X, X, X, X)$ and W^T, respectively, where $X = M_1^{-1}$. Third, let $Y = LX^T$ and $\bar{(\cdot)} = X(\cdot)X^T$. The final LMI is given by (4.17).
If the LMI given by (4.17) is satisfied, then $\dot{V} < 0$, $\forall \boldsymbol{z} \neq 0$. Therefore, considering (4.17), the system is stable. □

4.3.2 Known Linear Virtual Environment with Delay and Sampling

By sampling the environment position signal, the continuous signal $x_{e,d}$ is no longer available to the predictor. Instead, the predictor uses the sampled signal, resulting from a zero-order hold (ZOH) applied to $x_{e,d}$. The subscript z will be used to denote signals that have been sampled and then run through a ZOH. In order to reduce chattering in the feedback error signal, the estimated states are sampled and delayed before being subtracted with $x_{e,zd}$. Therefore the feedback error is

$$\boldsymbol{e}_{zd} = \begin{bmatrix} \hat{x}_{e,zd} - x_{e,zd} \\ \dot{\hat{x}}_{e,zd} - \dot{x}_{e,zd} \end{bmatrix}.$$

The new predictor is given in (4.25) and (4.26).

$$m_t \ddot{\hat{x}}_e + b_t \dot{\hat{x}}_e + k_t \hat{x}_e = F_h - \hat{F}_e + L\boldsymbol{e}_{zd}, \tag{4.25}$$

$$\hat{F}_e = b_e \dot{\hat{x}}_e + k_e \hat{x}_e. \tag{4.26}$$

The error dynamics of this new predictor are similar to (4.15), with an additional bounded disturbance term Δ_T so that $\boldsymbol{e}_z = \boldsymbol{e} + \Delta_T$:

$$\Delta_T = \begin{bmatrix} \hat{x}_{e,z} - \hat{x}_e \\ \dot{\hat{x}}_{e,z} - \dot{\hat{x}}_e \end{bmatrix} - \begin{bmatrix} x_{e,z} - x_e \\ \dot{x}_{e,z} - \dot{x}_e \end{bmatrix}. \tag{4.27}$$

Remark. Since the sampled states are constant for the sampling interval from t to $t + T_d$, the disturbance due to sampling is bounded by the maximum rate of change of the actual and predicted states as

$$|\Delta_T| \leq T_d \begin{bmatrix} |\dot{x}_e|_{max} + |\dot{\hat{x}}_e|_{max} \\ |\ddot{x}_e|_{max} + |\ddot{\hat{x}}_e|_{max} \end{bmatrix}, \tag{4.28}$$

where $|\cdot|$ represents the absolute value applied to each element

Therefore the new error dynamics are expressed in (4.29),

$$m_t\ddot{\tilde{x}}_e + (b_t + b_e)\dot{\tilde{x}}_e + (k_t + k_e)\tilde{x}_e = L\boldsymbol{e}_d - L\Delta_{T,d}. \tag{4.29}$$

The state space representation of the error dynamics is given in (4.30),

$$\dot{\boldsymbol{e}} = A\boldsymbol{e} + BL\boldsymbol{e}_d + BL\Delta_{T,d}. \tag{4.30}$$

Theorem 3. *For the given nonzero scalars a_2, a_3, γ, and T_d, if there exists symmetric positive definite matrices $\bar{P}$, $\bar{Q}$, $\bar{R}$, and $\bar{S}$, as well as matrices $\bar{N}_1$, $\bar{N}_2$, $\bar{N}_3$, and Y, and a non-singular matrix X such that*

$$\begin{bmatrix} \Phi_{11} & \Phi_{21}^T & \Phi_{31}^T & T_d\bar{N}_1 & \gamma I \\ \Phi_{21} & \Phi_{22} & \Phi_{32}^T & T_d\bar{N}_2 & \gamma a_2 I \\ \Phi_{31} & \Phi_{32} & \Phi_{33} & T_d\bar{N}_3 & \gamma a_3 I \\ T_d\bar{N}_1^T & T_d\bar{N}_2^T & T_d\bar{N}_3^T & -T_d\bar{R} & 0 \\ \gamma I & \gamma a_2 I & \gamma a_3 I & 0 & -\gamma I \end{bmatrix} < 0, \tag{4.31}$$

where

$$\begin{aligned} \Phi_{11} &= \bar{Q} + \bar{N}_1 + \bar{N}_1^T - AX^T - XA^T + \bar{S}, \\ \Phi_{21} &= \bar{N}_2 - a_2AX^T - \bar{N}_1^T - Y^TB^T, \\ \Phi_{22} &= -\bar{Q} - \bar{N}_2 - \bar{N}_2^T - a_2BY - a_2Y^TB^T, \\ \Phi_{31} &= \bar{P} + \bar{N}_3 - a_3AX^T + X, \\ \Phi_{32} &= -\bar{N}_3 - a_3BY + a_2X, \\ \Phi_{33} &= a_3X^T + a_3X + T_d\bar{R}, \end{aligned}$$

then the system given in (4.30) *is stable for $L = Y(X^T)^{-1}$ and the error is bounded by*

$$\|\boldsymbol{e}\|^2 \leq \gamma^{-1}\frac{\lambda_{max}(L^TB^TBL)}{\lambda_{min}(S)}\|\Delta_T\|^2. \tag{4.32}$$

Proof. The derivation of the Lyapunov function and its derivative follows the same procedure as the proof for Theorem 2, with the only differences being the addition of the disturbance term to ψ_1, and the addition of a third zero function ψ_2.

$$\psi_1 = 2\boldsymbol{z}^T M \left[\dot{\boldsymbol{e}} - A\boldsymbol{e} - BL\boldsymbol{e}_d - BL\Delta_{T,d}\right]$$
$$\psi_2 = \boldsymbol{e}^T S\boldsymbol{e} - \boldsymbol{e}^T S\boldsymbol{e} \qquad S = S^T > 0$$

The disturbance can be isolated using the inequality given by (4.33), where the parameter $\gamma \in \Re^+$ specifies the disturbance rejection.

$$-2\boldsymbol{z}^T MBL\Delta_{T,d} \leq \gamma \boldsymbol{z}^T MM^T\boldsymbol{z} + \gamma^{-1}\Delta_{T,d}^T L^T B^T BL\Delta_{T,d} \tag{4.33}$$

These changes result in the following inequality for $\dot{V}$, where the elements of Ξ are the same as in the proof of Theorem 2, with the exception of Ξ_{11}.

$$\dot{V} \leq \boldsymbol{z}^T \left(\Xi + T_d N R^{-1} N^T + \gamma MM^T\right)\boldsymbol{z} + \gamma^{-1}\Delta_{T,d}^T L^T B^T BL\Delta_{T,d} - \boldsymbol{e}^T S\boldsymbol{e} \tag{4.34}$$

$$\Xi_{11} = Q + N_1 + N_1^T - M_1 A - A^T M_1^T + S$$

Using the Shur complement, $\Psi = \Xi + T_d N R^{-1} N^T + \gamma MM^T < 0$ is equivalent to the matrix inequality expressed in (4.35).

$$\begin{bmatrix} \Xi_{11} & \Xi_{21}^T & \Xi_{31}^T & T_d N_1 & \gamma M_1 \\ \Xi_{21} & \Xi_{22} & \Xi_{32}^T & T_d N_2 & \gamma M_2 \\ \Xi_{31} & \Xi_{32} & \Xi_{33} & T_d N_3 & \gamma M_3 \\ T_d N_1^T & T_d N_2^T & T_d N_3^T & -T_d R & 0 \\ \gamma M_1^T & \gamma M_2^T & \gamma M_3^T & 0 & -\gamma I \end{bmatrix} < 0 \tag{4.35}$$

Linearizing using the same method as in the previous section, but extending the definition of W to $W = \mathrm{diag}(X, X, X, X, I)$, results in the LMI expressed in (4.31). The elements of Φ are the same as in the previous section, with the exception of Φ_{11}. If (4.31) is satisfied, then inequality (4.36) holds.

$$\dot{V} \leq \gamma^{-1}\Delta_{T,d}^T L^T B^T BL\Delta_{T,d} - \boldsymbol{e}^T S\boldsymbol{e} \tag{4.36}$$

Therefore, $\dot{V}$ is only guaranteed to be less than zero if

$$\boldsymbol{e}^T S\boldsymbol{e} \geq \gamma^{-1}\Delta_{T,d}^T L^T B^T BL\Delta_{T,d}. \tag{4.37}$$

Therefore the magnitude of the error must be bounded by

$$\|\boldsymbol{e}\|^2 \leq \gamma^{-1} \frac{\lambda_{max}(L^T B^T B L)}{\lambda_{min}(S)} \|\Delta_T\|^2,$$

where $\lambda_{max}(\cdot)$ and $\lambda_{min}(\cdot)$ are the maximum and minimum eigenvalues of a matrix, respectively. Therefore, if the environment state signal is stable, the predictor system is stable. □

4.3.3 Unknown Linear Virtual Environment with Delay and Sampling

This case differs from the previous case in that the parameters of the virtual environment are unknown but their bounds are known. The virtual environment force given in (3.31) can be expressed parametrically by (3.32).

Assumption: The boundedness of the parameter variation is known. This is expressed as

$$\boldsymbol{\theta}_e \in \Omega_{\theta_e} \triangleq \{\boldsymbol{\theta}_e : \boldsymbol{\theta}_{e,min} \leq \boldsymbol{\theta}_e \leq \boldsymbol{\theta}_{e,max}\} \tag{4.38}$$

where $\boldsymbol{\theta}_{e,min}$ and $\boldsymbol{\theta}_{e,max}$ are known constant vectors.

The predictor in this case differs from Case 2 in that the estimated virtual environment force is now calculated using an estimate of the virtual environment parameters $\hat{\boldsymbol{\theta}}_e$.

$$\hat{F}_e = \hat{\boldsymbol{\varphi}}_e^T \hat{\boldsymbol{\theta}}_e(t). \tag{4.39}$$

This changes the error dynamics to

$$m_t \ddot{\tilde{x}}_e(t) + b_t \dot{\tilde{x}}_e(t) + k_t \tilde{x}_e(t) = \boldsymbol{\varphi}_e^T \boldsymbol{\theta}_e - \hat{\boldsymbol{\varphi}}_e^T \hat{\boldsymbol{\theta}}_e + L\boldsymbol{e}_d - L\Delta_{T,d}. \tag{4.40}$$

By adding the zero expression $\hat{\boldsymbol{\varphi}}_e^T \boldsymbol{\theta}_e - \hat{\boldsymbol{\varphi}}_e^T \boldsymbol{\theta}_e$, (4.40) can be expressed as

$$m_t \ddot{\tilde{x}}_e(t) + (b_t + b_e) \dot{\tilde{x}}_e(t) + (k_t + k_e) \tilde{x}_e(t) = L(\boldsymbol{e}_d - \Delta_{T,d}) - \hat{\boldsymbol{\varphi}}_e^T \tilde{\boldsymbol{\theta}}_e, \tag{4.41}$$

where $\tilde{\boldsymbol{\theta}}_e = \hat{\boldsymbol{\theta}}_e - \boldsymbol{\theta}_e$.

The error dynamics can be expressed in a state space form. The parameter estimation error can be contained in a second disturbance term: $\Delta_\theta = \hat{\boldsymbol{\varphi}}_e^T \tilde{\boldsymbol{\theta}}_e$. The resulting error dynamics are then,

$$\dot{\boldsymbol{e}} = A\boldsymbol{e} + BL\boldsymbol{e}_d - B(L\Delta_{T,d} - \Delta_\theta). \tag{4.42}$$

Remark: The boundedness of the disturbance term Δ_θ is dependent on the boundedness of $\hat{\boldsymbol{\varphi}}$ and $\tilde{\boldsymbol{\theta}}_e$. We can determine the bounds of $\hat{\boldsymbol{\varphi}}_e$ based on the constraints of the

workspace, but the bounds of $\tilde{\boldsymbol{\theta}}_e$ cannot be determined without an assumption. If we assume that the estimate $\hat{\boldsymbol{\theta}}_e$ is constrained to the known parameter range Ω_{θ_e} (a reasonable assumption, see the next subsection) then $|\tilde{\boldsymbol{\theta}}_e| \leq \boldsymbol{\theta}_{e,max} - \boldsymbol{\theta}_{e,min}$. Therefore the bounds of Δ_θ can be estimated by

$$\|\Delta_\theta\| \leq |\boldsymbol{\varphi}_e|_{max} \left(\boldsymbol{\theta}_{e,max} - \boldsymbol{\theta}_{e,min}\right)$$

The predictor gain may be designed using Theorem 3, except that the disturbance term $L\Delta_{T,d}$ must be replaced by $\Delta_{tot} = L\Delta_{T,d} + \Delta_\theta$. This means the boundary on the prediction error will be larger when $\Delta_\theta \neq 0$. Also, since the actual environmental parameters are no longer known, A cannot be used in the predictor gain design. Instead, the gain must be designed for a nominal value of environment parameters $\boldsymbol{\theta}_{e,0}$. Once the gain is designed for the nominal set of virtual environment parameters it must be verified that it results in a stable system for any set of parameters in the parameter range. The system is guaranteed to be stable if there is a solution to (4.31) for each set of possible virtual environment parameters on a sufficiently dense grid over the range of $\boldsymbol{\theta}_e$ for a fixed value of L.

Projection Type Adaptation Law Design

The estimated parameter vector will be generated online using a projection type least square adaptation law. Let the adaptation error be defined by (4.43),

$$\epsilon = \boldsymbol{\varphi}_{e,zd}^T \hat{\boldsymbol{\theta}}_e - F_{e,zd}. \tag{4.43}$$

where $\boldsymbol{\varphi}_{e,zd} = [x_{e,zd} \quad \dot{x}_{e,zd}]^T$. Since $F_{e,zd} = \boldsymbol{\varphi}_{e,zd}^T \boldsymbol{\theta}_e$, the adaptation error can be expressed as

$$\epsilon = \boldsymbol{\varphi}_{e,zd}^T \tilde{\boldsymbol{\theta}}_e.$$

The parameter estimate is updated by

$$\dot{\hat{\boldsymbol{\theta}}}_e = \mathrm{Proj}_{\theta_e} \left(-\Gamma_e \boldsymbol{\zeta}_e\right), \qquad \hat{\boldsymbol{\theta}}_e(0) \in \Omega_{\theta_e}, \tag{4.44}$$

where Γ_e is a positive definite matrix and $\boldsymbol{\zeta}_e$ is an adaptation function, forming a recursive least square adaptation law given by

$$\dot{\Gamma}_e = \begin{cases} \alpha\Gamma_e - \dfrac{1}{1 + \nu\boldsymbol{\varphi}_e^T\Gamma_e\boldsymbol{\varphi}_e}\Gamma_e\boldsymbol{\varphi}_e\boldsymbol{\varphi}_e^T\Gamma_e, & \text{if } \lambda_{max}(\Gamma_e(t)) \leq \rho_M, \\ 0 & \text{otherwise,} \end{cases} \tag{4.45}$$

and,

$$\boldsymbol{\zeta}_e = \frac{1}{1+\nu\boldsymbol{\varphi}_e^T\Gamma_e\boldsymbol{\varphi}_e}\boldsymbol{\varphi}_e\epsilon, \tag{4.46}$$

where $\alpha \geq 0$ is the forgetting factor, $\nu \geq 0$, and ρ_M is a predetermined upper limit on Γ_e to prevent estimator windup.

The projection mapping ensures that the parameter estimate given by the adaptation law remains within the known boundary [44], and is defined by

$$\text{Proj}_{\hat{\theta}_e}(\bullet) = \begin{cases} \bullet, & \text{if } \hat{\boldsymbol{\theta}}_e \in \mathring{\Omega}_{\theta_e} \text{ or } n_{\hat{\theta}_e}^T \bullet \leq 0, \\ \left(I - \Gamma_e \dfrac{n_{\hat{\theta}_e} n_{\hat{\theta}_e}^T}{n_{\hat{\theta}_e}^T \Gamma_e n_{\hat{\theta}_e}}\right)\bullet & \text{if } \hat{\boldsymbol{\theta}}_e \in \partial\Omega_{\theta_e} \text{ and } n_{\hat{\theta}_e}^T \bullet > 0, \end{cases} \tag{4.47}$$

where $\mathring{\Omega}_{\theta_e}$ is the interior of Ω_{θ_e}, $\partial\Omega_{\theta_e}$ is the boundary of Ω_{θ_e}, and $n_{\hat{\theta}_e}$ is the outward unit normal vector of the boundary at $\hat{\boldsymbol{\theta}}_e$.

Lemma 4. [44] *When the adaptation law* (4.44) *is used with the projection mapping* (4.47) *the parameter estimate is always within the known boundary. Also, if the following persistent excitation condition is met*

$$\int_t^{t+T_1} \boldsymbol{\varphi}_{e,zd}^T \boldsymbol{\varphi}_{e,zd} dt > \beta I, \ \forall t > t_0, \tag{4.48}$$

for some $T_1 > 0$ *and* $\beta > 0$*, then* $\hat{\boldsymbol{\theta}}_e$ *converges to* $\boldsymbol{\theta}_e$*.*

4.4 Linear Parameter Varying System Design

For a nonlinear virtual environment the dynamics may be approximated by a linear time-varying system given in (3.31).

Although virtual environment parameters are time-varying, Theorem 3 could still be used to design the predictor gain just like in the unknown parameters case. However, rather than simply finding a solution to (4.31) for each point in the parameter range for a given L to guarantee stability, instead a single set of decision variables must be found which satisfies (4.31) over the entire parameter range [26].

Clearly finding a single value for L and the other decision variables for the entire parameter range is an onerous requirement. It would be advantageous to define a predictor gain and set of decision variables that adjust according to the current set of

parameters. To design such a predictor, the VE and predictor system can be analyzed using a Linear Parameter Varying (LPV) system approach [26].

The error dynamics are the same as (4.40), except that both $\boldsymbol{\theta}_e$ and $\hat{\boldsymbol{\theta}}_e$ are time-varying, and L is now a function of the scheduling variables $\boldsymbol{\rho} = [\rho_1 \quad \rho_2]^T = [\hat{k}_e(t) \quad \hat{b}_e(t)]^T$. Adding the zero expression $\boldsymbol{\varphi}_e^T\boldsymbol{\theta}_e - \boldsymbol{\varphi}_e^T\boldsymbol{\theta}_e$ to (4.40) results in

$$\dot{\boldsymbol{e}} = \hat{A}\boldsymbol{e} + BL(\boldsymbol{\rho})\boldsymbol{e}_d - B(L(\boldsymbol{\rho})\Delta_{T,d} - \boldsymbol{\varphi}_e^T\tilde{\boldsymbol{\theta}}_e), \tag{4.49}$$

where $\hat{A}$ is the state matrix containing estimates of the virtual environment parameters. $\hat{A}$ can be expressed using an LPV approach as

$$\hat{A}(\boldsymbol{\rho}) = \begin{bmatrix} 0 & 1 \\ -\frac{k_t+\rho_1}{m_t} & -\frac{b_t+\rho_2}{m_t} \end{bmatrix}.$$

The structure of this particular LPV system has the useful property that the parameter dependence can be expressed as a linear combination of matrices:

$$\hat{A}(\boldsymbol{\rho}) = A_0 + A_1\rho_1 + A_2\rho_2, \tag{4.50}$$

where

$$A_0 = \begin{bmatrix} 0 & 1 \\ -\frac{k_t}{m_t} & -\frac{b_t}{m_t} \end{bmatrix}, \qquad A_1 = \begin{bmatrix} 0 & 0 \\ -\frac{1}{m_t} & 0 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 0 & 0 \\ 0 & -\frac{1}{m_t} \end{bmatrix}.$$

Theorem 5. *For the given nonzero scalars a_2, a_3, γ, and T_d, if there exists symmetric positive definite matrices $\bar{P}(\boldsymbol{\rho})$, $\bar{Q}$, $\bar{R}$, and $\bar{S}$, as well as matrices $\bar{N}_1$, $\bar{N}_2$, $\bar{N}_3$, and $Y(\boldsymbol{\rho})$, and a non-singular matrix X such that*

$$\begin{bmatrix} \Phi_{11} & \Phi_{21}^T & \Phi_{31}^T & T_d\bar{N}_1 & \gamma I \\ \Phi_{21} & \Phi_{22} & \Phi_{32}^T & T_d\bar{N}_2 & \gamma a_2 I \\ \Phi_{31} & \Phi_{32} & \Phi_{33} & T_d\bar{N}_3 & \gamma a_3 I \\ T_d\bar{N}_1^T & T_d\bar{N}_2^T & T_d\bar{N}_3^T & -T_d\bar{R} & 0 \\ \gamma I & \gamma a_2 I & \gamma a_3 I & 0 & -\gamma I \end{bmatrix} < 0, \tag{4.51}$$

where

$$\begin{aligned}
\Phi_{11} &= \bar{Q} + \bar{N}_1 + \bar{N}_1^T - \hat{A}(\boldsymbol{\rho})X^T - X\hat{A}(\boldsymbol{\rho})^T + \bar{S} + \sum \pm|\dot{\rho}_i|_{max}\frac{\partial \bar{P}(\boldsymbol{\rho})}{\partial \rho_i},\\
\Phi_{21} &= \bar{N}_2 - a_2\hat{A}(\boldsymbol{\rho})X^T - \bar{N}_1^T - Y(\boldsymbol{\rho})^T B^T,\\
\Phi_{22} &= -\bar{Q} - \bar{N}_2 - \bar{N}_2^T - a_2 BY(\boldsymbol{\rho}) - a_2 Y(\boldsymbol{\rho})^T B^T,\\
\Phi_{31} &= \bar{P}(\boldsymbol{\rho}) + \bar{N}_3 - a_3\hat{A}(\boldsymbol{\rho})X^T + X,\\
\Phi_{32} &= -\bar{N}_3 - a_3 BY(\boldsymbol{\rho}) + a_2 X,\\
\Phi_{33} &= a_3 X^T + a_3 X + T_d\bar{R},
\end{aligned}$$

over the parameter range $\boldsymbol{\rho}_{min} \leq \boldsymbol{\rho} \leq \boldsymbol{\rho}_{max}$*, then the system given in* (4.49) *is stable for* $L(\boldsymbol{\rho}) = Y(\boldsymbol{\rho})(X^T)^{-1}$ *and the error is bounded by*

$$\|\boldsymbol{e}\|^2 \leq \gamma^{-1}\left(\frac{\lambda_{max}(L(\boldsymbol{\rho})^T B^T B L(\boldsymbol{\rho}))}{\lambda_{min}(S)}\|\Delta_T\|^2 + \frac{B^T B}{\lambda_{min}(S)}\|\Delta_\theta\|^2\right). \tag{4.52}$$

Proof. A new Lyapunov function can be constructed, similar to (4.18), except that the matrix P is now a function of the parameters, as $P(\boldsymbol{\rho})$.

$$V = \boldsymbol{e}^T P(\boldsymbol{\rho})\boldsymbol{e} + \int_{t-T_d}^{t} \boldsymbol{e}^T(s)Q\boldsymbol{e}(s)ds + \int_{-T_d}^{0}\int_{t+\delta}^{t} \dot{\boldsymbol{e}}^T(s)R\dot{\boldsymbol{e}}(s)dsd\delta. \tag{4.53}$$

The derivation of the LMI proceeds as in the proof for Theorem 3, except that an additional term appears during the differentiation of V due to the inclusion of the time-varying parameters $\boldsymbol{\rho}$ in $P(\boldsymbol{\rho})$.

$$\begin{aligned}
\frac{d}{dt}\left(\boldsymbol{e}^T P(\boldsymbol{\rho})\boldsymbol{e}\right) &= \dot{\boldsymbol{e}}^T P(\boldsymbol{\rho})\boldsymbol{e} + \boldsymbol{e}^T\left(\frac{dP(\boldsymbol{\rho})}{dt}\boldsymbol{e} + P(\boldsymbol{\rho})\dot{\boldsymbol{e}}\right)\\
&= \dot{\boldsymbol{e}}^T P(\boldsymbol{\rho})\boldsymbol{e} + \boldsymbol{e}^T P(\boldsymbol{\rho})\dot{\boldsymbol{e}} + \boldsymbol{e}^T\sum \dot{\rho}_i\frac{\partial P(\boldsymbol{\rho})}{\partial \rho_i}\boldsymbol{e}.
\end{aligned}$$

Continuing the analysis results in a matrix inequality similar to (4.35), except that P is replaced by $P(\boldsymbol{\rho})$ and the term Ξ_{11} includes the rate of change of $P(\boldsymbol{\rho})$. Performing the same linearizing operations as before results in the final LMI given by (4.51). Only the extreme values of $\dot{\boldsymbol{\rho}}$ (i.e. $|\dot{\rho}_i|_{max}$) need to be tested since the LMI is affine in $\dot{\boldsymbol{\rho}}$.

The error bound is derived by the same process as in the proof for Theorem 3. □

4.4.1 Gain-Scheduled Predictor Feedback Gain Design

In order to obtain a solution to (4.51), the dependence of P and Y on $\boldsymbol{\rho}$ must be specified. Restricting P and Y to a specific function of $\boldsymbol{\rho}$ reduces the potential solutions to the LMI, but is necessary to allow it to be solved. A linear function of the elements of $\boldsymbol{\rho}$ is selected since this mirrors the state matrix $A(\boldsymbol{\rho})$ given in (4.50) [26].

$$P(\boldsymbol{\rho}) = P_0 + P_1\rho_1 + P_2\rho_2, \qquad Y(\boldsymbol{\rho}) = Y_0 + Y_1\rho_1 + Y_2\rho_2$$

where $P_i = P_i^T$. Note that although $P(\boldsymbol{\rho}) > 0$, P_i does not need to be positive definite. Now $\frac{\partial P(\boldsymbol{\rho})}{\partial \rho_i} = P_i$.
The predictor feedback gain designed as follows:

1. Select nominal values of $\boldsymbol{\rho} = \boldsymbol{\rho}_0$ and $|\dot{\boldsymbol{\rho}}|_{max} = \dot{\boldsymbol{\rho}}_0$ and solve the LMI for Y_0, Y_1, and Y_2. Calculate the predictor gain as:

$$L(\boldsymbol{\rho}) = (Y_0 + Y_1\rho_1 + Y_2\rho_2)(X^T)^{-1}$$

2. Using the decision variables solved from Step 1, test (4.51) on a grid of points over the range of $\boldsymbol{\rho}$ and all extreme values of $\dot{\boldsymbol{\rho}}$ in order to determine the range of $\boldsymbol{\rho}$ and $\dot{\boldsymbol{\rho}}$ over which the system is stable. For a number of parameters n_ρ with n test points per parameter, a total of $(2^{n_\rho}n)^{n_\rho}$ LMIs must be tested. When testing the LMIs each instance of $Y(\boldsymbol{\rho})$ is replaced by $L(\boldsymbol{\rho})X^T$ since L is now known. The system stability is only guaranteed for parameter values where (4.51) holds.

3. If the stability range found in Step 2 is insufficient, select new values of $\boldsymbol{\rho}_0$ and $\dot{\boldsymbol{\rho}}_0$ and return to Step 1. It is also possible to adjust the results by tuning a_2, a_3, and γ.

Chapter 5

Experimental Setup

This chapter describes the preliminary work needed to set up a full-scale test of the haptic control algorithm. This involves calibrating the Phantom Omni®, selecting a force sensor, designing and building a custom force sensor mount, and calibrating the force sensor.

5.1 Control Software

The control software used to implement the control algorithm is QUARC 2.4 by Quanser, running in MATLAB R2013b (32-bit). QUARC allows real-time control to be performed using MATLAB and Simulink. Simulink is used to design the control algorithm, then QUARC compiles the Simulink model to create an executable that is capable of running in real-time at an update rate of up to 1000 Hz.

QUARC also contains a library of custom Simulink blocks used to communicate with a variety of hardware. This includes a block for communicating with the Phantom Omni®. The block allows inputs in either the joint space (torque input) or the task space (force input). The block can output either the joint angular positions or the end-effector position, as well as the end-effector gimbal angles. The block's internal software is not documented, therefore custom kinematics software is used instead of the unknown kinematics software of the block. The block is set to accept a torque input and output the joint angles.

5.2 Hardware: The Phantom Omni® Haptic Device

Fig. 5.1 shows the Phantom Omni® haptic device used to test the predictor experimentally. The Phantom Omni® connects to the computer via a Firewire connector. The joint position encoder resolution is approximately 4×10^{-4} rad. In order to use the Phantom Omni® in the control algorithm, some adjustments had to be made.

Figure 5.1: The Phantom Omni® haptic device with a custom force sensor mount.

5.2.1 Joint Angle Conversion

The measured joint angle outputs from the Phantom Omni® QUARC block do not match the conventions used in the kinematic model of the device. The conversion from the angle output $\boldsymbol{q}_{Omni}$ to the model joint variable $\boldsymbol{q}$ is given by

$$\boldsymbol{q} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \boldsymbol{q}_{Omni} - \begin{bmatrix} 0 \\ 0 \\ \frac{\pi}{2} \end{bmatrix}. \tag{5.1}$$

The measured gimbal angles from the Phantom Omni® end-effector ($\boldsymbol{\phi}_{Omni}$) can be converted to a standard roll-pitch-yaw convention using (5.2),

$$\begin{bmatrix} \psi \\ \vartheta \\ \phi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \boldsymbol{\phi}_{Omni}, \tag{5.2}$$

where ϕ, ϑ, and ψ are the roll, pitch, and yaw of the end-effector, respectively.

5.2.2 Joint Velocity Analysis

The Phantom Omni® is not capable of measuring the joint angular velocity directly, therefore it must be calculated from the derivative of the joint positions. Since the

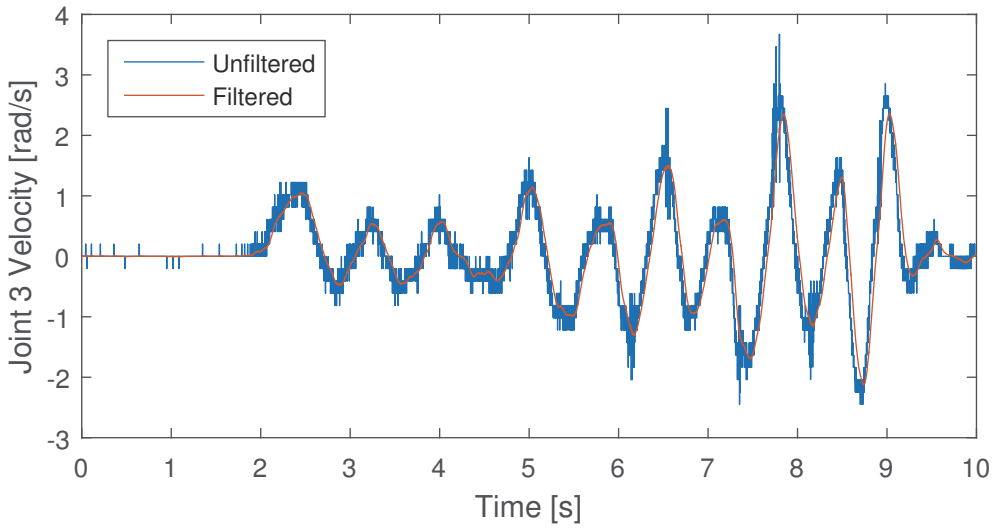


Figure 5.2: Plot of the unfiltered and filtered velocity signal calculated from position

derivative is being calculated in discrete time it introduces significant noise into the system. This noise is reduced by using a low-pass filter with a cutoff frequency of 4 Hz running at an update rate of 1000 Hz. A cutoff frequency of 4 Hz was selected as the best tradeoff between noise reduction and phase lag. The filter has the following transfer function,

$$G(s) = \frac{s}{0.0398s + 1}. \tag{5.3}$$

Fig. 5.2 shows the calculated velocity signal before and after applying the filter. The filter leads to greatly reduced chattering in the signal, while creating a delay of less than 50 ms in the velocity signal.

5.2.3 Position Limits

Due to the mechanical construction of the Phantom Omni® there are physical limits to the motion of each joint. Table 5.1 shows the limits of each of the actuated joints. Given that the limits should be avoided, the Omni is prevented from applying torque toward the joint limit when it is within 0.1 rad of the limit.

There is also a non-constant constraint. Since the third joint is actuated via cables that run from the base of the Phantom Omni®, there is a constraint on the sum of the angles of joints 2 and 3, given by (5.4).

Table 5.1: Mechanical properties of the Phantom Omni® joints

Joint	Position Limits (rad) Minimum	Maximum	Torque Limit (N·m)	Static Friction (N·m)
1	$-\frac{\pi}{3}$	$\frac{\pi}{3}$	0.30	-
2	0	1.79	0.29	0.065
3	-2.45	-0.25	0.20	0.028

$$q_2 + q_3 \leq 0.214 \text{ rad.} \tag{5.4}$$

Also, in order to prevent the end-effector from colliding with the base, the position of the end effector should be constrained to the region

$$\sqrt{x^2 + y^2} \geq 0.12 \text{ m.}$$

Finally, although not particularly important for safety, it is important to consider before performing inverse kinematics calculations that the position of the end-effector is constrained within the sphere described by

$$\sqrt{x^2 + y^2 + z^2} \leq 0.26 \text{ m.}$$

5.2.4 Torque Limits

According to the Phantom Omni® documentation, the device can exert a maximum force of $F_{max} = 3.3$ N when the links are in an orthogonal position. Therefore, a conservative estimate of the maximum torque each joint can apply would be to multiply the maximum force by the longest individual link length (l_2 in this case).

$$\tau_{max} \leq l_2 F_{max} \approx 0.446 \text{ N} \cdot \text{m.}$$

However, measurements taken using the force sensor (see Section 5.3 for the force sensor setup) show that the actual hardware torque limit is much lower. Each joint was tested by placing the force sensor perpendicular to the end of the link and sending a ramp input to the joint. The force sensor measured the actual force applied to the end of the link, which is easily converted to a joint torque. The setup for testing Joint 2 is shown in Fig. 5.3. An example of the measured output is shown in Fig. 5.4. Although the commanded torque increased linearly, the torque output had an offset and eventually stopped increasing. The torque limits for all three joints are summarized in Table 5.1.

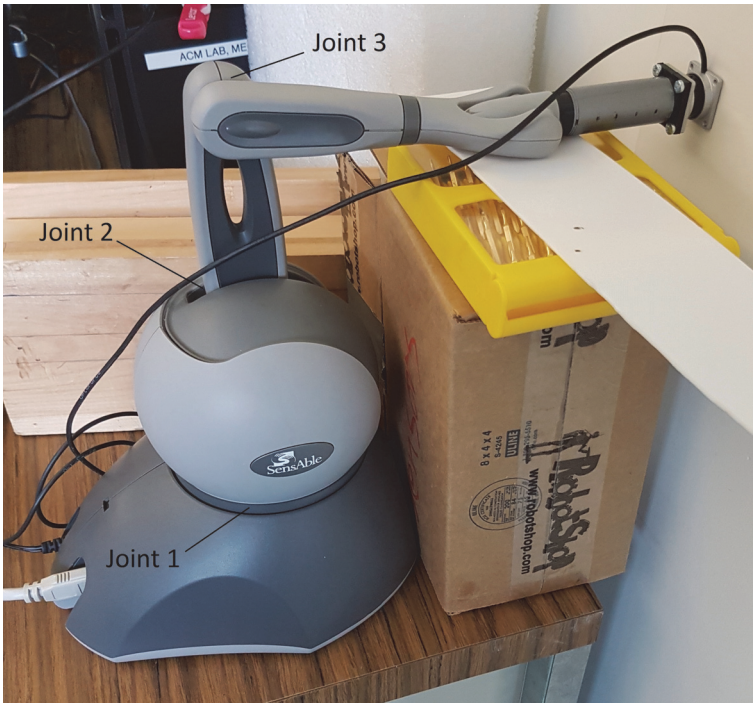


Figure 5.3: Experimental setup for measuring the output torque of Joint 2 of the Phantom Omni®.

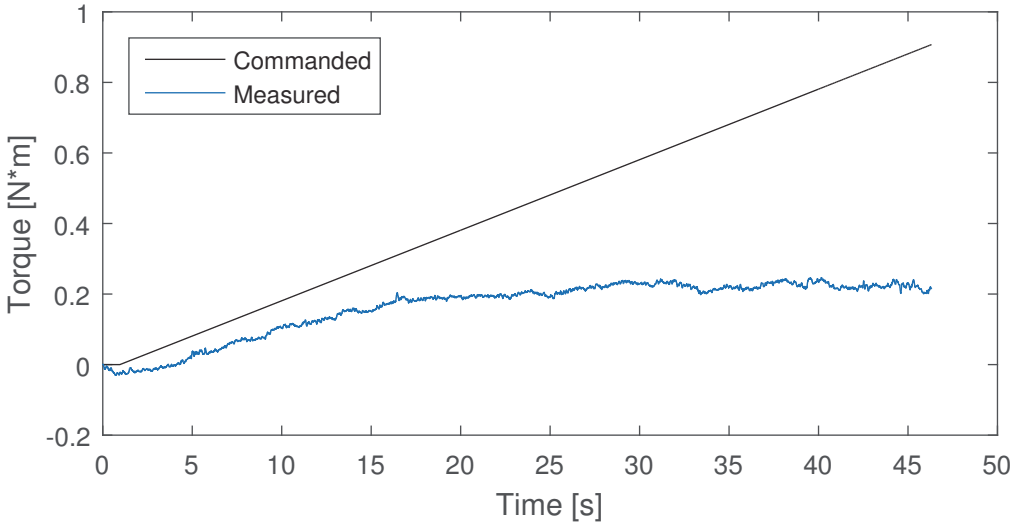


Figure 5.4: Measured torque output of Joint 3 of the Phantom Omni®

In order to ensure that the device is not damaged during testing, the applied torque τ is constrained to be less than the maximum exert-able torque by a safety factor:

$$|\tau| \leq \tau_{allow} = \frac{\tau_{max}}{SF_\tau}$$

The safety factor was set to $SF_\tau = 1.3$.

5.2.5 Torque Conversion

The slope of the measured torque in Fig. 5.4 does not match the slope of the commanded torque, therefore a conversion between the commanded and actual output torque is necessary. Applying a linear regression to the data measured during the torque limit testing results in the conversion matrix as,

$$\boldsymbol{\tau}_c = \begin{bmatrix} -1.0 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0 & 1.43 \end{bmatrix} \boldsymbol{\tau}_{command}. \tag{5.5}$$

5.2.6 Static Friction

The Phantom Omni® joints experience significant static friction. A series of increasing and decreasing ramp inputs are applied to the joints to generate a hysteresis loop. The results for Joint 2 are shown in Fig. 5.5. The static friction appears as the horizontal section of the plot, where the commanded torque changes but the measured torque does not. The static friction of each joint is given in Table 5.1, except for the static friction of Joint 1, which was not measured.

5.2.7 Adaptive Controller Tuning

The adaptive controller (4.7) was tuned experimentally. The Phantom Omni® was commanded to track a sinusoidal reference trajectory expressed by,

$$\boldsymbol{q}_d = \begin{bmatrix} \frac{\pi}{8}\cos(0.6\pi t) \\ \frac{\pi}{7}\cos(0.4\pi t) + 0.1 \\ \frac{\pi}{10}\cos(0.46\pi t) \end{bmatrix} + \boldsymbol{q}(0) \text{ rad},$$

and the control gains were tuned to minimize the joint tracking error $\boldsymbol{q} - \boldsymbol{q}_d$ while maintaining stability. The acceptable position error is less than 0.175 rad (10 deg) for each joint. The final control gains are given in (5.6),

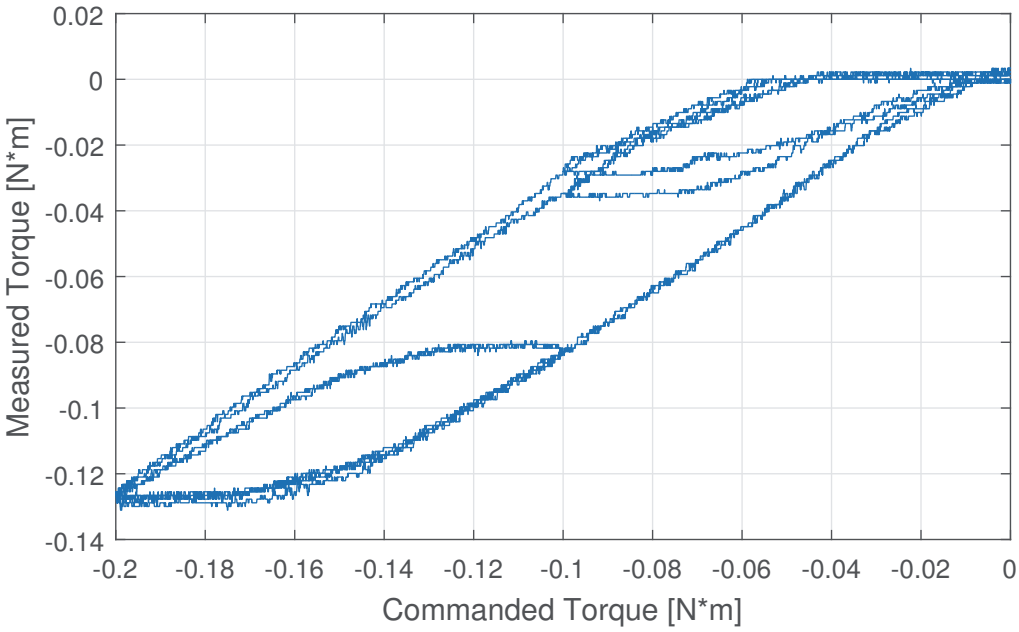


Figure 5.5: Measured versus commanded torque for Joint 2 of the Phantom Omni® for a series of increasing and decreasing ramp inputs.

$$K_d = \begin{bmatrix} 0.4 & 0 & 0 \\ 0 & 0.28 & 0 \\ 0 & 0 & 0.10 \end{bmatrix} \qquad \Lambda = \begin{bmatrix} 2.8 & 0 & 0 \\ 0 & 3.0 & 0 \\ 0 & 0 & 1.5 \end{bmatrix}, \tag{5.6}$$

and the parameter adaptation gain matrix is

$$\Gamma = \mathrm{diag}\,(0.01,\ 0.002,\ 0.002,\ 0.0005,\ 0.001,\ 0.001,\ 0.02,\ 0.02,\ 0.02)\,, \tag{5.7}$$

where diag(•) is a diagonal matrix with the main diagonal described by '•'.
The resulting joint error signals are shown in Fig. 5.6. The bounds on the position error are 0.035 rad (2.0 deg), 0.042 rad (2.4 deg), and 0.136 rad (7.8 deg) for Joint 1, Joint 2, and Joint 3, respectively.

5.2.8 Dynamic Parameter Identification

The dynamic parameters were estimated online by the adaptation law (4.8). The online estimates of the parameters are shown in Fig. 5.7. The parameters do not converge to a constant set of values due to static friction and other nonlinearities not accounted for in the controller model. An approximate set of dynamic parameters

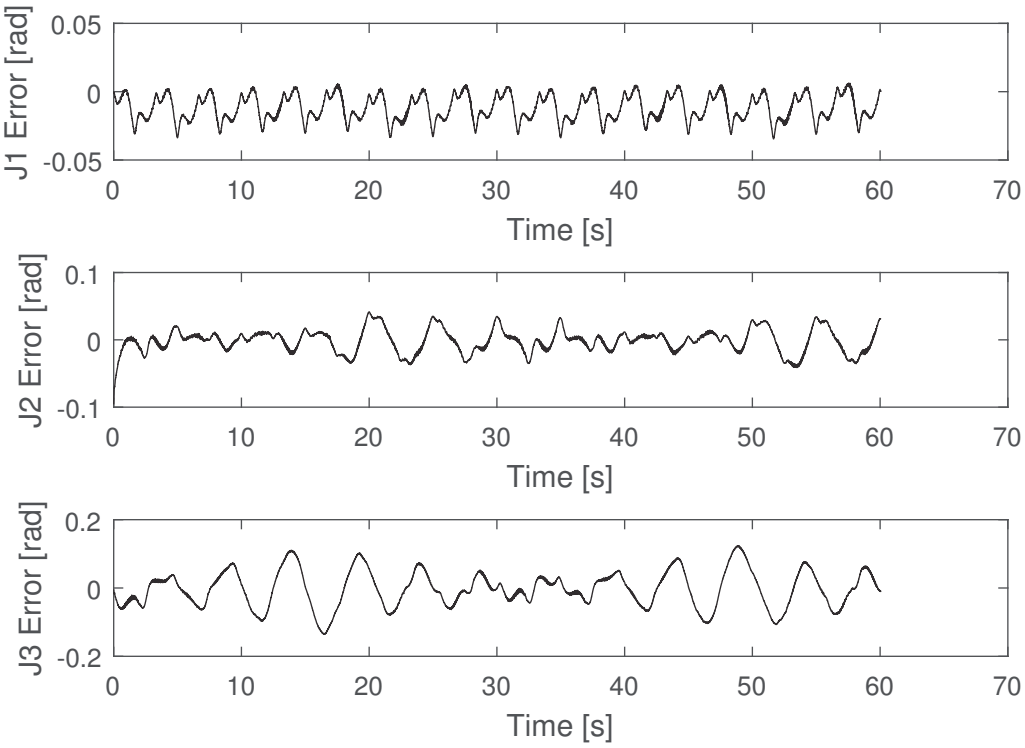


Figure 5.6: Phantom Omni® joint error signals for the tuned adaptive controller.

were selected to be used as initial estimates for the adaptation law based on the control tuning tests. The selected parameters are given in Table 5.2.

Table 5.2: Estimates of the Phantom Omni® dynamic parameters for initializing the adaptation law.

Parameter	Value	Parameter	Value
l_2 (m)	0.135	l_3 (m)	0.130
θ_1 (kg·m^2)	3.7×10^{-3}	θ_2 (kg·m^2)	7.0×10^{-3}
θ_3 (kg·m^2)	8.0×10^{-3}	θ_4 (kg·m^2)	0.4×10^{-3}
θ_5 (kg·m)	9.1×10^{-3}	θ_6 (kg·m)	5.2×10^{-3}
θ_7 (N·m·s/rad)	0.096	θ_8 (N·m·s/rad)	0.145
θ_9 (N·m·s/rad)	0.055		

5.3 Force Sensor Integration

The operator input force should be measured as part of the control algorithm. This section describes the selection of a force sensor, the installation of the sensor on the Phantom Omni®, and the integration of the sensor with the control software.

5.3.1 Sensor Selection

A 3 DOF force sensor is necessary to measure the operator force used in the controller given by (4.8). The force sensor must have a maximum force range greater than ±3.3 N, since that is the output range of the Phantom Omni®. It must have an accuracy of at least 0.5 N and a resolution of at least 0.1 N. It would be preferable to have an accuracy better than 0.2 N [45], however due to budget constraints that accuracy may not be attainable. The force sensor should also have a mass less than or equal to 30 g, as that is approximately the mass of the Phantom Omni® stylus. The selected sensor is the OptoForce OMD-20-FG-100N, hereafter referred to as the OMD-20 or 'force sensor'. The OMD-20 has a maximum force range of ±10 N in the x and y sensor axes and a range from 50 N (tension) to -100 N (compression) in the z axis. It has a nonlinearity of 2-5% over the full range and less than 5% crosstalk between axes, leading to a claimed accuracy that is close to the desired accuracy. The OMD-20 has a 0.006 N resolution in the z axis and a 0.001 N resolution in the x and y axes, i.e. much better than desired. It has a mass of 23 g including the attached

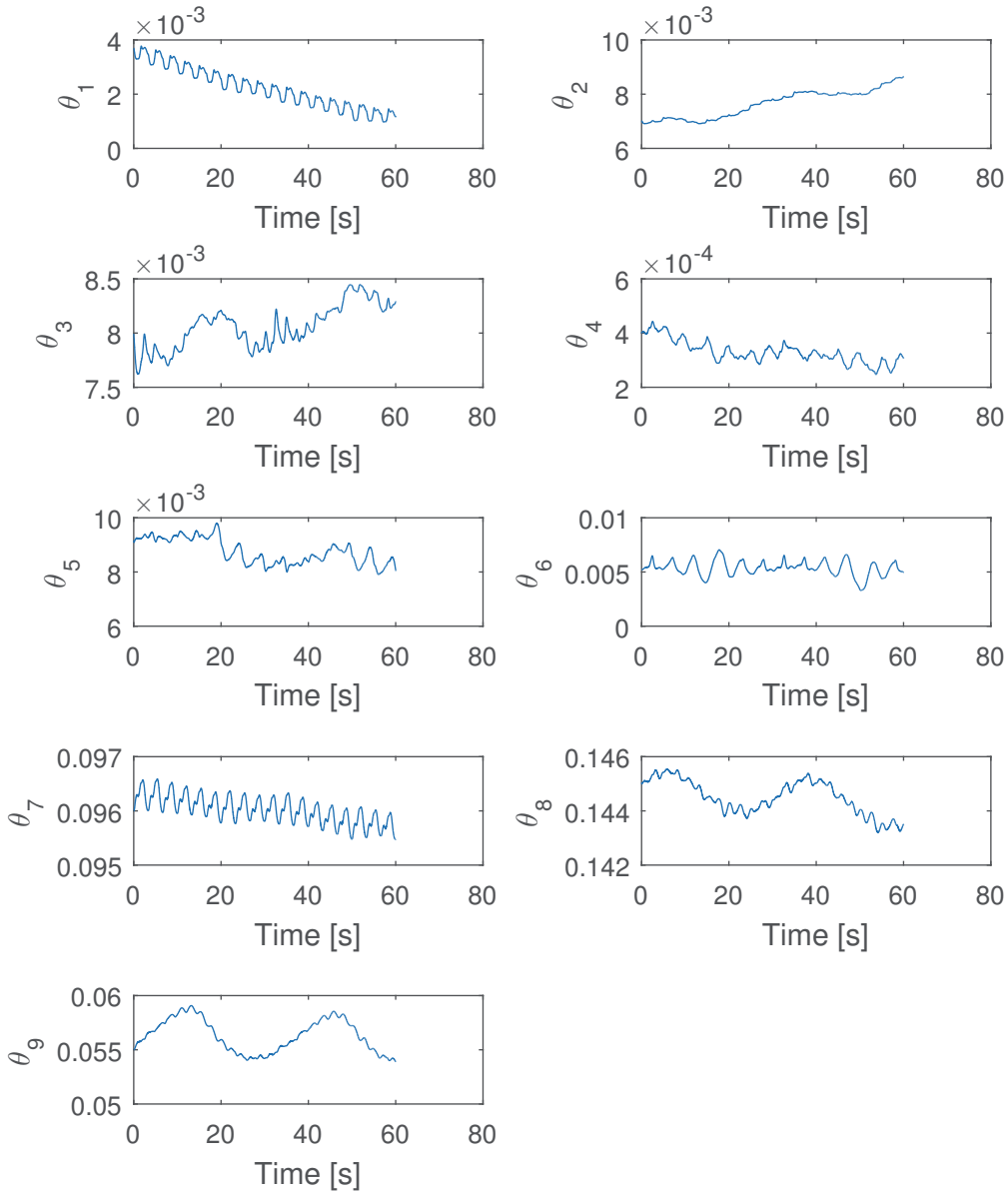


Figure 5.7: Phantom Omni® parameter estimation signals during the controller tuning test.

Table 5.3: Comparison of the desired and actual force sensor specifications

Specification	Desired	Actual Sensor
Max. Force Range (N)	±3.3	±10 or −100 to +50
Accuracy (N)	0.2	±5-10%
Resolution (N)	0.1	0.001 - 0.006
Mass (g)	30	23

cable. A comparison of the desired and actual force sensor specifications is shown in Table 5.3.

The OMD-20 comes with a DAQ card capable of running at a 1000 Hz sample rate, and has an internal low-pass filter. The DAQ card connects to the computer via a USB Type A connector and can be programmed in C++, C#, and MATLAB.

Two units were purchased by the lab with serial numbers IFG0A012 and IFG0A013, which will be abbreviated as FS012 and FS013. FS012 was used for the experimental work.

5.3.2 Sensor Mount Design

The OMD-20 has four threaded holes on each flange to use for mounting. The Phantom Omni® however is not designed to allow force sensors to be mounted on its end-effector. Therefore a custom sensor mount was designed.

Fig. 5.8 shows the end-effector of the Phantom Omni®. The stylus grip shown on the right attaches to the end-effector via a press-fit. A keyway in the stylus grip accommodates a key on the end-effector to synchronize the rotation of the stylus grip and an encoder inside the end-effector. The end-effector has a 1/4 inch (6.35 mm) jack plug which is used to receive button-press signals from the two buttons on the stylus grip.

The custom mount is designed in two parts: a base and a handle. Detailed part drawings for the custom mount are given in Appendix C. The base connects the end-effector to one end of the force sensor. The end of the base that connects to the end-effector is fixed in place using a press fit, similar to the original stylus grip. The end of the base that connects to the force sensor has a flange that matches the flanges on either end of the force sensor. Four clearance-fit M3 screw holes allow the force sensor to be fastened to the base. The handle attaches to the opposite side of the

Figure 5.8: The end-effector of the Phantom Omni® with detachable stylus grip.

force sensor using a similar flange. The operator interacts with the Phantom Omni® via the handle so that all interaction forces pass through the force sensor.
The base and handle are 3D printed out of ABS plastic. A picture of the complete hardware setup is shown in Fig. 5.1.

5.3.3 Sensor Calibration

The sensors have been calibrated by the supplier, OptoForce. The calibration information of FS012 is presented in Table 5.4. However the actual output is much less accurate, therefore the force sensor must be calibrated experimentally.
Brass weights were used to apply loads to each axis individually. Loads ranging from -5 to 5 N were applied to the X and Y axes, and loads ranging from -5 to 8 N were applied to the Z axis, with each load condition repeated twice. A linear model was fit to the sensor output. The experimentally derived conversion factors and the error between the linear model and the measured output of each axis are given in Table 5.5. The recorded data and calibration curves are shown in Fig. 5.9, Fig. 5.10, and Fig. 5.11.
There is significant cross-talk between the sensor axes. On average the cross-talk is within 5-10% of the applied force, at maximum it is 80-90% of the applied force (measured on the Z axis with a load applied to the X axis).
Another cause of inaccuracy is that applying loads farther from the force sensor origin results in an increased sensor output. This effect is shown in Fig. 5.12, where the sensor output is normalized with respect to the initial output for the load located at

Table 5.4: Manufacturer sensitivity information for FS012

	Deformation (mm)	Sensitivity (Counts/N)
F_x	1.7	955.53
F_y	1.7	962.22
F_z	1.3	162.44

Table 5.5: Experimental sensitivity information for FS012

	Sensitivity (Counts/N)	Linear model error (N)
F_x	1087.4	0.088
F_y	-1089.1	0.176
F_z	120.5	0.808

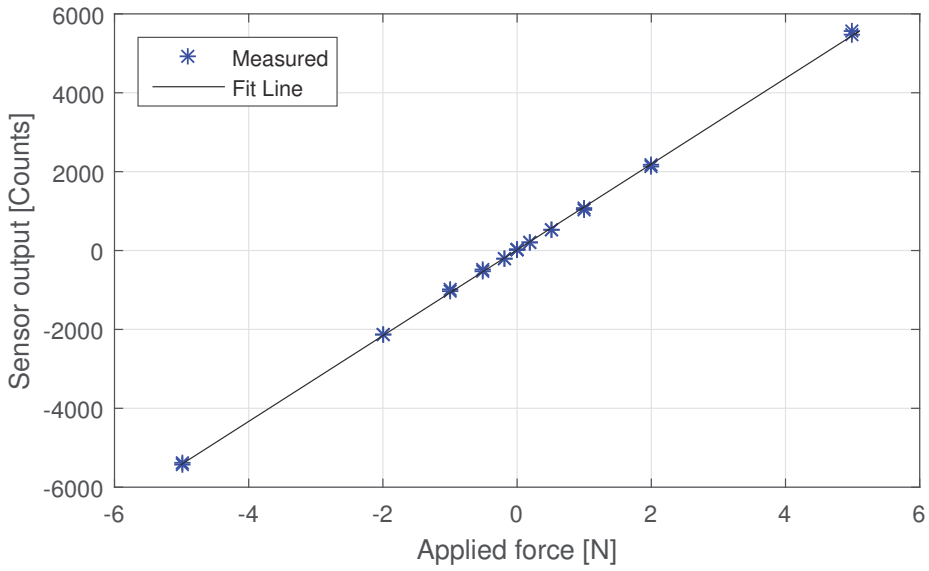


Figure 5.9: Calibration test data for FS012's x-axis

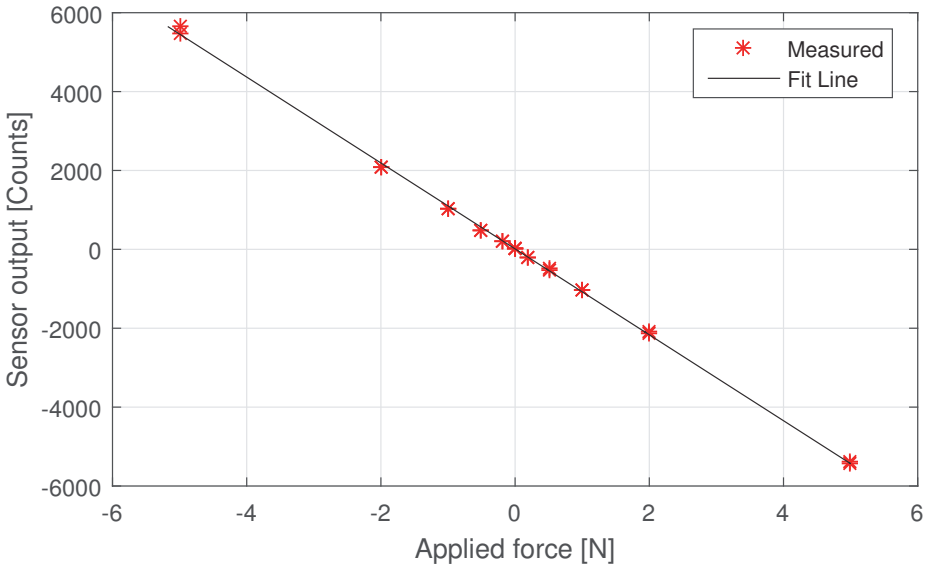


Figure 5.10: Calibration test data for FS012's y-axis

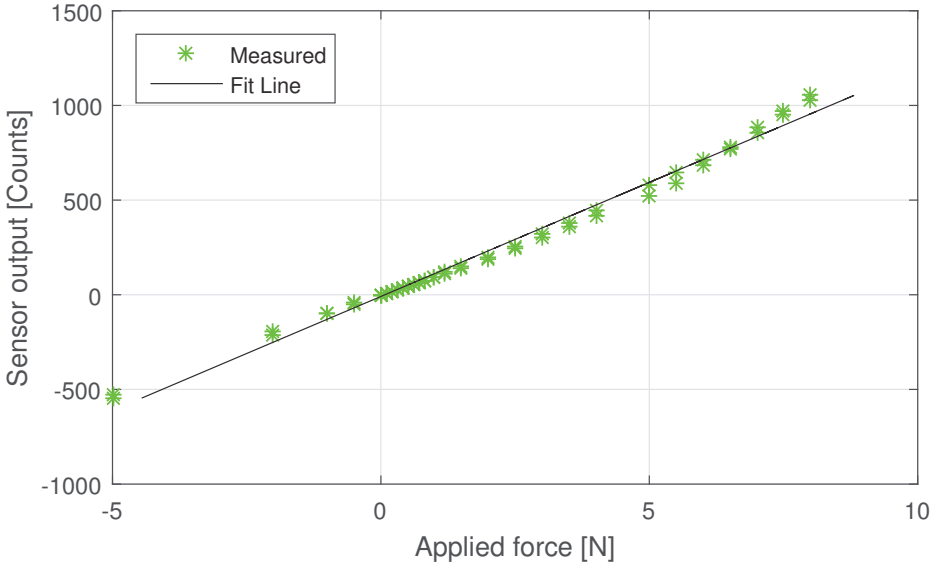


Figure 5.11: Calibration test data for FS012's z-axis

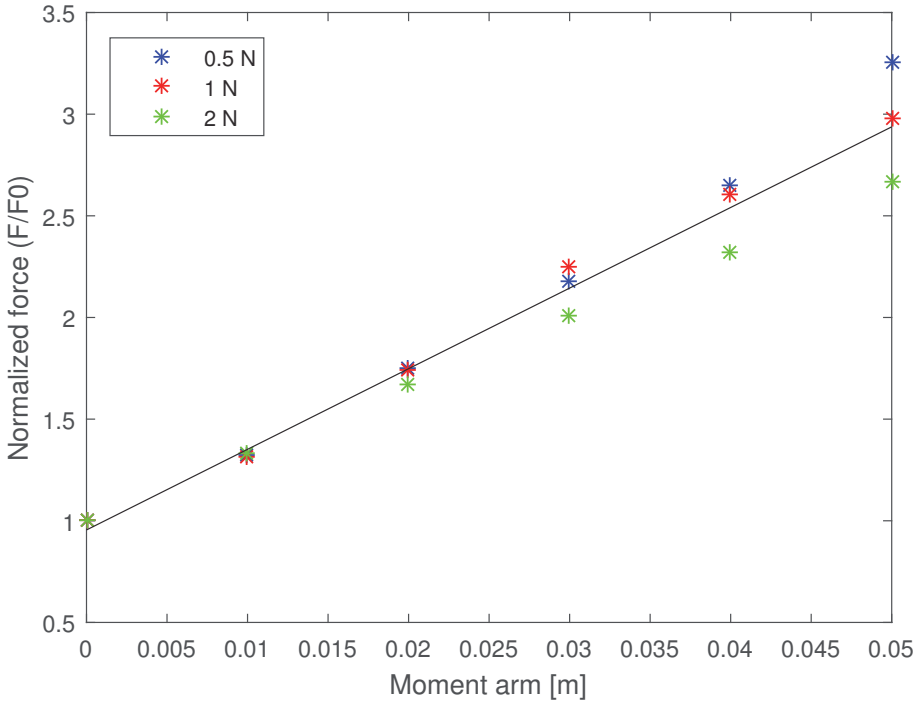


Figure 5.12: FS012 output normalized to the initial force applied in the positive x-axis at different distances from the axis

the origin. The relationship between the distance and the increase in applied force was approximated linearly with a slope of $a_m = 39.65$ m^{-1}, such that the actual force F_{actual} and the measured force F_{meas} are related by the following:

$$F_{actual} = \frac{F_{meas}}{a_m r},$$

where r is the distance from the force sensor flange. For the current force sensor mount design, $r = 0.073$ m.

5.3.4 Integration with QUARC

OptoForce provides a MATLAB API for communicating with the built-in OMD-20 DAQ card. This toolbox uses object-oriented programming, where all of the commands are implemented as the methods of two classes: `OptoPorts` and `OptoDAQ`. Instances of both classes are instantiated in MATLAB code and used to set up the connection with the OMD-20 DAQ and to acquire force measurements. Access to the objects is provided by MATLAB object handles.

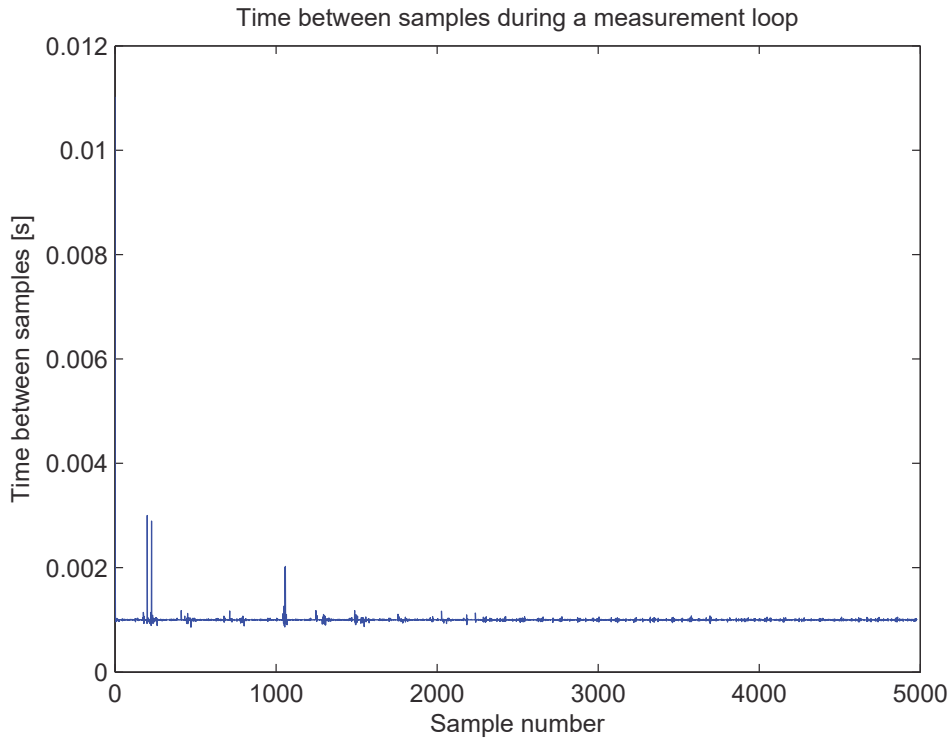


Figure 5.13: Plot of the time between samples for the QUARC force sensor server

The OptoForce MATLAB API uses object handles, therefore it cannot be easily used in Simulink since Simulink does not allow object handles to be passed on signal lines. Additionally it is important to perform error-handling and cleanup to prevent problems with the sensor, which are difficult to implement in Simulink. Therefore the sensor will be implemented as a QUARC Server running in a script in a separate instance of MATLAB. The control side will be implemented as a QUARC model with a QUARC client block receiving force measurements from the force sensor server.
Running the force sensor in a MATLAB script could introduce a delay if the script is incapable of maintaining as high an update rate as the client QUARC model. The force sensor server was run alongside a QUARC client to determine the maximum delay introduced. The time between force measurements is given in Fig. 5.13. The initial sampling interval was extremely long, around 10 ms, but quickly decreased to a range between 1 and 3 ms. Since the QUARC model was running at 1000 Hz, the sampling intervals would result in a delay between 1 to 3 samples.

Chapter 6

Simulation Results

This chapter describes the results of simulations performed to evaluate the performance of the designed predictor and controller in MATLAB Simulink. Four design cases are presented, corresponding to the design cases listed in Table 4.1.

6.1 Constant-Gain Predictor Results

The constant-gain cases are considered first. The predictor is simulated for several different cases without considering the haptic controller and Phantom Omni® dynamics. The last subsection shows the simulation results for the entire system including the Phantom Omni®.

6.1.1 Known Linear Virtual Environment with Delay (Case 1)

The first design case is a linear environment with delay but no discretization. The purpose of this design case is to demonstrate the effectiveness of the LMI predictor design in the ideal case without disturbances. The simulation parameters are given in Table 6.1. The time delay of the system is $T = 1/60$ s, and the initial conditions are $\tilde{\boldsymbol{x}}(0) = [0.01 \quad 0]^T$.

The effect of the arbitrary parameters a_2 and a_3 in Theorem 2 from Chapter 4 is unknown, so a variety of values are tested to determine their effect on the LMI solution feasibility and the resulting error dynamics. Theorem 2 is used to design the predictor gain over the range $0 < a_2 \leq 1$ and $0 < a_3 \leq 0.04$. For each combination

Table 6.1: Parameter values used in the predictor simulations (Design Case 1)

Parameter	Value	Parameter	Value
m_t (kg)	0.1	b_e (N s/m)	45
b_t (N s/m)	0.1	k_e (N/m)	2000
k_t (N/m)	1		

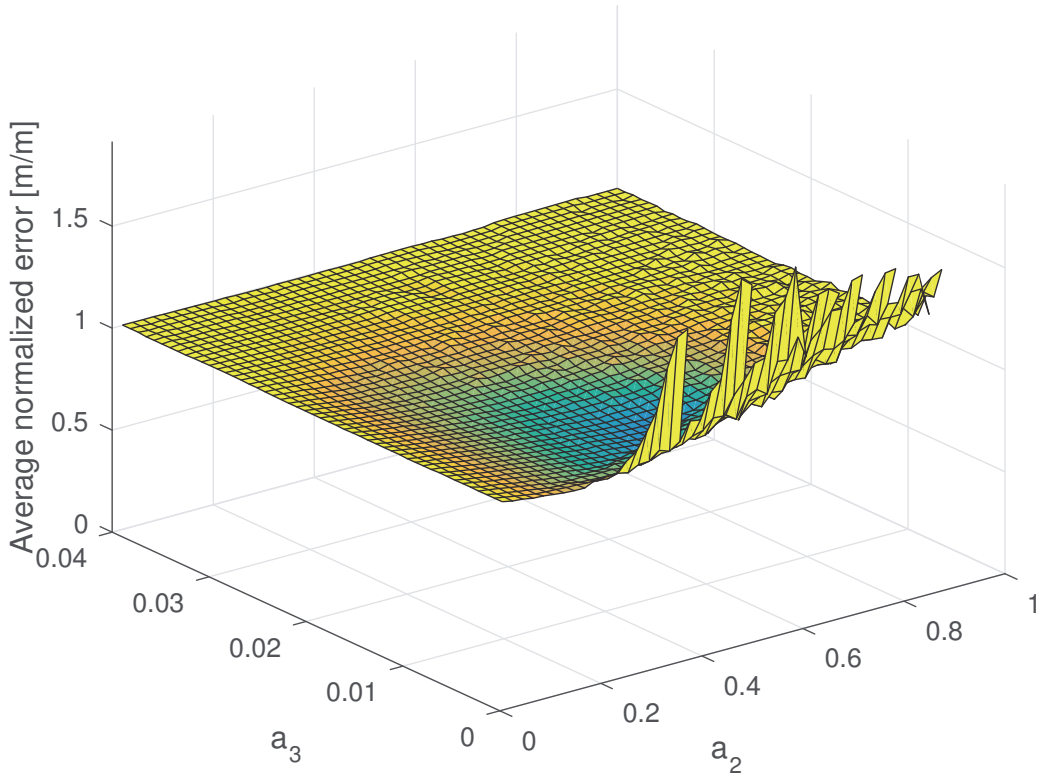


Figure 6.1: Normalized average magnitude of the position error with varying a. Colours darker than yellow indicate an improvement over no feedback.

of a_2 and a_3, the resulting predictor gain used in the simulation and the average position error magnitude is calculated. The average error magnitude values are then normalized against the average error magnitude resulting from a predictor without feedback (since the system is stable without feedback), so that values less than 1 represent an improved tracking capability. The resulting plot is shown in Fig. 6.1. A local minimum was found around $a_2 = 0.4694$ and $a_3 = 0.008980$, resulting in a predictor gain of $L = [-753.2 \quad -17.01]$. The predictor gain at this minimum was used in simulation, generating the error trajectory shown in Fig. 6.2.

6.1.2 Known Linear Virtual Environment with Delay and Sampling (Case 2)

The second design case is a linear environment with delay and sampling. The initial conditions are $\hat{x}_e = 0$, $x_e = 0.001$ m, and $\dot{x}_e = \dot{\hat{x}}_e = 0$. The operator input force was

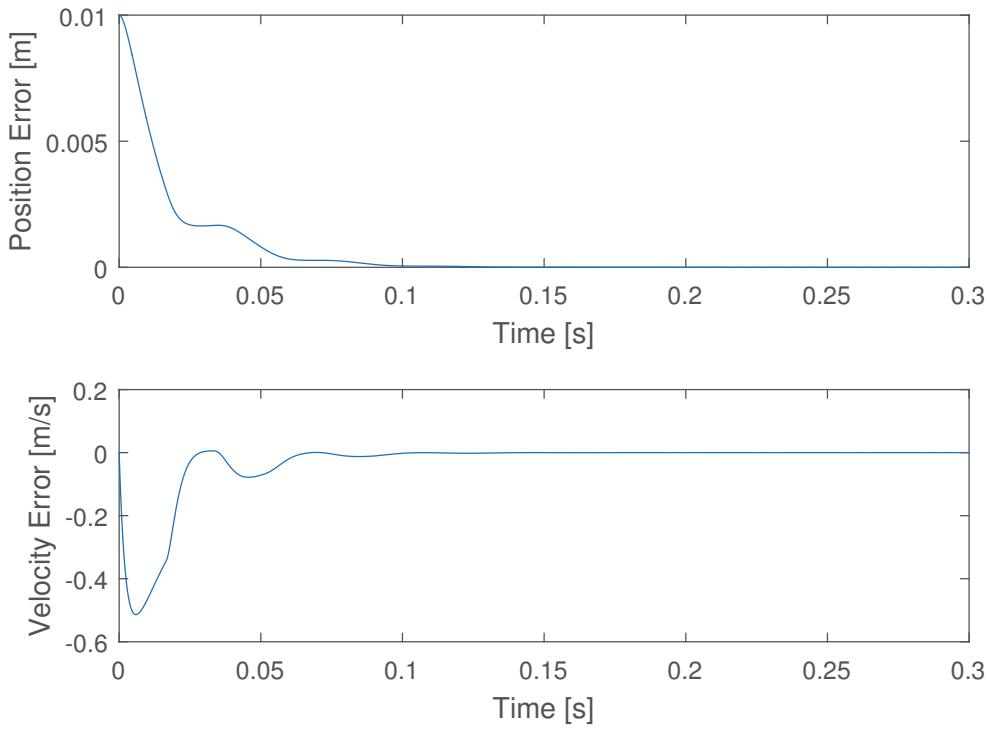


Figure 6.2: Predictor error for optimal predictor gain of $L = [-753.2 \quad -17.01]$ (Design Case 1)

Table 6.2: Parameter values used in the predictor simulations (Design Case 2)

Parameter	Value	Parameter	Value
m_t (kg)	0.1	b_e (N s/m)	25
b_t (N s/m)	0.1	k_e (N/m)	2000
k_t (N/m)	1	γ	100
a_2	0.4694	a_2	0.008980

simulated as a sinusoid described by $F_h(t) = 3\sin(t/\pi)$ N. The simulation parameters are given in Table 6.2. Using Theorem 3 from Chapter 4 to design the predictor gain results in $L = [-574.1 \quad -7.260]$. Fig. 6.3 shows the predicted and actual trajectories of the virtual tool.

The performance of the predictor is compared to the delayed and sampled signal by comparing the predictor error $\boldsymbol{e}$ to the delay and sampling error $\boldsymbol{e}_\Delta$, defined as,

$$\boldsymbol{e}_\Delta = \begin{bmatrix} x_{e,zd} - x_e \\ \dot{x}_{e,zd} - \dot{x}_e \end{bmatrix}. \tag{6.1}$$

Fig. 6.4(a) shows a comparison of the magnitude of the predictor error ($\|\boldsymbol{e}\|$) and the delay and sampling error ($\|\boldsymbol{e}_\Delta\|$). Fig. 6.4(b) shows a comparison of the $\mathcal{L}_2$ norms of both error signals ($\|\boldsymbol{e}\|_{\mathcal{L}_2}$ and $\|\boldsymbol{e}_\Delta\|_{\mathcal{L}_2}$). The error magnitudes appear to approach zero after 0.2 seconds however only the predictor error approaches zero; the delay and sampling error oscillates at an amplitude of around 5×10^{-4}.

The amplitude of the delay and sampling error becomes more significant at a lower virtual environment stiffness since the amplitude of the virtual tool trajectory increases. Fig. 6.5 shows the error signals of the same system with $k_e = 200$ N/m instead of 2000 N/m. This shows that the relative performance of the predictor is heavily influenced by the environment stiffness.

The disturbance rejection factor γ can be tuned to affect the error bound according to (4.32). To investigate the effect of the disturbance rejection factor, the system is simulated for a range of predictor gains designed using different values of γ. Fig. 6.6 shows the maximum of the predictor error magnitude, normalized to the maximum of the delay and sampling error magnitude (i.e. $\|\boldsymbol{e}\|_{max}/\|\boldsymbol{e}_\Delta\|_{max}$), over a range of $0 < \gamma \leq 1000$ for a virtual environment stiffness of $k_e = 2000$ N/m. Fig. 6.7 shows the predictor error bound over the same range as calculated by (4.32). The error bound guaranteed by (4.32) over this range is impractically large (on the order of

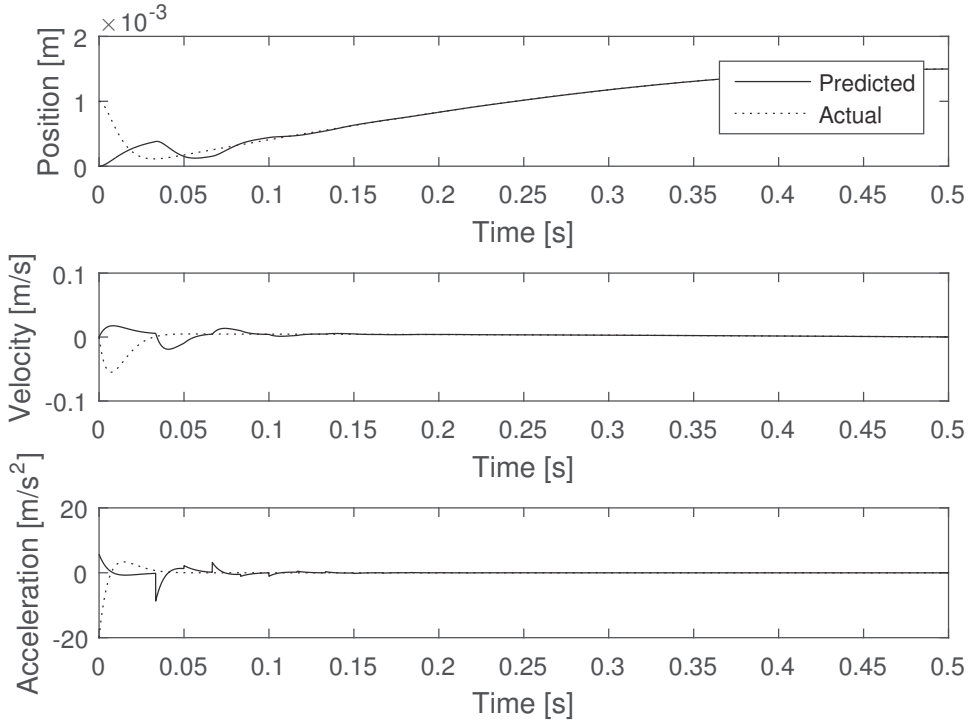


Figure 6.3: Predicted and actual virtual tool trajectories (Design Case 2)

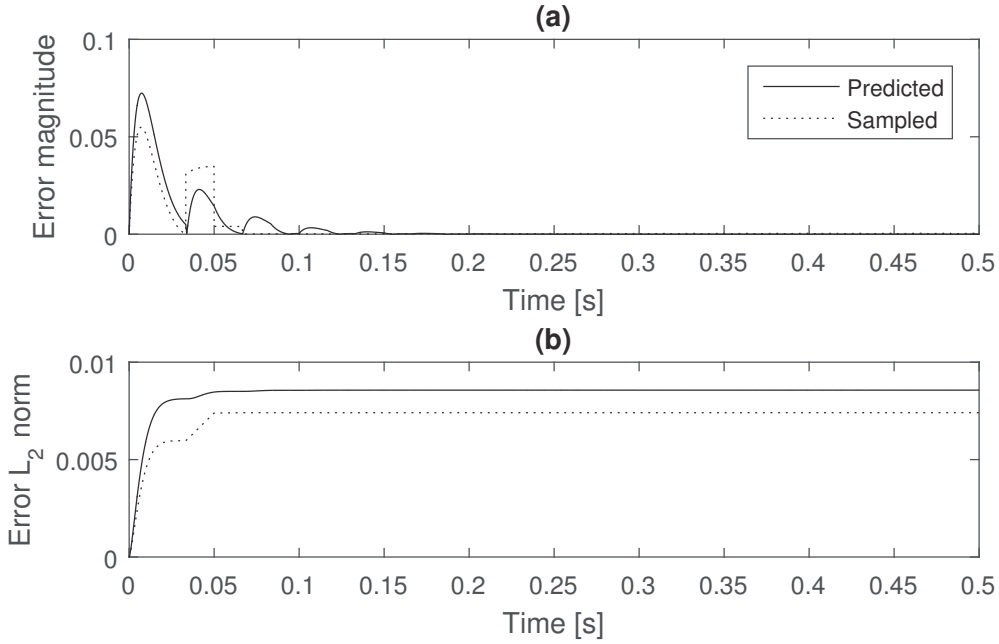


Figure 6.4: Comparison of the error magnitude and $\mathcal{L}_2$ norm of the error for the predicted, and delayed and sampled, signals (Design Case 2)

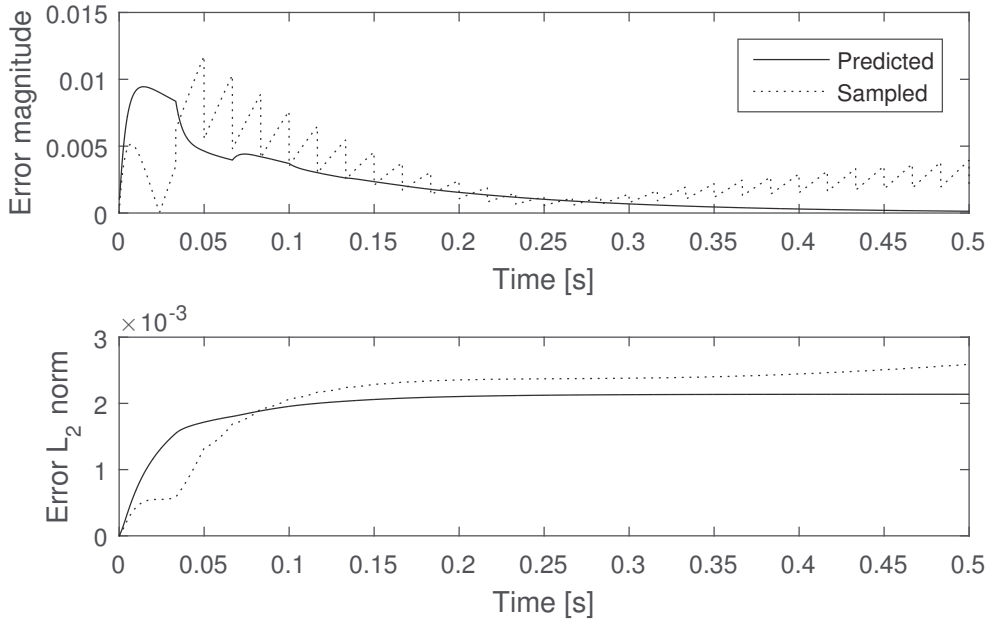


Figure 6.5: Comparison of the error magnitude and $\mathcal{L}_2$ norm of the error for a lower-stiffness VE (Design Case 2)

1×10^5 m or m/s), and does not correlate at all with the actual error bound.

Given the previously mentioned effect of the stiffness on the predictor performance, the same set of simulations are performed for $k_e = 200$ N/m, as shown in Fig. 6.8. Fig. 6.6 and Fig. 6.8 show that γ does not have the same effect on the error bound for different values of virtual environment stiffness. There is a more distinct decreasing trend in Fig. 6.8 for $\gamma > 200$, however the change is only on the order of 2-3% of the delay and sampling error. In both cases, γ does not seem to have a large effect on the maximum predictor error. Since the relationship between the error bound and γ is not clear nor very significant, the disturbance rejection factor is selected as $\gamma = 100$.

6.1.3 Unknown Linear Virtual Environment with Delay and Sampling (Case 3)

In this design case the virtual environment parameters are unknown, therefore the parameter adaptation law (4.44) is introduced. The simulation parameters are given in Table 6.3. The bounds on the stiffness and damping are $200 \leq k_e \leq 2000$ N/m and

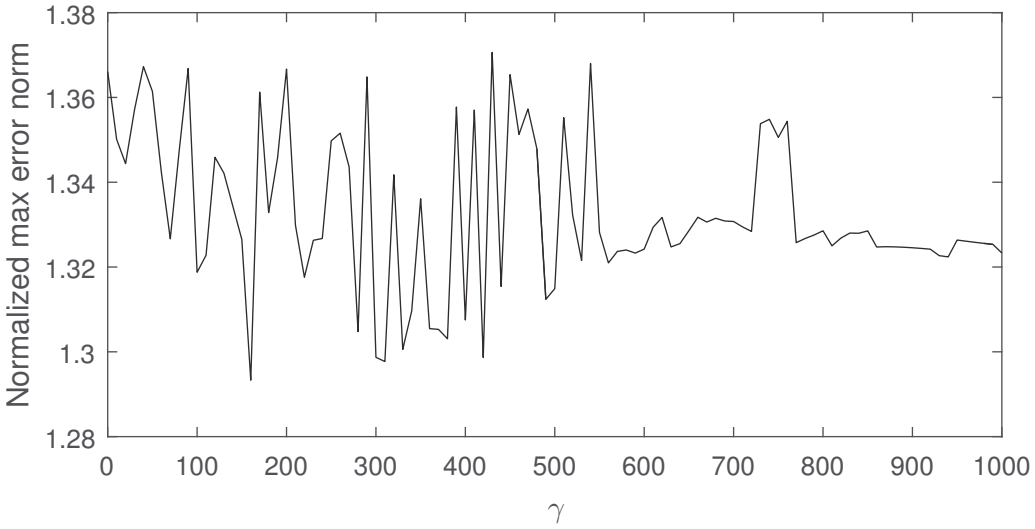


Figure 6.6: Maximum predictor error magnitude, normalized to the maximum delay and sampling error magnitude, for a range of disturbance rejection factors and k_e = 2000 N/m (Design Case 2)

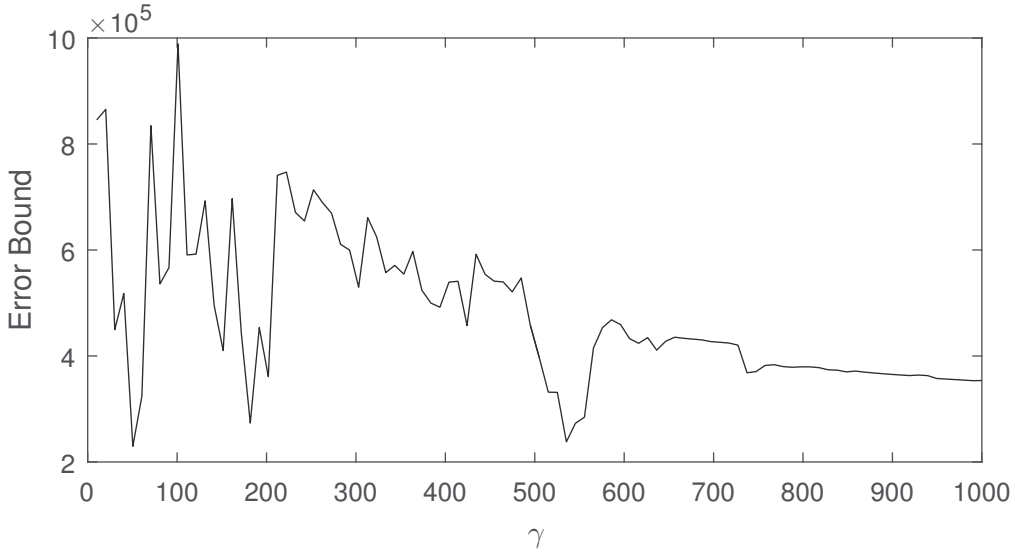


Figure 6.7: Guaranteed predictor error bound for a range of disturbance rejection factors and $k_e = 2000$ N/m (Design Case 2)

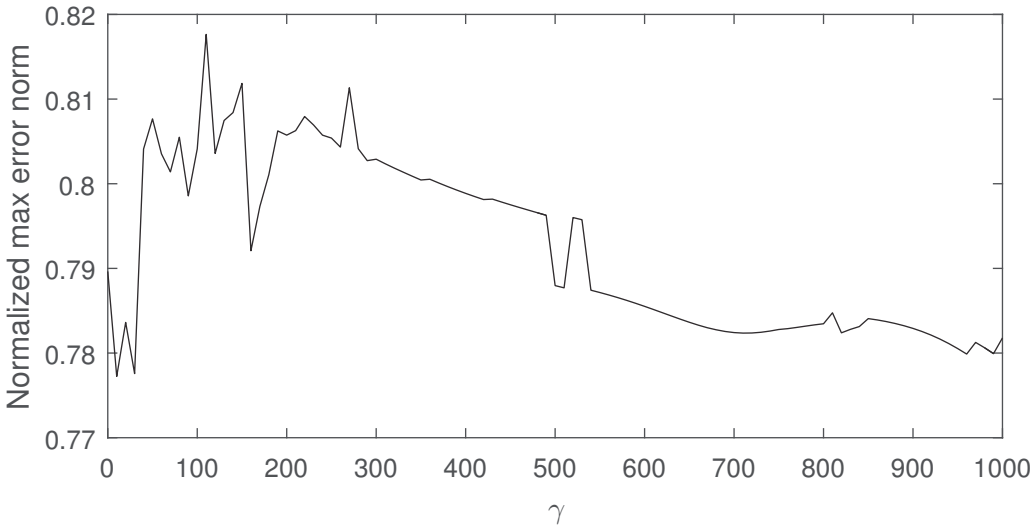


Figure 6.8: Maximum predictor error magnitude, normalized to the maximum delay and sampling error magnitude, for a range of disturbance rejection factors and $k_e = 200$ N/m (Design Case 2)

$20 \leq b_e \leq 40$ N·s/m, respectively. The stiffness range is selected based on the soft-tissue stiffness data from [21]. The damping range is selected to ensure the virtual environment is near critical damping or overdamped, since tissue vibrations tend to die out quickly. The initial adaptation gain matrix is $\Gamma(0) = \text{diag}(1 \times 10^9,\ 8 \times 10^6)$. The initial conditions of the states are $x_e(0) = \hat{x}_e(0) = \dot{x}_e(0) = \dot{\hat{x}}_e(0) = 0$. There is no longer an initial error because it is assumed that the state of the tool is accurately known until it contacts the environment.

Table 6.3: Parameter values used in testing the adaptation law

Parameter	Value	Parameter	Value
m_t (kg)	0.1	k_t (N/m)	1
b_t (N s/m)	0.1	γ	100
a_2	0.4694	a_2	0.008980
ν	0.01	α	2
ρ_M	5×10^9		

To demonstrate the effectiveness of the parameter adaptation law, the predictor is tested for the worst-case scenario where the initial parameter estimation error $\tilde{\boldsymbol{\theta}}_e$ is as large as possible. The initial estimate of the virtual environment parameters is $\hat{\boldsymbol{\theta}}_e = [200 \quad 20]^T$ while the actual environment parameters are $\boldsymbol{\theta}_e = [2000 \quad 40]^T$. The mean

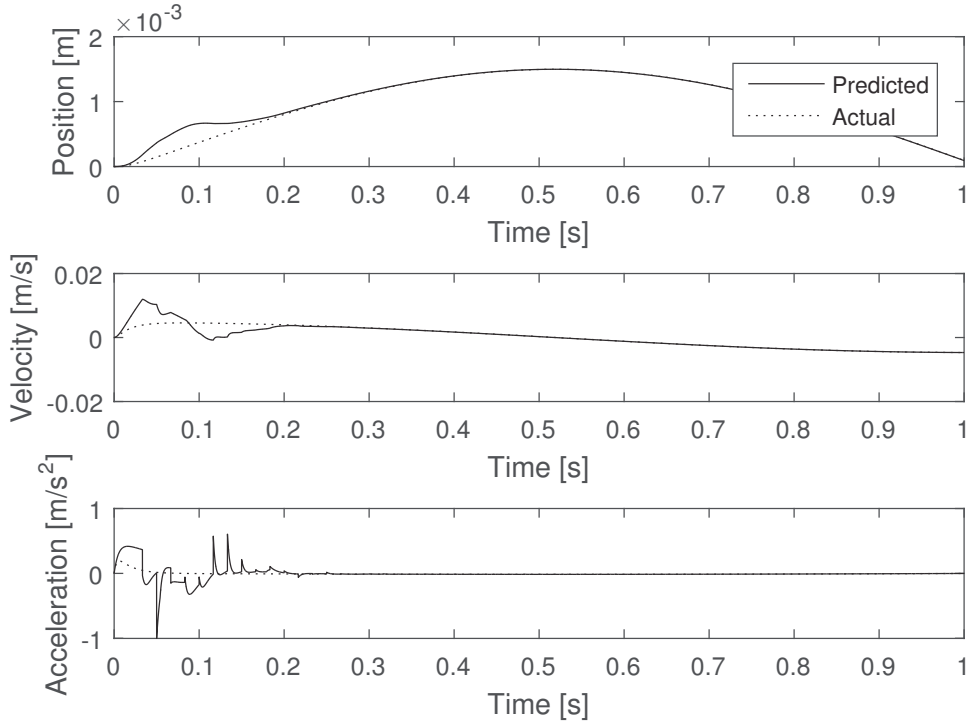


Figure 6.9: Virtual tool states for the worst-case parameter estimation error with $L = [-430.6 \quad -11.83]$ (Design Case 3)

values of the virtual environment parameters ($k_e = 1100$ N/m and $b_e = 30$ N·s/m) are used to design the predictor gain using Theorem 3. These values are selected to minimize the maximum possible parameter difference between the design parameters and the true parameters. The resulting predictor gain is $L = [-430.6 \quad -11.83]$. The LMI in (4.31) was verified over the parameter range for the resulting predictor gain, showing that the system is guaranteed to be stable over that range.

Fig. 6.9 shows the predicted and actual states of the virtual tool for the worst-case estimation error. Since the initial parameter estimates are smaller, there is an initial overshoot in the predicted position and velocity. The error magnitude and $\mathcal{L}_2$ norm are given in Fig. 6.10. The predictor error converges to zero shortly after the stiffness estimate converges to its actual value, as shown in Fig. 6.11. The damping estimate takes much longer to converge due to the averaging effect of the least-squares parameter adaptation law.

A less extreme case is considered where $\boldsymbol{\theta}_e = [1500 \quad 35]^T$ and $\hat{\boldsymbol{\theta}}_e = [1100 \quad 30]^T$.

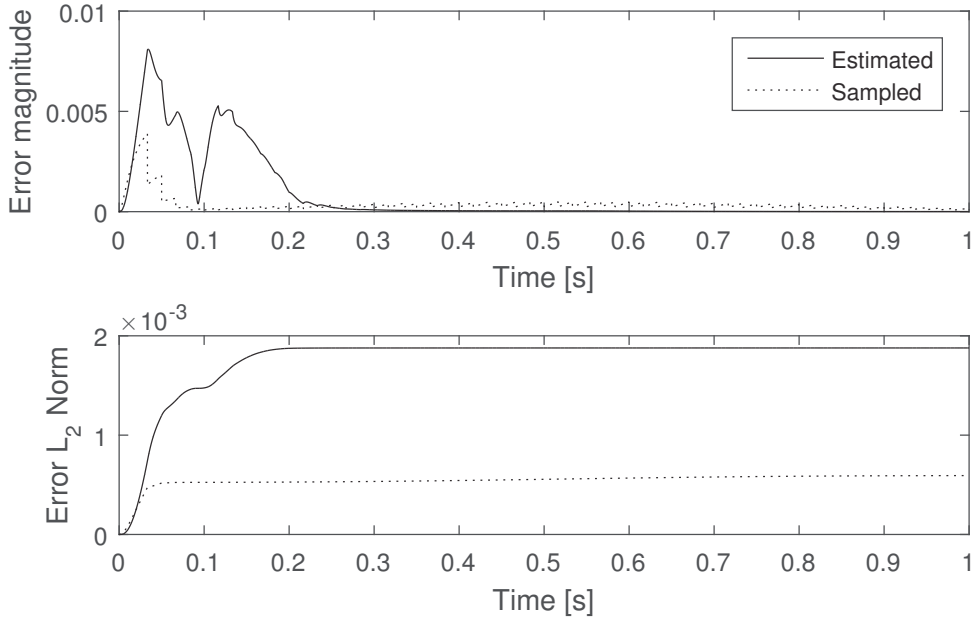


Figure 6.10: Error magnitude comparison for the worst-case parameter estimation error (Design Case 3)

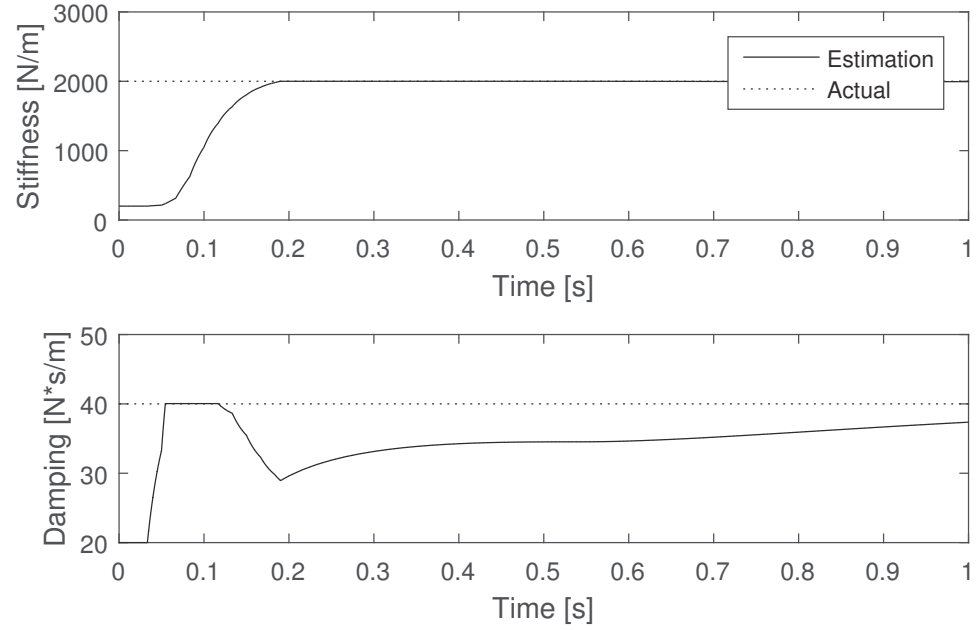


Figure 6.11: Estimated parameters for the worst-case parameter estimation error (Design Case 3)

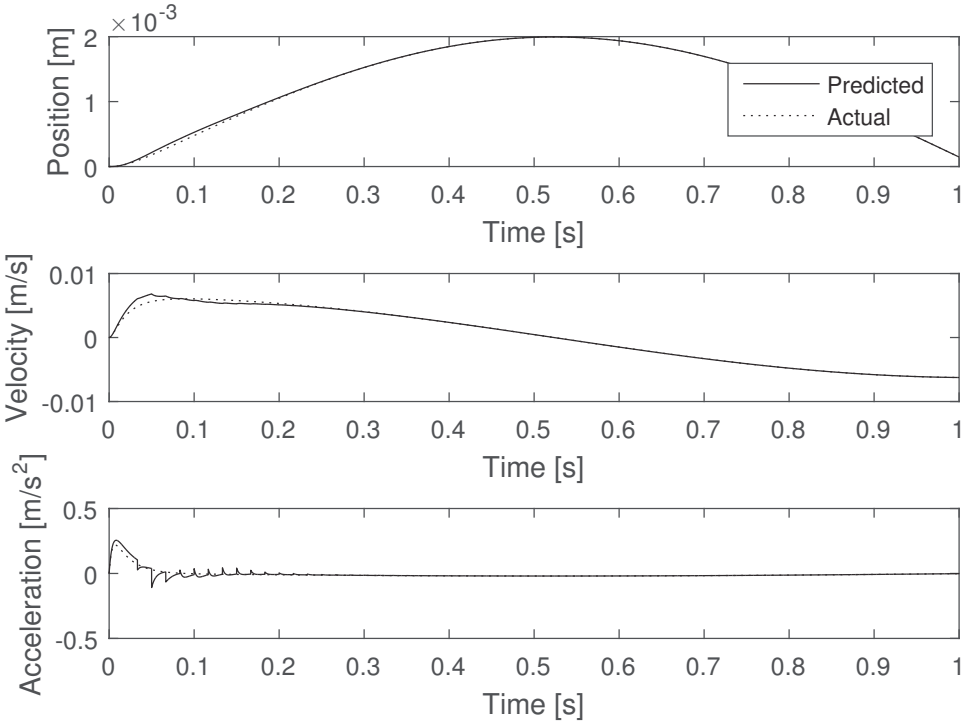


Figure 6.12: Virtual tool states for the typical parameter estimation error (Design Case 3)

The predictor gain does not change in this case. The estimated states are shown in Fig. 6.12, the error is shown in Fig. 6.13 and the estimated parameters are shown in Fig. 6.14. The estimated damping converges much faster and the maximum error magnitude is greatly reduced. The predictor easily outperforms the sampled and delayed signal.

6.1.4 Complete System

To fully demonstrate the effectiveness of the predictor the complete haptic system, including the controller and the Phantom Omni® dynamics, was simulated.

The predictor was simulated for constant VE parameters. Nine cases were tested, with a different combination of VE stiffness and sampling rate for each case. Three different stiffness values, soft ($k_e = 200$ N/m), medium ($k_e = 800$ N/m, and hard ($k_e = 1500$ N/m), and three different sampling rates, fast ($T = 0.016$ s), medium ($T = 0.032$ s), and slow ($T = 0.100$ s), were selected, for a total of nine test cases. Cases 1 through 3

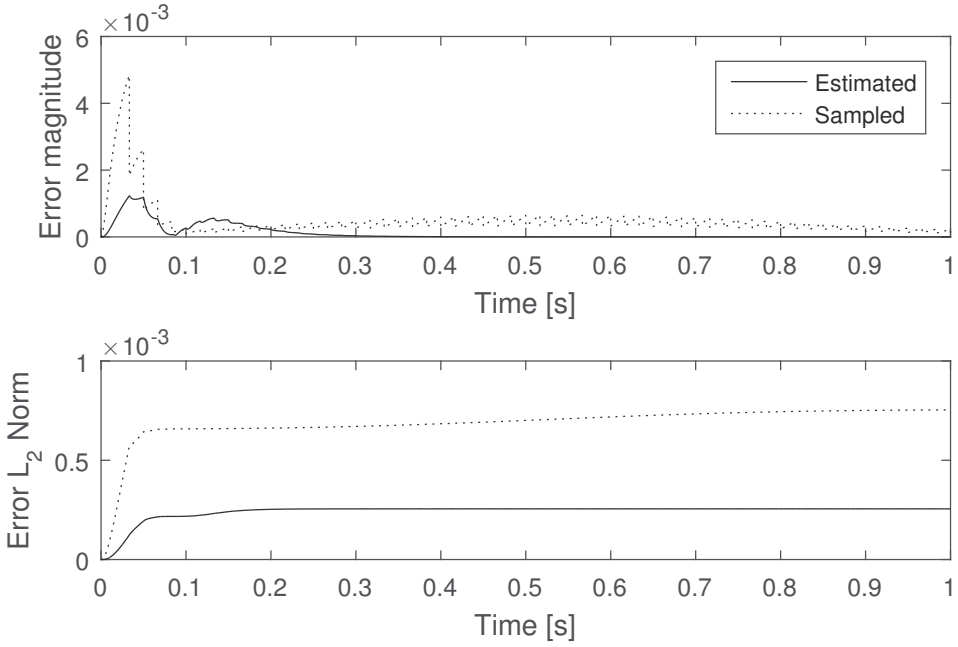


Figure 6.13: Error magnitude comparison for the typical parameter estimation error (Design Case 3)

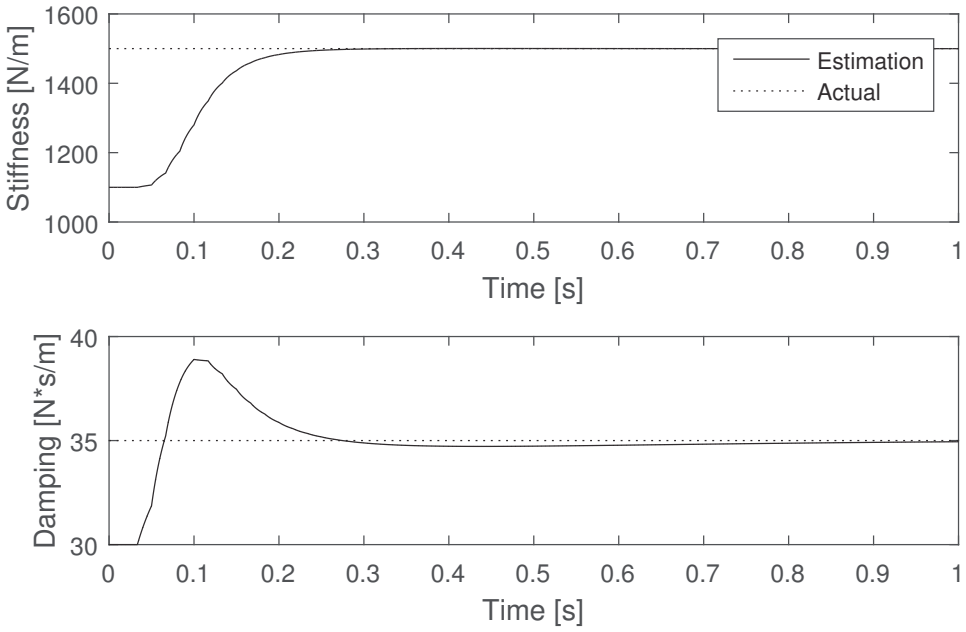


Figure 6.14: Estimated parameters for the typical parameter estimation error (Design Case 3)

Table 6.4: Designed predictor gains for constant VE parameters

Cases	Sampling Time (T, s)	Predictor Gains L_1	L_2	Full-Range Stable?
1, 4, 7	0.016	-441.3	-12.12	Yes
2, 5, 8	0.032	-203.3	-5.15	Yes
3, 6, 9	0.100	-195.1	-4.89	No

have soft stiffness and increasingly slow sampling, similarly for Cases 4 through 6 and 7 through 9, except with medium and hard stiffness, respectively. The damping was the same for all three tests ($b_e = 30$ N·s/m). The initial VE parameter estimate of the predictor was $\hat{k}_e(0) = 1100$ N/m and $\hat{b}_e(0) = 30$ N·s/m. The designed predictor gains are given in Table 6.4. Since the initial VE parameter estimate was the same for all nine cases, the only parameter change that results in a different predictor gain is the sampling time.

The designed control gain guarantees a stable predictor system over the entire parameter range for $T = 0.016$ s and $T = 0.032$ s, however it does not for $T = 0.100$ s. For $T = 0.100$ s, there is a small range where the LMI in (4.31) does not have a solution for $L = [-195.1 \quad -4.89]$, as shown in Fig. 6.15. Since the potentially unstable region corresponds to a much lower stiffness than is actually being used in this simulation, and the LMI is very conservative, the simulation was performed without changing the predictor design. However, if necessary, the predictor gain could be redesigned by changing the nominal stiffness and damping used to design it, or by changing a_2, a_3, or γ.

The initial conditions of the simulated Phantom Omni® were $\boldsymbol{q}(0) = [0 \quad \pi/4 \quad -1.25]^T$ and $\dot{\boldsymbol{q}}(0) = [0 \quad 0 \quad 0]^T$. The dynamic parameters of the Phantom Omni® identified in Table 5.2 were used in the simulation, except the joint friction parameters were set to $\theta_7 = \theta_8 = \theta_9 = 0$. The initial estimate of the dynamic parameters used in the adaptation law was $\hat{\boldsymbol{\theta}} = 1.1\boldsymbol{\theta}$. The control and adaptation gains were set to the values given by (5.6) and (5.7). The rest of the simulation parameters are given in Table 6.3.

The environment was simulated as a vertical wall in the global y-z plane so that the environment forces only act along the global x-axis. The wall was positioned so that the end-effector started on the surface. The operator force was simulated as a sinusoid

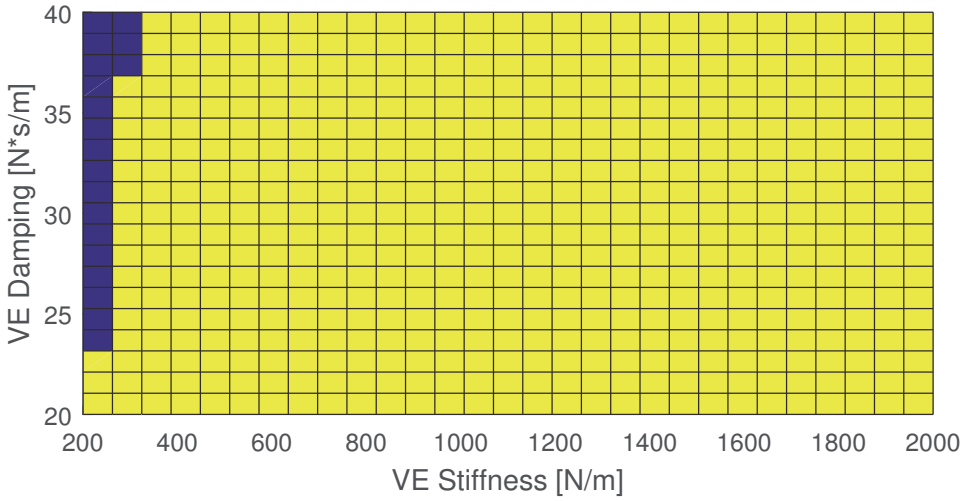


Figure 6.15: Stability range for the predictor when $T = 0.1$ s. (Yellow squares indicate the stable region; blue squares indicate the potentially unstable region)

similar to previous case, with the operator only applying force along the global x-axis.

An example end-effector trajectory and tracking error signal are shown in Fig. 6.16 for the case where $k_e = 200$ N/m and $T_d = 0.016$ s. The x position of the end effector tracks the desired trajectory well, with a position error of less than 0.5 mm after 2 s. The tracking was similar for the other cases.

The performance results are summarized in Table 6.5. Note that a decrease in the error indicates an increased performance. The error bound difference and the $\mathcal{L}_2$ norm difference are calculated based on (3.36), (3.37), and (3.38). In all cases, the performance of the system is greatly improved by adding the predictor, resulting in the error bound decreasing by over ten times the Weber fraction of 7%. Generally the relative performance of the predictor increased as the sampling time increased. This was expected given that increasing the sampling time increases the sampling disturbance and delay. Changes in the virtual environment stiffness did not result in significant changes in the performance.

An example of the predictor error is given in Fig. 6.17 for the case where $k_e = 200$ N/m and $T_d = 0.016$ s. The predictor error is consistently lower than the sampling error, as in previous cases. Comparing it to Fig. 6.18 shows that including the Phantom Omni® and controller dynamics results in more total error than simply considering the output of the predictor in isolation. The total error magnitude and

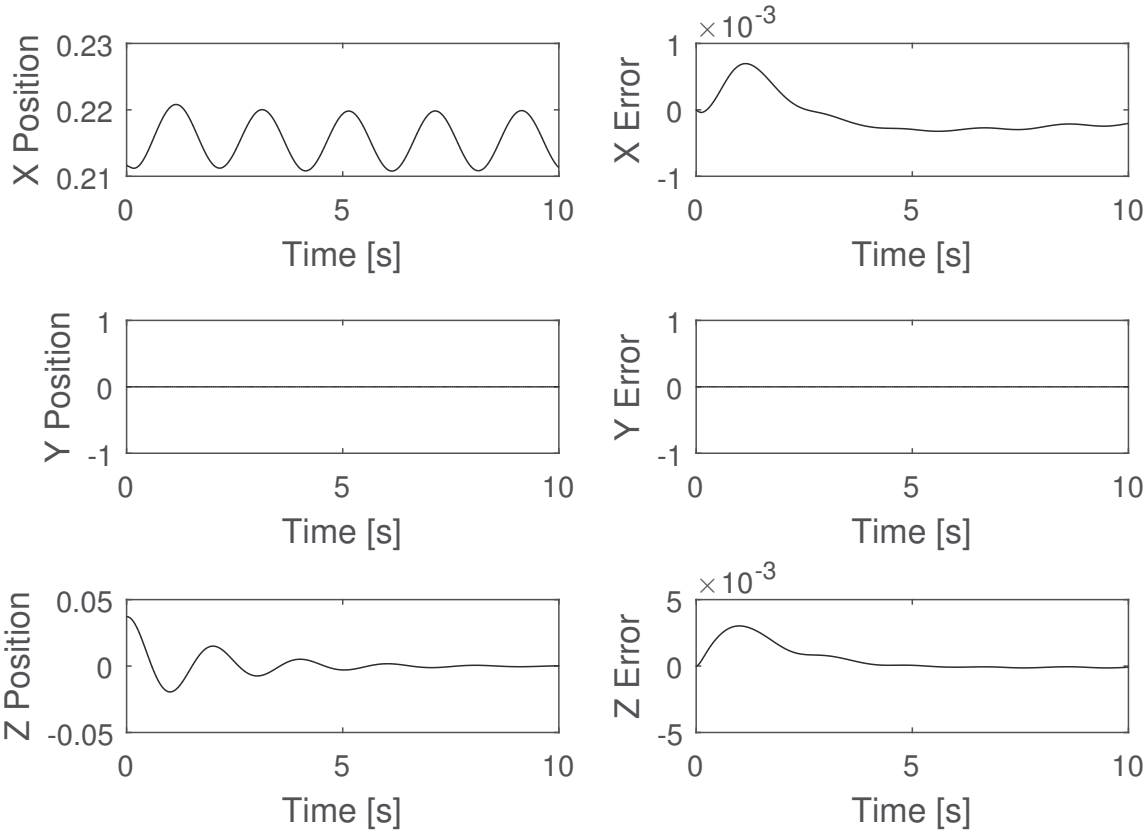


Figure 6.16: End-effector position and tracking error ($\boldsymbol{x} - \hat{\boldsymbol{x}}_e$) for a linear virtual environment with $k_e = 200$ N/m and $T_d = 0.016$ s

Table 6.5: Summary of the performance of the predictor for constant VE parameters

Case	Stiffness (k_e, N/m)	Sampling Time (T_d, s)	Error Bound Difference	$\mathcal{L}_2$ Norm Difference
1	200	0.016	-0.0123 (-69.2%)	-39.7%
2		0.032	-0.0210 (-78.7%)	-62.5%
3		0.100	-0.0348 (-83.9%)	-85.7%
4	800	0.016	-0.0148 (-73.2%)	-38.9%
5		0.032	-0.0245 (-81.9%)	-62.0%
6		0.100	-0.0372 (-87.3%)	-85.7%
7	1500	0.016	-0.0166 (-75.4%)	-39.0%
8		0.032	-0.0255 (-82.4%)	-61.9%
9		0.100	-0.0371 (-87.1%)	-85.6%

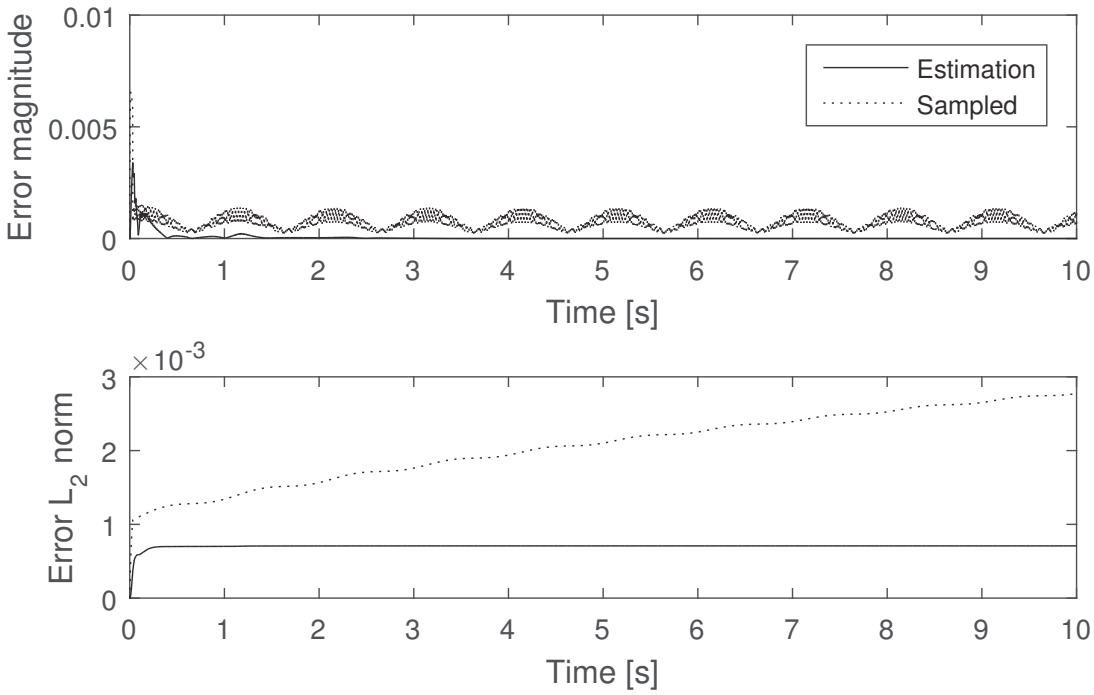


Figure 6.17: Predictor error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 1)

$\mathcal{L}_2$ norm of the total error for each of the nine test cases are presented in Fig. 6.18 to Fig. 6.26.

The addition of the predictor not only reduces the total error, it also reduces chattering in the control torque by producing a continuous position and velocity signal. Fig. 6.27 shows the control torque applied to each joint with the predictor, while Fig. 6.28 shows the control torque for the same case without the predictor. The chattering in the torque for the case without the predictor is approximately the same order of magnitude as the ideal torque signal. A real-world system would likely be driven unstable by such large-amplitude chattering.

The overall performance of the predictor is greatly affected by the virtual tool stiffness k_t. For example, Fig. 6.26 shows the total error for the haptic system when $k_t = 1$ N/m, while Fig. 6.29 shows the total error for the haptic system when $k_t = 0$ N/m. The total error magnitude decreases when k_t decreases, however the difference between the system with and without prediction decreases as well: the peak errors are effectively the same (0.1% difference) and the $\mathcal{L}_2$ norm difference is just noticeable at -7% (compared to -87.1% and -85.6% respectively, when $k_t = 1$ N/m).

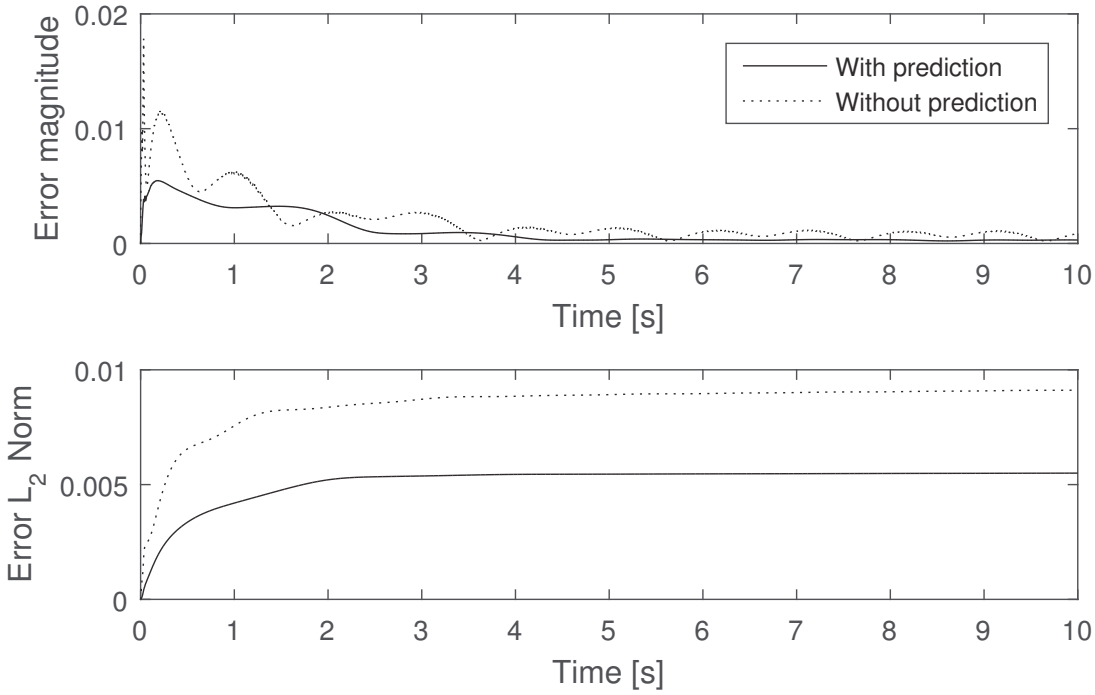


Figure 6.18: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 1)

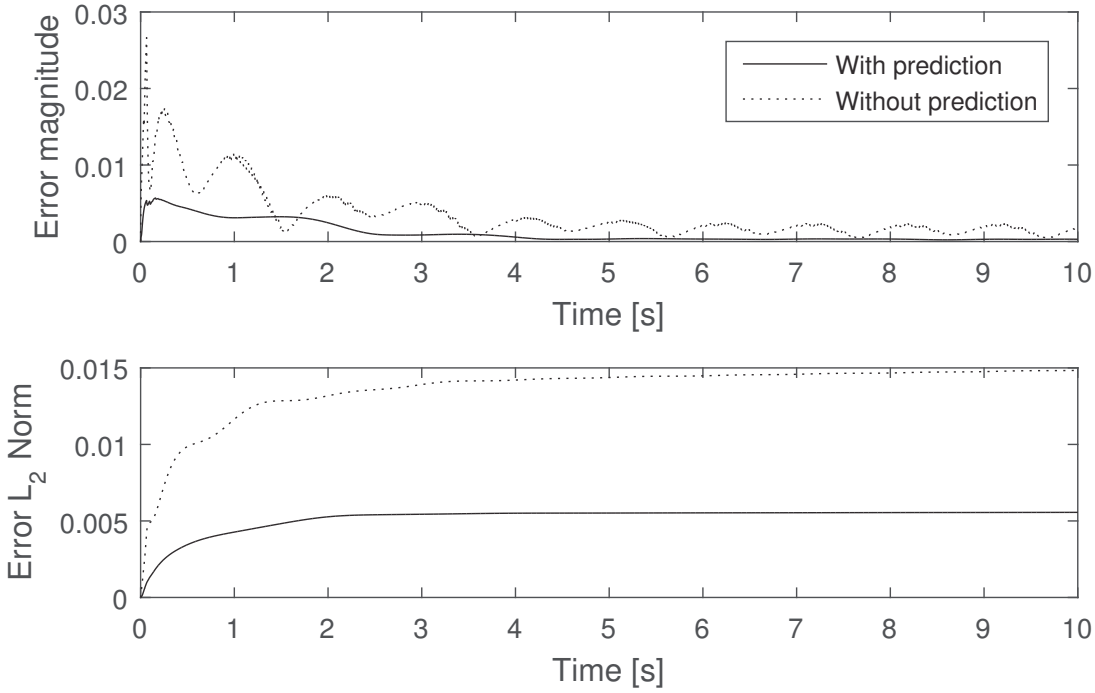


Figure 6.19: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 2)

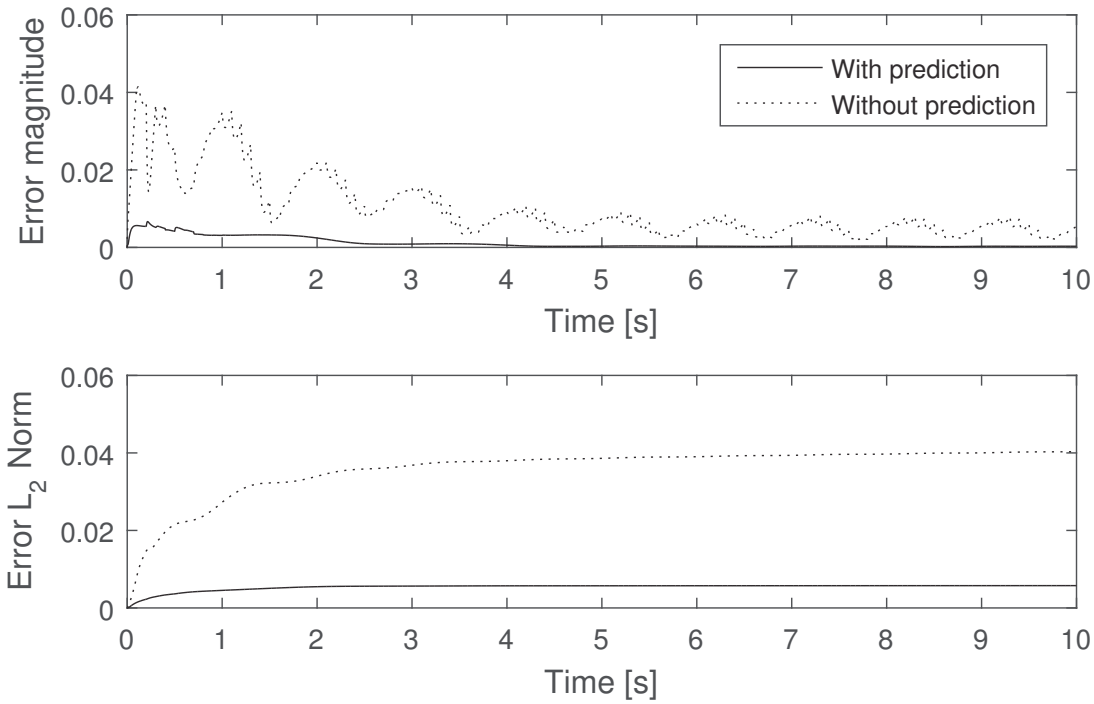


Figure 6.20: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 3)

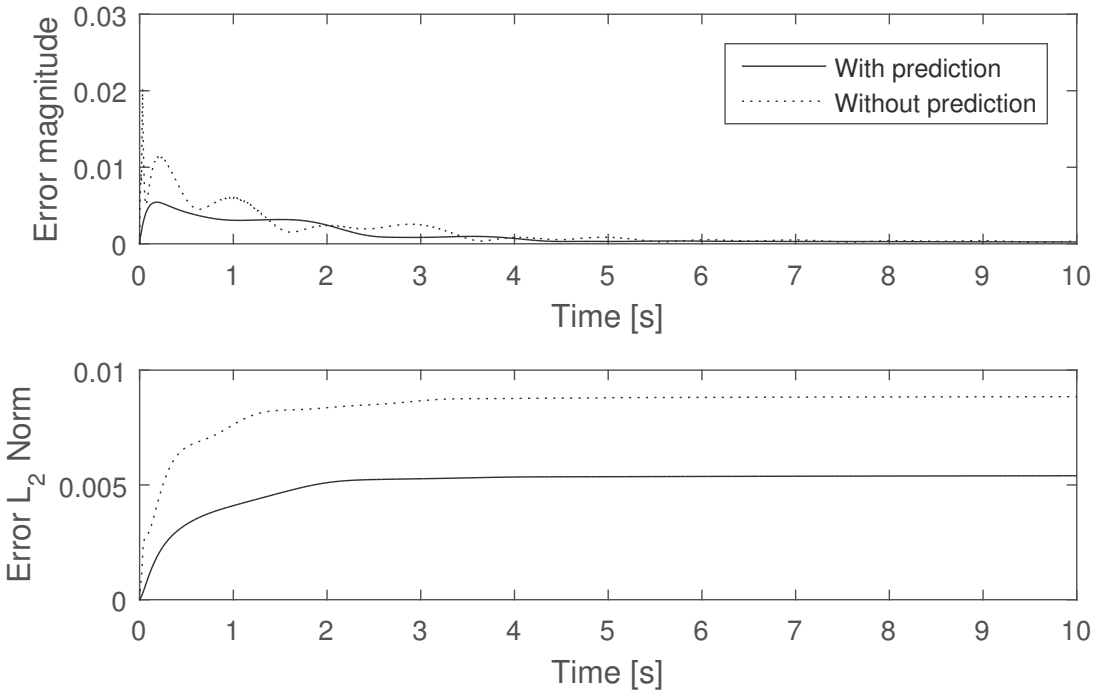


Figure 6.21: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 4)

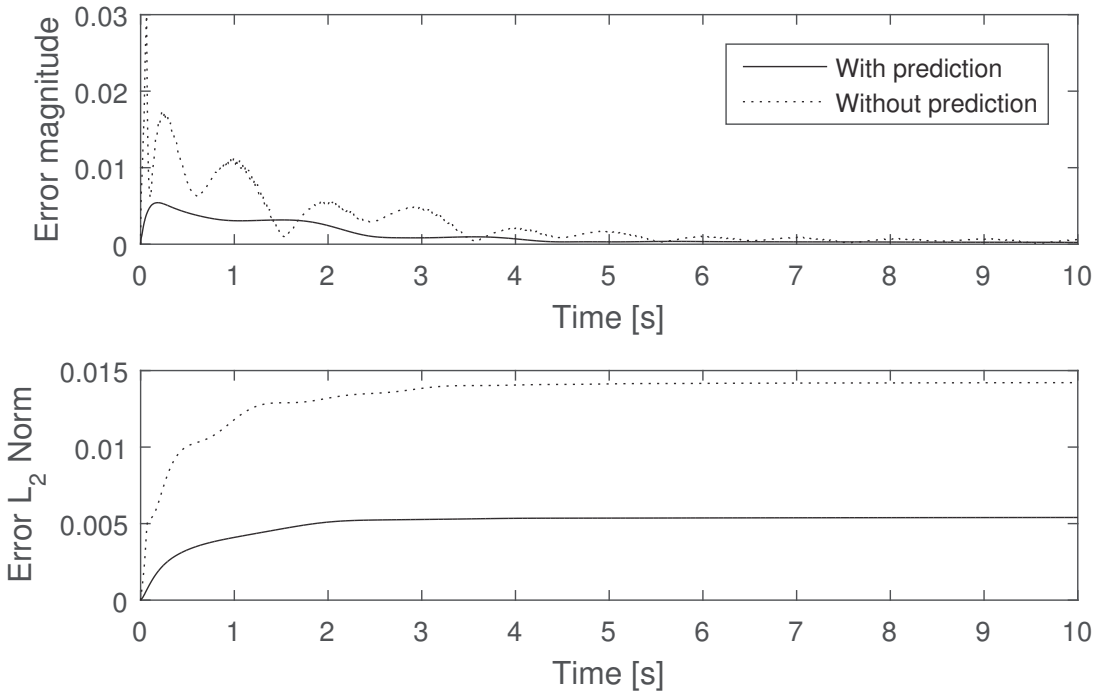


Figure 6.22: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 5)

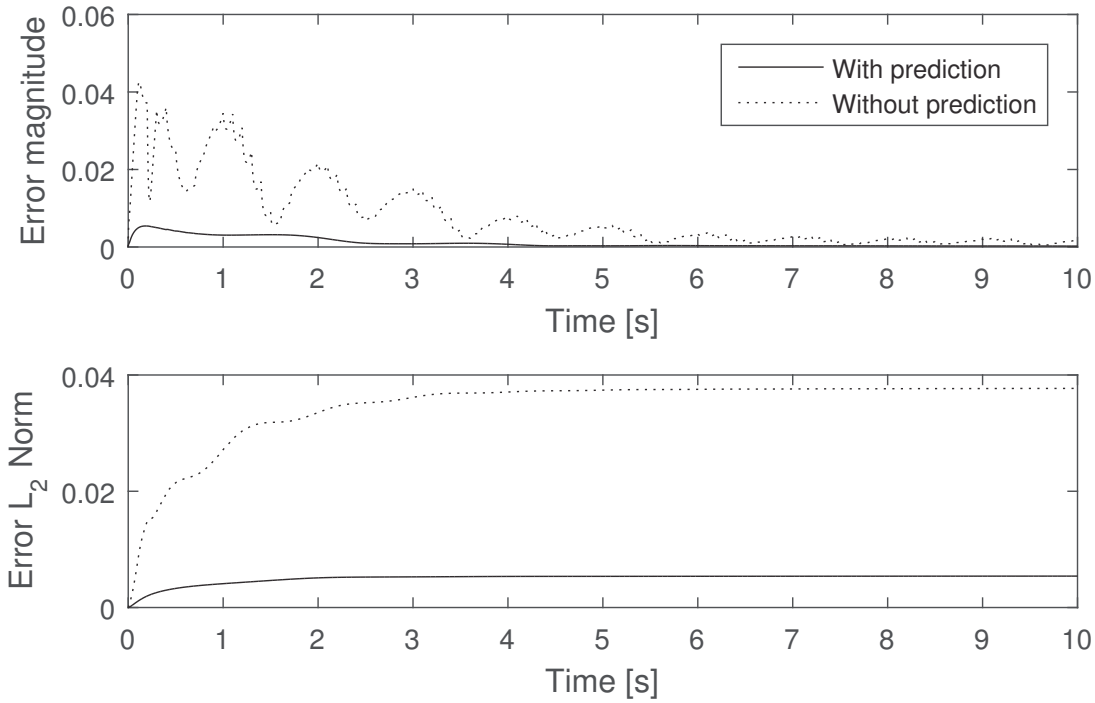


Figure 6.23: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 6)

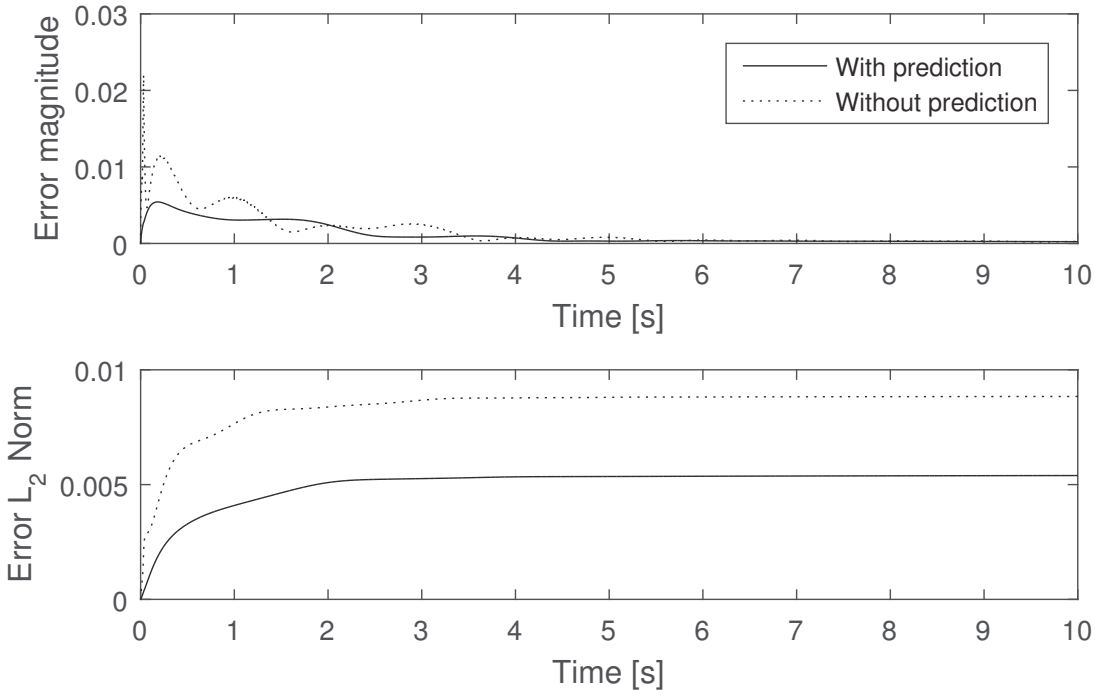


Figure 6.24: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 7)

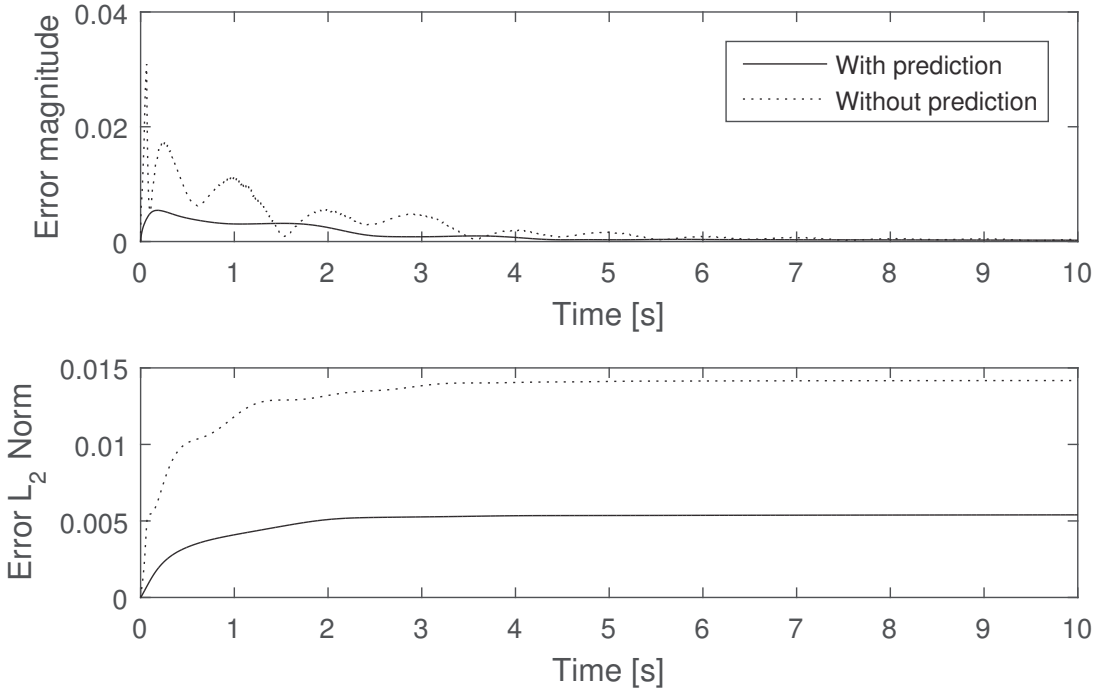


Figure 6.25: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 8)

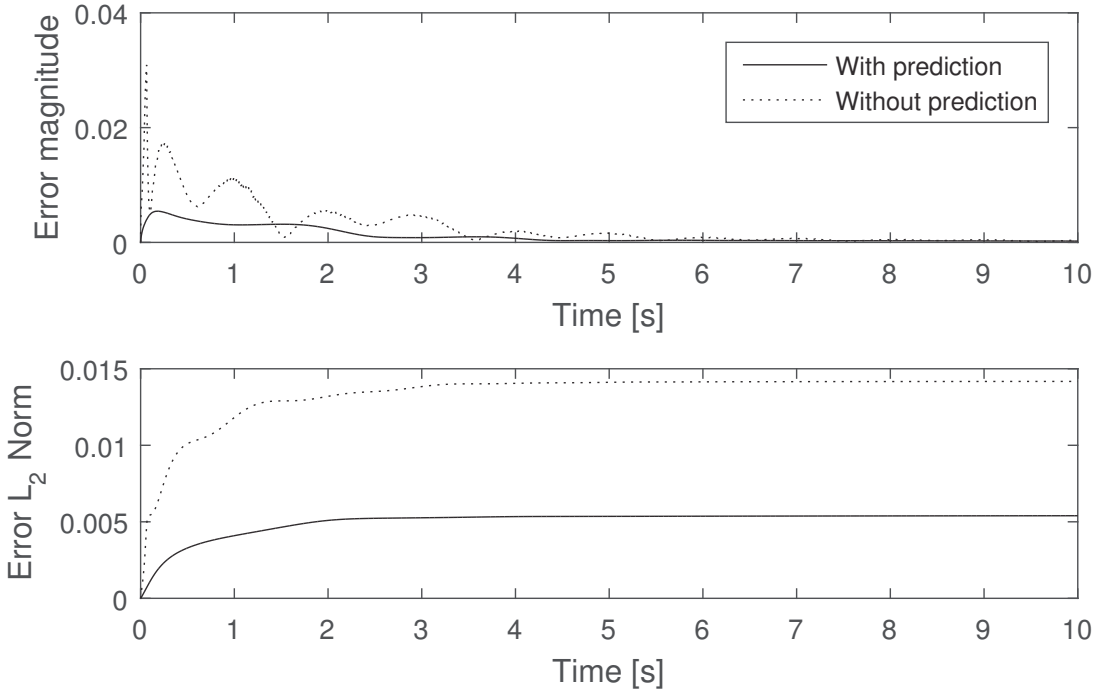


Figure 6.26: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment (Test Case 9)

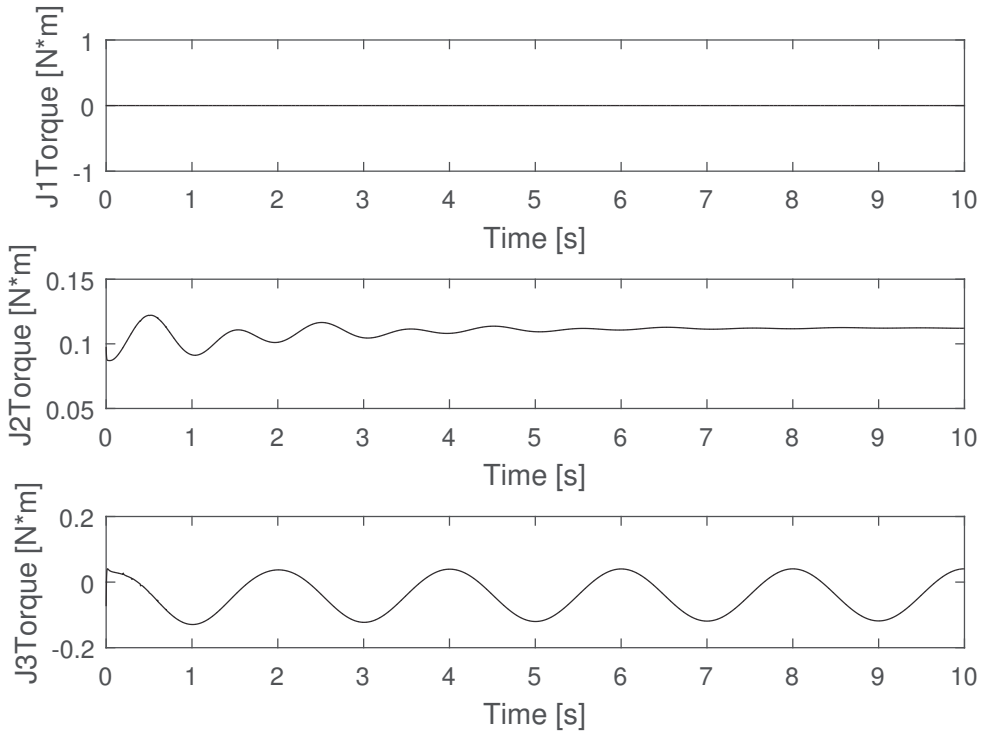


Figure 6.27: Control torque for a linear virtual environment with $k_e = 1500$ N/m and $T_d = 0.100$ s, with prediction

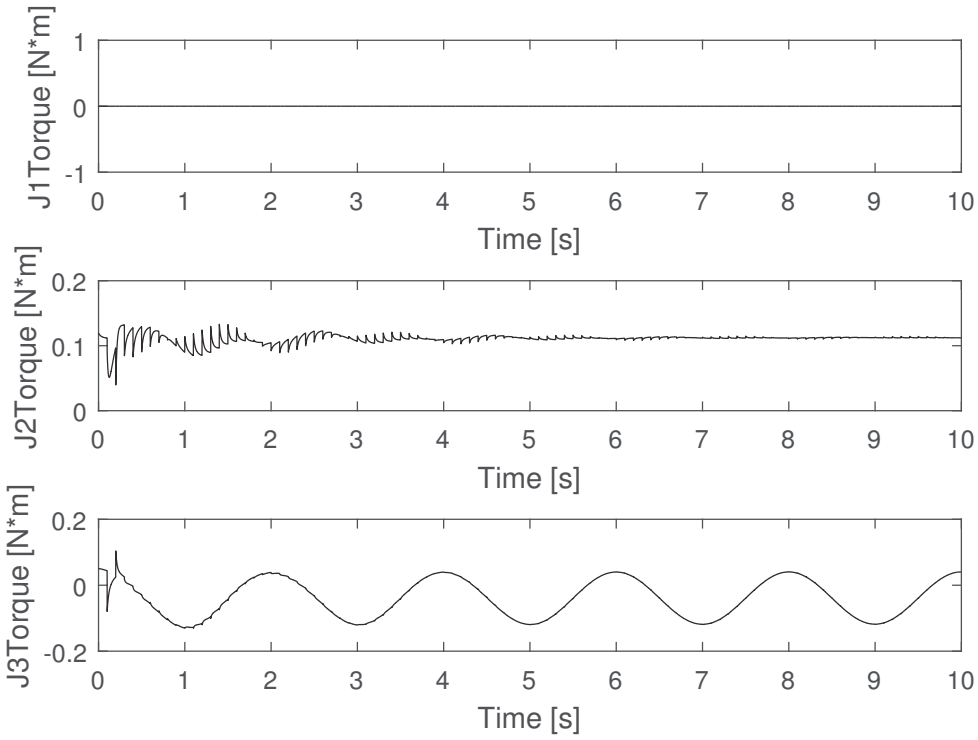


Figure 6.28: Control torque for a linear virtual environment with $k_e = 1500$ N/m and $T_d = 0.100$ s, without prediction

There are several reasons why k_t has such a significant effect. Firstly, when the virtual tool stiffness is nonzero it leads to an initial force on the virtual tool at time $t = 0$ since the tool does not start at the origin. The initial force causes a sudden acceleration of the virtual tool, and larger accelerations result in higher sampling error, as stated in (4.27). Secondly, the predictor design assumes that the virtual tool stiffness is known, therefore the more the virtual tool stiffness dominates the virtual tool dynamics the more of an advantage the predictor has in accurately predicting the signal.

6.2 Gain-Scheduled Predictor Results

This section shows the performance of a gain-scheduled predictor and compares it to a constant-gain predictor.

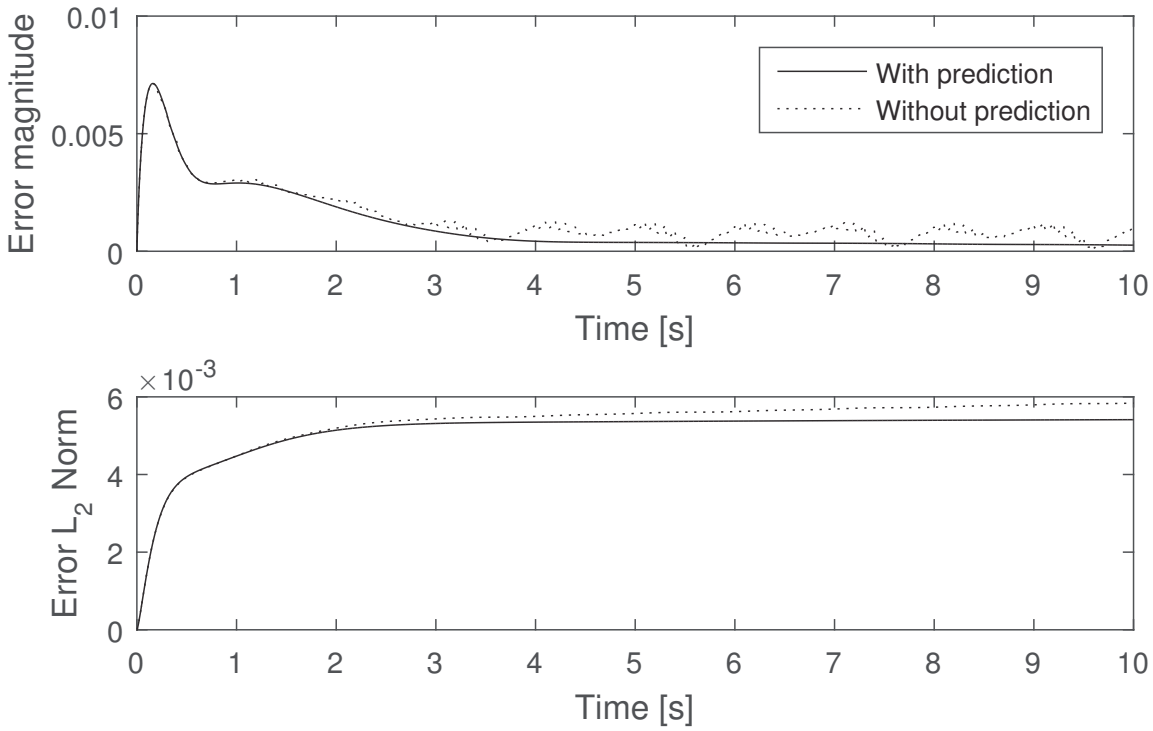


Figure 6.29: Total error magnitude and $\mathcal{L}_2$ norm for a linear virtual environment with $k_e = 1500$ N/m, $k_t = 0$ N/m and $T_d = 0.100$ s

6.2.1 Unknown Linear Virtual Environment with Delay and Sampling

The gain-scheduled predictor design was simulated for an unknown linear virtual environment with delay and sampling. Although the gain-scheduled predictor is intended to be used for nonlinear systems it is useful to compare its performance to the constant-gain predictor in the simpler case.

The simulation parameters are the same as in Section 6.1.4, except that $k_t = 0$. The virtual tool stiffness was removed to investigate a 'worst-case' scenario for the predictor. The predictor was designed using Theorem 5 in Chapter 4. The maximum rate of change of the scheduling variables ($\boldsymbol{\rho} = \hat{\boldsymbol{\theta}}_e$) is estimated by considering (4.44) and (4.46) when $\lambda_{max}(\Gamma_e) = \rho_M$,

$$\begin{aligned} |\dot{\boldsymbol{\rho}}|_{max} &\leq \frac{\rho_M}{1 + \nu \boldsymbol{\varphi}_{e,max}^T \rho_M \boldsymbol{\varphi}_{e,max}} \boldsymbol{\varphi}_{e,max} \boldsymbol{\varphi}_{e,max}^T (\boldsymbol{\theta}_{e,max} - \boldsymbol{\theta}_{e,min}), \\ &\leq \frac{1}{1/(\rho_M \|\boldsymbol{\varphi}_e\|_{max}^2) + \nu} (\boldsymbol{\theta}_{e,max} - \boldsymbol{\theta}_{e,min}). \end{aligned} \tag{6.2}$$

Therefore the initial parameters used to design the predictor gain are $\boldsymbol{\rho}_0 = [1100 \;\; 30]^T$ and $\dot{\boldsymbol{\rho}}_0 = [1.8 \times 10^5 \;\; 2000]^T$. The predictor was simulated for $k_e = 200$ N/m, $b_e = 30$

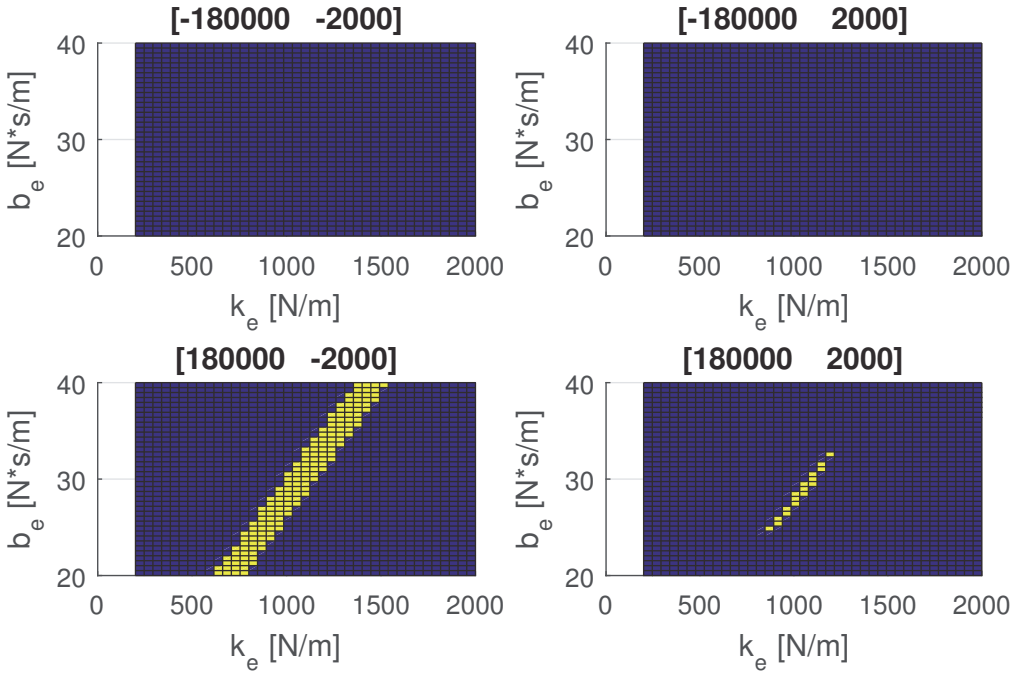


Figure 6.30: Stability region for the gain-scheduled predictor, where the yellow regions are guaranteed to be stable. Each plot is for a different set of $|\dot{\boldsymbol{\rho}}|_{max}$.

N·s/m, and $T = 0.100$ s. The gains for the gain scheduled predictor designed using Theorem 5 were $L_0 = [-7.31\times10^{-5} \;\; -2.5\times10^{-8}]^T$, $L_1 = [-2.61\times10^{-3} \;\; 2.75\times10^{-4}]^T$, and $L_2 = [-9.27\times10^{-4} \;\; -2.89\times10^{-5}]^T$. The gain for the constant-gain predictor designed using Theorem 3 was $L = [-194.8\times10^{-5} \;\; -4.89\times10^{-8}]^T$. The stability regions of the gain-scheduled predictor are shown in Fig. 6.30 and the stability region of the constant-gain predictor is shown in Fig. 6.31.

The gain-scheduled predictor has a much smaller guaranteed stability region, limited to a narrow band along two extremes of $|\dot{\boldsymbol{\rho}}|_{max}$. The gain-scheduled predictor has a smaller stability region because it attempts to find a single set of decision variables to satisfy the entire region, while the constant-gain predictor is stable as long as there is a solution for each parameter set. However, within the region that it is stable, the gain-scheduled predictor has a much smaller guaranteed error bound, suggesting the gain-scheduled predictor should result in better performance within that region.

Fig. 6.32 shows the gain for the gain-scheduled predictor adjusting over time as the virtual environment parameter estimate $\hat{\boldsymbol{\theta}}_e$ changes. The gain for the gain-scheduled

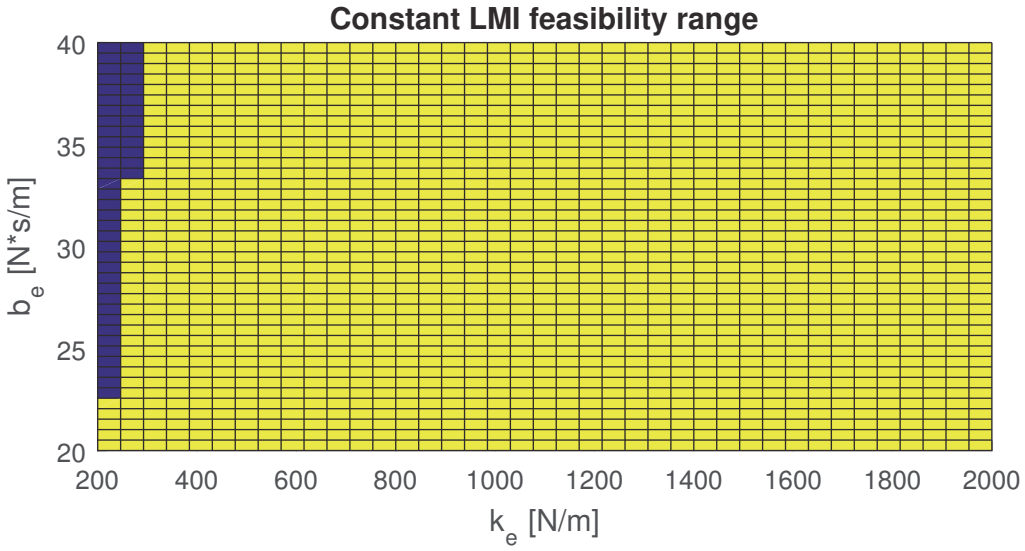


Figure 6.31: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable.

predictor is consistently smaller than the constant-gain case. A comparison of the prediction error for both predictors is shown in Fig. 6.33, and a comparison of the total error for both predictors is shown in Fig. 6.34. The difference in the total error $\mathcal{L}_2$ norm between the two predictors is only 1%, in favour of the gain-scheduled predictor, and the difference between their peak total error is less than 1%. In other words, the performance difference between the two is negligible. Both predictors result in a performance improvement compared to the system without a predictor, with the peak total error decreasing by 7% and the $\mathcal{L}_2$ norm decreasing by 26%.

6.2.2 Unknown Nonlinear Virtual Environment with Delay and Sampling

The gain-scheduled predictor design was simulated for an unknown nonlinear virtual environment with delay and sampling. The same parameters were used as in Section 6.2.1, with the following changes.

The virtual environment is described by,

$$F_e = F_k(x_e) + b_e\dot{x}_e, \tag{6.3}$$

where F_k is the nonlinear stiffness function described by (3.34). Two different environments were tested: Case A where $k_{e,1} = 370$ N/m, $k_{e,2} = 1980$ N/m, and $x_s = 0.0028$

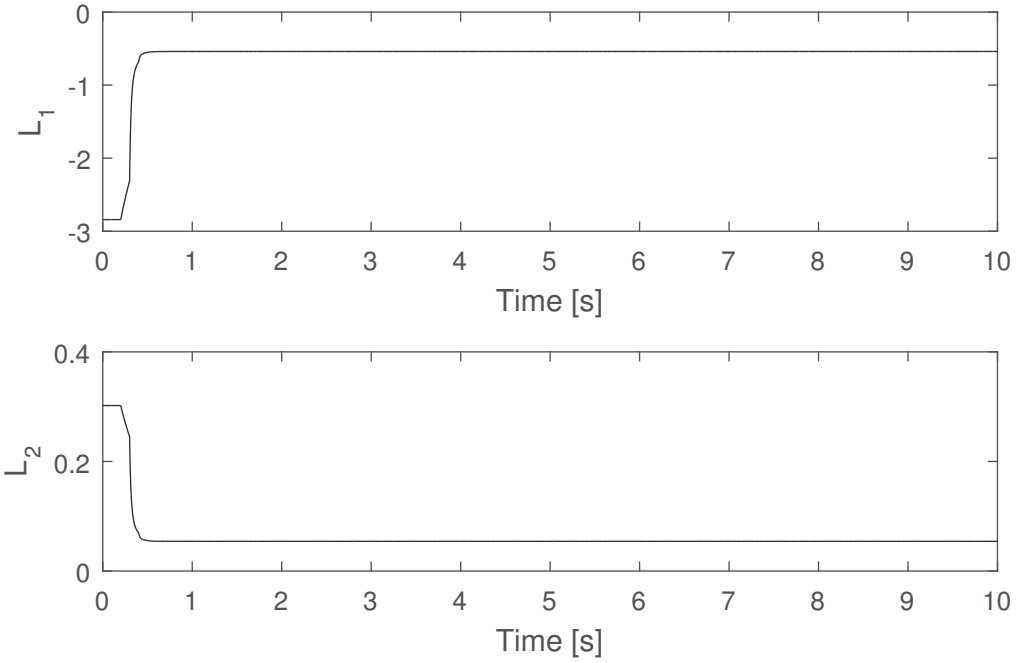


Figure 6.32: Gain for the gain-scheduled predictor adjusting according to $\hat{\boldsymbol{\theta}}_e$

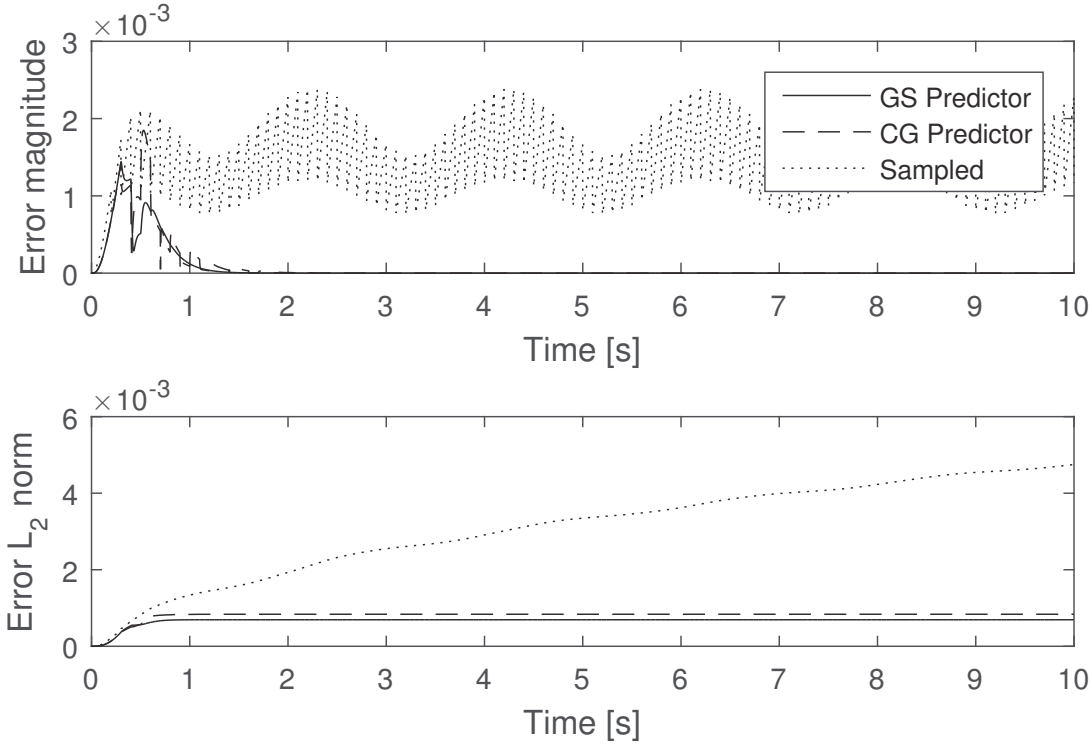


Figure 6.33: Predictor error magnitude and $\mathcal{L}_2$ norm for a gain-scheduled predictor and a constant-gain predictor for an LTI virtual environment

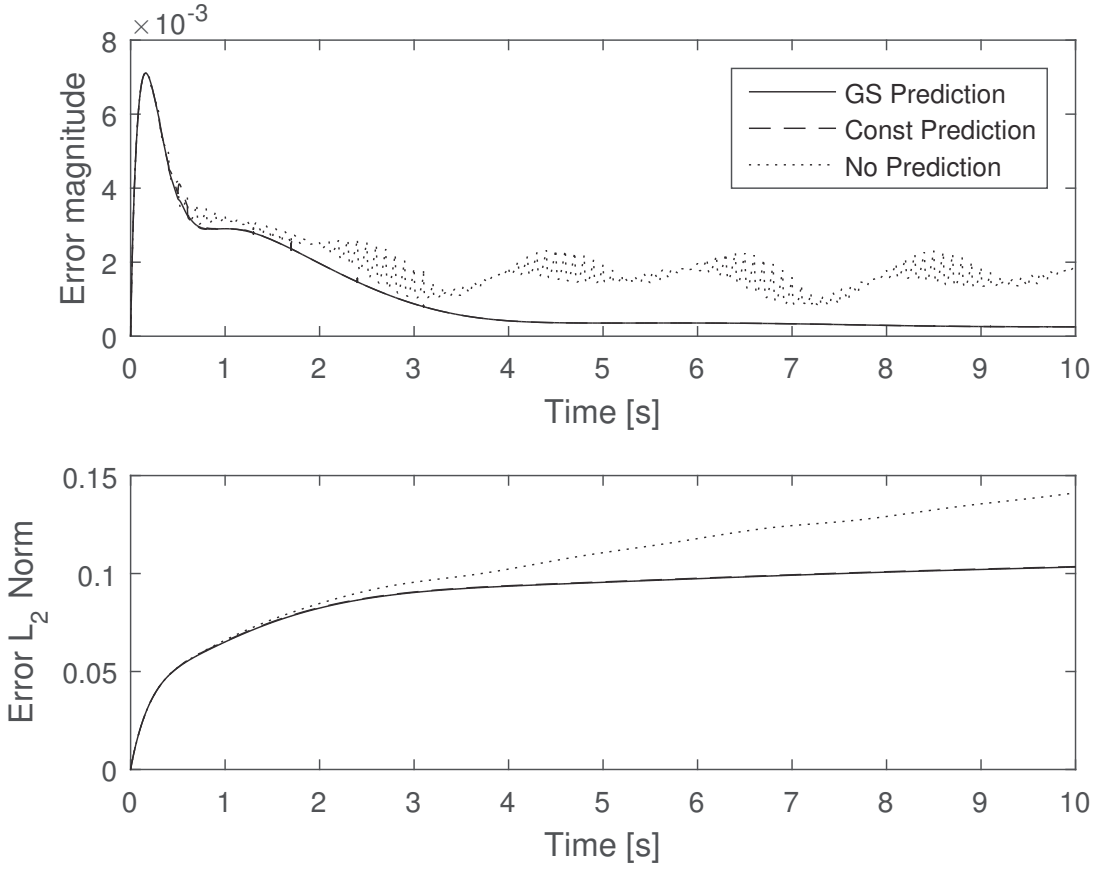


Figure 6.34: Total error magnitude and $\mathcal{L}_2$ norm for a gain-scheduled predictor and a constant-gain predictor for an LTI virtual environment

Table 6.6: Designed predictor gains for constant VE parameters

T_d (s)	0.016	0.032	0.100
L_0	[-2.2 × 10^{-3} -5.9 × 10^{-5}]	[6.7 × 10^{-5} 2.1 × 10^{-6}]	[-7.3 × 10^{-5} -2.5 × 10^{-7}]
L_1	[-3.5 × 10^{-2} -1.0 × 10^{-3}]	[-2.8 × 10^{-3} 6.0 × 10^{-4}]	[-2.6 × 10^{-3} 3.0 × 10^{-4}]
L_2	[-2.7 × 10^{-2} -8.7 × 10^{-4}]	[9.8 × 10^{-4} 2.2 × 10^{-5}]	[-9.3 × 10^{-4} -2.9 × 10^{-5}]
L_{const}	[-440.8 -12.1]	[-202.6 -5.13]	[-194.8 -4.89]

m; and Case B where $k_{e,1} = 450$ N/m, $k_{e,2} = 1000$ N/m, and $x_s = 0.005$ m. Case A and B are based on stiffness data of real human tissue [21]. In both cases $R = 20$ and $b_e = 30$ N·s/m. Case A and B are both tested for three different sampling times: $T_d = 0.0016$, 0.0032, and 0.100 s.

The linear environment model used in the predictor is changed to the alternate model described by (3.33). The parameter estimate is now $\hat{\boldsymbol{\theta}}_e = [\hat{k}_e \;\; \hat{b}_e \;\; \hat{c}_e]^T$. The bounds on the estimate of c_e are $-55 \leq \hat{c}_e \leq 0$ N. This does not affect the design of the predictor gain, it only affects the force estimation and parameter adaptation. Adding a third parameter to the parameter adaptation law makes it very difficult for the estimated parameters to converge to the actual values, especially since the parameters are changing regularly. Therefore b_e is assumed to be known by the predictor and is no longer estimated. Also, the forgetting factor α is increased to 20 to improve the parameter adaptation law's ability to track the frequently changing parameters. The period of the operator input force was increased to 4 s to help reduce the rate at which the virtual environment parameters change.

Table 6.6 shows the gains designed for the gain-scheduled predictor (L_0, L_1, and L_2) and for the constant-gain predictor for each sampling time. Just as in the previous section, the gain-scheduled predictor gains are much smaller than the constant-gain values, even after multiplying by the scheduling variables.

Fig. 6.35 to Fig. 6.40 show the stability regions for the gain-scheduled predictor and the constant-gain predictor. In each case the gain-scheduled predictor has a larger guaranteed stability region than the constant-gain predictor. Even though the stability ranges of both systems do not cover the range of parameters used in the simulation, they were simulated to see the effects of adding the predictor.

Fig. 6.41 shows the virtual environment parameters and the estimates of the parameters during the simulation. The estimate of the virtual environment parameters

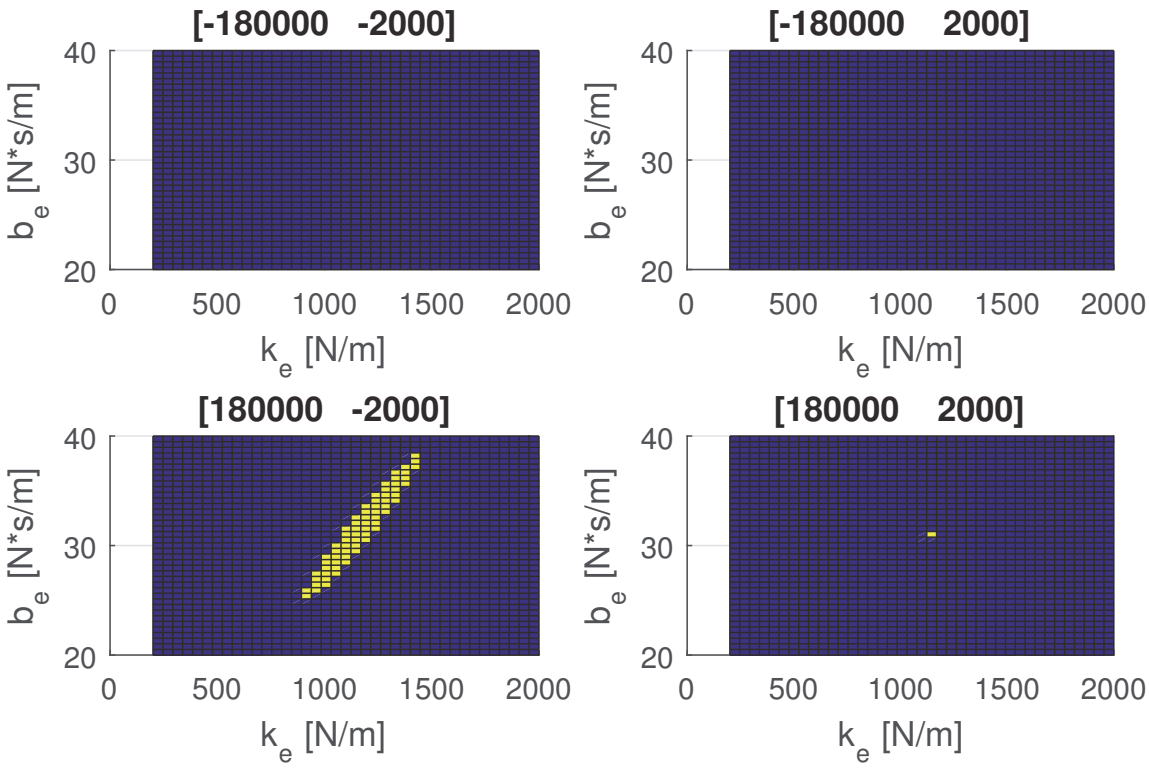


Figure 6.35: Stability region for the gain-scheduled predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.016$ s.

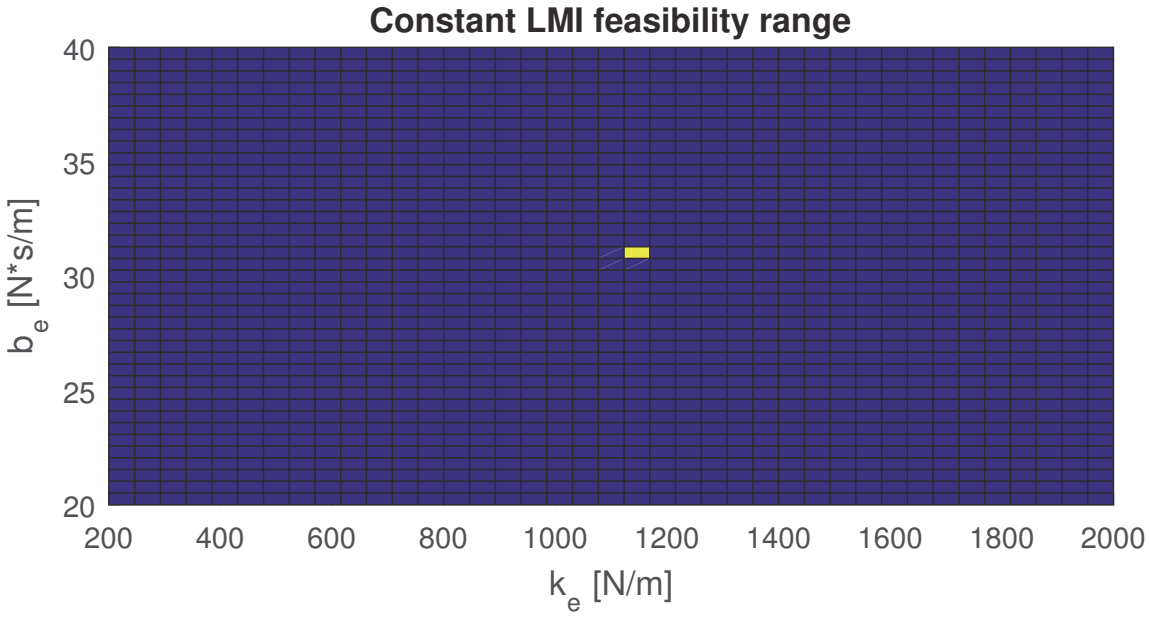


Figure 6.36: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.016$ s.

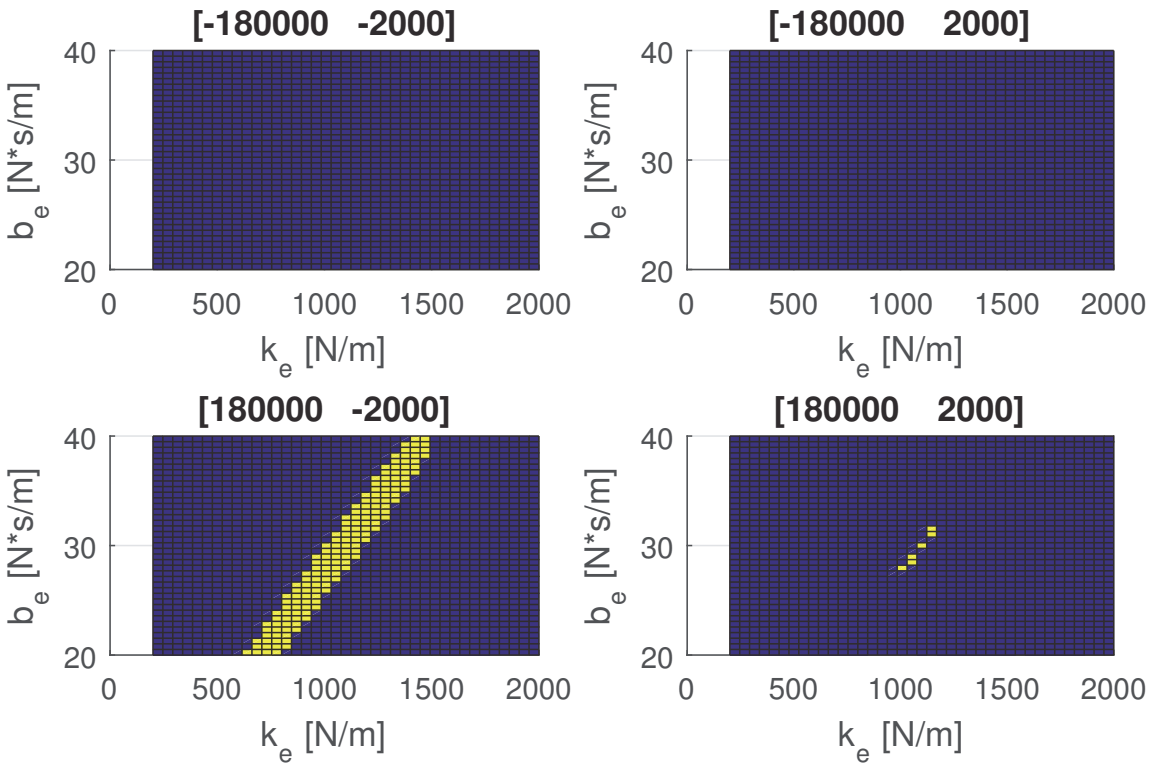


Figure 6.37: Stability region for the gain-scheduled predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.032$ s.

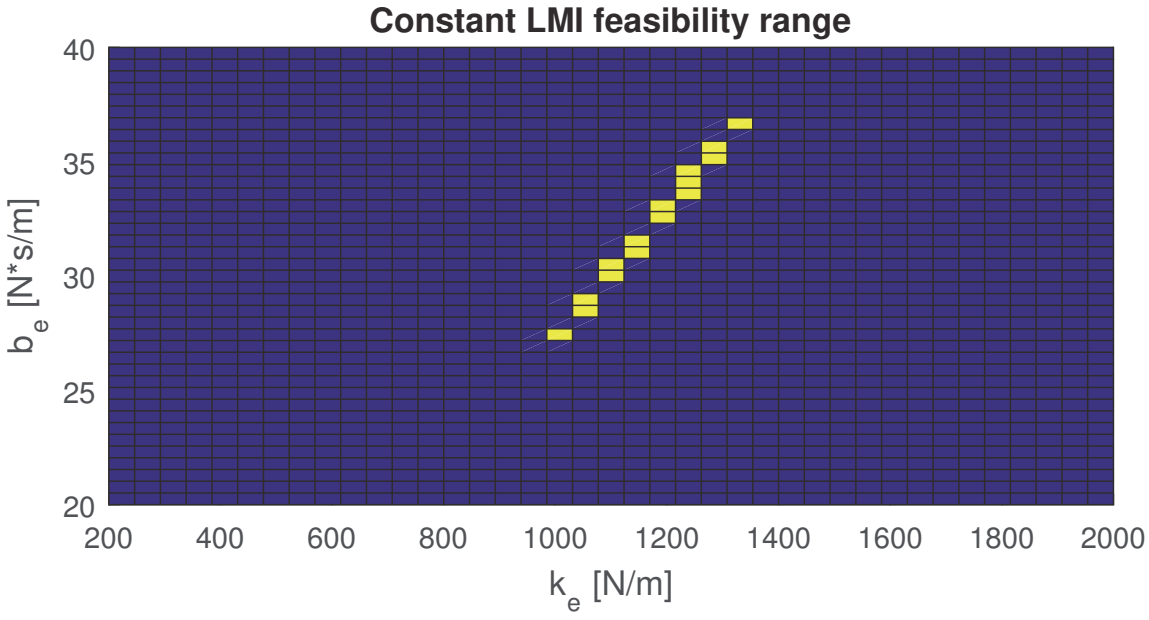


Figure 6.38: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.032$ s.

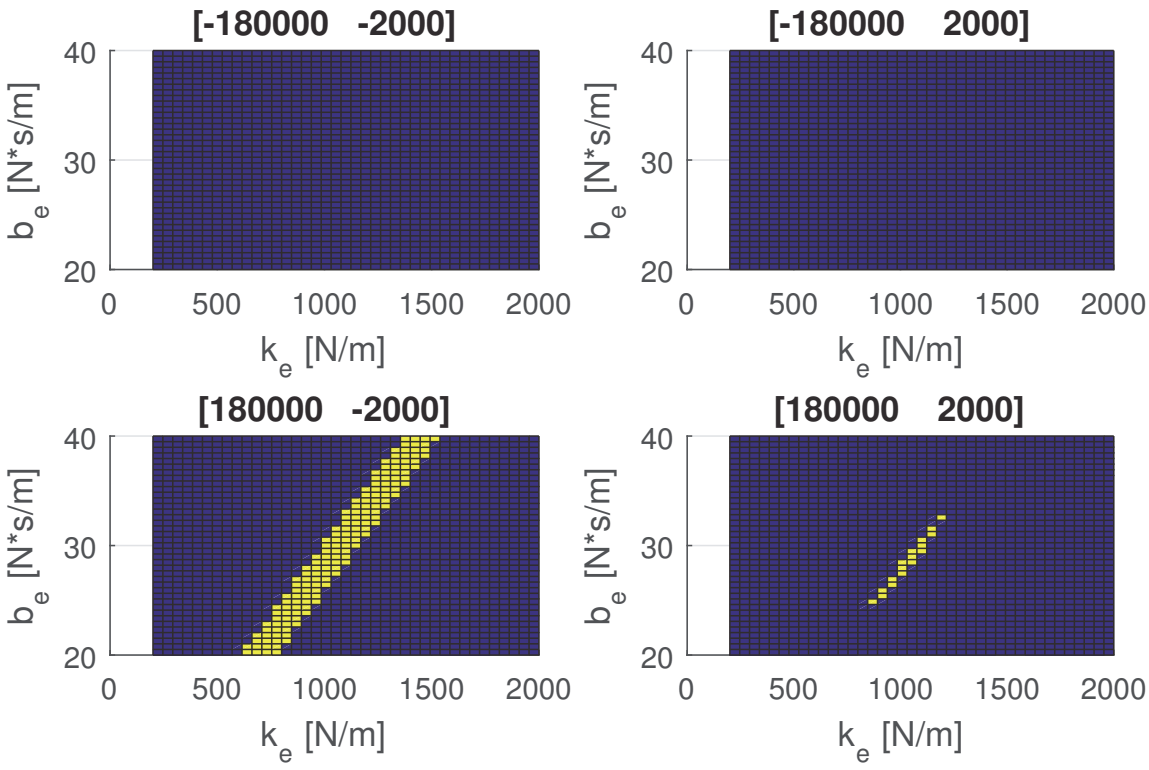


Figure 6.39: Stability region for the gain-scheduled predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.100$ s.

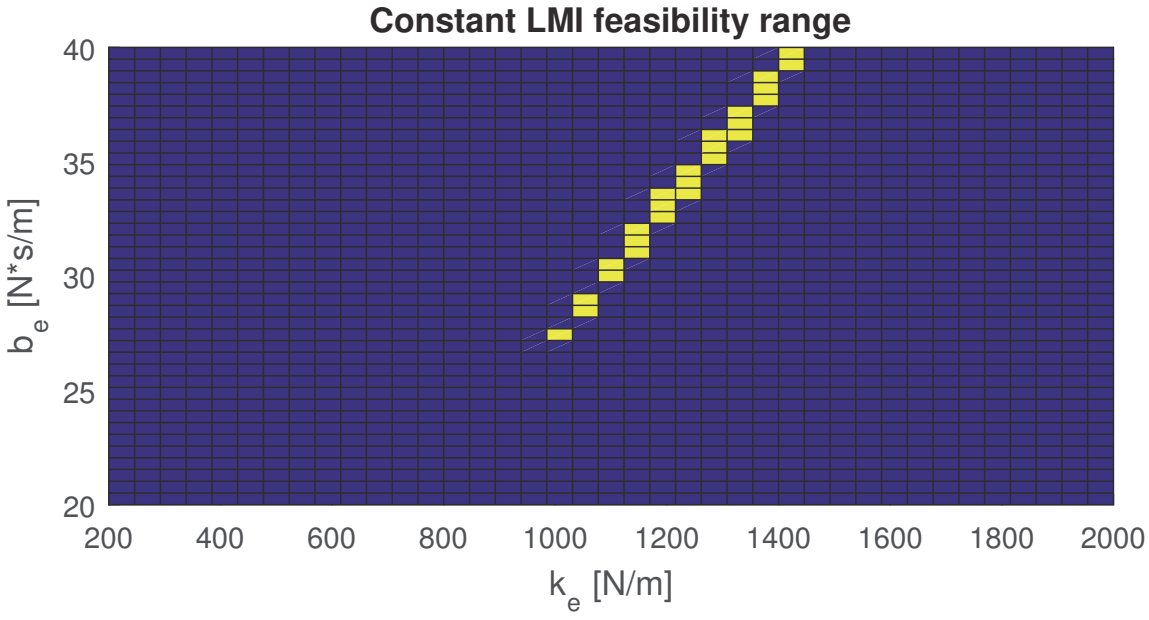


Figure 6.40: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.100$ s.

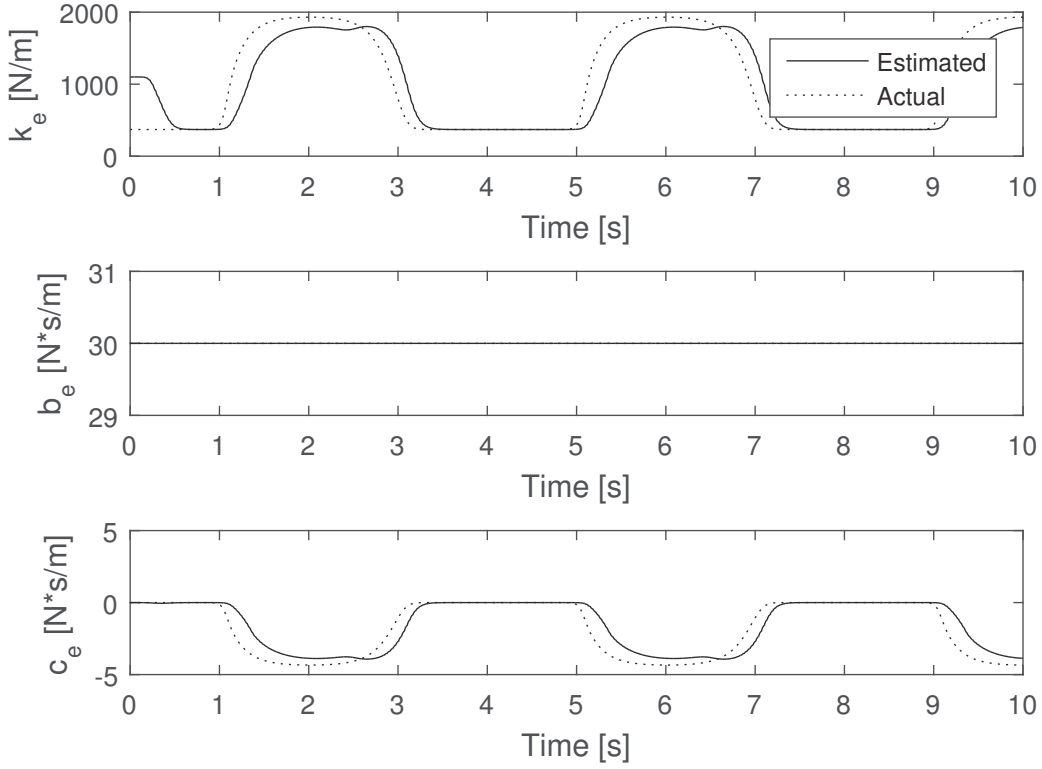


Figure 6.41: Estimated parameters for Case A, with $T_d = 0.016$ s

tended to lag by approximately 0.2 s, and only reached approximately 90% of the maximum value.

Table 6.7 shows the performance results for the gain-scheduled predictor applied to the nonlinear system. In all the simulations, the gain-scheduled predictor performed effectively identically to the constant-gain predictor. In all the simulations except for Case B when $T_d = 0.100$ s, the predictor did not significantly outperform the sampled signal.

Fig. 6.42 to Fig. 6.47 show the total error results for each sampling time for both Case A and B. When the sampling time is small there is practically no difference between the system with or without the predictors. However, when $T_d = 0.100$ s a clear difference emerges after the initial transient from the controller has passed.

Table 6.7: Summary of the performance of the gain-scheduled predictor for a nonlinear virtual environment

Case	Sampling Time (T_d, s)	Error Bound Difference	$\mathcal{L}_2$ Norm Difference	Improvement over CG Predictor?
A	0.016	-4×10^{-6} (-0.06%)	-0.34%	Negligible
	0.032	-1×10^{-5} (-0.15%)	-1.40%	Negligible
	0.100	-3×10^{-6} (-0.05%)	-6.07%	Negligible
B	0.016	-3×10^{-6} (-0.05%)	-0.66%	Negligible
	0.032	-9×10^{-6} (-0.13%)	-2.09%	Negligible
	0.100	-1×10^{-6} (-0.02%)	-11.8%	Negligible

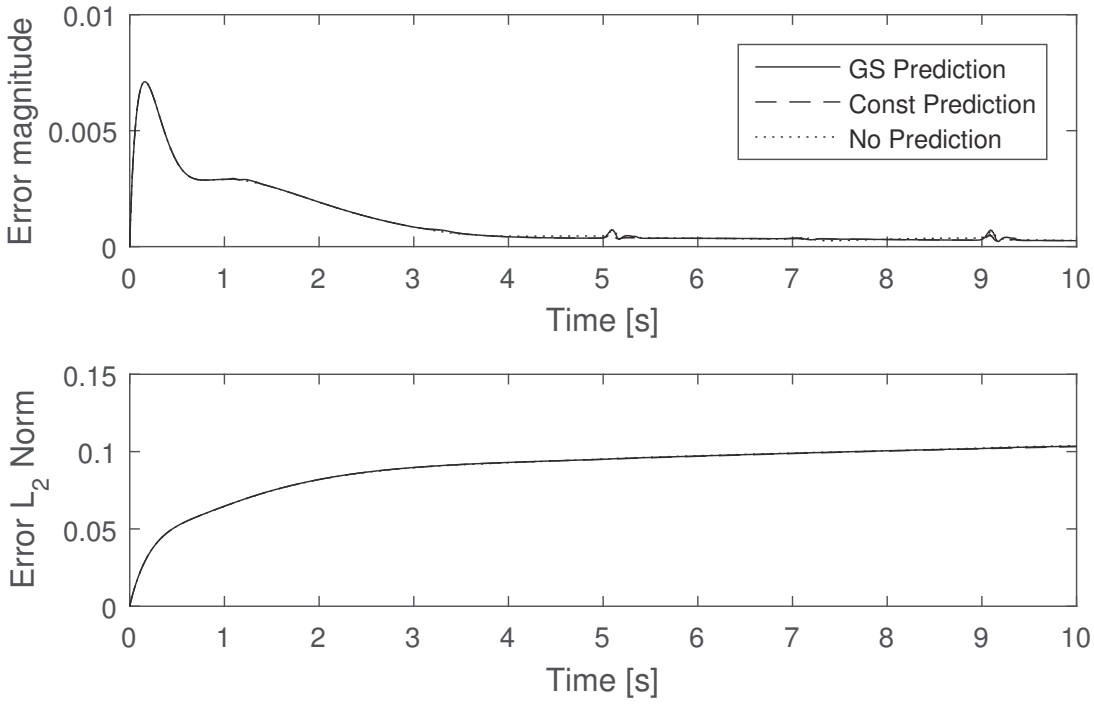


Figure 6.42: Total error magnitude and $\mathcal{L}_2$ norm for Case A with $T_d = 0.016$ s

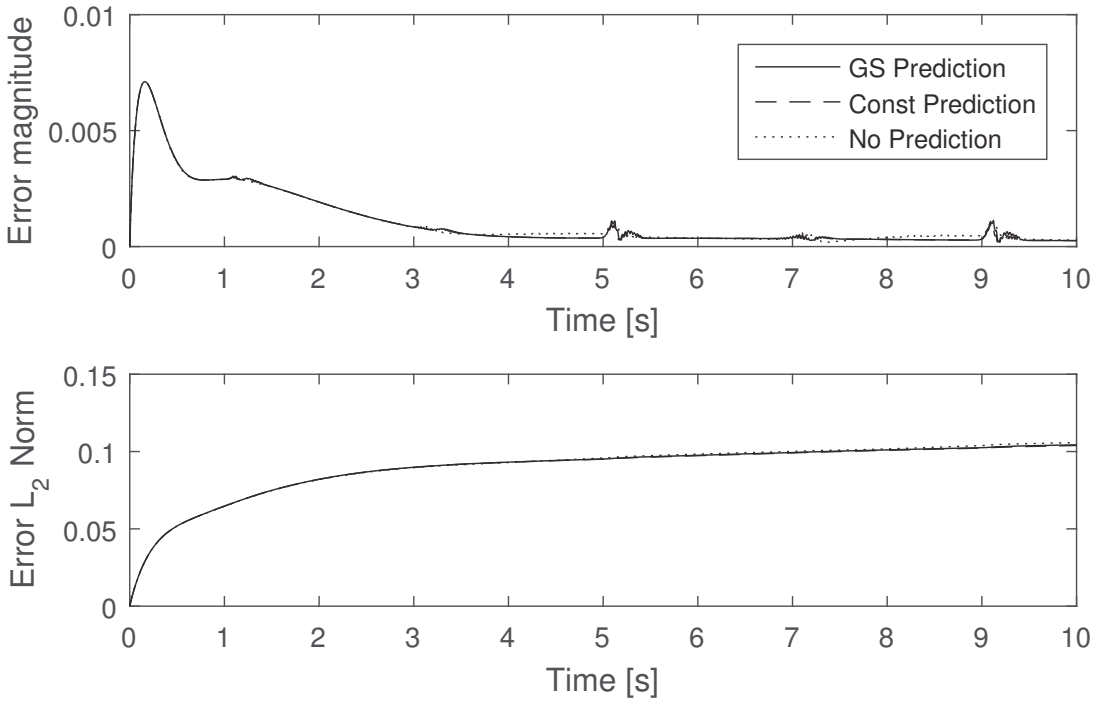


Figure 6.43: Total error magnitude and $\mathcal{L}_2$ norm for Case A with $T_d = 0.032$ s

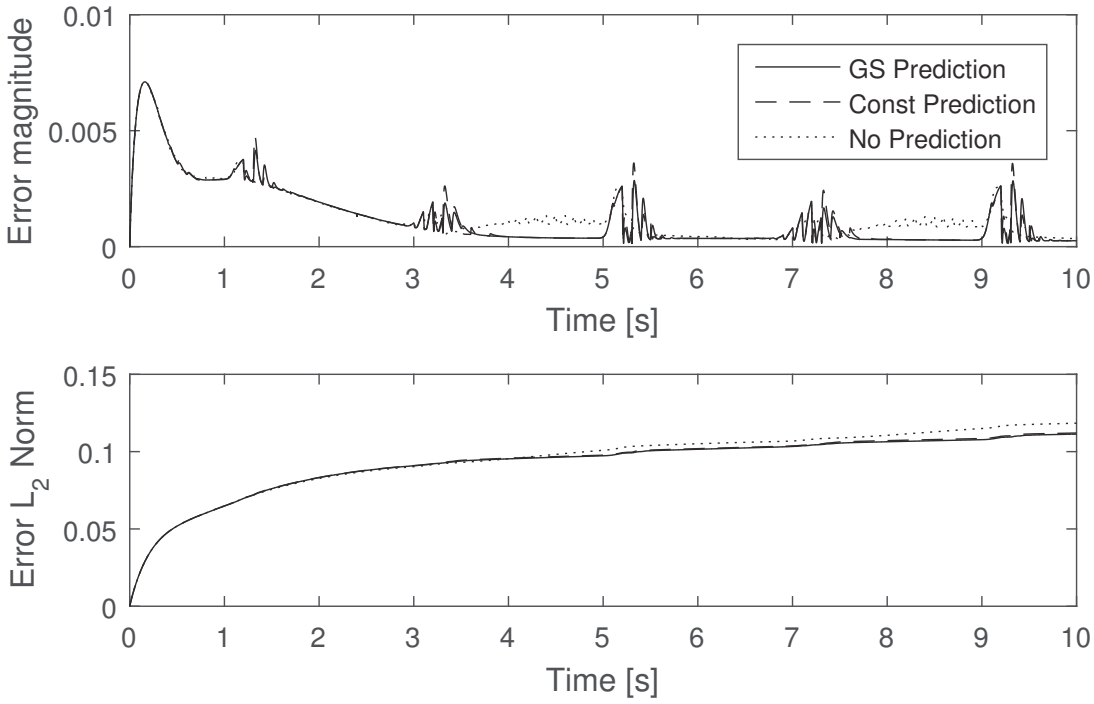


Figure 6.44: Total error magnitude and $\mathcal{L}_2$ norm for Case A with $T_d = 0.100$ s

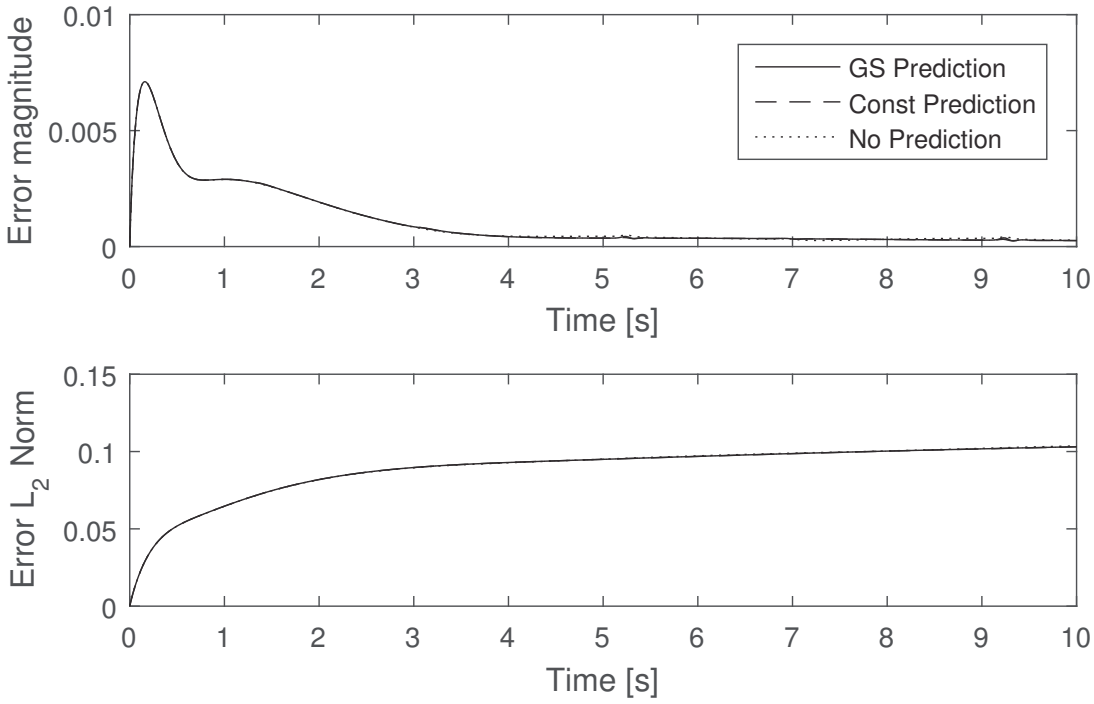


Figure 6.45: Total error magnitude and $\mathcal{L}_2$ norm for Case B with $T_d = 0.016$ s

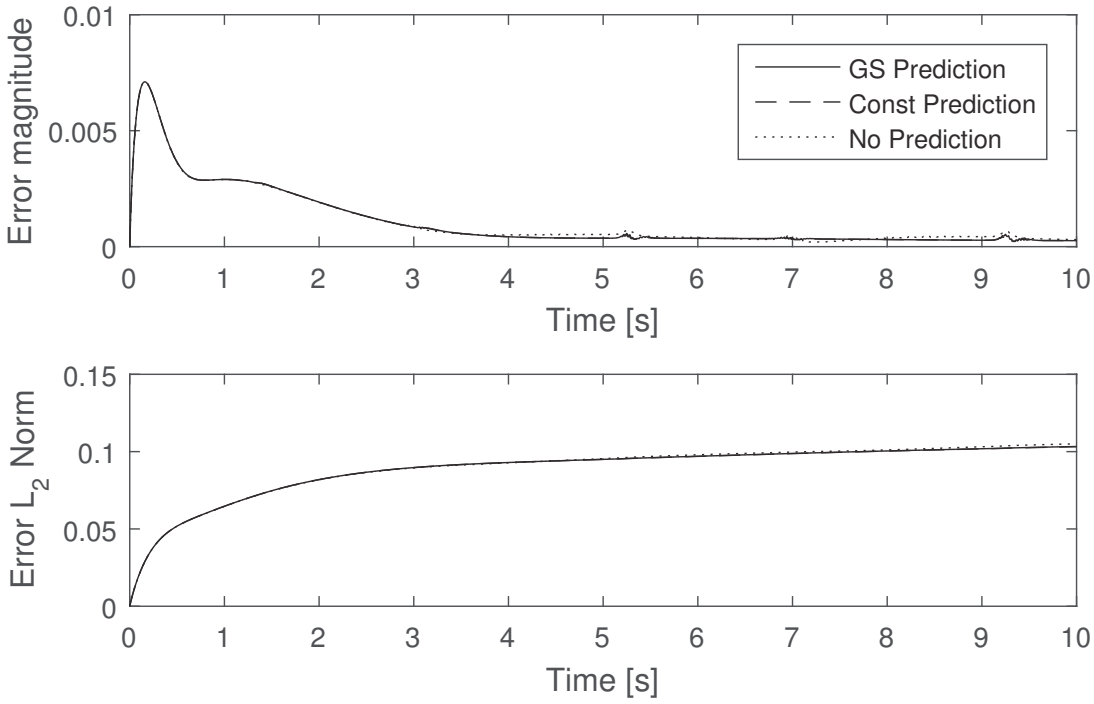


Figure 6.46: Total error magnitude and $\mathcal{L}_2$ norm for Case B with $T_d = 0.032$ s

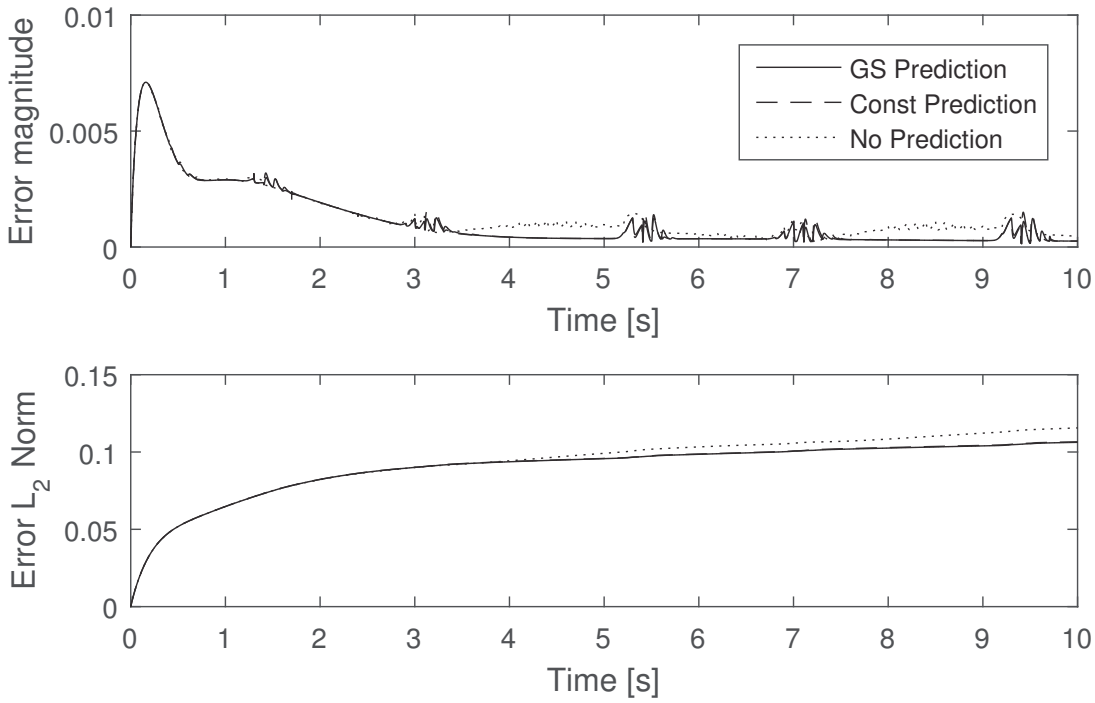


Figure 6.47: Total error magnitude and $\mathcal{L}_2$ norm for Case B with $T_d = 0.100$ s

6.3 Discussion

Based on the simulation results, the predictor shows some promise for reducing the error introduced by sampling and delay. For the cases where the virtual environment is linear but unknown the predictor leads to large increases in performance. However, for the unknown nonlinear environment the results are much less impressive. In the nonlinear virtual environment cases the sampling time must be equal to or larger than 0.1 s for the predictor to provide any noticeable benefit.

It is also important to consider that the predictor has many free parameters that may be tuned. For the gain-scheduled predictor, one may choose different values of a_2, a_3, γ, $k_{e,0}$, $b_{e,0}$, and $\dot{\boldsymbol{\rho}}_0$. The effects of some of those values have been investigated for the simpler cases, however given the fact that the LMI in (4.51) is not affine in any of the given parameters except $\dot{\boldsymbol{\rho}}_0$, it may be incorrect to extrapolate the results from the simpler cases. A more successful LPV approach could use a numerical solver to attempt to optimize either the stability range or the error bound of the gain-scheduled predictor over the parameter range.

It is worth noting from the comparison of the gain-scheduled and constant-gain predictors that the parameter adaptation law dominates the performance of the predictor. Both predictors had gains that were different by an order of magnitude, yet their performance was nearly identical and tended to improve as the parameter estimation improved. Also, the performance of the haptic controller tends to dominate the performance near the start of the simulation when the virtual tool is experiencing the most acceleration. It may be more beneficial to the transparency to focus on improving the controller and parameter adaptation law response over tuning the predictor gains.

Chapter 7

Experimental Results

7.1 Constant-Gain Predictor Results

As mentioned in Section 5.3.3, the nonlinearity and cross-talk in the force sensor is high, to the point where using the force sensor with the controller leads to instability. Therefore, in order to experimentally validate the predictor system, the force sensor was used to measure a force signal which would be 'played back' during testing. The recorded force serves as the input to the predictor, while the haptic controller receives a signal of $F_h = 0$. This is approximately equivalent to having a perfect force sensor that allows the controller to completely cancel the operator input force. The measured force was generated by hand and is shown in Fig. 7.1 after removing the Y and Z components and scaling the force to be within the ± 10 N. This exceeds the limits of the Phantom Omni® torque, but since the force is not actually being applied to the Omni it is acceptable. 10 N was selected because smaller forces lead to displacements that are too small for the Omni to measure for high-stiffness environments.

The environment was selected as a wall parallel to the Y-Z plane, located at the initial X axis position of the Omni. Since the operator and environment forces are only in the X axis, the predictor was set up to predict only the X axis trajectory, while Y and Z were fed directly through to the inverse kinematics without sampling or delay.

A constant predictor gain was designed for each test. Each experimental case involved two tests: one where the haptic system was operated using the predictor, and the other where the haptic system used the delayed and sampled signal from the virtual environment. The controller ran at 500 Hz.

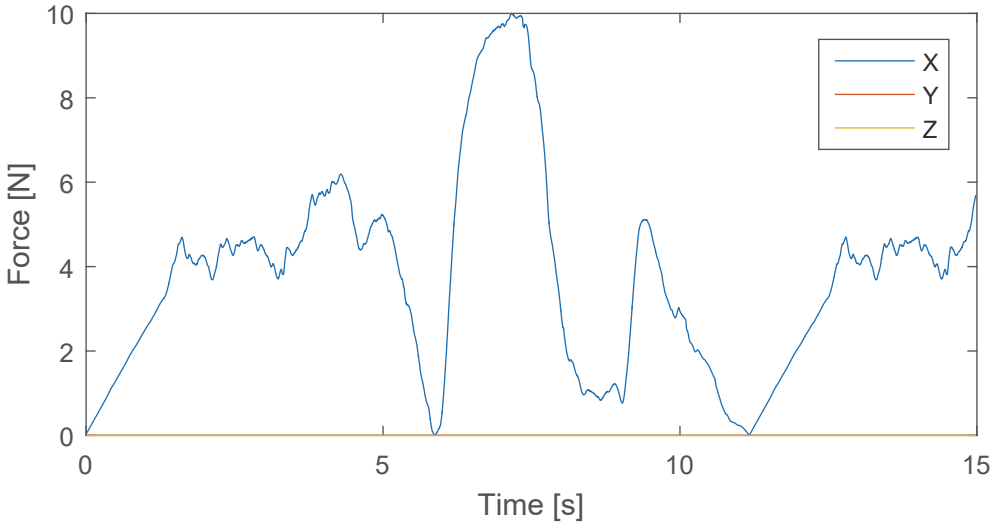


Figure 7.1: Operator force input F_h after scaling and other adjustments.

7.1.1 Unknown Linear Virtual Environment with Delay and Sampling

The predictor was tested for constant VE parameters. Nine cases were tested, with a different combination of VE stiffness and sampling rate for each case. Three different stiffness values, soft ($k_e = 200$ N/m), medium ($k_e = 800$ N/m, and hard ($k_e = 1500$ N/m), and three different sampling rates, fast ($T = 0.016$ s), medium ($T = 0.032$ s), and slow ($T = 0.100$ s) were selected. The damping was the same for all three tests ($b_e = 30$ N·s/m). The initial VE parameter estimate of the predictor was $\hat{k}_e(0) = 600$ N/m and $\hat{b}_e(0) = 20$ N·s/m.

The initial adaptation gain matrix was $\Gamma_e(0) = \text{diag}(1\times10^9,\ 2.67\times10^6)$. The adaptive controller gains are given in (5.6) and (5.7). The initial Phantom Omni® parameter estimates are given in Table 5.2. The rest of the experiment parameters are given in Table 7.1. Theorem 3 was used to design the predictor gains, with the nominal environment parameters equal to $\hat{\boldsymbol{\theta}}_e(0)$. The designed predictor gains are given in Table 7.2. Since the nominal VE parameters used in the gain design are the same for all nine cases the predictor gains only change according to the sampling time.

Table 7.3 shows the performance of the system with prediction versus the system without prediction The error bound difference and the $\mathcal{L}_2$ norm difference are calculated based on (3.36), (3.37), and (3.38). For both performance metrics, a negative value indicates a decrease in the total error and therefore an improvement in the

Table 7.1: Virtual environment and predictor parameters used in the linear virtual environment experiments

Parameter	Value	Parameter	Value
m_t (kg)	0.5	k_t (N/m)	0
b_t (N s/m)	0.1	γ	100
a_2	0.4694	a_2	0.008980
ν	0.01	α	1
ρ_M	5×10^9		

Table 7.2: Designed predictor gains for constant VE parameters

Cases	Sampling Time (T, s)	Predictor Gains L_1	L_2
1, 4, 7	0.016	−241.94	−8.50
2, 5, 8	0.032	−144.47	−2.01
3, 6, 9	0.100	−108.75	−1.44

transparency of the system.

As shown in Table 7.3, the system with prediction generally outperformed the system without prediction, except for Case 1. The predictor's relative performance improved as the sampling time increased, which was expected. As the sampling time increases the error introduced by sampling increases significantly, both because the state is able to change by a large amount in a single sampling interval and because the sampled signal is unable to represent the higher frequencies in the original signal. The relative performance decreases as the stiffness increases. Increased stiffness leads to smaller displacements, therefore static friction has a much larger effect on the controller's

Table 7.3: Summary of the performance of the predictor for constant VE parameters

Case	Stiffness (k_e, N/m)	Sampling Time (T_d, s)	Error Bound Difference	$\mathcal{L}_2$ Norm Difference
1	200	0.016	+0.0033 (+13.1%)	+0.07%
2		0.032	−0.0093 (−26.4%)	−7.4%
3		0.100	−0.0598 (−68.5%)	−42.0%
4	800	0.016	−0.0027 (−13.4%)	−4.0%
5		0.032	−0.0051 (−24.1%)	−7.1%
6		0.100	−0.0148 (−49.2%)	−19.7%
7	1500	0.016	−0.0014 (−8.8%)	−1.2%
8		0.032	−0.0025 (−15.9%)	−5.9%
9		0.100	−0.0042 (−24.1%)	−10.4%

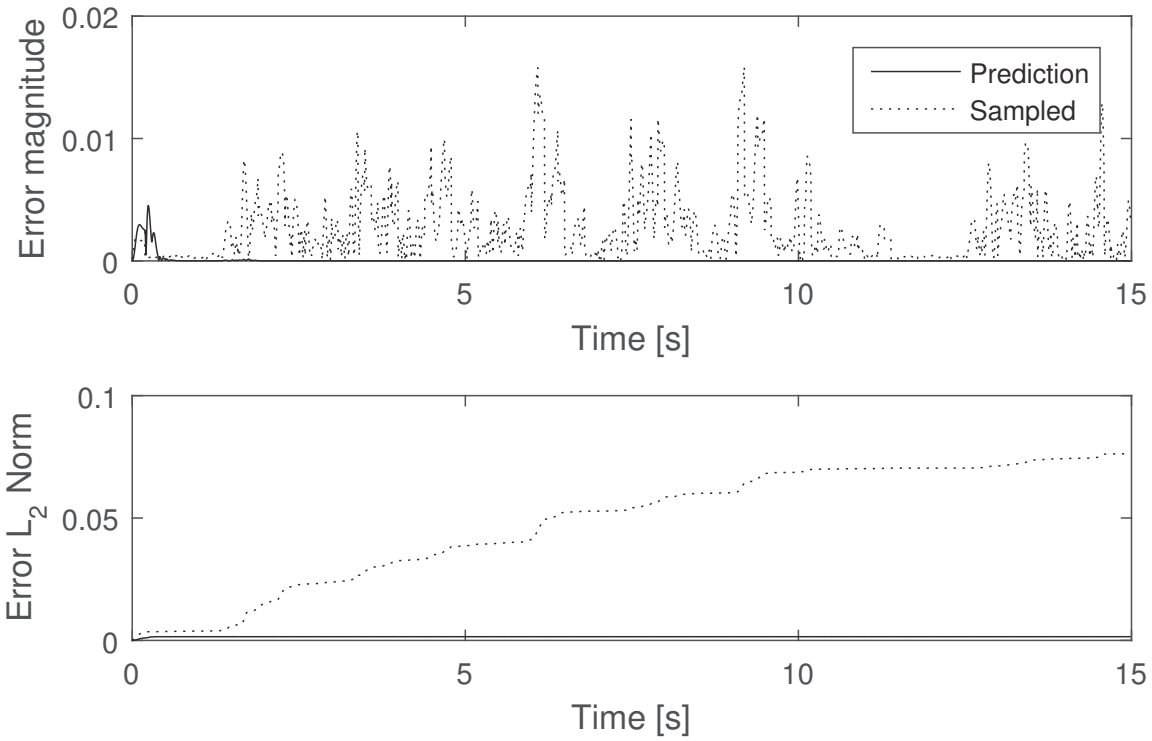


Figure 7.2: Predictor error compared to sampling error for the simplified system (Case 9).

ability to track the desired signal and the controller dynamics dominate the response.

For each case the predictor reduced the prediction error to near zero in less than one second far outperforming the sampled signal. The worst case is shown in Fig. 7.2. Once the estimated VE parameters converge to the actual VE parameters the predicted trajectory should track the ideal trajectory perfectly.

The desired trajectory was not as well followed compared to the simulations. For the system with the predictor, Fig. 7.3 shows the end-effector trajectory and the error between the end-effector trajectory and the predicted trajectory ($\boldsymbol{x} - \hat{\boldsymbol{x}}_e$). As mentioned in the Chapter 5, a significant amount of error is introduced by static friction and modelling errors.

Fig. 7.4 to Fig. 7.12 show the error magnitude and $\mathcal{L}_2$ norm of the error for each case. As the sampling time increases the peaks in the error magnitude for the haptic system without prediction increase more than for the system with prediction. The predictor seems to have the greatest effect in decreasing the peak error, when the system reaches a steady state the error with and without the predictor is generally

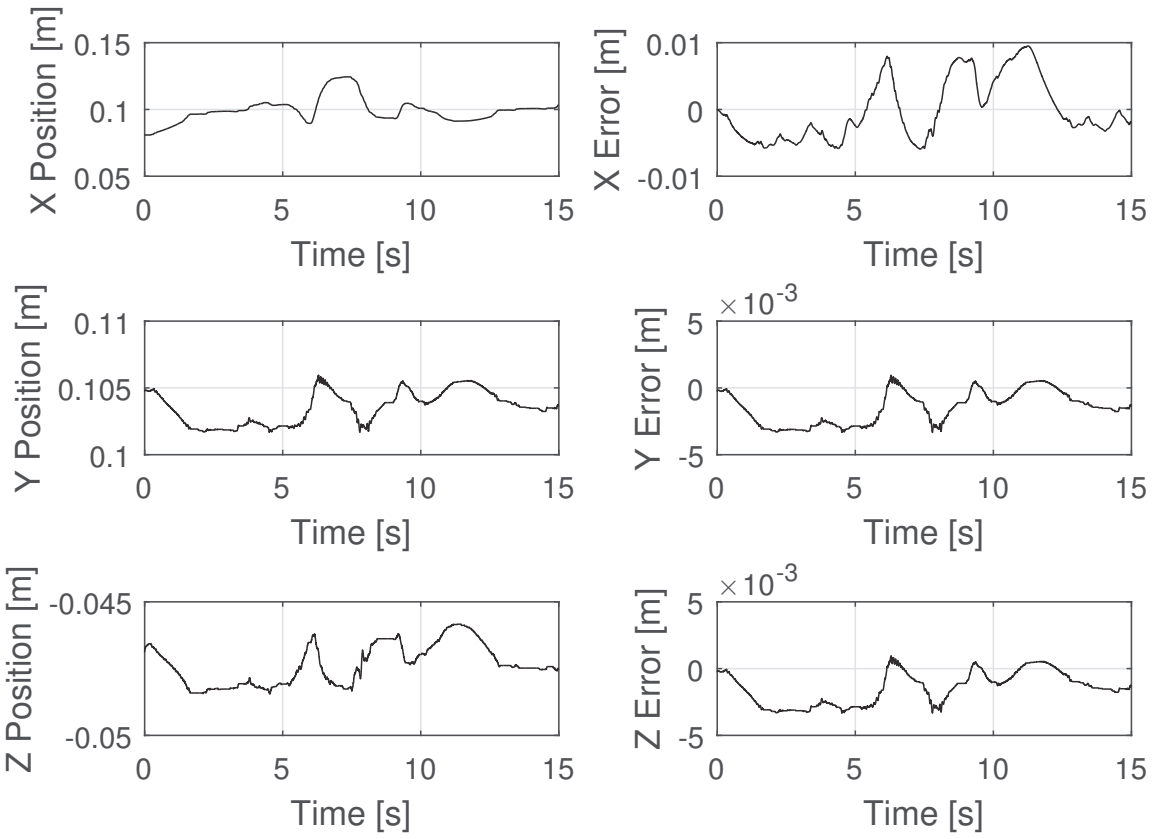


Figure 7.3: End-effector trajectory and tracking error for the simplified system (Case 3).

similar.

7.1.2 Unknown Nonlinear Virtual Environment with Delay and Sampling

The system was tested using a VE with constant damping but nonlinear stiffness. The constant damping was $b_e = 30$ N·s/m. The stiffness was governed by $k_e = kx_e^n$ where $k = 1500$ and $n = 1.3$, based on the Hunt-Crossley model. The time-varying part of the Hunt-Crossley model was not used since the parameter adaptation law is not able to adapt to two rapidly changing parameters. The rest of the simulation parameters, control and predictor gains were the same as in the previous section. Three cases, following the numbering from the previous section, were tested. Case 10, 11, and 12 have the same virtual environment but sampling times of 0.016 s, 0.032 s, and 0.100 s, respectively.

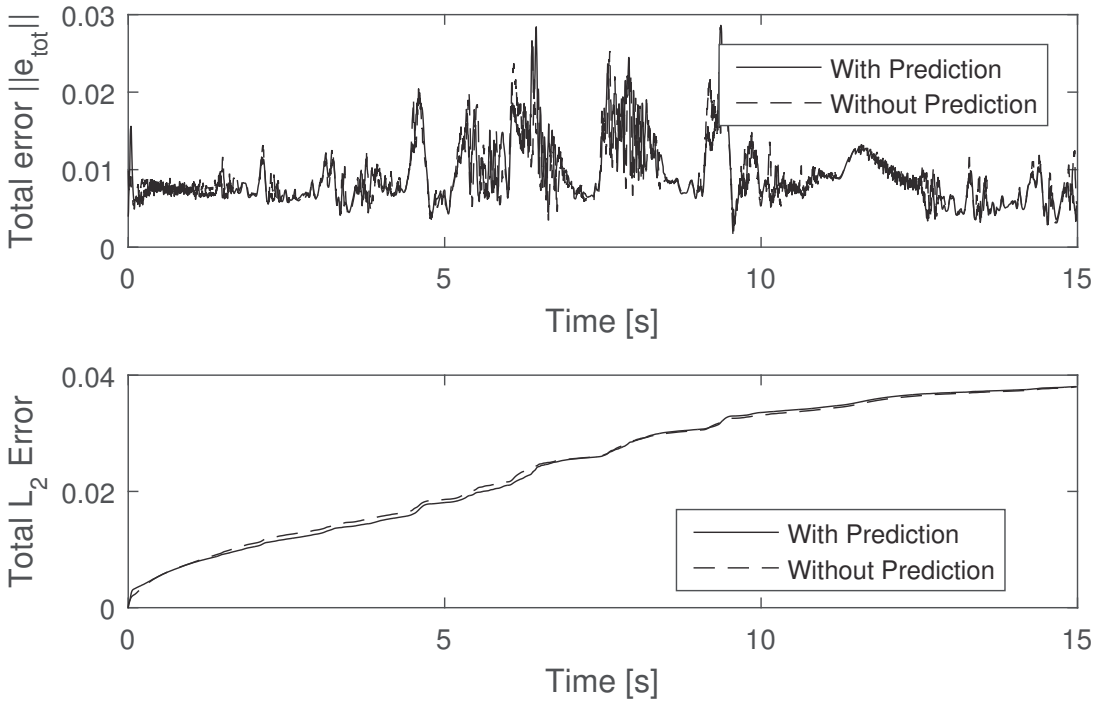


Figure 7.4: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 1).

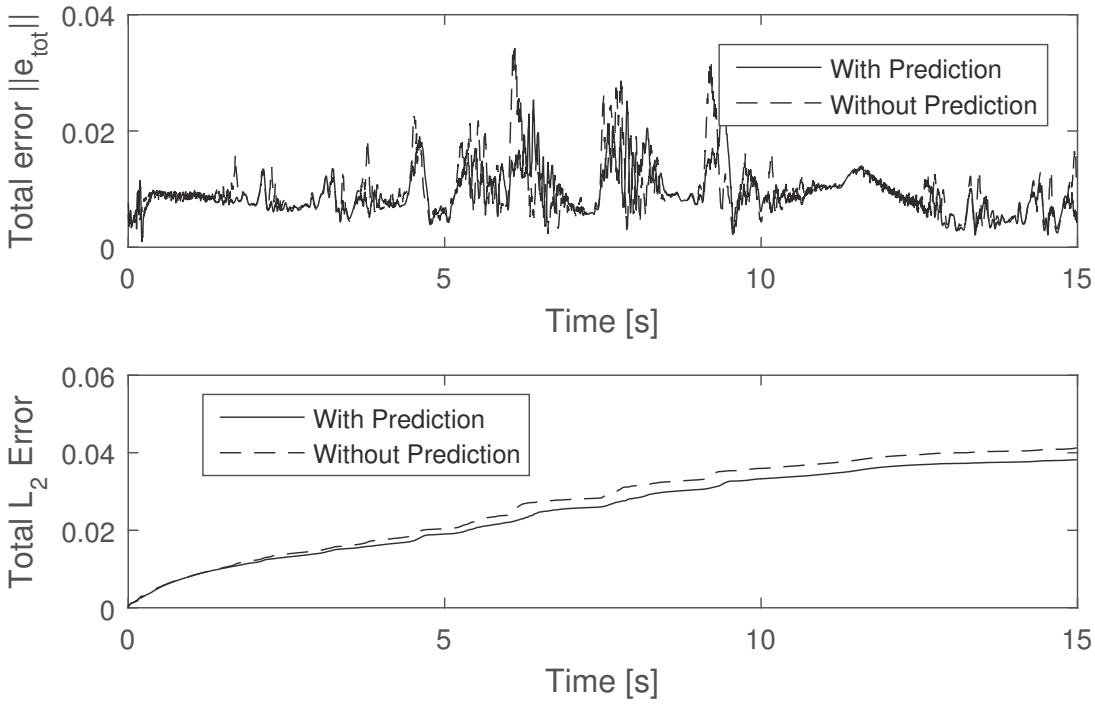


Figure 7.5: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 2).

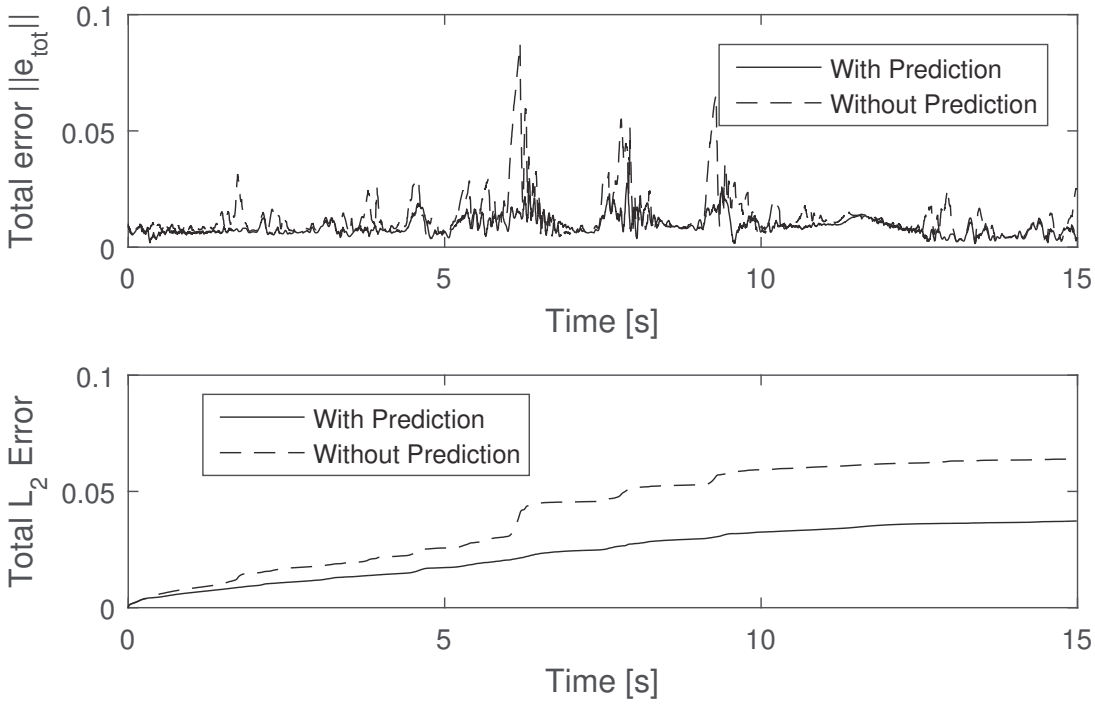


Figure 7.6: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 3).

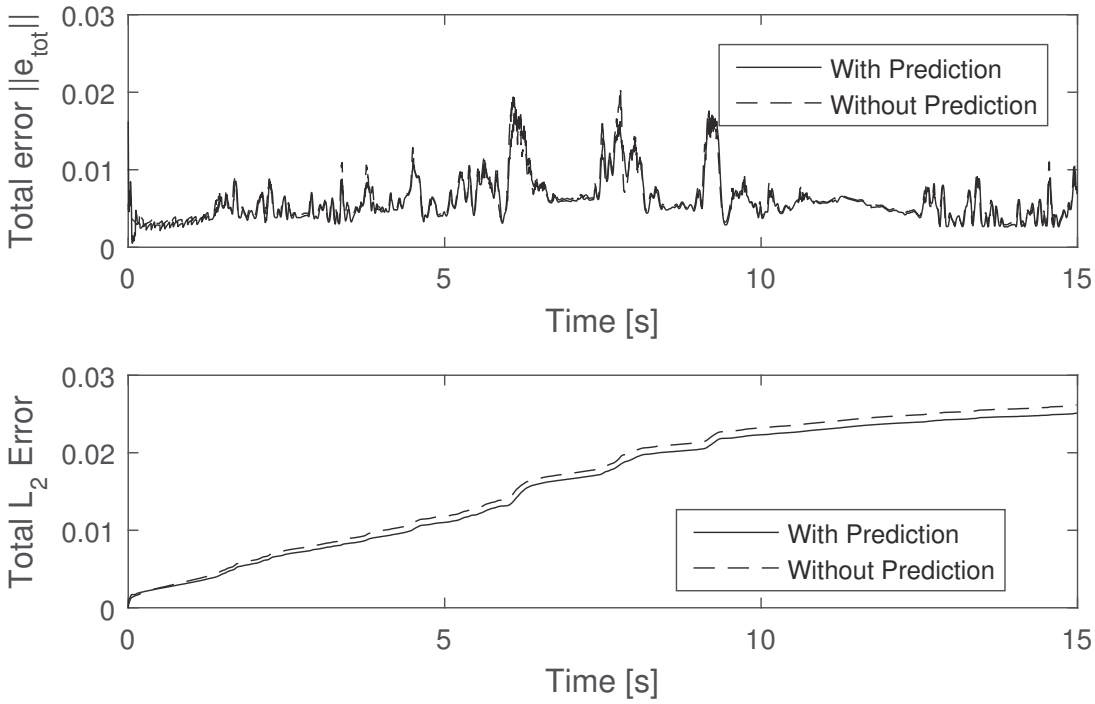


Figure 7.7: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 4).

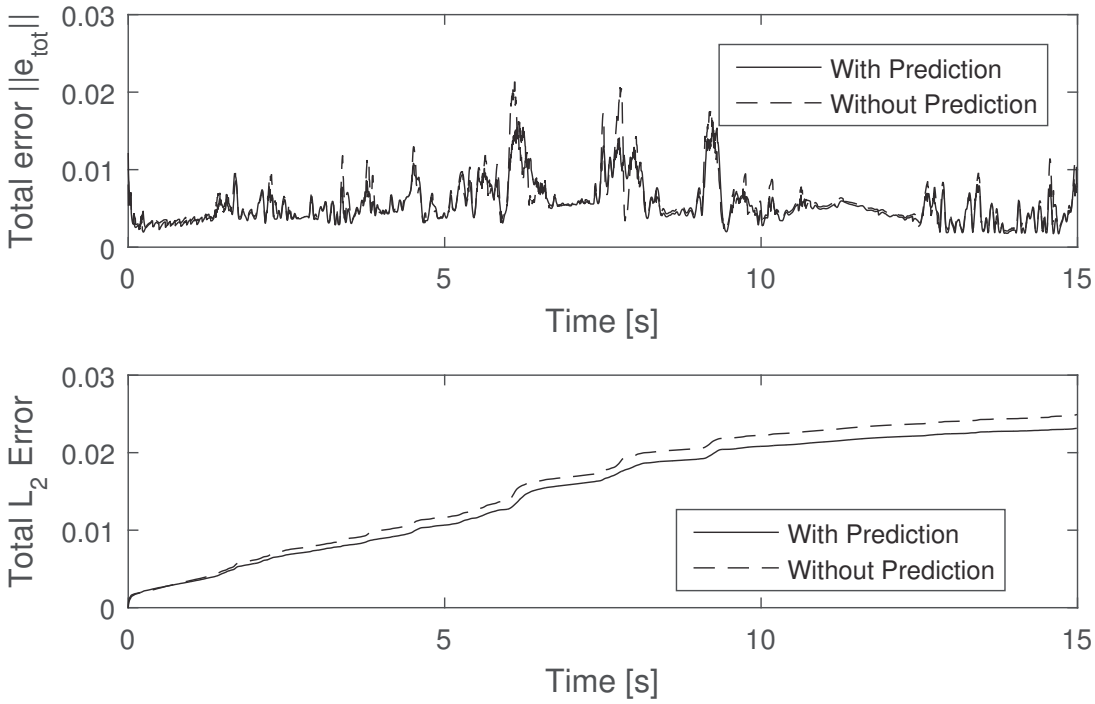


Figure 7.8: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 5).

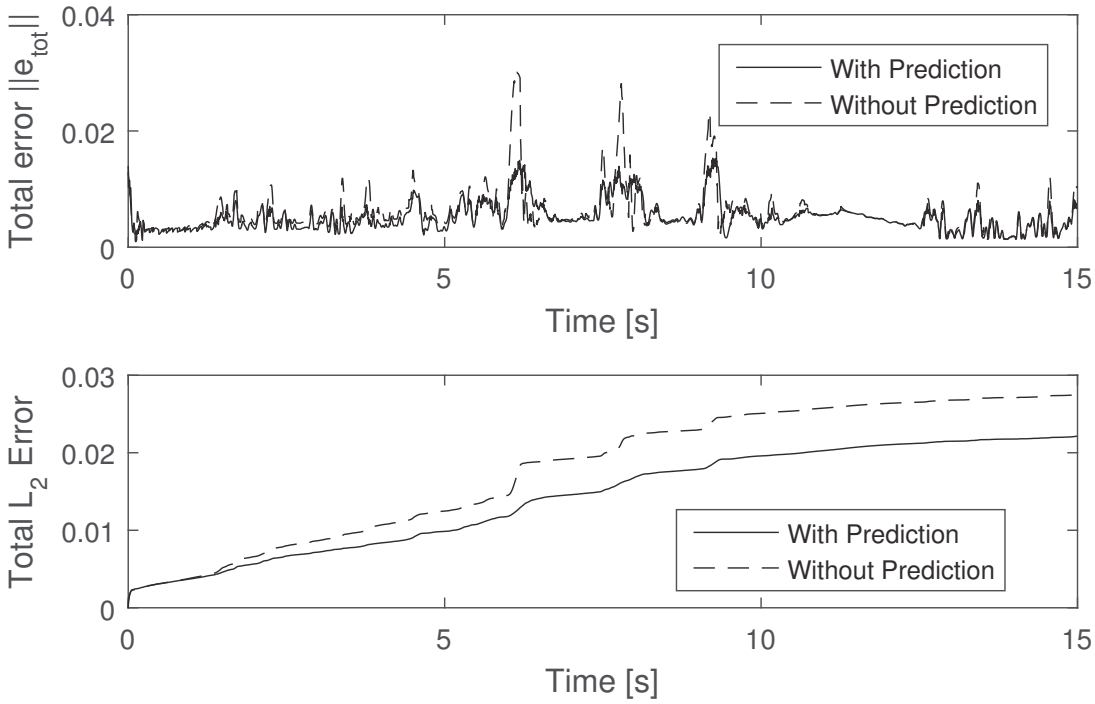


Figure 7.9: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 6).

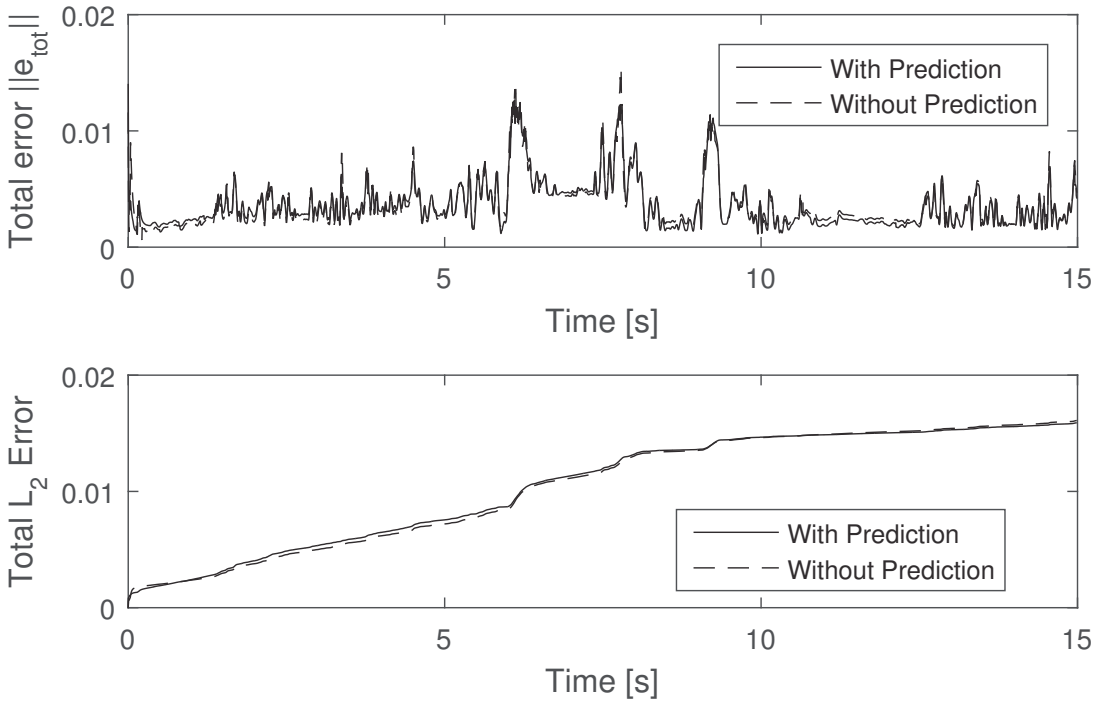


Figure 7.10: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 7).

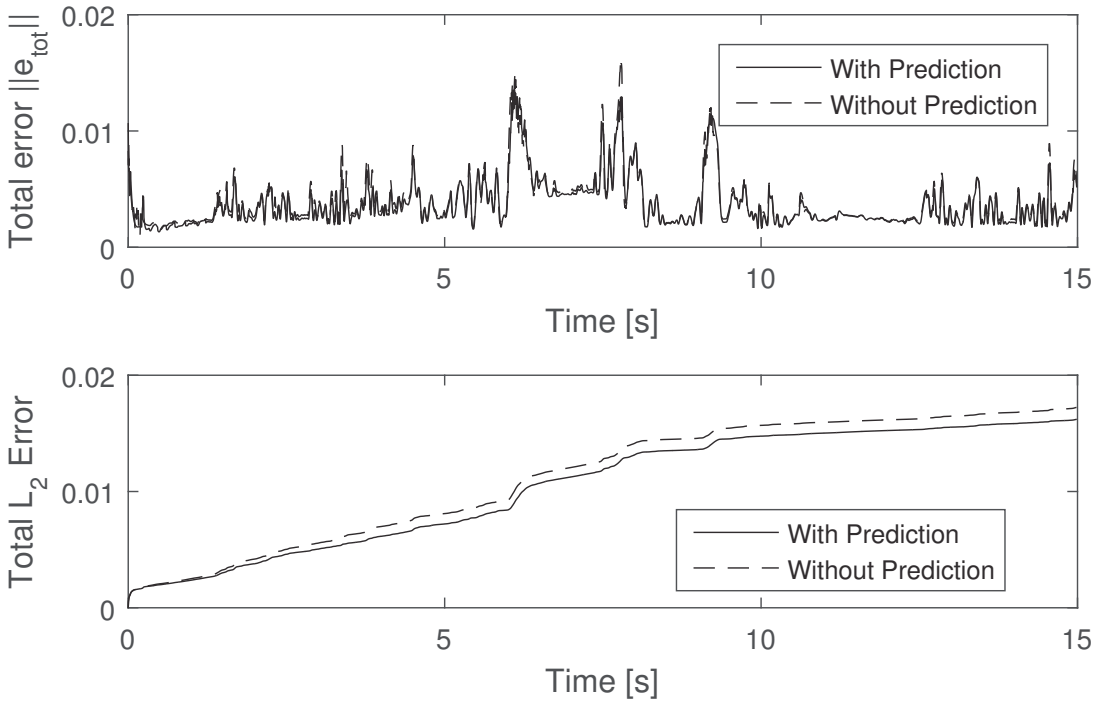


Figure 7.11: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 8).

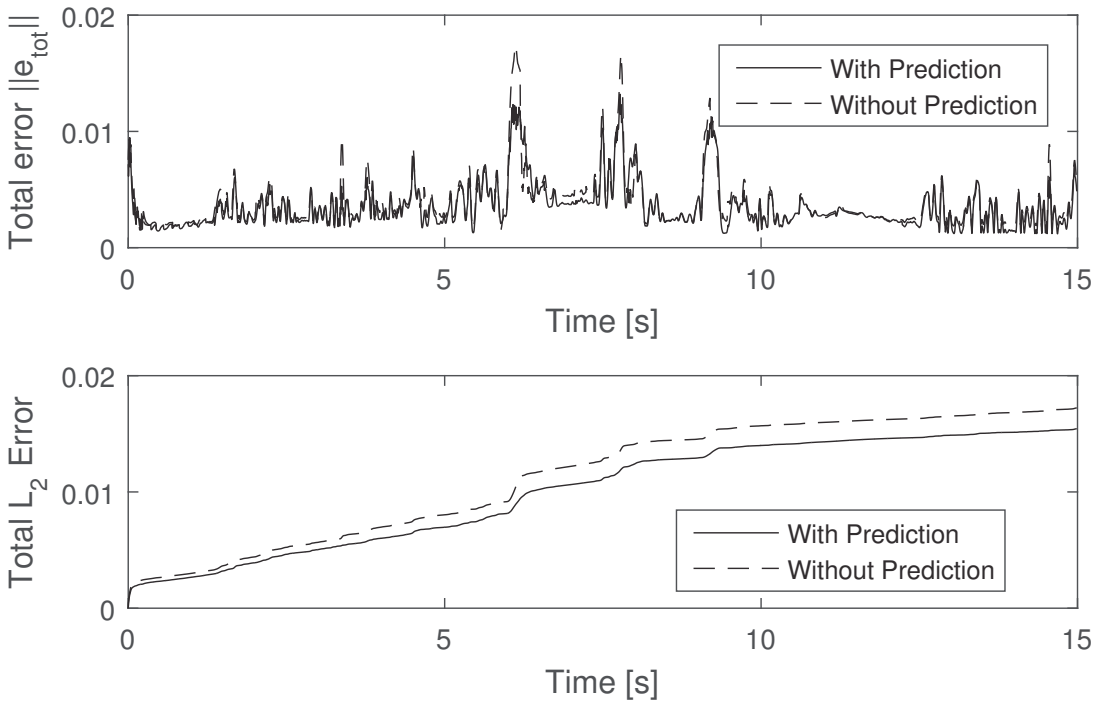


Figure 7.12: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 9).

Fig. 7.13, 7.14, and 7.15 show the guaranteed stability ranges for the predictor. The guaranteed stability range is quite small in each case, increasing in size as the sampling time increases. Although that seems counter-intuitive given that increased sampling time should decrease stability, the predictor gain design becomes more conservative as the sampling time increases, widening the range over which the LMI is feasible.

Fig. 7.16 shows an example of the parameter adaptation law output for Case 12. The stiffness changes very rapidly as $x_e \to 0$ because the derivative of the spring force with respect to x_e approaches infinity. The parameter adaptation law estimated the stiffness well when the rate of change was slow, but when the rate of change was high neither the stiffness nor the damping were accurately estimated.

Although the time-varying virtual environment parameters shown in Fig. 7.16 are outside of the guaranteed stability range of the predictor in each case, during testing each case was stable. Table 7.4 summarizes the results of the experiments. Fig. 7.17, Fig. 7.18, and Fig. 7.19 show the total error magnitude and $\mathcal{L}_2$ norm of the error for each of the experiments. At low sampling times (T_d = 0.016, 0.032 s) the predictor results in increased peaks in the total error magnitude by +62% and

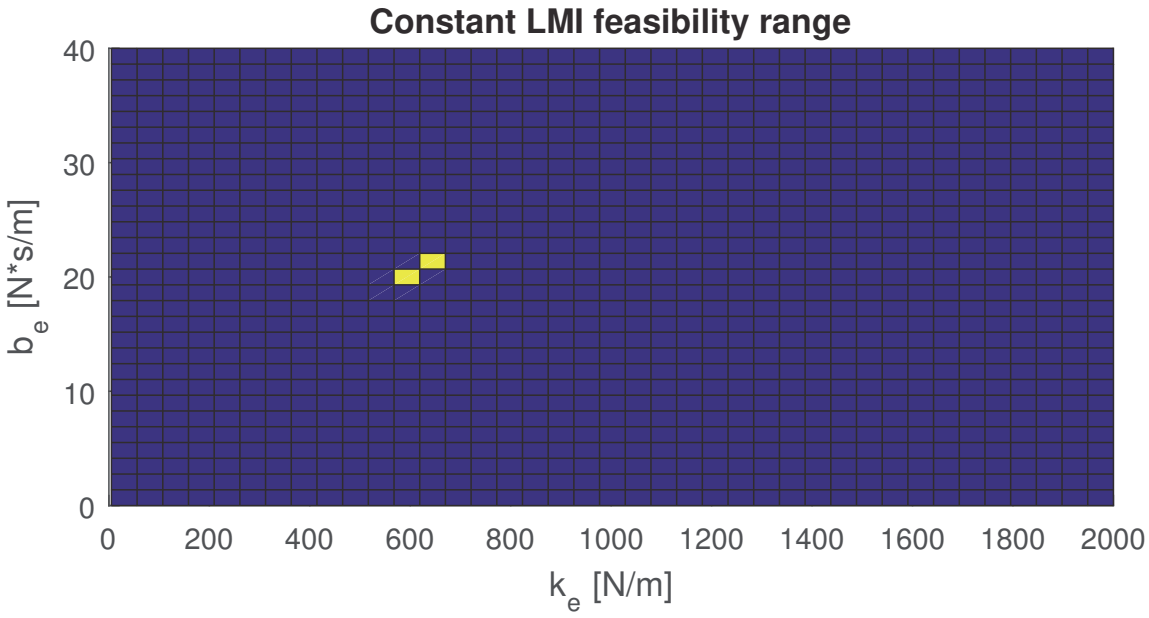


Figure 7.13: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable. (Case 10)

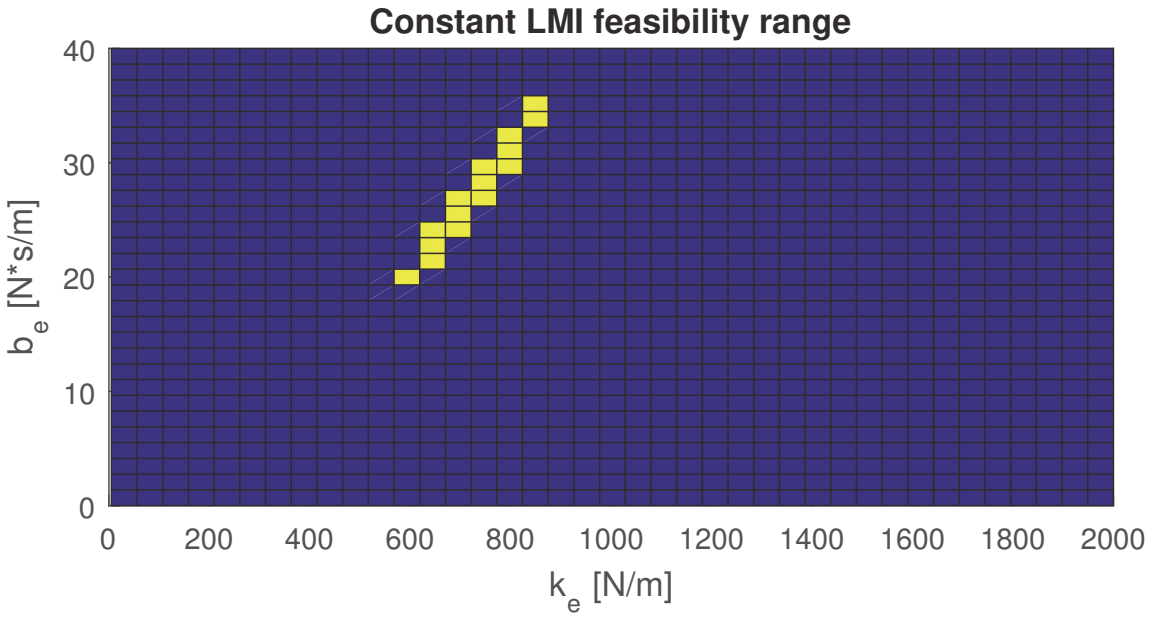


Figure 7.14: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable. (Case 11)

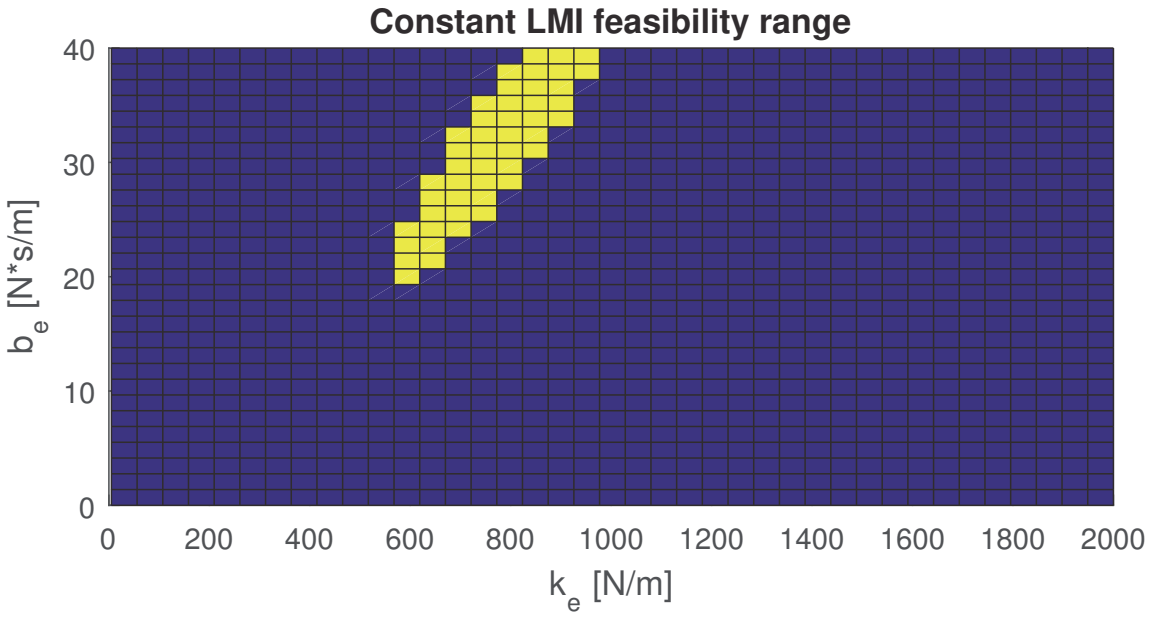


Figure 7.15: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable. (Case 12)

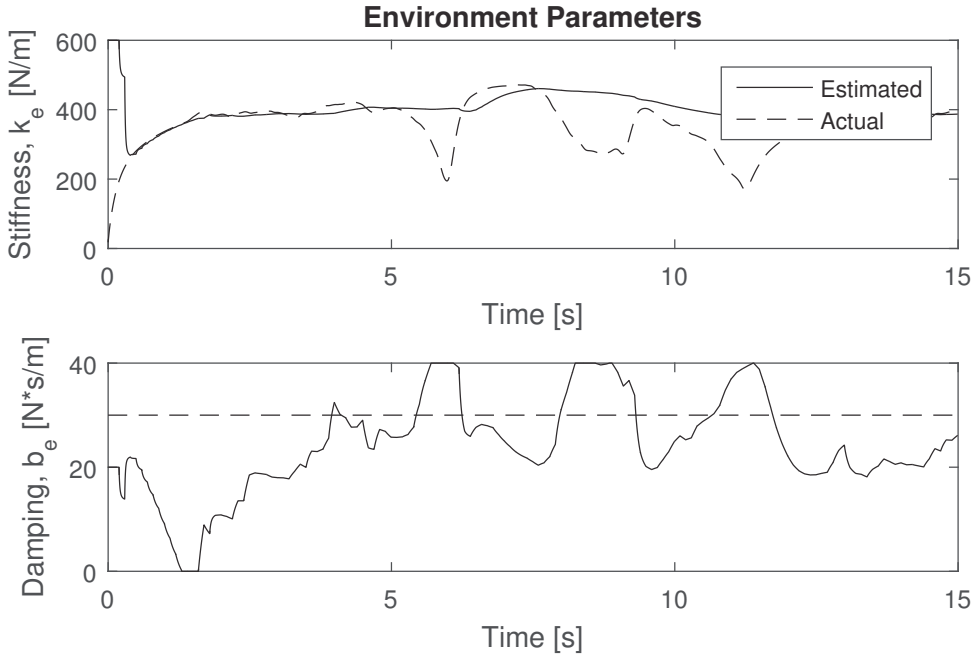


Figure 7.16: Sample output of the parameter adaptation law, showing the estimated and actual environment parameters (Case 12).

Table 7.4: Summary of the performance of the predictor for time-varying VE parameters

Case	Sampling Time (T_d, s)	Error Bound Difference	$\mathcal{L}_2$ Norm Difference
10	0.016	+0.0125 (+62.0%)	−7.7%
11	0.032	+0.0062 (+19.8%)	−9.7%
12	0.100	−0.0297 (−53.0%)	−32.2%

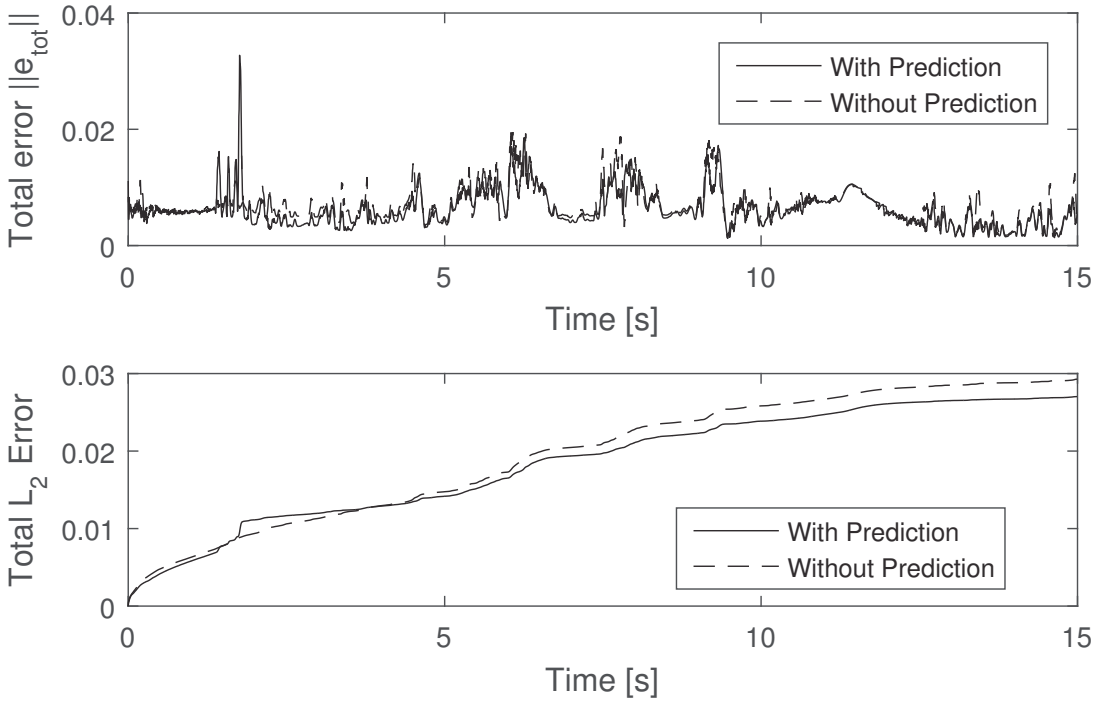


Figure 7.17: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 10).

+20%, respectively. However the chattering in the error signal decreases leading to an overall decrease in the $\mathcal{L}_2$ norm of -8% and -10%, respectively. For the high sampling time ($T_d = 0.100$ s) the predictor lead to significantly improved performance in both performance metrics. These results are similar to the simulation results for a nonlinear virtual environment in Section 6.2.2, where sampling times 0.100 s or larger were necessary for any significant performance improvement.

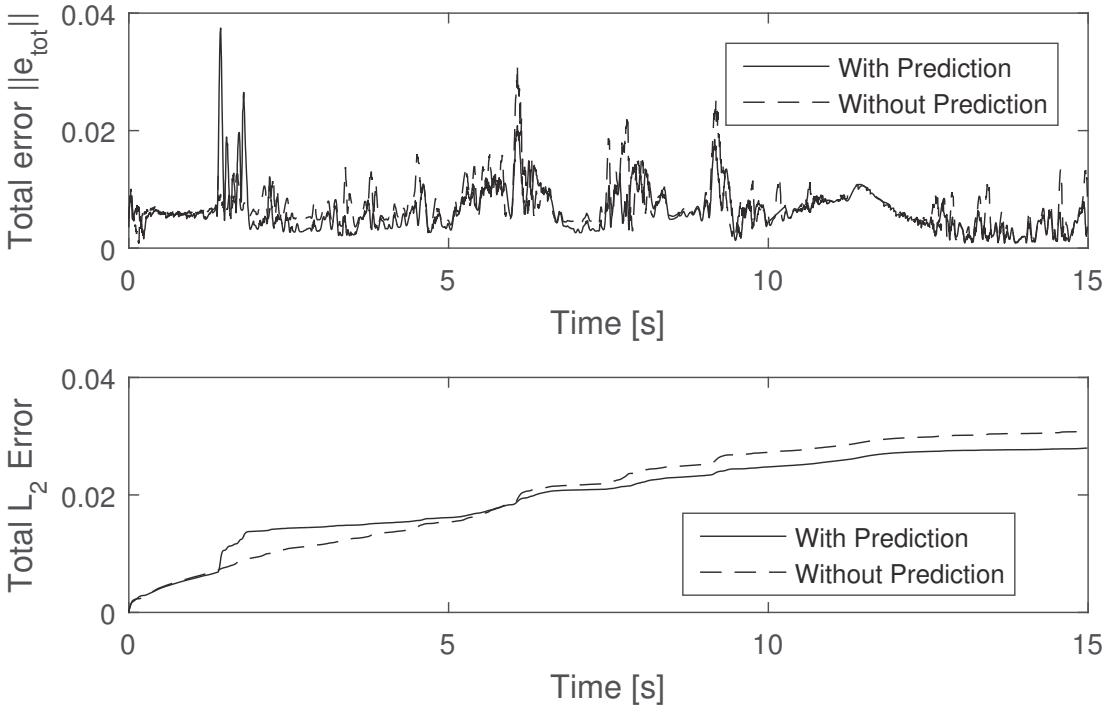


Figure 7.18: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 11).

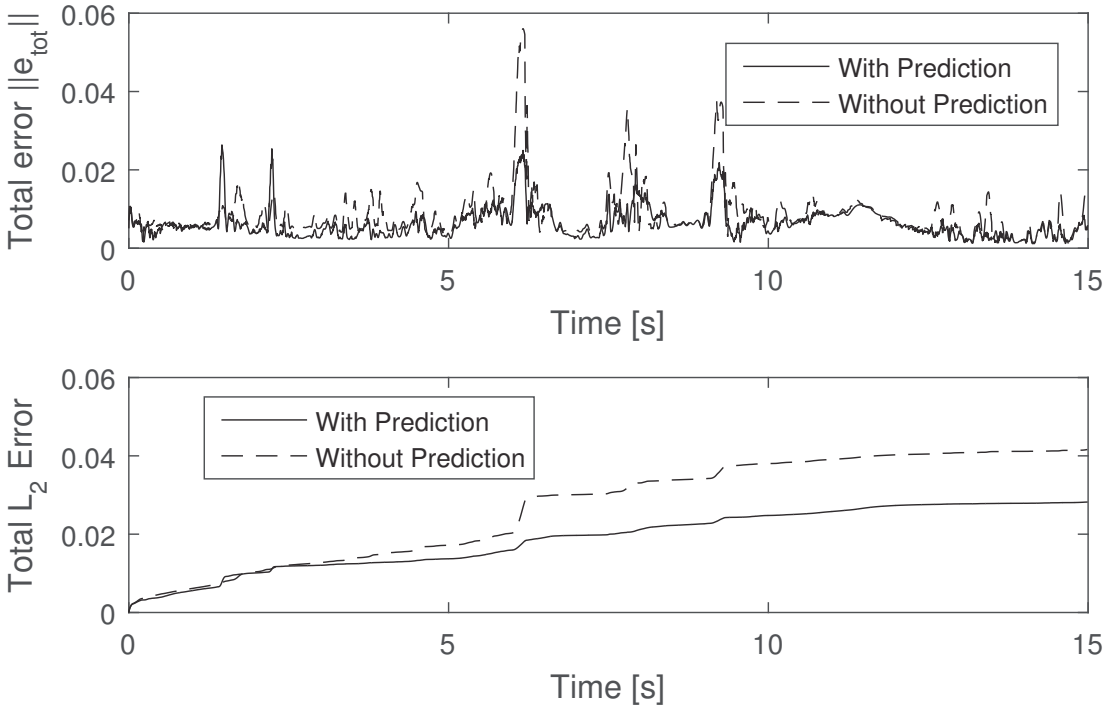


Figure 7.19: Comparison of the total tracking error magnitude and $\mathcal{L}_2$ norm with and without prediction (Case 12).

7.2 Gain-Scheduled Predictor Results

This section gives the results for the gain-scheduled predictor. Each experiment case involved three tests: one where the controller tracked the output of the gain-scheduled predictor, one where it tracked the output of the constant-gain predictor, and one where it tracked the delayed and sampled signal from the virtual environment.

7.2.1 Unknown Nonlinear Virtual Environment with Delay and Sampling

The same cases were tested experimentally as were simulated in Section 6.2.2, with an additional set of tests for $T_d = 0.200$ s. The virtual environment stiffness was nonlinear according to (3.34), and the damping was constant ($b_e = 30$ N·s/m). The bounds on the environment parameters were $\boldsymbol{\theta}_{e,min} = [200 \;\; 1 \;\; -10.26]^T$ and $\boldsymbol{\theta}_{e,max} = [2000 \;\; 40 \;\; 0]^T$. Two different sets of virtual environment parameters were tested. For cases A1 to A4, the environment parameters were $k_{e,1} = 370$ N/m, $k_{e,2} = 1980$ N/m, and $x_s = 0.0028$ m. For cases B1 to B4, the environment parameters were $k_{e,1} = 450$ N/m, $k_{e,2} = 1000$ N/m, and $x_s = 0.005$ m. In all eight cases $R = 20$.

The predictor used the linearized environment model given by (3.33). The gains for the gain-scheduled predictor were designed using Theorem 5 with $\boldsymbol{\rho}_0 = [1100 \;\; 20.5]^T$ and $\dot{\boldsymbol{\rho}}_0 = [1.8 \times 10^5 \;\; 3900]^T$. The gains designed for each case are given in 7.5. A set of predictor gains for the constant-gain predictor were also designed using Theorem 3, and are given in 7.5. The stability range for both the gain-scheduled and constant gain predictors are given in Fig. 7.20 to Fig. 7.27. The guaranteed stability range for both predictors increases as T_d increases from 0.016 s to 0.100 s, however it decreases when T_d increases from 0.100 s to 0.200 s.

A summary of the performance of the gain-scheduled predictor is given in Table 7.6. In every test case the difference in the total error $\mathcal{L}_2$ norm is negligible. The total error bound decreases by a significant amount for most of the tests, except for Case A and Case B when $T_d = 0.200$ s, in which case there are very large peaks in the total error for the system with the predictor which are not present in the system without the predictor. Case B4 in particular results in a very large error peak approximately 300% larger than for the system without prediction.

Table 7.5: Designed predictor gains for constant VE parameters

T_d (s)	0.016	0.032	0.100
L_0	[-3.3 × 10^{-5} -2.0 × 10^{-6}]	[8.6 × 10^{-6} 4.7 × 10^{-7}]	[1.4 × 10^{-6} 2.8 × 10^{-6}]
L_1	[-2.6 × 10^{-2} -1.6 × 10^{-3}]	[1.1 × 10^{-2} 1.7 × 10^{-3}]	[-2.5 × 10^{-3} 4.7 × 10^{-3}]
L_2	[-1.1 × 10^{-3} -4.1 × 10^{-5}]	[-1.2 × 10^{-4} 3.3 × 10^{-5}]	[1.1 × 10^{-4} 4.2 × 10^{-5}]
L_{const}	[-466.4 -9.13]	[-344.5 -3.54]	[-236.6 -2.37]

T_d (s)	0.200
L_0	[2.6 × 10^{-5} 6.8 × 10^{-6}]
L_1	[1.7 × 10^{-3} 5.6 × 10^{-3}]
L_2	[-1.2 × 10^{-4} 1.6 × 10^{-4}]
L_{const}	[-265.7 -2.54]

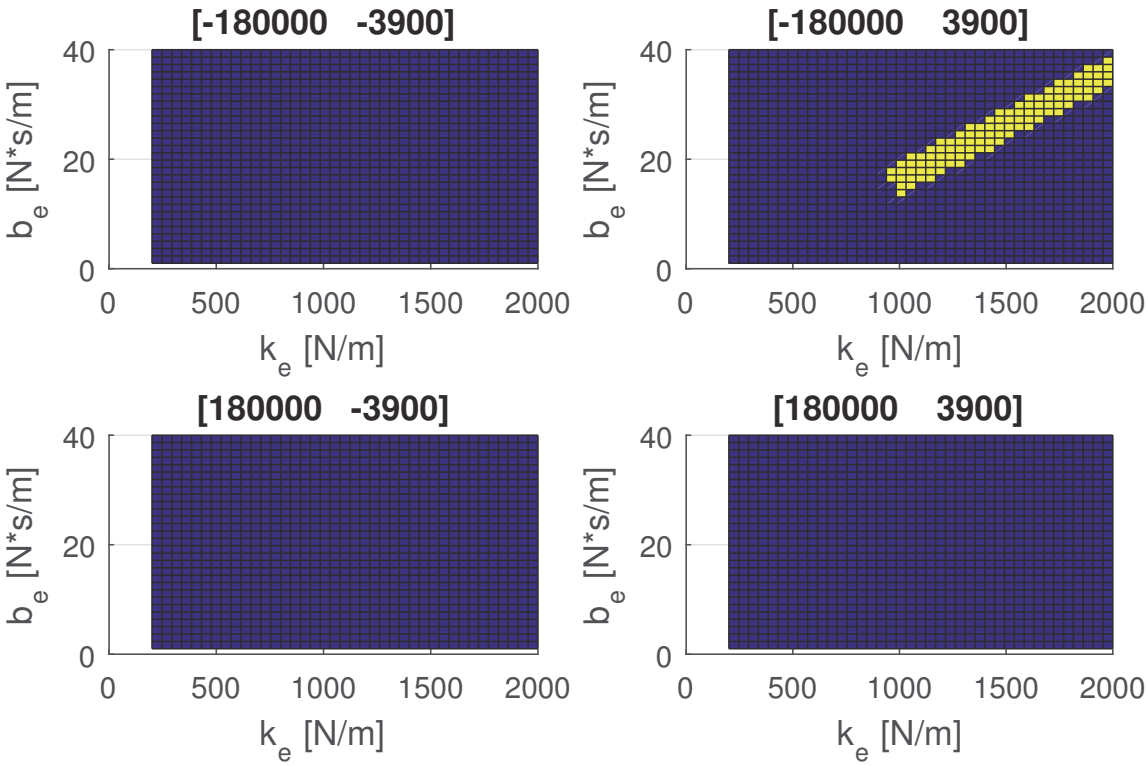


Figure 7.20: Stability region for the gain-scheduled predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.016$ s.

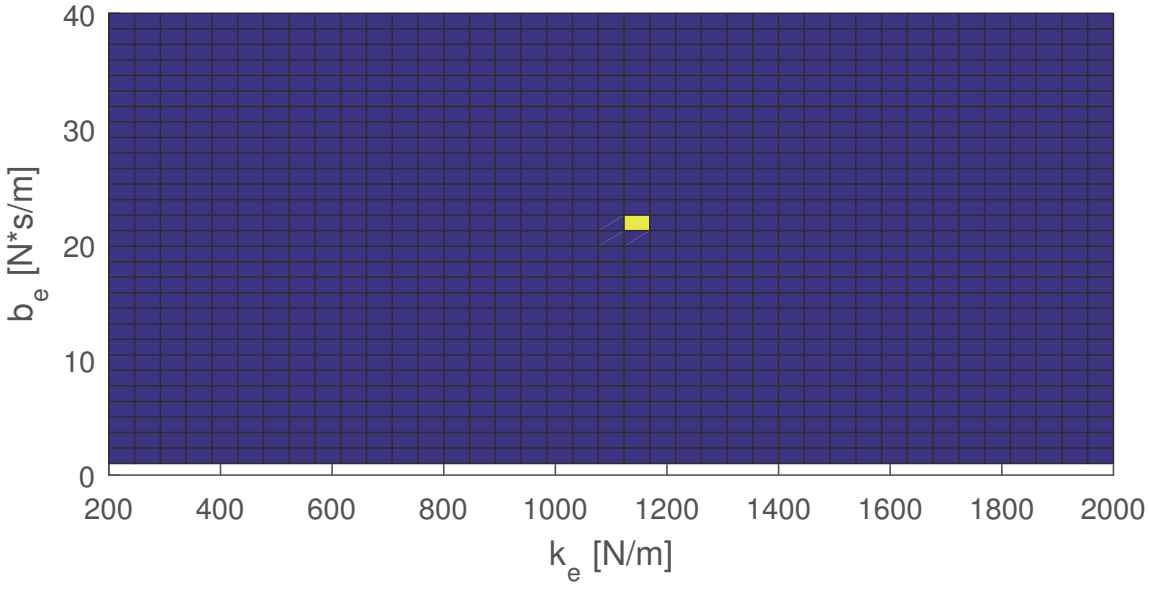


Figure 7.21: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.016$ s.

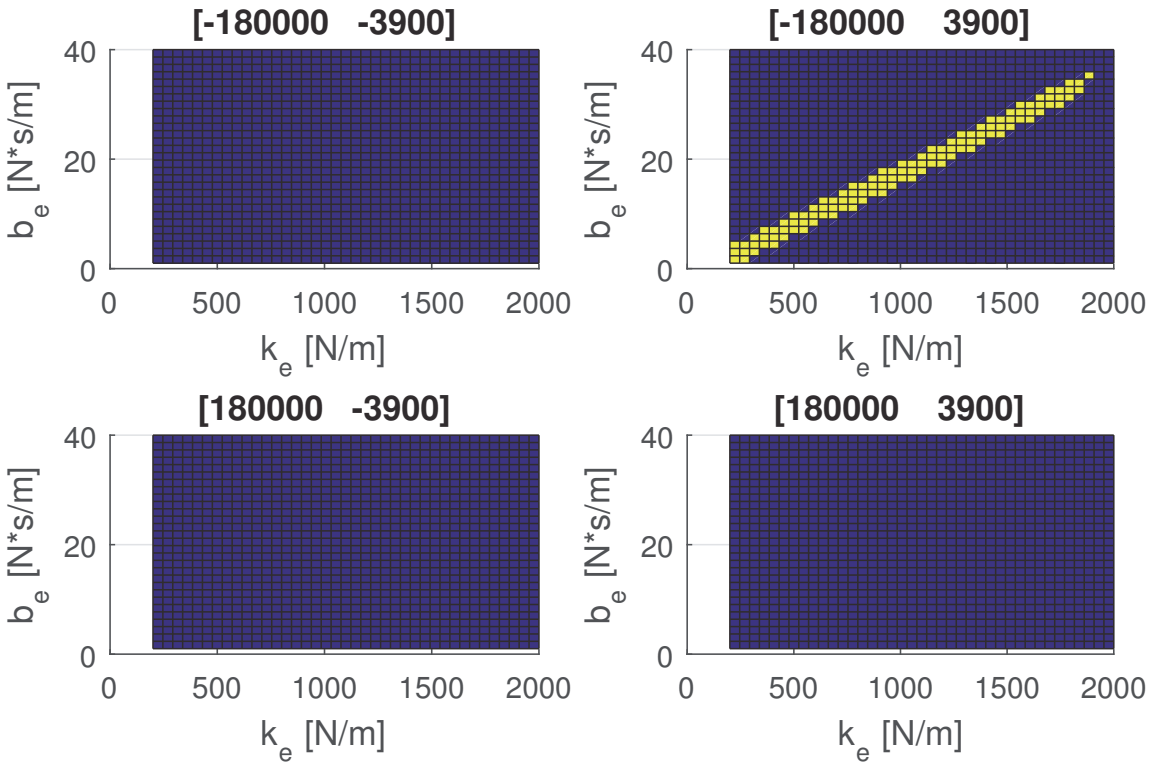


Figure 7.22: Stability region for the gain-scheduled predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.032$ s.

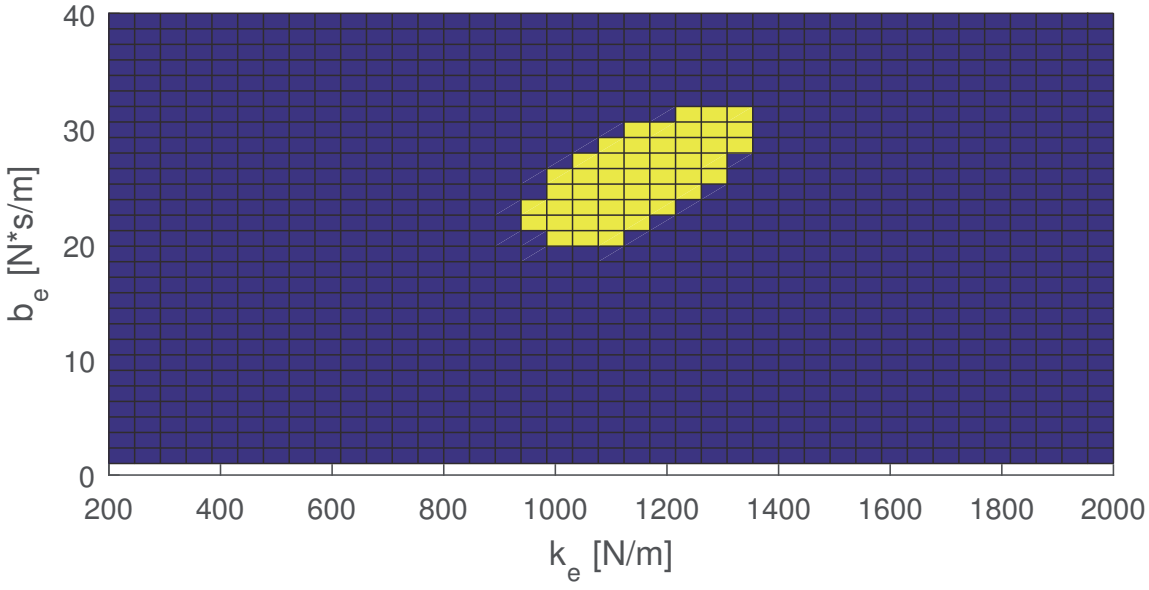


Figure 7.23: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.032$ s.

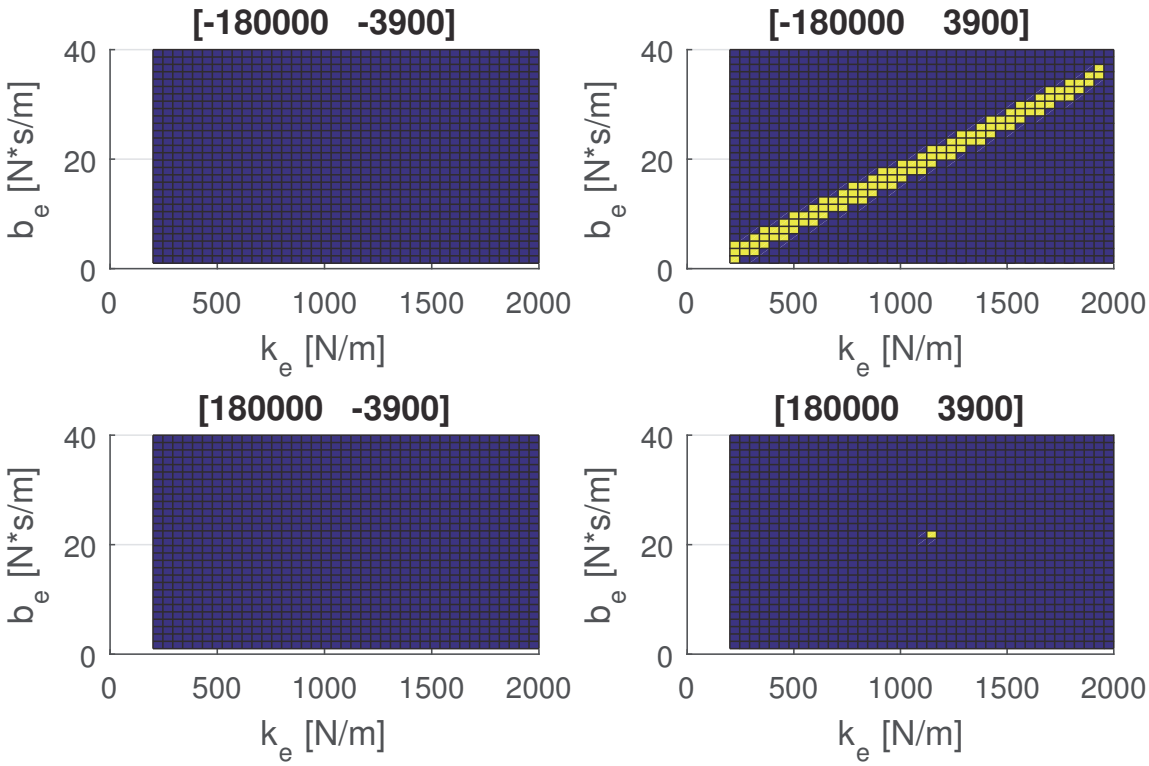


Figure 7.24: Stability region for the gain-scheduled predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.100$ s.

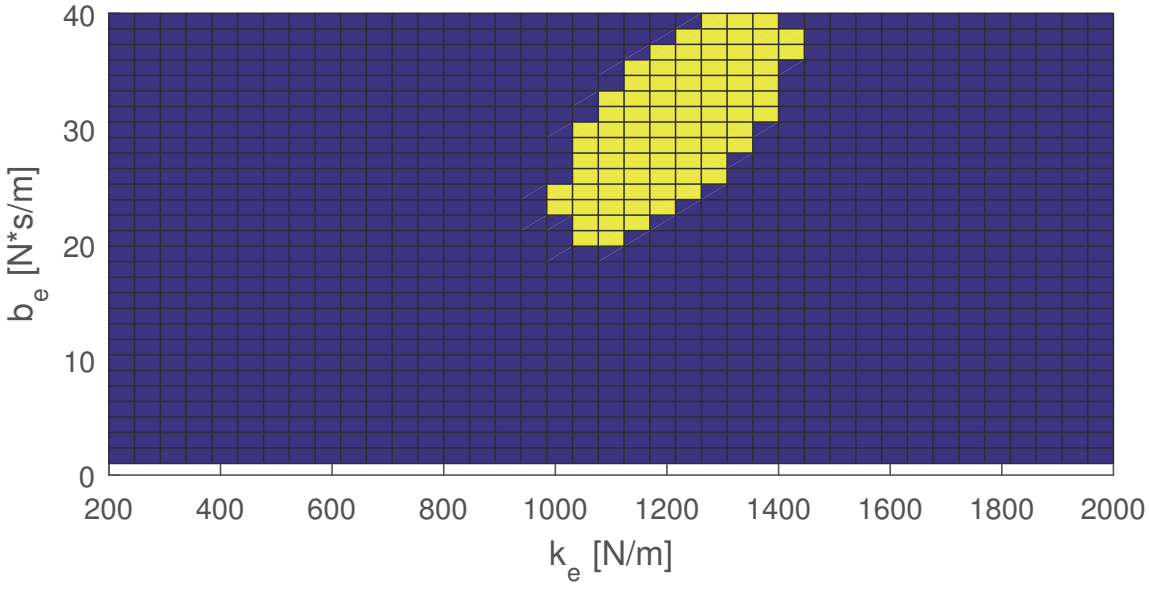


Figure 7.25: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.100$ s.

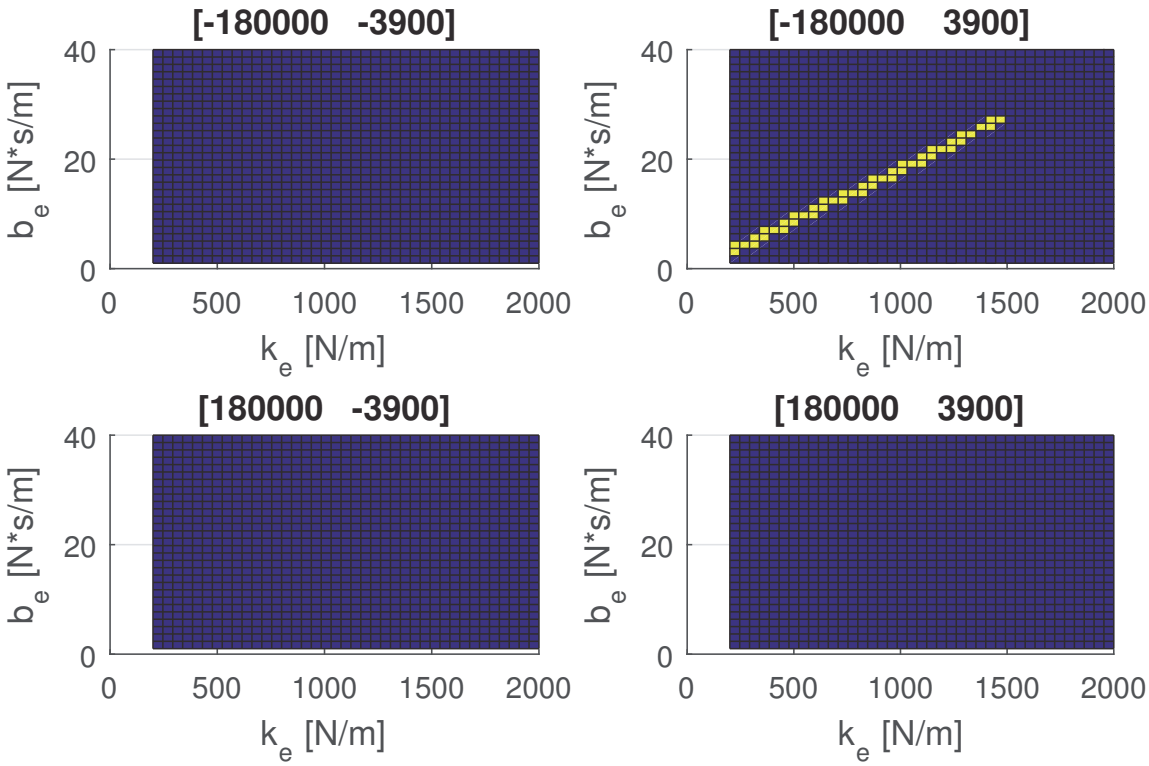


Figure 7.26: Stability region for the gain-scheduled predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.200$ s.

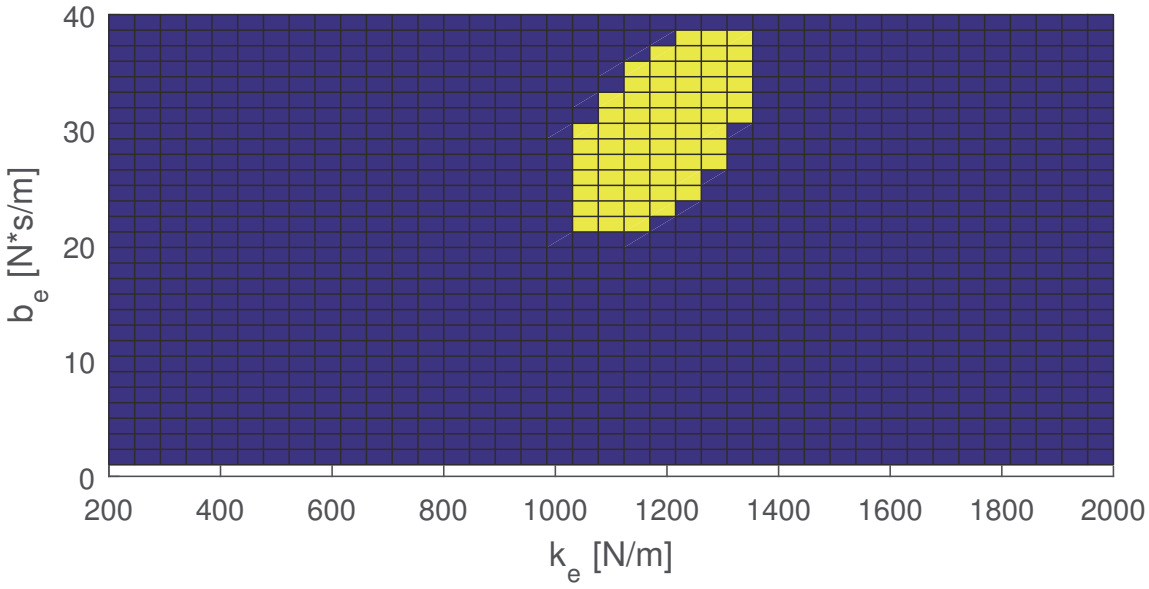


Figure 7.27: Stability region for the constant-gain predictor, where the yellow regions are guaranteed to be stable, for $T_d = 0.200$ s.

The gain-scheduled predictor's performance is not significantly different from the constant-gain predictor, similar to the simulation results in Chapter 6. In case A2 the total error $\mathcal{L}_2$ norm of the gain-scheduled predictor system was 22% higher than for the constant-gain predictor. In case A4 the total error bound for the gain-scheduled predictor system was 12% lower than the constant-gain system.

Fig. 7.28 to Fig. 7.43 show the predictor error and the total error for each case. Comparing the predictor error to the sampling and delay error, the gain-scheduled

Table 7.6: Summary of the performance of the gain-scheduled predictor for a nonlinear virtual environment

Case	Sampling Time (T_d, s)	Error Bound Difference	$\mathcal{L}_2$ Norm Difference	Improvement over CG Predictor?
A1	0.016	−0.00258 (−27.1%)	−1.62%	Negligible
A2	0.032	−0.00308 (−30.5%)	−3.48%	Worse
A3	0.100	−0.00224 (−24.9%)	−4.67%	Negligible
A4	0.200	+0.00213 (+21.0%)	−2.06%	Better
B1	0.016	−0.00062 (−5.32%)	−4.96%	Negligible
B2	0.032	−0.00231 (−15.6%)	−9.04%	Negligible
B3	0.100	−0.00228 (−14.1%)	−5.56%	Negligible
B4	0.200	+0.05135 (+318.4%)	+1.81%	Negligible

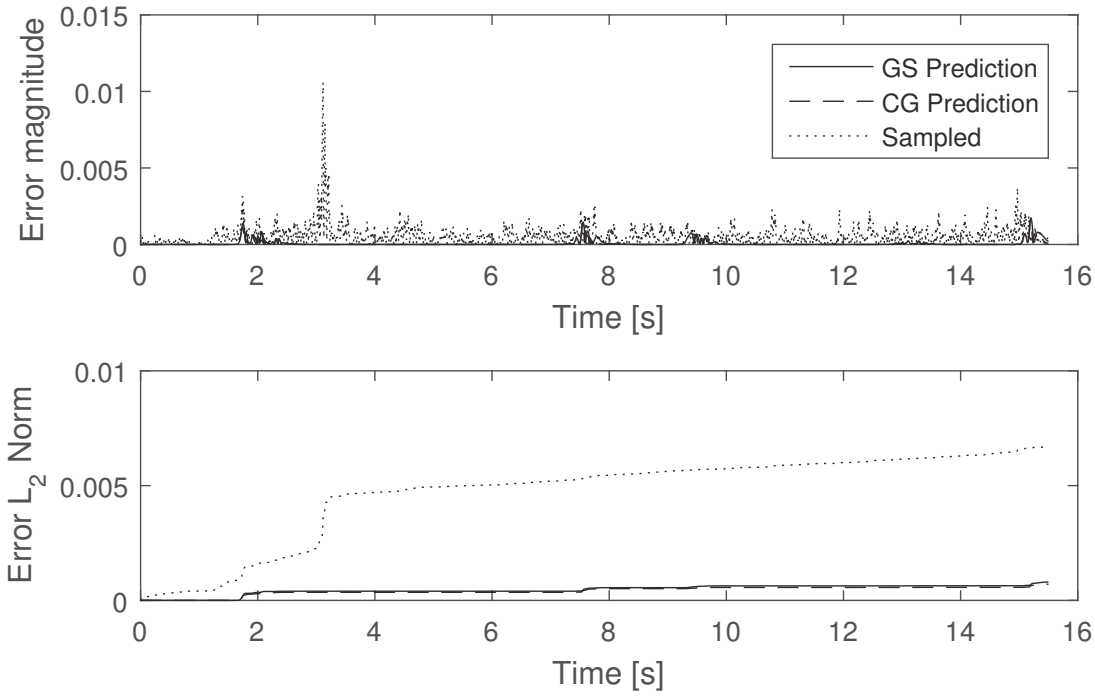


Figure 7.28: Predictor error magnitude and $\mathcal{L}_2$ norm for Case A1

predictor results in smaller peaks and less chattering in most of the cases. The gain scheduled predictor also leads to smaller peaks in the total error, but not as much of a decrease in the chattering. For example, Fig. 7.28 shows that the error in the predicted signal is greatly reduced, leading to an $\mathcal{L}_2$ norm that is approximately 90% of the sampled signal's error. However, the total error when tracking that predicted signal, shown in Fig. 7.29, is almost identical to the total error when tracking the sampled signal.

7.3 Summary of Results

The haptic system with the predictor performed extremely well for the unknown linear virtual environment. The unknown linear virtual environment is the best-case scenario for the predictor, where the environment parameters are constant or could be considered to be changing extremely slowly. Eight out of the nine cases show significant improvements in the total error bound, and five out of nine show a significant improvement in the total error $\mathcal{L}_2$ norm. Only one case shows a decrease in performance, and that is for the smallest sampling time. The relative performance of the

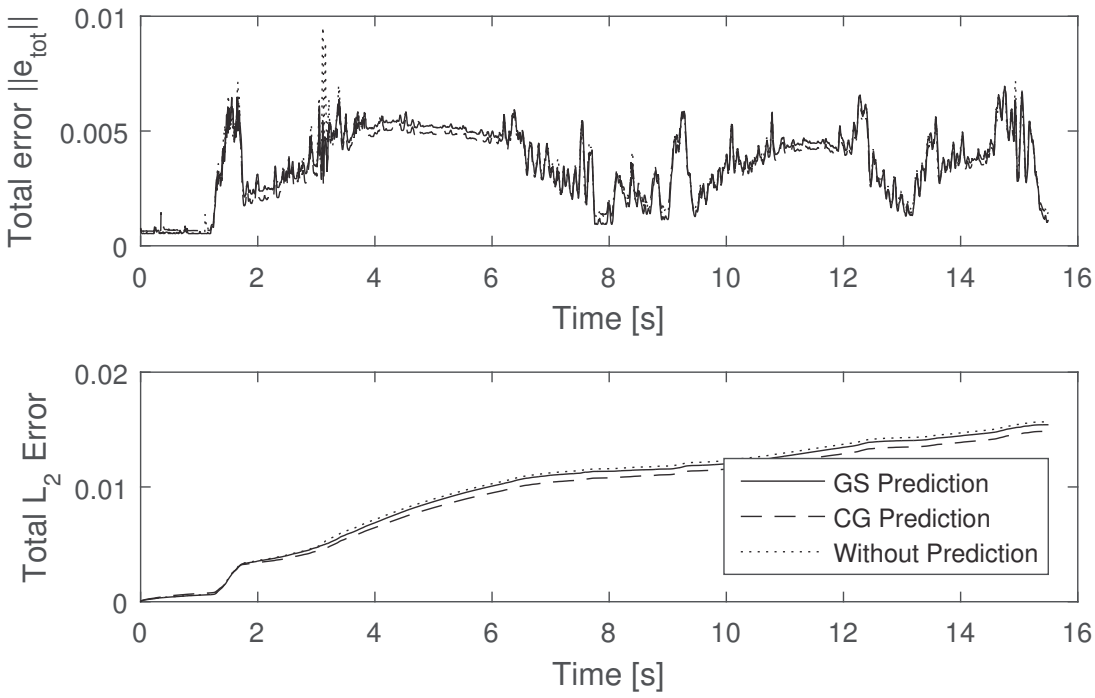


Figure 7.29: Total error magnitude and $\mathcal{L}_2$ norm for Case A1

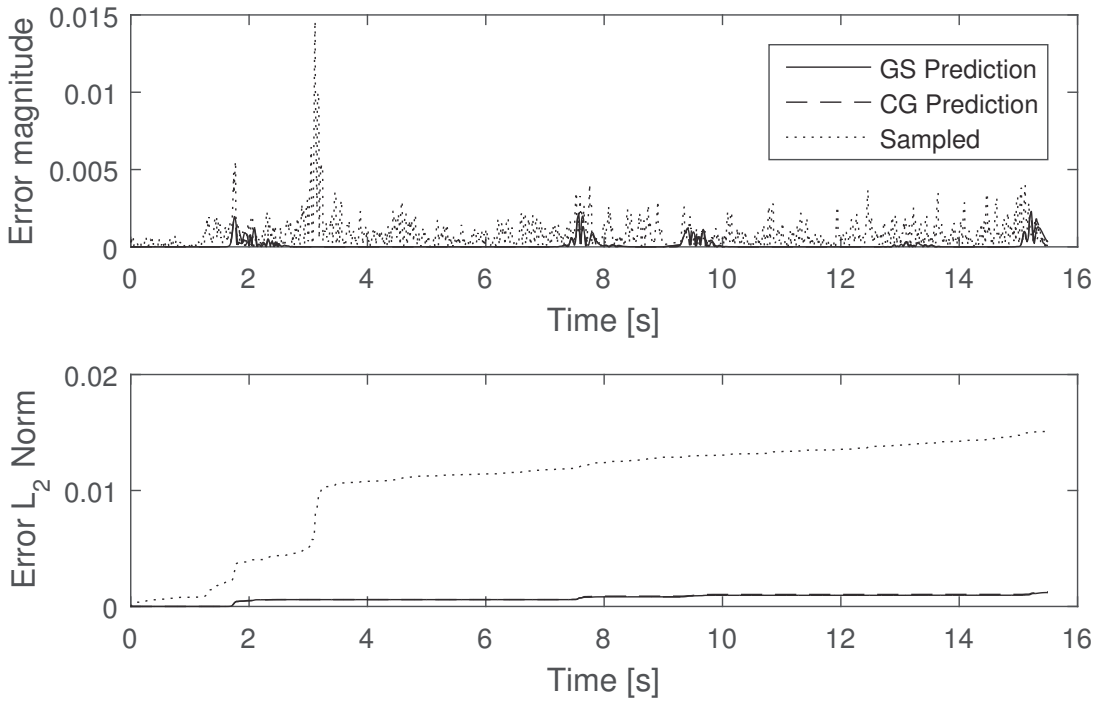


Figure 7.30: Predictor error magnitude and $\mathcal{L}_2$ norm for Case A2

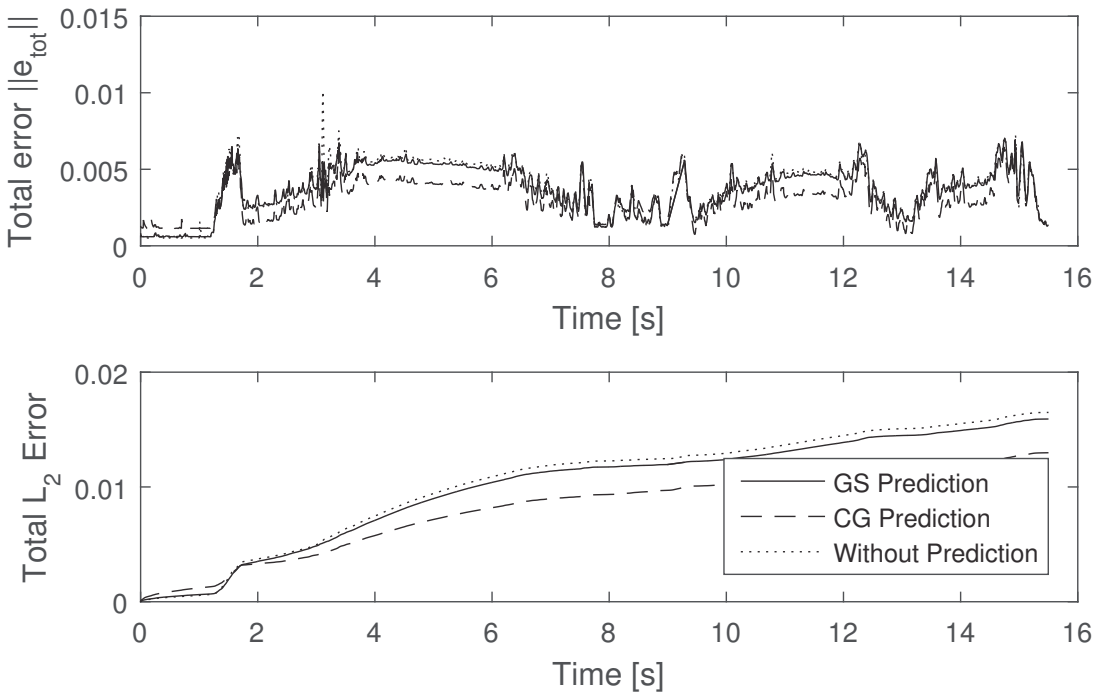


Figure 7.31: Total error magnitude and $\mathcal{L}_2$ norm for Case A2

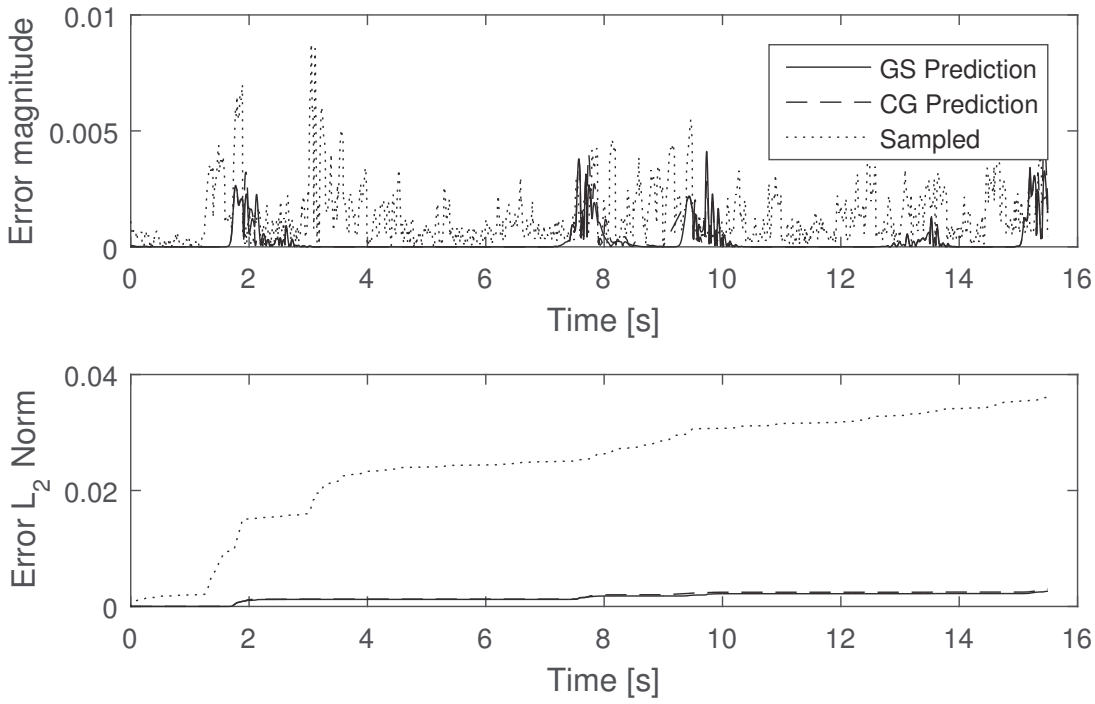


Figure 7.32: Predictor error magnitude and $\mathcal{L}_2$ norm for Case A3

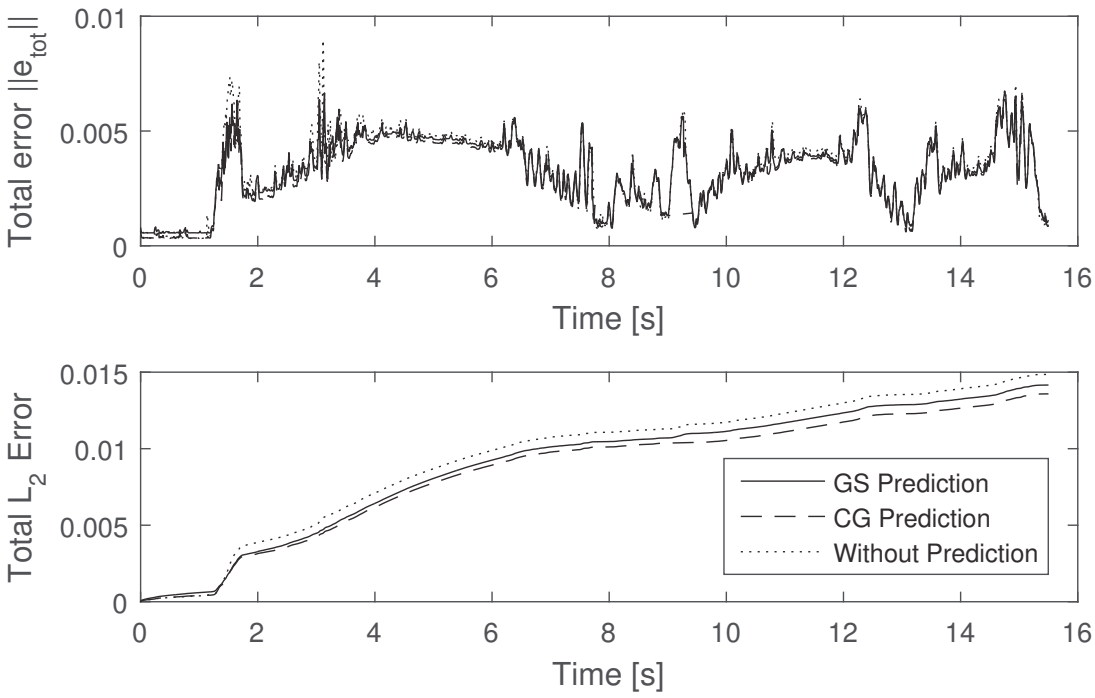


Figure 7.33: Total error magnitude and $\mathcal{L}_2$ norm for Case A3

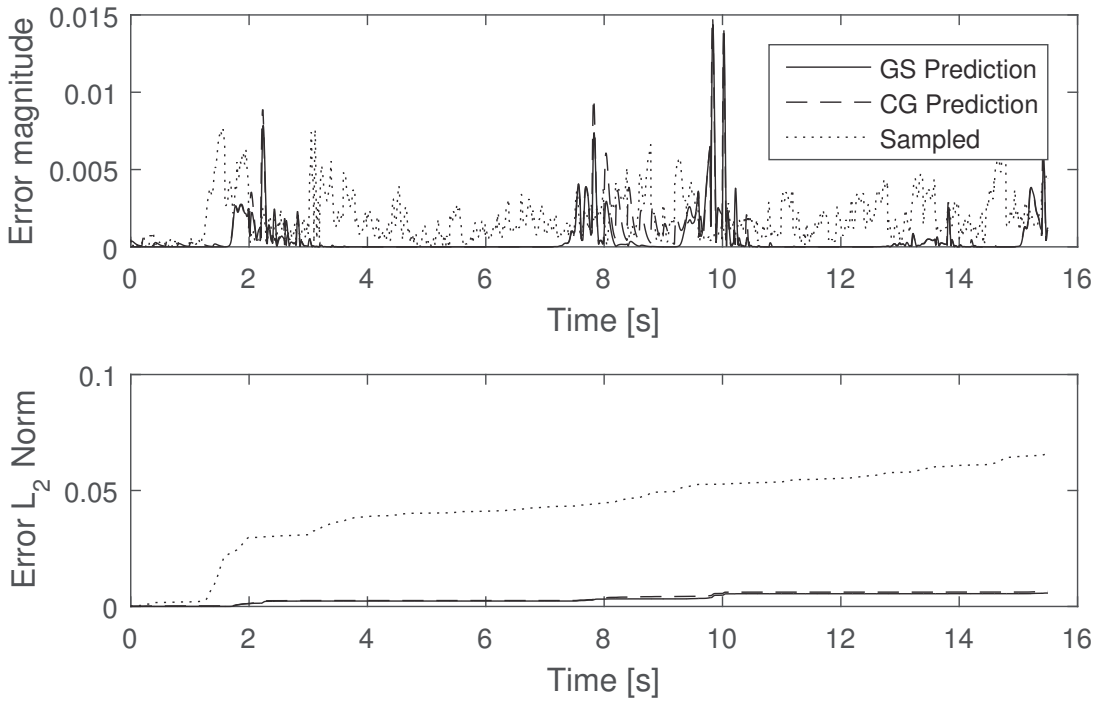


Figure 7.34: Predictor error magnitude and $\mathcal{L}_2$ norm for Case A4

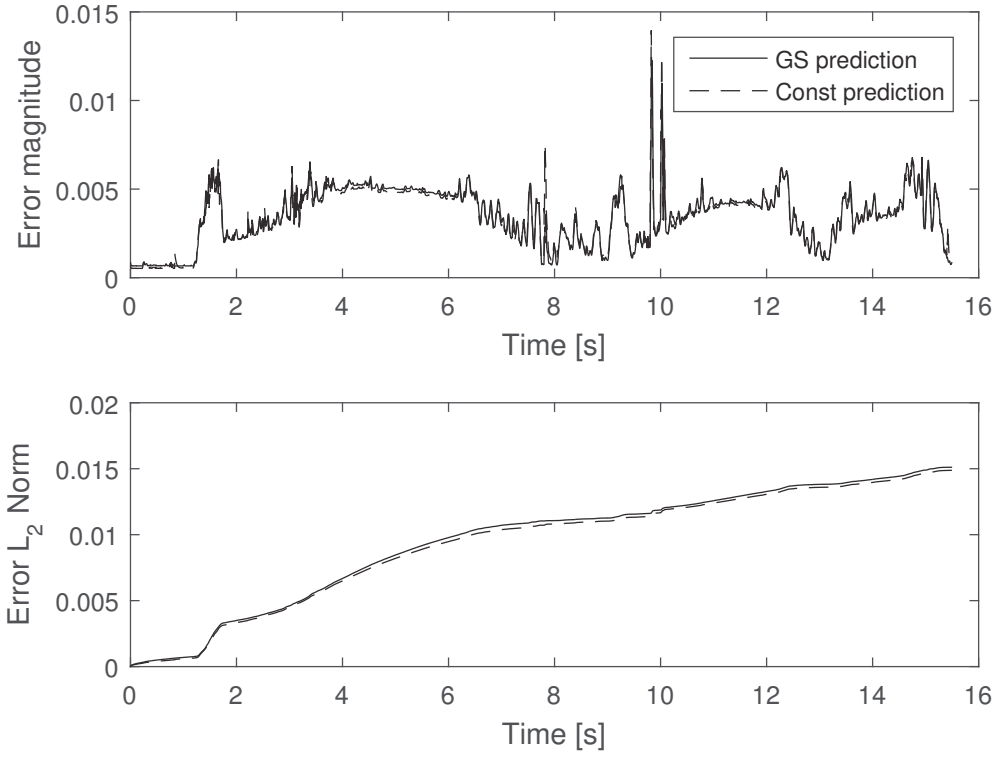


Figure 7.35: Total error magnitude and $\mathcal{L}_2$ norm for Case A4

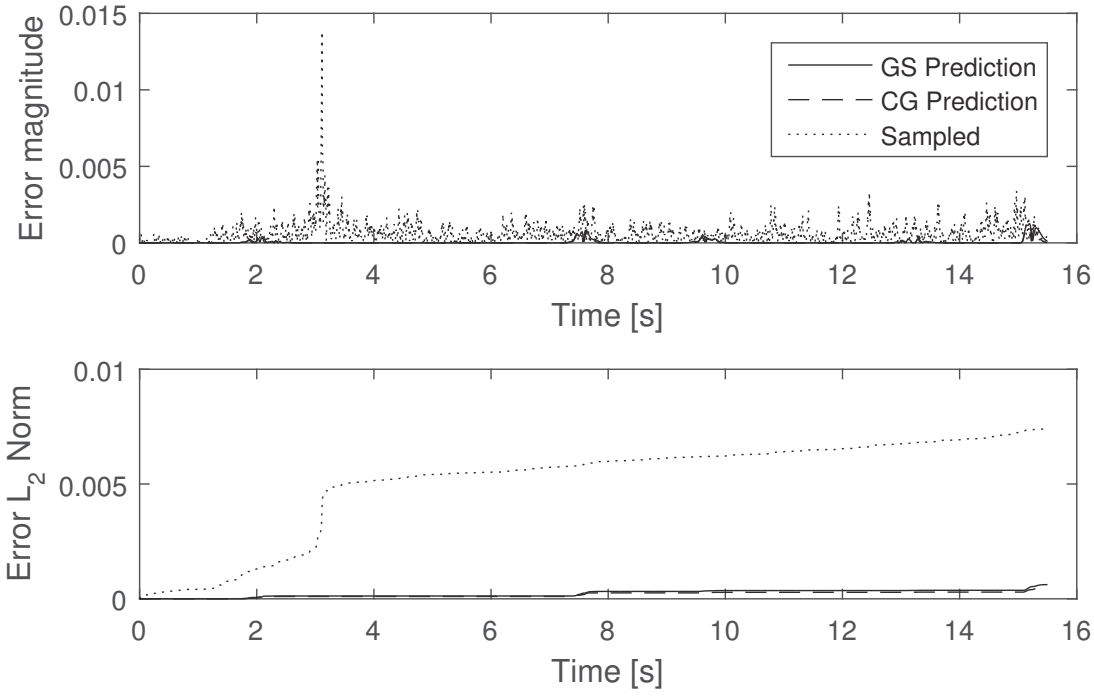


Figure 7.36: Predictor error magnitude and $\mathcal{L}_2$ norm for Case B1

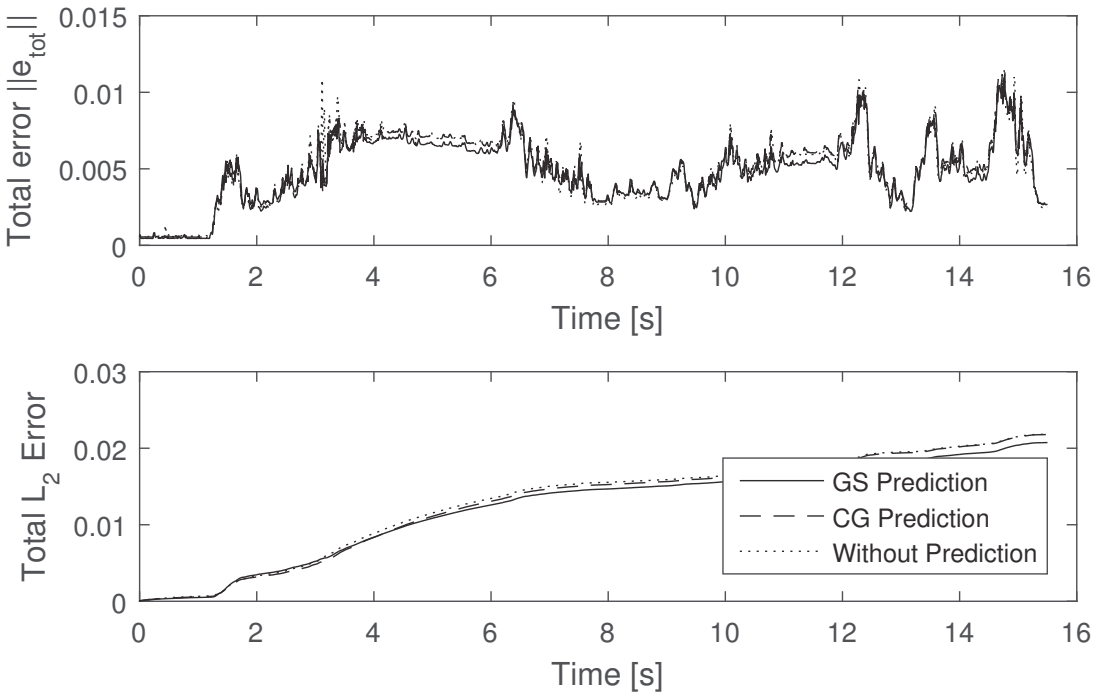


Figure 7.37: Total error magnitude and $\mathcal{L}_2$ norm for Case B1

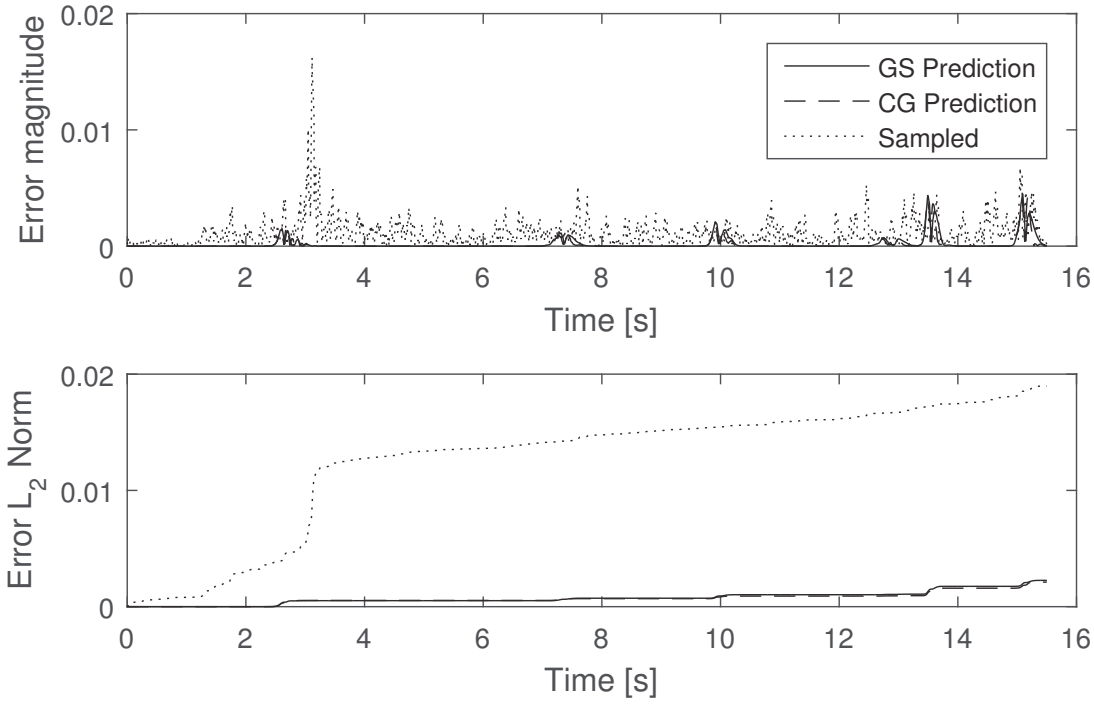


Figure 7.38: Predictor error magnitude and $\mathcal{L}_2$ norm for Case B2

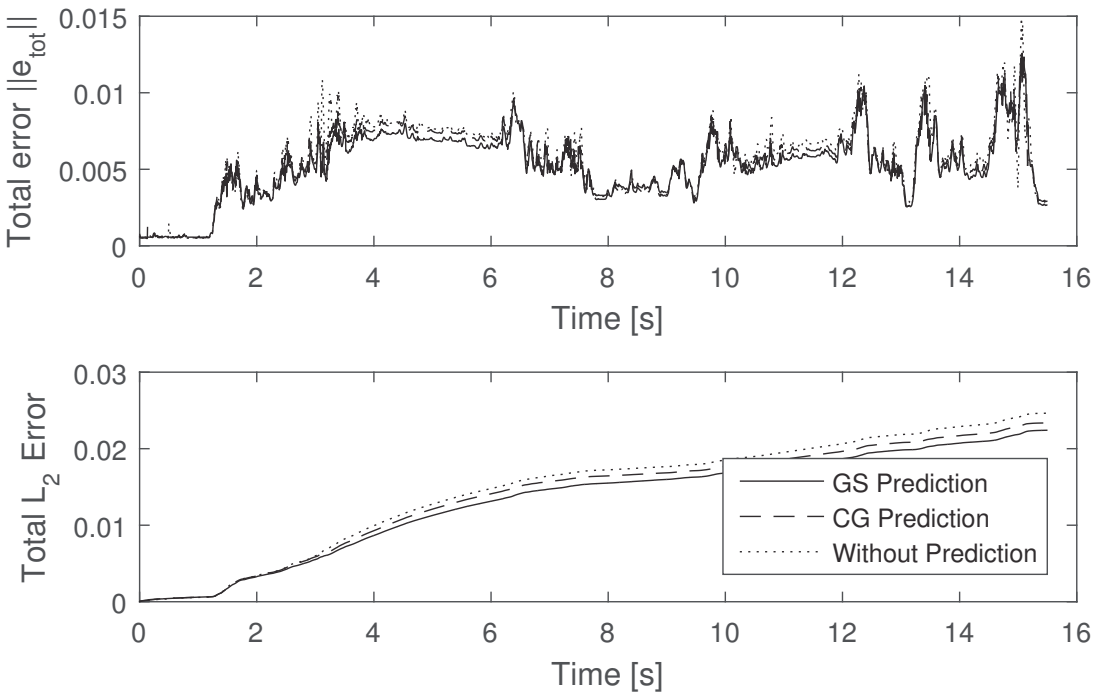


Figure 7.39: Total error magnitude and $\mathcal{L}_2$ norm for Case B2

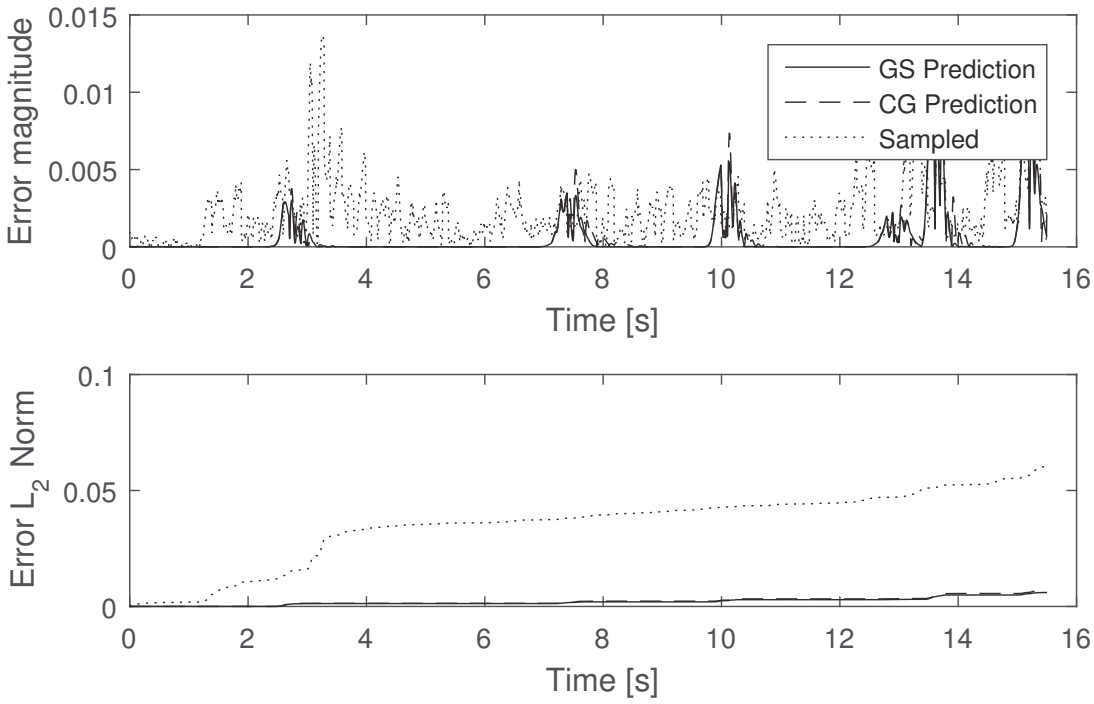


Figure 7.40: Predictor error magnitude and $\mathcal{L}_2$ norm for Case B3

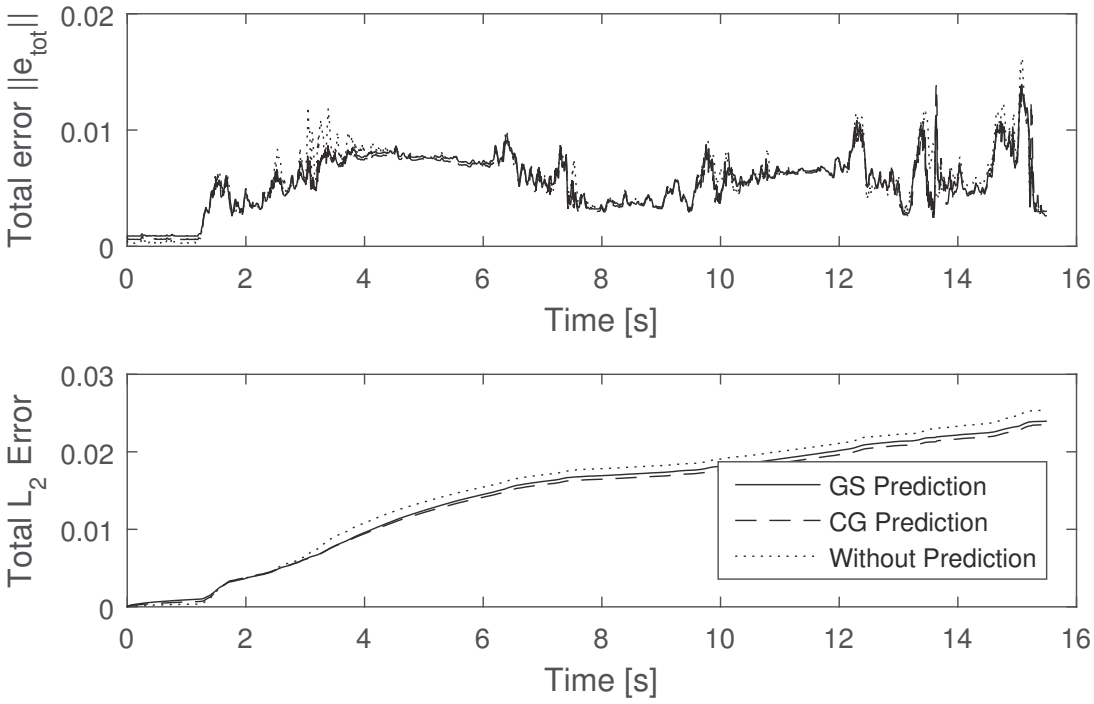


Figure 7.41: Total error magnitude and $\mathcal{L}_2$ norm for Case B3

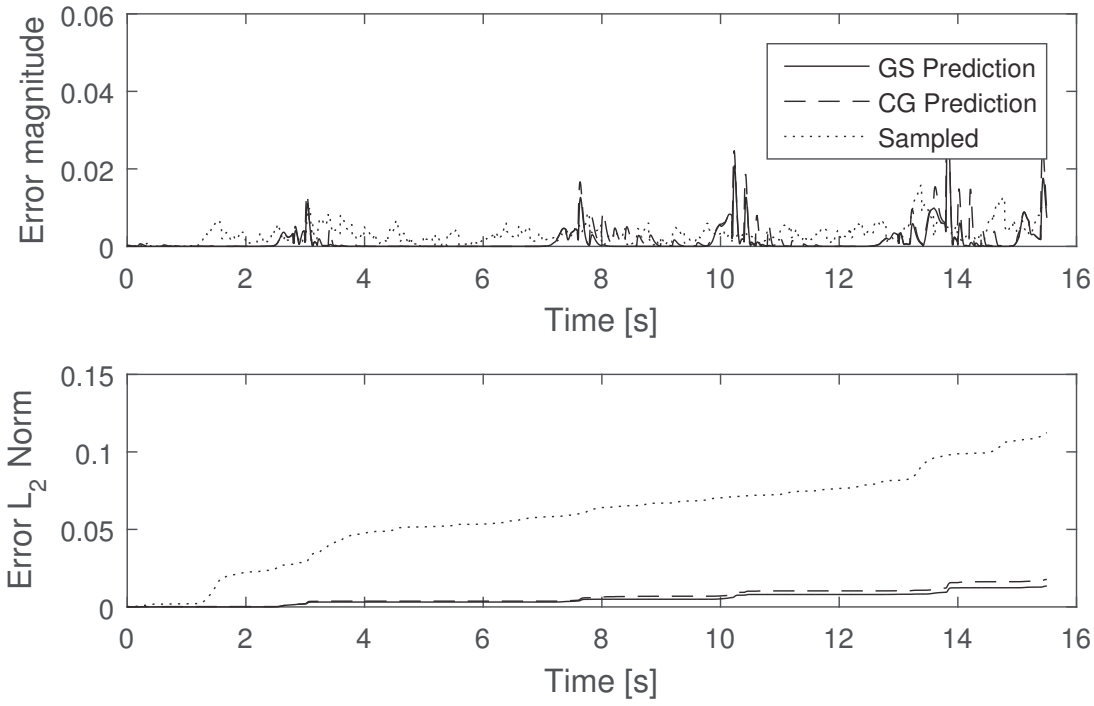


Figure 7.42: Predictor error magnitude and $\mathcal{L}_2$ norm for Case B4

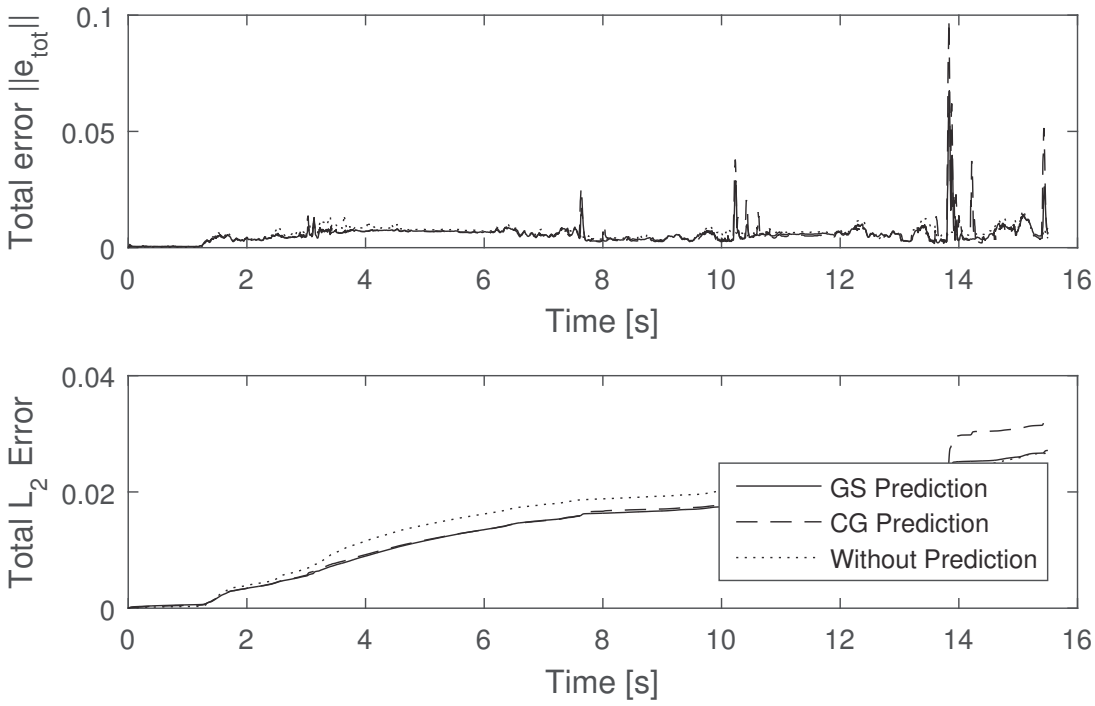


Figure 7.43: Total error magnitude and $\mathcal{L}_2$ norm for Case B4

system with the predictor compared to the system without the predictor improves as the sampling time increases, although the absolute error of the system does increase. The predictor is particularly effective at reducing the peak error.

The unknown nonlinear virtual environment is a much greater challenge. When attempting to track a nonlinear environment with rapidly changing parameters as with the Hunt-Crossley environment model, the parameter adaptation law of the predictor does not track very well. Consequently the system sampling time must be very large (approximately 0.1 s) before the predictor leads to a benefit for the overall system performance. In theory there is a point at which increasing the sampling time will no longer lead to an improvement in the relative performance, as the parameter adaptation law will not have enough information to reliably track the virtual environment parameters. The predictor does show that for a rapidly changing system it can reduce chattering due to the sampling.

The gain-scheduled predictor's performance was not as good as expected. Although the theoretical guaranteed stability range of the gain-scheduled predictor tends to be larger than that for the constant-gain predictor, it does not cover the entire range

for the rate of change of the scheduling variables. Even so, when both predictor are tested with parameters outside of their guaranteed stability range the haptic system was still stable. Clearly the LMIs in Theorem 3 and Theorem 5 are very conservative, leading to a wider stability range than expected. In terms of performance the gain-scheduled predictor does not show any significant improvement over the constant-gain predictor. The gain-scheduled predictor does tend to reduce the error bound slightly more than the constant-gain predictor.

7.3.1 Limitations

The predictor is limited to virtual environments whose parameters do not change too quickly. The parameter adaptation law used in the predictor is intended for constant or slowly-varying parameters, therefore it is not very effective when the parameters change rapidly. The parameter adaptation law was also limited by the number of parameters it could estimate simultaneously. Although it theoretically could estimate any number of parameters, increasing the number of parameters used to describe the virtual environment makes it more difficult for the adaptation law to converge before the parameters change again.

The overall haptic system was tested without actual human operation, as mentioned at the start of the chapter. Generally speaking, the operator tends to introduce a stabilizing effect to the system since their arm provides additional damping [23]. However, to fully verify the effectiveness of the system it would be important to perform a test using a real human operator. Using a more accurate force sensor, or limiting the system to fewer degrees of freedom to avoid sensor cross-talk could allow an operator to test the system.

Although the performance of the predictor generally improves as the sampling time increases, for the nonlinear virtual environment cases A4 and B4, i.e. when $T_d = 0.200$ s, the performance decreased quite significantly. This decrease in performance may be avoidable if the predictor design is tuned differently, for example higher sampling times may require an increase in γ.

7.3.2 Advantages

The predictor is effective for reducing the peaks in the total error, which are very important to the operator's perception of the system. Such peaks would lead to a 'bumping' or 'jostling' sensation which would greatly reduce the realism of the system. The performance improvement due to the predictor is larger when the virtual tool stiffness and damping are more significant, as discussed in Chapter 6. All the experiments were performed with a virtual tool stiffness of zero in order to get an idea of the worst-case performance. In the simulation study increasing the virtual tool stiffness from 0 N/m to 1 N/m changed the performance improvement from essentially 0% to almost 90%. Therefore, separating the virtual tool dynamics from the virtual environment dynamics drastically improves performance.

Although the predictor does not always produce the ideal system dynamics, it does have the advantage of providing near-instantaneous feedback to the user. As the sampling time of the haptic system increases, it reaches a point where the operator can notice the delay in feedback. Even if the predictor does not provide the ideal feedback force, it does provide the operator with a continuous, rapid response to their motion. Maintaining a continuous sensation may be more important than providing a perfect representation of the environment parameters at all times. It would be interesting to investigate how this tradeoff, continuous feedback versus more accurate environment parameters, affects the operator's performance.

Finally, although the total error results for the nonlinear system do not show a great performance improvement, the output of both the constant-gain and gain-scheduled predictors were much closer to the ideal signal. For example, in case B3 the difference between the total error of the system with and without prediction was negligible even though the difference between predicted and sampled signals was quite large. Therefore the effects of the predictor may be more significant if the tracking of the controller were improved.

Chapter 8

Conclusions and Future Work

The goal of this work was to design a predictor and control system that take advantage of an increased sampling rate to improve the transparency of a slowly updating virtual environment. The final results of the simulation and experimental testing are promising.

For an LTI virtual environment with unknown parameters, the performance of the haptic system is excellent. The simulation results suggested significant improvements in transparency, and that was reflected in the experimental results as well. The transparency of the overall haptic system improved by up to 40% in experimental tests, which would be a very noticeable improvement for a human operator.

For a nonlinear virtual environment with unknown parameters, the performance of the haptic system is acceptable. The simulation results suggested that sampling times at or above 0.100 s would be necessary to see improvements in transparency, which was borne out by the experimental results. The experimental results show that the predictor was effective at reducing the error bound, except for a couple extreme cases.

The gain-scheduled predictor resulted in a negligible change in performance compared to the constant gain predictor. Even though the resulting gains from the two design methods were very different, the resulting performance did not change significantly. This indicates that the parameter adaptation law may dominate the predictor response. However, the gain-scheduled predictor design relies heavily on the somewhat arbitrary selection of design parameters. A more rigorous approach to the LPV system analysis could lead to improved gain-scheduled predictor performance.

Overall the effectiveness of the predictor is very situation-dependent. For systems with low-quality haptic devices, or quickly changing parameters, the addition of the predictor will not have a large effect on the transparency. For systems with high-quality haptic devices (with less static friction and more accurate motor torque) and very large computational delays, the addition of the predictor could greatly improve

transparency.

Future Work

There are many potential opportunities to continue the research described in this work. A detailed study of the effects of changing the LMI design parameters on the resulting predictor performance for the complete system would lead to much better insight in designing the predictor gain.

The LPV system analysis and subsequent gain-scheduled predictor design could be greatly improved. There are other methods for solving LPV LMIs that were not used in this work, such as LMI relaxation using B-splines, which could result in better predictor performance. The predictor design could also be improved by using a hybrid predictor method rather than a strictly model-based predictor. A hybrid predictor could change between predictor types in real time depending on the situation, or the predictor could use a weighted average of multiple estimates similar to a Kalman filter. Also, since the performance of the predictor is greatly affected by the parameter adaptation law it would be beneficial to investigate other parameter adaptation methods in order to improve the performance.

Another potential area for further research is to build a virtual surgical training simulator and have medical trainees perform tasks with and without the predictor. Evaluating the actual perceived performance of the predictor is important to determining if it actually meets its goal of improving performance.

References

[1] Z. MA and P. Ben-Tzvi, "RML glove - An exoskeleton glove mechanism with haptics feedback," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 641–652, 2015.

[2] F. Danieau, A. Lecuyer, P. Guillotel, J. Fleureau, N. Mollet, and M. Christie, "Enhancing audiovisual experience with haptic feedback: A survey on HAV," *IEEE Transactions on Haptics*, vol. 6, no. 2, pp. 193–205, 2013.

[3] U. Meier, O. López, C. Monserrat, M. C. Juan, and M. Alcañiz, "Real-time deformable models for surgery simulation: a survey," *Computer methods and programs in biomedicine*, vol. 77, no. 3, pp. 183–197, Mar 2005.

[4] C. Basdogan, S. De, J. Kim, M. Muniyandi, H. Kim, and M. A. Srinivasan, "Haptics in minimally invasive surgical simulation and training," *IEEE Comput. Graph. Appl.*, vol. 24, no. 2, pp. 56–64, 2004.

[5] C. D. Combs, "Medical simulators: Current status and future needs," in *2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, 2010, pp. 124–129.

[6] A. L. Trejos, "A sensorized instrument for minimally invasive surgery for the measurement of forces during training and surgery: Development and applications," Ph.D. dissertation, University of Western Ontario, 2012.

[7] S. S. Nudehi, R. Mukherjee, and M. Ghodoussi, "A shared-control approach to haptic interface design for minimally invasive telesurgical training," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 588–592, 2005.

[8] C. Basdogan, M. Sedef, M. Harders, and S. Wesarg, "VR-based simulators for training in minimally invasive surgery," *IEEE Computer Graphics and Applications*, vol. 27, no. 2, pp. 54–66, 2007.

[9] A. Breda and A. Territo, "Virtual reality simulators for robot-assisted surgery," *European Urology*, vol. 69, no. 6, pp. 1081–1082, 2015.

[10] A. Moglia, V. Ferrari, M. Ferrari, L. Morelli, F. Mosca, and A. Cuschieri, "A systematic review of virtual reality simulators for robot-assisted surgery," *European Urology*, vol. 69, no. 6, pp. 1065–1080, 2015.

[11] C. R. Wagner and R. D. Howe, "Mechanisms of performance enhancement with force feedback," in *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. World Haptics Conference*, 2005, pp. 21–29.

[12] A. M. Okamura, L. N. Verner, C. E. Reiley, and M. Mahvash, "Haptics for robot-assisted minimally invasive surgery," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, M. Kaneko and Y. Nakamura, Eds. Berlin, Heidelberg: Springer, 2010, vol. 66, pp. 361–372.

[13] M. Kibsgaard, K. K. Thomsen, and M. Kraus, "Simulation of surgical cutting in deformable bodies using a game engine," in *International Conference on Computer Graphics Theory and Applications (GRAPP)*, Jan 2014.

[14] D. Zerbato and P. Fiorini, "A unified representation to interact with simulated deformable objects in virtual environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 2710–2717.

[15] J. E. Colgate and G. Schenkel, "Passivity of a class of sampled-data systems: application to haptic interfaces," in *Proceedings of the 1994 American Control Conference*, vol. 3, 1994, pp. 3236–3240.

[16] N. Diolaiti, G. Niemeyer, F. Barbagli, and J. K. Salisbury, "Stability of haptic rendering: Discretization, quantization, time delay, and coulomb effects," *IEEE Transactions on Robotics*, vol. 22, no. 2, pp. 256–268, 2006.

[17] R. J. Adams and B. Hannaford, "Stable haptic interaction with virtual environments," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 465–474, 1999.

[18] P. Naghshtabrizi and J. P. Hespanha, "Designing transparent stabilizing haptic controllers," in *2006 American Control Conference*. IEEE, 2006, pp. 2475–2480.

[19] J. Huang, Y. Shi, and J. Wu, "Transparent virtual coupler design for networked haptic systems with a mixed virtual wall," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 3, pp. 480–487, 2012.

[20] N. Diolaiti, C. Melchiorri, and S. Stramigioli, "Contact impedance estimation for robotic systems," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 925–935, 2005.

[21] M. Makhsous, G. Venkatasubramanian, A. Chawla, Y. Pathak, M. Priebe, W. Z. Rymer, and F. Lin, "Investigation of soft-tissue stiffness alteration in denervated human tissue using an ultrasound indentation system," *The Journal of Spinal Cord Medicine*, vol. 31, no. 1, pp. 88–96, 2008.

[22] B. Hannaford and J.-H. Ryu, "Time-domain passivity control of haptic interfaces," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 1–10, 2002.

[23] A. Abdossalami and S. Sirouspour, "Adaptive control for improved transparency in haptic simulations," *IEEE Transactions on Haptics*, vol. 2, no. 1, pp. 2–14, 2009.

[24] J.-J. E. Slotine and W. Li, "On the adaptive-control of robot manipulators," *International Journal of Robotics Research*, vol. 6, no. 3, pp. 49–59, 1987.

[25] K. Walker, Y.-J. Pan, and J. Gu, "Control gain design for bilateral teleoperation systems using linear matrix inequalities," in *Proceedings of the 2007 IEEE International Conference on Systems, Man and Cybernetics*, 2007, pp. 2588–2593.

[26] H. Werner, *Advanced Topics in Control*. Hamburg, Germany: Hamburg University of Technology, 2015.

[27] G. Hilhorst, E. Lambrechts, and G. Pipeleers, "Control of Linear Parameter-Varying Systems using B-Splines," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 3246–3251.

[28] Y. J. Pan, C. C. de Wit, and O. Sename, "A new predictive approach for bilateral teleoperation with applications to drive-by-wire systems," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1146–1162, 2006.

[29] X. Hou and O. Sourina, "A prediction method using interpolation for smooth six-dof haptic rendering in multirate simulation," in *Proceedings of the 2013 International Conference on Cyberworlds (CW)*, 2013, pp. 294–301.

[30] N. Sakr, N. D. Georganas, J. Zhao, and X. Shen, "Motion and force prediction in haptic media," in *2007 IEEE International Conference on Multimedia and Expo*, 2007, pp. 2242–2245.

[31] H. Arioui, S. Mammar, and T. Hamel, "A smith-prediction based haptic feedback controller for time delayed virtual environments systems," in *Proceedings of the 2002 American Control Conference*, vol. 5, 2002, pp. 4303–4308.

[32] H. Caballero-Barragán, L. P. Osuna-Ibarra, and A. G. Loukianov, "Dynamic predictor for linear time-delay systems with disturbances," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 2290–2295.

[33] V. Lechappé, E. Moulay, and F. Plestan, "Dynamic observation-prediction for lti systems with a time-varying delay in the input," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 2302–2307.

[34] A. Selivanov and E. Fridman, "Predictor-based networked control in the presence of uncertain time-varying delays," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 501–506.

[35] X. Xu, J. Kammerl, R. Chaudhari, and E. Steinbach, "Hybrid signal-based and geometry-based prediction for haptic data reduction," in *2011 IEEE International Workshop on Haptic Audio Visual Environments and Games (HAVE)*, 2011, pp. 68–73.

[36] L. Huijun and S. Aiguo, "Virtual-environment modeling and correction for force-reflecting teleoperation with time delay," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 1227–1233, 2007.

[37] Z. Chen, Y. J. Pan, and J. Gu, "Adaptive robust control of bilateral teleoperation systems with unmeasurable environmental force and arbitrary time delays," *IET Control Theory & Applications*, vol. 8, no. 15, pp. 1456–1464, 2014.

[38] V. Scheinman and J. M. McCarthy, "Mechanisms and actuation," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer International Publishing, 2016, ch. 3, pp. 67–86.

[39] K. Waldron and J. Schmiedeler, "Kinematics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer International Publishing, 2016, ch. 1, pp. 9–33.

[40] R. Featherstone and D. E. Orin, "Dynamics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer International Publishing, 2016, ch. 2, pp. 35–65.

[41] H. K. Khalil, *Nonlinear systems*, 3rd ed. Upper Saddle River, N.J.: Prentice Hall, 2002.

[42] D.-W. Gu, P. H. Petkov, and M. M. Konstantinov, *Robust Control Design with MATLAB*, ser. Advanced Textbooks in Control and Signal Processing. Springer London, 2013.

[43] L. A. Jones and H. Z. Tan, "Application of psychophysical techniques to haptic research," *IEEE Transactions on Haptics*, vol. 6, no. 3, pp. 268–284, 2013.

[44] B. Yao, "Integrated direct/indirect adaptive robust control of siso nonlinear systems in semi-strict feedback form," in *Proceedings of the 2003 American Control Conference*, vol. 4, 2003, pp. 3020–3025.

[45] M. Lazeroms, G. Villavicencio, W. Jongkind, and G. Honderd, "Optical fibre force sensor for minimal-invasive-surgery grasping instruments," in *Proceedings of 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 1, 1996, p. 235.

Appendix A

Simulink Model

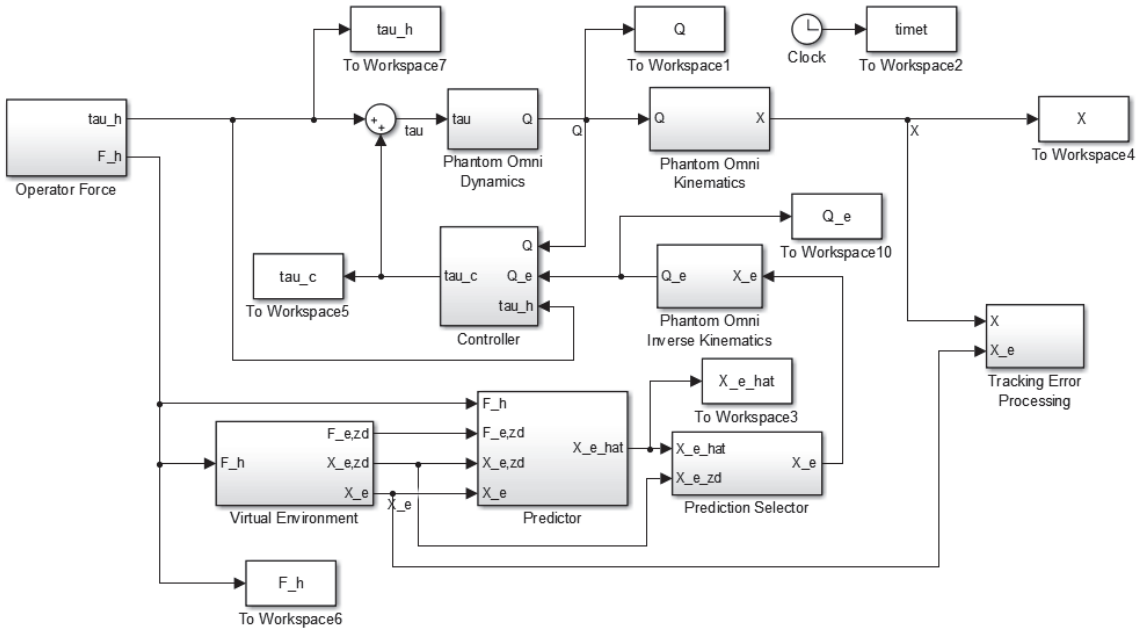


Figure A.1: Top-level view of the Simulink model

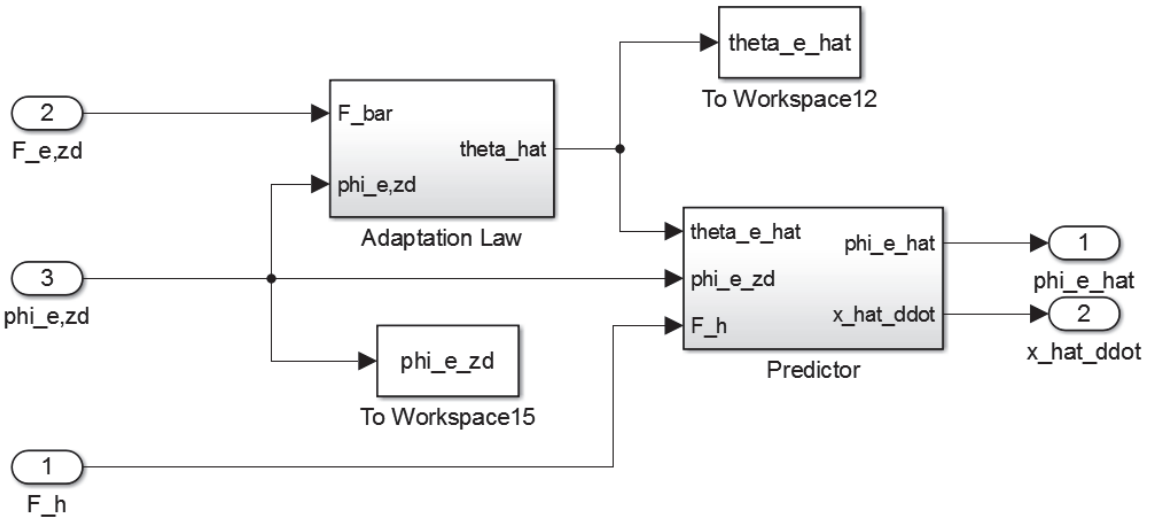


Figure A.2: Simulink model predictor structure

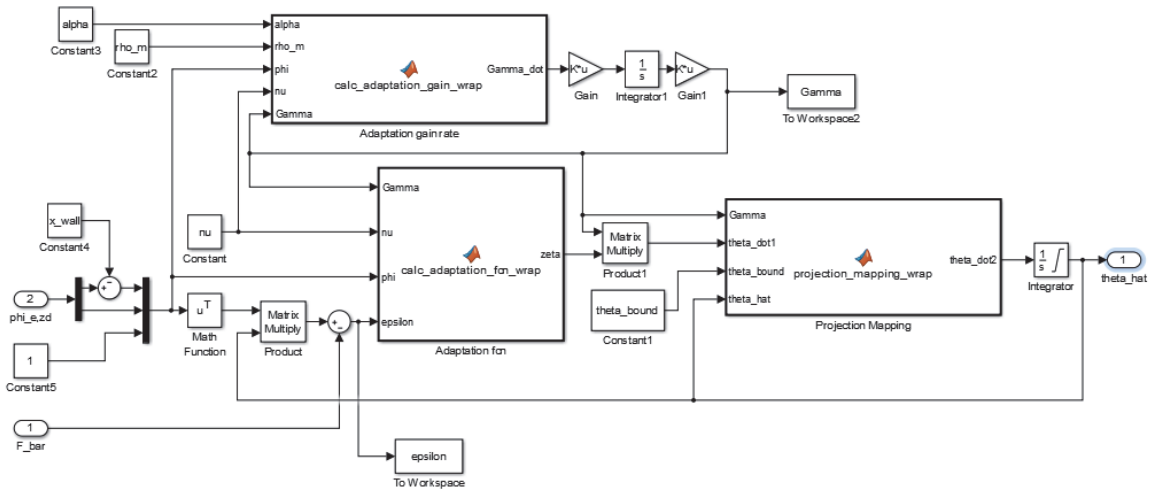


Figure A.3: Simulink model parameter adaptation algorithm

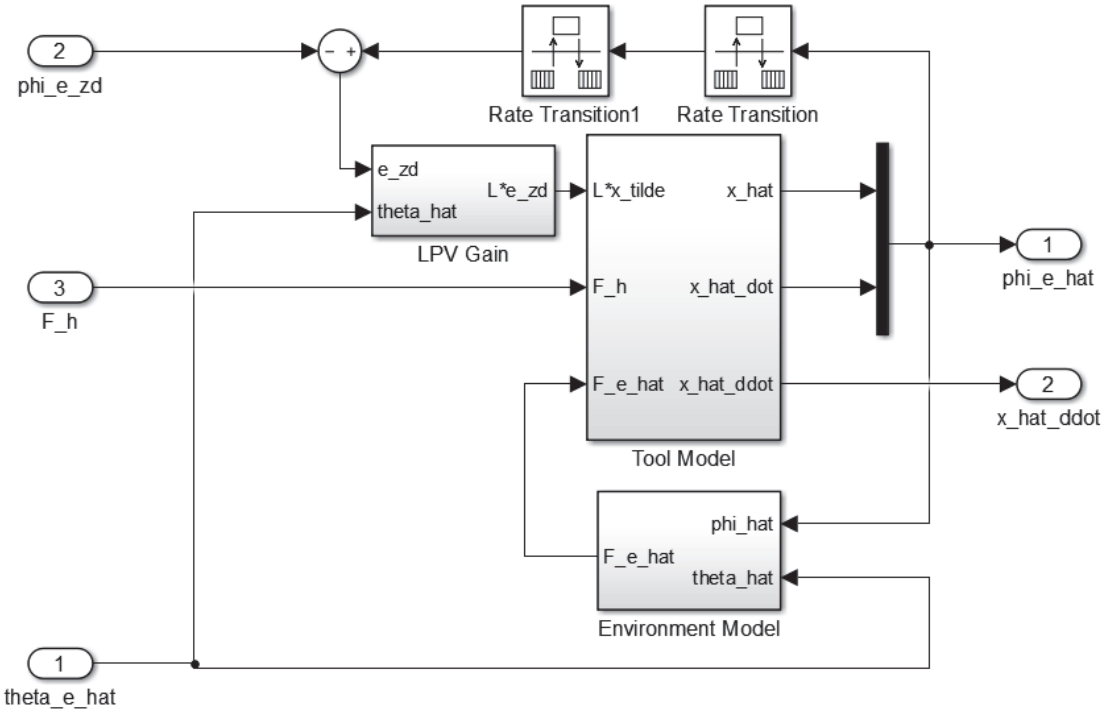


Figure A.4: Simulink model predictor dynamics

Appendix B

Author's Publications

Peer-Reviewed

S. Forbrigger and Y.-J. Pan, "Improved transparency for haptic systems with complex environments," *Proceedings of the IFAC 2017 World Congress*, 2017. (Accepted)

M. Zou, Y.-J. Pan, **S. Forbrigger** and U. Ahmad, "Adaptive robust control for bilateral teleoperated robotic manipulators with arbitrary time delays," *2016 2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, Rawalpindi, 2016, pp. 105-111.

Z. Chen, Y.-J. Pan, J. Gu and **S. Forbrigger**, "A Novel Multilateral Teleoperation Scheme with Power-Based Time-Domain Passivity Control," *Transactions of the Institute of Measurement and Control*, 2016. (Published online)

Other Publications

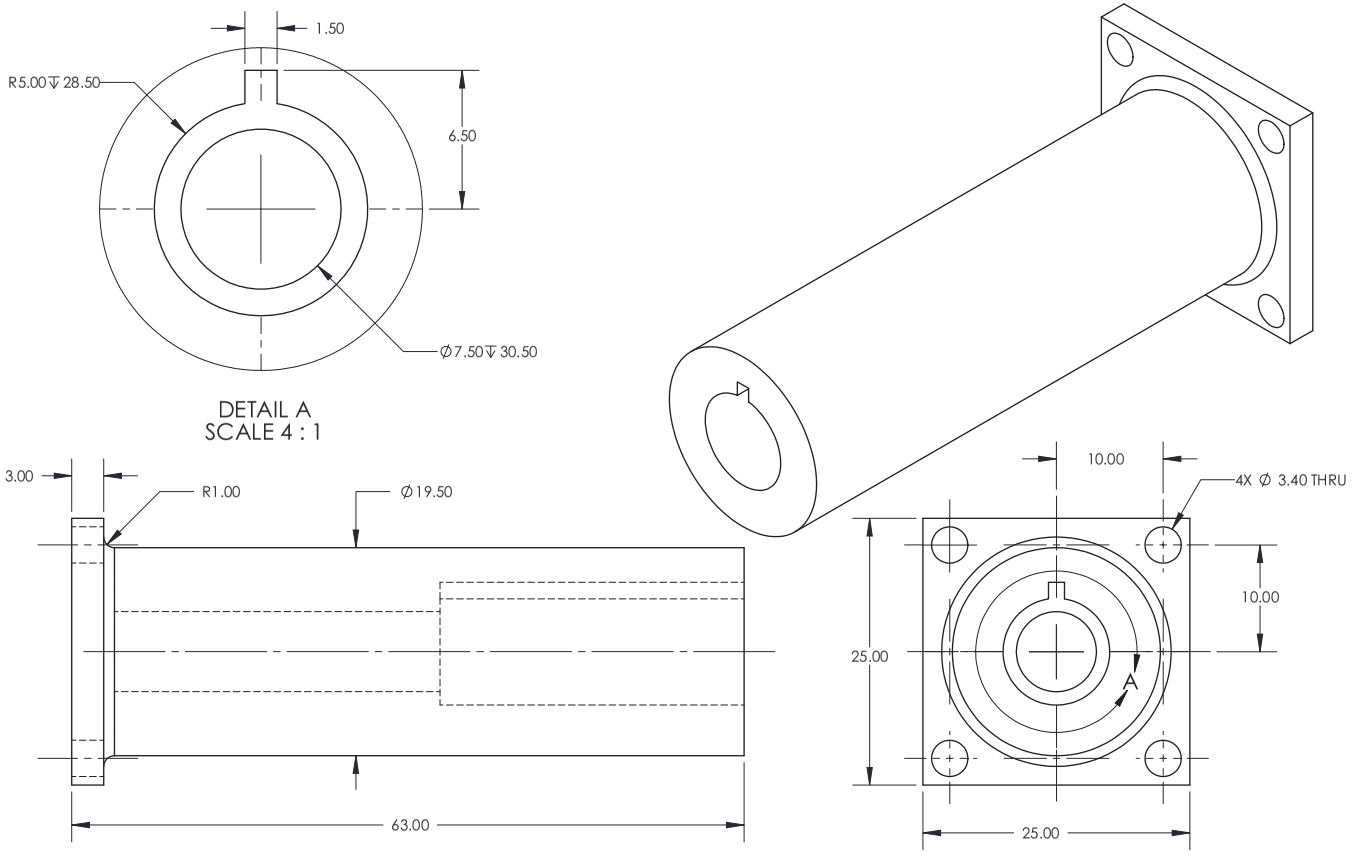
S. Forbrigger and Y.-J. Pan, "Improved haptics for complex virtual environments," (Presentation) *2017 Dalhousie University Mechanical and Materials Engineering Conference (DUMMEC)*, 2017.

S. Forbrigger and Y.-J. Pan, "Haptic interfaces: Challenges, control, and applications," (Poster) *2016 Dalhousie University Mechanical Engineering Research Conference (MERC)*, 2016.

Appendix C

CAD Drawings of the Force Sensor Mount

Note: Drawings not to scale due to page margin restrictions.

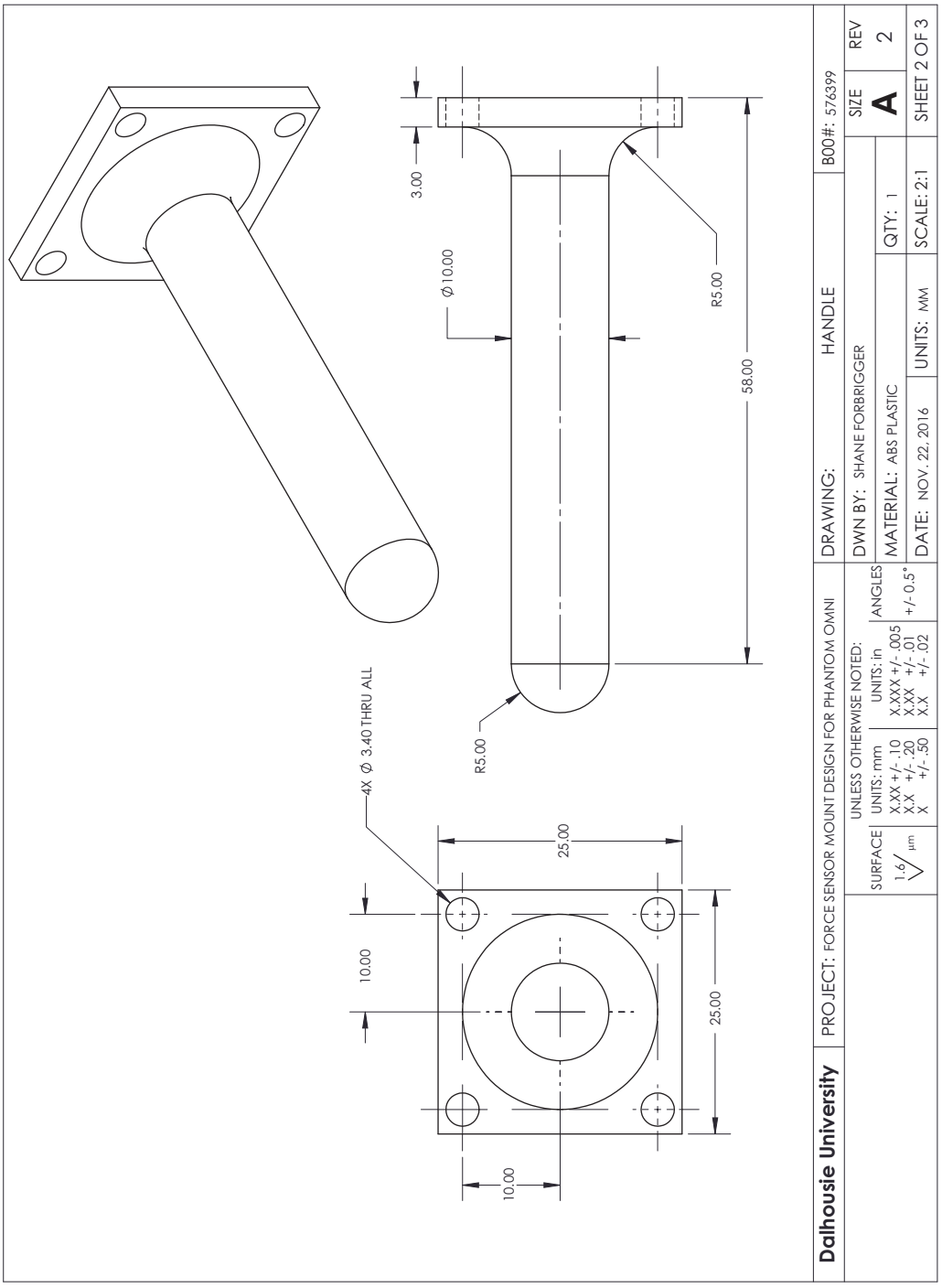


Dalhousie University	PROJECT: FORCE SENSOR MOUNT DESIGN FOR PHANTOM OMNI	DRAWING: BASE MOUNT	B00#: 576399

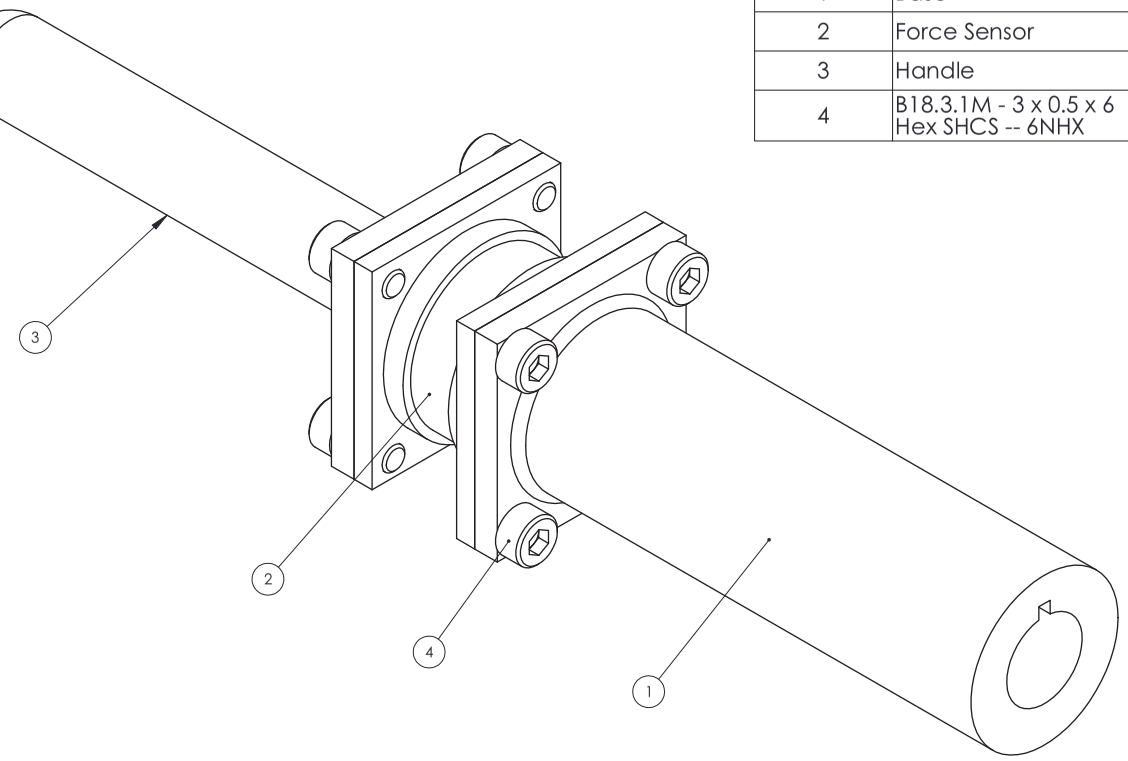
UNLESS OTHERWISE NOTED:

SURFACE	UNITS: mm	UNITS: in	ANGLES
1.6 µm	X.XX +/- .10 X.X +/- .20 X +/- .50	X.XXX +/- .005 X.XX +/- .01 X.X +/- .02	+/- 0.5°

DWN BY: SHANE FORBRIGGER			SIZE	REV
MATERIAL: ABS PLASTIC		QTY: 1	A	2
DATE: NOV. 22, 2016	UNITS: MM	SCALE: 2:1	SHEET 1 OF 3	


3.00
Ø10.00
R5.00
58.00
R5.00
4X Ø 3.40 THRU ALL
25.00
10.00
25.00
10.00
Dalhousie University
PROJECT: FORCE SENSOR MOUNT DESIGN FOR PHANTOM OMNI
DRAWING: HANDLE
BOO#: 576399
DWN BY: SHANE FORBRIGGER
MATERIAL: ABS PLASTIC
QTY: 1
DATE: NOV. 22, 2016
UNITS: MM
SCALE: 2:1
SIZE A
REV 2
SHEET 2 OF 3
SURFACE 1.6 µm
UNLESS OTHERWISE NOTED:
UNITS: mm
X.XX +/- .10
X.X +/- .20
X +/- .50
UNITS: in
X.XXX +/- .005
X.XX +/- .01
X.X +/- .02
ANGLES
+/- 0.5°

ITEM NO.	PART NUMBER	QTY.
1	Base	1
2	Force Sensor	1
3	Handle	1
4	B18.3.1M - 3 x 0.5 x 6 Hex SHCS -- 6NHX	8



Dalhousie University	PROJECT: FORCE SENSOR MOUNT DESIGN FOR PHANTOM OMNI	DRAWING: SENSOR ASSEMBLY		B00#: 576399	
UNLESS OTHERWISE NOTED: SURFACE 1.6 µm; UNITS: mm X.XX +/- .10, X.X +/- .20, X +/- .50; UNITS: in X.XXX +/- .005, X.XX +/- .01, X.X +/- .02; ANGLES +/- 0.5°		DWN BY: SHANE FORBRIGGER		SIZE A	REV 2
		MATERIAL: ABS PLASTIC	QTY: 1		
		DATE: NOV. 22, 2016	UNITS: MM	SCALE: 2:1	SHEET 3 OF 3