ROBUST ITERATIVE LEARNING CONTROL FOR LINEAR PARAMETER-VARYING SYSTEMS WITH TIME DELAYS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Florian Maurice Browne III

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Neera Jain, Chair

School of Mechanical Engineering, Purdue University

Dr. George Chiu

School of Mechanical Engineering, Purdue University

Dr. Martin Corless

School of Aeronautics and Astronautics, Purdue University

Dr. Douglas Bristow

Department of Mechanical and Aerospace Engineering, Missouri University of Science and Technology

Approved by:

Dr. Nicole Key

Associate Head Graduate Studies

ACKNOWLEDGMENTS

There are many people that I need to thank for their support and guidance throughout my graduate studies. First, and foremost, I would like to thank my advisor Professor Neera Jain. She has taught me so much and her guidance and mentorship has helped me become the person that I am today. I also want to thank Professor George Chiu for his guidance with my research since I arrived on campus in 2015. Both professors were instrumental in helping me progress through my graduate studies and have helped me mature as an engineer and as an individual. I have learned many lessons from them that I will continue to use for the rest of my life. Furthermore, I would like to thank the other members of my committee – Professors Martin Corless and Doug Bristow – for their valuable insights and encouragement throughout my graduate studies.

The work I present in this dissertation would not be possible without the sponsorship and support of Castrip, LLC. I want to extend my deepest gratitdue to Brad Rees, Wal Bledje, and all of the other Nucor and Castrip, LLC. employees for teaching me about their process and giving me the opportunity to test my controllers at their plant. Their insight into how their process works and also how the industry operates helped me better understand the problems that can occur during the casting process.

I would not have been able to finish this dissertation without the support, encouragement, and feedback from my colleagues in the Jain Research Lab – Austin, Yesh, Akash, Trevor, Ana, Aaron, Karan, Jack, and Matt – and in the Ray W. Herrick Labs. Both groups fostered a collaborative and supportive environment that helped me grow as a researcher. I will always value the friendships that I formed during my time at Purdue.

To my family, I would have never completed my studies without your support. Even though I moved three states away, I always knew that I could count on my parents, Florian and Ruth, and brother, Ben, if I ever needed them for anything.

Finally, and most importantly, I want to thank my fiance Sarah for giving me the moral support I needed during the hardest parts of the process. I can't wait to begin the next chapter of our life together and see where the future takes us.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

Figure Page

GLOSSARY

drive side	side of the strip and/or casting rolls farthest from the operator
mushy	semi-solid metal with temperature between the solidus and liquidus temperatures
operator side	side of the strip and/or casting rolls closest to the operator
tilt	drive side position minus operator side position
wedge	a strip profile perturbation where one side of the strip is wider than the other

ABSTRACT

Browne, Florian M. Ph.D., Purdue University, August 2020. Robust Iterative Learning Control for Linear Parameter-Varying Systems with Time Delays. Major Professor: Neera Jain, School of Mechanical Engineering.

The work in this dissertation concerns the construction of a robust iterative learning control (ILC) algorithm for a class of systems characterized by measurement delays, parametric uncertainty, and linear parameter varying (LPV) dynamics. One example of such a system is the twin roll strip casting process, which provides a practical motivation for this research. I propose three ILC algorithms in this dissertation that advance the state of the art. The first algorithm compensates for measurement delays that are longer than a single iteration of a periodic process. I divide the delay into an iterative and residual component and show how each component effects the asymptotic stability properties of the ILC algorithm. The second algorithm is a coupled delay estimation and ILC algorithm that compensates for time-varying measurement delays. I use an adaptive delay estimation algorithm to force the delay estimate to converge to the true delay and provide stability conditions for the coupled delay estimation and ILC algorithm. The final algorithm is a norm optimal ILC algorithm that compensates for LPV dynamics as well as parametric uncertainty and time delay estimation error. I provide a tuning method for the cost function weight matrices based on a sufficient condition for robust convergence and an upper bound on the norm of the error signal. The functionality of all three algorithms is demonstrated through simulated case studies based on an identified system model of the the twin roll strip casting process. The simulation testing is also augmented with experimental testing of select algorithms through collaboration with an industrial sponsor.

1. INTRODUCTION

1.1 Motivation

Near-net-shape manufacturing processes are becoming a major contributor to the reduction of both environmental and economic costs in the industrial sector [1]. For the steel industry, twin roll strip casting is one of the most prominent near-net-shape manufacturing processes. In twin roll casting, molten steel is poured directly onto the surface of two counter-rotating casting rolls which simultaneously cool and compress the steel into a strip that is approximately the desired thickness. However, combining the cooling and compression steps into a single continuous casting process introduces coupling between the rapid thermal solidification dynamics and the mechanical stiffness of the resulting steel strip.

Introducing coupling between multiple manufacturing steps is a common approach used by near-net-shape manufacturing processes to reduce operating costs. The coupling, however, can make such processes more difficult to model and control. Some common features that need to be addressed when controlling a near-net-shape manufacturing process are their variability as a function of operating condition and susceptibility to parametric uncertainty. Additionally, these processes can experience significant measurement delays due to the inability to co-locate sensors and actuators. Given that these manufacturing processes are commonly periodic, a useful control approach to overcome these challenges is iterative learning control (ILC).

1.2 Iterative Learning Control

The concept of having the system automatically learn and improve its performance after each iteration has been around for more than 50 years. A patent filed in 1967 by Murray Garden described a control system where a command signal is stored in

memory and iteratively "corrected by an amount related to the error" [2]. The first academic contribution published on the topic was written in Japanese by Uchiyama [3]. The term "iterative learning control" was first applied in 1984 [4], the same year ILC became an active area of research [5–7]. Since the papers in 1984, ILC has been a very rich area of research, with many surveys [8, 9] and books [10, 11] written on the topic.

In industrial manufacturing processes, the desire is that the system not only has good performance but that it is robust to changes in the process. As outlined by the motivating example, many manufacturing processes require control algorithms that are robust to model uncertainty and that can overcome measurement delays, which may be time-varying.

Many researchers have studied robust ILC algorithms in the past [12–15] with robustness defined relative to various signals and sources of uncertainty. In [12], a higher-order ILC algorithm that is robust to initial state inaccuracies is proposed. In [13], an ILC algorithm is synthesized using an H_∞ approach that explicitly accounts for uncertain system knowledge. In [14], 2-D systems theory is used to analyze the robustness of a D-type ILC algorithm when applied to systems with multiple state time delays and initial output shifts. In [15], an $\mathcal{L}_1$ adaptive feedback controller is combined with an ILC algorithm to compensate for low-frequency parametric uncertainty in the time domain.

Norm optimal ILC is a type of ILC algorithm that has recently been used to improve the robust performance of the closed loop system in the presence of multiplicative uncertainty [16–20]. The norm optimal ILC formulation uses a cost function to design the ILC filters so that the closed loop system achieves optimal performance with respect to the cost function. The formulation is useful because parametric uncertainty is incorporated into the design of the cost function's weighting matrices. A significant gap in existing norm optimal ILC literature, however, is that no one has examined how delay uncertainty effects the norm optimal ILC formulation.

Some recent developments have been focused on developing ILC algorithms for linear parameter-varying (LPV) systems [21–24], which is a useful way to model systems whose dynamics change as a function of the operating condition [25, 26]. de Rozario et al. show that an ILC algorithm that directly accounts for the LPV dynamics of the system is able to achieve better accuracy and convergence rates than a linear time-invariant ILC algorithm that has been made robust to model uncertainty [24]. These algorithms, however, assume perfect LPV models of the process and do not address time-varying measurement delays.

Regarding the effects of measurement delays, there is limited research concerning the effect that measurement delays have on ILC algorithms and overall process stability. Algorithms have been developed to compensate for state delays [27–32]. These algorithms, however, focus on delays that occur within a single iteration of the process. A higher order ILC algorithm is needed for control of processes that have measurement delays longer than a single iteration of the process. Higher order ILC algorithms were introduced in [33] and were shown to have better convergence performance than first-order ILC algorithms.

Furthermore, a control algorithm that is robust to time-varying measurement delay will typically require an estimate of the delay itself. Correlation-based methods [34–36] are common algorithms used to estimate the time delay within a process. The periodicity of a process, however, makes correlation-based methods unreliable when the delay is multiple periods in length. This is because the periodicity causes the correlation function to have a local maximum for every period within the search window.

1.3 Thesis Objective

In this thesis, I contribute three ILC algorithms that advance the state of the art of ILC research. These algorithms focus on compensating for time delay estimation errors as well as LPV dynamics and parametric uncertainty. The first algorithm is

developed to compensate for measurement delays that are longer than one iteration of a periodic process. To incorporate the delay into the ILC algorithm, I develop a delay model that has both an iterative component and a residual component. This enables the estimation of each component separately, and their sum represents the total measurement delay in the system. I then use that model to design and synthesize an ILC algorithm that accounts for the measurement delay and is robust to delay estimation error.

The second algorithm is a coupled delay estimation and ILC algorithm that compensates for time-varying measurement delays using an adaptive delay estimate. The adaptive delay estimation algorithm is designed such that the delay estimate converges to the true delay as the number of process iterations increases, resulting in improved tracking error performance.

The final contribution is a norm optimal ILC algorithm that compensates for LPV dynamics, parametric uncertainty, and measurement delay uncertainty. I provide a tuning method that details how the weighting functions of the cost function can be designed in order to achieve error attenuation, input saturation avoidance, and robust convergence.

All three algorithms' functionality is demonstrated through simulated case studies on an system model that is motivated by the twin roll strip casting process. The simulation results are further augmented with experimental testing of select algorithms at an industrial sponsor's site.

1.4 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, I discuss the twin roll strip casting process and identify some of the key parameters that are used to control the process. Chapter 3 introduces two ILC algorithms for time delay systems. Chapter 3 also illustrates the performance of both algorithms when applied to the twin roll strip casting process. Chapter 4 then describes a norm-optimal LPV ILC algorithm

that can be used to account for parametric uncertainty and linear parameter-varying dynamics. Finally, I present conclusions and discuss potential future research directions in Chapter 5.

2. TWIN ROLL STRIP CASTING

This chapter provides a description of the twin roll strip casting process that motivates the research in this thesis and presents modeling information about the process that I published with Professors George T.-C. Chiu and Neera Jain in the *ASME Journal of Dynamic Systems Measurements and Control* [37]. A table of nomenclature used in this chapter is provided in Table 2.1.

Table 2.1. Nomenclature for the twin roll strip casting model

Variables		**Subscripts**	
c	Specific heat of steel	1	Liquidus
f_m	Mushy fraction	2	Solidus
F	Force	g	Gap
h	Heat transfer coefficient	k	Kiss
k	Effective thermal conductivity	ℓ	Liquid
L	Latent heat	Lev	Steel pool level
R	Radius	m	Mushy
T	Temperature	O	Outer boundary
x	Horizontal distance	R	Steel at the roll
y	Height from the nip	$Roll$	Roll surface
Z	Transverse direction	s	Solid
ε	Switching parameter		
ρ	Density		
θ	Angle		
Ω	Rotational speed		

2.1 Modelling Introduction

Twin-roll strip casting is a near-net-shape manufacturing process used in the steel industry to improve energy efficiency and reduce operating costs [38]. It requires just one-tenth of the facility space and reduces the energy consumption by a factor of nine, as compared to traditional steel casting [39]. One reason for this is that in traditional casting, molten steel is first solidified into thick slabs and then reheated and compressed via a series of rollers to create thin strips of steel. On the other hand, in twin-roll casting, molten steel is directly poured onto the surface of two casting rolls which simultaneously cool and compress the steel into a strip with a thickness of $1-3$ millimeters. In this process, the steel solidification and accompanying casting dynamics evolve within a fraction of a second [40]. This suggests that an accurate model of the process will require simultaneous consideration of the solidification (thermal) dynamics with the rolling (mechanical) dynamics.

Combining these two steps into a single continuous casting process also introduces coupling between the control inputs—the casting speed, Ω, and the gap distance between the rolls, x_G—and their effect on the solidification dynamics. To address this coupling, I require a simple and accurate model that characterizes how Ω and x_G affect the periodic steady state behavior of the process. Many researchers have modeled the solidification process in twin-roll casting [41–52], but developing a reduced-order model requires a balanced modeling approach that captures the relevant input-output dynamics of the process while using a relatively small number of dynamic states. Some researchers [41–43] propose simple models of the twin-roll casting process, but these models assume an abrupt transition from liquid to solid steel when, in reality, this transition involves the storage of latent heat in a two-phase region known as mushy steel. Characterizing the amount of mushy steel in the steel pool is essential to understanding the solidification dynamics of the process because the size of the mushy layer affects the overall cooling rate of the steel [45]. Mushy steel can also have a significant impact on the compression dynamics at the exit of the casting rolls

because the liquid portion of the steel creates a region that is much easier to compress than a completely solid strip [46, 53, 54]. This is important in practice because, as discussed in [53], allowing small amounts of mushy steel to remain in the strip as it leaves the casting rolls can reduce chatter during casting.

Other researchers [47–52] have developed high-resolution numerical simulations of the solidification process that include the mushy region. Their models, however, are too complex to be used for control design or real-time simulation. For example, Santos et al. derived a model with over 400 states and Liu et al. built a model for use in ANSYS. This level of detail is useful for creating high fidelity models of the system but requires computational resources that may not always be available. For that reason, I propose a reduced-order model that characterizes the dynamics of the process, including the mushy region, in a computationally efficient manner.

In [55], I proposed a model of the solidification dynamics using a lumped parameter moving boundary approach. In this chapter, I extend my preliminary work by modeling the thermal dynamics of the twin-rolls themselves as well as the coupling of these dynamics with the steel solidification dynamics. Given the sensing limitations inherent in a twin-roll casting process, e.g. high molten steel temperatures that can damage sensors placed in or near the steel pool, validation of such a model is difficult. Nonetheless, I compare the water temperature leaving the rolls in the proposed model with water temperature measurements obtained from an industrial twin-roll strip casting facility operated by Nucor Corporation. Finally, I illustrate how the roll speed and gap distance influence the system dynamics through simulated case studies.

2.2 Background

In twin-roll casting, molten steel is poured onto the surface of two casting rolls, covering them up to an angle, θ_{Lev}. Thermal energy is then transferred from the steel, through the rolls, and into a set of water cooling channels which extract the

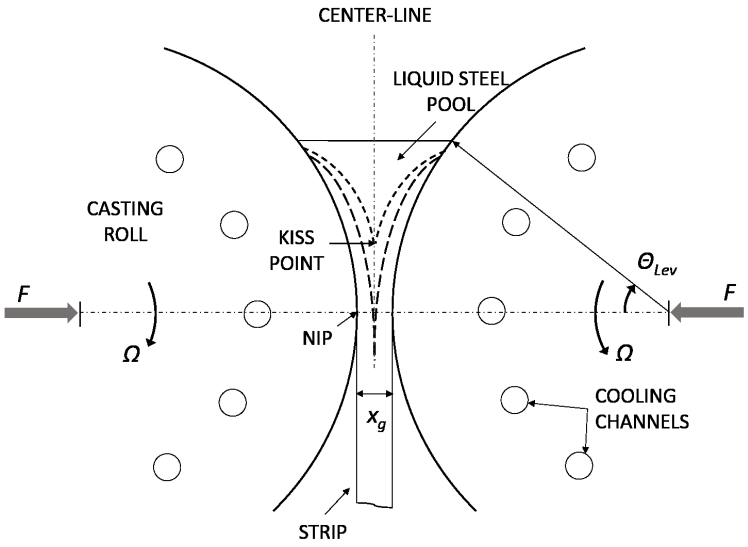


Figure 2.1. A twin-roll strip caster with dashed lines representing the boundaries between the liquid, mushy, and solid steel phases.

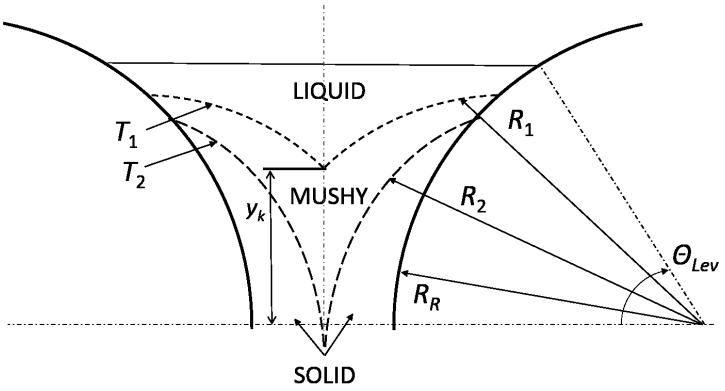


Figure 2.2. The moving boundaries, R_1 and R_2, corresponding to the liquidus, T_1, and solidus, T_2, isotherms respectively.

heat from the roll assembly. As the heat leaves the steel, the liquid steel begins to solidify, causing it to transition into a two-phase state known as mushy steel. The mushy steel then cools further and completely solidifies. Figure 2.1 shows a schematic of the twin-roll casting process after the shell formation reaches a quasi-steady state.

The phase transition of steel, from liquid to solid, begins when the steel cools to below the liquidus temperature and concludes when the steel temperature drops below the solidus temperature. Within our model, steel with a temperature above the liquidus temperature, T_1, is treated as completely liquid. Likewise, steel below the solidus temperature, T_2, is treated as completely solid. The remaining volume of steel, whose temperature is between T_1 and T_2, is treated as mushy. Thus, the steel pool is divided into three regions – liquid, mushy, and solid – with borders coinciding with the T_1 and T_2 isotherms as shown in Fig. 2.2.

Mushy and solid shells begin to form on each roll's surface as the rolls rotate through the steel pool; they continue to grow as the rolls rotate and adhere to each roll's surface. A completely solid width of steel is formed when the micro-structures of the mushy and solid shells from each roll intersect at the center-line. If the width is larger than the desired strip thickness, the steel must be compressed to achieve the desired thickness. This compression is achieved by applying a force, F, to both rolls using a set of actuators that are in line with the *nip*. The nip is defined as the location of the minimum displacement between the rolls, as shown in Fig. 2.1. The amount of force required for the compression is dependent on the location of the *kiss point*, which is defined as the intersection point between the two mushy shells at the centerline. The location of the kiss point relative to the nip is, in turn, dependent on the rolls' rotational speed, Ω and the gap distance between the rolls, x_g [41, 44]. I define the *nip region* as the volume of steel located between the nip and the kiss height, y_k. Within this region, the force required to compress the steel is influenced by the amount of mushy steel present, which is dependent on the growth rate of the solid steel shell.

I am interested in developing a computationally efficient model that captures how both the rotational speed and the gap distance affect one's ability to manufacture a steel strip of a desired thickness. To achieve this, I must understand how Ω and x_G influence the locations of the liquidus and solidus isotherms, which define both the kiss height and the amount of mushy steel present in the nip region. In the next

section, I discuss the modeling approach that I use to predict the locations of these isotherms.

2.3 Steel Pool Model

I model the locations of the liquidus and solidus isotherms using a lumped parameter moving boundary approach. This approach is commonly used in models of multi-phase flows to simplify the physics and track the boundaries between the phases using thermodynamic analysis techniques [56, 57]. The twin-roll casting process is well suited to be modeled with this approach because the steel pool can be divided into three distinct regions – liquid, mushy, and solid – by defining boundaries at the solidus and liquidus isotherms.

To begin, I assume that the physics of the solidification process are mirrored onto both rolls. As a result, the steel pool is divided in half, and I model the process acting on one roll with the center-line of the pool treated as the outer boundary of each half pool. The half pool model is then divided into smaller, angular discretized control volumes, which I call *slices*, with an angular thickness $\delta\theta$ as shown in Fig. 2.3. Within each slice, I assume that the mushy and solid shells adhere to the roll's surface as the roll rotates and that there is no slippage between the shells and the roll. The liquid steel is assumed to be separate from the mushy and solid shells. As the rolls rotate, the amount of liquid steel in each slice changes to fill the entire volume of the slice, adjusting for changes in the size of the mushy and solid shells.

2.3.1 Governing Equations

I define a lumped parameter set within each slice to model the dynamics through a combination of energy balances and boundary continuity equations.

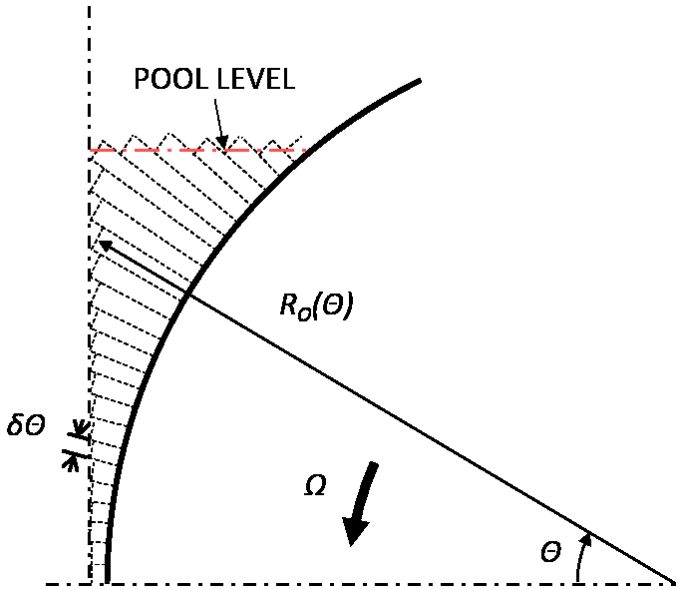


Figure 2.3. A schematic of a half steel pool model divided into control volume slices with angular thickness $\delta\theta$.

Energy Balance.

The energy balance for each region of steel is represented by the equation

$$\frac{\partial \rho c T}{\partial t} = \bigtriangledown \left(k \bigtriangledown T \right) + S \ , \tag{2.1}$$

where ρ is the density, k is the effective thermal conductivity [58], T is the temperature, and c is the specific heat corresponding to the steel phase. The variable S represents the heat source associated with phase transition and is only necessary when modeling the mushy region. The heat source characterizes the release of latent heat needed to fully solidify the steel within the mushy region. Similar to the approaches used in [47] and [59], the concept of pseudo-specific heat is used here to model the release of latent heat. The concept of pseudo-specific heat is predicated on the assumption that the release of the latent heat needed to fully solidify the steel varies only with respect to temperature. Then the latent heat model can be combined with the specific heat model to give an effective specific heat model. Both [47] and [59] propose using Scheil's equation to model how the release of latent heat relates to the

temperature of the mushy region. This representation, however, requires a model of the eutectic properties of the steel, which introduces more complexity to the model and makes it more difficult to tune. To simplify the model, I assume that the amount of latent heat released during the solidification process is linear with respect to the temperature of the mushy steel. This assumption is acceptable because the primary goal of this model is to characterize the total fraction of mushy steel in the strip, rather than the liquid fraction of the mushy region itself. Our approach results in a pseudo-specific heat given by

$$c_m = c + \frac{L}{T_1 - T_2} \ , \tag{2.2}$$

where T_1 is the liquidus temperature, T_2 is the solidus temperature, and L is the latent heat of fusion.

Within each slice, the total rate of change of the energy in each steel phase is obtained by integrating Eqn. (2.1) over the volume of each phase as shown in Eqn. (2.3). This equation is simplified by substituting $dV = rdrd\theta dz$ and assuming that the parameters are uniform in both the angular (θ) and transverse (z) directions of the roll for a given slice. The result is Eqn. (2.4), where Z denotes the transverse length of the slice along the length of the roll and $\delta\theta$ denotes the angular thickness of the slice.

$$\iiint_V \frac{\partial \rho c T}{\partial t} dV = \iiint_V \bigtriangledown \left(k \bigtriangledown T\right) dV \tag{2.3}$$

$$Z\delta\theta \int \frac{\partial \rho c T}{\partial t} r dr = \iiint_V \bigtriangledown \left(k \bigtriangledown T\right) dV \tag{2.4}$$

Leibniz's rule and the product rule are then applied to the left hand side of Eqn. (2.4) resulting in Eqn. (2.5) for the mushy region. The liquid and solid regions are similarly simplified as given by Eqns. (2.6) and (2.7). While the control inputs do not explicitly appear in either equation, note that R_O is dependent on x_G, the gap distance. Moreover, the roll speed Ω dictates the duration of time that the

steel is present in each slice. More details regarding the roll speed's influence on the system dynamics are given in Section 2.5.2.

$$\begin{aligned} &Z\delta\theta\left(\frac{\rho_m c_m}{2}(R_1^2-R_2^2)\frac{dT_m}{dt}+\rho_m c_m(T_m-T_1)R_1\frac{dR_1}{dt}\right)\\ &+Z\delta\theta\rho_m c_m(T_2-T_m)R_2\frac{dR_2}{dt}=\iiint_V \bigtriangledown\left(k\bigtriangledown T\right)dV \end{aligned} \tag{2.5}$$

$$\begin{aligned} &Z\delta\theta\left(\frac{\rho_\ell c}{2}(R_O^2-R_1^2)\frac{dT_\ell}{dt}+\rho_\ell c(T_1-T_\ell)R_1\frac{dR_1}{dt}\right)\\ &=\iiint_V \bigtriangledown\left(k\bigtriangledown T\right)dV \end{aligned} \tag{2.6}$$

$$\begin{aligned} &Z\delta\theta\left(\frac{\rho_s c}{2}(R_2^2-R_R^2)\frac{dT_s}{dt}+\rho_s c(T_s-T_2)R_1\frac{dR_2}{dt}\right)\\ &=\iiint_V \bigtriangledown\left(k\bigtriangledown T\right)dV \end{aligned} \tag{2.7}$$

The right hand side of Eqn. (2.4) is simplified by assuming that the heat transfer through the steel is dominated by the steel-roll interface and, thus, occurs only in the radial direction. Divergence theorem is then applied and the resulting simplification, in its most general form, is

$$\left(\frac{1}{r}\frac{\partial}{\partial r}\left(kZ\delta\theta r\frac{\partial T}{\partial r}\right)\right)_i+\left(\frac{1}{r}\frac{\partial}{\partial r}\left(kZ\delta\theta r\frac{\partial T}{\partial r}\right)\right)_o, \tag{2.8}$$

where k is the effective thermal conductivity of each phase of steel and the subscripts i and o denote the inner and outer radial boundaries of each phase.

Equation (2.8) is further simplified by assuming the process reaches a quasi-steady state and by applying a thermal impedance model similar to Fig. 2.4. The resulting simplification is

$$\frac{k_p Z\delta\theta(T_i-T_p)}{\ln\left(\frac{R_p}{R_i}\right)}+\frac{k_p Z\delta\theta(T_o-T_p)}{\ln\left(\frac{R_o}{R_p}\right)}\ , \tag{2.9}$$

where the subscript $p\in\{\ell,m,s\}$ denotes parameters that are specific to the phase of steel being modeled.

Finally, the boundary between the steel and the roll is modeled by adding a contact resistance between T_R, the temperature of the steel at the roll, and T_{Roll}, as shown in Fig. 2.4. Then as the roll temperature changes (see Section 2.4), so does

the inner boundary temperature of the steel pool model. In reality, the heat transfer between the steel pool and the roll depends on the roll topography and the microscopic processes that govern the solidification behavior, such as nucleation [40, 60]. In this model, I simplify these dynamics into a single heat transfer coefficient, h_o.

Continuity Equation.

The rate of energy crossing the liquidus isotherm should be the same for both the liquid and mushy regions. Likewise, the rate of energy crossing the solidus isotherm should be the same for both mushy and solid regions. I apply the following two continuity equations to ensure that these constraints are enforced.

$$0 = k_\ell \frac{T_\ell - T_1}{\ln\left(\frac{R_\ell}{R_1}\right)} - k_m \frac{T_1 - T_m}{\ln\left(\frac{R_1}{R_m}\right)} \tag{2.10}$$

$$0 = k_s \frac{T_2 - T_s}{\ln\left(\frac{R_1}{R_s}\right)} - k_m \frac{T_m - T_2}{\ln\left(\frac{R_m}{R_2}\right)} \tag{2.11}$$

Building upon our lumped parameter assumption, in both Eqns. (2.10) and (2.11), R_ℓ is the radius corresponding to the center of the liquid phase in a single slice (control volume) and is calculated by $R_\ell = \sqrt{0.5(R_1^2 + R_O^2)}$. Similarly, R_m is the center of the mushy phase and R_s is the center of the solid phase, determined by $R_m = \sqrt{0.5(R_1^2 + R_2^2)}$ and $R_s = \sqrt{0.5(R_2^2 + R_R^2)}$, respectively. Equations (2.10) and

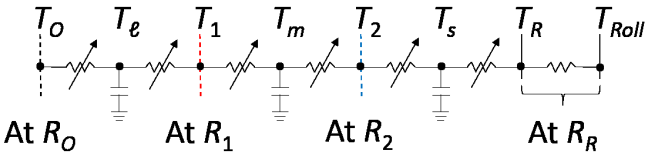


Figure 2.4. The thermal impedance model used to characterize the steel pool thermal dynamics.

(2.11) are true for all time. Thus, the time derivative of both equations is also equal to zero.

State Matrix Equation.

The combination of the three energy balances and the two time derivatives of the continuity equations yields a system of equations that describes the interactions between the state variables, R_1, R_2, T_s, T_m, and T_ℓ. The five state equations can then be written in a matrix form similar to Eqn. (2.12). For example, when all states are actively being modeled (see Section 2.3.2) the state equations are written as Eqn. (2.13) with the nonzero terms listed in Table 2.2. This matrix equation representation applies to *each* slice, and the number of slices can fluctuate based on the engineer's needs. The total number of states needed to describe the solidification dynamics is then five times the number of slices used.

$$M \begin{bmatrix} \dot{R}_1 \\ \dot{R}_2 \\ \dot{T}_s \\ \dot{T}_m \\ \dot{T}_\ell \end{bmatrix} = A \tag{2.12}$$

$$\begin{bmatrix} 0 & M_{12} & M_{13} & 0 & 0 \\ M_{21} & M_{22} & 0 & M_{24} & 0 \\ M_{31} & 0 & 0 & 0 & M_{35} \\ M_{41} & M_{42} & 0 & M_{44} & M_{45} \\ M_{51} & M_{52} & M_{53} & M_{54} & 0 \end{bmatrix} \begin{bmatrix} \dot{R}_1 \\ \dot{R}_2 \\ \dot{T}_s \\ \dot{T}_m \\ \dot{T}_\ell \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ 0 \\ 0 \end{bmatrix} \tag{2.13}$$

2.3.2 Solidification Stages

The state vector in Eqn. (2.12) can be used to represent the dynamics of every slice. Therefore, switching between solidification stages requires selecting the appropriate

Table 2.2. The nonzero values of the state matrix equation (2.13)

Term	**Value**
M_{12}	$\rho_s c(T_s - T_2)R_2$
M_{13}	$\frac{\rho_s c}{2}(R_2^2 - R_R^2)$
M_{21}	$\rho_m c_m(T_m - T_1)R_1$
M_{22}	$\rho_m c_m(T_2 - T_m)R_2$
M_{24}	$\frac{\rho_m c_m}{2}(R_1^2 - R_2^2)$
M_{31}	$\rho_\ell c(T_1 - T_\ell)R_1$
M_{35}	$\frac{\rho_\ell c}{2}(R_O^2 - R_1^2)$
M_{41}	$\frac{k_m(T_1-T_m)\left(\frac{1}{R_m}-\frac{0.5R_1^2}{R_m^3}\right)R_m}{\ln\left(\frac{R_1}{R_m}\right)^2 R_1} - \frac{k_\ell(T_\ell-T_1)\left(\frac{0.5}{R_\ell}-\frac{R_\ell}{R_1^2}\right)R_1}{\ln\left(\frac{R_\ell}{R_1}\right)^2 R_\ell}$
M_{42}	$\frac{.5k_m(T_1-T_m)R_2}{\ln\left(\frac{R_1}{R_m}\right)^2 R_m^2}$
M_{44}	$\frac{k_m}{\ln\left(\frac{R_1}{R_m}\right)}$
M_{45}	$\frac{k_\ell}{\ln\left(\frac{R_\ell}{R_1}\right)}$
M_{51}	$\frac{0.5k_m(T_m-T_2)R_1}{\ln\left(\frac{R_m}{R_2}\right)^2 R_m^2}$
M_{52}	$\frac{k_s(T_2-T_s)\left(\frac{1}{R_s}-\frac{0.5R_2^2}{R_s^3}\right)R_s}{\ln\left(\frac{R_s}{R_s}\right)^2 R_2} - \frac{k_m(T_m-T_2)\left(\frac{0.5}{R_m}-\frac{R_m}{R_2^2}\right)R_2}{\ln\left(\frac{R_m}{R_2}\right)^2 R_m}$
M_{53}	$\frac{k_s}{\ln\left(\frac{R_2}{R_s}\right)}$
M_{54}	$\frac{k_m}{\ln\left(\frac{R_m}{R_2}\right)}$
A_1	$\frac{k_s(T_2-T_s)}{\ln\left(\frac{R_2}{R_s}\right)} - \frac{T_s-T_{Roll}}{\frac{\ln(R_s/R_R)}{k_s}+\frac{1}{h_o R_R}}$
A_2	$\frac{k_m(T_1-T_m)}{\ln\left(\frac{R_1}{R_m}\right)} - \frac{k_m(T_m-T_2)}{\ln\left(\frac{R_m}{R_2}\right)}$
A_3	$\frac{k_\ell(T_O-T_\ell)}{\ln\left(\frac{R_O}{R_\ell}\right)} - \frac{k_\ell(T_\ell-T_1)}{\ln\left(\frac{R_\ell}{R_1}\right)}$

equation set to be solved for the next time step. The solidification dynamics of a given slice can be divided into 6 possible stages:

1. only liquid steel;

2. liquid and mushy steel;

3. liquid, mushy, and solid steel;

4. mushy and solid steel;

5. only mushy steel; and

6. only solid steel.

Each stage has its own dynamic equations. Depending on which phases of steel are represented in each stage of the solidification process, some of the states may become inactive. In this case, I hold them at a constant value.

Stage 1.

When liquid steel is the only phase of steel within the control volume, as it is when the steel initially contacts the roll, the only state that is modeled is the temperature of the liquid steel, T_ℓ. All other states are held at a constant value. The mushy and solid temperatures are set to the liquidus and solidus temperatures, respectively, because those are the temperatures that the states will have when they initially form. Similarly, the liquidus and solidus isotherm locations are set to the roll radius, because that is where the isotherms will initially form. In other words, Iset $T_m = T_1,\ T_s = T_2,\ R_1 = R_R,$ and $R_2 = R_R$. The stage 1 dynamics are then described by Eqn. (2.14).

$$\begin{bmatrix} 0 & 0 & 0 & 0 & M_{15} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{R}_1 \\ \dot{R}_2 \\ \dot{T}_s \\ \dot{T}_m \\ \dot{T}_\ell \end{bmatrix} = \begin{bmatrix} A_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.14}$$

$$M_{15} = \frac{\rho_\ell c}{2}(R_O^2 - R_R^2)$$

$$A_1 = \frac{k_\ell (T_O - T_\ell)}{\ln\left(\frac{R_O}{R_\ell}\right)} - \frac{(T_\ell - T_{Roll})}{\frac{\ln\left(\frac{R_\ell}{R_R}\right)}{k_\ell} + \frac{1}{h_o R_R}}$$

Stage 2.

Stage 2 occurs when the mushy shell starts to grow on the surface of the roll but the solid shell has not yet grown. In this stage, the active states are $T_\ell,\ T_m,$ and R_1. The values of the solidus isotherm and solid steel temperature are held constant at $R_2 = R_R$ and $T_s = T_2$, respectively, as in Stage 1. The dynamics of Stage 2 are described by Eqn. (2.15).

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ M_{21} & 0 & 0 & M_{24} & 0 \\ M_{31} & 0 & 0 & 0 & M_{35} \\ M_{41} & 0 & 0 & M_{44} & M_{45} \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{R}_1 \\ \dot{R}_2 \\ \dot{T}_s \\ \dot{T}_m \\ \dot{T}_\ell \end{bmatrix} = \begin{bmatrix} 0 \\ A_2 \\ A_3 \\ 0 \\ 0 \end{bmatrix} \tag{2.15}$$

$$M_{24} = \frac{\rho_m c_m}{2}(R_1^2 - R_R^2)$$

$$A_2 = \frac{k_m(T_1 - T_m)}{\ln\left(\frac{R_1}{R_m}\right)} - \frac{(T_m - T_{Roll})}{\frac{\ln\left(\frac{R_m}{R_R}\right)}{k_m} + \frac{1}{h_o R_R}}$$

All other terms are as defined in Table 2.2

Stage 3.

Stage 3 is the stage when both the mushy and solid shells are growing. All state are active. The dynamics of this stage are described by Eqn. (2.13) with the nonzero terms listed in Table 2.2.

Stage 4.

Once the mushy shell intersects the centerline of the process, the liquid phase of steel is no longer present in the control volume. Icall this Stage 4 of the solidification process. In this stage, the active states are $T_m,\ T_s,$ and R_2. The values of T_ℓ and R_1 are held constant at $T_\ell = T_1$ and $R_1 = R_O$. The dynamics of this stage are described by Eqn. (2.16).

$$\begin{bmatrix} 0 & M_{12} & M_{13} & 0 & 0 \\ 0 & M_{22} & 0 & M_{24} & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & M_{52} & M_{53} & M_{54} & 0 \end{bmatrix} \begin{bmatrix} \dot{R}_1 \\ \dot{R}_2 \\ \dot{T}_s \\ \dot{T}_m \\ \dot{T}_\ell \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.16}$$

$$M_{24} = \frac{\rho_m c_m}{2}(R_O^2 - R_2^2)$$

$$A_2 = \frac{k_m(T_O - T_m)}{\ln\left(\frac{R_O}{R_m}\right)} - \frac{k_m(T_m - T_2)}{\ln\left(\frac{R_m}{R_2}\right)}$$

All other terms are as defined in Table 2.2

Stage 5.

Stage 5 occurs when the mushy shell grows so fast that the liquidus isotherm intersects the outer boundary of the slice, but the solid steel has not yet formed. It can also occur if the steel is in Stage 4 and the solid shell remelts. In this case, the only active state is T_m and all other states are held constant at $T_\ell = T_1$, $R_1 = R_O$, $T_s = T_2$, and $R_2 = R_R$. The dynamics for this stage are described by Eqn. (2.17).

$$\begin{bmatrix} 0 & 0 & 0 & M_{14} & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{R}_1 \\ \dot{R}_2 \\ \dot{T}_s \\ \dot{T}_m \\ \dot{T}_\ell \end{bmatrix} = \begin{bmatrix} A_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.17}$$

$$M_{14} = \frac{\rho_m c_m}{2}(R_O^2 - R_R^2)$$

$$A_1 = \frac{k_m(T_O - T_s)}{\ln\left(\frac{R_O}{R_m}\right)} - \frac{T_s - T_{Roll}}{\frac{\ln\left(\frac{R_m}{R_R}\right)}{k_s} + \frac{1}{h_o R_R}}$$

Stage 6.

The final stage, stage 6, occurs if the solid shell grows sufficiently such that the solid kiss point is above the nip. In this case, no mushy nor liquid steel will be present in the slice and only the solid steel temperature state, T_s, is active. The other states

are held constant at $R_1 = R_2 = R_O$, $T_\ell = T_1$, and $T_m = T_2$. The dynamics of this stage are described by Eqn. (2.18).

$$\begin{bmatrix} 0 & 0 & M_{13} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{R}_1 \\ \dot{R}_2 \\ \dot{T}_s \\ \dot{T}_m \\ \dot{T}_\ell \end{bmatrix} = \begin{bmatrix} A_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.18}$$

$$M_{13} = \frac{\rho_s c}{2}(R_O^2 - R_R^2)$$

$$A_1 = \frac{k_s(T_O - T_s)}{\ln\left(\frac{R_O}{R_s}\right)} - \frac{T_s - T_{Roll}}{\frac{\ln\left(\frac{R_s}{R_R}\right)}{k_s} + \frac{1}{h_o R_R}}$$

2.3.3 Switching Criteria

In Stages 2, 3, and 4, a new phase of steel may begin forming on the surface of the roll. When this occurs, the M matrix is singular because either R_1 or R_2 is initially equivalent to R_R. This singularity results in a numerical instability that can produce large errors in the simulation when I attempt to invert M. I compensate for this by introducing a switching parameter, ε, and simulating a reduced-state model until the isotherm has grown a distance ε away from the surface of the roll. At that point, I switch to the full simulation of the solidification stage as described in Section 2.3.2.

2.4 Roll Dynamics

To obtain an accurate representation of how the rotational speed impacts the solidification model, I have to understand how Ω affects the boundary temperature between the the steel and the roll. This section explores the thermodynamic model of the roll, which provides a characterization of the roll surface temperature that is used in the steel pool model.

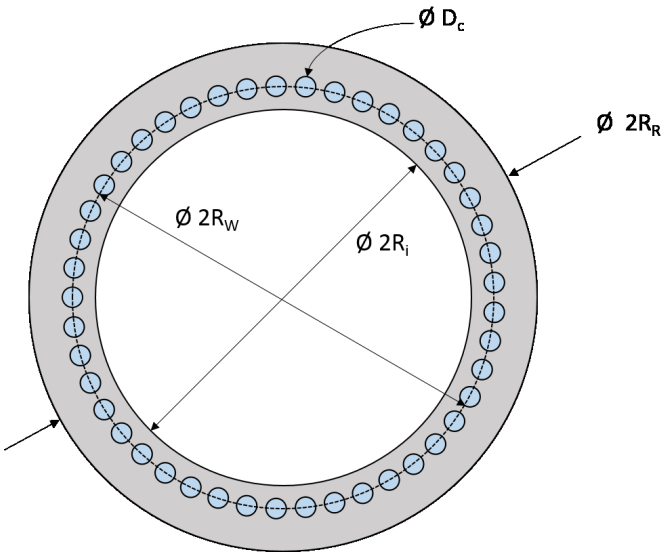


Figure 2.5. A schematic of the roll composed of a copper sleeve encasing a set of water channels.

A common roll assembly consists of a copper sleeve that encloses a set of water channels, as shown in Fig. 2.5. The water channels act as a heat sink for the process and help regulate the temperature of the copper. As the roll rotates, its surface temperature will increase as the roll passes through the steel pool and then decrease once the roll rotates past the nip. During periodic steady-state operation, I assume that the surface temperature at the steel meniscus will be periodic with the same period as one roll revolution. This means that, in the period of one roll revolution, the heat absorbed from the steel pool is transferred from the roll to either the water flowing through the cooling channels or the air surrounding the roll. Then, as the water flows out of the roll, the energy stored in the water is removed from the roll assembly entirely.

2.4.1 Thermal Model

The dynamics of the roll assembly are modeled by dividing the roll into equal volume radial slices such that each water channel coincides with one slice. I assume that the heat transfer into the water is much higher than the conduction into the interior portion of the copper sleeve. Therefore I modify our model of the water channel by extending it so that it encompasses the entire inner boundary as shown in Fig. 2.6. With this architecture, I extend our radial resistance analogy into the roll model using the structure shown in Fig. 2.7 and by applying a lumped parameter assumption to the temperatures of both the copper and the water contained within each slice. Then the dynamics for each slice are written as two differential equations with two states – T_{Cu} and T_W (see Eqn. (2.19) and Eqn. (2.20)).

$$\frac{\partial T_{Cu}}{\partial t} = \frac{2\left(\frac{T_R - T_{Cu}}{\frac{1}{k_{Cu}} ln\left(\frac{R_R}{R_{Cu}}\right) + \frac{1}{h_o R_R}} + \frac{T_W - T_{Cu}}{\frac{1}{k_{Cu}} ln\left(\frac{R_{Cu}}{R_{@W}}\right) + \frac{1}{h_i R_{@W}}}\right)}{\rho_{Cu} c_{Cu} (R_R^2 - R_{@W}^2)} \tag{2.19}$$

$$\frac{\partial T_W}{\partial t} = \frac{2\left(\frac{T_{Cu} - T_W}{\frac{1}{k_{Cu}} ln\left(\frac{R_{Cu}}{R_{@W}}\right) + \frac{1}{h_i R_{@W}}}\right)}{\rho_W c_W (R_{@W}^2 - R_W^2)} \tag{2.20}$$

In Eqn. (2.19), T_R represents the outer boundary temperature, h_o is the heat transfer coefficient between the surface of the roll and the outer boundary, and h_i is the convection coefficient of the water. The value of h_i is determined using the Dittus-Boelter equation for calculating the Nusselt number,

$$h_i = \frac{Nu_W k_W}{D_C} = \frac{k_W}{D_C} 0.0233 \left(\frac{4\dot{V}}{\nu_W \pi D_C^2}\right)^{0.8} Pr_W^{0.4} \ , \tag{2.21}$$

where $\dot{V}$, Pr_W, ν_W, and k_W are, respectively, the volumetric flow rate, Prandtl number, kinematic viscosity, and conduction coefficient of the water, and D_C is the water channel diameter.

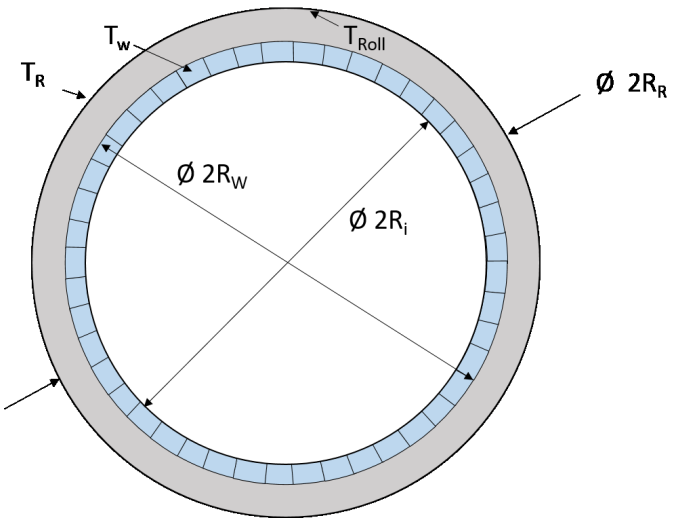


Figure 2.6. A schematic depicting how the water channels are extended so as to encompass the entire inner radius of each roll slice.

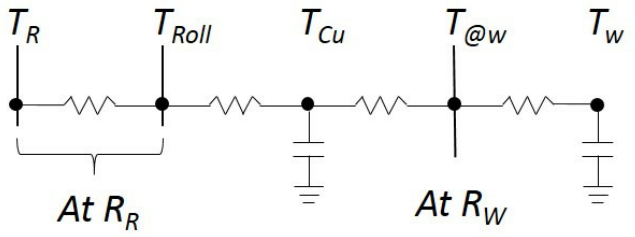


Figure 2.7. The thermal resistance network for the roll.

2.4.2 Outer Boundary Condition

As the roll rotates, the outer boundary condition of a slice of the roll changes when the slice transitions between contacting steel and contacting air. During the time that a slice is in contact with the steel, both T_{Roll} and T_{Cu} increase. When the slice stops contacting the steel, T_{Roll} and T_{Cu} begin to cool as the heat continues to be transferred into the water channels. When the slice is contacting the steel pool, I introduce the variable T_R as the surface temperature of the steel. Then, when the

slice is contacting air, I assume that the outer boundary is *almost* adiabatic. This approach is similar to that taken in [51] and [52], in which an effective heat transfer coefficient of $50W/m^2K$ between the rolls and their surroundings is used. Comparing this value to the value that [51] and [52] use for the effective heat transfer coefficient between the rolls and the water, $10000W/m^2K$, results in approximately 0.5% of the heat in the rolls being transferred to the surroundings. Therefore, I assume that the heat transfer from the rolls to the surrounding air is negligible compared to the heat transferred into the water channels. The reason I cannot assume the boundary is adiabatic is that $h_o = 0$ results in a numerical instability in the lumped impedance model. Thus, I reduce h_o to a number that is close to zero. In other words, I condition h_o such that the copper sleeve almost exclusively cools by transferring its heat into the water channels. This form of analysis provides us with a way to compare the heat transfer from our model with temperature measurements taken from the exiting water flows.

2.5 Simulation/Numerical Implementation

Simulating the steel pool and roll models together is challenging due to the difference in how each model is discretized (which is a function of their individual geometry). In this section I discuss the numerical implementation of the integrated model as well as parameter selection for the steel pool model.

2.5.1 Simulation Setup

One of the first parameters that must be selected is the number of slices to use for discretization of the roll and steel pool models. In this manuscript, I consider water channels arranged in a circular pattern with 8 degrees of separation between each channel. Therefore, I choose an 8 degree wide angular slice structure for both the steel pool and roll models. This enables alignment of the steel pool slices with the roll slices so that the heat transfer between the steel pool and roll models is only affected

by a single slice in both directions. This eliminates the complexity that would be introduced by having multiple states acting on the boundary between the models. The one-to-one configuration also allows for a better estimate of the inner boundary conditions in the steel pool model.

The second parameter to select is ε, the switching parameter that I introduced in Section 2.3.3. If the value is too small, the simulation becomes unstable due to the numerical singularity that occurs when a new phase of steel forms on the surface of the roll. Selecting a large value, conversely, further propagates the error that is introduced by our switching criteria simplification. I found that a value of $50\mu m$, or 2.5% of the nominal gap distance, x_G, results in numerical stability while keeping the initial transient error confined to a small percentage of the overall shell profile.

The third parameter to select is the heat transfer coefficient between the roll and the steel pool, h_O. In practice, this value is difficult to determine because it depends on the the roll topology and the microscopic processes that govern the solidification behavior [40,60]. Instead, I treat h_O as a tuning parameter, where the value is chosen such that the mushy fraction at the nip is between $1 - 10\%$. This range of possible mushy fraction values is based on the information provided in [61]. At the nip, the mushy fraction, as a percentage of the gap distance, can be calculated from the shell profile as

$$f_m = 100(x_G/2 - R_2) \ . \tag{2.22}$$

2.5.2 Model Integration

Figure 2.8 provides a flowchart of the solution procedure. There are two design decisions that must be made regarding how the roll rotation is introduced into the steel pool and roll models. Recall that the steel pool model is composed of stationary slices (control volumes) with angular thickness $\delta\theta$, and the roll model is composed of angular slices defined by the water channel configuration, which rotate with the roll. Rotating the models at every time step would thus disrupt alignment of the

slice geometry that I chose earlier. Instead, I simulate the model for $\delta\theta/\Omega$ seconds in one alignment before rotating both the steel pool and the roll models by an angular distance of $\delta\theta$. This approach allows us to maintain the alignment between the two models while also simulating the effects that the rotational speed has on the solidification dynamics.

When the models rotate, I must update their boundary conditions. The roll model is updated according to the procedure described in Section 2.4.2. The steel pool, on other hand, is updated based on the geometry of the new slice location. Recall that the steel pool can be divided into regions based on the phase of the steel, as shown in Fig. 2.2. Within a given slice, the inner phases of steel, in a radial sense from the roll's surface, are assumed to be identical before and after the rotation occurs. The outer phase, however, must be updated to reflect the change in the slice volume. Recall from Fig. 2.3 that a slice's outer boundary either intersects with the centerline of the process or the surface of the steel pool. Along the top slices which intersect the surface of the steel pool, the outer temperature is set to the inlet steel temperature; the lumped parameter temperature state is determined using the continuity equation between the outer phase of steel and the adjacent (radially, in the direction of the roll) phase of steel. When the outer boundary does intersect the centerline, I use a projection method to determine the outer boundary temperature. I calculate the temperature of the steel, in the slice before the rotation, at the location that would correspond to the outer boundary after the rotation. This value becomes the outer boundary temperature after the rotation and the temperature state is updated according to the same continuity assumption used in the top slices.

2.6 Simulation Results

Validating the proposed model requires an unorthodox approach due to the high operating temperatures of the process ($> 1000°$C) and the limitations this imposes on real-time measurements of the thermodynamic state of the steel. This section

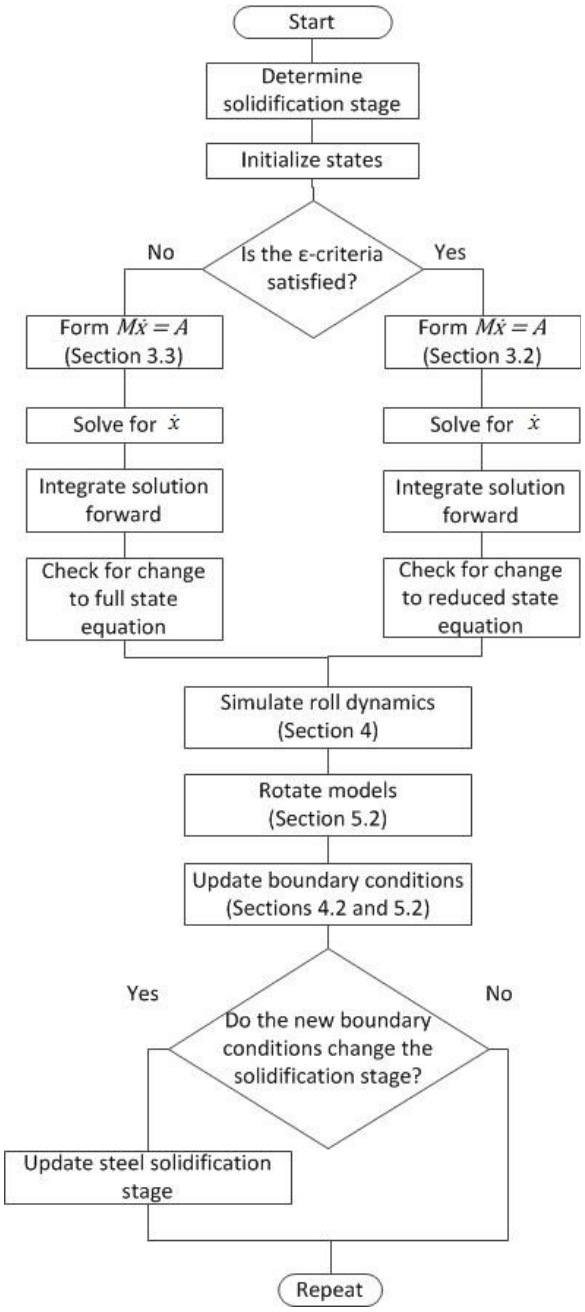


Figure 2.8. A flowchart describing the solution procedure.

discusses our approach for model validation. I also illustrate, through two case studies, how the model responds to different input commands for the rotational speed and the roll gap distance.

2.6.1 Model Validation

While the model is defined by many fewer dynamic states than some previously developed models, it still requires more states than I can individually validate. Furthermore, complete validation of the solidification model is impractical due to 1) the high temperatures of molten steel ($> 1000°C$), which can melt many sensors and limit their use, and 2) spatial limitations, such as the size and position of the rolls relative to the gap distance, which prevent the use of imaging techniques to measure the temperature and shell profiles within the steel pool. These limitations hinder one's ability to measure both the isotherm locations and the temperature profiles that emerge during a cast. However, I can use industrial data describing the total amount of energy transferred to the water, as well as the mushy fraction calculated from the manufactured strip [61], as surrogate measurements to verify that the heat transfer dynamics captured by the model are consistent with an actual twin-roll casting process.

To compare the model with the experimental data, I consider a roll model with 45 water channels. This results in a roll model that has 45 angular slices, each with an angular thickness of 8 degrees. I then divide our solidification model into 6 slices, each with an angular thickness of 8 degrees, for the reasons described in Sec. 2.5. I also use the set of material properties summarized in Table 2.3, which are a reasonable approximation of the properties of the steel associated with the cast data that is used for comparison [62]. The simulation parameters are listed in Table 2.4.

Once the meniscus conditions reach periodic steady state, the shell profile stabilizes as shown in Fig. 2.9. The liquid steel cools and starts to transition into mushy steel 0.03 seconds after it comes into contact with the roll. The solid shell then gradually grows as more energy is extracted from the mushy steel. As a given slice of

Table 2.3. Steel material properties

Property	**Value**
c	750 $J/kg°C$
L	272 kJ/kg
k_ℓ	4000 W/mK
k_m	3800 W/mK
k_s	77 W/mK
T_1	1524° C
T_2	1502° C
ρ_ℓ	7200 kg/m^3
ρ_s	7860 kg/m^3

Table 2.4. Simulation parameters

Parameter	**Value**
h_o	20000 W/m^2K
T_{in}	1600° C
$T_{W,in}$	30° C
$\dot{V}$	0.0835 m^3/s
x_G	2 mm
θ_{Lev}	48 deg
Ω	4.47 rad/sec

the steel pool model rotates toward the nip, the growth rate of both shells begins to slow due to less heat being transferred into roll as the roll heats up. The mushy shell intersects the centerline at a distance of 45.2μm above the nip. This indicates that, because the solid shell does not intersect the centerline, there is a region of mushy steel at the nip when the strip exits the casting rolls, as shown in Fig. 2.10.

In the case shown in Fig. 2.10, the mushy fraction at the nip, as defined by Eqn. (2.22), is 9%, which is similar to the measurements obtained by Nucor Corp. In [61], it is claimed that the mushy material between the metal shells of the strip cast can be controlled to between $10\mu m$ and $200\mu m$ for a strip with a thickness between $0.6mm$ and $2.4mm$. That means that for our chosen gap distance ($x_G = 2mm$), there can be up to 10% mushy steel present at the nip.

The results are also similar to the work described in [50]. Li et al. showed that for an inlet (molten) steel temperature of $1600^\circ\ C$ and a casting speed of $\approx 1.11m^{-1}s$, their model estimated that the solidification point of the steel would be below the nip. This implies that, at the nip, there would be a region of mushy steel as I show in Figure 2.10. Li et al. considered stainless steel instead of low-carbon steel; nonetheless the properties of the steel strip in the nip region as estimated by our model are consistent with published literature.

In the absence of experimental measurements of the solidification dynamics, changes in the cooling water temperature between inlet and exit are a reasonable proxy for the aggregate heat removal of the process during periodic steady state operation. Measurements from the Nucor facility show that the water temperature increases by approximately 3.6°C per pass through the roll. In our simulation, T_W increases by 3.66°C in the period of a roll revolution. Because the water is the primary source of cooling in the system, the similarity between the predicted and actual increase in the water temperature indicates that the model is accurately estimating the amount of heat being extracted from the steel at any given time.

2.6.2 Case Studies

The primary motivation for this model is to predict how different values of the control inputs—Ω and x_G—affect the shell profile. To illustrate this, I propose two case studies. The first case study examines how a change in the commanded rotational roll speed, Ω, affects the shell profile. The second shows that a change in the

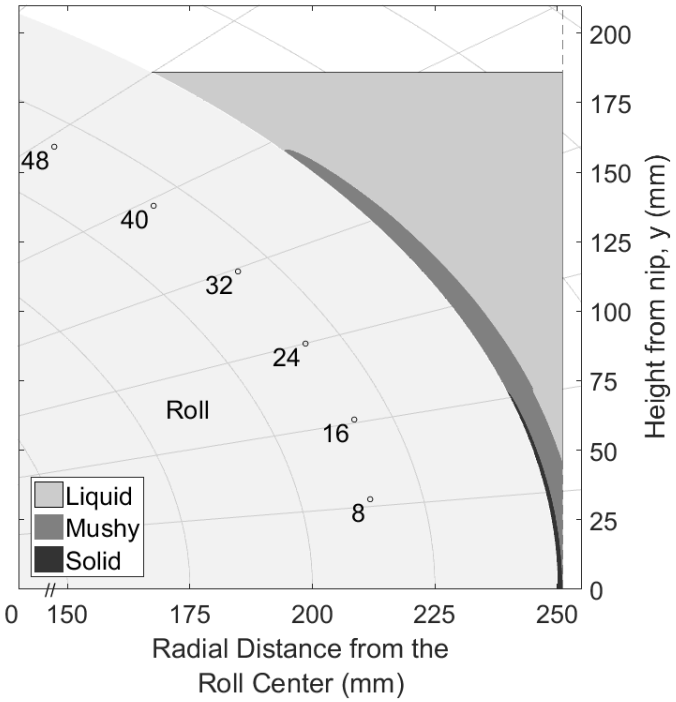


Figure 2.9. The phases of steel on the roll surface.

commanded gap distance, x_G, results in a change in mushy fraction, f_m, and kiss point, y_k. The analysis I provide in each case study focuses on the periodic steady state response of the process in response to a change in the commanded values for Ω and x_G. I focus on this type of analysis because, as stated in Section 2.6.1, the steel solidification begins after only 0.03 seconds whereas the period of one roll revolution is approximately 1.4 seconds in length. Based upon this separation in time scales, I assume that neither Ω nor x_G can be actuated sufficiently fast enough to affect the transient response of the solidification dynamics.

As Fig. 2.11 shows, when the rotational speed, Ω, is changed from 4.47 to 3.49 radians per second, the solid shell (shown by the thin dashed line) intersects the centerline. This corresponds to a mushy fraction of 0, which is 9% lower than when $\Omega = 4.47$ radians per second (shown by the thick dashed line). The mushy shell when

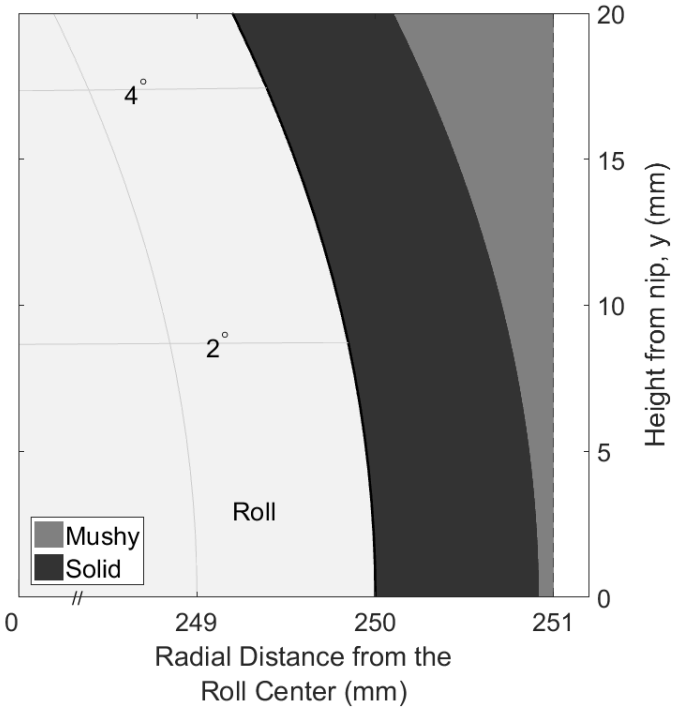


Figure 2.10. A zoomed in view of the nip in Fig. 2.9 shows that there is mushy steel present at the nip.

$\Omega = 3.49$ radians per second (shown by the thin solid line) also intersects the centerline much earlier than when $\Omega = 4.47$ radians per second (shown by the thick solid line). This results in a kiss point, y_k, that is $450\mu m$ higher. *Both changes affect the size and composition of the compression region, which affects the force requirements for consistent casting* [55].

The existence of the solid shell kiss point above the nip is a common assumption that is made in many of the published models on twin-roll casting [41–44], but that assumption is violated when the process is operating at the speeds currently used in industry (> 4rad/s). Nevertheless, when our model is simulated at speeds of 0.7rad/s as is the case in Santos et al., the solid shell kiss point moves farther above the nip, resulting in no mushy steel in the nip region, as shown in Fig. 2.12. In this way, the proposed model is consistent with previously published models but *extends the state-of-the-art by relaxing the aforementioned assumption.*

In the second case study I show that an increase in x_G results in a decrease in the shell thicknesses relative to the overall gap distance. Figure 2.13 shows the differences between the case with $x_G = 2mm$ and the case with $x_G = 2.2mm$. The shell profiles are identical, but the location of these shells within the process changes. The increase in the gap distance results in a larger mushy fraction at the nip (i.e. $y = 0$ mm) and a smaller nip region.

2.7 Control Implementation Case Study: Twin-Roll Strip Casting

In this section I apply the combined delay estimation and ILC algorithm from Sec. 3.1 to a twin roll strip casting process at Nucor Corporation's Castrip plant in Crawfordsville, Indiana. The objective of this section is to demonstrate how the combined delay estimation and ILC algorithm proposed in Chapter 3 can be used to reduce the wedge produced in the twin roll strip casting process. To accomplish this objective, I define an iteration of the process as equal to one revolution of the casting rolls and then use data from Nucor to identify the periodic dynamics of the system.

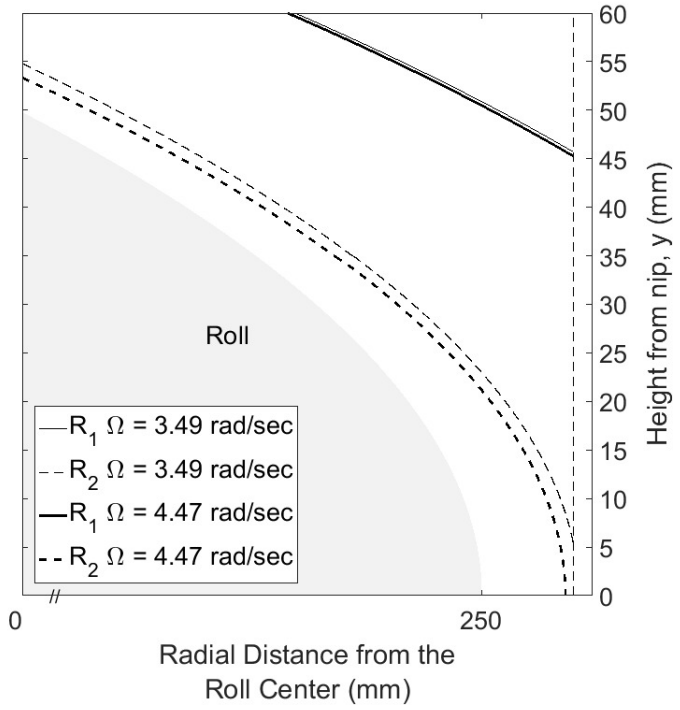


Figure 2.11. The shell profile at different rotational speeds.

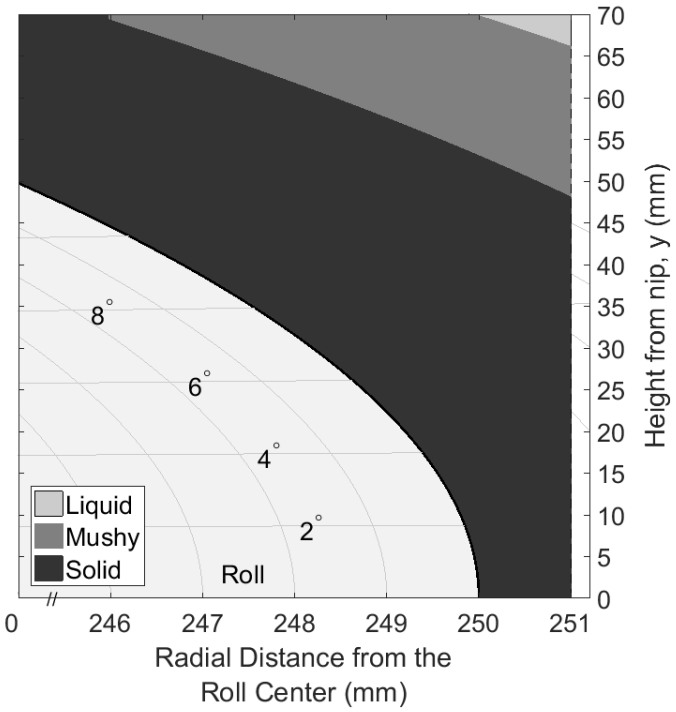
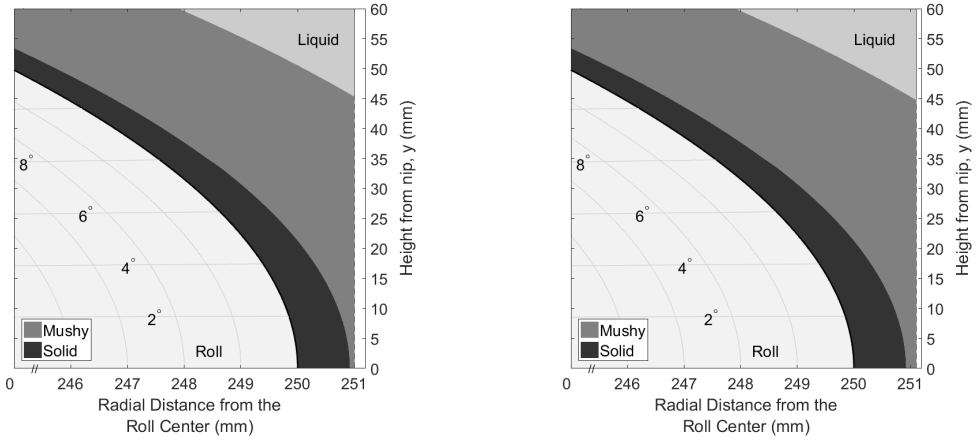


Figure 2.12. The shell profile produced at a casting speed of $\Omega = 0.7$ rad/s has a *solid* kiss point that is approximately 48 μm above the nip.



(a) $x_G = 2$mm (b) $x_G = 2.2$mm

Figure 2.13. Changing x_G results in a change in the mushy fraction at the nip.

As shown in Fig. 2.14, after the strip has formed, it passes into an environmentally controlled box, called a hot box, where it continues to passively cool before undergoing a final round of compression in a hot roll stand. Due to the high temperature of the metal exiting the casting rolls at point A, as well as the physical limitations of accurately measuring the thickness of the strip near the nip, the strip thickness is not measured until the strip is on the tables rolls leading into the hot rolling stand. The measurement delay is thus the amount of time that it takes for the strip to move from the actuation point at the nip of the casting rolls, point A, to the measurement location, point C. This time can change during the cast based on the amount of steel that is in the free hanging loop, shown in Fig. 2.14 as the length of strip between points A and B. The depth of this loop is variable and depends on a number of parameters, including the casting roll speed, the hot rolling stand speed, and the grade of steel being cast. Below I describe in detail how I estimate this variable time delay.

2.7.1 Delay Estimation

While the periodic nature of the twin roll strip casting process makes it well suited for learning-based control, the periodicity also complicates the use of a data-based approach, such as correlation methods [34], to estimate the delay. To overcome this difficulty, I divide the estimation of the delay T_D into two estimation problems, based on the definition of the time delay that I introduced in (3.4). The first problem is to estimate the iterative component of the delay, n_k. Once I establish a value for n_k, I estimate the residual component of the delay, τ.

To estimate the delay, I relate it to the length of the strip between the nip of the casting rolls and the measurement location. I can express the length of the strip as

$$L = n_k C_{CR} + \delta L \ , \tag{2.23}$$

where C_{CR} is the circumference of a single casting roll, n_k is the number of complete roll revolutions that occur during the delay, and δL is the remainder of L/C_{CR}.

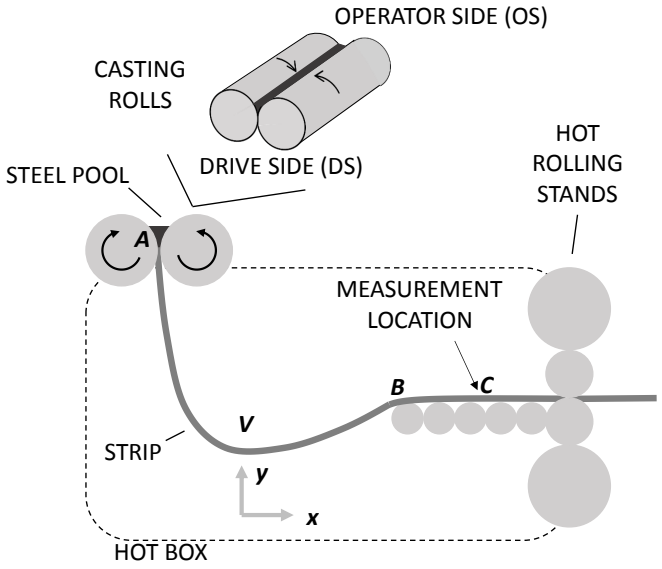


Figure 2.14. The steel strip leaves the casting rolls and enters a hotbox where it passively cools before being compressed on a hot rolling stand. The thickness measurements are obtained at point C as the strip moves along the table rolls.

As shown in Fig. 2.14, when the strip leaves the casting rolls, it initially forms a free-hanging loop in the hot box between point A and point B. At point B, the strip passes onto a set of table rolls that transport it past the measurement location at point C and into the hot rolling stand. From the layout of the hot box, I can determine the distances between A and B: $x_B - x_A =: \bar{x}_{AB}$ and $y_A - y_B =: \bar{y}_{AB}$; and the distance from B to C: $x_C - x_B =: \bar{x}_{BC}$.

I assume that the loop between A and B can be described by a catenary curve [63],

$$y = a \cosh\left(\frac{x}{a}\right) \quad , \tag{2.24}$$

where x and y are defined such that the x-coordinate of the vertex of the curve, x_V, is at $x = 0$. The term $a > 0$ is a parameter of the curve and is related to the material

that forms the curve. This description enables us to write the arc length of the curve as

$$s = a\sinh\left(\frac{|x_B|}{a}\right) + a\sinh\left(\frac{|x_A|}{a}\right) \quad . \tag{2.25}$$

Then the length of the strip can be rewritten as

$$L = s + \bar{x}_{BC} \quad . \tag{2.26}$$

To compute the length, I first need to determine a. This is done by solving the following system of equations:

$$y_A = a\cosh\left(\frac{x_A}{a}\right) \quad , \tag{2.27a}$$

$$y_B = a\cosh\left(\frac{x_B}{a}\right) \quad , \tag{2.27b}$$

$$x_B - x_A = \bar{x}_{AB} \quad , \tag{2.27c}$$

$$y_A - y_B = \bar{y}_{AB} \quad , \tag{2.27d}$$

$$y_A - h_{Loop} = a\cosh(0) = a \quad , \tag{2.27e}$$

where h_{Loop} is the measured loop depth relative to the nip ($h_{Loop} = y_A - y_V$). The value of a is then the solution to

$$\begin{aligned} \bar{x}_{AB} =& a\cosh^{-1}\left(\frac{a + h_{Loop}}{a}\right) \\ &+ a\cosh^{-1}\left(\frac{a + h_{Loop} - \bar{y}_{AB}}{a}\right) \quad . \end{aligned} \tag{2.28}$$

Once the value of a has been determined, the total length of the strip between A and B is calculated using (2.25) and (2.26), and the values of x_A and x_B, which are calculated as part of solving the system of equations (2.27). With the calculated value of L, I can estimate n_k and τ using (2.29) and (2.30), respectively, where T_R is the period of one revolution and L_k is length of strip produced during each revolution of the casting rolls, which I assume to be equivalent to the circumference of the casting rolls, C_{CR}. In Eqn. (2.30), the operator $\mathrm{mod}(L/L_k)$ represents the modulus of L/L_k.

$$\hat{n}_k = \mathrm{floor}(L/L_k) \tag{2.29}$$

$$\hat{\tau} = \text{mod}(L/L_k)T_R \tag{2.30}$$

Remark 1 *In practice, the circumference of the casting roll may not be known exactly after it has thermally expanded due to the high temperatures of the casting process. This can introduce inaccuracies in the time delay estimates. To compensate for the expansion of the casting rolls and other measurement uncertainties, an adaptive parameter can be introduced so that the estimated length of strip produced during each revolution is modified. For example, I can approximate the circumference of the casting roll as* $\hat{C}_{CR} \approx \alpha C_{CR}$ *where* C_{CR} *is the circumference of the casting roll at room temperature and* α *is an adaptive parameter. The approximated circumference of the casting roll circumference can then be used in place of* L_k *in* (2.29) *and* (2.30)*. The modification allows the delay estimate to converge toward the true delay, assuming* α *makes* $\hat{C}_{CR}$ *converge to the true circumference of the casting roll under operating conditions.*

Remark 2 *As previously discussed, correlation-based delay estimation techniques are unreliable for estimating delays in a periodic process that are multiple periods in length. However, by using* (2.29) *as the definition of the iterative component of the delay, the region of potential delays can be reduced to a single period of the process. This enables the use of correlation-based delay estimation techniques to estimate the residual component of the delay,* $\hat{\tau}$*. One example of how this can be accomplished is discussed in [64].*

I validated the time-delay estimation algorithm using a dataset where the *tilt* of one of the casting rolls (the position of one side of the roll minus the position of the other side) undergoes a step sequence and the wedge signal tracks the step changes. During the step sequence, the normalized loop height had a mean value of 0.45 and a variance of 0.03 throughout the duration of the test. This results in a mean strip length estimate of 5970mm with a variance of 60mm. Then, with the rotational

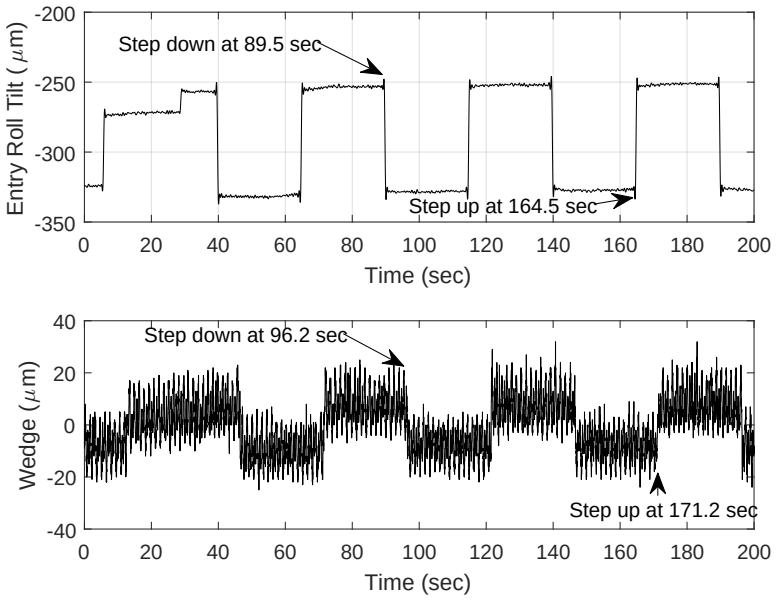


Figure 2.15. The time delay can be measured in post processing by comparing the time at which the steps occur in both the caster roll tilt signal and the wedge measurement.

period being approximately $T_R = 1.47$ seconds, the resulting time delay estimate is 675 samples on average, with a variance of 13 samples.

Furthermore, the estimate can be manually verified offline by measuring the delay between the step sequence in the tilt signal versus the step sequence in the measured wedge signal. As shown in Fig. 2.15, the delay between the tilt signal and the wedge signal is approximately 6.7 seconds which means the estimate of T_D is accurate to within 0.1 seconds.

2.8 Chapter Summary

In this chapter I provided more details about the twin roll strip casting process and how the input dynamics affect the solidification process of the strip. In Chapter 3, I will leverage that understanding to implement an ILC algorithm that adjusts

the tilt of the casting rolls, which affects the gap distance between the casting rolls, to reduce the strip wedge that is formed due to eccentricities that appear during the casting process. The eccentricities appear, in part, due to the nonuniform heat transfer between the casting rolls and the molten steel as the rolls rotate through the steel pool. The nonuniform heat transfer can cause localized thermal expansion in the roll, which leads to a change in the gap distance that the steel is cast through.

Another key takeaway from this chapter is the insight I provided into the uncertainty and aperiodic behavior that is experienced during the casting process. I showed that a change in the rolls' rotational speed or in the gap distance set point affects the mushy fraction of the strip as it exits the nip of the casting rolls. The mushy fraction provides a measure of the amount of steel exiting the nip of the casting rolls that is not completely solidified. A different mushy fraction at the nip of the casting rolls can alter the mechanical dynamics associated with adjusting the gap distance. This means that variations in the solidification dynamics could change the effective damping and stiffness coefficients of the strip being produced. Because there are some uncertainties associated with the solidification parameters, there is also uncertainty in the mechanical parameters. Furthermore, the knowledge that the casting speed can alter the magnitude of the mushy fraction means that when the casting process is operating at a different speed, the nominal effective stiffness and damping of the strip may also be different. These two findings—the parametric uncertainty of the damping and stiffness coefficients and the ability of the casting speed to alter the nominal behavior of the mechanical dynamics—motivate the work in Chapter 4, which develops a robust linear parameter-varying ILC algorithm that accounts for delays within the system.

3. ITERATIVE LEARNING CONTROL FOR TIME DELAY SYSTEMS

This chapter discusses the first two ILC algorithms that I propose in this thesis. This chapter contains information that I published with Brad Rees, George T.-C. Chiu, and Neera Jain in *IEEE Transactions of Control Systems Technology* [65] and at the 2019 ASME Dynamic Systems and Control Conference [66].

3.1 Iterative Learning Control

In this chapter, I consider the following discrete-time plant model for a periodic process:

$$\begin{aligned} x(t+1,k) &= Ax(t,k) + Bu(t,k) \ , \\ y(t,k) &= Cx(t-T_D,k) \\ &= C(zI-A)^{-1}Bu(t-T_D,k) + d(t-T_D) \\ &= G(z)u(t-T_D,k) + d(t-T_D) \end{aligned} \tag{3.1}$$

where t is the time index, k is the iteration index, x is the state of process, u is the input signal, and y is a delayed measurement of the output, which is taken some time T_D after the input signal u is applied to the process. The signal $d(t-T_D)$ is an iteration-invariant exogenous signal that includes disturbances and the effects of the initial condition of x. For the purpose of this paper, the initial condition of x in iteration k is assumed to be equal to the final condition of x in iteration $k-1$, i.e., $x(0,k) = x(T_R, k-1)$, where T_R is the number of samples in each iteration. The parameter $G(z) \triangleq C(zI-A)^{-1}B$ represents the discrete transfer function from $u(t-T_D,k)$ to $y(t,k)$, where z is a forward shift operator in the time domain $zx(t,k) \equiv x(t+1,k)$. The matrices A, B, and C are assumed to be appropriately dimensioned state space matrices.

To analyze the effect of the delay on the ILC algorithm, I consider the widely used ILC control law [9]

$$u(t,k+1) = Q(z)[u(t,k) + L(z)e(t,k)] \ , \tag{3.2}$$

where $u(t,k)$ is the control input signal and $e(t,k)$ is the error signal at time t in iteration k. The linear time-invariant functions $Q(z)$ and $L(z)$ are the Q-filter and learning function, respectively. The error of the process, $e(t,k)$, relative to an iteration-independent desired output for the process $y_d(t)$, is defined as

$$\begin{aligned} e(t,k) &= y_d(t) - y(t,k) \ , \\ &= y_d(t) - G(z)u(t-T_D,k) - d(t-T_D) \ . \end{aligned} \tag{3.3}$$

To compensate for a time delay longer than a single iteration of the process, i.e, $T_D > T_R$ where T_R is the period of one iteration, I construct a model of T_D as follows:

$$T_D(k) = n_k T_R + \tau(k) \ , \tag{3.4}$$

where n_k is the number of iterations that occur during the delay, and $\tau(k)$ is the residual of $T_D(k) - n_k T_R$. This definition allows us to treat the delay as the sum of two delays: a delay of n_k in the iteration domain and a delay of τ in the time domain. With this definition, I can rewrite the error signal as

$$\begin{aligned} e(t,k) &= -G(z)u(t-\tau,k-n_k) + y_d(t) - d(t-\tau) \\ &= -G(z)u(t-\tau,k-n_k) + W(t) \ , \end{aligned} \tag{3.5}$$

where $W(t) \triangleq y_d(t) - d(t-\tau)$ represents the iteration-invariant component of the error that does not depend on the input signal u.

Using (3.4) and (3.5), the control law in (3.2) can be rewritten as

$$\begin{aligned} u(t,k+1) = Q(z)[u(t,k) + L(z)(&- G(z)u(t-\tau,k-n_k) \\ &+ W(t))] \ . \end{aligned} \tag{3.6}$$

The mixed indices of u on the right hand side of (3.6), however, can result in stability and performance problems because the controller modifies $u(t,k+1)$ without

knowledge of how $u(t,k)$ actually affected the process. To address this misalignment, I modify the control law so that the control signal being defined is based on a prior control signal and the error generated by it. In this modification, I shift the error signal forward by an estimate of the delay. To maintain continuity, I shift the left hand side of (3.6) forward by $\bar{n}_k$, which is the smallest positive integer that satisfies $\bar{n}_k T_R > T_D$. I then use our estimate of T_D to align the error signal with $u(t,k)$. This results in the control law given by

$$\begin{aligned} u(t,k+\bar{n}_k+1) =& Q(z)[u(t,k) \\ &+ L(z)(-G(z)u(t+\hat{\tau}-\tau,k+\hat{n}_k-n_k) \\ &+ W(t+\hat{\tau})] \ , \end{aligned} \tag{3.7}$$

where $\hat{\tau}$ and $\hat{n}_k$ are the estimates of the two components of T_D.

By introducing a forward shift operator q in the k-domain $qx(t,k) \equiv x(t,k+1)$ and reusing the z operator from before, (3.7) is rewritten as

$$\begin{aligned} q^{\bar{n}_k+1}u(t,k) =& Q(z)[I - L(z)q^{\hat{n}_k}z^{\hat{\tau}}e(t,k)] \\ =& Q(z)(I - L(z)G(z)q^{\hat{n}_k-n_k}z^{\hat{\tau}-\tau})u(t,k) \\ &+ Q(z)L(z)q^{\hat{n}_k}z^{\hat{\tau}}W(t) \ . \end{aligned} \tag{3.8}$$

I can then state the following two proposition, which are extensions of the theorems presented in [67].

Proposition 3.1.1 *The system is asymptotically stable if and only if*

$$\rho(Q(z)[I - L(z)G(z)q^{\hat{n}_k-n_k}z^{\hat{\tau}-\tau}]) < 1 \ , \tag{3.9}$$

where $\rho(A)$ is the spectral radius of A.

A sufficient condition for stability of the system can be obtained by requiring that $Q(z)[I - L(z)G(z)q^{\hat{n}_k-n_k}z^{\hat{\tau}-\tau}]$ be a contraction mapping [9,67]. For a z-domain system $G(z)$, I define $||G(z)||_\infty = \sup_{\theta\in[-\pi,\pi]} |G(e^{i\theta})|$.

Proposition 3.1.2 *The ILC system defined in* (3.1), (3.8) *is asymptotically stable if*

$$\left\|Q(z)(I - L(z)G(z)q^{\hat{n}_k-n_k}z^{\hat{\tau}-\tau})\right\|_\infty < 1 \ . \tag{3.10}$$

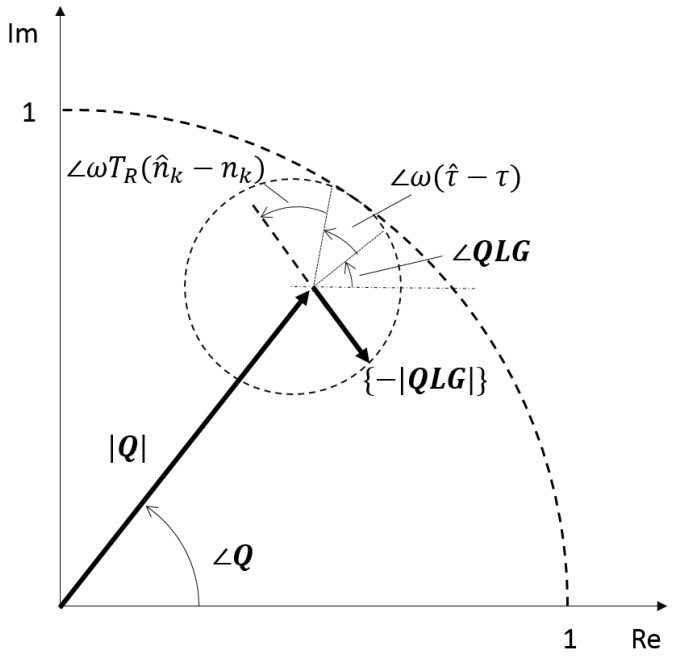


Figure 3.1. For SISO systems, (3.10) can be expressed as the summation of vectors in the frequency domain.

The condition in (3.10) is a sufficient case and is generally more conservative than the necessary and sufficient case in (3.9).

For a single-input single-output (SISO) system, (3.10) can be expressed as a restriction on the magnitude of the sum of two vectors in the frequency domain as shown in Fig. 3.1. The time delay estimation error is equivalent to adding additional phase to a vector of magnitude $-|QLG|$.

Corollary 1 *For a SISO system and a control law* (3.8) *with positive scalar values for* Q *and* L*, i.e.,* $Q(z) = Q > 0$ *and* $L(z) = L > 0$*, if* $\hat{n}_k = n_k$*, then the system is asymptotically stable if*

$$\begin{aligned}&[Q - QL|G(\omega)|\cos(\omega(\hat{\tau} - \tau) + \angle G(\omega))]^2 \\ &+ [-QL|G(\omega)|\sin(\omega(\hat{\tau} - \tau) + \angle G(\omega))]^2 < 1 \ ,\end{aligned}$$

for all $\omega \in \mathbb{R}$.

For SISO systems where τ is known and n_k is unknown, an equivalent inequality to the one stated in Corollary 1 can be obtained by substituting $T_R(\hat{n}_k - n_k)$ for $\hat{\tau} - \tau$. The resulting inequality and its counterpart in Corollary 1 describe the effect that estimation errors in τ and n_k, respectively, have on the stability of the controller.

Remark 3 *When there is non-zero delay estimation error, it can be shown that the ILC algorithm is only stable if $|Q| < 1$ [9]. The error signal, however, cannot converge to zero when $|Q| < 1$. For a stable controller, the asymptotic error of the system is given by*

$$\begin{aligned}
&\|e(t,\infty)\|_2 = \lim_{k\to\infty} \|e(t,k)\|_2 \\
&= \lim_{q\to 1} \left\| (I - G(q^{\bar{n}_k+1} I - Q + QLGq^{\hat{n}_k - n_k} z^{\hat{\tau}-\tau})^{-1} QLq^{\hat{n}_k - n_k} z^{\hat{\tau}-\tau})(y_d(t) - d(t)) \right\|_2 \\
&= \left\| (I - G(I - Q + QLGz^{\hat{\tau}-\tau})^{-1} QLz^{\hat{\tau}-\tau})(y_d(t) - d(t)) \right\|_2 \quad . \qquad (3.11)
\end{aligned}$$

Note that the asymptotic error is not dependent on the n_k estimation error. However, as shown in Section 3.3.2, the n_k estimation error affects the transient behavior of the system.

Remark 4 *For a stable SISO system with a sinusoidal output disturbance at the frequency ω,* (3.11) *can be reduced to the following sensitivity function from $\|d(t)\|_2$ to $\|e(t,\infty)\|_2$:*

$$\|e(t,\infty)\|_2 = \frac{(1-Q)\,\|d(t)\|_2}{\left[(1-Q)^2 + Q^2L^2G^2 + 2(1-Q)QLG\cos(\omega(\hat{\tau}-\tau))\right]^{1/2}} \quad .$$

This expression provides a convenient way to calculate the norm of the asymptotic error of the system given the values of Q, L, and $\hat{\tau} - \tau$. Specifically, I can examine the effect that the delay estimation error has on the coefficient multiplying $\|d(t)\|_2$. In order for the disturbance to be attenuated, the coefficient must have a value less than 1. Using this relationship, I can derive the following inequality which provides a

bound on how much delay estimation error can be tolerated before the error from the disturbance signal is amplified:

$$cos(\omega(\hat{\tau} - \tau)) > \frac{-QLG}{2(1-Q)} \quad . \tag{3.12}$$

3.2 Adaptive Delay Estimation

In this section, I introduce an adaptive measurement delay estimation algorithm for determining $\hat{\tau}$ in Eqn. (3.2), assuming that $\hat{n}_k = n_k$. The goal of this algorithm is to estimate the delay using the information that is available to the ILC algorithm, as described in Sec. 3.1, and to improve the overall performance of the controller.

3.2.1 Nomenclature

To enable this analysis, I propose the following nomenclature. First, I will shift the representation of the plant model and the ILC algorithm into the lifted domain. A lifted-system representation of the linear time-invariant plant is created by expanding $G(z)$ as an infinite power series yielding

$$G(p) = g_0 + g_1 z^{-1} + \cdots + g_{T-1} z^{-(T-1)}$$

where the coefficients g_k are Markov parameters [68]. The sequence $g_0,\ g_1,\ \ldots$ is the impulse response of G. Stacking the input and output signals in vectors, the plant dynamics in Eqn. (3.1) can be written as the $T\ \times\ T$- dimensional lifted system

$$\begin{aligned}\underbrace{\begin{bmatrix} y(\tau,k) \\ y(\tau+1,k) \\ \vdots \\ y(\tau+T-1,k) \end{bmatrix}}_{\boldsymbol{y_k}(\tau)} &= \underbrace{\begin{bmatrix} g_\tau & 0 & \ldots & 0 \\ g_{\tau+1} & g_\tau & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{\tau+T-1} & g_{\tau+T-2} & \ldots & g_\tau \end{bmatrix}}_{\boldsymbol{G}} \underbrace{\begin{bmatrix} u(0,k) \\ u(1,k) \\ \vdots \\ u(T-1,k) \end{bmatrix}}_{\boldsymbol{u_k}} \\ &+ \underbrace{\begin{bmatrix} d(\tau,k) \\ d(\tau+1,k) \\ \vdots \\ d(\tau+T-1,k) \end{bmatrix}}_{\boldsymbol{d}},\end{aligned} \tag{3.13}$$

with the error vector expressed as

$$\underbrace{\begin{bmatrix} e(0,k) \\ e(1,k) \\ \vdots \\ e(T-1,k) \end{bmatrix}}_{\boldsymbol{e_k}} = \underbrace{\begin{bmatrix} y_d(0,k) \\ y_d(1,k) \\ \vdots \\ y_d(T-1,k) \end{bmatrix}}_{\boldsymbol{y_d}} - \underbrace{\begin{bmatrix} y(0,k) \\ y(1,k) \\ \vdots \\ y(T-1,k) \end{bmatrix}}_{\boldsymbol{y_k}}. \tag{3.14}$$

The ILC algorithm (3.7), with $n_k = 0,\ \bar{n}_k = 1$, can be represented in a lifted format as

$$\begin{aligned}\boldsymbol{u}_{k+1} = &\underbrace{\begin{bmatrix} q_0 & q_1 & \cdots & q_{T-1} \\ q_{-1} & q_0 & \cdots & q_{T-2} \\ \vdots & \vdots & \ddots & \vdots \\ q_{-T+1} & q_{-T+2} & \cdots & q_0 \end{bmatrix}}_{\boldsymbol{Q}} \boldsymbol{u_{k-1}} \\ &+ \underbrace{\begin{bmatrix} \ell_0 & \ell_1 & \dots & \ell_{T-1} \\ \ell_{-1} & \ell_0 & \dots & \ell_{T-2} \\ \vdots & \vdots & \ddots & \vdots \\ \ell_{-T+1} & \ell_{-T+2} & \dots & \ell_0 \end{bmatrix}}_{\boldsymbol{L}} \underbrace{\begin{bmatrix} e(\hat{\tau}, k-1) \\ e(\hat{\tau}+1, k-1) \\ \vdots \\ e(\hat{\tau}+T-1, k-1) \end{bmatrix}}_{\boldsymbol{e_{k-1}}(\hat{\tau})} .\end{aligned} \tag{3.15}$$

Then, due to the assumption that each iteration begins at the conclusion of the previous iteration, any samples referring to an index $t > T$ in iteration $k-1$ would correspond to samples at index $t - T$ in iteration k. For the error vector in Eqn. (3.15), this can be represented in the lifted-system form

$$\begin{aligned}\boldsymbol{e_{k-1}}(\hat{\tau}) = &\underbrace{\begin{bmatrix} 0_{T-\hat{\tau}\times\hat{\tau}} & I_{T-\hat{\tau}\times T-\hat{\tau}} \\ 0_{\hat{\tau}\times\hat{\tau}} & 0_{\hat{\tau}\times T-\hat{\tau}} \end{bmatrix}}_{\boldsymbol{R}(\hat{\tau})} \boldsymbol{e_{k-1}} \\ &+ \underbrace{\begin{bmatrix} 0_{T-\hat{\tau}\times\hat{\tau}} & 0_{T-\hat{\tau}\times T-\hat{\tau}} \\ I_{\hat{\tau}\times\hat{\tau}} & 0_{\hat{\tau}\times T-\hat{\tau}} \end{bmatrix}}_{\boldsymbol{S}(\hat{\tau})} \boldsymbol{e_k} \ ,\end{aligned} \tag{3.16}$$

where 0_x and I_x are, respectively, zero and identity matrices of dimension x.

Furthermore, I make the following definition to simplify some of the notation used when deriving the adaptive time delay estimation,

$$e_n(k) \triangleq \left\| \boldsymbol{R}(\hat{\tau}_k)\boldsymbol{e_{k-1}} + \boldsymbol{S}(\hat{\tau}_k)\boldsymbol{e_k} \right\|_\infty \ .$$

3.2.2 Coupled Delay Estimation

As I showed in Eqn. (3.11), the time delay estimate affects the performance of the system described by Eqns. (3.1) and (3.2). Assuming the system is asymptotically stable for all delay estimates, i.e. even when $\hat{\tau}$ is not equal to τ, the asymptotic norm of the error will be greater than the minimum value achievable for a given gain set. The minimum error norm for a given plant and ILC algorithm is determined by setting $\hat{\tau} = \tau$ in Eqn. (3.11) and solving for $\|e(t, \infty)\|_\infty$. I denote this value as e^*.

To drive the norm of the measurement error toward e^*, I propose an adaptation algorithm that updates $\hat{\tau}$ after each iteration to move toward τ. The proposed adaptation law is

$$\hat{\tau}(k+1) = \hat{\tau}(k) + \beta(e_n(k) - e^*) \ , \tag{3.17}$$

where β is the adaptation gain associated with the error between $e_n(k)$ and e^*.

Note that because I am working with a discrete system of period T, the value $\hat{\tau}(k)$ must be kept at integer values in the range $[0,\ T-1]$. This can be accomplished by rounding the value of $\hat{\tau}(k)$ from Eqn. (3.17) down to the nearest integer value using a *floor* operation and then adding or subtracting T, as needed, until the estimate is within the range $[0,\ T-1]$. The latter operation is akin to wrapping a phase angle between $[0,\ 2\pi]$.

Because the adaptation law relies on the error generated by the ILC algorithm, the two are coupled. Therefore, satisfying Eqn. (3.10) does not guarantee asymptotic stability of the coupled algorithm. I provide a new stability criteria in Theorem 3.2.1.

Theorem 3.2.1 *If* $\boldsymbol{Q}$, $\boldsymbol{L}$, *and* β *are chosen such that*

$$\|\boldsymbol{Q}\|_\infty + (\|\boldsymbol{L}\|_\infty + \beta)\,\|\boldsymbol{G}\|_\infty < 1 \ , \tag{3.18}$$

the closed loop adaptive time delay estimation and ILC system described by Eqns. (3.1), (3.2), *and* (3.17) *is asymptotically stable.*

Proof Combining Eqns. (3.15) and (3.16) results in

$$\boldsymbol{u_{k+1}} = \boldsymbol{Q u_{k-1}} + \boldsymbol{L}[\boldsymbol{R}(\hat{\tau}_k)\boldsymbol{e_{k-1}} + \boldsymbol{S}(\hat{\tau}_k)\boldsymbol{e_k}] \ . \tag{3.19}$$

The error vectors in Eqn. (3.19) are obtained by combining Eqns. (3.13) and (3.14) and representing the delay similar to the representation in Eqn. (3.16).

$$\begin{aligned} \boldsymbol{e_{k-1}} &= \boldsymbol{y_d} - [\boldsymbol{R}(\tau)^T \boldsymbol{y_{k-1}} + \boldsymbol{S}(\tau)^T \boldsymbol{y_{k-2}}] \\ &= \boldsymbol{y_d} - \boldsymbol{R}(\tau)^T [\boldsymbol{G u_{k-1}} + \boldsymbol{d}] - \boldsymbol{S}(\tau)^T [\boldsymbol{G u_{k-2}} - \boldsymbol{d}] \\ \boldsymbol{e_k} &= \boldsymbol{y_d} - [\boldsymbol{R}(\tau)^T \boldsymbol{y_k} + \boldsymbol{S}(\tau)^T \boldsymbol{y_{k-1}}] \\ &= \boldsymbol{y_d} - \boldsymbol{R}(\tau)^T [\boldsymbol{G u_k} + \boldsymbol{d}] - \boldsymbol{S}(\tau)^T [\boldsymbol{G u_{k-1}} - \boldsymbol{d}] \end{aligned} \tag{3.20}$$

This system is asymptotically stable if there exists a Lyapunov function $V(X(k))$ such that $V(X(k)) > 0$ for $X \in D - \{0\}$ and $V(X(k+1)) - V(X(k)) \leq 0$ for $X \neq 0$ in a domain D of X [69]. For this system, the fundamental dynamics are $\boldsymbol{u_k}$, $\boldsymbol{u_{k-1}}$, and $\hat{\tau}_k$. Therefore, I consider $X(k)$ to be a vector that contains $\boldsymbol{u_k}$, $\boldsymbol{u_{k-1}}$, and $\hat{\tau}_k$ and that has been defined such that $X = 0$ is an equilibrium point. To accomplish both objectives I define the following terms: $\tilde{\boldsymbol{u}}_{\boldsymbol{k}} = \boldsymbol{u_k} - \boldsymbol{u}_\infty$, $\tilde{\boldsymbol{u}}_{\boldsymbol{k-1}} = \boldsymbol{u_{k-1}} - \boldsymbol{u}_\infty$, and $\tilde{\tau}_k = \tau_k - \tau$, where u_∞ and τ are the equilibrium values of the control signal and the time delay estimate, respectively. Then I can define $X(k)$ as

$$X(k) = \begin{bmatrix} \tilde{\boldsymbol{u}}_{\boldsymbol{k}} \\ \tilde{\boldsymbol{u}}_{\boldsymbol{k-1}} \\ \tilde{\tau} \end{bmatrix} .$$

Consider the following candidate Lyapunov function

$$V(X(k)) = \|\tilde{\boldsymbol{u}}_{\boldsymbol{k}}\|_\infty + \|\tilde{\boldsymbol{u}}_{\boldsymbol{k-1}}\|_\infty + |\tilde{\tau}_k| \ . \tag{3.21}$$

From the definition of the vector ∞-norm and the definition of the absolute value function, $V(X(k)) > 0$ for all nonzero X.

The value of $V(X(k+1))$ is given by

$$\begin{aligned} V(X(k+1)) &= \|\tilde{\boldsymbol{u}}_{\boldsymbol{k+1}}\|_\infty + \|\tilde{\boldsymbol{u}}_{\boldsymbol{k}}\|_\infty + |\tilde{\tau}_{k+1}| \\ &= \|\boldsymbol{Q}\tilde{\boldsymbol{u}}_{\boldsymbol{k-1}} + \boldsymbol{L}[\boldsymbol{R}(\hat{\tau}_k)\tilde{\boldsymbol{e}}_{\boldsymbol{k-1}} + \boldsymbol{S}(\hat{\tau}_k)\tilde{\boldsymbol{e}}_{\boldsymbol{k}} + \boldsymbol{F}(\tilde{\tau}_k)\boldsymbol{e}_\infty]\|_\infty \\ &\quad + \|\tilde{\boldsymbol{u}}_{\boldsymbol{k}}\|_\infty + |\tilde{\tau}_k + \beta(e_n(k) - e^*)| \ , \end{aligned}$$

where $\tilde{\boldsymbol{e}}_{\boldsymbol{k}} = \boldsymbol{e}_{\boldsymbol{k}} - \boldsymbol{e}_\infty$ and $\boldsymbol{F}(\tilde{\tau}_k) = \boldsymbol{R}(\hat{\tau}_k) - \boldsymbol{R}(\tau) + \boldsymbol{S}(\hat{\tau}_k) - \boldsymbol{S}(\tau)$. Then the difference between $V(X(k+1))$ and $V(X(k))$ is given by

$$\begin{aligned} V(X(k+1)) - V(X(k)) = \\ &\|\boldsymbol{Q}\tilde{\boldsymbol{u}}_{\boldsymbol{k-1}} + \boldsymbol{L}[\boldsymbol{R}(\hat{\tau}_k)\tilde{\boldsymbol{e}}_{\boldsymbol{k-1}} + \boldsymbol{S}(\hat{\tau}_k)\tilde{\boldsymbol{e}}_{\boldsymbol{k}} + \boldsymbol{F}(\tilde{\tau}_k)\boldsymbol{e}_\infty]\|_\infty \\ &+ \|\tilde{\boldsymbol{u}}_{\boldsymbol{k}}\|_\infty + |\tilde{\tau}_k + \beta(e_n(k) - e^*)| - \|\tilde{\boldsymbol{u}}_{\boldsymbol{k}}\|_\infty \\ &- \|\tilde{\boldsymbol{u}}_{\boldsymbol{k-1}}\|_\infty - |\tilde{\tau}_k| \end{aligned} \tag{3.22}$$

Expanding $e_n(k)$, and noting $e^* = \|\boldsymbol{e}_\infty\|_\infty = \|[\boldsymbol{R}(\tau) + \boldsymbol{S}(\tau)]\boldsymbol{e}_\infty\|_\infty$ results in

$$\begin{aligned} |\tilde{\tau}_k + \beta(e_n(k) - e^*)| = \\ &|\tilde{\tau}_k + \beta\left(\|\boldsymbol{R}(\hat{\tau}_k)\boldsymbol{e}_{\boldsymbol{k-1}} + \boldsymbol{S}(\hat{\tau}_k)\boldsymbol{e}_{\boldsymbol{k}}\|_\infty - \|[\boldsymbol{R}(\tau) + \boldsymbol{S}(\tau)]\boldsymbol{e}_\infty\|_\infty\right)| \quad . \end{aligned}$$

Substituting this expression into Eqn. (3.22) and using the triangle inequality produces

$$\begin{aligned} V(X(k+1)) - V(X(k)) \leq \\ &\|\boldsymbol{Q}\|_\infty \|\tilde{u}_{k-1}\|_\infty + \|\boldsymbol{L}\|_\infty \|[\boldsymbol{R}(\hat{\tau}_k)\tilde{\boldsymbol{e}}_{\boldsymbol{k-1}} + \boldsymbol{S}(\hat{\tau}_k)\tilde{\boldsymbol{e}}_{\boldsymbol{k}} + \boldsymbol{F}(\tilde{\tau}_k)\boldsymbol{e}_\infty]\|_\infty \\ &+ |\tilde{\tau}_k| + |\beta| \|\boldsymbol{R}(\hat{\tau}_k)\boldsymbol{e}_{\boldsymbol{k-1}} + \boldsymbol{S}(\hat{\tau}_k)\boldsymbol{e}_{\boldsymbol{k}} - [\boldsymbol{R}(\tau) + \boldsymbol{S}(\tau)]\boldsymbol{e}_\infty\|_\infty \\ &- \|\tilde{\boldsymbol{u}}_{\boldsymbol{k-1}}\|_\infty - |\tilde{\tau}_k| \\ =&(\|\boldsymbol{Q}\|_\infty - 1)\|\tilde{u}_{k-1}\|_\infty \\ &+ (\|\boldsymbol{L}\|_\infty + |\beta|)\|[\boldsymbol{R}(\hat{\tau}_k)\tilde{\boldsymbol{e}}_{\boldsymbol{k-1}} + \boldsymbol{S}(\hat{\tau}_k)\tilde{\boldsymbol{e}}_{\boldsymbol{k}} + \boldsymbol{F}(\tilde{\tau}_k)\boldsymbol{e}_\infty]\|_\infty \quad . \end{aligned}$$

Substituting in Eqn. (3.20) and again using the triangle inequality results in

$$\begin{aligned} V(X(k+1)) - V(X(k)) \leq \\ &(\|\boldsymbol{Q}\|_\infty - 1)\|\tilde{u}_{k-1}\|_\infty + (\|\boldsymbol{L}\|_\infty + |\beta|)\|\boldsymbol{F}(\tilde{\tau}_k)\|_\infty \|\boldsymbol{e}_\infty\|_\infty \\ &+ (\|\boldsymbol{L}\|_\infty + |\beta|)\Big\|\boldsymbol{R}(\hat{\tau}_k)[-\boldsymbol{S}(\tau)^T\boldsymbol{G}\tilde{\boldsymbol{u}}_{\boldsymbol{k-2}} - \boldsymbol{R}(\tau)^T\boldsymbol{G}\tilde{\boldsymbol{u}}_{\boldsymbol{k-1}}] \\ &+ \boldsymbol{S}(\hat{\tau}_k)[-\boldsymbol{S}(\tau)^T\boldsymbol{G}\tilde{\boldsymbol{u}}_{\boldsymbol{k-1}} - \boldsymbol{R}(\tau)^T\boldsymbol{G}\tilde{\boldsymbol{u}}_{\boldsymbol{k}}]\Big\|_\infty \end{aligned}$$

By defining $U = \max_{i\in\{k,\ k-1,\ k-2\}} \|\boldsymbol{u}_{\boldsymbol{i}}\|_\infty$ and noting that $\|\boldsymbol{F}(\tilde{\tau})\|_\infty = 0$, I obtain the following inequality, which is satisfied by assumption:

$$V(X(k+1)) - V(X(k)) \leq [\|\boldsymbol{Q}\|_\infty + (\|\boldsymbol{L}\|_\infty + |\beta|)\boldsymbol{G} - 1]U < 0 \quad .$$

■

3.3 Simulation Results

In this section I will demonstrate the functionality of the ILC algorithm that I propose in Section 3.1 through a series of simulations. The simulations will be based on the twin roll strip casting process described in Chapter 2. The rotational nature of the casting process introduces repetitive dynamics into the system, which makes it a good candidate for ILC. Furthermore, due to the high temperatures associated with casting process, there is also a large time delay present in the system. This time delay can span more than one roll revolution in length, which makes feedback control difficult, if not practically infeasible. The feedforward nature of the ILC algorithm is therefore an attractive approach for regulating the strip thickness of the casting process.

3.3.1 Plant Model

Controlling the gap distance between the casting rolls is the primary actuation mechanism used for regulating the thickness profile of the strip. [70]. In order to study the effect that the proposed ILC algorithm has on reducing the strip wedge, I require a plant model that describes how a gap reference signal, specifically a tilt reference command, affects the wedge measurement signal. The tilt of a casting roll is defined as the relative displacement of one end of the roll compared to the other end of the roll. The wedge measurement signal is generated by comparing the thickness of one edge of the strip to the thickness on the opposite edge of the strip.

To construct a model that I can use in simulation and for tuning the ILC algorithm, I use system identification techniques on a dataset where the input tilt signal is a square wave. The resulting normalized measured wedge signal is shown in Fig. 3.2. The normalization is done such that the maximum allowable value for the signal is given a value of 1.

The effect of the input square wave is apparent in Fig. 3.2, but the dynamic response is masked by the presence of the periodic disturbance as well as measurement

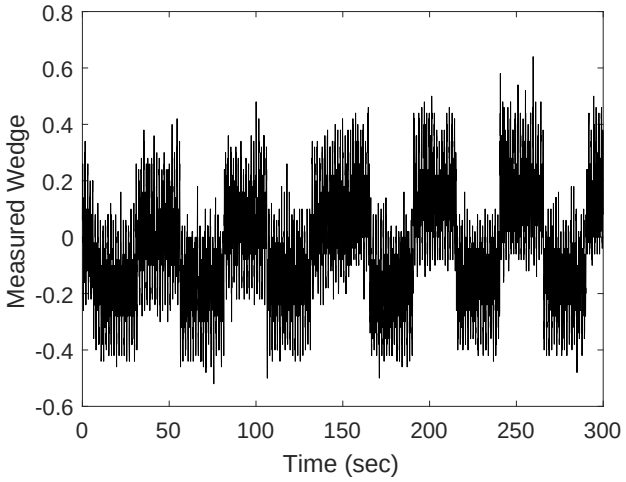


Figure 3.2. The normalized magnitude of the measured wedge signal changes in response to the input signal. The sign of the measurements signifies which side of the strip is thicker.

noise. The measured wedge signal represents a noisy, delayed measurement of the plant's response to the input tilt signal in addition to a periodic disturbance that is caused by the unmodeled eccentric properties of the casting rolls. The relationship between the input, output, and disturbance signals is shown in Fig. 3.3.

To identify a model of the plant using the data shown in Fig. 3.2, I apply filters to remove the disturbance and noise signals. The magnitude plot of the fast Fourier transform of the measured signal is shown in Fig. 3.4. There are large peaks at both the rotational frequency (0.68 Hz) and twice the rotational frequency (1.36 Hz). These correspond to the periodic disturbance signal. There are also peaks at higher harmonics, but the first two harmonics are more than five times greater than the other peaks. Significant measurement noise exists above 1.5 Hz, which can also hinder the plant identification process. To reduce the effect of the disturbance and noise signals, I used MATLAB's `filtfilt` command to filter the measured wedge signal. To attenuate the disturbance signals, I applied two third-order Butterworth band-stop filters: one with cutoff frequencies at 3 rad/sec and 6 rad/sec and another

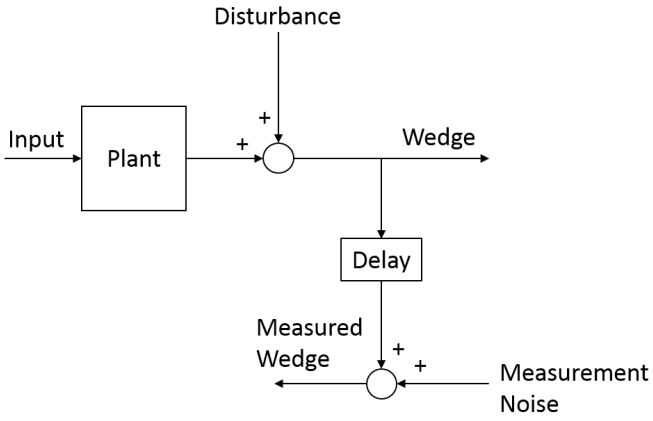


Figure 3.3. The measured wedge signal is a delayed measurement of the plant's response to the input tilt signal summed with a periodic disturbance and measurement noise.

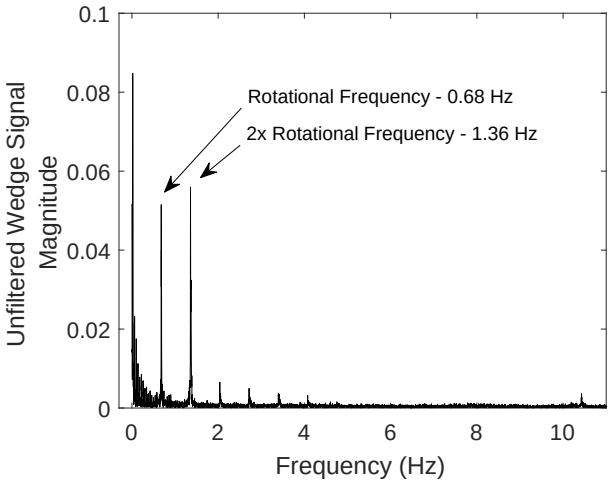


Figure 3.4. The fast Fourier transform of the measured wedge signal shows large peaks at the rotational frequency and twice the rotational frequency.

with cutoff frequencies of 6 rad/sec and 10 rad/sec. The high frequency noise was removed using a sixth-order low-pass Butterworth filter with a cutoff frequency of 9 rad/sec. The resulting filtered signal is plotted in Fig. 3.5.

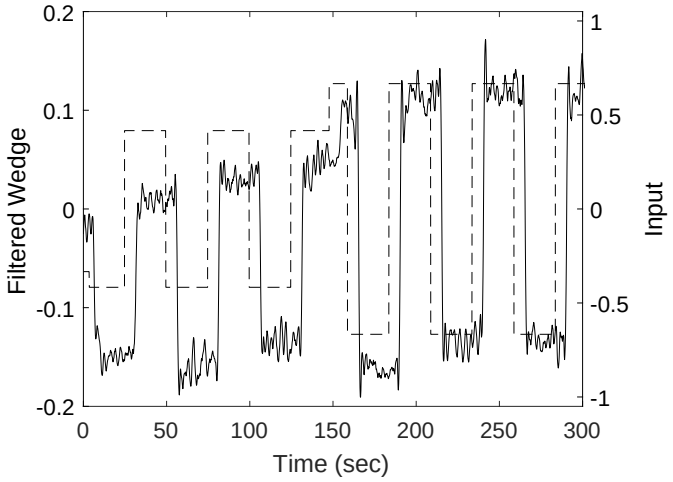


Figure 3.5. The filtered wedge signal reflects the step changes in the input signal. The solid line is the normalized filtered wedge signal and the dashed line is the normalized input signal.

After filtering the wedge signal to obtain $X_{W,f}$, I used the SysID Toolbox in MATLAB to find a polynomial model of the form $A(z)X_{W,f}(t) = B(z)u(t)$ given by

$$X_{W,f}(k) = 0.186z^{-671}u(k) \ . \tag{3.23}$$

The model is able to achieve a normalized root mean square error fit percentage of 81.65% as shown in Fig. 3.6. Note that the exponent of z represents the time delay of the process, in samples. The value that the SysID Toolbox determined was 671 samples.

3.3.2 Simulation Results

Using the plant model identified in Section 3.3.1 and the error function defined in (3.3), I obtain

$$e(t,k) = y_d(t) - 0.186u(t-\tau, k-n_k) + \Delta(t) \ . \tag{3.24}$$

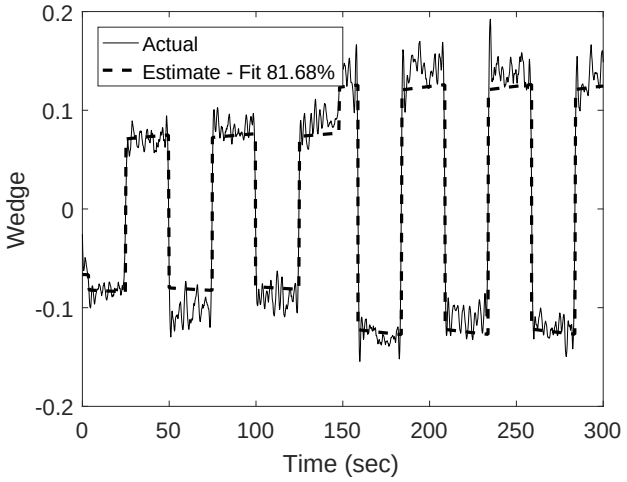


Figure 3.6. A comparison of the estimated plant dynamics to the filtered wedge dynamics.

For this simulation I consider the case in which $y_d(t) = 0 \forall t$, $\tau = 10$, $n_k = 4$, and $\Delta(t) = \sin(\frac{2\pi}{T_R}t)$, where $T_R = 180$ samples. I use a control law of the same form as (3.7), where

$$\begin{aligned} u(t, k + \bar{n_k} + 1) = Q(z)[u(t,k) \\ + L(z)(-\sin\left(\frac{2\pi}{T_R}(t + \hat{\tau} - \tau)\right) \\ - 0.186u(t + \hat{\tau} - \tau, k + \hat{n}_k - n_k))] \ . \end{aligned} \tag{3.25}$$

If both $\hat{\tau} = \tau = 10$ and $\hat{n}_k = n_k = 4$, Proposition 3.1.2 states that the system will be asymptotically stable if I choose $Q > 0$ and $L > 0$ such that

$$\|Q - 0.186QL\|_\infty < 1 \ .$$

Choosing $Q = 1$ means I may choose any $L < 10.75$. Considering $L = 5$, the norm of the error signal converges to zero as shown in Fig. 3.7.

If $\hat{\tau} \neq \tau$, but $\hat{n}_k = n_k = 4$, Corollary 1 states that the system will be asymptotically stable if I choose $Q > 0$ and $L > 0$ such that

$$(Q - 0.186QL\cos(10\omega))^2 + (0.186QL\sin(10\omega))^2 < 1 \ ,$$

for all $\omega \in \mathbb{R}$. Choosing a gain set of $Q = 0.7$ and $L = 10/7$ satisfies this criteria for all $\hat{\tau} \in [0,\ T_R)$. As Figs. 3.8 and 3.9 show, the norm of the error signal resulting from an ILC algorithm with this gain set converges for all $\hat{\tau} \in [\tau,\ \tau + T_R/2]$, but the final value is never zero. This is expected, as mentioned in Remark 3, because $Q < 1$ and there are errors in the estimate of τ. Note that the range $\hat{\tau} \in [\tau,\ \tau + T_R/2] = [10,\ 100]$ represents the full range of errors in the estimate of τ in this example because of the periodicity of the process.

Furthermore, I can see in Fig. 3.9 that when

$$\cos\left(\frac{\pi}{90}(\hat{\tau} - \tau)\right) < \frac{-(0.7)(10/7)(0.186)}{2(1 - 0.7)}$$

the asymptotic error is greater than the initial error. In these cases, the delay estimation error is too large for the ILC algorithm to improve system performance over open loop operation. Note that in the case when $\hat{\tau} = 100$, the angle of the $-QLG$ vector in Fig. 3.1 is $2\pi/180(100 - 10) = \pi$ radians, which places the $-QLG$ arrow on the positive real axis, pointing away from the origin. This is the worst possible case for the delay estimation.

As noted in Remark 3, the n_k estimate does not affect the asymptotic error. This is illustrated in Fig. 3.10, where an ILC algorithm of the form (3.8) and gain set $Q = 0.7$ and $L = 10/7$ causes the norm of the error signal to converge to the same steady-state value, regardless of the n_k estimate. The transient behavior of the system, however, varies considerably. Underestimating n_k leads to faster convergence, but the behavior becomes oscillatory in the iteration-domain. This may be unacceptable for a given application.

Overall, these demonstrations illustrate how the proposed delay estimation algorithm can be incorporated into an ILC framework. They also show that the ILC algorithm is capable of reducing the magnitude of the error signal, even in the presence of nonzero estimation errors when (3.12) is satisfied.

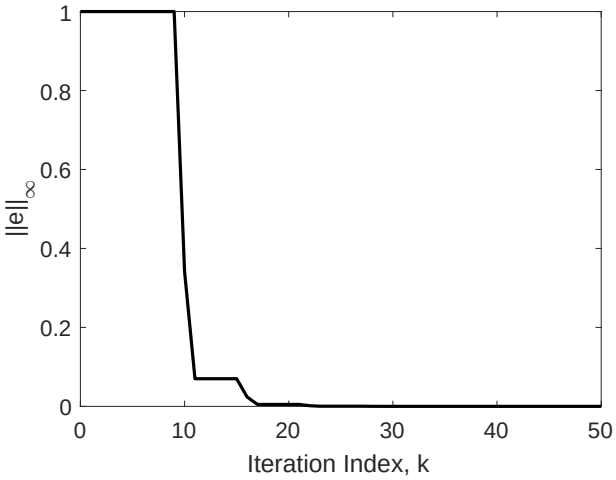


Figure 3.7. When the estimated values of n_k and τ are equal to their true values, the norm of the error signal converges to zero asymptotically.

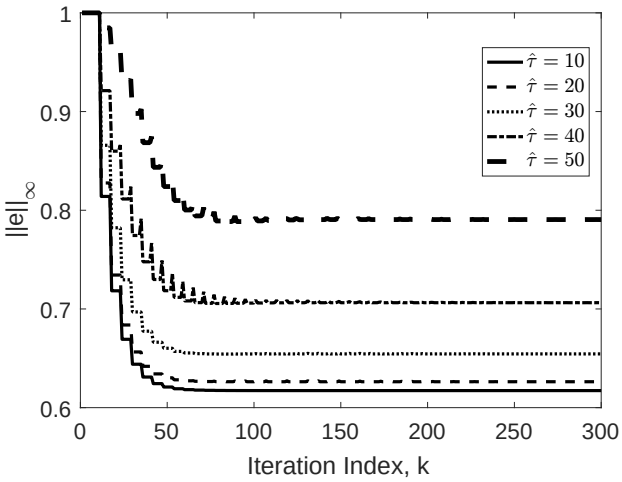


Figure 3.8. When the estimated value τ differs from its true value by a small amount, the norm of the error signal still converges to a value that is less than the initial error.

3.4 Simulation Results for Adaptive Time Delay Estimation

In this section, we demonstrate the performance of the combined iterative learning control and adaptive time delay estimation algorithm proposed in Section 3.2 using

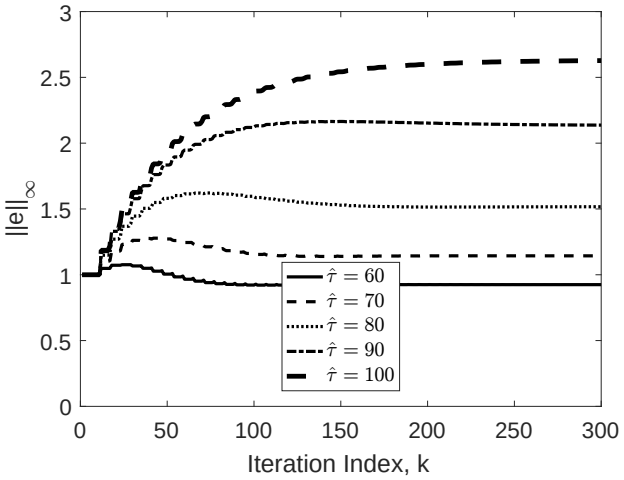


Figure 3.9. When the estimated value τ differs from its true value by a large amount, the norm of the error signal converges to a value greater than its initial value.

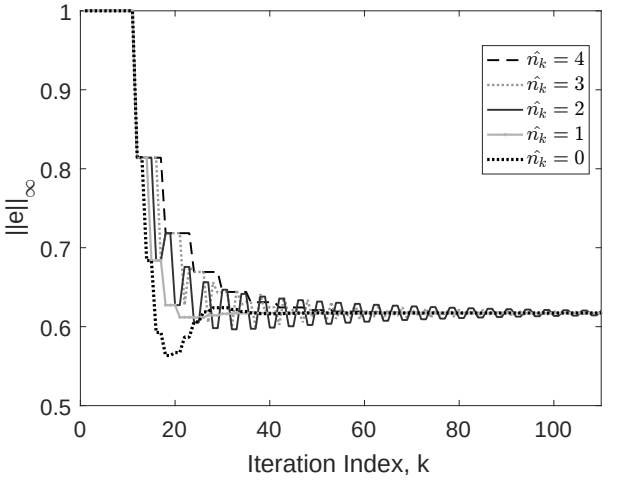


Figure 3.10. When the estimated value n_k differs from its true value by a small amount, the norm of the error signal still converges to a value that is less than the initial error, but the transient response changes.

a simulated case study. For these simulations, we consider the plant model in Eqn.

$$\begin{aligned}\|e(t,\infty)\| &= \left\| \frac{1-Q}{[(1-Q)^2+L^2|G(\omega)|^2+2(1-Q)L|G(\omega)|\cos(\omega(\hat{\tau}-\tau)+\angle G(\omega))]^{1/2}}(y_d(t)-d(t)) \right\|_\infty \\ &= \left\| \frac{0.3}{[(1-0.7)^2+(1)^2(0.186)^2+2(1-0.7)(1)(0.186)\cos\left(\frac{2\pi}{180}(\hat{\tau}-(50))\right)]^{1/2}}(0-d(t)) \right\|_\infty \\ &= \left| \frac{0.3}{[0.124596+.1116\cos\left(\frac{2\pi}{180}(\hat{\tau}-50)\right)]^{1/2}} \right|\end{aligned} \tag{3.26}$$

(3.1) with $G = 0.186$, $\tau = 50$ samples, $d(t) = \sin(\frac{2\pi}{T}t)$, and $T = 180$ samples. The control objective is to reduce the output of the system to zero, i.e. $y_d(t) = 0$. To do this we will use the ILC control law defined in Eqn. (3.2) and $\hat{\tau}(k)$ given by Eqn. (3.17). Different values of $\hat{\tau}(0)$ will be considered in the case study. Based on the plant model G, we choose $Q(p) = 0.7$, $L(p) = 1$, and $\beta = 0.5$ so that Eqn. (3.18) is satisfied. With this system description, the asymptotic error for a static $\hat{\tau}$ in Eqn. (3.11) can be expressed as Eqn. (3.26). This enables us to compare the proposed algorithm's performance against the performance of an ILC algorithm that uses a static delay estimate.

The first case we illustrate is one in which $\hat{\tau}(k) = \tau$, for all k. In this case, we expect that the ILC algorithm will reduce the norm of the error to a value of 0.61, which is equivalent to e^* for the given plant and ILC algorithm. As shown in Fig. 3.11, this is indeed achieved. The reason the error does not converge to zero is because a nonunity Q value was used to make the algorithm robust to time delay estimation error.

In the second case, we use the adaptation algorithm in Eqn. (3.17) with an initial delay estimate of $\hat{\tau}(0) = 80$ samples. As shown in Fig. 3.12, this estimation error causes the error signal to deviate away from the minimal error. If we used $\hat{\tau} = 80$ for all k, the norm of the error converges to 0.71. However, with the proposed adaptation law, we are able to reduce the norm of the error to the minimal value by changing the value of $\hat{\tau}$. The value of $\hat{\tau}$ is shown in Fig. 3.13. Due to the sign of β, the delay estimate initially deviates further away from the true value of τ, resulting in the norm

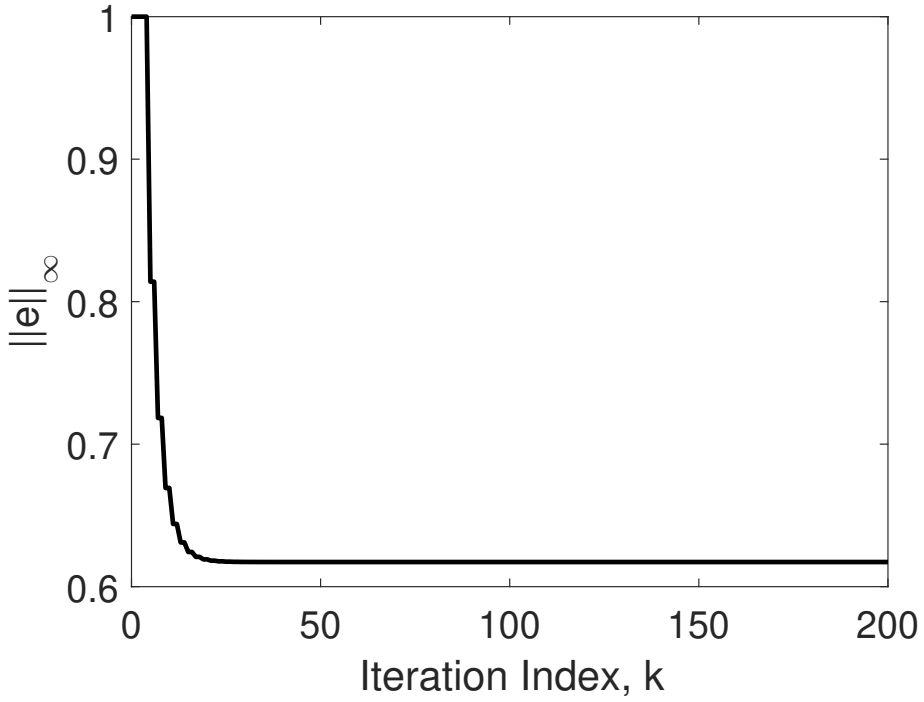


Figure 3.11. The norm of the error signal in case 1 converges to the theoretical minimum value of 0.61.

of the error signal deviating from the optimal trajectory. After the value of $\hat{\tau}$ moves above $\hat{\tau} = \tau + T/2$, the norm of the error signal begins to decrease until it converges to e^* as $\hat{\tau}$ wraps back to zero and converges to τ.

In the third case, we again use the adaptation algorithm in Eqn. (3.17), but with an initial delay estimate of $\hat{\tau}(0) = 140$ samples. This represents the worst case scenario, where $\hat{\tau}$ is $T/2$ samples away from the true value of τ and the cosine function in Eqn. (3.26) has a value of -1. In this case, the ILC algorithm we have proposed remains asymptotically stable with a steady state norm of the error signal of 2.63, meaning that the ILC algorithm results in a larger error than if the system was operating open loop. With the adaptive time delay estimate, however, we are able to reduce the norm of the error signal to the minimal value 0.61, as shown in Fig. 3.14. To do this, the adaptation law increased the value of $\hat{\tau}$ until it reached T, at which point we wrapped the value back to 0, so that $\hat{\tau}$ stayed in the range

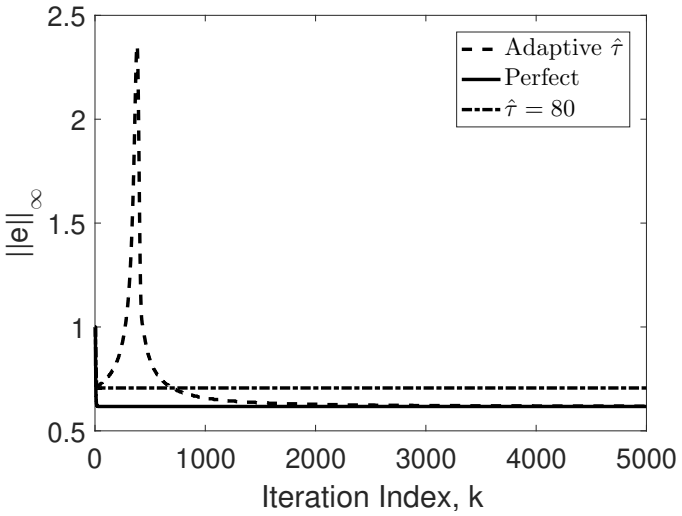


Figure 3.12. The norm of the error signal in case 2 initially is larger than if a static $\hat{\tau} = 80$ is used. It eventually converges to the same value as case 1, which represents the performance with a perfect delay estimate.

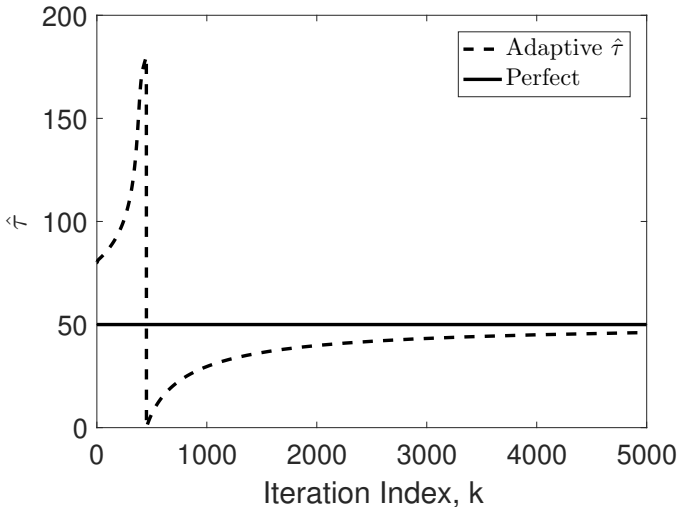


Figure 3.13. Time delay estimate versus the true time delay for case 2.

$0 \leq \hat{\tau} < T$. Then the delay estimate increased from 0 until it converged to the true value of $\tau = 50$. The wrapping effect and the convergence to $\tau = 50$ is shown in Fig. 3.15.

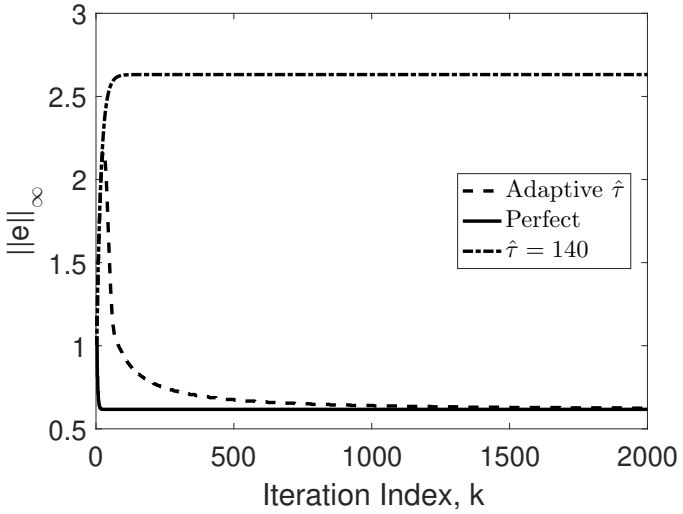


Figure 3.14. The norm of the error signal in case 3 is always smaller than if a static $\hat{\tau} = 140$ is used. It eventually converges to the same value as case 1, which represents the performance with a perfect delay estimate.

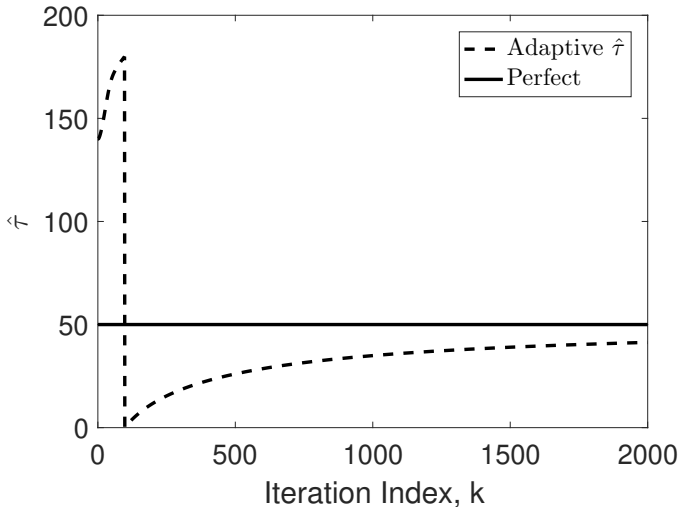


Figure 3.15. Time delay estimate versus the true time delay for case 3.

Note that in both cases 2 and 3, the delay estimate increases until it wraps around from $\hat{\tau} = T$ to $\hat{\tau} = 0$ and then continues to increase until $\hat{\tau}$ converges to τ. The reason for this is that $\beta(e_n(k) - e^*)$ is a strictly positive number with my choice of

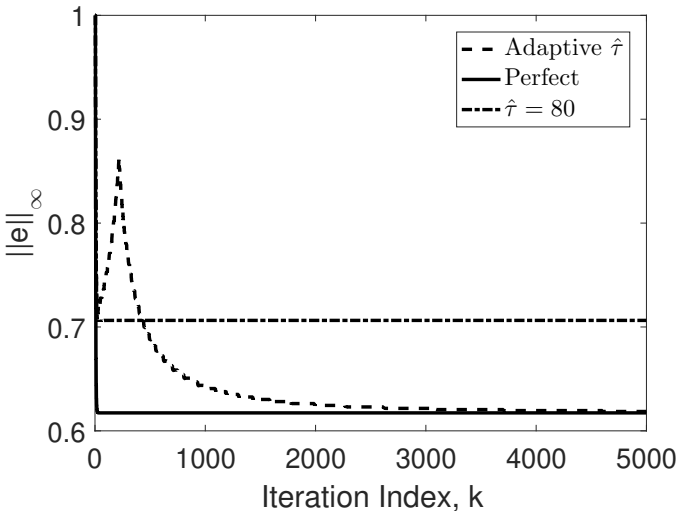


Figure 3.16. The norm of the error signal in case 4 behaves similarly to the norm of the error signal in case 2. In this case, however, the norm never exceeds $e_n(0)$.

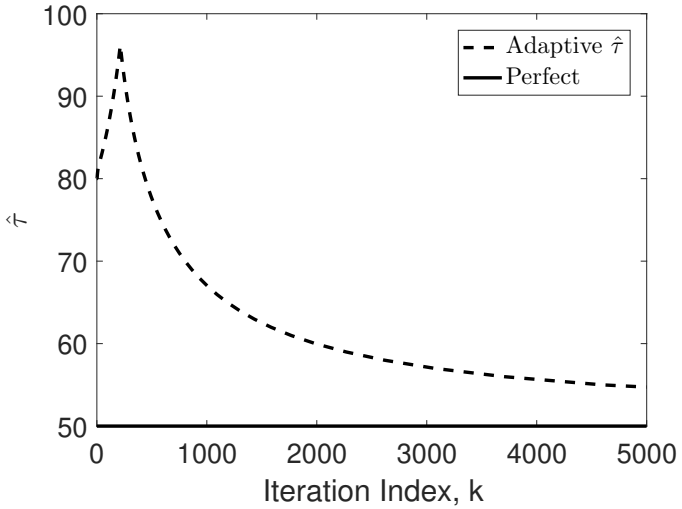


Figure 3.17. Time delay estimate versus the true time delay for case 4.

β. To correct for this, we can multiply β by the term $\gamma(k) \in \{-1,\ 1\}$, which we can use to adapt the direction in which the delay estimate is updated, i.e. increased or decreased. We denote the new adaptation gain as $\bar{\beta} = \beta\gamma(k)$. Due to the definition of

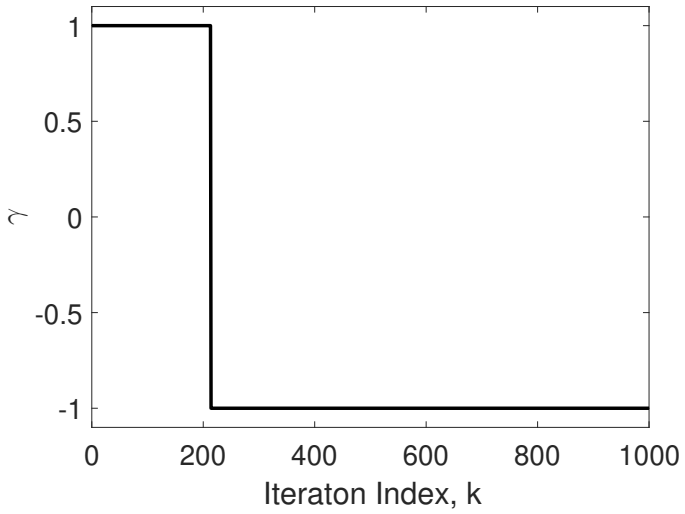


Figure 3.18. Switching parameter, $\gamma(k)$, for case 4.

$\gamma(k)$, $|\bar{\beta}| = |\beta|$ and condition (3.18) is still satisfied, guaranteeing asymptotic stability. One way to update the direction of the adaptation law is to change the sign of $\gamma(k)$ if $e_n(k-1) - e_n(k-2) > \epsilon$ and $e_n(k-1) - 2e_n(k-2) + e_n(k-3) > 0$ for some small positive number ϵ. This update causes the adaptation law to switch directions if the norm of the error signal increases from iteration to iteration and if the rate of that increase is increasing. This makes the algorithm perform in a similar manner to a gradient descent method.

Case 4 repeats the simulation from case 2 using the new $\bar{\beta}$ adaptation gain in place of β, with $\gamma(0) = 1$ and $\epsilon = 0.001$. This results in the error profile shown in Fig. 3.16 and the delay estimate shown in Fig. 3.17. After the change in the error profile has satisfied both $e_n(k-1) - e_n(k-2) > \epsilon$ and $e_n(k-1) - 2e_n(k-2) + e_n(k-3) > 0$, the sign of γ changes, as shown in Fig. 3.18. This change in the adaptation direction causes $\hat{\tau}$ to converge to τ without having to wrap from $\hat{\tau} = T$ to $\hat{\tau} = 0$. This also results in a norm of the error signal being below the value of $e_n(0)$ at every iteration, meaning that the combined ILC and adaptive time delay estimation algorithm improved performance of the system at every iteration.

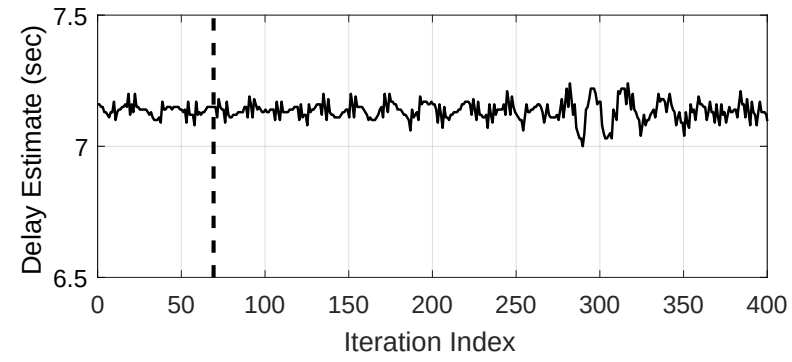
3.5 Experimental Validation

In addition to the simulation studies, I tested the combined time delay estimation and ILC algorithm at Castrip's facility in Nucor Corporation's plant in Crawfordsville, Indiana. In one test, I used a constant gain of $Q = 0.8$ and $L = 10/7$. The delay estimate, T_D, and the normalized $\|e_W\|_\infty$ for each iteration k are plotted in Fig. 3.19. In this case, $\|e_W\|_\infty$ is defined as the vector ∞-norm of the error signal over all samples t in iteration k. The dashed line at iteration 70 designates the time at which the ILC algorithm was initiated. The normalization was done in such a way that the average $\|e_W\|_\infty$ during the period before the ILC was initiated was assigned a value of 1. The wedge measurement is not perfectly periodic, which causes the scatter seen in the $\|e_W\|_\infty$ plot. Therefore, I analyze the average reduction that the algorithm is able to achieve. From the time the ILC algorithm was initiated until iteration 400, the ILC algorithm reduced the wedge by approximately 48%, on average.

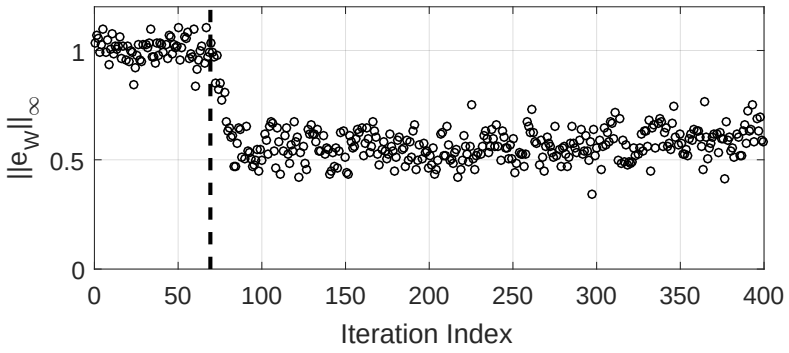
In another test conducted on a different day with a different roll set, I used a constant gain of $Q = 0.8$ and $L = 1.25$. The delay estimate, T_D, and the normalized $\|e_W\|_\infty$ for each iteration k are plotted in Fig. 3.20. The dashed line at iteration 100 designates the point at which the ILC algorithm was initiated.The results are normalized such that the average value of $\|e_W\|_\infty$ during the first 100 iterations is assigned a value of 1.

To analyze the effectiveness of the ILC algorithm in this test, I focus on the results from the first 400 iterations, which are shown in Fig. 3.21. From the time the ILC algorithm was initiated until iteration 400, the ILC algorithm was able to reduce the wedge by approximately 29%, on average.

At approximately iteration 2900, the grade of steel being cast changed, and the casting speed changed in reaction. The change in the casting speed is shown indirectly by the changes in the delay estimate in Fig. 3.20(a). Focusing on the region of the test where the grade of steel changed, Fig. 3.22 shows that, after some initial transients, the ILC algorithm was able to recover from the change in the process and reduce the



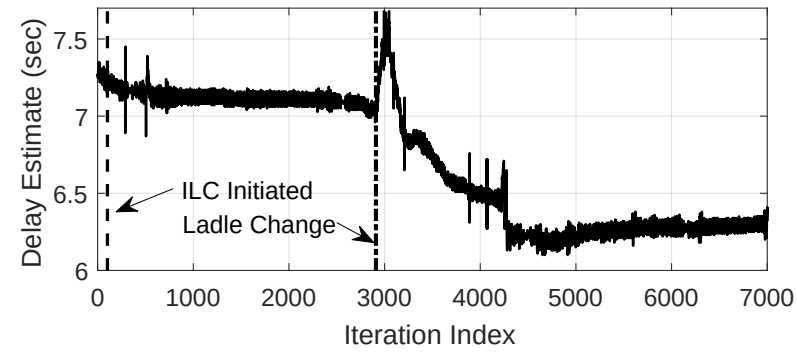
(a)



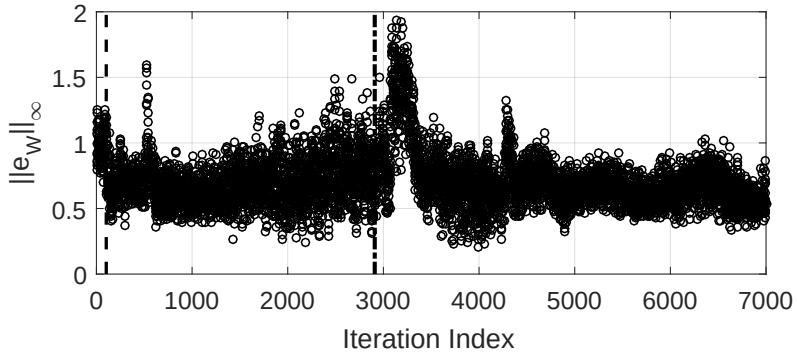
(b)

Figure 3.19. In the first test, the combined time delay estimation and ILC algorithm is able to reduce the norm of the wedge signal by approximately 48%, on average.

norm of the wedge signal beginning at approximately iteration 3500. The change in the grade of steel occurred when the operator changed ladles during the cast. The ladles are used to pour molten steel on the casting rolls. During a single cast, multiple ladles may be used to maximize the strip production. However, each ladle may contain a slightly different grade of steel, which can introduce an aperiodic disturbance to the casting process. The use of a non-unity Q-filter makes the ILC algorithm robust to these aperiodic disturbance because the non-unity Q-filter discounts the learning from previous control inputs.



(a)

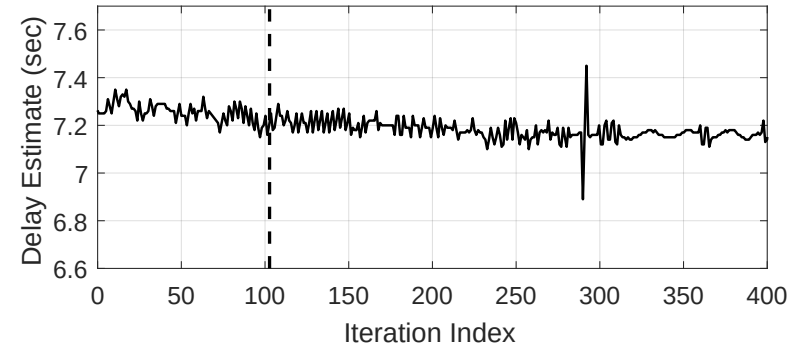


(b)

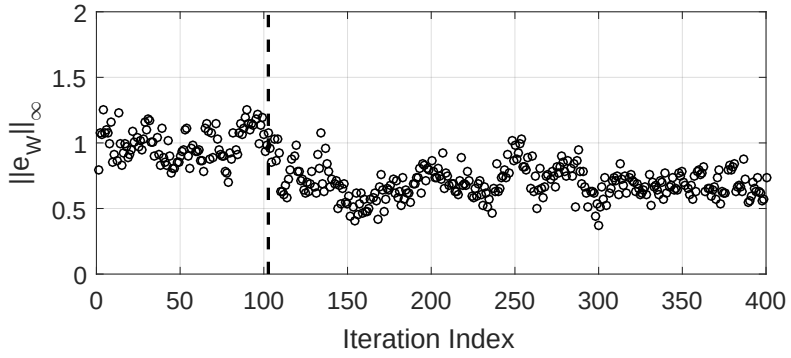
Figure 3.20. In the second test, the ILC algorithm was initiated at approximately iteration 100. There was a ladle change at approximately iteration 2900, which caused the wedge to increase and the casting speed to change.

3.6 Chapter Summary

In this chapter I described two different iterative learning control (ILC) algorithms designed to compensate for time-delays in periodic systems. First, I proposed an ILC algorithm for a class of periodic processes with a variable time-delay that is greater than one iteration in length. I separated the total delay estimate into two components: an iterative estimate, n_k, based on the number of iterations contained within one delay period and a residual estimate, τ. This structure enabled the derivation of an ILC stability law that is a function of the estimation error in n_k and in τ.



(a)

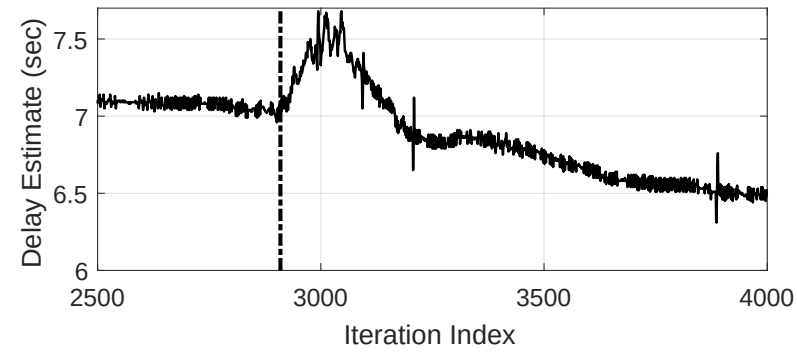


(b)

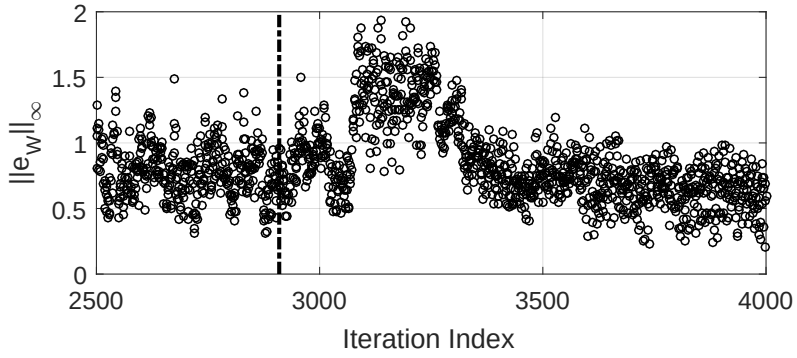
Figure 3.21. In the second test, the combined time delay estimation and ILC algorithm is able to reduce the norm of the wedge signal by approximately 29%, on average.

Through a simulated case study, I demonstrated the sensitivity of the ILC algorithm to estimation error in n_k and in τ as well trade-offs in performance that arise through errors in each estimate.

I also proposed a coupled adaptive measurement delay estimation and ILC algorithm for a class of periodic systems characterized by unknown measurement delay. The delay estimation algorithm works similarly to a gradient descent method that drives the norm of the output error signal from the ILC system to its minimal value. I provided a sufficient condition that can be used to design an asymptotically stable algorithm. Through simulation, I demonstrated that the combined measurement de-



(a)



(b)

Figure 3.22. After a ladle change causes a change in the grade of steel being cast in the second test, the ILC algorithm is able to recover from the change in the process and reduce the norm of the wedge signal within approximately 500 roll revolutions.

lay estimation and ILC algorithm is capable of reducing the norm of the error signal to its minimal value, even if the initial delay estimate is 180 degrees out of phase with the process.

In addition to the simulations, I applied an uncoupled ILC and time delay estimation algorithm to a twin roll strip casting process operated by Nucor Steel in Crawfordsville, Indiana. The ILC algorithm was used to counteract one measure of the strip eccentricity that appears during the casting process. By defining one iteration of the process as one roll revolution, the ILC law I proposed in Section 3.1

reduced the norm of the strip wedge signal by approximately 50% compared the strip wedge that is present before the ILC algorithm is used.

Overall, the two ILC algorithms described in this chapter demonstrated that incorporating the delay estimate into the ILC formulation enables more robustness and better performance. However, both algorithms are sensitive to plant variations. In Fig. 3.22, the process parameters changed when the casting speed changed, as described in Chapter 2. This resulted in the increase in the norm of the wedge signal seen at approximately iteration 3100 in Fig. 3.22(b) because the algorithm did not account for the change in operating condition. In the coupled ILC and delay estimation algorithm, a similar problem arises if the plant model is not known perfectly and e^* is incorrect. In that case, the adaptive time delay does converge to the true delay and the algorithm is not guaranteed to reduce the error. Both of these scenarios motivate the work discussed in Chapter 4, where both parametric and delay uncertainty are factored into the formulation of a norm-optimal ILC algorithm.

4. LINEAR PARAMETER VARYING ITERATIVE LEARNING CONTROL

As stated in Chapter 2, the twin roll strip casting process motivating this research can be characterized as a periodic system that can experience parametric uncertainty, time delay estimation error, and linear parameter varying dynamics, especially if the rotational speed changes. These features can result in poor system performance or even instability if not considered in the controller design. For that reason, it is important to examine how all three types of features affect the iterative learning control algorithm and how the algorithm can be formulated to improve its robustness.

In this chapter I describe a norm-optimal ILC algorithm that defines the ILC filters such that both performance and robustness criteria are satisfied. Norm-optimal ILC algorithms have been proposed recently, e.g., [17–20], for achieving robust control in the presence of multiplicative plant uncertainty. However, as discussed in Chapter. 1, existing literature does not consider how norm-optimal ILC can be extended to plants with time delay uncertainty and LPV dynamics.

4.1 Problem Definition

To begin the analysis, consider the following discrete-time plant model with measurement delay

$$y(t,k,\theta) = G(p,\theta)p^{-\tau}u(t,k) + d(t) \ , \tag{4.1}$$

where $G(p,\theta)$ is an asymptotically stable, parameter-varying plant model mapping the input signal $u(t,k)$ at sample $t \in [0,\ T]$ in iteration $k \in \mathbb{Z}_+$ to the measured output signal, $y(t,k)$, and p is a forward shift operator in the sample t-domain. The signal $d(t)$ represents a periodic (iteration-invariant) disturbance signal that is assumed to be bounded such that $\|d(t)\|_\infty \leq d_m$. The variable θ represents the scheduling parameter about which a set of plant models is defined. The measurement

delay, τ, is assumed to be an unknown integer value in the range $0 \leq \tau < T$ samples. Additionally, I assume that the initial conditions are reset at the beginning of each iteration such that $y(t,k) = 0$ for $t \in [-T,\ 0]$.

The control objectives for this plant are defined as follows:

O1 *Tracking Performance* - Minimize the tracking error between $y(t,k)$ and an iteration-invariant desired output signal $y_d(t)$ where the tracking error is given by

$$e(t,k) = y_d(t) - y(t,k) \ . \tag{4.2}$$

O2 *Input Saturation Avoidance* - Prevent input saturation by satisfying,

$$|u(t,k)| \leq u_{sat} \ , \tag{4.3}$$

where u_{sat} is the saturation level.

O3 *Convergence Speed* - Achieve convergence of the error signal to a maximum tolerable error norm value, $\bar{E}$, within K iterations.

To achieve the control objectives, I assume a process model of the following form with multiplicative uncertainty:

$$y(t,k) = \hat{G}(p,\theta,\hat{\tau})[1 + W_o(p,\theta)\Delta(p)]u(t,k) + d(t) \ , \tag{4.4}$$

where $\hat{G}$ is the nominal process model, $\hat{\tau}$ is the delay estimate, $W_o(p,\theta)$ is a stable, invertible, and potentially parameter-dependent weighting matrix, and Δ is a stable transfer transfer function such that $\|\Delta(p)\|_\infty < 1$. The vectors $y(t,k)$, $u(t,k)$, and $d(t)$ are as defined in (4.1).

To reject the periodic disturbance, I define an ILC algorithm of the form

$$u(t,k+1) = Q(\theta)u(t,k) + L(\theta,\hat{\tau})e(t+1,k) \ , \tag{4.5}$$

where $Q(\theta)$ and $L(\theta,\hat{\tau})$ are filters to be designed.

Because ILC operates over a finite period of time in each iteration, a lifted setting can be used to analyze the system. A lifted-system representation of the linear time-invariant plant is created by expanding $G(p,\theta,\tau)$ as an infinite power series yielding

$$G(p,\theta,\tau) = g_\tau(\theta)p^{-\tau} + g_{\tau+1}(\theta)p^{-\tau-1} + \cdots + g_{\tau+T-1}(\theta)p^{-\tau-(T-1)}$$

where the coefficients $g_k(\theta)$ are Markov parameters that may depend on θ [68]. The sequence $g_\tau(\theta),\ g_{\tau+1}(\theta),\ \ldots$ is the impulse response of $G(p,\theta,\tau)$. Stacking the input and output signals in vectors, the plant dynamics in Eqn. (4.4) can be written as the $T \times T$ dimensional lifted system

$$\boldsymbol{y_k} = \hat{\boldsymbol{G}}(\boldsymbol{\theta},\hat{\boldsymbol{\tau}})[\boldsymbol{I} + \boldsymbol{W_o\Delta}]\boldsymbol{u_k} + \boldsymbol{d} \tag{4.6}$$

with the error vector expressed as

$$\underbrace{\begin{bmatrix} e(0,k) \\ e(1,k) \\ \vdots \\ e(T-1,k) \end{bmatrix}}_{\boldsymbol{e_k}} = \underbrace{\begin{bmatrix} y_d(0,k) \\ y_d(1,k) \\ \vdots \\ y_d(T-1,k) \end{bmatrix}}_{\boldsymbol{y_d}} - \underbrace{\begin{bmatrix} y(0,k) \\ y(1,k) \\ \vdots \\ y(T-1,k) \end{bmatrix}}_{\boldsymbol{y_k}} . \tag{4.7}$$

The ILC update law can also be represented in the lifted domain as

$$\boldsymbol{u_{k+1}} = \boldsymbol{Q}\boldsymbol{u_k} + \boldsymbol{L}\boldsymbol{e_k} \tag{4.8}$$

One desired outcome of the ILC algorithm is *robust convergence*, which, for the ILC system (4.6), (4.8), is defined as having the property that there exists $0 < \gamma < 1$ for all $\boldsymbol{G}$ such that for some $k > K \in \mathbb{N}$:

$$\|\boldsymbol{e_{k+1}}\|_2 \leq \gamma \|\boldsymbol{e_0}\|_2 \quad ,$$

where $\boldsymbol{e_0}$ is the initial error.

Satisfying control objectives O1 and O3 is directly influenced by the the robust convergence property of the system. Control objective O1 is effected by the size of γ. To minimize the tracking error of the system, a minimal value of γ is needed. Similarly, to satisfy control objective O3 and converge to a maximum tolerable norm value of the error signal, γ must satisfy $\gamma \leq \bar{E}/\left\|\boldsymbol{e_0}\right\|_2$.

4.2 Robust Norm Optimal ILC

To achieve robust convergence and also satisfy control objectives O1, O2, O3, I propose a robust norm-optimal ILC algorithm. Norm-optimal ILC algorithms have previously been used in literature [18, 19] to address the robust convergence property but have not considered the effect that delays and parameter variations have on the design procedure.

For a norm optimal ILC algorithm, the control objectives are recast as an optimization problem with the cost function

$$J = \boldsymbol{e_{k+1}}^T R \boldsymbol{e_{k+1}} + \boldsymbol{U_{k+1}}^T S \boldsymbol{U_{k+1}} + (\boldsymbol{U_{k+1}} - \boldsymbol{U_k})^T T (\boldsymbol{U_{k+1}} - \boldsymbol{U_k}) \ , \tag{4.9}$$

where $R = R^T > 0$, $S = S^T > 0$, $T = T^T > 0$ denote weighting matrices that are to be designed. The minimal value of the cost function can be solved for the nominal model of the process by substituting (4.6) and (4.7) with $\boldsymbol{W_o} = 0$ into the cost function and solving $\frac{\partial J}{\partial U_{k+1}} = 0$. After substituting (4.6) and (4.7) into (4.9) and dropping the $\hat{\tau}$ term from $\boldsymbol{G(\theta, \hat{\tau})}$ to simplify the notation, J is given as,

$$\begin{aligned} J &= (\boldsymbol{y_d} - \boldsymbol{y_{k+1}})^T R (\boldsymbol{y_d} - \boldsymbol{y_{k+1}}) + \boldsymbol{U_{k+1}}^T S \boldsymbol{U_{k+1}} + (\boldsymbol{U_{k+1}} - \boldsymbol{U_k})^T T (\boldsymbol{U_{k+1}} - \boldsymbol{U_k}) \\ &= (\boldsymbol{y_d} - \hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}}) \boldsymbol{U_{k+1}} - \boldsymbol{d})^T R (\boldsymbol{y_d} - \hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}}) \boldsymbol{U_{k+1}} - \boldsymbol{d}) \\ &\quad + \boldsymbol{U_{k+1}}^T S \boldsymbol{U_{k+1}} + (\boldsymbol{U_{k+1}} - \boldsymbol{U_k})^T T (\boldsymbol{U_{k+1}} - \boldsymbol{U_k}) \ . \end{aligned} \tag{4.10}$$

Then $\frac{\partial J}{\partial U_{k+1}}$ is given by,

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{U_{k+1}}} &= 2\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R(\boldsymbol{y_d} - \hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})\boldsymbol{U_{k+1}} - \boldsymbol{d}) + 2S\boldsymbol{U_{k+1}} + 2T(\boldsymbol{U_{k+1}} - \boldsymbol{U_k}) \\ &= 2\left(\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}}) + S + T\right)\boldsymbol{U_{k+1}} \\ &\quad - 2\left(T + \hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_k})\right)\boldsymbol{U_k} - 2\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\boldsymbol{e_k}\end{aligned} \tag{4.11}$$

Setting (4.11) equal to zero and solving for $\boldsymbol{U_{k+1}}$ results in

$$\begin{aligned}\boldsymbol{U_{k+1}} = &\left(\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}}) + S + T\right)^{-1} \\ &\times \left(\left(T + \hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_k})\right)\boldsymbol{U_k} + \hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\boldsymbol{e_k}\right) \ .\end{aligned} \tag{4.12}$$

Noting that (4.12) has the same structure as (4.8), setting both equations equal to each other yields the following optimal values for $\boldsymbol{Q}$ and $\boldsymbol{L}$:

$$\boldsymbol{Q} = \left(\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}}) + S + T\right)^{-1}\left(T + \hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_k})\right) \tag{4.13}$$

$$\boldsymbol{L} = \left(\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}}) + S + T\right)^{-1}\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R \tag{4.14}$$

While the optimal ILC filters are designed based on the nominal model $\hat{\boldsymbol{G}}$, a sufficient condition for achieving robust convergence is needed and provided in [19] as

$$\|\boldsymbol{Q} - \boldsymbol{L}\boldsymbol{G}(\boldsymbol{\theta_k}, \boldsymbol{\tau})\|_{i2} < 1 \ . \tag{4.15}$$

Substituting (4.8), (4.13), and (4.14) into (4.15) results in (4.16), which can be used to design the weighting matrices ($R,\ S,\ T$) such that robust convergence is achieved for the true system with multiplicative uncertainty.

$$\max_{\Delta}\left\|\left(\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}}) + S + T\right)^{-1}\left(T - \hat{\boldsymbol{G}}(\boldsymbol{\theta_{k+1}})^T R\hat{\boldsymbol{G}}(\boldsymbol{\theta_k})\boldsymbol{W_o}\boldsymbol{\Delta}\right)\right\|_{i2} < 1 \tag{4.16}$$

In (4.16), it can be seen that S is the only matrix that is exclusively in the inverted portion of the expression. Because of this, S is the most powerful term for ensuring

robust convergence. Increasing $\|S\|_{i2}$ yields a smaller quantity on the left hand side of (4.16). However, S also has an effect on the error attenuation in the system because it is the weighting matrix for the input signal in the cost function, and increasing $\|S\|_{i2}$ results in less emphasis on tracking performance.

The error attenuation of the system can be evaluated by examining the error as the iteration index increases. For an LTI system, the notion of $e_\infty \triangleq \lim_{k\to\infty} e_k$ is commonly used to evaluate the performance. For an LPV case such as the one considered in this thesis, the error signal does not necessarily converge as $k \to \infty$. Instead, I define e^* as $e^* \triangleq \limsup_{k\to\infty} \|e_k\|$ and use e^* to evaluate the error attenuation ability of the ILC algorithm [20]. The value e^* exists if the system is bounded, which is guaranteed if (4.16) is satisfied. To evaluate e^*, I assume $|y_d(t)| < \zeta,\ \forall\ t \in [0, T]$ and that the ILC update law satisfies

$$\|\boldsymbol{Q} - \boldsymbol{LG}\|_2 < \lambda < 1\ , \tag{4.17}$$

where $\boldsymbol{G}$ is given by (4.6) and represents all admissible plants, and $\boldsymbol{Q}$ and $\boldsymbol{L}$ are given by (4.13) and (4.14), respectively. This then yields the same inequality as (4.16), with the notable addition of requiring that the left hand side is less than $\lambda < 1$.

By assuming (4.17) is true, it can be shown that the input signal converges to a bounded region about the nominal input signal, $\bar{u}_k$, which is defined as the input that is produced by the ILC algorithm assuming the nominal model is correct:

$$\limsup_{k\to\infty} \left\|U_k - \bar{U}_k\right\| \leq \|L\| \frac{\left\|\hat{G}W_o\right\| \left\|\bar{U}_\infty\right\| + \zeta + d_M}{1-\lambda}\ . \tag{4.18}$$

Then the error can be shown to converge to a bounded region about the nominal error, $\bar{e}_k$, which is defined as the error that is achieved if the nominal model is correct:

$$\limsup_{k\to\infty} \|e_k - \bar{e}_k\| \leq \|L\| \left(\frac{\left\|\hat{G}\right\| + \left\|\hat{G}W_o\right\|}{1-\lambda} + 1 \right) \left(\left\|\hat{G}W_o\right\| \left\|\bar{U}_\infty\right\| + \zeta + d_M \right)\ . \tag{4.19}$$

In (4.18) and (4.19), it is clear that the quantity $\|L\|/(1-\lambda)$ affects the magnitude of the upper bound. I can then tune the values of $(R,\ S,\ T)$ to minimize the value of $\|L\|/(1-\lambda)$ subject to satisfying objectives O1-O3 and (4.17).

To satisfy all three control objectives, I propose the following tuning method.

Method 1 *Norm Optimal LPV ILC Tuning Method*

1. *Design R: Let $R = I$ for uniform weighting of the error.*
2. *Design T: Use T to minimize λ where $\left\| Q - L\hat{G}(I + W_o) \right\| < \lambda$ and satisfy O3.*
3. *Design S: Use S to minimize $\|\boldsymbol{L}\| / (1-\lambda)$ and to make sure that O2 is satisfied by constraining the right hand side of (4.18) to be less than u_{sat} with $\bar{U} = 0$.*
4. *Iterate process: If ILC performance does not meet design specifications, one may need to return to any previous step, including system identification, to alter the ILC tuning in order to satisfy the desired convergence and performance objectives.*

I use T to minimize λ because, as shown in (4.16), adjusting T affects the relative weighting of the uncertainty within the convergence criteria. It is also the natural candidate to achieve O3 because its role in the cost function (4.9) is to weigh the change in $\boldsymbol{U}$ between iterations, which is directly related to how fast the ILC algorithm can converge.

I use S in step 3 to minimize $\|\boldsymbol{L}\| / (1 - \lambda)$ because the S term is in the inverted portion of the definition of $\boldsymbol{L}$ in (4.14), which means scaling S directly affects the numerator of $\|\boldsymbol{L}\| / (1 - \lambda)$. S is also the natural candidate for achieving O2 because it provides the weight for the input signal's contribution to the cost function in (4.9).

4.3 Case Study

4.3.1 Simulation Plant: Second Order System

Many systems can be described as second order with variable damping and stiffness. For example, in Chapter 2, I discussed how a variable damping and stiffness model can be used to represent the compression dynamics in a twin roll strip casting

process. A second order system with variable damping and stiffness can be represented as

$$m\frac{d^2x(t)}{dt^2} + b(\theta)\frac{dx(t)}{dt} + k(\theta)x(t) = u(t) \ , \tag{4.20}$$

where m is the mass of the system, $b(\theta)$ is the damping coefficient, and $k(\theta)$ is the stiffness coefficient. Both $b(\theta)$ and $k(\theta)$ are dependent on the scheduling parameter θ. The signal $x(t)$ is the state of the system and $u(t)$ is the input to the system. In the twin roll casting example, $x(t)$ would correspond to the thickness of the strip and $u(t)$ would correspond to the force applied to the strip by the casting rolls.

In this model, I assume that the damping and stiffness coefficients can be modeled by the following relationships:

$$b(\theta) = b_0 + f_b(\theta) \ , \tag{4.21}$$

$$k(\theta) = k_0 + f_k(\theta) \ , \tag{4.22}$$

where b_0 and k_0 represent the nominal damping and stiffness coefficients and $f_b(\theta)$ and $f_k(\theta)$ are linear functions that describe how the scheduling parameter affects the coefficients. For this thesis, I assume that $f_b(\theta)$ and $f_k(\theta)$ are known but that there may be some uncertainty in the nominal damping and stiffness coefficients.

Additionally, I assume that there is a measurement delay τ such that the output signal, $y(t)$, is not measured until τ samples after $u(t)$ is applied to the system. The output can be described by the following relationship where C describes measurement dynamics and $d(t)$ represents a disturbance that repeats during every iteration of the system,

$$y(t) = Cx(t-\tau) + d(t) \ . \tag{4.23}$$

In the following subsections I examine how to design the norm optimal ILC described in Section 4.2 for different kinds of model features. The model features tested in each case are summarized in Table 4.1.

Table 4.1. The model features tested in each case.

Case	Parametric Uncertainty	Delay Uncertainty	LPV Dynamics
1	X		
2		X	
3			X
4	X		X
5	X	X	X

For this case study, I consider a second order LPV model described by (4.20), (4.21), (4.22), and (4.23) that operates over multiple iterations in succession with the initial conditions resetting after each iteration. For each simulation I use model parameters listed in Table 4.2 and assume that the system is effected by the periodic disturbance shown in Fig. 4.1 during every iteration.

Table 4.2. The nominal parameters used in the simulated case study.

Parameter	Value
b_0	100
C	1
f_b	$(10-\theta)$
f_k	$(\theta-10)/10$
k	10
m	10
u_{sat}	200
y_d	0
τ	0
θ	3

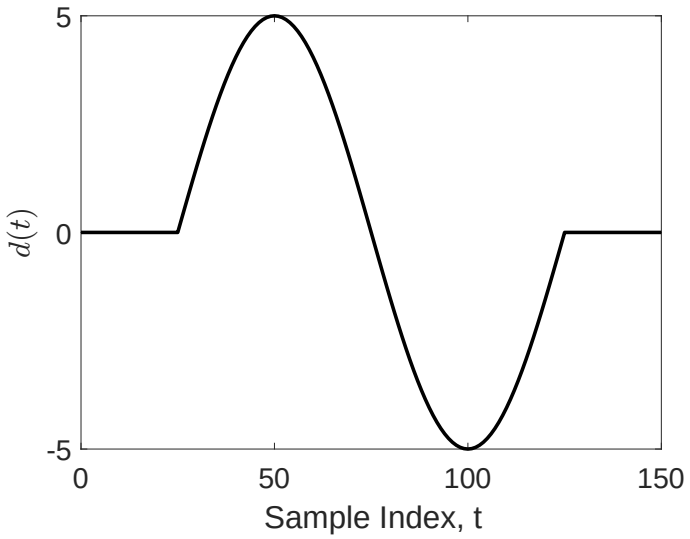


Figure 4.1. The disturbance signal that is repeated during every iteration of the simulated case study.

4.3.2 Case 1: Plant Parameter Uncertainty

In Case 1, I evaluate the performance of the proposed control algorithm in the presence of plant parameter uncertainty. I assume that both b_0 and k_0 are within a known bounded range of possible values, $b_l \leq b_0 \leq b_u$ and $k_l \leq k_0 \leq k_u$. From those bounds, I define two models to represent the edges of the admissible model set. I denote G^- as the model produced using $b_0 = b_u$ and $k_0 = k_u$, which represent the model with the lowest magnitude. I also denote G^+ as the model produced using $b_0 = b_l$ and $k_0 = k_l$, which represents the model with the highest magnitude. It follows that G is defined as

$$G = G^- \left(1 + \frac{G^+ - G^-}{G^-}\Delta\right) \tag{4.24}$$

where $0 \leq \Delta \leq 1$ represents the uncertainty. Note that this has the same form as (4.6), with $W_o = \frac{G^+ - G^-}{G^-}$. Also, note that larger model uncertainty results in larger values of W_o.

With the parametric uncertainty relationship given by (4.24), Method 1 takes the the following form.

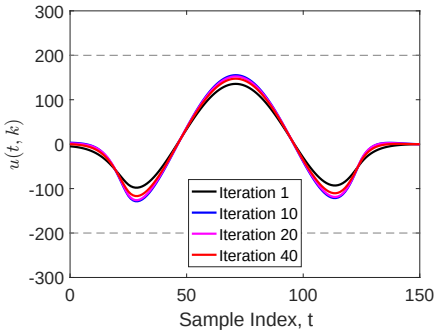
1. Design R: Let $R = I$ for uniform weighting of the error.

2. Design T: Use T to minimize λ where $\|Q - LG^{+}\| < \lambda$ and satisfy O3.

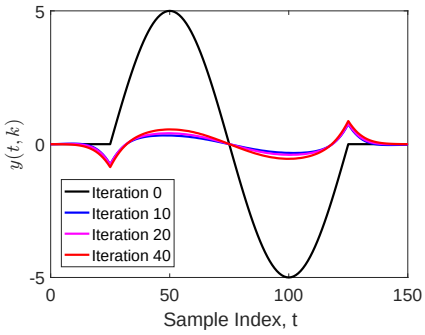
3. Design S: Use S to minimize $\|L\|/(1-\lambda)$ and to make sure that O2 is satisfied by constraining the right hand side of (4.18) to be less than u_{sat} with $\bar{U} = 0$.

4. Iterate process: If the ILC's performance does not meet design specifications, one may need to return to any previous step, including system identification, to alter the ILC tuning in order to satisfy the desired convergence and performance objectives.

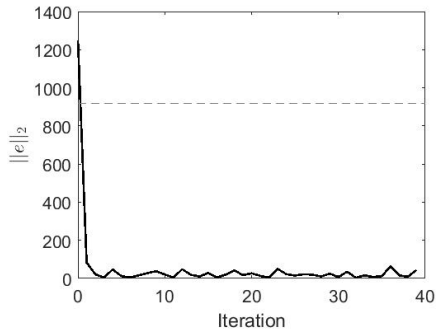
For this case study, I assume that G^{-} is the nominal model. Then I assume that the maximum amount of parametric variability in each iteration is 50%, which corresponds to a multiplicative uncertainty filter value of $W_o = 0.5$. After applying the proposed tuning guidelines, I find that setting $R = I$, $T = (1 \times 10^{-4})\, I$, and $S = (1 \times 10^{-4})\, I$ results in $\lambda = 0.5$ and $\|L\|/(1-\lambda) = 66.8$. This results in the magnitude of $u(t,k)$ remaining less than $u_{sat} = 200$ in each iteration and at each sample as shown in Fig. 4.2(a). With these tuning values, the closed loop system's ability to reject the disturbance improves by a factor of 50, as shown in Fig. 4.2(b). This corresponds to $\|e(t,k)\|_2 < 50$ for every iteration after the ILC algorithm is applied to the system in iteration 1, as shown in Fig. 4.2(c). Additionally, Fig. 4.2(d) shows that the chosen norm optimal ILC design reduces the cost function from approximately 1250 when the ILC is not applied to the system to approximately 150 after the ILC is applied. Note that the error norm in this case is significantly lower than the value of e^* calculated using (4.19), which is shown by the dashed line in Fig. 4.2(c). This is because parametric variations are rarely at the peak deviation away from the nominal model and the model is mostly accurate.



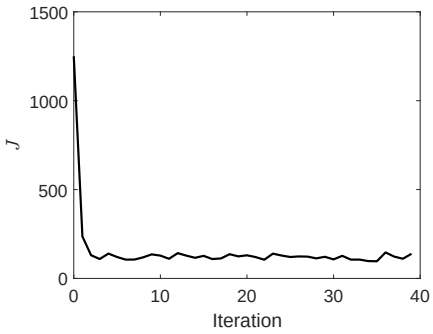
(a) Input signal at different iterations. The dashed lines represent the saturation limit u_{sat}.



(b) Output signal at different iterations.



(c) The norm of the error signal at each iteration. The dashed line represents e^*.



(d) The cost at each iteration.

Figure 4.2. Simulation results for Case 1 - parametric uncertainty.

4.3.3 Case 2: Delay Estimation Uncertainty

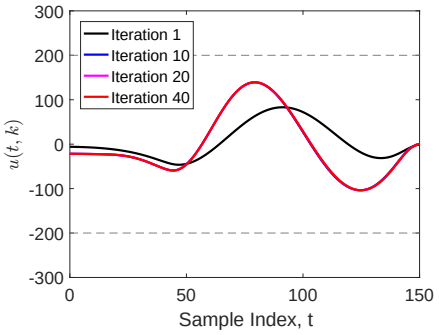
To control a system with an uncertain time delay, an approach similar to that described in [71] can be used to characterize the delay uncertainty as a multiplicative uncertainty. In [71], Wang et al. show that if the delay uncertainty is bounded above by δ_τ, then the weighting function

$$w_0(s) = \frac{\delta_\tau s}{1 + \delta_\tau s/2} \tag{4.25}$$

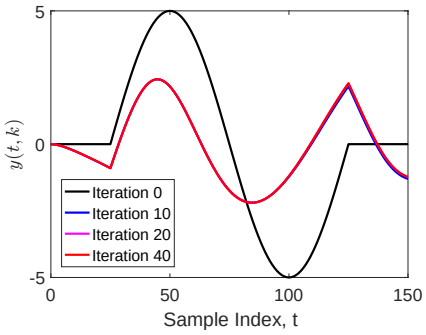
can be used to represent the delay uncertainty in a multiplicative uncertainty framework. After representing (4.25) as a lifted domain matrix W_o, I use Method 1 to tune the norm-optimal LPV ILC algorithm.

Assuming that the delay is known to within 20 samples, i.e., $\delta_\tau = 20$, Method 1 can be used to find that the values $R = I$, $S = (5 \times 10^{-4})\, I$, and $T = (5 \times 10^{-4})\, I$ will result in stable and convergent behavior. This is illustrated in Fig. 4.3, which shows the results produced when applying the ILC algorithm to a simulated plant with a measurement delay of $\tau = 20$ samples but a delay estimate of $\hat{\tau} = 0$ samples. Figure 4.3(a) shows that the input signal never exceeds a magnitude of $u_{sat} = 200$, which means O2 is satisfied. Figure 4.3(b) shows that, despite having a delay estimation error of 20 samples, the ILC algorithm is able to reduce the magnitude of the error signal by a approximately 60%. There is also an 80% reduction in the 2-norm sense, as seen in Fig. 4.3(c). Finally, Fig. 4.3(d) shows that the cost is reduced from approximately 1250 to approximately 650.

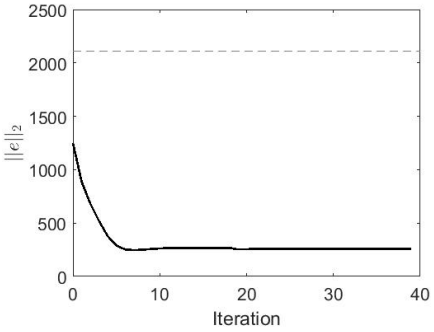
Note that this improvement in performance relative to the first iteration is much less than the performance improvements achieved in Case 1. The reason for this is that by making the controller robust to a large range of potential time delay estimation errors, I had to make the controller quite conservative. This is reflected in the choice of S: $S = (5 \times 10^{-4})\, I$ in Case 2 versus $S = (1 \times 10^{-4})\, I$ in Case 1. The larger eigenvalues of S are needed in Case 2 to offset the phase mismatch that occurs due to the introduction of delay estimation error. Using the same tuning values from Case 1 on the system in Case 2 results in the magnitude of the input signal exceeding u_{sat}. The large input values are a product of the integral-like behavior that is inherent in the ILC algorithm structure. Under ideal conditions where the delay is known and compensated for appropriately, the ILC algorithm updates $u(t, k+1)$ using $u(t, k)$ and the error signal generated by applying $u(t, k)$. When there is delay estimation error δ_τ, however, $u(t, k+1)$ is updated using $u(t, k)$ and error information that is generated by $u(t - \delta_\tau, k)$. This causes the input signal to be out of phase with the error signal that it is trying to control, in turn resulting in a situation where the input signal may



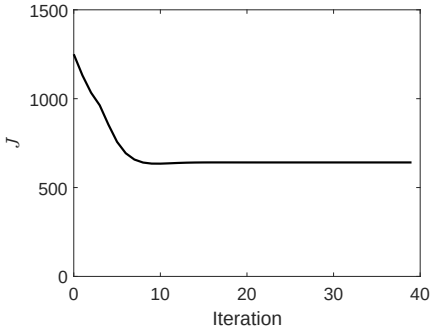
(a) Input signal at different iterations. The dashed lines represent the saturation limit u_{sat}.



(b) Output signal at different iterations.



(c) The norm of the error signal at each iteration. The dashed line represents e^*.



(d) The cost at each iteration.

Figure 4.3. Simulation results for Case 2 - Time Delay Uncertainty.

increase after each iteration in an effort to decrease the tracking error. This behavior resembles integrator wind up seen in proportional-integral feedback controllers with input saturation. The way to counteract the "wind up" is to detune the controller by decreasing the eigenvalues of $\boldsymbol{Q}$, as also shown in Chapter 3. In the norm-optimal framework proposed in this chapter, a method of reducing the eigenvalues of $\boldsymbol{Q}$ is to increase the eigenvalues of S. By increasing the eigenvalues of S, I reduce the magnitude of the input signal so that control objective O2 is satisfied. The results I show in this case apply to the situation when the delay estimation uncertainty

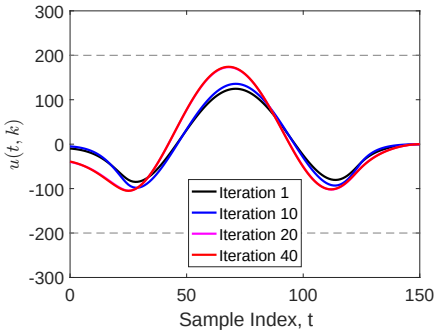
is relatively small (20 samples out of a possible 150 samples in an iteration in this simulation). If the estimation error were larger, the ILC algorithm would not be able to simultaneously reduce the tracking error and satisfy the saturation condition. This is similar to the results shown in Sec. 3.3.2. In those situations, an adaptive time delay estimate, such as the algorithm proposed in Sec. 3.2, could be used to decrease the estimation uncertainty.

4.3.4 Case 3: LPV Dynamics

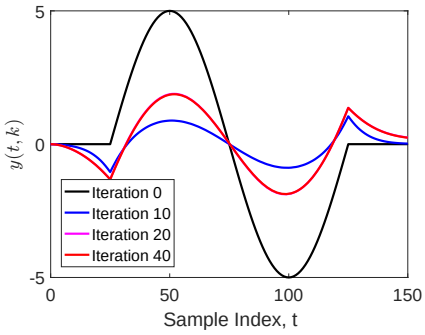
In this case, I study the effects that the LPV dynamics have on the performance of the ILC algorithm. I assume that the model is known exactly and that the scheduling parameter, θ, changes at iteration $k = 20$ from $\theta = 3$ to $\theta = 5$. This alters the damping and stiffness coefficients as described in (4.21) and (4.22). For this case, the tuning guidelines I propose in Method 1 show that the weighting matrices $R = I$, $S = (2 \times 10^{-4})\, I$, and $T = 10^{-4} I$ result in satisfactory performance for both $G(\theta = 3)$ and $G(\theta = 5)$, which are the models used in this case. The results of this simulation are shown in Fig. 4.4. During the iterations when $\theta = 3$, the results are similar to the results in Case 1, with the noticeable absence of the fluctuations in Figs. 4.4(c) and 4.4(d) because I assume that the plant parameters are known in this case. After iteration 20, when $\theta = 5$, the error signal increases both in magnitude and in the 2-norm sense because the system becomes more stiff, as described by (4.22). This causes the system to be less responsive to the input signal.

By incorporating an LPV model structure in the problem formulation, the ILC control law depends on both $G(\theta_k)$ and $G(\theta_{k+1})$. Allowing the ILC filters to change as the scheduling parameter changes results in better performance than if an LTI model of the plant is used as is done in existing literature [20].

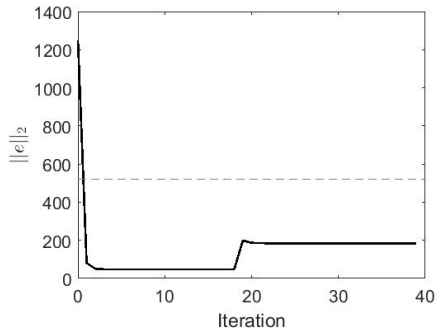
Additionally, by using both θ_k and θ_{k+1} when calculating $\boldsymbol{Q}$ in (4.13), I am able to improve the transient response of the system when the scheduling parameter changes. This is demonstrated in Fig. 4.5, which shows the value of $\|e_k\|_2$ produced when using



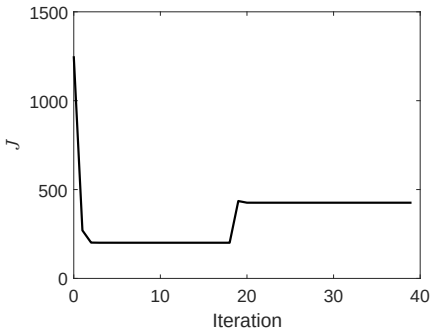
(a) Input signal at different iterations. The dashed lines represent the saturation limit u_{sat}.



(b) Output signal at different iterations.



(c) The norm of the error signal at each iteration. The dashed line represents e^*.



(d) The cost at each iteration.

Figure 4.4. Simulation results for case 3 - LPV dynamics.

different definitions of $\boldsymbol{Q}$. The performance shown in the solid line, which is the result of defining $\boldsymbol{Q}$ using (4.13), allows for a smoother transition after θ changes. At the transition point, the LTI approach causes the 2-norm of the error to approximately double compared to the LPV approach I propose in this thesis.

4.3.5 Case 4: LPV Dynamics with Parametric Uncertainty

To illustrate that the proposed control algorithm can achieve robust stability and performance in the presence of more than one type of plant variation, in Case 4 I apply

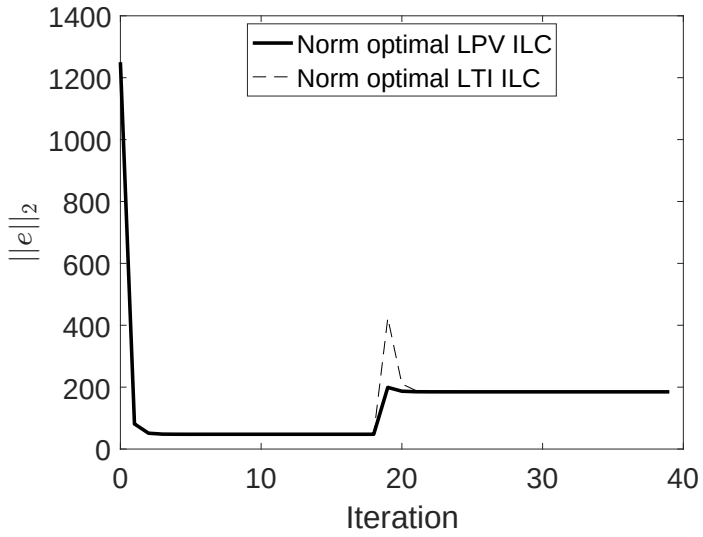


Figure 4.5. The norm of the error signal obtained when using (4.13) has a better transient response to changes in θ than an ILC law based solely on $G(\theta_{k+1})$.

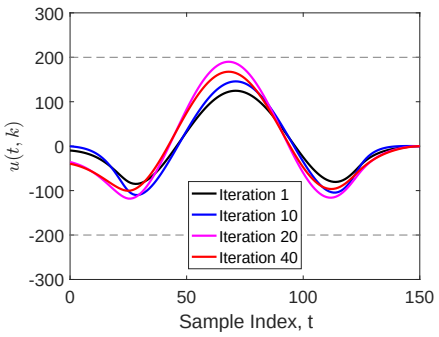
the proposed LPV ILC algorithm to a plant that experiences both the parametric uncertainty described in Case 1 and the LPV dynamics described in Case 3. Here the design procedure proposed in Method 1 is again used with the uncertainty model proposed in Case 1, and the robust convergence criteria is tested for both $G(\theta = 3)$ and $G(\theta = 5)$. Because the formulation is similar to Case 3, the same tuning set can be used for Case 4, i.e, $R = I$, $S = (2 \times 10^{-4})\, I$, and $T = (1 \times 10^{-4})\, I$.

The results of this case study are shown in Fig. 4.6. As Fig. 4.6(a) shows, control objective O2 is satisfied because the magnitude of the input signal is less than $u_{sat} = 200$. Figures 4.6(b) and 4.6(c) show that the error is attenuated within 10 iterations despite the parametric uncertainty. This means that control objectives O1 and O3 are satisfied. During the first 20 iterations, the performance of the system is similar to the performance in Case 1 because $\theta = 3$ is the scheduling parameter used during Case 1 and the first half of Case 4. The error attenuation in Case 4 is slightly worse than in Case 1 because the eigenvalues of S are larger in Case 4. During iterations 21-40, the amount of error attenuation achieved in Case 4 is again

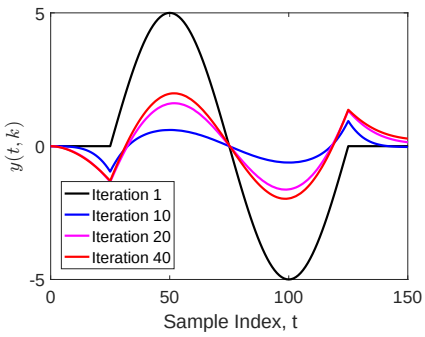
less than the amount of error attenuation achieved in Cases 1 and 3. In addition to the larger eigenvalues of S, less error attenuation is achieved after iteration 20, when $\theta = 5$, because the nominal system has a lower process gain due to the variations in b and k described in (4.21) and (4.22) respectively. The 50% variability associated with the nominal plant gain can then result in some iterations of the process that are less reactive to a given control input than the previous iterations, resulting in larger values of $\|e_k\|_2$. Conversely, when the parametric variability produces higher process gains than expected, the system can become more reactive to the input signal. This may reduce $\|e_k\|_2$ in some iterations but may also generate overreactions to the input signal and produce larger values of $\|e_k\|_2$. By incorporating the LPV dynamics and the multiplicative representation of the parametric uncertainty into the tuning guidelines, I generate an ILC tuning set that limits the effects of the uncertainty and enables the controller to limit the value of $\|e_k\|_2$ to less than e^*, as shown in Fig. 4.6(c). This in turn enables the controller to achieve all three control objectives described in Section 4.1.

4.3.6 Case 5: LPV Dynamics with Parametric and Delay Estimation Uncertainty

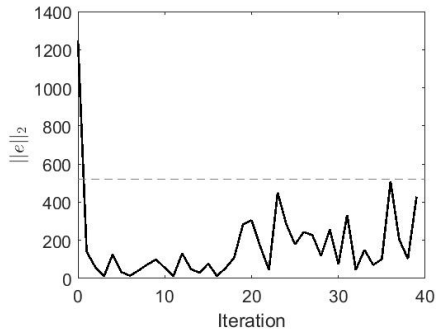
Case 5 highlights the combined effect that all three plant features have on the tuning of the LPV ILC algorithm. The LPV dynamics are accounted for by using the ILC formulation (4.5). The parametric and delay estimation uncertainty are accounted for by multiplying the two weighting functions presented in Cases 1 and 2 to generate a comprehensive W_o that can then be used in Method 1. Similar to Cases 3 and 4, I again consider two possible values of θ: $\theta = 3$ and $\theta = 5$. Assuming that the parametric uncertainty is $\pm 50\%$, as in Cases 1 and 4, and that the delay uncertainty is less than or equal to 20 samples, i.e., $\delta_\tau = 20$, a tuning set of $R = I$, $S = (5 \times 10^{-4})\, I$ and $T = (5 \times 10^{-4})\, I$ can be used to achieve some error attenuation while satisfying control objective O2, as shown in Fig. 4.7. Satisfying control objective O3, however,



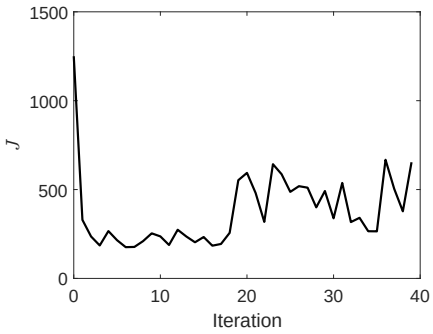
(a) Input signal at different iterations. The dashed lines represent the saturation limit u_{sat}.



(b) Output signal at different iterations.



(c) The norm of the error signal at each iteration. The dashed line represents e^*.



(d) The cost at each iteration.

Figure 4.6. Simulation results for Case 4 - LPV dynamics and parametric uncertainty.

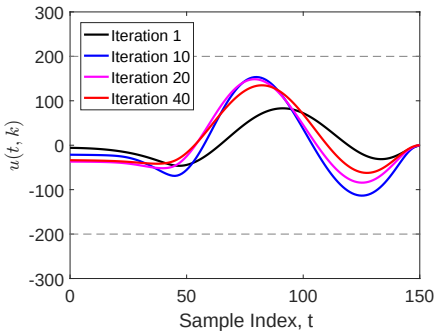
is difficult in this situation because the uncertainty in the delay generates the "wind up" behavior (described earlier as a result of a phase delay between the input and error signals) in the learning algorithm while the magnitude uncertainty affects the sensitivity of the system to the input signal. Because the plant's sensitivity to the input varies from iteration to iteration, the delay uncertainty results in large shifts in the error signal between iterations. The controller bounds the effect on the norm of the error such that $\|e_k\|_2$ is less than e^*, as shown in Fig. 4.7(c), but e^* is not much lower than the initial error norm of the system. To reduce the effect of the variable

sensitivity, the eigenvalues of S should be increased. Increasing the eigenvalues of S, however, causes the the controller to become less reactive to the plant parameter variations, and can also result in large tracking errors.

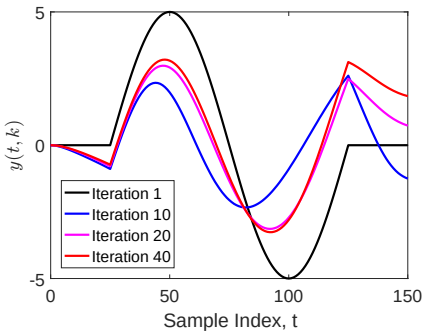
Ultimately, if the uncertainty in the system is large, satisfying all three control objectives may not be possible without reducing the uncertainty in some way. This could include enhancing the LPV model so that it better captures the parameter variations or by adding a more sophisticated delay estimation algorithm. With either of those implemented, it may be possible to use a more aggressive tuning to achieve all three control objectives.

4.4 Chapter Summary

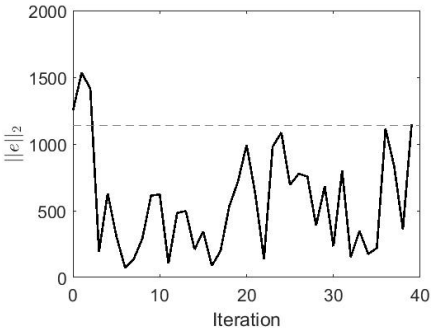
In this chapter I developed a norm optimal ILC algorithm that accounts for LPV dynamics along with parametric and time delay uncertainty. Based on a sufficient condition for robust convergence as well as an upper bound on the norm of the error signal, I constructed a tuning method for the cost function weighting matrices that are used to define the optimal ILC filters. Through a numerical case study, I demonstrated that compensating for parametric and delay uncertainty simultaneously can be difficult. The phase issues that occur due to delay estimation errors generate large input signals that are then applied to a process that experiences varying sensitivity to the input signal due to parameter variations. This can result in large tracking errors that do not satisfy the control objectives. Improving the model to reduce the plant uncertainty would improve the ILC algorithm's performance, but that may not be feasible. An alternative approach for improving the algorithm's performance is defining the necessary condition for robust convergence and using that to tune the ILC algorithm.



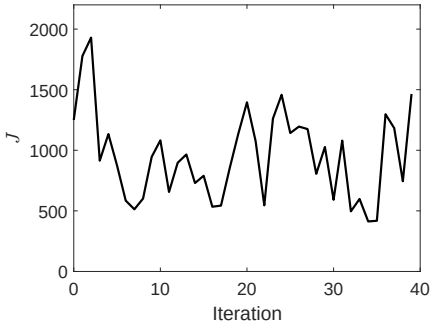
(a) Input signal at different iterations. The dashed lines represent the saturation limit u_{sat}.



(b) Output signal at different iterations.



(c) The norm of the error signal at each iteration. The dashed line represents e^*.



(d) The cost at each iteration.

Figure 4.7. Simulation results for Case 5 - LPV dynamics with parametric and time delay uncertainty.

5. CONCLUSION

5.1 Summary of Research Contributions

Iterative learning control (ILC) is a useful approach for controlling periodic systems, such as many manufacturing processes. While ILC has successfully been applied to many different systems, one major gap in literature was developing an ILC algorithm that is robust to time delay estimation error, especially for systems that have delays longer than one iteration or that are time-varying. One example of such a system is the twin roll strip casting process, where the important outputs of the system, such as the strip thickness, cannot be measured until after a long and variable time delay that spans multiple revolutions of the casting rolls. The variable time delay, as well as the complex dynamics of the process, motivated the design of three algorithms that advance the ILC literature.

The first algorithm I designed addresses the problem of having a delay that is longer than one iteration. I proposed that the delay should be divided into two components: an iterative component that measures the integer number of iterations that occur during the delay and a residual delay that measures the length of the delay remaining after the iterative component is measured. Dividing the delay into two components allowed me to develop separate estimation algorithms for both components. Using the delay components in an ILC algorithm enabled the definition of sufficient conditions for asymptotic stability in the presence of both iterative and residual delays.

To compensate for time-varying delays I proposed a second ILC algorithm that coupled the time delay estimation with the ILC formulation. I defined a sufficient condition for asymptotic stability of the coupled algorithm. Simulation results illustrated that the proposed time delay estimate converges to the true time delay as the

number of iterations goes to infinity, which enables the norm of the process error to converge to its optimal value.

The coupled algorithm, however, depended on having an accurate model of the plant, which is not always realistic. To compensate for plant uncertainty, I developed a third ILC algorithm that is based on the norm optimal ILC formulation. The third algorithm accounts for parametric uncertainty as well as time delay uncertainty. It is also constructed assuming a linear parameter (LPV) plant model, which can be used to more accurately capture some nonlinear behaviors that may otherwise be difficult to capture in a control-oriented model. Based on a sufficient condition for robust convergence as well as an upper bound on the norm of the error signal, I constructed a tuning method for the cost function weighting matrices that are used to define the optimal ILC filters for plant models that feature plant uncertainty, delay uncertainty, and/or LPV dynamics. Through a case study involving these different system features, I demonstrated some of the benefits and disadvantages of using the norm optimal framework particularly for systems that experience all three plant feature simultaneously.

In addition to my theoretical contributions, I also demonstrated the first ILC algorithm on a commercial twin roll strip casting process. After extensive testing, I showed that the ILC algorithm reduced the impact of the strip wedge disturbance by approximately 50 percent compared to measurements obtained before the ILC algorithm was enabled. This level of reduction was achieved despite not knowing a precise model of the system dynamics nor having a way to validate the time delay estimation during the experiments.

5.2 Future Research Directions

The research presented in this dissertation focused on making ILC algorithms robust to time delay estimation errors and parametric uncertainty based on a sufficient condition for stability. The use of a sufficient condition in the formulation and tuning

of the ILC algorithms meant that the tuning sets may be overly conservative. The performance of the algorithm could potentially be improved if a necessary condition for stability can be derived for ILC algorithms operating on systems that feature delay estimation errors or parametric uncertainty.

Another area where a sufficient condition may be limiting the ability to achieve better performance is with the definition of robust convergence. Defining the necessary condition for robust convergence could potentially yield a method for improving the robustness of the ILC algorithm toward both delay and parametric uncertainty.

Future research could also focus on improving the coupled delay estimation and ILC algorithm. The estimation scheme that I proposed is only applicable to systems that have a known model. In cases where there is some model uncertainty, calculating the converged state of the ILC algorithm may not be feasible. A time delay estimation algorithm that does not rely upon knowledge of the exact plant model would expand the class of systems that the coupled ILC and delay estimation algorithm can be applied to.

A final direction for future research could be on how to relax the initial condition resetting assumption used in the LPV ILC algorithm. A relaxed initial condition assumption would expand the application base of the ILC algorithms to more processes that operate in a continuous manner, such as the twin roll strip casting process, where the initial condition of one iteration is the final condition of the previous iteration.

Publications

Peer Reviewed Journal Articles

J1 **F. Browne**, G.T.-C. Chiu, and N. Jain, "A Nonlinear Dynamic Switched-Mode Model of Twin-Roll Steel Strip Casting." *ASME Journal of Dynamic Systems, Measurement and Control*, 2019. doi:10.1115/1.4042952.

J2 **F. Browne**, B. Rees, G.T.-C. Chiu, and N. Jain, "Iterative Learning Control with Time Delay Compensation: An Application to Twin Roll Strip Casting." *IEEE Transactions on Control Systems Technology*, 2020. doi:10.1109/TCST.2020.2971452

Peer Reviewed Conference Papers

C1 **F. Browne**, G.T.-C. Chiu, and N. Jain, "Dynamic Modeling of Twin-Roll Steel Strip Casting." Proceedings of the 2016 ASME Dynamic Systems and Control Conference, Minneapolis, MN, October 12-14, 2016.

C2 **F. Browne**, G.T.-C. Chiu, and N. Jain, "Iterative Learning Control For Periodic Disturbances in Twin-Roll Strip Casting with Measurement Delay." Proceedings of the 2018 American Control Conference, Milwaukee, WI, June 27-29, 2018.

C3 **F. Browne**, B. Rees, G.T.-C. Chiu, and N. Jain, "ILC With Time-Varying Delay Estimation: A Case Study on Twin Roll Strip Casting." Proceedings of the 2018 ASME Dynamic Systems and Control Conference, Atlanta, GA, September 30 - October 3, 2018.

C4 **F. Browne**, G.T.-C. Chiu, and N. Jain, "A Coupled Adaptive Measurement Delay Estimation and Iterative Learning Control Algorithm." Proceedings of the 2019 ASME Dynamic Systems and Control Conference, Park City, UT, October 8-11, 2019.

Patents

P1 **F.M. Browne III**, G.T.C. Chiu, N. Jain, H.B. Rees, "Iterative Learning Control for Periodic Disturbances in Twin-Roll Strip Casting With Measurement Delay." U.S. Patent US20190091761A1.

REFERENCES

REFERENCES

[1] Jean-Pierre Birat, Rolf Steffen, and Stephan Wilmotte. *State of the Art and Developments in Near-Net-Shape Casting of Flat Steel Products: Final Ecsc Report*. Number 16671 in EUR Technical Steel Research - Steelmaking. European Commision, Luxembourg, 1995.

[2] Murray Garden. Learning Control of Actuators in Control Systems, January 1971.

[3] M. Uchiyama. Formation of High Speed Motion Pattern of Mechanical Arm by Trial. *Transactions of the Society of Instrumentation and Control Engineers*, 19(5):706–712, May 1978.

[4] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Iterative Learning Control for Robot Systems. In *Proceedings of IECON*, Tokyo, Japan, October 1984.

[5] Suguru Arimoto, Sadao Kawamura, and Fumio Miyazaki. Bettering Operation of Robots by Learning. *Journal of Robotic Systems*, 1(2):123–140, June 1984.

[6] J.J. Craig. Adaptive Control of Manipulators Through Repeated Trials. In *Proceedings of American Controls Conference*, pages 1566–1573, San Diego, June 1984.

[7] G. Casalino and G. Bartolini. A Learning Procedure for the Control of Movements of Robotic Manipulators. In *IASTED Symposium on Robotics and Automation*, pages 108–111, New Orleans, LA, 1984.

[8] Hyo-Sung Ahn, YangQuan Chen, and Kevin L. Moore. Iterative Learning Control: Brief Survey and Categorization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1099–1121, 2007.

[9] D. A. Bristow, M. Tharayil, and A. G. Alleyne. A Survey of Iterative Learning Control. *IEEE Control Systems*, 26(3):96–114, June 2006.

[10] Kevin L. Moore. *Iterative Learning Control for Deterministic Systems*. Advances in Industrial Control. Springer London, London, 1993.

[11] David H. Owens. *Iterative Learning Control*. Advances in Industrial Control. Springer London, London, 2016.

[12] Y. Chen, M. Sun, B. Huang, and H. Dou. Robust Higher Order Repetitive Learning Control Algorithm for Tracking Control of Delayed Repetitive Systems. In *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, pages 2504–2510 vol.3, December 1992.

[13] D. de Roover. Synthesis of a robust iterative learning controller using an H-Infinity approach. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 3, pages 3044–3049 vol.3, December 1996.

[14] D. Meng, Y. Jia, J. Du, and F. Yu. Robust Design of a Class of Time-Delay Iterative Learning Control Systems With Initial Shifts. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(8):1744–1757, August 2009.

[15] Kira Barton, Sandipan Mishra, and Enric Xargay. Robust Iterative Learning Control: L1 adaptive feedback control in an ILC framework. In *Proceedings of the 2011 American Control Conference*, pages 3663–3668, San Francisco, CA, June 2011. IEEE.

[16] Douglas A. Bristow. Weighting matrix design for robust monotonic convergence in Norm Optimal iterative learning control. In *2008 American Control Conference*, pages 4554–4560, Seattle, WA, June 2008. IEEE.

[17] Kira Barton, Jeroen van de Wijdeven, Andrew Alleyne, Okko Bosgra, and Maarten Steinbuch. Norm optimal Cross-Coupled Iterative Learning Control. In *2008 47th IEEE Conference on Decision and Control*, pages 3020–3025, Cancun, Mexico, 2008. IEEE.

[18] D A Bristow. Optimal iteration-varying Iterative Learning Control for systems with stochastic disturbances. In *Proceedings of the 2010 American Control Conference*, pages 1296–1301, Baltimore, MD, June 2010. IEEE.

[19] Kira L. Barton and Andrew G. Alleyne. A Norm Optimal Approach to Time-Varying ILC With Application to a Multi-Axis Robotic Testbed. *IEEE Transactions on Control Systems Technology*, 19(1):166–180, January 2011.

[20] Berk Altın, Jeroen Willems, Tom Oomen, and Kira Barton. Iterative Learning Control of Iteration-Varying Systems via Robust Update Laws with Experimental Implementation. *Control Engineering Practice*, 62:36–45, May 2017.

[21] M. Butcher and A. Karimi. Linear Parameter-Varying Iterative Learning Control With Application to a Linear Motor System. *IEEE/ASME Transactions on Mechatronics*, 15(3):412–420, June 2010.

[22] D. J. Hoelzle and K. Barton. Flexible Iterative Learning Control Using a Library Based Interpolation Scheme. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3978–3984, December 2012.

[23] D. Huang, J. Xu, V. Venkataramanan, and T. C. T. Huynh. High-Performance Tracking of Piezoelectric Positioning Stage Using Current-Cycle Iterative Learning Control With Gain Scheduling. *IEEE Transactions on Industrial Electronics*, 61(2):1085–1098, February 2014.

[24] R. de Rozario, T. Oomen, and M. Steinbuch. Iterative Learning Control and feedforward for LPV systems: Applied to a position-dependent motion system. In *2017 American Control Conference (ACC)*, pages 3518–3523, May 2017.

[25] D. J. Leith and W. E. Leithead. Survey of Gain-Scheduling Analysis and Design. *International Journal of Control*, 73(11):1001–1025, January 2000.

[26] C. Hoffmann and H. Werner. A Survey of Linear Parameter-Varying Control Applications Validated by Experiments or High-Fidelity Simulations. *IEEE Transactions on Control Systems Technology*, 23(2):416–433, March 2015.

[27] L. M. Hideg. Stability and Convergence Issues in Iterative Learning Control: Part II. In *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, pages 480–485, September 1996.

[28] L. M. Hideg. Time Delays in Iterative Learning Control Schemes. In *Proceedings of Tenth International Symposium on Intelligent Control*, pages 215–220, August 1995.

[29] E. Rogers and D.H. Owens. On the Stability of Linear Repetitive Processes Described by a Delay-Difference Equation. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 51(7):359–363, July 2004.

[30] K.-H. Park, Zeungnam Bien, and D.-H. Hwang. Design of an Iterative Learning Controller for a Class of Linear Dynamic Systems with Time Delay. *IEE Proceedings - Control Theory and Applications*, 145(6):507–512, 1998.

[31] Yangquan Chen, Zhiming Gong, and Changyun Wen. Analysis of a High-Order Iterative Learning Control Algorithm for Uncertain Nonlinear Systems with State Delays. *Automatica*, 34(3):345–353, 1998.

[32] Xiao-Dong Li, T. W. S. Chow, and J. K. L. Ho. 2-D System Theory Based Iterative Learning Control for Linear Continuous Systems with Time Delays. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(7):1421–1430, July 2005.

[33] Z. Bien and K. M. Huh. Higher-Order Iterative Learning Control Algorithm. *IEE Proceedings D - Control Theory and Applications*, 136(3):105–112, May 1989.

[34] C. Knapp and G. Carter. The Generalized Correlation Method for Estimation of Time Delay. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(4):320–327, August 1976.

[35] G. C. Carter. Coherence and Time Delay Estimation. *Proceedings of the IEEE*, 75(2):236–255, February 1987.

[36] A. Cabasson and O. Meste. Time Delay Estimation: A New Insight Into the Woody's Method. *IEEE Signal Processing Letters*, 15:573–576, 2008.

[37] Florian Browne, George T.-C. Chiu, and Neera Jain. A Nonlinear Dynamic Switched-Mode Model of Twin-Roll Steel Strip Casting. *Journal of Dynamic Systems, Measurement, and Control*, 141(8):081004, August 2019.

[38] R. I. L. Guthrie and M. M. Isac. Conventional and near net shape casting options for steel sheet †. *Ironmaking & Steelmaking*, 43(9):650–658, October 2016.

[39] T. Matsushita, K. Nakayama, H. Fukase, and S. Osada. Development and Commercialization of Twin Roll Strip Caster. *IHI Engineering Review*, 42(1):1–9, 2009.

[40] C. R. Killmore, H. Creely, A. Phillips, H. Kaul, P. Campbell, M. Schueren, J. G. Williams, and W. Blejde. Development of Ultra-Thin Cast Strip Products by the CASTRIP® Process. *Materials Forum*, 32:13–28, 2007.

[41] John B. Edwards and Alberto Cavazos. Interaction Analysis of the Twin-Roller Strip-Casting Process and the Implications for Process Control. *Journal of Materials Engineering and Performance*, 14(3):395–407, 2005.

[42] S. Bernhard, M. Enning, and H. Rake. Automation of a Laboratory Plant for Direct Casting of Thin Steel Strips. *Control Engineering Practice*, 2(6):961–967, 1994.

[43] Wenyu Zhang, Dongying Ju, Hongyang Zhao, Xiaodong Hu, Yao Yao, and Yujun Zhang. A Decoupling Control Model on Perturbation Method for Twin-Roll Casting Magnesium Alloy Sheet. *Journal of Materials Science & Technology*, 31(5):517–522, May 2015.

[44] Keum-Shik Hong, Jeom-Goo Kim, and Masayoshi Tomizuka. Control of Strip Casting Process: Decentralization and Optimal Roll Force Control. *Control Engineering Practice*, 9(9):933–945, 2001.

[45] A Hadadzadeh and M A Wells. Thermal fluid mathematical modelling of twin roll casting (TRC) process for AZ31 magnesium alloy. *International Journal of Cast Metals Research*, 26(4):228–238, August 2013.

[46] Jian Zeng, Roger Koitzsch, Herbert Pfeifer, and Bernd Friedrich. Numerical Simulation of the Twin-Roll Casting Process of Magnesium Alloy Strip. *Journal of Materials Processing Technology*, 209(5):2321–2328, March 2009.

[47] C. A. Santos, J. A. Spim, and A. Garcia. Modeling of Solidification in Twin-Roll Strip Casting. *Journal of Materials Processing Technology*, 102(1):33–39, 2000.

[48] Lianlian Liu, Bo Liao, Jing Guo, Ligang Liu, Hongyan Hu, Yue Zhang, and Qingxiang Yang. 3D Numerical Simulation on Thermal Flow Coupling Field of Stainless Steel During Twin-Roll Casting. *Journal of Materials Engineering and Performance*, 23(1):39–48, January 2014.

[49] R. I. L Guthrie and R. P Tavares. Mathematical and physical modelling of steel flow and solidification in twin-roll/horizontal belt thin-strip casting machines. *Applied Mathematical Modelling*, 22(11):851–872, November 1998.

[50] Q. Li, Y. K. Zhang, L. G. Liu, P. Zhang, Y. Zhang, Y. Fang, and Q. X. Yang. Effect of Casting Parameters on the Freezing Point Position of the 304 Stainless Steel During Twin-Roll Strip Casting Process by Numerical Simulation. *Journal of Materials Science*, 47(9):3953–3960, May 2012.

[51] Manish Gupta and Yogeshwar Sahai. Mathematical Modeling of Fluid Flow, Heat Transfer, and Solidification in Two-roll Melt Drag Thin Strip Casting of Steel. *ISIJ International*, 40(2):144–152, 2000.

[52] Amit Saxena and Yogeshwar Sahai. Modeling of Fluid Flow and Heat Transfer in Twin-Roll Casting of Aluminum Alloys. *MATERIALS TRANSACTIONS*, 43(2):206–213, 2002.

[53] Peter Woodberry, Wai Yee Daniel Yuen, and Nikolco Nikolovski. Method of Operation of Twin Roll Strip Caster to Reduce Chatter, October 2017.

[54] Miroslaw Glowacki. *Inverse Analysis Applied to Mushy Steel Rheological Properties Testing Using Hybrid Numerical-Analytical Model*. INTECH Open Access Publisher, 2012.

[55] Florian Browne, George Chiu, and Neera Jain. Dynamic Modeling of Twin-Roll Steel Strip Casting. In *Proceedings of the ASME 2016 Dynamic Systems and Control Conference*, Minneapolis, MN, 2016. American Society of Mechanical Engineers.

[56] Eric W. Grald and J.Ward MacArthur. A Moving-Boundary Formulation for Modeling Time-Dependent Two-Phase Flows. *International Journal of Heat and Fluid Flow*, 13(3):266–272, September 1992.

[57] G. L. Wedekind and W. F. Stoecker. Theoretical Model for Predicting the Transient Response of the Mixture-Vapor Transition Point in Horizontal Evaporating Flow. *Journal of Heat Transfer*, 90(1):165–174, 1968.

[58] Pilvi Oksman, Shan Yu, Heli Kytönen, and Seppo Louhenkilpi. The Effective Thermal Conductivity Method in Continuous Casting of Steel. *Acta Polytechnica Hungarica*, 11(9):6–22, 2014.

[59] V. R. Voller and C. R. Swaminathan. ERAL Source-Based Method for Solidification Phase Change. *Numerical Heat Transfer, Part B: Fundamentals*, 19(2):175–189, January 1991.

[60] H Fukase, S Osada, and H Otsuka. Solidification Model for Steel Strip Casting. *Iron and Steelmaker*, 30(7):48–58, 2003.

[61] Walter N. Blejde, Rama Ballav Mahapatra, and Mark Schlichting. Method and Apparatus for Controlling Strip Temperature Rebound in Cast Strip, March 2011.

[62] Eugene A Mizikar. Mathematical Heat Transfer Model For Solidification of Continuously Cast Steel Slabs. *Transactions of the Metallurgical Society of AIME*, 239(11):1747–1758, November 1967.

[63] Tom M. Apostol and Mamikon A. Mnatsakanian. *New Horizons in Geometry*. Number 47 in Dolciani Mathematical Expositions. Mathematical Association of America, Washington, D.C., 2012.

[64] Florian Browne, Brad Rees, George Chiu, and Neera Jain. ILC with Time-Varying Delay Estimation: A Case Study on Twin Roll Strip Casting. In *Proceedings of the ASME 2018 Dynamic Systems and Control Conference*, Atlanta, GA, October 2018.

[65] Florian Browne, Brad Rees, George T.C. Chiu, and Neera Jain. Iterative Learning Control With Time Delay Compensation: An Application to Twin Roll Strip Casting. *IEEE Transactions on Control Systems Technology*, pages 1–10, February 2020.

[66] Florian Browne, George T.-C. Chiu, and Neera Jain. A Coupled Adaptive Measurement Delay Estimation and Iterative Learning Control Algorithm. In *Proceedings of the ASME 2019 Dynamic Systems and Control Conference*, page 8, Park City, UT, 2019. American Society of Mechanical Engineers.

[67] Mikael Norrlöf and Svante Gunnarsson. Time and Frequency Domain Convergence Properties in Iterative Learning Control. *International Journal of Control*, 75(14):1114–1126, January 2002.

[68] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, Inc., New York, NY, USA, 3rd edition, 1999.

[69] Hassan K. Khalil. *Nonlinear Systems*, volume 3. Pearson, Upper Saddle River, NJ, 2002.

[70] Hongbin Wang, Le Zhou, Yongwen Zhang, Yuanhua Cai, and Jishan Zhang. Effects of Twin-Roll Casting Process Parameters on the Microstructure and Sheet Metal Forming Behavior of 7050 Aluminum Alloy. *Journal of Materials Processing Technology*, 233:186–191, July 2016.

[71] ZI-QIN WANG, PETTER LUNDSTRÖM, and SIGURD SKOGESTAD. Representation of uncertain time delays in the H∞ framework. *International Journal of Control*, 59(3):627–638, March 1994. _eprint: https://doi.org/10.1080/00207179408923097.

VITA

VITA

Florian (Rian) Browne III was born in Topeka, KS in 1992. He attended Shawnee Heights High School in nearby Tecumseh KS, graduating in 2011. He then became a Wildcat and enrolled at Kansas State University. He graduated *summa cum laude* with a Bachelor of Science degree in mechanical engineering in 2015. During his junior year at K-State, he spent a semester working as a control theory co-op at Fisher Controls in Marshalltown, IA. That motivated him to pursue graduate school and he enrolled as a direct Ph.D. student in the school of mechanical engineering at Purdue University in 2015. Under the advisement of Prof. Neera Jain, his studies have focused on dynamic modeling and control of advanced manufacturing systems and iterative learning control.