Controller Gain Optimization for Position Control of an SMA Wire

by

Roger Chor Chun Chau

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Controller Gain Optimization for Position Control of an SMA Wire

There has been an increasing interest in the field of 'smart structures' and 'smart materials'. In constructing smart structures, a class of materials called smart materials are often used as sensors and actuators. An example of a smart material is shape memory alloy (SMA). A common actuator configuration uses an SMA wire with a constant load. The non-linear input-output behaviour of SMAs, known as hysteresis, made them difficult to model and control.

The research in this thesis examines the effect of PID-controller gain optimization on SMA wire control at different frequencies of operation. A constant-load SMA wire actuator with a PID-controller is used in the study. Heat is applied to the wire using an input electric current. The system is cooled through convection with the surrounding area. The lack of active cooling prevents the system from operating at high frequencies.

Three different cost functions are proposed for various applications. The Preisach model is chosen to model the hysteretic behaviour of the SMA wire contraction. Varying material properties such as electrical resistance and heat capacities are modelled to give a more accurate representation of the system's physical behaviour. Simulations show that by optimizing the controller gain values, the bandwidth of the system is improved.

An interesting observation is made in the heating cycle of the SMA wire. In order to achieve faster cooling, overshoot is observed at low frequencies. This is a result of the system hysteresis. The system hysteresis allows different input signals to achieve the same output value. Since the rate of cooling is proportional to the temperature above ambient, better cooling is achieved by reaching a higher temperature. The error caused by the overshoot is compensated by the better cooling phase, which is not actively controlled.

Acknowledgements

I would like to take this opportunity to thank all the people who have provided help and support during the two years of this research. Without any one of them, none of this would have been possible.

First I would like to thank my supervisors, Professors Kirsten Morris and Robert Gorbet. They have provided constant support and guidance in every stage of this work. Their vast knowledge has been an invaluable resource for me. I would also like to thank them for enduring a very tight timetable despite their own busy schedules.

In the past two years, I shared my office with Miss Lijun Wang. I would like to thank her for showing me around the maze that is otherwise known as the Mathematics and Computer Science building. Bernard Chan, fellow math enthusiast, provided technical expertise in LaTeX coding and unrivaled interest in everything related to math. The friendships of Danny Mak, Fion Tong and the rest of the Vickers FC family helped me keep a healthy balance between work and play.

I would like to thank my family for their unconditional love and support in every aspect of my life. My sisters have been great friends and role models for me. Although I do not get to see my father very often, his words of wisdom and sense of humour helped me keep a positive attitude to face the different challenges in life. I am very grateful to my mother, who sacrificed valuable time with her husband to take care of me and my sisters in Canada for our education and development.

Last but certainly not least, I would like to thank my girlfriend Vivian for her love and encouragement. She took great care of me and was a very good listener during this very stressful period. Her support gave me the strength to complete this task which I once thought was impossible.

My sincere thanks once again to everyone for their kind support during my two years at Waterloo.

To my family and Vivian

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There has been an increasing interest in the field of 'smart structures' and 'smart materials'. Smart structures are designed to adapt to changes in their surrounding environment, such as vibrations or airflow, by adjusting their shape or other physical properties such as stiffness. In constructing smart structures, a class of materials called smart materials are often used as sensors and actuators. An example of a smart material is shape memory alloy (SMA). As the name suggests, SMAs possess the ability to 'remember' an undeformed shape. An SMA wire under load contracts when it is heated, and returns to its initial length upon cooling. A commonly used shape memory alloy is a nickel-titanium alloy called nitinol. We will concentrate on the application of SMA wires as actuators.

There are many advantages to using SMAs in actuator applications. It is shown in [27] that SMAs have a very high power to weight ratio among common actuators. Power to weight ratio is the amount of power generated per kilogram of actuator. Motors tends to yield high power, but at the same time they can be very heavy. The most intriguing finding is that even at very small scales, SMAs maintain a high power to weight ratio. This is useful if weight is a concern in the structure. For example, in the three-link robotic arm discussed in [1], the inertia and moments that the SMA actuators add to the dynamics of the arm are found to be negligible due to their light weight.

SMA actuators are also very simple and quiet. This is because the actual material can be used for actuation without addition of mechanical parts. The simplest SMA actuator is a constant-load SMA wire. Upon heating, the wire goes through a phase change and

contracts due to an associated change in material modulus. As it cools, the wire returns to its original length. Since heating can be done by applying electric current through the wire, electric power is converted to do mechanical work using the material alone.

Despite the advantages mentioned above, there are drawbacks to using SMAs for various applications. The phase change in SMAs exhibits a highly nonlinear behaviour called hysteresis. Hysteresis of these materials makes them very difficult to model. Hysteresis models can be divided into two categories: physical and phenomenological. Physical models relate the physical properties of the material such as internal energy with the hysteretic behaviour. Phenomenological models produce behaviours similar to the physical system, but the parameters used in the models do not necessarily have any physical meaning. Hence it is often difficult to identify these parameters experimentally.

The phase transformation involves heating and cooling the material to different temperatures. In most applications, heating is done by applying current through the wire. The cooling of SMAs can be difficult in practice since active cooling mechanisms will add more weight and complexity to the actuator. Cooling by heat convection to the surrounding environment is slow and this prevents the actuators from operating at high frequencies. This problem is reduced in situations where the system is operating in high altitudes, as in airplane wings and helicopter blades, due to lower ambient temperature and air flow. Underwater applications also allow for more efficient cooling as discussed in the review by Seelecke and Müller [49].

1.1 Thesis Goals

In the case where active cooling methods are not used, the slow cooling of the SMA material limits the frequencies of operation. Hence, it is worthwhile to study the control of SMA wires without active cooling mechanisms. The goal of this work is to study a closed-loop position control system: an SMA wire under constant load with a proportional-integral-derivative(PID) controller. The wire contraction is required to reach specified periodic target values. Optimal control is applied to try to improve the cooling time by heating the wire 'just enough', allowing the system to operate at higher frequencies.

1.2 Outline

This work is organized as follows. Chapter 2 provides background on modeling shape memory alloys. Actuator designs using SMAs are discussed. Different modelling approaches are mentioned. A state-space representation for the most common model, the Preisach model is described. Model identification using experimental data is briefly discussed.

A summary of research on the control of shape memory alloys is given in Chapter 3. A brief background on dissipativity theory is presented. Results on the dissipativity of the Preisach model are given and the application to control discussed. Optimal control strategies in various smart structures applications are discussed. An example of an inverse model used to reduce hysteretic nonlinearity is given.

Chapter 4 provides the framework for the optimal control problem. The control objectives using the SMA wire actuator are described. An attempt to derive a derivative for the Preisach model is given. Different optimization algorithms that do not require derivative information are investigated. The Nelder-Mead simplex algorithm for optimization is chosen and described in the chapter.

In Chapter 5, the numerical implementation of the optimal control problem is discussed. Different numerical methods are compared to reduce the simulation time of the model. A number of cost functions are considered for optimization. The numerical results are presented and discussed in Chapter 6. The final chapter describes the contributions of this work and future research directions.

Chapter 2

SMA Modelling

This chapter provides background on modeling shape memory alloys. A physical description of the SMA phase transformation is given. A state-space representation for the Preisach model is described. Model identification using experimental data is briefly discussed. It provides the foundation for simulating the SMA wire behaviour in the control problems discussed later.

2.1 Physical Behaviour

Hysteresis is a phenomenon that appears in a variety of ferromagnetic and electromagnetic materials. For static hysteretic systems, the output only depends on the past input extrema, and not the rate at which the input is applied. Static hysteresis is also called *rate-independent* hysteresis. Assume we have two inputs $u_1(t)$ and $u_2(t)$ having the same maximum and minimum values, but the signals are not necessarily identical. Then, a static hysteretic system will produce identical input-output graphs for the same initial conditions.

Hysteretic behaviour is observed in smart materials, including SMAs. A commonly used shape memory alloy is a nickel-titanium alloy called nitinol. As temperature changes, nitinol goes through a continuous phase change between martensite and austenite crystalline phases.

At low temperatures, the alloy is in full martensite state. As the material is heated, fractions of the alloy become austenite. The percentage of the alloy that is in the austenite

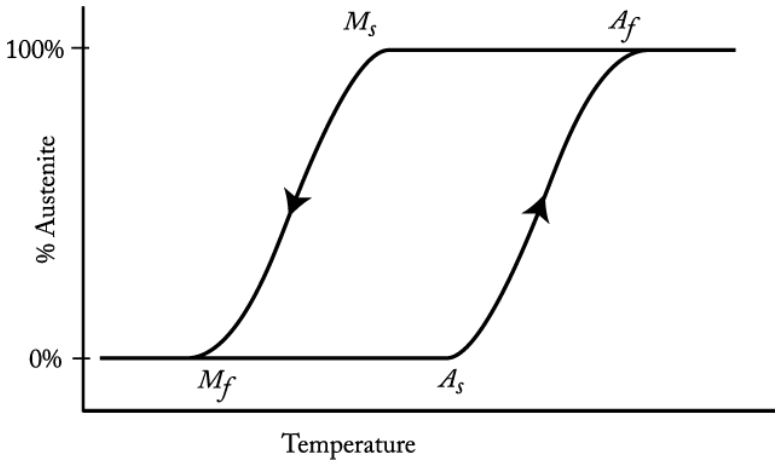

Figure 2.1: SMA Phase Transition Hysteresis Loop

state is called the austenite phase fraction. The phase transformation is characterized by a hysteresis loop as in Figure 2.1. As the alloy is heated, it changes from full martensite to austenite continuously between temperatures A_s and A_f. Similarly, it changes from full austenite to martensite between temperatures M_s and M_f during cooling.

During the phase transformation, the Young's modulus of the material changes as a function of phase fraction. Consider a wire with length l with a constant load of mass m, which corresponds to fixed strain at room temperature (see Figure 2.2). When the wire is heated, it goes through a phase change from martensite to austenite. The phase change results in an increase in stiffness, and hence reduced strain. On the other hand, when the wire is cooled, it goes through a phase change from austenite to martensite. The opposite happens and as the stiffness of the wire decreases, strain is increased.

This phase transformation process provides a way to convert electricity into mechanical work. Besides the constant load actuator of Figure 2.2, there are other actuator configurations using SMAs. In [21, 22], a series of springs made of SMAs are used to control vibrations for a rotor-bearing system. By heating the individual springs, the authors are able to change the spring constants and alter the damping of the overall system. SMAs are also used in making adaptive tuned vibration absorbers (ATVA) [47]. Heating or cooling of the SMA changes its stiffness and thus tunes the frequency of the ATVA.

In [17], a constant-load wire similar to Figure 2.2 is used and stability results on PI-

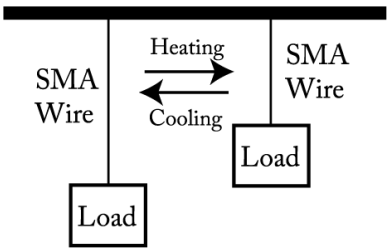

Figure 2.2: Constant-load SMA Wire Actuator

controllers are given. The modelling and control of two types of SMA actuators are discussed in [36]. The first is a differential type with two SMA wires. One wire is heated while the other cools and the contraction of these wires create the differential mechanism. Another actuator is a bias type with an SMA wire and a bias spring. The bias spring acts as reverse actuation to the wire during cooling.

Torsional actuators using SMAs are discussed in [45]. An SMA rod or tube is pre-twisted and connected to a torsional spring. As the temperature increases, the actuator attempts to return to its pre-twisted configuration, and thus it applies a torque to the torsional spring. This type of actuator is used to change the rotor twist of tilt-rotor aircrafts to accommodate both hover and forward flight modes.

The constant-load actuator will be studied in this work. The position of the mass can be measured by the wire contraction. Controlling the wire contraction means controlling the material phase fraction. To control the phase fraction, the temperature of the wire needs to be controlled. In the actuator setup, temperature is controlled through an applied electric current. A good simulation model for the system's hysteretic behaviour is needed for the controller gains optimization.

We first take a look at the simplest case of hysteretic systems, a relay. The simple relay can be found, for example, in valves where a certain threshold needs to be reached before the valve is turned on or off. To illustrate the memory of the hysteresis, we use a simple relay $\gamma_{1,0}$ that is centred at 0 with half-width 1 and output values of -1 and +1 as

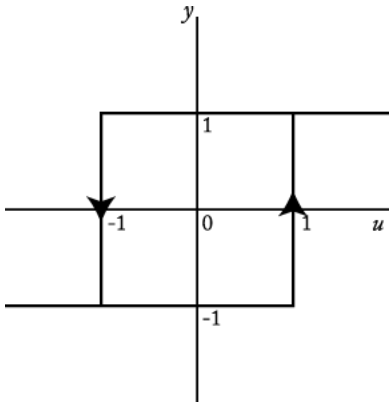

Figure 2.3: A Simple Relay

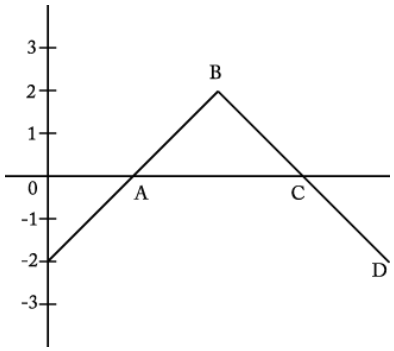

Figure 2.4: Relay Input

illustrated in Figure 2.3. The input-output behaviour of the relay can be defined as:

$$[\gamma_{1,0}u(t)] = y(t) = \begin{cases} 1, & \text{if } u \geq 1 \\ y(t-\epsilon), & \text{if } -1 < u < 1 \\ -1, & \text{if } u \leq -1 \end{cases} \tag{2.1}$$

for small $\epsilon > 0$.

Suppose the input signal $u(t)$ is as shown in Figure 2.4. Starting at $t = 0$, the input is at $u = -2$, therefore the output is at $y = -1$. As we increase the input signal up to point A in Figure 2.4, the output will remain at $y = -1$. Once the input passes $u = 1$, the output value will switch to $y = +1$. At point B, the output of the system would have

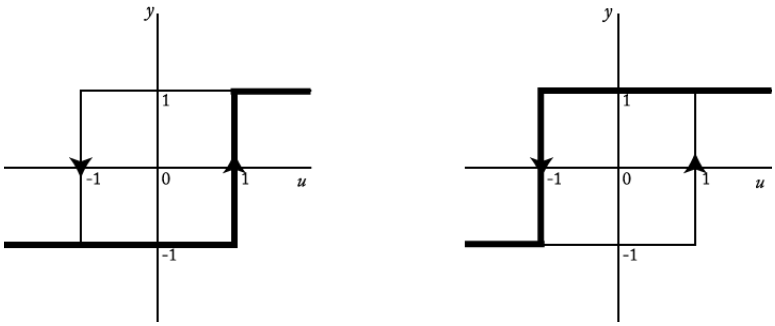

Figure 2.5: Relay Output

followed the path shown on the left in Figure 2.5. If we now decrease the output to point C, since the system has memory, the output remains at $y = +1$. Once the input decreases past $u = -1$, the output will also switch to $y = -1$. Arriving at point D, the output would have followed the path shown on the right in Figure 2.5.

Hysteretic systems are classified by the type of memory they exhibit. A *local memory* hysteretic system's future output only depends on the current output and current input values. The simple relay in the previous example is a local memory hysteretic system. A *non-local memory* hysteretic system's future output depends on current output and the history of input extrema.

The non-local memory of the hysteresis in SMAs creates branches and minor loops inside the main hysteresis loop. Figure 2.6 shows an example of an input-output map of a sample hysteretic system. The *major loop* bounds the region between the input and output saturation values. *Branches* are created as the input changes. New branches are created only when an input reversal occurs. Successive branches inside the major loop may cross to create *minor loops*.

2.2 SMA Modelling

There are numerous models used to describe hysteretic systems. The Preisach model [38] was originally used to describe hysteresis in magnetic materials, but the generality of the model made it a suitable candidate for modelling smart materials as well. The mathe-

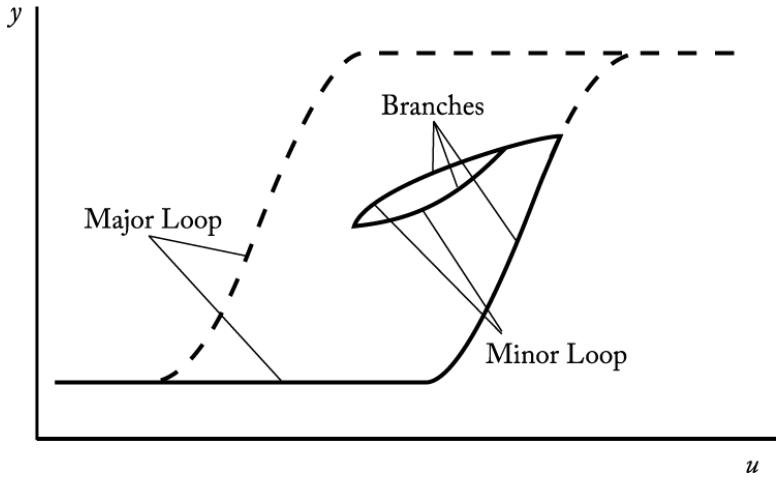

Figure 2.6: Branches and Loops in Hysteresis

matical models provide the tools for analysis, but often contain parameters that have no physical meaning. A dynamic hysteresis model presented in [46] divides the hysteresis into static and eddy current components. Simulation results show that the model agrees with experimental measurements. A free energy model using Helmholtz and Gibbs free energy relations is described in [51] for ferroelectric materials by looking at energies in the lattice level.

In 1986, Jiles and Atherton [30] presented a mathematical model of the hysteresis mechanism in ferromagnets. The model is based on the idea of *domain wall* movements. The domain walls separate regions of the material of different polarity. When an external magnetic field is applied, the interactions between different domains cause the domain walls to move, and thus the magnetization of the overall material changes. The model can be extended to other materials with hysteresis. In SMAs for example, the two 'polarities' can be thought of as the martensite and austenite phases. A discussion on determining the parameters for the Jiles-Atherton model is given in [43].

For this work, the Preisach model is chosen over the other models because it is very easy to implement. The Preisach model is able to reproduce minor loops inside the main hysteresis loop. This is due to the model's ability to model non-local memory hysteresis. The Preisach model simulation output has been shown to accurately reproduce experimen-

tal data. Furthermore, the inverse of the Preisach model can be calculated using different numerical methods, for example [28]. The model inverse is useful in reducing the effect of hysteresis in control applications and is discussed later

In many cases, determining the parameters for a model can be very difficult. Model identification and implementation of the Preisach model can be found in [38]. Mathematical properties of the Preisach operator are discussed in [5]. In addition to its mathematical properties, the Preisach model is based on the physical structure of magnetic materials. A physical interpretation is often lacking in models that only try to fit the input-output map of hysteresis using arbitrary equations. Furthermore, the Preisach model has been shown to be suited for piezoceramic and shape memory alloy representation [26]. Because of this generality, findings based on the Preisach model can be extended to other materials that exhibit static hysteretic behaviour.

There are three steps to modelling electrically heated SMA wire actuators: converting current to temperature, temperature to phase fraction and phase fraction to the output y (cf. Figure 2.7). Temperature above ambient is the input to the Preisach model, but since we cannot change temperature directly, an electric current is applied to the wire. The heating model is used to convert input current to temperature.

The change in temperature causes a change in the phase fraction of the alloy. The Preisach model structure captures the static hysteresis of the phase transition. The model is identified through experimental data of the observable output y. Because the phase fraction is not observable, the relationship between the phase transition and the output is often incorporated in the model identification process. Since the Preisach model reproduces only static hysteresis, this can only be done if the output depends only on the phase fraction and no other dynamics are involved. This is the case for the constant load actuator of Figure 2.2 investigated in this work and described in more detail later.

2.3 Preisach Model

In this section, we will look at the *Preisach model* and its state-space representation. We will follow the formulation used in [15]. The input-output map of SMA wire studied in this work, using the experimental data obtained in [14], is shown in Figure 2.8. The input

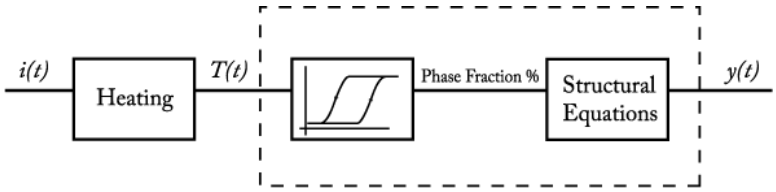

Figure 2.7: Preisach Model Block Diagram

is the temperature in degrees Celsius above ambient and the output is wire contraction measured in millimetres. The Preisach model reconstructs this relationship as a weighted sum of relays. Each of these individual relays, $\gamma_{r,s}$, is characterized by the input offset s and half-width $r > 0$ and has output of +1 or -1 (Figure 2.9). The *weighting function* is denoted by $\mu(r, s)$, and the output at time t is given by the following weighted sum of the relays

$$y(t) = \int_{-\infty}^{\infty} \int_{0}^{\infty} \mu(r, s)[\gamma_{r,s} u(t)] dr ds \tag{2.2}$$

where $[\gamma_{r,s} u(t)]$ denotes the output of the relay $\gamma_{r,s}$ subject to input $u(t)$.

2.3.1 The Preisach Plane

Each of the relays used in the Preisach Model can be described by a point in the (r, s)-plane. This plane is the domain $\mathcal{P}$ for the weighting function μ. Limitations exist for the input signal $u(t)$ for any physical system. One common limitation is input saturation where the signal $u(t)$ lies in the range $[-u_{sat}, u_{sat}]$. Input saturation can be a result of the electrical limitation of the power supply. More importantly, in repeated applications, overheating of the wire will damage its memory and thus shortens the wire's lifetime.

This means that not all the relays in $\mathcal{P}$ will be used. The restricted domain, called the *Preisach Plane*, is a triangle in $\mathcal{P}$ given by $\mathcal{P}_r = \{(r, s)\, s.t.\, |s| \leq u_{sat} - r\}$ (cf. Figure 2.10). The weighting function can be assumed to be zero outside the Preisach plane, hence μ has compact support.

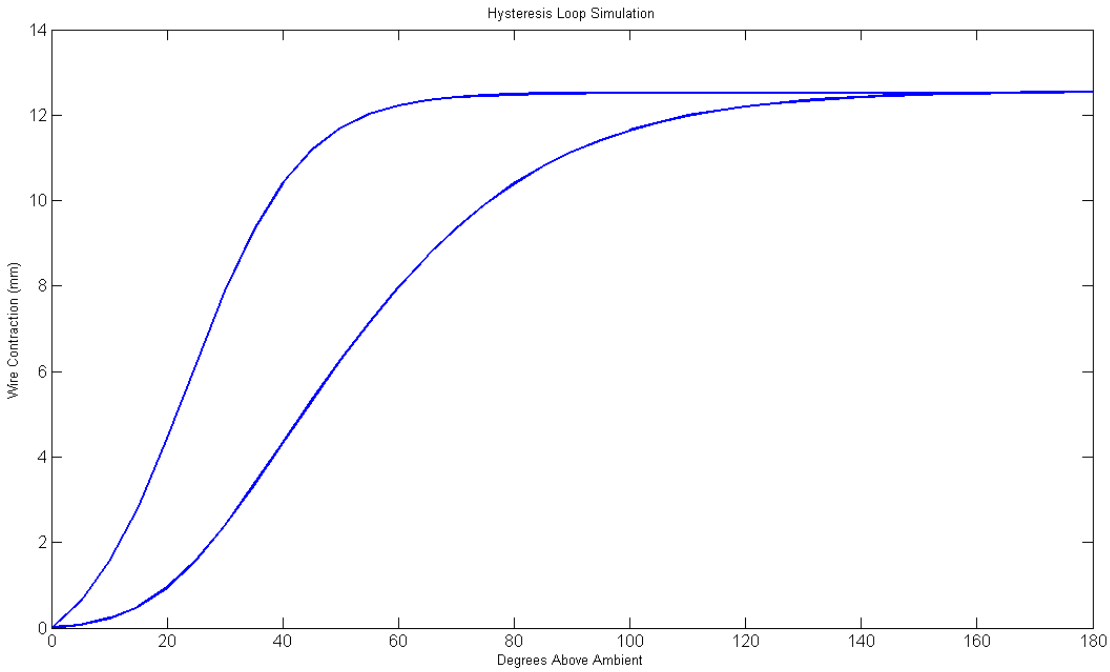

Figure 2.8: Hysteresis Loop

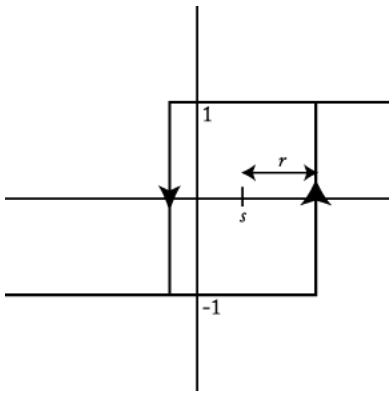

Figure 2.9: A Relay of Centre s and Half-Width r

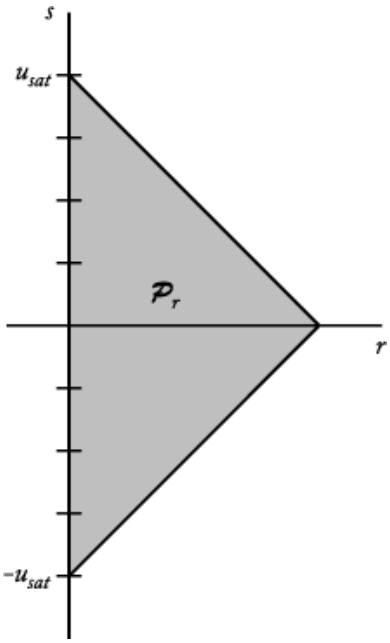

Figure 2.10: The Preisach Plane

2.3.2 Boundary and Initial State

Recall that the system behaviour is dependent on input history, thus an initial state of the relays is required. The weighting functions of magnetic materials are symmetric about the origin, so a logical choice of the initial state is the boundary defined by $s = 0$. This boundary ψ^* represents relays with output of +1 if $s < 0$ and -1 if $s > 0$. In SMAs however, the hysteresis is not symmetric about the origin, as shown previously in Figure 2.1.

Since the input to the SMA wire is temperature above ambient, which is always positive without active cooling, the input range is shifted down to accommodate the symmetric nature of the Preisach model. Given a temperature range from 0 to $temp_{max}$ degrees above ambient, $u = 0$ corresponds to $\frac{temp_{max}}{2}$ degrees above room temperature. $-u_{sat}$ and u_{sat} corresponds to room temperature and $temp_{max}$ degrees above ambient respectively. The boundary ψ^*_{SMA} defined by the line $s = -u_{sat} + r$ is chosen as an initial boundary. This represents the system in negative saturation, or full martensite for the SMA wire at room temperature.

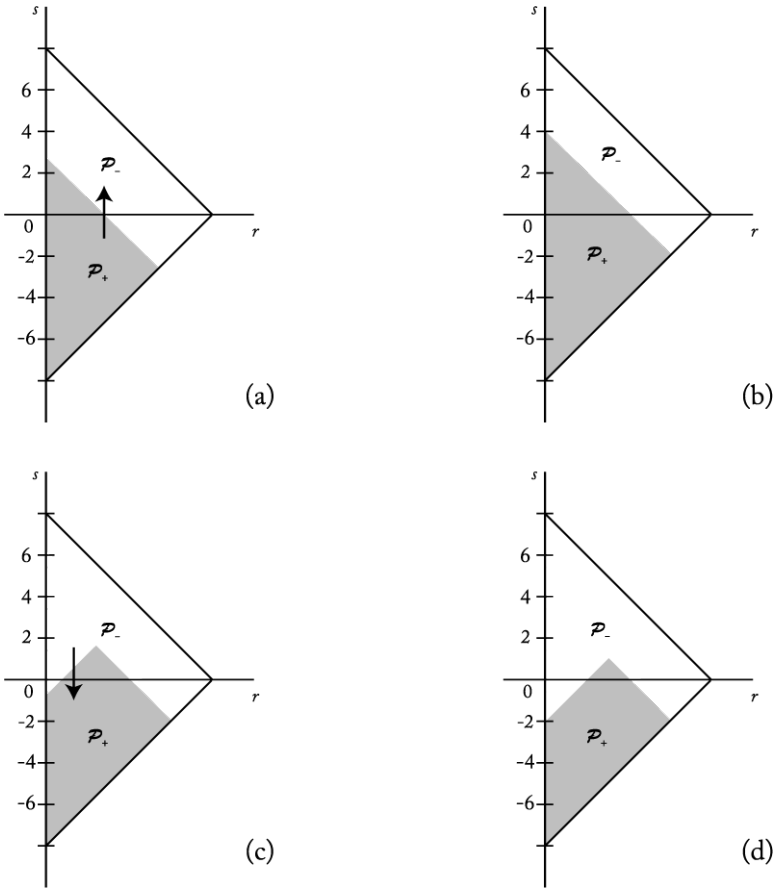

Figure 2.11: Boundary Evolution

The Preisach plane can be divided into two sets, $\mathcal{P}_+(t)$ and $\mathcal{P}_-(t)$, corresponding to the relays that have output +1 and -1 at time t respectively. One can describe the changes of $\mathcal{P}_+(t)$ and $\mathcal{P}_-(t)$ by following the evolution of the boundary, denoted $\psi(t)$. By the definition of the initial boundary ψ^*_{SMA}, the set $\mathcal{P}_+(t)$ is empty and $\mathcal{P}_-(t) = \mathcal{P}_r$ initially.

We will look at an example to illustrate the evolution of the Preisach plane and its boundary, as shown in Figure 2.11. Assume that the input $u(t)$ starts at negative saturation and increases to $u = 4$ monotonically. As our input increases, relays that satisfy $u = s + r$ switch from an output of -1 to +1 (Fig 2.11a). The part of $\mathcal{P}_-(t)$ that lies below the line $s = u - r$ becomes a part of $\mathcal{P}_+(t)$ as a result. $\mathcal{P}_+(t)$ grows until a segment of the

boundary reaches $s = 4 - r$ (Fig 2.11b). If we now decrease the input from $u = 4$ to $u = -2$ monotonically, relays that satisfy $u = s - r$ switch from an output of +1 to -1. The part of $\mathcal{P}_+(t)$ that lies above the line $s = u + r$ now becomes a part of $\mathcal{P}_-(t)$ (Fig 2.11c). $\mathcal{P}_-(t)$ grows until a segment of the boundary reaches $s = -2 + r$ (Fig 2.11d).

Since the relays in the sets $\mathcal{P}_+(t)$ and $\mathcal{P}_-(t)$ have output values of +1 and -1 respectively, the output function of the Preisach model in equation (2.2) can be rewritten as:

$$\begin{aligned} y(t) &= \iint_{\mathcal{P}_+(t)} \mu(r,s)drds - \iint_{\mathcal{P}_-(t)} \mu(r,s)drds \\ &= \iint_{\mathcal{P}_+(t)} \mu(r,s)drds - \left[\iint_{\mathcal{P}} \mu(r,s)drds - \iint_{\mathcal{P}_+(t)} \mu(r,s)drds\right] \\ &= 2\iint_{\mathcal{P}_+(t)} \mu(r,s)drds - \iint_{\mathcal{P}_r} \mu(r,s)drds \end{aligned} \tag{2.3}$$

Since $\mathcal{P}_+(t)$ and $\mathcal{P}_-(t)$ can be defined by the boundary ψ, the output of the Preisach model can be written as a function of the boundary:

$$y(\psi) = 2\iint_{\mathcal{P}_+(\psi)} \mu(r,s)drds - \iint_{\mathcal{P}_r} \mu(r,s)drds \tag{2.4}$$

2.3.3 Wiping Out Property

The Preisach model is used to describe the class of rate independent or static hysteretic systems. Rate independence means that the relationship between the input and output is invariant to the frequency of the input. An important consequence of rate independence is that the output of these systems only depends on the input extrema of the past, and not all of the input history. This is called the *wiping out property*.

To illustrate this property, we will continue with the previous example, shown in Figure 2.12. Suppose now that the input is increased monotonically to $u = 6$. As before, relays that satisfy $u = s + r$ switch from an output of -1 to +1. The part of $\mathcal{P}_-(t)$ that lies below the line $s = u - r$ becomes a part of $\mathcal{P}_+(t)$ as a result (Fig 2.12b). $\mathcal{P}_+(t)$ grows until a segment of the boundary reaches $s = 6 - r$ (Fig 2.12d). Note that once the boundary have moved past $s = 4 - r$ (Fig 2.12c), the boundary segments corresponding to the previous input extrema are "wiped out"(the dotted-line in Figure 2.12 represents

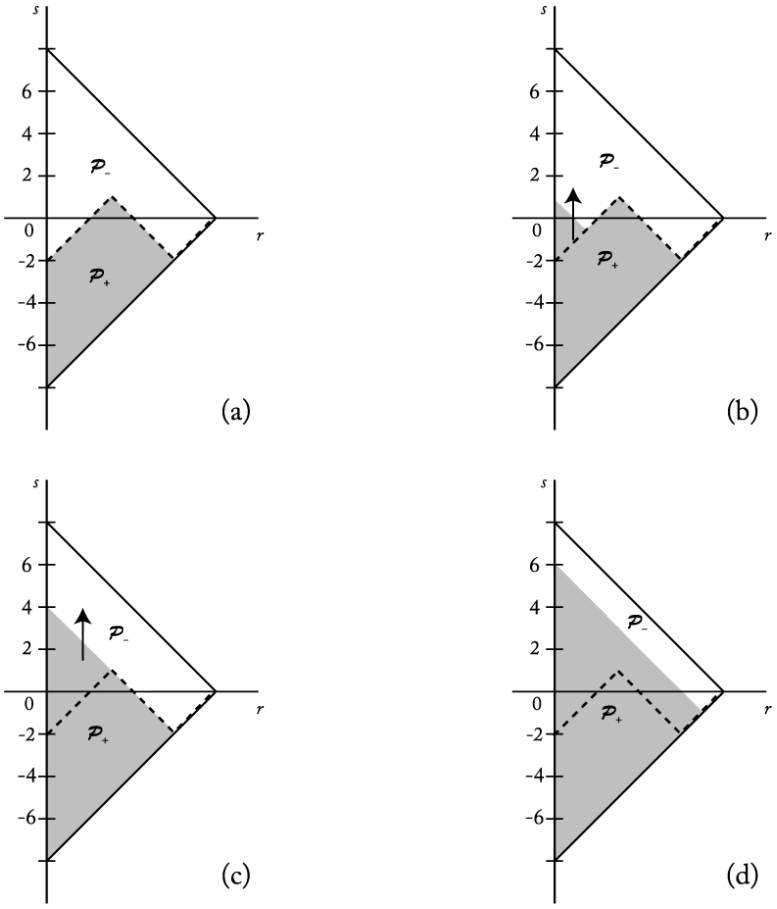

Figure 2.12: Wiping Out Property

the previous boundary). We can represent the history of input extrema using a *reduced memory sequence*, which will be discussed later.

2.4 Preisach Model as a Dynamical System

In order to employ the tools in control theory, one would prefer to work with a state space representation where stability techniques such as Lyapunov and dissipativity theory may be applied. The formulation is given in this section. We first introduce the definition of a dynamical system as presented in [55]:

Definition 2.4.1 *A (continuous stationary) dynamical system Σ is defined through the sets U, $\mathcal{U}$, Y, $\mathcal{Y}$, X and the maps ϕ and r. These satisfy the following axioms:*

(i) $\mathcal{U}$ *is called the* input space *and consists of a class of U-valued functions on $\mathbb{R}$. The set U is called the set of* input values. *The space $\mathcal{U}$ is assumed to be closed under the shift operator, i.e., if $u \in \mathcal{U}$ then the function u_T defined by $u_T(t) = u(t+T)$ also belongs to $\mathcal{U}$ for any $T \in \mathbb{R}$;*

(ii) $\mathcal{Y}$ *is called the* output space *and consists of a class of Y-valued functions on $\mathbb{R}$. The set Y is called the set of* output values. *The space $\mathcal{Y}$ is assumed to be closed under the shift operator, i.e., if $y \in \mathcal{Y}$ then the function y_T defined by $y_T(t) = y(t+T)$ also belongs to $\mathcal{Y}$ for any $T \in \mathbb{R}$;*

(iii) X *is an abstract set called the* state space*;*

(iv) ϕ *is called the* state transition function *and is a map from $\mathbb{R}^2_+ \times X \times \mathcal{U}$ into X. It obeys the following axioms:*

$(iv)_a$ *(consistency): $\phi(t_0, t_0, x_0, u) = x_0$, $\forall t_0 \in \mathbb{R}$, $x_0 \in X$, and $u \in \mathcal{U}$;*

$(iv)_b$ *(determinism): $\phi(t_1, t_0, x_0, u_1) = \phi(t_1, t_0, x_0, u_2)$, $\forall (t_1, t_0) \in \mathbb{R}^2_+$, $x_0 \in X$, and $u_1, u_2 \in \mathcal{U}$ satisfying $u_1 = u_2$ for $t_0 \le t \le t_1$;*

$(iv)_c$ *(semi-group property): $\phi(t_2, t_0, x_0, u) = \phi(t_2, t_1, \phi(t_1, t_0, x_0, u), u)$, $\forall t_0 \le t_1 \le t_2$, $x_0 \in X$, and $u \in \mathcal{U}$;*

$(iv)_d$ *(stationarity): $\phi(t_1 + T, t_0 + T, x_0, u_T) = \phi(t_1, t_0, x_0, u)$, $\forall (t_1, t_0) \in \mathbb{R}^2_+$, $T \in \mathbb{R}$, $x_0 \in X$, and $u, u_T \in \mathcal{U}$ related by $u_T = u(t + T)$ $\forall t \in \mathbb{R}$;*

(v) r *is called the* read-out function *and is a map from $X \times U$ into Y ;*

(vi) the Y-valued function $r(\phi(t, t_0, x_0, u), u(t))$ defined for $t = t_0$ is, $\forall x_0 \in X$, $t_0 \in \mathbb{R}$ and $u \in \mathcal{U}$, the restriction to $[t_0, \infty)$ of a function $y \in \mathcal{Y}$. This means that $\exists\, y \in \mathcal{Y}$ such that $y = r(\phi(t, t_0, x_0, u), u(t))$ for $t \ge t_0$.

2.4.1 Input, Output and State Spaces

Definition 2.4.2 *[15] The* input space U *is defined as the set of continuous real-valued functions that satisfy*

$$U = \left\{ u \in C^0 | \; |u| \leq u_{sat} \, \forall t \; and \lim_{t \to -\infty} u(t) = u^* \right\}$$

where u^* is the input value corresponding to the relaxed state of the system, for example at room temperature. The boundedness of the inputs allow a bounded Preisach Plane $\mathcal{P}_r$ and the limit restriction avoids discontinuity at the initial boundary.

Definition 2.4.3 *[15] The* output space Y *is the set of continuous real-valued functions.*

Note that both the input and output spaces are closed under the shift operator.

Definition 2.4.4 *[15] The* state space B *is the set of continuous functions $\psi : [0, u_{sat}] \mapsto \mathbb{R}$ with Lipschitz constant 1 and initial condition $\psi(u_{sat}) = 0$.*

This definition ensures that the state space is complete and all elements of B are inside the Preisach plane $\mathcal{P}_r$.

A state-space representation is defined using the input, output and state spaces defined above. The state-transition and read-out operators of the state-space representation for the Preisach model are given in detail in Appendix A. The state-space representation allows the use of dissipativity and Lyapunov theory to obtain stability results and is discussed later.

2.4.2 Reduced Memory Sequences

As we have seen in the previous section on the wiping out property, only the a subset of previous input extrema affects the output. Furthermore, once the input exceeds the magnitude of the previous extrema, the history of these extrema is wiped out. At any time t, we record the input extrema that have an effect on the output. An alternating sequence of input extrema is formed. The magnitude of the values in the sequence is decreasing and converges to the present input value $u(t)$. This sequence is called the *reduced memory sequence.*

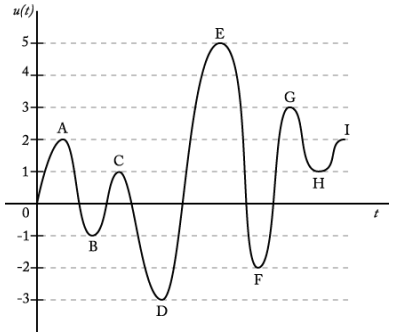

Figure 2.13: Sample Input Signal

Table 2.1: Reduced Memory Sequence

	Reduced Memory Sequence
A	{2}
B	{2, -1}
C	{2, -1, 1}
D	{2, -3}
E	{5}
F	{5, -2}
G	{5, -2, 3}
H	{5, -2, 3, 1}
I	{5, -2, 3, 1, 2}

We will use an example to illustrate the idea of reduced memory sequences. Suppose we would like to record the extrema of the input shown in Figure 2.13. The extrema are labeled from A to I and their corresponding reduced memory sequences are shown in Table 2.1. Note that at point E, the new input extremum exceeds the previous extremum A in magnitude, therefore the entire history up to that point is wiped out.

A reduced memory sequence stores the subset of input extrema that affects the current output value. This reduces the memory needed in numerical simulations. Instead of storing all the input values, only a countable set of input extrema and the current input value are stored. The elements of the reduced memory sequence defines the corners of the

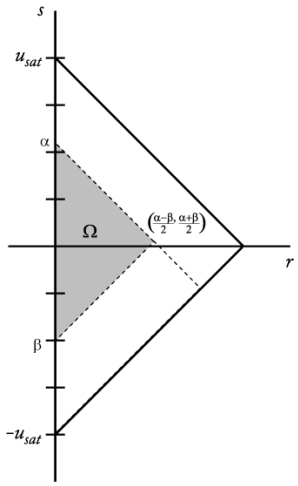

Figure 2.14: Region Ω Corresponding to Output Change $y_\alpha - y_{\alpha\beta}$

Preisach boundary. This property is used to implement the Preisach model efficiently and is discussed in the following section.

2.5 Model Identification

In [38], Mayergoyz introduced a method to obtain information on the weighting surface $\mu(r, s)$ from experimental data. The method uses *first-order descending curves* or FOD curves. A first-order descending curve involves first bringing the system to negative saturation. The input is then increased monotonically to a value α, and then decreased monotonically to a value β. The output at the end of the sequence is recorded. The *first-order* in the term 'FOD' comes from the fact that the input changes from increasing to decreasing once, and it ends when the input is decreasing.

The formulation in [38] uses a different coordinate system for the Preisach plane. We will develop a similar result with the (r, s)-plane configuration. Let y_α denote the output of the system after the input has increased monotonically to α from negative saturation and let $y_{\alpha\beta}$ denote the output of the system after the input has further decreased to β from α. Then the change in output during cooling, as a function of α and β is

$$F(\alpha, \beta) = y_\alpha - y_{\alpha\beta}.$$

Using equation (2.4),

$$\begin{aligned} y_\alpha &= 2\iint_{\mathcal{P}_+(\alpha)} \mu(r,s)drds - \iint_{\mathcal{P}_r} \mu(r,s)drds \\ y_{\alpha\beta} &= 2\iint_{\mathcal{P}_+(\alpha\beta)} \mu(r,s)drds - \iint_{\mathcal{P}_r} \mu(r,s)drds \\ y_\alpha - y_{\alpha\beta} &= 2\iint_{\mathcal{P}_+(\alpha)} \mu(r,s)drds - 2\iint_{\mathcal{P}_+(\alpha\beta)} \mu(r,s)drds \end{aligned} \tag{2.5}$$

where $\mathcal{P}_+(\alpha)$ and $\mathcal{P}_+(\alpha\beta)$ denote the region $\mathcal{P}_+$ corresponding to outputs y_α and $y_{\alpha\beta}$ respectively. Equation (2.5) can be rewritten as

$$y_\alpha - y_{\alpha\beta} = 2\iint_{\Omega} \mu(r,s)drds \tag{2.6}$$

where Ω is the region shown in Figure 2.14.

The relationship between the weighting surface μ and F, using Leibniz integral rule, is given by

$$\begin{aligned} F(\alpha,\beta) &= 2\int_0^{\frac{\alpha-\beta}{2}} \int_{\beta+r}^{\alpha-r} \mu(r,s)dsdr \\ \frac{\partial F}{\partial \beta} &= 2\left(\underbrace{-\frac{1}{2}\int_{\frac{\alpha+\beta}{2}}^{\frac{\alpha+\beta}{2}} \mu(\frac{\alpha-\beta}{2},s)ds}_{=0} + \int_0^{\frac{\alpha-\beta}{2}} \frac{\partial}{\partial\beta}\left(\int_{\beta+r}^{\alpha-r} \mu(r,s)ds\right)dr \right) \\ &= 2\int_0^{\frac{\alpha-\beta}{2}} \frac{\partial}{\partial\beta}\left(\int_{\beta+r}^{\alpha-r} \mu(r,s)ds\right)dr \\ &= -2\int_0^{\frac{\alpha-\beta}{2}} \mu(r,\beta+r)dr \\ \frac{\partial^2 F}{\partial\alpha\partial\beta} &= -2\left(\frac{1}{2}\mu(\frac{\alpha-\beta}{2},\beta+\frac{\alpha-\beta}{2})\right) \\ &= -\mu(\frac{\alpha-\beta}{2},\frac{\alpha+\beta}{2}) \end{aligned} \tag{2.7}$$

The output of the system is recorded for α and β in $[-u_{sat}, u_{sat}]$. A smooth surface is fit onto the experimental data. The weighting surface $\mu(r,s)$ can be identified by differentiating the fitted surface twice using equation (2.7). The weighting surface is mainly used

in analysis and is not necessary for the purposes of this work. Furthermore, differentiating a fitted surface and then integrating again will introduce unwanted errors for a small experimental data set. Fortunately, even with a limited set of FOD data, the output of the Preisach model can be approximated by interpolating the known output values.

Numerical implementation of the Preisach model is demonstrated with the following example. First, a smooth surface is fit onto the experimental data. This gives the function $F(\alpha, \beta)$ over the possible input range. To calculate the output given the boundary in Figure 2.15, $\mathcal{P}_+$ defined by the boundary ψ is divided into regions Ω_1 and Ω_2, shown in Figure 2.16. The corresponding reduced memory sequence s for this input history is

$$s = \{-8, 4, -6, 2, 0\} \tag{2.8}$$

Note that each region can be defined by elements of s. By the definition of the FOD data, the integral of μ over the two regions are given by:

$$\begin{aligned} 2\iint_{\Omega_1} \mu(r,s)drds &= F(4,-8) - F(4,-6) \\ 2\iint_{\Omega_2} \mu(r,s)drds &= F(2,-6) - F(2,0) \end{aligned}$$

The positive saturation value is

$$\iint_{\mathcal{P}_r} \mu(r,s)drds = \frac{F(8,-8)}{2}$$

Using equation (2.4), the system output is

$$\begin{aligned} y(\psi) &= 2\iint_{\mathcal{P}_+(\psi)} \mu(r,s)drds - \iint_{\mathcal{P}_r} \mu(r,s)drds \\ &= 2\iint_{\Omega_1} \mu(r,s)drds + 2\iint_{\Omega_2} \mu(r,s)drds - \iint_{\mathcal{P}_r} \mu(r,s)drds \\ &= F(4,-8) - F(4,-6) + F(2,-6) - F(2,0) - \frac{F(8,-8)}{2} \end{aligned}$$

In general, we can write the output of the Preisach model in terms of elements in the reduced memory sequence s. Let M_i and m_i denote the i^{th} maximum and minimum value in s. For example, in the reduced memory sequence (2.8), $M_2 = 2$ and $m_2 = -6$. If the

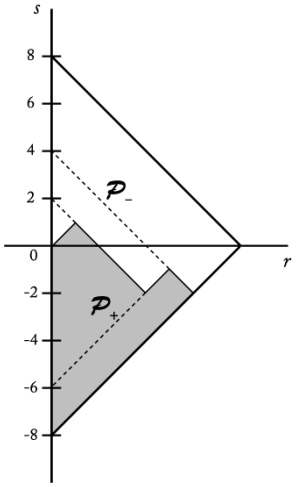

Figure 2.15: Sample Boundary

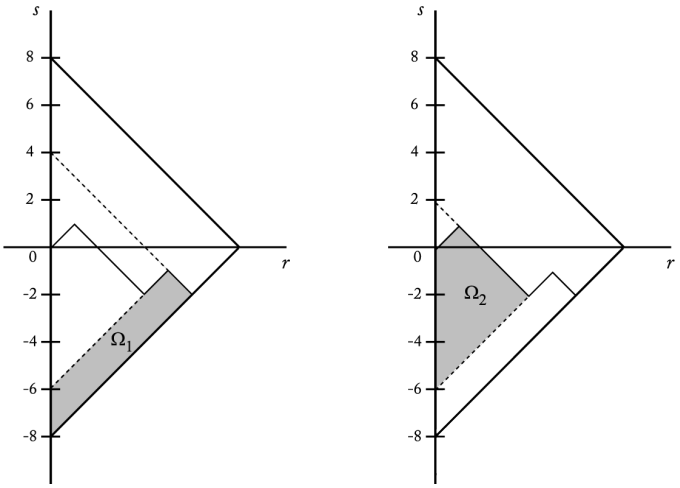

Figure 2.16: Regions Ω_1 and Ω_2

input is increasing, then the number of maximum and minimum values are the same. There are $2n$ elements in s. The first minimum value corresponds to negative input saturation and the last maximum value corresponds to the current input. The output of the system is given by

$$y = \sum_{i=1}^{n-1}[F(M_i, m_i) - F(M_i, m_{i+1})] + F(M_n, m_n) - \frac{F(u_{sat}, -u_{sat})}{2} \tag{2.9}$$

For decreasing input, there is one more minimum value than maximum value. Let n be the number of maximum values in s. The output of the system is then given by

$$y = \sum_{i=1}^{n}[F(M_i, m_i) - F(M_i, m_{i+1})] - \frac{F(u_{sat}, -u_{sat})}{2} \tag{2.10}$$

These equations can be easily implemented using a reduced memory sequence and information on F over all possible input values. The output resulting from any input can be calculated by a sum of trapezoids as described above.

It has been mentioned that the Preisach model is symmetric about the origin, but it is clearly not the case for the SMA wire. The input to the SMA wire is temperature above ambient temperature. Therefore, an offset term is added so that the output is zero when the system is at negative saturation. The offset term is

$$\frac{y_+^{sat} + y_-^{sat}}{2}$$

where y_+^{sat} and y_-^{sat} are the output values corresponding to the input saturation values.

2.6 Summary

In this chapter, background on different shape memory alloy modelling techniques was given. The Preisach model for SMA was introduced. The state-space representation for the Preisach model was described and will be useful in the discussion of dissipativity theory in controlling SMAs. Model identification using experimental data was briefly discussed. This allows the computation of the output of the Preisach model without explicitly finding the weighting surface. This method will be used in the simulations later on.

Chapter 3

Control of Shape Memory Alloys

In this chapter, we will concentrate our discussion on the following control strategies. A brief background on dissipativity of the Preisach model is given for velocity control. An example of an inverse model used to reduce hysteretic nonlinearity is given. Optimal control strategies in various smart structures applications are discussed.

3.1 Dissipativity of Preisach Model

We first give the definition of dissipativity [55].

Definition 3.1.1 *A dynamical system is said to be* dissipative with respect to the supply rate $w : U \times Y \mapsto \mathbb{R}$ *if there exists a non-negative* storage function $S : X \mapsto \mathbb{R}_+$ *such that* $\forall t_1 \geq t_0$*, initial state* $x_0 \in X$*, and input* $u \in \mathcal{U}$*,*

$$S(x_0) + \int_{t_0}^{t_1} w(u(t), y(t))dt \geq S(\phi(t_1, t_0, x_0, u)) \tag{3.1}$$

where $y(t) = r(\phi(t, t_0, x_0, u), u(t))$*.*

Dissipativity can be used to obtain stability result of a system. Since stability is often described using the distance between trajectories, we need to introduce the notion of norms in order to quantify distance.

Definition 3.1.2 *A real-valued function $\|\cdot\|$ on a set $\mathcal{U}$ is called a* norm *if, for all $u, v \in \mathcal{U}$, it satisfies the following axioms:*

(i) (Positivity) $\|u\| \geq 0$;

(ii) (Strict Positivity) $\|u\| = 0 \Leftrightarrow u = 0$;

(iii) (Homogeneity) $\|au\| = |a|\|u\|$, $\forall a \in \mathbb{R}$;

(iv) (Triangle Inequality) $\|u + v\| \leq \|u\| + \|v\|$.

For the spaces of real-valued functions, a very common norm is the $\mathcal{L}_p$ norm. The $\mathcal{L}_p$ norm of a function $f(t)$ on the interval $[a, b]$ is defined as:

$$\begin{aligned} \|f(t)\|_p &= \left(\int_a^b |f(t)|^p dt\right)^{\frac{1}{p}}, \ for 1 \leq p < \infty; \\ \|f(t)\|_\infty &= \max_{a \leq t \leq b} |f(t)|. \end{aligned}$$

The space of all functions with finite $\mathcal{L}_p$-norm over the interval $[a, b]$ is denoted $\mathcal{L}_p(a, b)$.

For most physical systems, the mathematical model is given in input-output form. We have the following definition of stability relating the input and output signals.

Definition 3.1.3 *[54] A system is called* input-output *stable if for any inputs $u(t) \in \mathcal{U}$, there exists finite constants k and b such that the corresponding output $y(t)$ satisfies*

$$\|y(t)\| \leq k\|u(t)\| + b, \ \forall t \geq 0.$$

A system satisfying the above definition is said to be *finite gain stable with gain k*.

A major result in dissipativity theory that can be used in feedback control is that the interconnection of dissipative systems is also dissipative under some simple assumptions [24, 25]. We first define a class of dissipative systems.

Definition 3.1.4 *[40] Given matrices P, R, S of appropriate dimensions with P, S symmetric. A system is called (P, R, S)-*dissipative *if it is dissipative with respect to the supply rate*

$$w(u, y) = \langle y, Py \rangle + 2\langle y, Ru \rangle + \langle u, Su \rangle \tag{3.2}$$

where $\langle , \rangle$ denotes the scalar product.

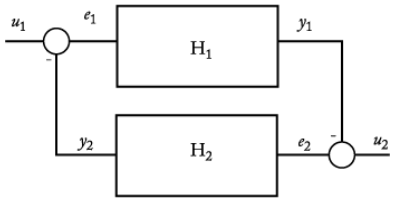

Figure 3.1: Feedback Configuration

Note that $(-I, 0, k^2 I)$-dissipativity is equivalent to

$$\|y\| \leq k\|u\|$$

for some positive constant k, or the system has *finite gain* k. Furthermore, a system is called *passive* if $P = 0,\ R = I,\ S = 0$.

The *Small-Gain Theorem* and the *Passivity Theorem*(eg. [31, 54]) are stated below without proof.

Theorem 3.1.5 *Consider the feedback system of Figure 3.1. Assume that the subsystems H_1 and H_2 have finite gains k_1 and k_2 respectively. Then the feedback connection is finite gain stable if $k_1 k_2 < 1$.*

Theorem 3.1.6 *[31] The feedback connection of two passive systems is passive.*

The two theorems stated above can be used to show stability of an interconnected system by examining each subsystem individually.

In [16], the Preisach plane $\mathcal{P}_r$ is divided into the following regions (cf. Figure 3.2):

$$\begin{aligned} \mathcal{P}_1 &= \{(r,s) \in \mathcal{P}_r |\ s > r\} \\ \mathcal{P}_2 &= \{(r,s) \in \mathcal{P}_r |\ |s| \leq r\} \\ \mathcal{P}_3 &= \{(r,s) \in \mathcal{P}_r |\ s < -r\} \end{aligned} \tag{3.3}$$

The energy transfer of the relays when they go through a change in output is defined. Let $q_+ = 2\mu(r,s)(s+r)$ and $q_- = -2\mu(r,s)(s-r)$ where $q+$ represents the energy loss when the relay is switched from output of -1 to $+1$ and similarly q_- represents the energy

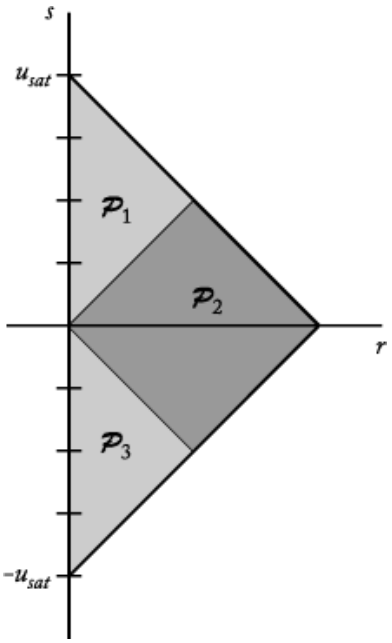

Figure 3.2: Preisach Plane Regions

loss when the relay is switched from output of $+1$ to -1. Since a negative energy loss means that the system is gaining energy, the total stored energy is given by [16]:

$$Q(\psi(t)) = 2\int_{\mathcal{P}_1 \cap \mathcal{P}_+(t)} \mu(r,s)(s-r)dsdr - 2\int_{\mathcal{P}_3 \cap \mathcal{P}_-(t)} \mu(r,s)(s+r)dsdr \qquad (3.4)$$

By the definitions of the regions $\mathcal{P}_1$ and $\mathcal{P}_3$, $Q(\psi)$ is nonnegative. Furthermore, since μ and $\mathcal{P}_r$ are bounded, so is $Q(\psi)$. Hence $Q(\psi)$ is a valid storage function.

Theorem 3.1.7 *[16] If $\mu \in \mathcal{M}_p$, the Preisach model with storage function Q as defined in equation (3.4) is dissipative with respect to the supply rate $w = u\dot{y}$, where $\dot{y}$ is the time derivative of the read-out operator $r(\psi)$.*

Experimental results in [16] show stability of the velocity control of an SMA actuator using a PD-controller.

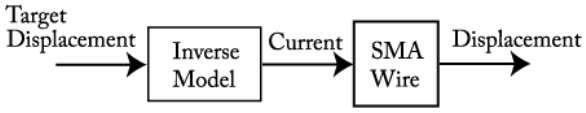

Figure 3.3: Inverse Model in Open-loop Control

3.2 Inverse Model for SMA Wire Actuator

A popular method to control the hysteresis in smart materials is to implement an inverse compensator, for example in [12, 20]. Inverse compensators attempt to remove the hysteresis nonlinearity and reduce the problem complexity and are discussed later. The idea of an inverse model is to reconstruct the input given output information of the system. The inverse model calculates the necessary input that is required to obtain a desired output. The inverse model is put in series with the actual system as shown in Figure 3.3.

The inverse model is used as a feedforward controller in open-loop applications. It calculates the corresponding current for the desired reference position. The current is then applied to the actual SMA wire to obtain the desired wire position. Hysteresis model inversion can be done both analytically and numerically. Analytical or model-based inverse filter is used to control piezoelectric transducers in an atomic force microscope [20] for an energy-based hysteresis model. Two numerical algorithms to determine inverse Preisach models are presented in [28].

In [52], a neural network is used to create an inverse model for an SMA wire actuator. The idea is as follows: first, the neural network is designed to model position as a function of input voltage. The neural network takes input voltage, and a tag signal indicating whether the input voltage is increasing or decreasing, as inputs. The tag signal is included so that the network would behave differently when the input is increasing/decreasing, as it would be in a hysteretic system. The output is displacement.

The major hysteresis loop of wire displacement with respect to input voltage is used to train the network. Experimental results show that the neural network can effectively recreate the hysteresis loop of the wire. The neural network model can be used for simulation purposes.

Next, the same set of data is used to train the inverse model. A schematic of the neural network training is shown in Figure 3.4. A tag signal is included to serve the same purpose

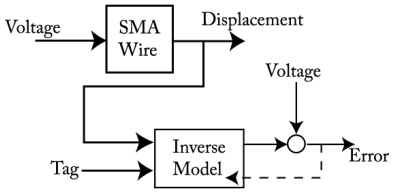

Figure 3.4: Neural Network Inverse Training Schematic

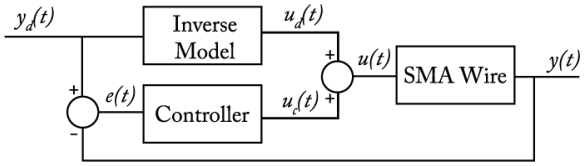

Figure 3.5: Inverse Model with Feedback Control

as before. The new neural network is trained to map the displacement of the wire to the input voltage. The results of the inverse model are compared with the training data.

In training the neural network, a sinusoidal input at $\frac{1}{60}$Hz is used to allow sufficient cooling time for the wire. The resulting inverse model is tested experimentally with a reference target of $\frac{1}{60}$Hz. Since the neural network models both the hysteretic behaviour of the wire and the heating dynamics converting voltage to temperature, it is expected to perform well at the same frequency at which it is trained. The inverse model is tested with a 10Hz signal in simulation, but its performance is not verified experimentally. Experimental results show reasonably good tracking, and encourage further study in improving this technique.

The disadvantage of the above open-loop control scheme is that it relies on the accuracy of the inverse model. Uncertainties in the model will result in a possibly unstable system. A feedback controller is included in [12] to provide better tracking results. The controller corrects modelling uncertainty and disturbances by comparing the desired and actual output values of the system. The schematic of an inverse model with feedback control is shown in Figure 3.5.

The signal $y_d(t)$ is the desired output trajectory. The inverse model calculates the corresponding input signal $u_d(t)$. The error between the actual output and the desired output is fed into the controller. The controller regulates this error by adding an extra

control signal $u_c(t)$. This method is shown to have better tracking performance in [12]. An inverse model is also used in $\mathcal{H}_\infty$ control design for magnetostrictive materials in [41].

3.3 Control of SMA Actuated Smart Structures

Many smart structures use SMAs as actuators and sensors. There has been research on controlling smart structures using SMA actuators, for example [35] and [7]. In order to devise control strategies for these structures, a lumped model of the actuator and the dynamics of the structure is often used. In the numerical analysis in [7], combined models of the structure and actuators are used. The actuator dynamics are simplified and do not account for the hysteretic behaviour.

One of the main application of SMA actuators is in vibration control. In [35], the modal equations of a cantilever beam are used to describe the effects of different frequencies on the system. The uncertainty of this problem is the natural frequencies of the structure. The authors proposed a new $\mathcal{H}_\infty$ robust control algorithm for natural frequency variations. There is no consideration for the hysteretic behaviour of the actuators. The model assumes that any arbitrary actuation can be achieved. In another paper [7] on cabin noise control, a linear approximation of the actuator behaviour is used.

The hyperbolic tangent function is used in [19] along with reinforcement learning to adjust the parameters of the hyperbolic tangent function to approximate the local behaviour in the hysteresis. Different simplifications to the modelling of the SMA actuator allow for the use of linear control theory techniques such as linear quadratic regulator and variable structure control [18]. But since the actuator is highly nonlinear, the simplifications come at the cost of accuracy. These examples motivate the need for good modelling and control algorithms for the hysteretic behaviour of SMA.

A real-time control application is presented in [47]. The objective is to design an adaptive tuned vibration absorber (ATVA) using SMA elements. Young's modulus of the ATVA changes as the SMA undergoes a phase transformation. A piecewise model is used to relate Young's modulus to temperature. The phase change resulting from heating and cooling are treated separately. The heat capacity of the SMA are assumed to take on two different values based on the current temperature. Proportional controller, PD-controller

and fuzzy control are considered in the paper. The PD-controller has superior performance and is very simple to implement.

3.3.1 Optimal Control

In applications using smart materials as actuators and sensors, the optimal placement of the actuators and sensors are often of interest (e.g. [8]). In order to use standard optimal control techniques such as a linear quadratic regulator (LQR), a reduced-order model of a thin cylindrical shell structure is used in [2]. The resulting PDE-model is implemented numerically. The algebraic Ricatti equations for the LQR control problem are then solved using the reduced-order model. Similar techniques are used in [50] and [7] to obtain optimal controllers for piezoelectric material applications.

In [48], an energy-based model is used for SMA actuators. It uses Helmholtz free energy and Gibb's energy, and probabilities of phase transition. Experimental results show that the model is able to reproduce hysteretic behaviour very well. The authors took this model and implemented it in NUDOCCS [23] for optimal control. NUDOCCS uses a sequential quadratic programming method to solve the non-linear optimization problem. The optimal control signal is calculated to drive the system from one fixed reference point to another. The calculations are performed in the order of nano-seconds, making it possible to be incorporated in real-time control applications.

The performance of the optimal controller is compared with a PI-controller. Simulated results show that the optimal controller performs better than the PI-controller, whose gain values are tuned manually. Due to the lack of mathematical analysis of the different models, only numerical optimization methods are used, see [11].

3.4 Summary

In this chapter, different techniques used to control SMA actuators are discussed. A definition of dissipativity is given, and results on the dissipativity of the Preisach model is presented. The stability result for velocity control is discussed. A neural network inverse model for the SMA actuator is described. Several control strategies for SMA actuated smart structures are discussed.

Because of the complexity of hysteresis, analysis involving hysteresis is usually avoided. The varying physical parameters of the material needs to be taken into account for a more accurate model. This will be addressed in the simulation of the SMA wire. Since the use of SMA actuators in smart structures is popular, it is useful to consider the control of the actuators individually, as well as how they interact with the structure as a whole.

Chapter 4

Optimal Control Problem

This chapter provides the framework for the optimal control problem. The objectives of the control problem are given. An attempt to derive a derivative for the Preisach model is given. Different optimization algorithms that do not require derivative information are investigated. The Nelder-Mead simplex algorithm for optimization is chosen and described in the chapter.

4.1 Objectives

In many control problems, the objective is to find the 'best' control for a given task. It is then necessary to define a measure of performance. We assume that control signals are 'better' if they result in a lower performance index. In addition, there are usually constraints on the controller or system. Common physical constraints are input and output saturations. The *optimal controller* is the admissible control $u(t)$ that minimizes the performance index, or *cost function*, under the constraints in the overall system.

As described in Chapter 2, the system output is the contraction of the SMA wire under a constant load subject to input current. In Gorbet and Wang [17], stability results on position control of SMA wire are given. The results are used to show stability using approximated proportional-integral-derivative (PID), PI and PD controllers. In particular, two PI controllers are tested and have shown to give good performance. For the purposes

of this study, we will be using a PID controller. A PID-controller is defined by

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{\mathrm{d}e}{\mathrm{d}t} \tag{4.1}$$

where $e(t) = r(t) - y(t)$ is the difference between the reference signal and the system output. Instead of determining the optimal control signal, the objective is to find the optimal PID-controller gain values K_p, K_i, and K_d.

We will look at three different cost functions for output tracking. A common objective is for the output of the system to track the input signal over a time interval of T seconds. Imagine an inkjet printer cartridge or a laser cutter actuated by SMA wires. It is important that the system is able to track the reference signal as closely as possible. A common cost function for the above problem is

$$J_1(K_p, K_i, K_d) = \int_0^T (r(t) - y(t))^2 dt \tag{4.2}$$

where

$$\begin{aligned} y(t) &= \text{wire contraction} \\ r(t) &= \text{desired contraction} \\ T &= \text{length of control time interval} \end{aligned}$$

In certain applications, it is not necessary to track a reference signal over a time interval. A possible objective is for the actuator to move the load from point A to point B and back every T seconds. If there is no restriction on the path it takes to go from point A and point B, only the output values at every T seconds need to match the reference. A simple cost function for this problem is

$$J_2(K_p, K_i, K_d) = \sum_{i=1}^{n} |y_s^i - r_s^i|^2 \tag{4.3}$$

where

$$\begin{aligned} y_s^i &= \text{contraction at } i^{th} \text{ switch time} \\ r_s^i &= \text{desired contraction at } i^{th} \text{ switch time} \\ n &= \text{number of switching times} \end{aligned}$$

In certain applications, the displacement of the wire over the time interval is of interest. Then it is not necessary to specify the exact contraction value of the wire. Furthermore, the reference signal can be used as an extra design parameter to the problem. A proposed cost function for this problem is

$$J_3(K_p, K_i, K_d) = \sum_{i=1}^{n-1} |\Delta y^i - \Delta y_d|^2 \tag{4.4}$$

for a fixed reference signal $r(t)$ where

$$\begin{aligned} \Delta y^i &= |y_s^i - y_s^{i+1}| \\ y_s^i &= \text{contraction at } i^{th} \text{ switch time} \\ \Delta y_d &= \text{desired wire travel} \\ n &= \text{number of switching times} \end{aligned}$$

In output tracking problems for periodic signals, the common practice is to find the optimal controller for the step response of the system. This approach works when there is full control over both the heating and cooling phases of the system. In the absence of active cooling, it is important not to overheat the SMA wire so that it will have enough time to cool before the next heating cycle. Since the step response does not contain a cooling phase, it is reasonable to assume that the optimal controller obtained from the step response may not be optimal for periodic reference signals.

The goal of this research is to compare the performance of a controller that is optimized for a step reference, and controllers that are optimized for periodic reference signals of different frequencies. Furthermore, we want to investigate the bandwidth of each of these

optimal controllers. A common definition of bandwidth is given in terms of the transfer function $G(j\omega)$ of the system:

$$\bar{\omega} = \sup_{\omega}\{\omega \mid \|G(j\omega)\| > k^*\} \tag{4.5}$$

where k^* is a specified gain value.

The system in this study is nonlinear and hence it does not have a transfer function. Therefore, a different definition of bandwidth is needed and is defined below.

Definition 4.1.1 *For a given cost function J_i and fixed reference signal frequency ω, controller H_1 has better performance over another controller H_2 if*

$$J_i(H_1, \omega) < J_i(H_2, \omega)$$

where $J_i(H_l, \omega)$ is the cost function value of the overall system with controller H_l and a reference signal frequency ω.

The bandwidth *of the system with controller H_l is defined by*

$$\bar{\omega}_l = \sup_{\omega}\{\omega | J_i(H_l, \omega) < e^*\} \tag{4.6}$$

where e^* is a specified error value.

Furthermore, controller H_1 is said to have a wider bandwidth *than another controller H_2 if $\bar{\omega}_1 > \bar{\omega}_2$.*

For systems that may be subject to reference signals of varying frequencies, it is useful to have a controller that works over a range of frequencies, rather than changing to a different controller for each frequency.

The performances of the various controllers will be compared using the cost functions described in this section. In the following sections, we will look at the tools to determine the controller gains that minimizes the above cost functions. Since the system output is calculated numerically, numerical optimization methods will be used to solve the optimal control problem.

4.2 Derivative of Preisach Model

In most optimization algorithms, a derivative function is needed to efficiently solve the optimization problem. The derivative function provides a means to decide the direction of search in each iteration. An attempt was made to find a derivative of the Preisach model and the PID-controller to help us solve the optimal control problem.

The derivative of the first cost function, equation (4.2), by Leibniz Rule, is

$$\nabla_{K_p,K_i,K_d}(f) = -\int_0^T 2(r(t) - y(t))D_u(y)\nabla_{K_p,K_i,K_d}(u)\ dt \tag{4.7}$$

First, we need to determine the derivative of the Preisach model output $y(t)$ with respect to the input signal $u(t)$. Since the input signal is a function, and the output is dependent on past history, a point-wise derivative is not sufficient. In finding the derivative at time t, any changes to the input before time t will have an effect on the output and hence the derivative. $D_u(y)$ needs to be evaluated for all time t and integrated over the time period.

Consider a small change in the input signal $u_1(t)$ to $u_2(t)$. If the change in $u_2(t)$ causes a previous input extremum before time t to be wiped out, the derivative $D_{u_1}(y)$ will be different from $D_{u_2}(y)$ since the input history has changed. Hence it is not clear how the change in the control signal would influence the output. Determining the derivative of the Preisach model remains a problem to be solved, and will greatly benefit the development of optimal control of systems with Preisach representations.

The main problem is due to the fact that the Preisach model output is dependent on past input history. Because of the complexity of the derivative, an algorithm that does not require derivative information is used and is described in the next section. The derivative function, if it exists, would be very useful in optimal control of systems with Preisach representations.

4.3 Optimization Algorithms

In optimization problems, derivative information on the cost function can be very useful in determining the solution. However, the convexity of the cost function is more important

than the availability of a derivative. Convexity of the cost function implies that a global minimum exists and is unique under mild assumptions [44]. While many optimization algorithms uses the convexity property, it is sometimes very difficult to prove that a cost function is indeed convex.

For optimization problems involving simulations, a closed form of a derivative function is usually not available. Finite difference approximations of the derivatives are used instead. The accuracy of the finite difference approximations depend on the smoothness of the function. However, the cost functions in simulations can be noisy and discontinuous due to numerical calculations. It is discussed in [32] that derivative-based algorithms are not robust to noise and discontinuities of the cost function. The derivative approximations will lead to a local minimum within the noise, which is not very useful.

In our optimization problem, the cost functions are found to be non-convex. Since the output of the Preisach model is calculated using an interpolation of the experimental data, it is subject to noise from numerical calculations. Furthermore, the derivative information of the Preisach model is not available. Therefore, a non-derivative-based algorithm needs to be chosen.

4.3.1 Direct Search Methods

Optimization algorithms that do not require derivatives or their approximations are often called *direct search methods*. Without explicit derivative information, the direct search methods use a set of predefined rules to traverse the domain towards a minimum. Because these methods do not take advantage of any auxiliary information about the problem, their performance is inferior to algorithms designed for the specific situation. On the other hand, this property gives these methods robustness for different types of problems.

A simple method is called the *compass search* [32]. For an n-dimensional problem, starting at an initial point x_0, evaluate the function at points a fixed distance Δ away from x_0 in each of the n directions. The direction that yields the smallest function value becomes the new initial point for the next iteration. If none of the n directions yield an improvement, the distance Δ is adjusted and the iteration is repeated.

The advantage of this method is that it is very simple to understand and implement. It also guarantees convergence to a local minimum by the definition. On the other hand,

there is no guarantee on the rate of convergence to the minimum.

Another example of a direct search method is Powell's algorithm [9]. While it is similar to the compass search, the direction of search is modified during each iteration. Starting at an initial point x_0 and a set of n linearly independent vectors $s^1, s^2, ..., s^n$, a line search is performed along s^1 to locate a minimum point x_1. Then, a line search is performed along s^2 to locate a minimum point x_2 and so on. After searching along all n vectors, a new search direction is obtained by connecting the newest point x_n with the initial point x_0 to get s^{n+1}. The process is repeated until convergence is achieved.

Powell's algorithm relies on the fact that the function is strictly convex in the line search directions. If it is not convex, there is no guarantee that the line searches will return a minimum. The algorithm simplifies the optimization problem to one dimension in each iteration, and the line searches can be performed using any optimization algorithm.

4.3.2 Genetic Algorithms

Genetic algorithms [13] are based on the rules of natural selection. They attempt to capture the natural selection phenomenon with a mathematical structure. Starting with an initial population, they combine with each other and produce the next generation. The new generation maintains some of the good traits of their predecessors, and new parts are added as well. The later generations are expected to improve upon the initial population.

A simple genetic algorithm is given in [13] to illustrate the idea. Assume that there is a black box with five on-off switches. The output from different configurations of the switches can be regarded as a payoff value. Hence the higher the output value, the better the configuration is. The input configuration is coded in binary: a string of five digits is used, with 1's and 0's representing the 'on' and 'off' position of the switches respectively.

An initial population of size $2n$ is generated randomly. Each member of the initial population are associated with a corresponding output value of the function. These can be viewed as the fitness levels of each member. The likelihood of a member to produce a good 'offspring' can be quantified by their fitness as a percentage of the total population fitness. Using these probabilities, n couples are chosen from the initial population.

The reproduction process is performed in the following manner: for each of the n couples, a number i between 1 and 4 is chosen at random. Members of the new generation

are created by swapping the part of the string from position i to the end between the couple. Hence for each couple, two new members are created and only top $2n$ members of the entire population is used to produce the next generation. The optimization terminates when the 'fitness levels' of each of the members of the population converges.

The idea of genetic algorithms is very intriguing. The main disadvantage of genetic algorithms is the complexity. While the operations in each iterations seem to be simple, it requires a coding of the domain of the function to be minimized. Hence, we will make use of a popular, yet easy to implement, algorithm described in the following section.

4.4 Nelder-Mead Simplex Algorithm

One of the more popular direct search methods is the Nelder-Mead Simplex Algorithm [42]. A simplex in n dimensions is a geometric object defined by $n+1$ vertices, where the lines connecting any two vertices are linearly independent. For example, a simplex in 2-D is a triangle, while a simplex in 3-D is a tetrahedron. Each iteration of the algorithm involves a simplex with $n+1$ vertices. The function values at each of the vertices are calculated, and a new simplex is generated after each iteration. The algorithm terminates when the size of the simplex, and the function values at its vertices satisfy some specified condition.

Since the publication of [42], many different variations have been introduced. One of these variations is called the *multi-directional search method* [53, 56]. In the multi-directional search method, the entire simplex is adjusted rather than individual vertices as in the Nelder-Mead Algorithm. This method is more complicated and is more numerically costly to implement.

There has been little discussion in literature on the convergence of the Nelder-Mead method, limited to low dimensions ($n \leq 2$) [34]. In [39], a counter example was used to show that the Nelder-Mead method converged to a non-minimizing point for $n = 2$.

Despite the lack of theoretical analysis and its deficiencies, the method is widely popular. The Nelder-Mead algorithm is used to solve an optimal transmitter location problem in [37]. The Nelder-Mead provided faster and better solutions than other methods that the authors used. It is also used in solar cell designs [6].

Combinations of the Nelder-Mead algorithm and other optimization methods are also

very popular. A random method is used to choose initial points for the Nelder-Mead algorithm for antenna optimization [33] with many local minimums of similar function values. Genetic algorithms discussed in the previous section are used in combination with the Nelder-Mead algorithm to create a hybrid optimization method [10]. A similar approach is used to optimize road noise barrier design in [3]

One of the reasons for its popularity is due to the small number of function evaluations. If derivatives of the cost function is not available, and the function evaluations are costly to calculate, then using finite-difference approximations will be expensive and slow. The algorithm does not guarantee convergence, but the same applies to other algorithms for a non-convex problem. Different initial values are used and the 'best' result from different trials will be used as the optimal solution. The Nelder-Mead algorithm is described in the following section.

4.4.1 Algorithm Description

The algorithm starts with an initial estimate $\mathbf{x}_0 = (x_0^1, x_0^2, \ldots, x_0^n)$ and constructs a non-degenerate simplex with vertices $\mathbf{x}_0$ and $\mathbf{x}_i = (x_0^1, x_0^2, \ldots, x_0^i + \delta, \ldots, , x_0^n)$ for $i = 1 \ldots n$ and predefined parameter $\delta > 0$. Four parameters must be specified: coefficients of reflection (ρ), expansion (χ), contraction (γ), and shrinkage (σ). The parameters should satisfy

$$\rho > 0,\ \chi > 1,\ \chi > \rho,\ 0 < \gamma < 1,\ \text{and}\ 0 < \sigma < 1. \tag{4.8}$$

Using these parameters, the Nelder-Mead method either finds a new vertex for the simplex, or performs a shrink of the simplex, leaving only the vertex that yields the best function value at each iteration.

Each iteration of the Nelder-Mead Algorithm is as follows [34]:

1. **Order.** Order the $n + 1$ vertices to satisfy $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \ldots \leq f(\mathbf{x}_{n+1})$, $\mathbf{x}_1$ is called the *best* point and $\mathbf{x}_{n+1}$ is the *worst* point.

2. **Reflect.** Compute the *reflection point* $\mathbf{x}_r$ from

$$\mathbf{x}_r = \bar{\mathbf{x}} + \rho(\bar{\mathbf{x}} - \mathbf{x}_{n+1}) = (1 + \rho)\bar{\mathbf{x}} - \rho\mathbf{x}_{n+1} \tag{4.9}$$

where $\bar{\mathbf{x}} = \sum_{i=1}^{n} \frac{\mathbf{x}_i}{n}$ is the centroid of the n best points. Evaluate $f_r = f(\mathbf{x}_r)$. If $f_1 \leq f_r < f_n$, replace the worst point $\mathbf{x}_{n+1}$ with the reflection point $\mathbf{x}_r$ and terminate the iteration.

3. **Expand.** If $f_r < f_1$, calculate the *expansion point* $\mathbf{x}_e$

$$\mathbf{x}_e = \bar{\mathbf{x}} + \chi(\mathbf{x}_r - \bar{\mathbf{x}}) = \bar{\mathbf{x}} + \rho\chi(\bar{\mathbf{x}} - \mathbf{x}_{n+1}) = (1 + \rho\chi)\bar{\mathbf{x}} - \rho\chi\mathbf{x}_{n+1} \tag{4.10}$$

and evaluate $f_e = f(\mathbf{x}_e)$. If $f_e < f_r$, replace the worst point $\mathbf{x}_{n+1}$ with $\mathbf{x}_e$ and terminate the iteration; otherwise, replace the worst point $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$ and terminate the iteration.

4. **Contract.** If $f_r \geq f_n$, perform a *contraction* between $\bar{\mathbf{x}}$ and the better of $\mathbf{x}_{n+1}$ and $\mathbf{x}_r$.

   a. **Outside.** If $f_n \leq f_r < f_{n+1}$, perform an *outside contraction*:

   $$\mathbf{x}_c = \bar{\mathbf{x}} + \gamma(\mathbf{x}_r - \bar{\mathbf{x}}) = \bar{\mathbf{x}} + \gamma\rho(\bar{\mathbf{x}} - \mathbf{x}_{n+1}) = (1 + \gamma\rho)\bar{\mathbf{x}} - \gamma\rho\mathbf{x}_{n+1} \tag{4.11}$$

   and evaluate $f_c = f(\mathbf{x}_c)$. If $f_c \leq f_r$, replace the worst point $\mathbf{x}_{n+1}$ with $\mathbf{x}_c$ and terminate the iteration; otherwise, go to Step 5.

   b. **Inside.** If $f_r \geq f_{n+1}$, perform an *inside contraction*:

   $$\mathbf{x}_{cc} = \bar{\mathbf{x}} - \gamma(\bar{\mathbf{x}} - \mathbf{x}_{n+1}) = (1 - \gamma)\bar{\mathbf{x}} + \gamma\mathbf{x}_{n+1} \tag{4.12}$$

   and evaluate $f_{cc} = f(\mathbf{x}_{cc})$. If $f_{cc} < f_{n+1}$, replace the worst point $\mathbf{x}_{n+1}$ with $\mathbf{x}_{cc}$ and terminate the iteration; otherwise, go to Step 5.

5. **Shrink.** Evaluate f at the n points

$$\mathbf{v}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_i - \mathbf{x}_1),\ i = 2, \ldots, n+1. \tag{4.13}$$

The unordered vertices of the simplex at the next iteration consists of $\mathbf{x}_1, \mathbf{v}_2, \ldots \mathbf{v}_{n+1}$.

In the case where some of the vertices have the same function value, we employ the following tie-breaking rules:

Non-Shrink Ordering Rule. When a non-shrink step (Steps 1 to 4) occurs, the worst vertex $\mathbf{x}_{n+1}$ is discarded. The accepted point created during the iteration, denoted by $\mathbf{v}$, becomes a new vertex and takes position $j+1$ where

$$j = \max_{0<l<n} \{l | f(\mathbf{v}) < f(\mathbf{x}_{l+1})\}$$

and all other vertices retain their relative ordering from the previous iteration.

Shrink Ordering Rule. When a shrink step (Step 5) occurs, the only vertex that remains from the previous iteration is the best point $\mathbf{x}_1$. If $\mathbf{x}_1$ and one or more of the new points are tied as the best point, keep $\mathbf{x}_1$ as the best point, and order the rest of the vertices using the Non-Shrink Ordering Rule above.

The Nelder-Mead method attempts to find a new point that is better than the worst point at each iteration. The resulting points are then arranged from best to worst and a new iteration takes place. The worst point is improved after each iteration. The best point is improved if the expand step (Step 3) occurs.

The algorithm terminates when either 1) the radius of the simplex is less than a prescribed threshold, or 2) the difference between the value of the cost function at the best point and the worst point is less than a threshold value. It can be seen from the algorithm that in the worst case, i.e. a shrink step, each iteration require $n+3$ function evaluations, and $n+2$ function values need to be stored in memory at any given time.

4.5 Summary

The objectives of the optimization problem are presented. Three different cost functions are proposed to compare the performances of controllers. A summary of an attempt to determine the derivative of the Preisach model and the PID-controller is given. Several direct search methods are discussed and the Nelder-Mead simplex algorithm is chosen. The advantages of the Nelder-Mead method and the algorithm description are presented. We now have the tools needed to solve the optimal control problem.

Chapter 5

Simulation

In this chapter, the simulation setup for the optimization problem is presented. Different numerical methods are compared to reduce the simulation time of the model.

5.1 Simulation Setup

The schematic of the simulation program is shown in Figure 5.1. The reference signal $r(t)$ is defined by

$$r(t) = \begin{cases} q, & (i-1)\tau < t \leq \frac{i\tau}{2} \\ 0, & \frac{i\tau}{2} < t \leq i\tau \end{cases} \tag{5.1}$$

where q is the magnitude of the reference signal, $i = 1...n$, n is the number of cycles and τ is the period of the reference signal.

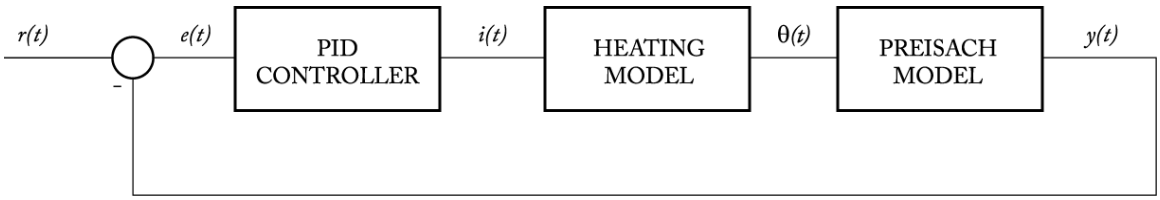

Figure 5.1: Simulation Schematic

The PID-controller is given by

$$i(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{\mathrm{d}e}{\mathrm{d}t}$$

where $e(t) = r(t) - y(t)$, $y(t)$ is the wire contraction measured in millimeters. The integral term is calculated using the trapezoid rule, whereas the derivative is a simple finite difference approximation. The controller gain values are parameters to be optimized. The gains are restricted to have positive values.

The control signal input $i(t)$ is electric current. The heating model used in the simulations is

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{Ri(t)^2 - hA(T(t) - T_{amb})}{\rho c_p V} \tag{5.2}$$

with parameters

ρ density of the wire,

V volume of the wire,

A surface area of the wire,

R electrical resistance of the wire,

h heat convection coefficient to surrounding area,

c_p specific heat of the wire,

T_{amb} ambient temperature of surrounding area.

In equation (5.2), the term $Ri^2(t)$ corresponds to the electrical power generated by the input current. Since this term is always positive, it contributes to the heating of the wire. The second term in the equation corresponds to the exchange of heat with the surrounding area through convection. Since the input to the Preisach simulator is the temperature above ambient and not the absolute temperature, define $\theta(t) = T(t) - T_{amb}$. Rewriting equation (5.2) using $\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{\mathrm{d}T}{\mathrm{d}t}$ gives

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{Ri(t)^2 - hA\theta(t)}{\rho c_p V} \tag{5.3}$$

Table 5.1: FOD Surface Fit Data

x_2	0.0483
x_3	-47.2120
x_4	0.1041
x_5	-23.7060
$\bar{u}$	174.1
$\underline{u}$	0.0
$y(\bar{u}, \bar{u})$	12.54
$y(\bar{u}, \underline{u})$	0.0

Table 5.2: NiTi Wire Parameters at 25°C

Length	380mm
Diameter	0.3048mm
Density	6500kg/m^3
Heat Capacity (Full Martensite or Austenite)	460J/Kg°C
Resistance (Austenite)	4.5Ω
Resistance (Martensite)	5Ω
Current Saturation	1A
Load	2 kg
Max. Contraction	12.54mm

The Preisach simulator was described in Sections 2.2 and 2.5. The output of Preisach simulator is the wire contraction in millimetres. The experimental data used in the simulator is taken from [14]. A surface is fit onto the FOD data and is given by the following equation [14]:

$$\tilde{F}(\alpha, \beta) = c_1 \frac{e^{-x_4(\underline{u}+x_5)} - e^{-x_4(\beta+x_5)}}{[1 + e^{-x_2(\alpha+x_3)}][1 + e^{-x_4(\beta+x_5)}]} + y(\bar{u}, \underline{u}) \tag{5.4}$$

with

$$c_1 = [y(\bar{u}, \bar{u}) - y(\bar{u}, \underline{u})] \frac{[1 + e^{-x_2(\bar{u}+x_3)}][1 + e^{-x_4(\bar{u}+x_5)}][1 + e^{-x_4(\underline{u}+x_5)}]}{[1 + e^{-x_4(\underline{u}+x_5)}][e^{-x_4(\underline{u}+x_5)} - e^{-x_4(\bar{u}+x_5)}]}$$

The surface parameters are given in Table 5.1.

The parameters of the NiTi wire are listed in Table 5.2. The constant-load one wire

actuator setup described in Section 2.2 is used. The electrical resistance and heat capacity of the wire varies as the wire temperature changes. The electrical resistance is different for the martensite and austenite phases. The resistance of the wire can be approximated using *phase fraction* information. The martensite phase fraction is the fraction of the wire that is in the martensite phase. Since the wire only has two phases, the sum of the austenite and marteniste phase fraction is one. The martensite phase fraction can be approximated by the fraction of the actual wire contraction to the maximum contraction

$$F_a = \frac{y}{y_{max}} \tag{5.5}$$

The resistance of the wire is then approximated by the following equation

$$R = F_a R_a + (1 - F_a) R_m \tag{5.6}$$

where F_m is the martensite phase fraction, R_m and R_a are the martensite and austenite resistances respectively.

The heat capacity is shown to vary with temperature in [4]. At full martensite and full austenite phases, the heat capacity is constant. During phase transformations, part of the energy applied to the wire contributes to the heating, while most of it is used to change the phase of the material. Therefore, the heat capacity of the wire is much higher during phase transformations. In the heating process, the heat capacity is given by [4]

$$c_p = c_0 + H \frac{log(100)}{|A_s - A_f|} e^{-\frac{2 \ln 100}{|A_s - A_f|}\left|T - \frac{A_s + A_f}{2}\right|}, \; A_s \leq T \leq A_f \tag{5.7}$$

where c_0 is the heat capacity given in Table 5.2, H is the latent heat, A_s and A_f are the starting and ending phase transition temperatures for austenite. During cooling, the heat capacity is given by [4]

$$c_p = c_0 + H \frac{log(100)}{|M_s - M_f|} e^{-\frac{2 \ln 100}{|M_s - M_f|}\left|T - \frac{M_s + M_f}{2}\right|}, \; M_f \leq T \leq M_s \tag{5.8}$$

where M_s and M_f are the starting and ending phase transition temperatures for martenite. The parameter values used to determine the electrical resistance and heat capacity are listed in Table 5.3.

Table 5.3: Resistance and Heat Capacity Parameters

A_s	53°C
A_f	80°C
M_s	47°C
M_f	19°C
C_0	460J/Kg°C
R_a	4.5Ω
R_m	5Ω
Maximum Wire Contraction	12.54mm

5.2 Numerical Implementation

The MATLAB routine *'fminsearch'* is used to determine the optimal PID-controller values. The Nelder-Mead algorithm parameter values used in *'fminsearch'* are

$$\rho = 1,\ \chi = 2,\ \gamma = \frac{1}{2},\ \text{and}\ \sigma = \frac{1}{2}. \tag{5.9}$$

Since the model does not involve an active cooling component, the input signal is restricted to be positive. The heat equation (5.3) involves the square of the input current. A negative input current will not cause the temperature to decrease since the cooling mechanism is through heat convection with the surrounding area and is not controlled by the input current. The heat equation along with the Preisach simulation were first evaluated using the standard MATLAB routine *'ode45'*. It is a variable time step method that uses the Runge-Kutta method. Two problems occurred when this method was used. First, the simulation run time for this routine were large. It is not desirable when the optimization routine requires many iterations.

The second problem is due to the step size change in the variable time step method. In a variable time step method, an initial step size is chosen. Then, the differential equation is solved using this step size. The resulting function value is compared with the value obtained in the previous time step. If the change in the function value is too low, the iteration is repeated with a bigger step size. Similarly, if the difference is too large, a smaller step size is used to adapt to the high frequency behaviour. The current time step ends when the change in function value falls within a threshold interval.

Before moving on to the next time step, a number of function evaluations are required. For each of these evaluations, the Preisach simulator is called and the reduced memory sequence in the simulator is updated. Let s_i denote the reduced memory sequence at the start of the i^{th} time step. Assume that the step size is set to be Δ. The numerical method calculates the resulting temperature T_i of this time step. In the process, the Preisach simulator is called with T_i and it is added to s_i to form a new sequence $\bar{s}_i$.

If the numerical method decides that a change in step size is required, a new temperature $\bar{T}_i$ is calculated using a different step size $\bar{\Delta}$. This new temperature is then fed to the Preisach simulator. The new input $\bar{T}_i$ is added to $\bar{s}_i$ because the memory in the Preisach simulator is not reset. The correct reduced memory sequence should have been found by adding $\bar{T}_i$ to s_i, the original reduced memory sequence at the beginning of this iteration. Further changes of the step size within this iteration will further affect the reduced memory sequence. Since the outputs corresponding to two different reduced memory sequences are not necessarily equal, the variable time step method will cause error in the Preisach simulator output.

Since the temperature does not vary rapidly, simpler lower-order numerical methods can be used. Time step sizes are predetermined to prevent unwanted Preisach simulator evaluations. Two numerical methods are chosen to evaluate the heat equation (5.3): the 4^{th}-order Runge-Kutta method and finite differences. Two different time-step size configurations were investigated using the above methods.

The 'RK4' scheme uses the Runge-Kutta method with a fixed time step size of 0.001s. It is observed that the temperature changes more during the first second after the switching time of the reference signal. During the remaining time of the period, including the cooling phase, the temperature changes more gradually. A simple variable time step scheme is used. For the first second after the switching time, a fixed step size of 0.001s is used. A fixed step size of 0.01s is used for the remaining time of the period. Different numerical methods using this time-step scheme were programmed and tested. 'V-RK4' and 'V-FD' uses the variable time step scheme with the Runge-Kutta method and finite difference methods respectively. 'V-RKFD' uses the Runge-Kutta method for the first second and finite difference method for the remaining time using the variable time step scheme.

The simulation run times and tracking error with different controller gains are summa-

Table 5.4: Simulation Time-step Scheme Comparison

Method	Trial 1 Time (s)	Trial 1 Error		Trial 2 Time (s)	Trial 2 Error		Trial 3 Time (s)	Trial 3 Error
RK4	183.81	0.38053		69.859	0.41316		169.16	1088.4
V-RK4	74.938	0.38356		26.297	0.41668		47.375	1089.2
V-FD	33.781	0.38386		23.687	0.41689		34.312	1088.2
V-RK4	42.438	0.38357		28.921	0.41809		41.547	1089.1

rized in Table 5.4. The tracking error is calculated using equation (4.2). It is true that two signals with similar tracking error may be very different. Looking at the actual output responses confirms that they are indeed similar signals. Figure 5.2 shows the wire contraction and temperature responses for 'RK4' and 'V-FD' for the same set of controller gains. The two responses are visually indistinguishable. It can be seen from Table 5.4 that the tracking error for a given set of controller gains is similar for the four numerical schemes. The 'V-FD' scheme has the shortest simulation time in each of the trials, therefore, this scheme is chosen.

The 'V-FD' scheme is used because it has the fastest run-time for different controller gain values. The tracking error with this scheme is also very consistent with the other methods. Since the temperature fluctuations are small, a simple numerical method is sufficient. The optimizations are implemented in MATLAB and the results are given in the next chapter.

5.3 Summary

The simulation setup for the optimization problem was presented in this chapter. Different numerical methods were compared to reduce the simulation time of the model. The optimizations are performed and results are presented in Chapter 6.

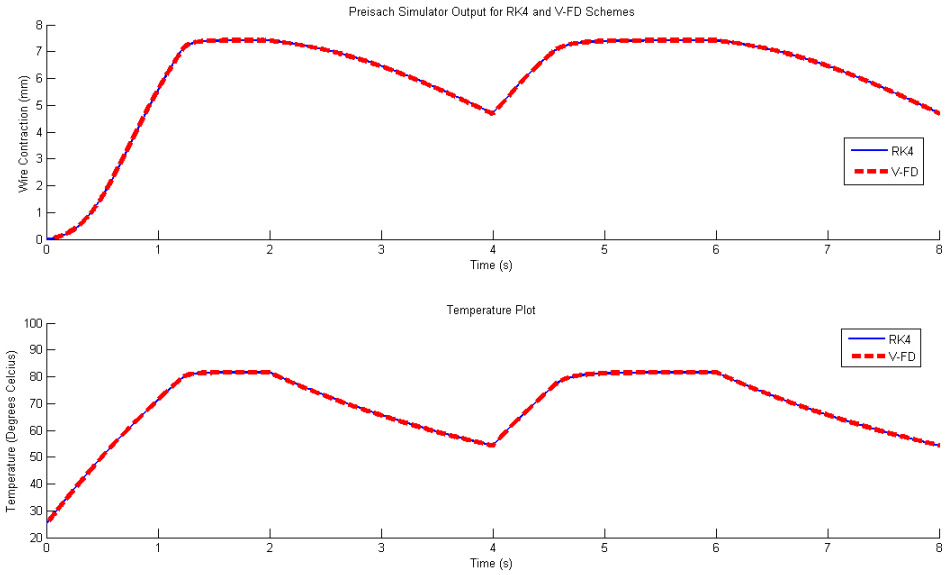

Figure 5.2: Wire Contraction and Temperature responses for RK4 and V-FD Schemes

Chapter 6

Optimal Controllers and Bandwidth

In this chapter, the optimization results are presented for the three cost functions defined in Section 4.1. For each cost function, an optimal controller is found for each design frequency. Optimization is performed using different initial values, and the best result is used. Each optimal controller is then used for the different frequencies and the tracking error is noted.

The constant-load SMA wire actuator configuration is used. The wire contraction is the measured output and is modelled using the Preisach model. An Nitinol wire with an un-stretched length of 380mm is used. The varying electrical resistance and heat capacity of the wire is modelled using equations (5.6) to (5.8). An input saturation of 1A is used according to the wire specifications. The input current is also saturated at 0A to take into account for the lack of cooling mechanisms in the system. The parameters of the NiTi wire are listed in Table 5.2.

The reference signal used is defined by

$$r(t) = \begin{cases} 8, & (i-1)\tau < t \leq \frac{i\tau}{2} \\ 0, & \frac{i\tau}{2} < t \leq i\tau \end{cases} \tag{6.1}$$

where $i = 1...n$, n is the number of cycles, τ is the period of the reference signal. The amplitude of 8mm corresponds to approximately 2.1% strain of the SMA wire.

Table 6.1: Optimal Controllers using J_1

Reference Frequency (Hz)	K_p	K_i	K_d	Normalized Error e_n
Step	4.5228	0.094519	0.0013198	0.0384
0.02	11.1	4.7152	0	0.07496
0.05	6.8696	4.1866	5.0807	0.1896
0.0625	3.8741	5.7286	0.00028438	0.2035
0.08	0.116	0.2216	0	0.2248
0.125	18.684	7.8235	0	0.5262
0.25	1.1477	0.4265	1.3460	0.7228

6.1 Output Tracking

For the output tracking problem, the following cost function is used:

$$J_1(K_p, K_i, K_d) = \int_0^T (r(t) - y(t))^2 dt \tag{6.2}$$

where

$$\begin{aligned} y(t) &= \text{wire contraction} \\ r(t) &= \text{desired contraction} \\ T &= \text{length of control time interval} \end{aligned}$$

The reference signal is given in equation (6.1). The reference signal frequencies are 0.02, 0.05, 0.0625, 0.08, 0.125, and 0.25Hz.

The optimal controller for each design frequency is given in Table 6.1. The normalized error is calculated using the following formula:

$$e_n = \frac{\int_0^T (r(t) - y(t))^2 dt}{\int_0^T (r(t))^2 dt} \tag{6.3}$$

The denominator corresponds to the tracking error of a zero output or the energy of the reference signal. The duration of each simulation is different to ensure that each reference signal has a 50% duty cycle. Therefore, the errors have to be normalized to take into

account the different energies of each reference signal. Each optimal controller is then used for the different frequencies and the tracking error is recorded. The normalized error is shown in Figure 6.1.

As expected, each optimal controller has the lowest normalized error at the frequency for which it is optimized. For the controller optimized using the the 0.02Hz reference signal, the error is much higher than the controllers optimized at higher frequencies. It is an expected result since the controllers are not optimized to deal with the rapid change of the input signal.

The controller optimized at 0.08Hz seems to be an exception to the above observation. While this controller has high error values between 0.125 and 0.333Hz, the performance of this controller is similar to the controllers that were optimized for high frequencies. This may be a result of multiple near optimal solutions. In some cases, using different initial values, the different optimized controller gains achieve similar error values at the optimization frequency. While they have similar error values at one frequency, the different gain values may have an effect on the error at a different frequency.

The step-optimized controller has good error values for low frequencies and high error values at high frequencies. Optimizing at each frequency does improve the performance of the system. The 0.02Hz controller performs better than the step controller at low frequencies. At higher frequencies, the 0.25Hz controller gives better performance than the step controller.

On the other hand, controllers optimized for higher frequency have higher error for low frequencies. Furthermore, one would expect the error would vary continuously with respect to the optimization frequency. However, at 0.25Hz, the controller optimized at 0.05Hz has a lower error than controllers optimized at 0.0625 and 0.02Hz. This could be a result of the optimization converging to a local optimum rather than a global optimum.

Since the Nelder-Mead method does not guarantee convergence, the local behaviour of the cost function near the optimal values are plotted. Figure 6.2 shows that the optimal controllers found in each case are at least local minimum values.

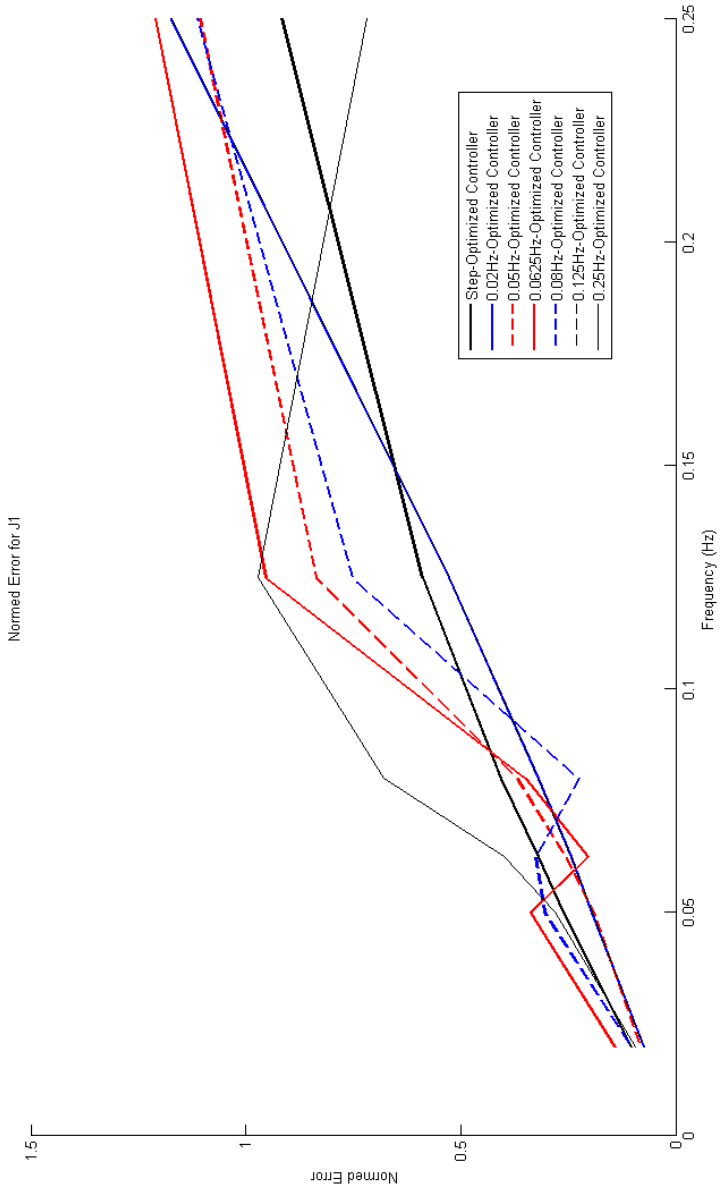

Figure 6.1: Normalized Error for Different Optimal Controllers for J_1

Two level curves of the cost function for the 0.25Hz reference signal are shown in Figures 6.3 and 6.4. The optimal controller is found near the minimum in Figure 6.3 at $K_p = 1.1477$, $K_i = 0.42653$ and $K_d = 1.346$. On the other hand, starting at an initial value of $K_p = K_i = K_d = 5$, the Nelder-Mead method converged to a point $K_p = 5.1667$, $K_i = 5.1667$ and $K_d = 4.75$ on the plateau in Figure 6.4. This solution is clearly not optimal. It is a result of the non-convexity of the solution space. Figure 6.4 shows that the cost function is not convex with two flat areas and a valley.

To improve the 'optimality' of the solutions, different initial values are used and the best solution is recorded. Although global optimality is not guaranteed, all of the optimizations converged to a solution. There are three termination conditions for the optimization method: 1) both the size of the simplex and the function values at the vertices converge within a set radius; 2) a maximum number of iterations is reached; or 3) a maximum number of function evaluations is reached. Using different initial values, all optimizations terminated under condition 1). As shown previously in Figure 6.2, the optimizations converged to local minimum values.

The wire contraction, temperature, and input current plots of each of the optimal controllers at each frequency are shown in Figures 6.5 to 6.10. As previously discussed, the input current is saturated between 0 and 1A. In most cases, there is a lot of chatter in the input current. They also seem to be multi-valued. This is because the data is plotted as points rather than lines. The rapid change of the input current between the two saturation points is likely caused by numerical differentiation. The only responses that do not have input current chattering are the 0.02, 0.08 and 0.125Hz controllers, and they are the only controllers with $K_d = 0$.

Looking at the error values alone does not indicate whether our objective has been met. An output that stays in the middle between the reference values will have a better error than an output that reaches the upper value and not the lower one. Figures 6.10 and 6.11 show the responses of the controllers optimized at 0.25Hz and step response respectively for a 0.25Hz reference signal. Even though the step controller manages to reach the target of 8mm, it fails to return close to 0mm during the cooling phase. The 0.25Hz controller does not reach either of the reference points, but since the signal stayed near the middle, the error is 'balanced' out.

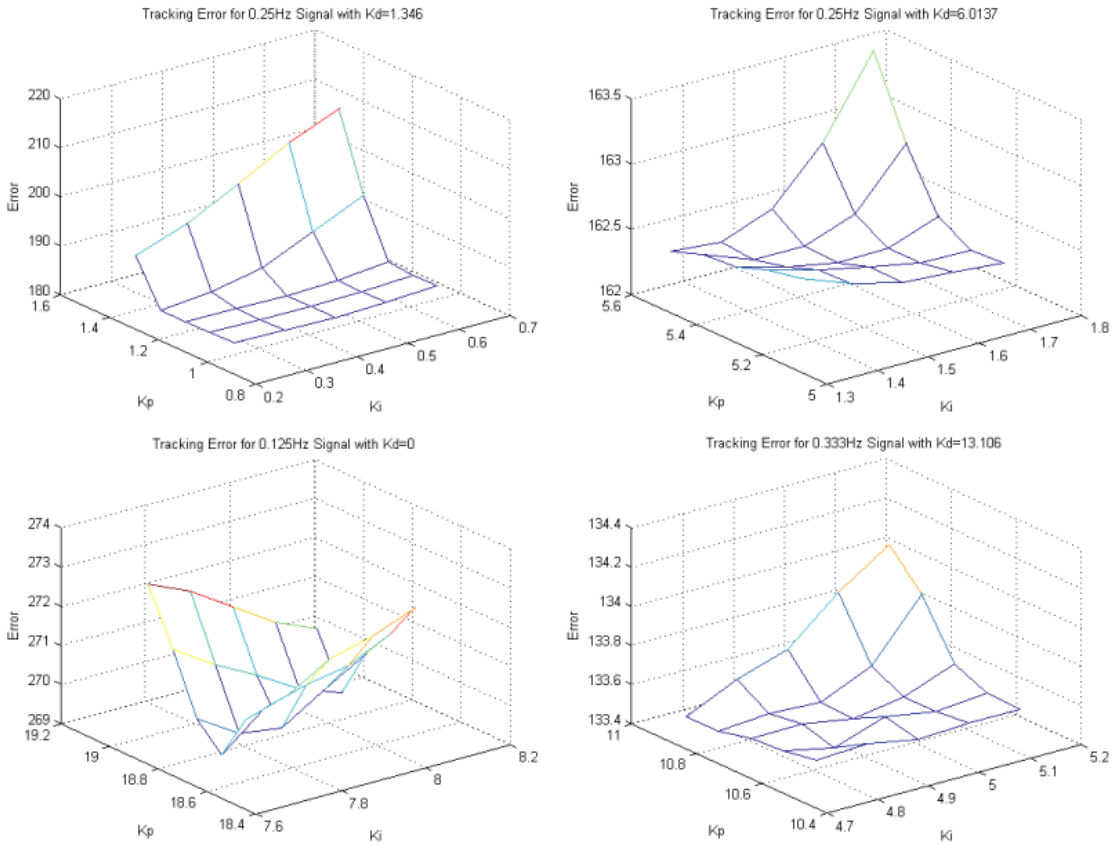

Figure 6.2: Cost Functions Near Optimal Controllers

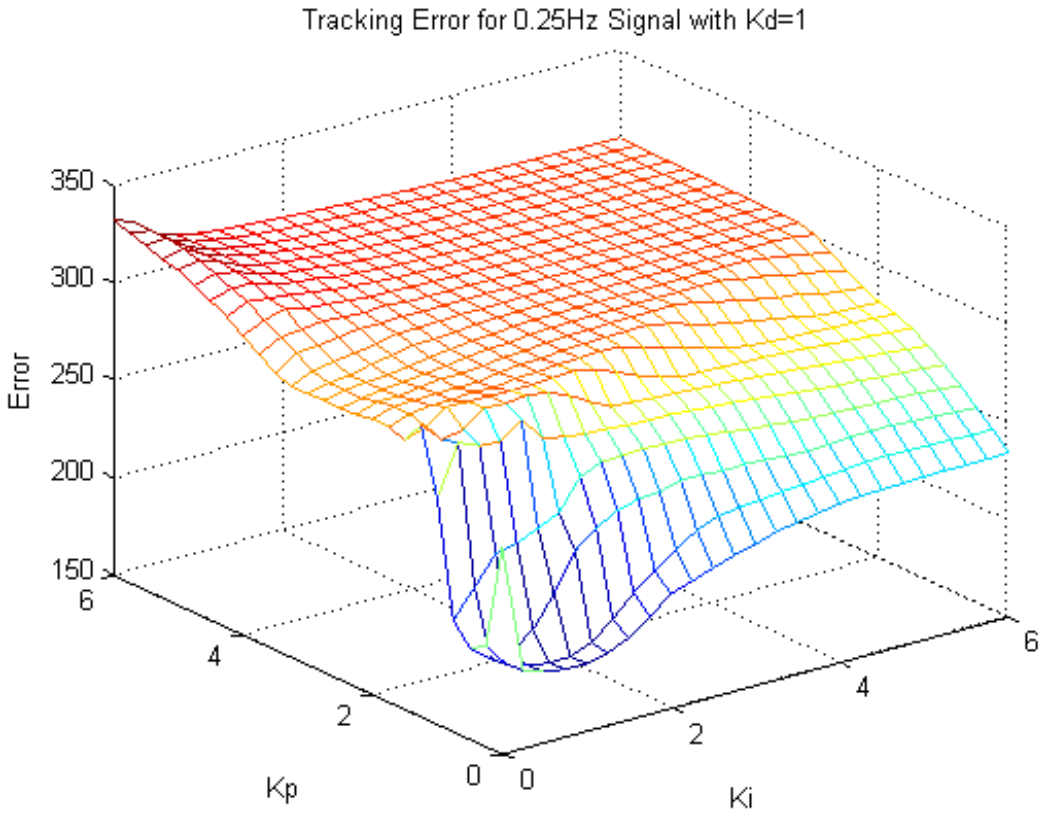

Figure 6.3: Tracking Error for 0.25Hz Reference Signal with K_d=1

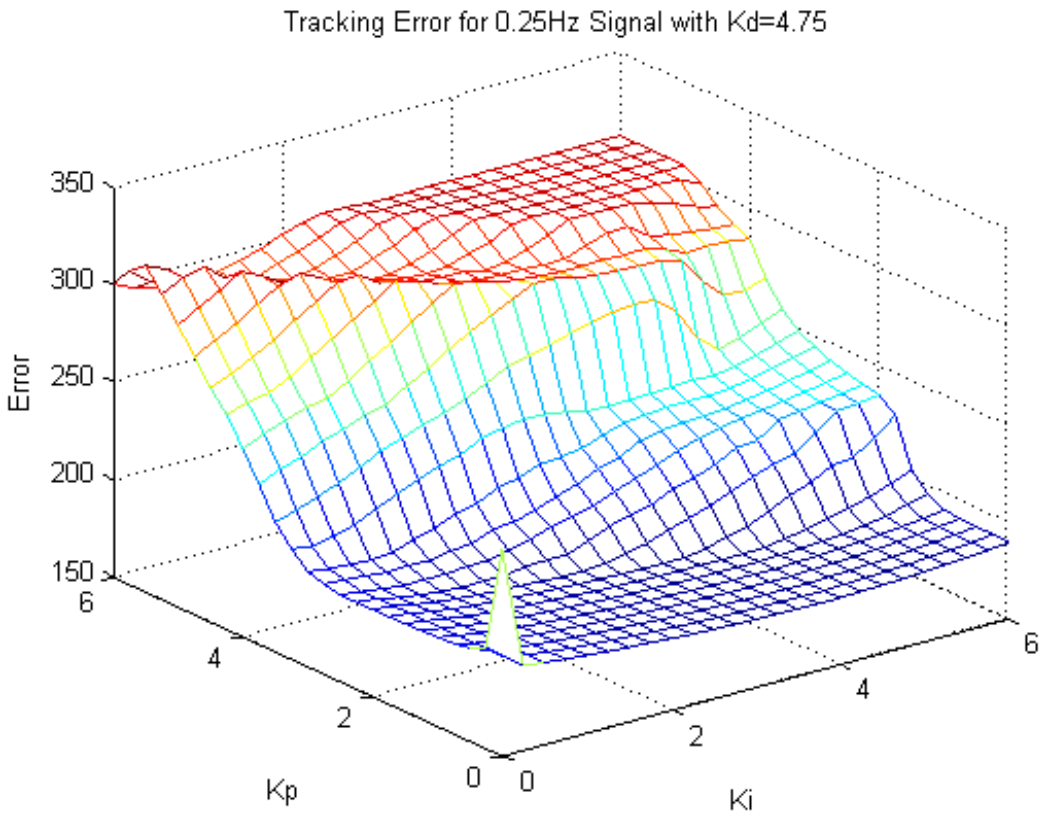

Figure 6.4: Tracking Error for 0.25Hz Reference Signal with K_d=4.75

Since the Preisach model is static, only the heating dynamics govern the response time of the system. The time constant of the heat equation (5.3) is

$$\tau = -\frac{\rho c_p V}{hA}. \tag{6.4}$$

For the SMA wire used in the simulation, $\tau = 3.0378$s at full martensite or austenite. This corresponds to a 98% cooling time of 12.1512s. A cooling phase of 12.1512s translates to a periodic signal of approximately 0.04Hz. Since the heat capacity c_p increases during phase transformations, the cooling time increases as well. Therefore, it is reasonable for the system to have poor performance for higher frequencies.

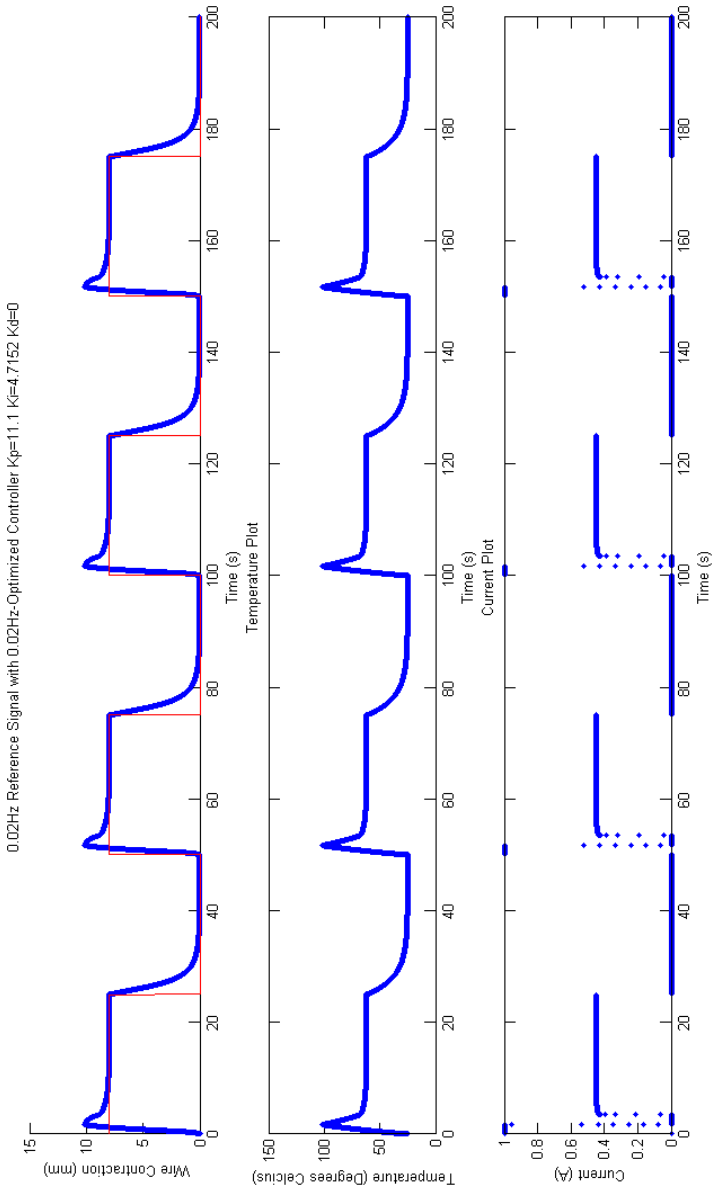

Figure 6.5: 0.02Hz Reference Signal with 0.02Hz-Optimized Controller

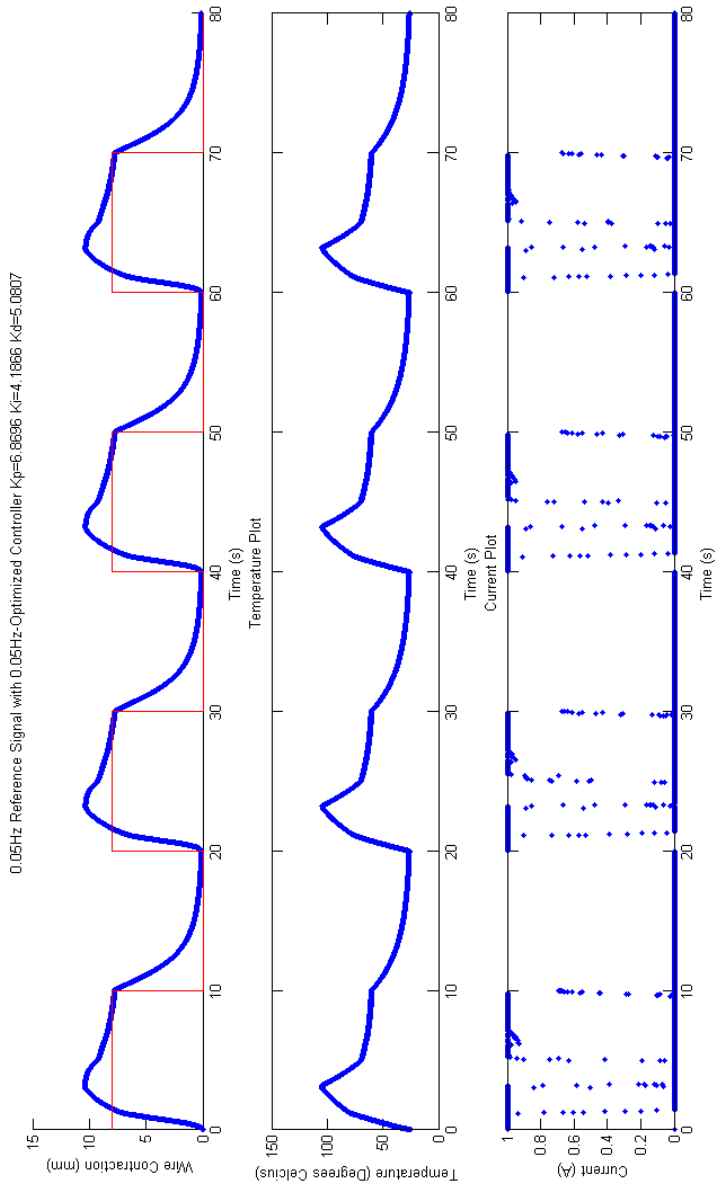

Figure 6.6: 0.05Hz Reference Signal with 0.05Hz-Optimized Controller

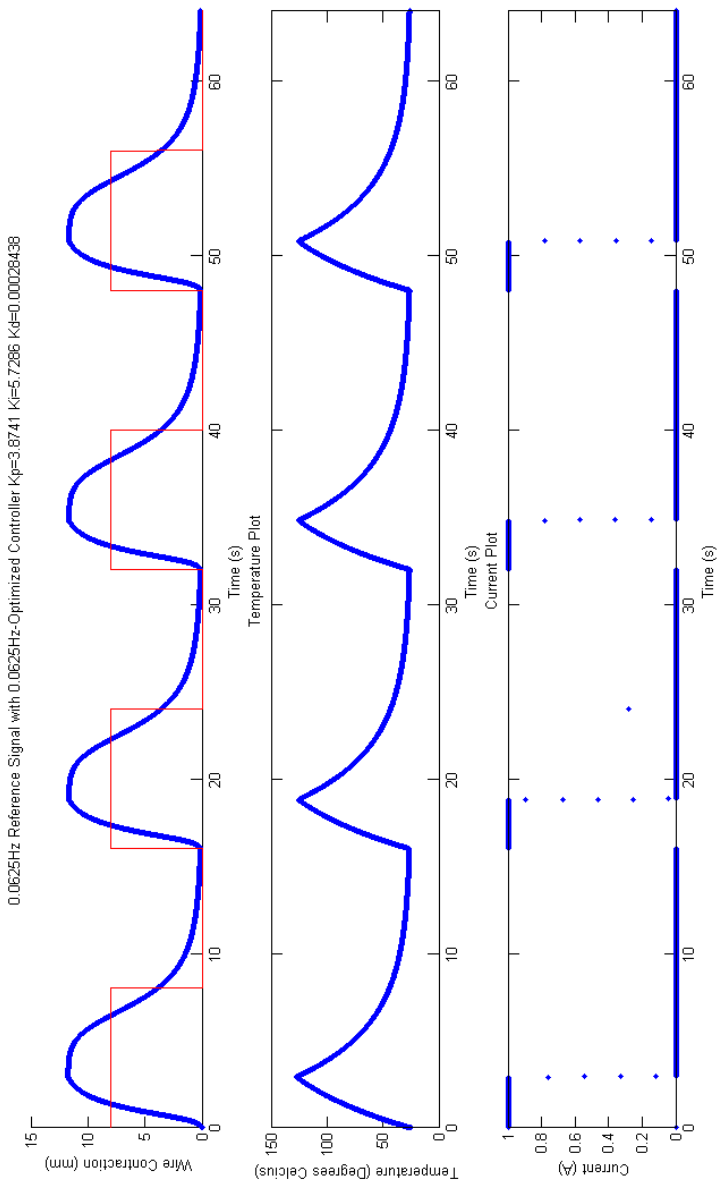

Figure 6.7: 0.0625Hz Reference Signal with 0.0625Hz-Optimized Controller

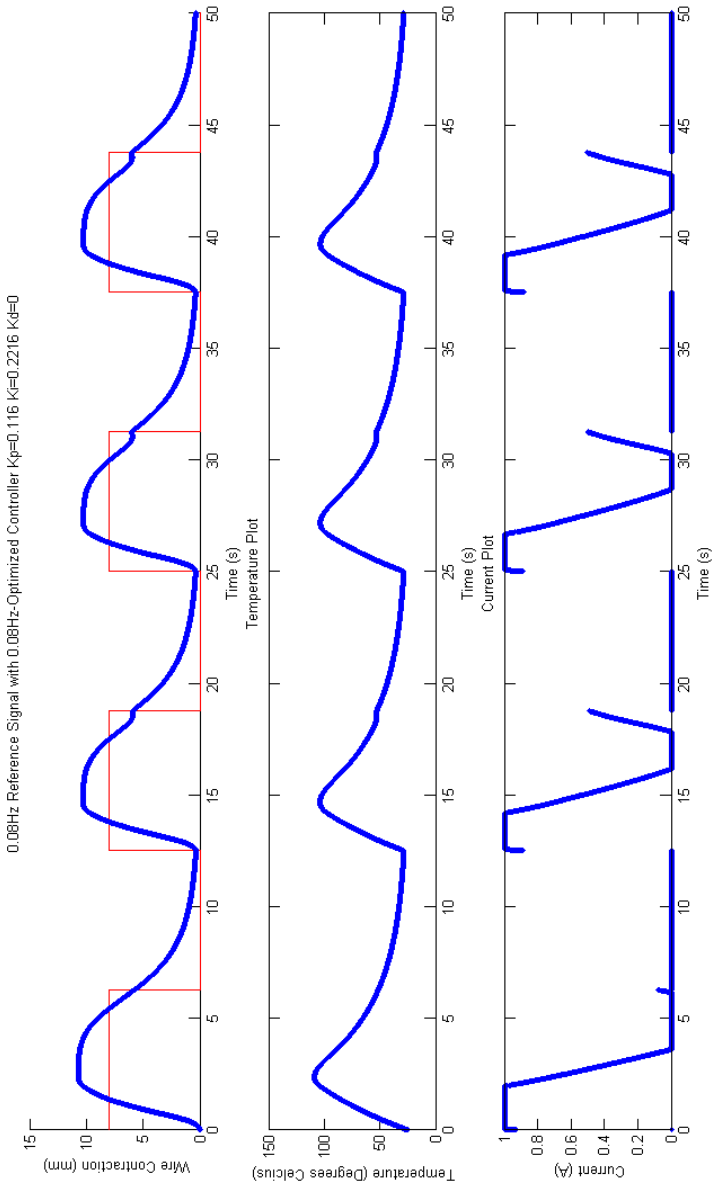

Figure 6.8: 0.08Hz Reference Signal with 0.08Hz-Optimized Controller

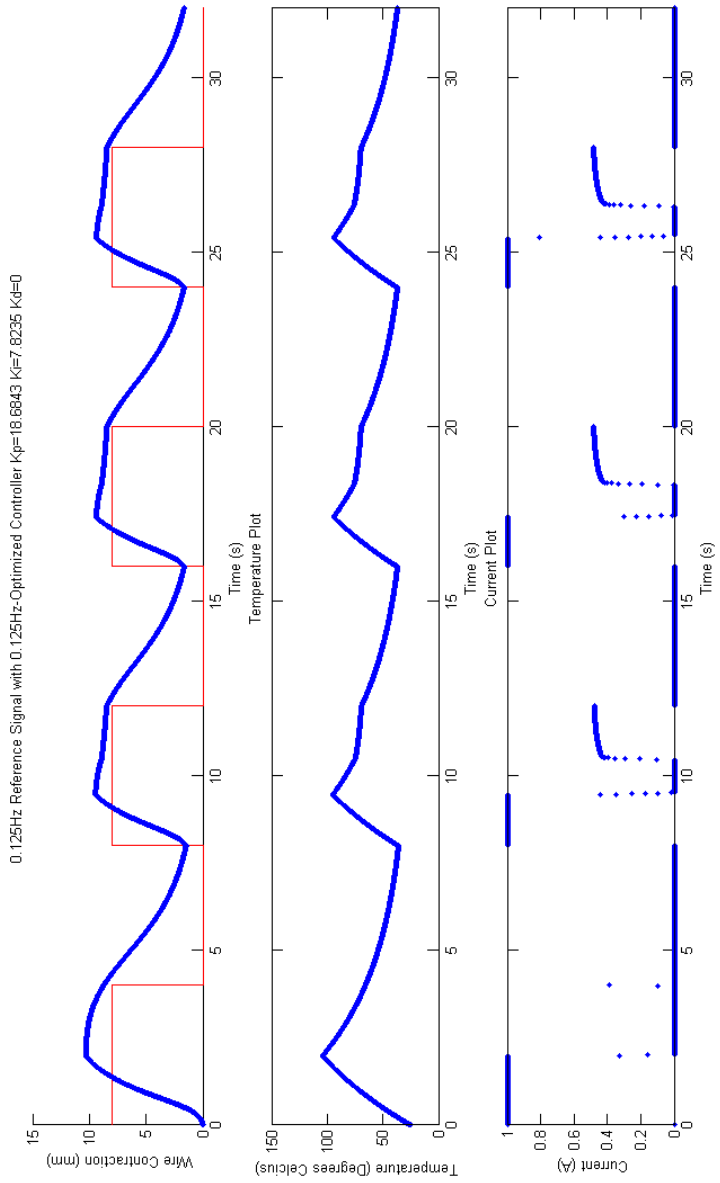

Figure 6.9: 0.125Hz Reference Signal with 0.125Hz-Optimized Controller

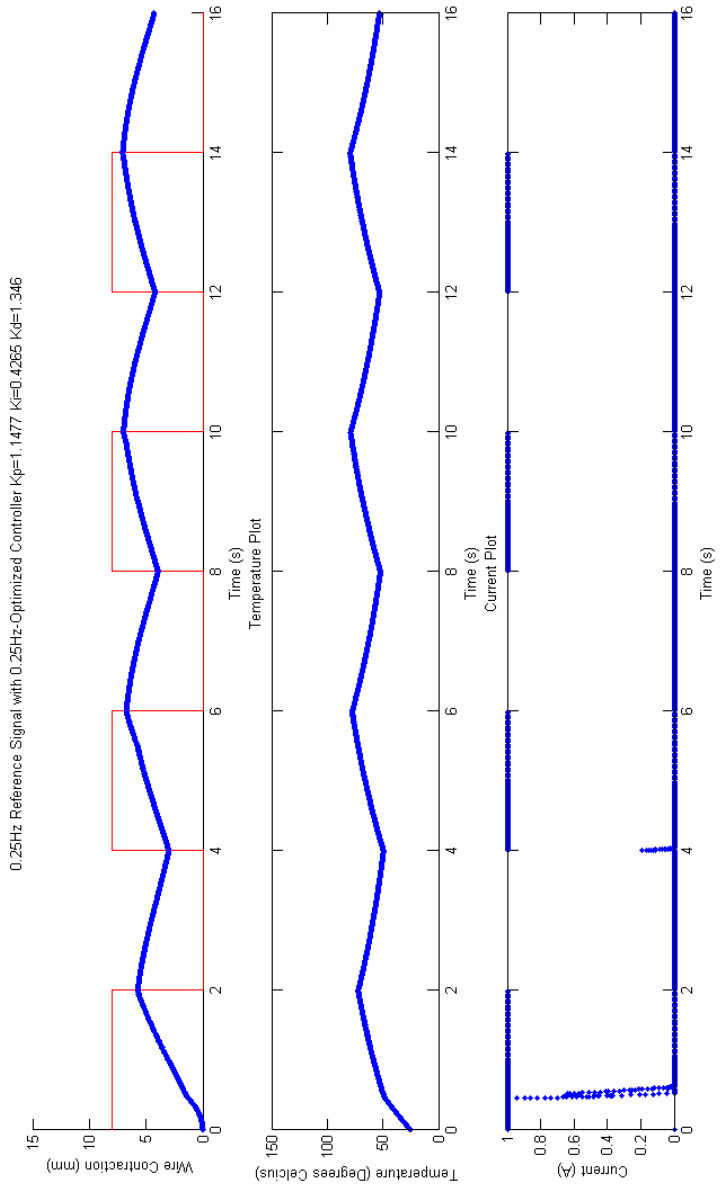

Figure 6.10: 0.25Hz Reference Signal with 0.25Hz-Optimized Controller

The response of the system with the step-optimized controller at 0.02Hz is shown in Figure 6.12. At first glance, it seems that the step-optimized controller has better tracking than the 0.02Hz-optimized controller (Fig 6.5). The two responses are plotted together in Figure 6.13. While the 0.02Hz controller has overshoot, it has a better cooling phase than the step controller. This is because the 0.02Hz controller cools to a lower temperature of approximately 60°C at steady state.

This observation can be illustrated by looking at the input-output maps in Figure 6.14. The black dashed-line represents the major loop of the SMA wire's output hysteresis. The blue and the red line represent the input-output maps of Figures 6.5 and 6.12 respectively. By entering the interior of the major loop, the blue curve (Fig 6.5) is able to achieve the same output values as the red curve (Fig 6.12), but at lower temperatures after the overshoot (Fig 6.14). This is a result of the hysteretic behaviour of the SMA wire. Since the step response does not have a cooling phase, the system only travels along the major loop to reach the target point and tries to stay constant by maintaining the temperature.

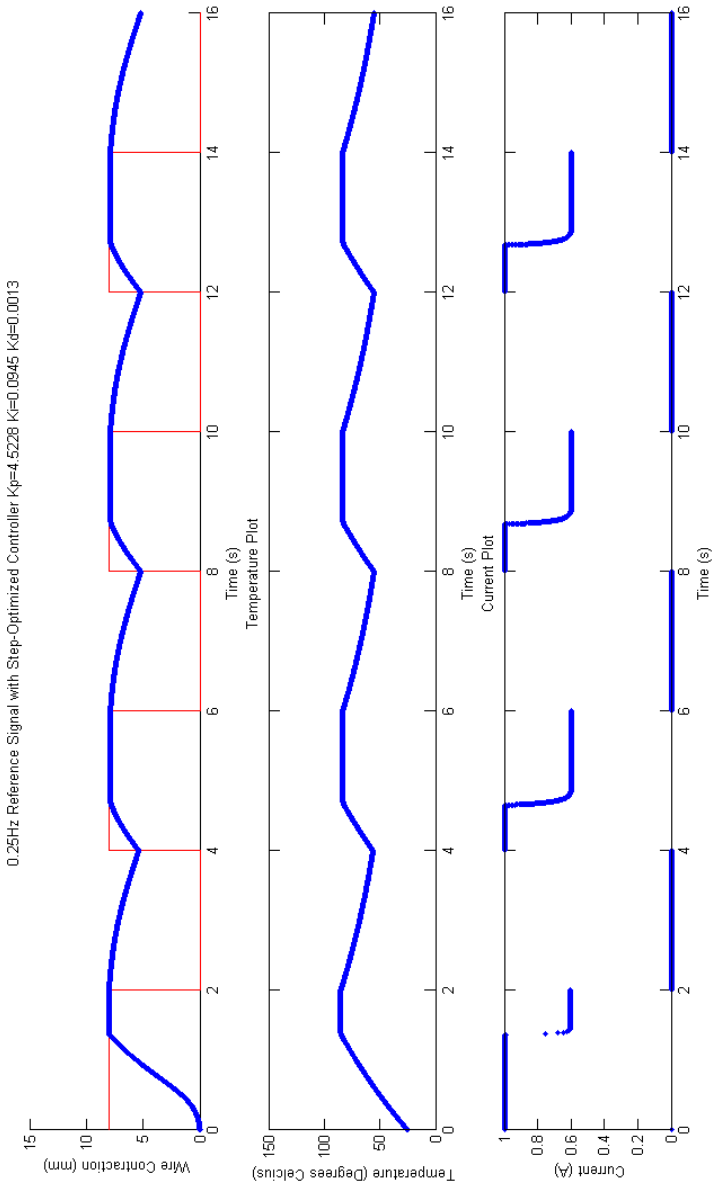

Figure 6.11: 0.25Hz Reference Signal with Step-Optimized Controller

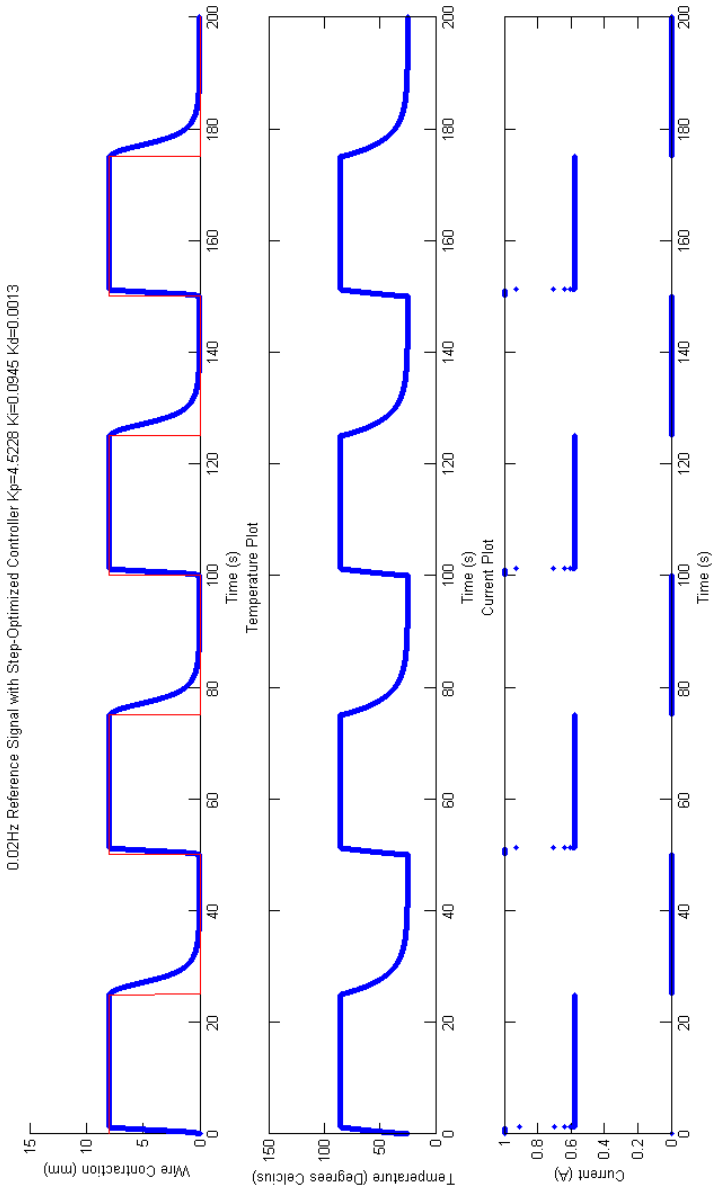

Figure 6.12: 0.02Hz Reference Signal with Step-Optimized Controller

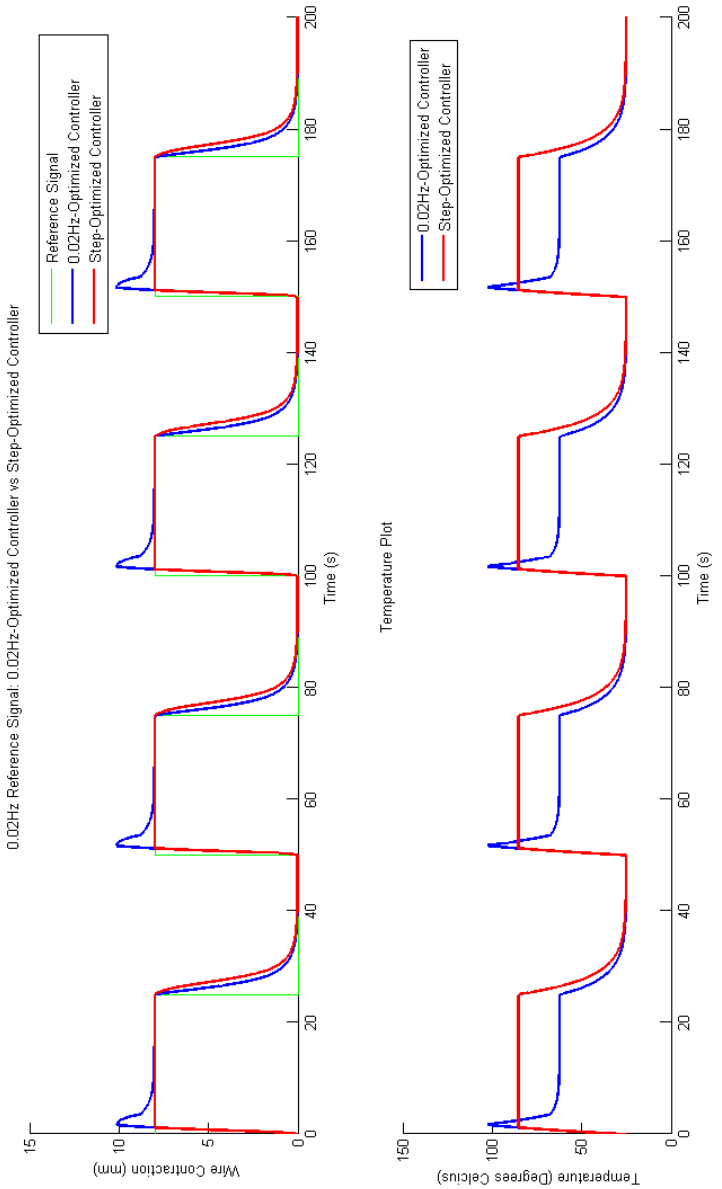

Figure 6.13: 0.02Hz Reference Signal: 0.02Hz-Optimized Controller vs Step-Optimized Controller

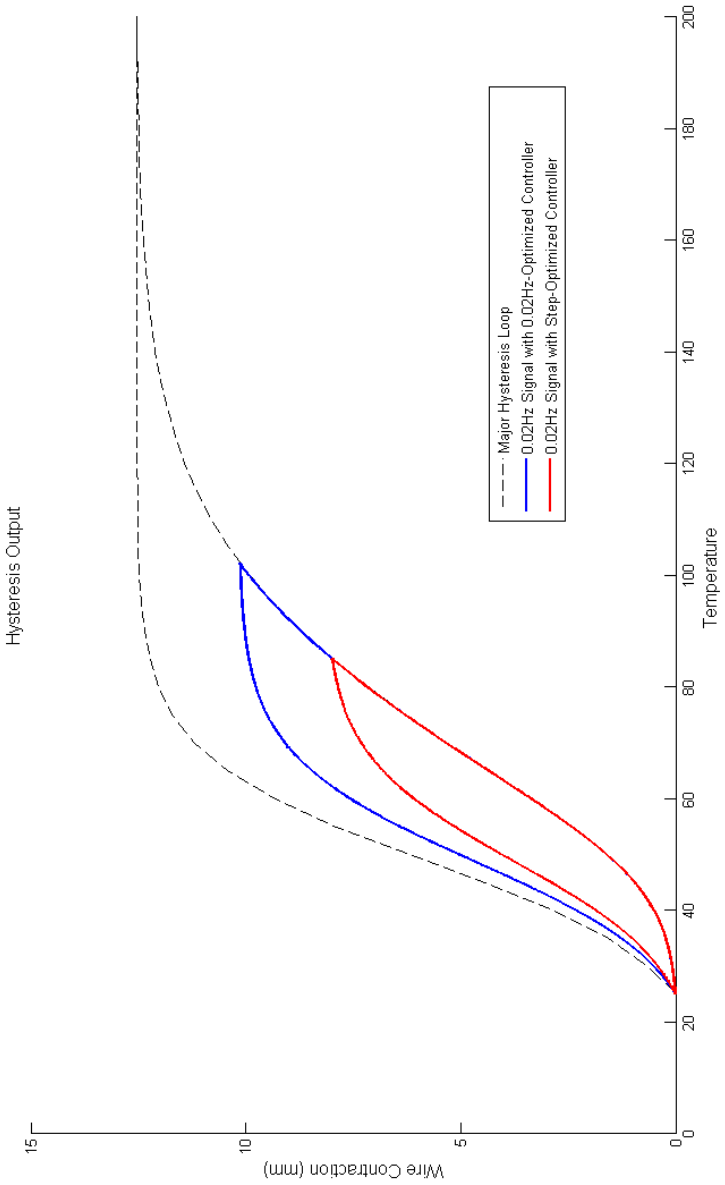

Figure 6.14: Input-Output Map Corresponding to Figures 6.5 and 6.12

Table 6.2: Optimal Controllers using J_2

Reference Frequency (Hz)	K_p	K_i	K_d	Normalized Error e_n
0.02	5.9967	3.1898	0.00051125	0.0000
0.05	7.9383	4.4434	2.0528	0.0002
0.0625	5.9807	4.175	5.0342	0.0010
0.08	10.6033	9.2409	10.5912	0.0035252
0.125	13.8728	5.8458	0	0.0442
0.25	1.1475	0	1.928	0.2443

6.2 Point Tracking

For the point tracking problem, the following cost function is used:

$$J_2(K_p, K_i, K_d) = \sum_{i=1}^{n} |y_s^i - r_s^i|^2 \tag{6.5}$$

where

$$\begin{aligned} y_s^i &= \text{contraction at } i^{th} \text{ switch time} \\ r_s^i &= \text{desired contraction at } i^{th} \text{ switch time} \\ n &= \text{number of switching times} \end{aligned}$$

The reference signal is given in equation (6.1). The reference signal frequencies are 0.02, 0.05, 0.0625, 0.08, 0.125, and 0.25Hz.

The optimal controller for each design frequency is given in Table 6.2. The normalized error is calculated using the following formula:

$$e_n = \frac{\sum_{i=1}^{n} |y_s^i - r_s^i|^2}{\sum_{i=1}^{n} |r_s^i|^2} \tag{6.6}$$

The denominator corresponds to the tracking error of a zero output. Each optimal controller is then used for the different frequencies and the tracking error is recorded. The normalized error is shown in Figure 6.15.

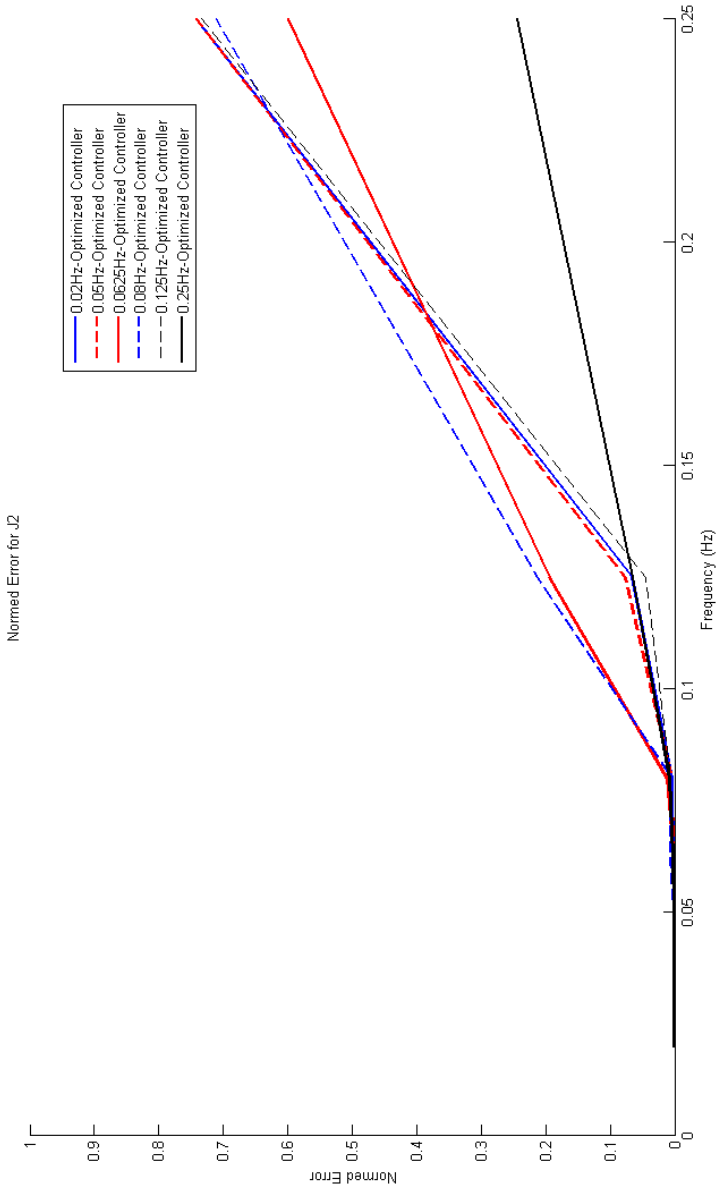

Figure 6.15: Normalized Error for Different Optimal Controllers for J_2

A few observations can be made from Figure 6.15. For low frequency signals, the error for the controllers optimized at lower frequencies are smaller. On the other hand, for high frequency signals, the error for the controllers optimized at higher frequencies are smaller. This result suggests that a controller optimized at a frequency f will also have good performance for a range of frequencies near f.

One might think that for a fixed reference frequency f, the performance of a controller A optimized at a frequency f_A will be better than controller B optimized at f_B if $|f_A - f| < |f_B - f|$. That is to say that the controller optimized at 0.08Hz will have a higher error than the controller optimized at 0.125Hz for a reference signal of 0.25Hz. Figure 6.15 shows otherwise. Although it is true that the controllers each have a certain bandwidth where they have good performance, there is no apparent rule to rank their performances for a given frequency. This may be a result of the sub-optimality of the controllers found. If the global optimal solutions are guaranteed to be found, then they may result in the expected ordering. The controller optimized at 0.25Hz has low errors at each tested frequency. We can say that this controller has the best bandwidth among the sample set.

The wire contraction, temperature, and input current plots of each of the optimal controllers at each frequency are shown in Figures 6.16 to 6.21. The controllers optimized at 0.02 (Fig 6.16), 0.05 (Fig 6.17) and 0.0625Hz (Fig 6.18) give good performance for the cost function J_2. This is a result of the cooling time of the system. As the frequency increases, the output of the system can only manage to stay 'near the middle' to achieve the best error values possible. Therefore, the SMA wire is not suitable for this application at frequencies higher than 0.125Hz.

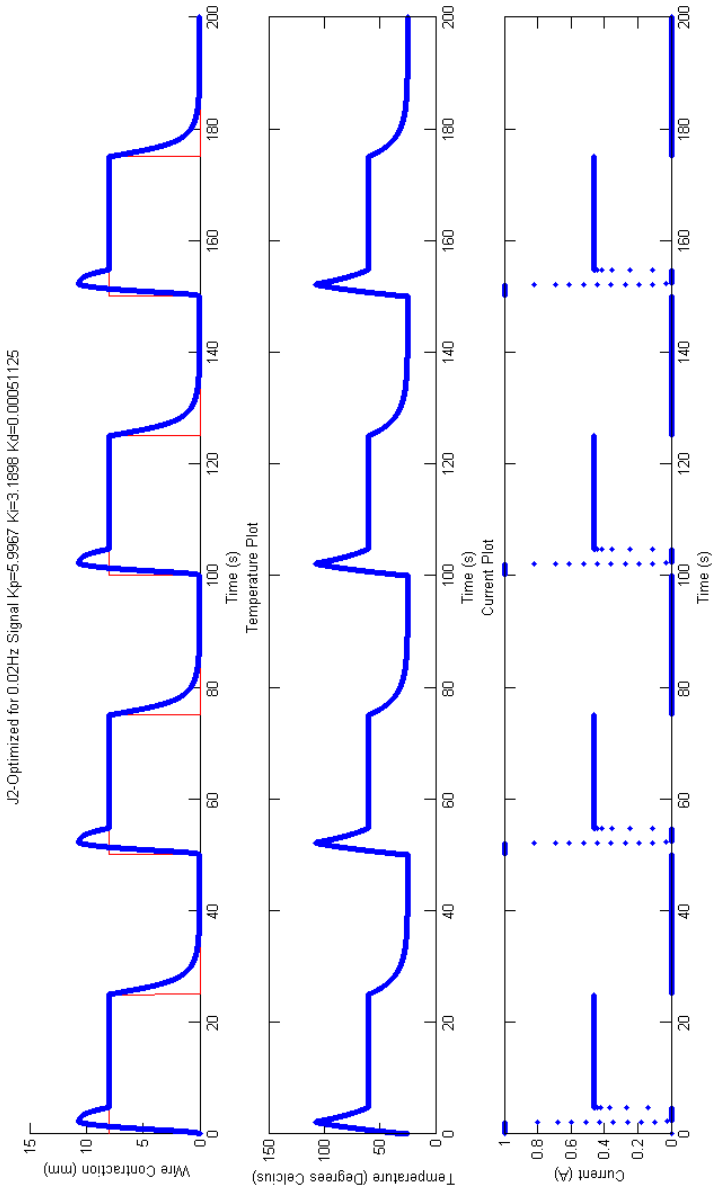

Figure 6.16: J2-Optimized for 0.02Hz Signal

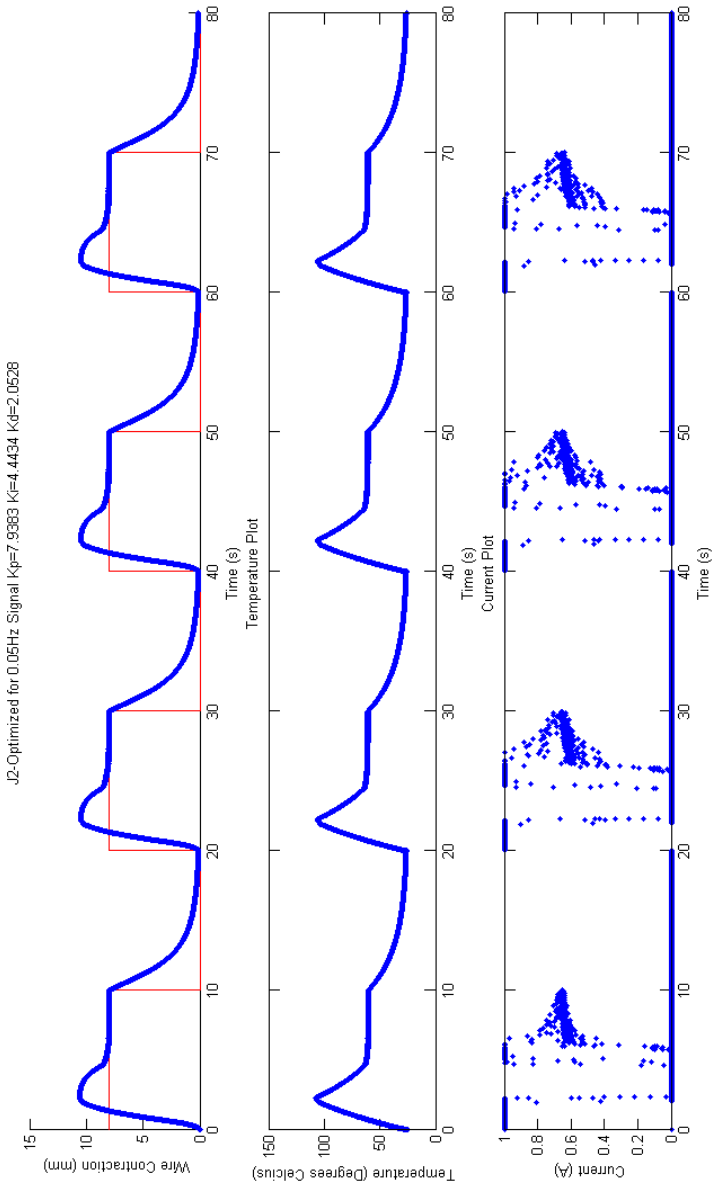

Figure 6.17: J2-Optimized for 0.05Hz Signal

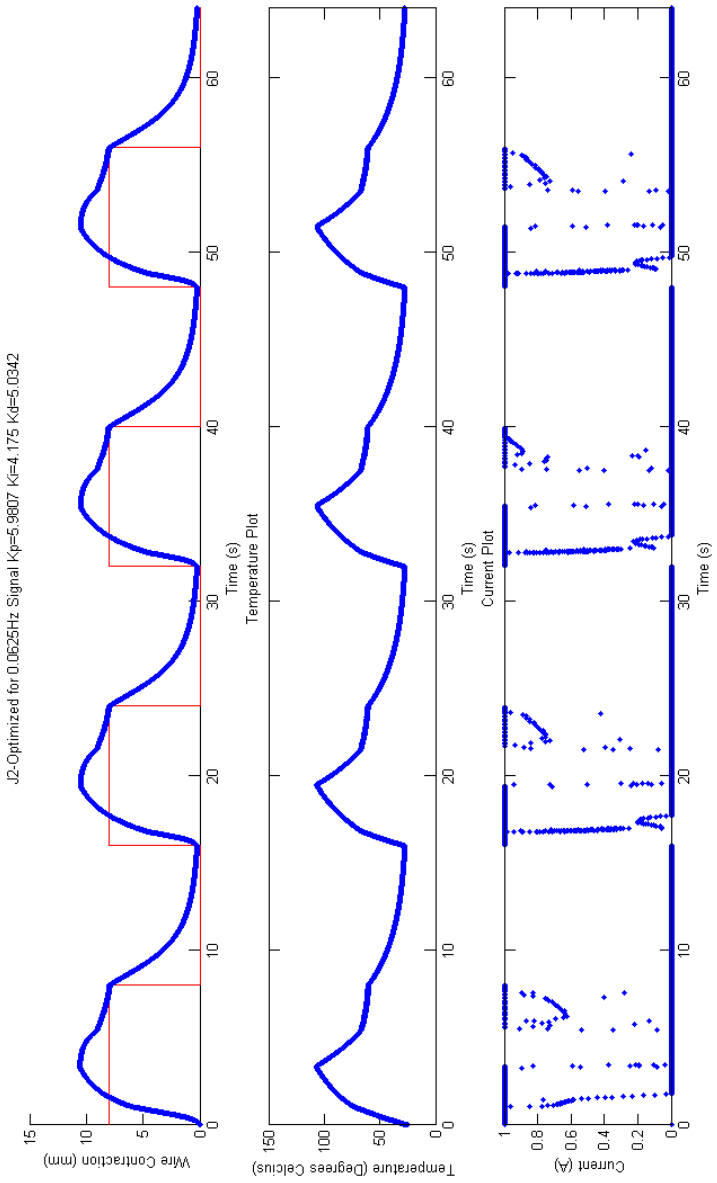

Figure 6.18: J2-Optimized for 0.0625Hz Signal

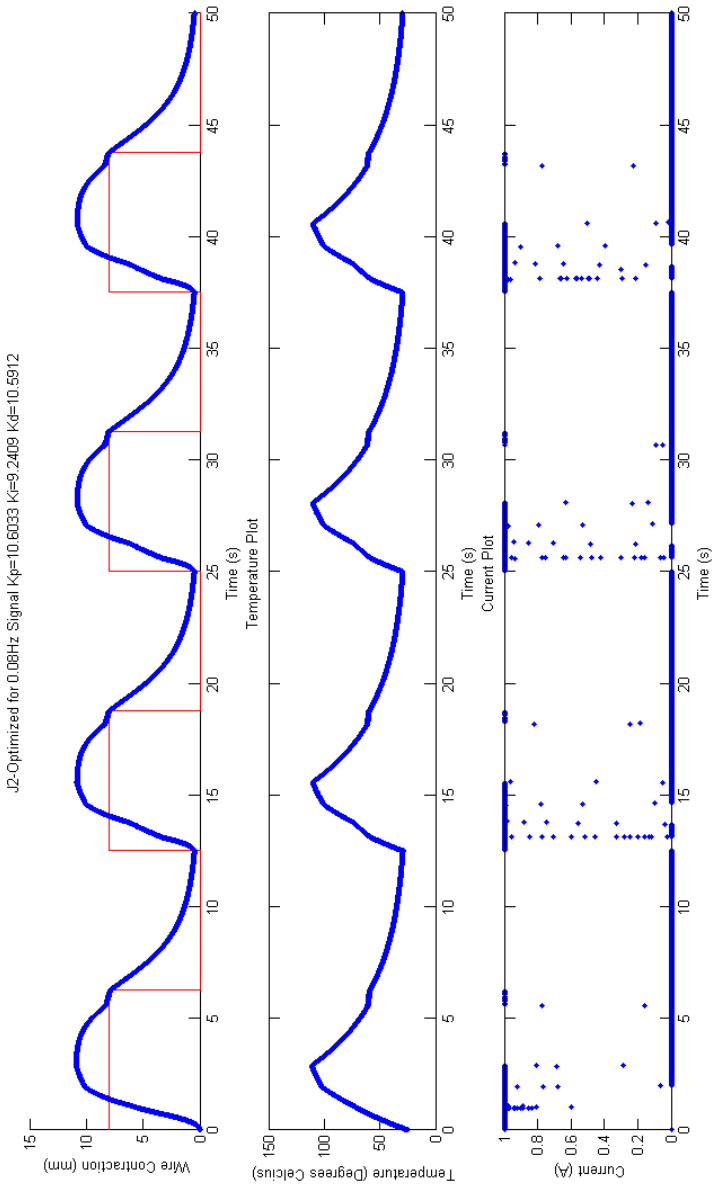

Figure 6.19: J2-Optimized for 0.08Hz Signal

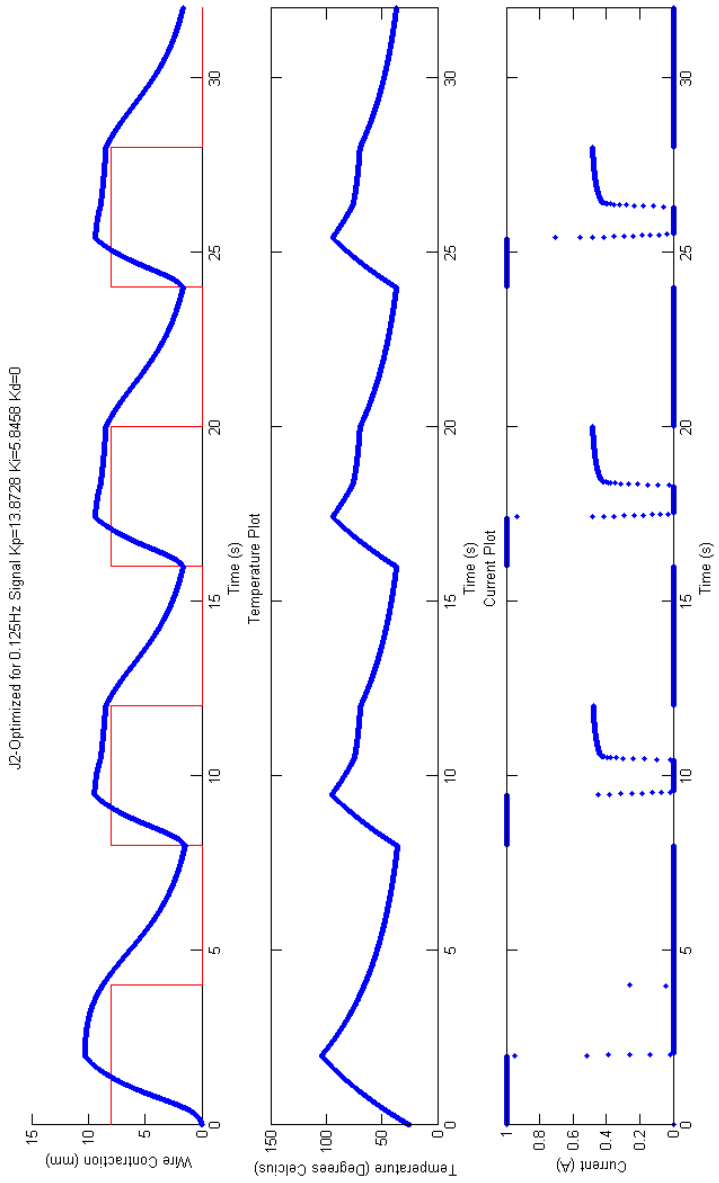

Figure 6.20: J2-Optimized for 0.125Hz Signal

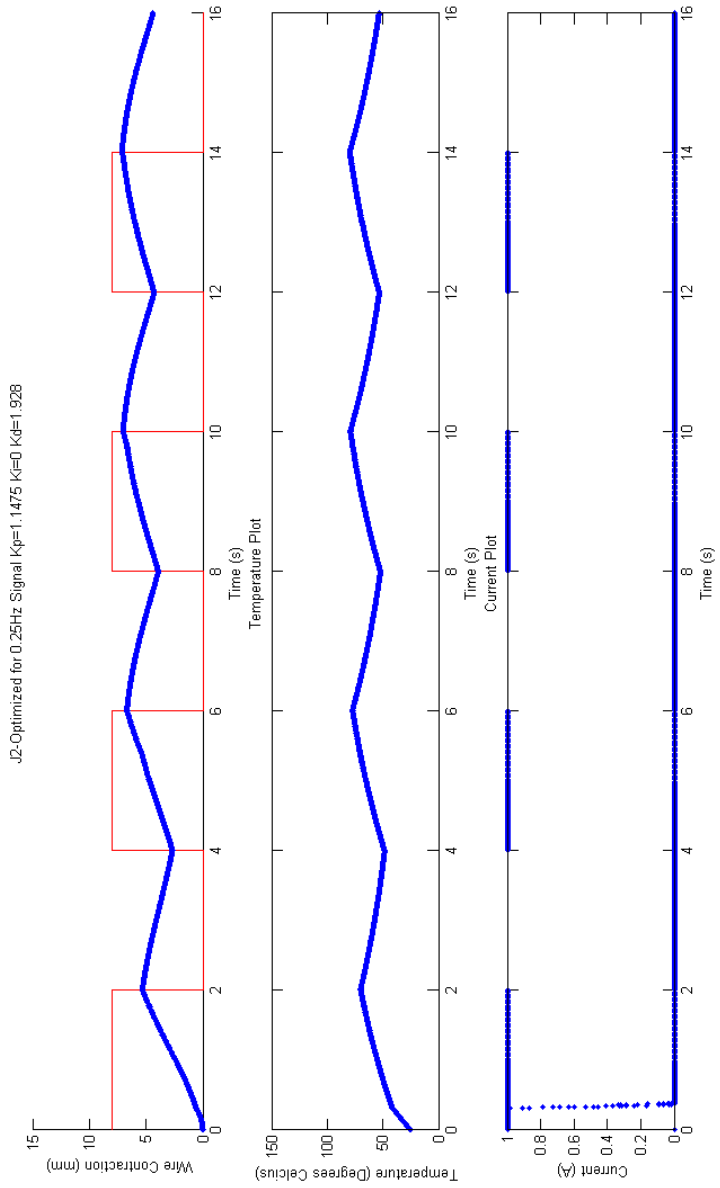

Figure 6.21: J2-Optimized for 0.25Hz Signal

Table 6.3: Optimal Controllers using J_3

Reference Frequency (Hz)	K_p	K_i	K_d	Normalized Error e_n
0.08	10.7056	9.0962	10.4593	0.00002321
0.125	5.3877	4.577	5.1752	0.0022
0.25	0.9731	0.0933	2.0231	0.3514
0.333	1.0686	0.0905	1.9207	0.4792
0.5	6.2908	0	0	0.5471
1	9.4945	10.5247	10.2634	0.8004

6.3 Spread Tracking

For the spread tracking problem, the following cost function is used:

$$J_3(K_p, K_i, K_d) = \sum_{i=1}^{n-1} |\Delta y^i - \Delta y_d|^2 \tag{6.7}$$

where

$$\begin{aligned} \Delta y^i &= |y_s^i - y_s^{i+1}| \\ y_s^i &= \text{contraction at } i^{th} \text{ switch time} \\ \Delta y_d &= \text{desired wire travel} \\ n &= \text{number of switching times} \end{aligned}$$

The reference signal is given in equation (6.1). The reference signal frequencies are 0.08, 0.125, 0.25, 0.333, 0.5 and 1Hz. Δy_d is chosen to be 7.4mm, which corresponds to 2% strain for 100,000 cycle lifetime [29].

The optimal controller for each design frequency is given in Table 6.3. The normalized error is calculated using the following formula:

$$e_n = \frac{\sum_{i=1}^{n-1} |\Delta y^i - \Delta y_d|^2}{\sum_{i=1}^{n-1} |\Delta y_d|^2} \tag{6.8}$$

The denominator corresponds to the tracking error of a zero output. Each optimal controller is then used for the different frequencies and the tracking error is recorded. The normalized error is shown in Figure 6.22.

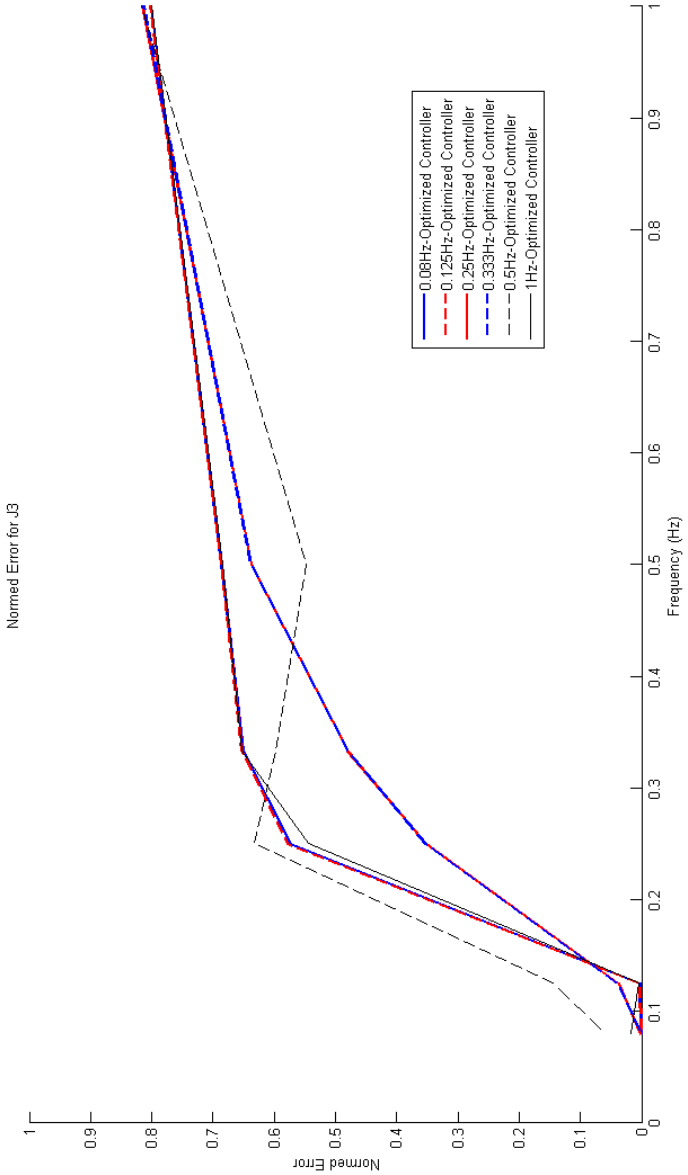

Figure 6.22: Normalized Error for Different Optimal Controllers for J_3

A few observations can be made from Figure 6.22. For low frequency signals, the error for all the controllers are small. This is because at low frequencies, the wire has more time to cool. Since the controllers do not contribute to the cooling phase, the low frequency of the signal allows for better tracking during cooling.

A similar observation can be made at high frequencies such as 1Hz. The errors of each of the controllers are all very close to 0.8. The high error is more a result of the reference signal frequency than the controller gain values. At high frequencies, the system does not have adequate time for cooling. This results in poor tracking during the cooling phase. Optimizing for the higher frequencies can only improve the tracking during the heating phase slightly.

There are two main groups of curves in Figure 6.22. The first group consists of the error curves for 0.08, 0.125 and 1Hz reference signals. While it is expected to see the lower frequency controllers have similar error behaviour, the shape of the 1Hz error curve is unexpected. The second group consists of the error curves for 0.25 and 0.333Hz signals. These two error curves are almost identical because of similar controller gains. They also have relatively low error values across all tested frequencies. One hypothesis is that there is also an *optimal frequency* for the system. Optimizing near this optimal frequency will give good error values across a wide range of frequencies.

The error curve of the controller optimized at 0.5Hz does not follow the general shape of other controllers. A possible explanation for the discrepancy is that the controller found may not be the global optimal controller. However, this controller does have significantly lower error value at 0.5Hz. It is also the best controller among the optimization results using different initial values. Therefore, it may be the case that some controllers only work for the specific frequency it is optimized for, and others work well over a range of frequencies.

The wire contraction, temperature, and input current plots of each of the optimal controllers for each frequency are shown in Figures 6.23 to 6.28. Similar to the previous two cost functions, the controllers give good performance at low frequencies as a result of the cooling time. Once again, the system output stays near the middle to achieve good error values at higher frequencies. There is an exception to the above observation at 0.5Hz (Figure 6.27). Since J_3 only considers the displacement over consecutive switching times,

the control objective can be achieved without alternating between heating and cooling.

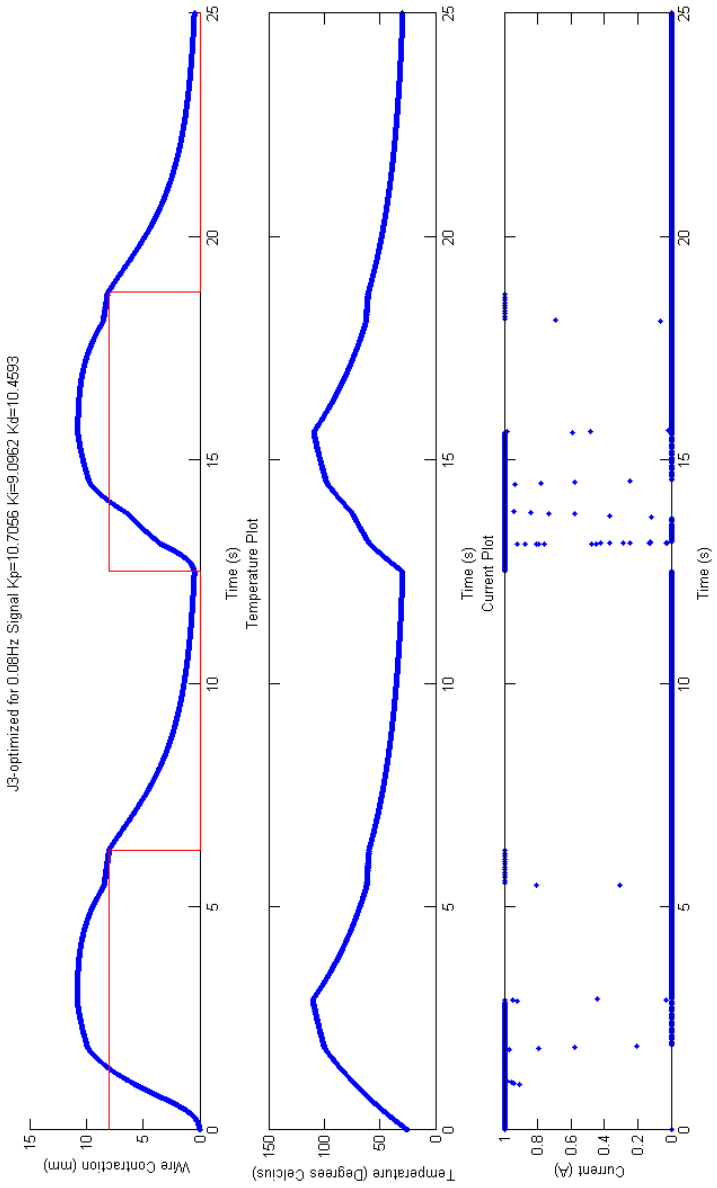

Figure 6.23: J3-Optimized for 0.08Hz Signal

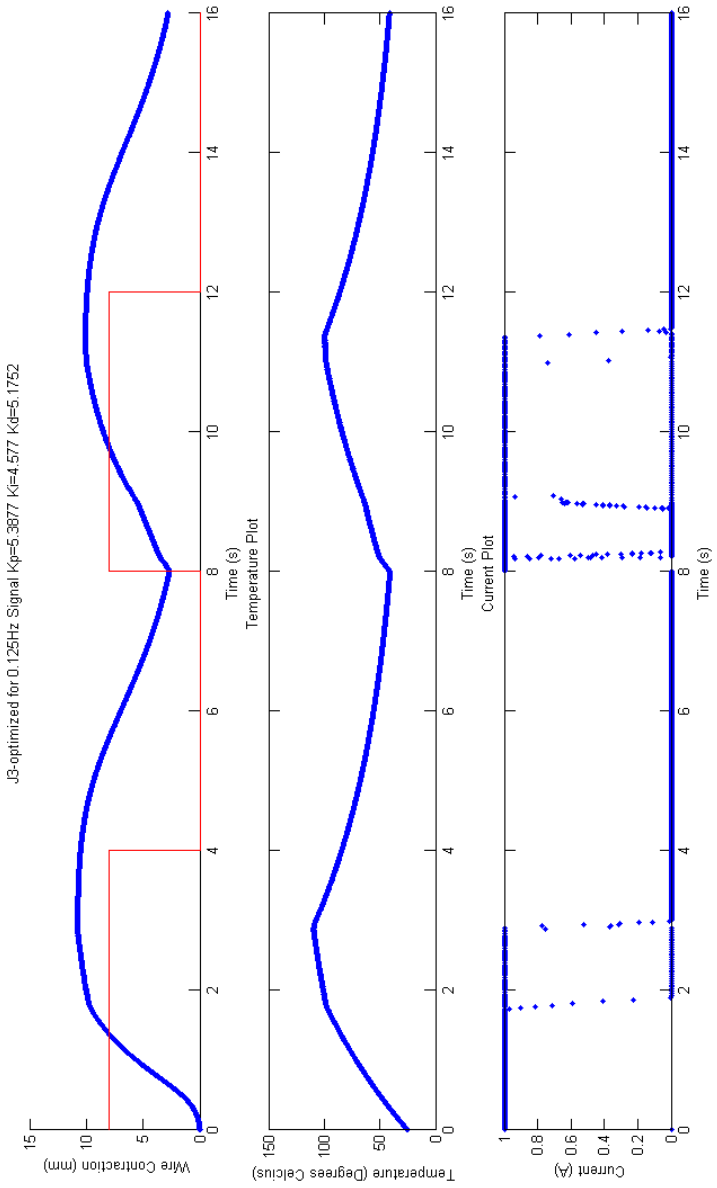

Figure 6.24: J3-Optimized for 0.125Hz Signal

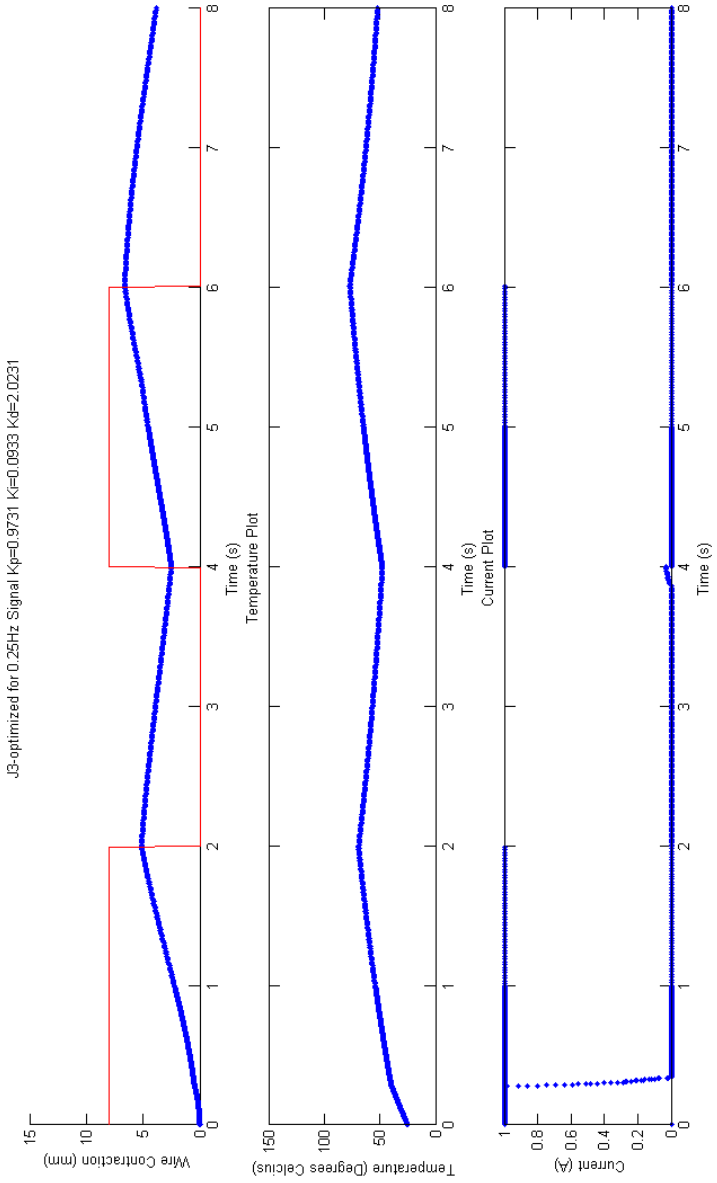

Figure 6.25: J3-Optimized for 0.25Hz Signal

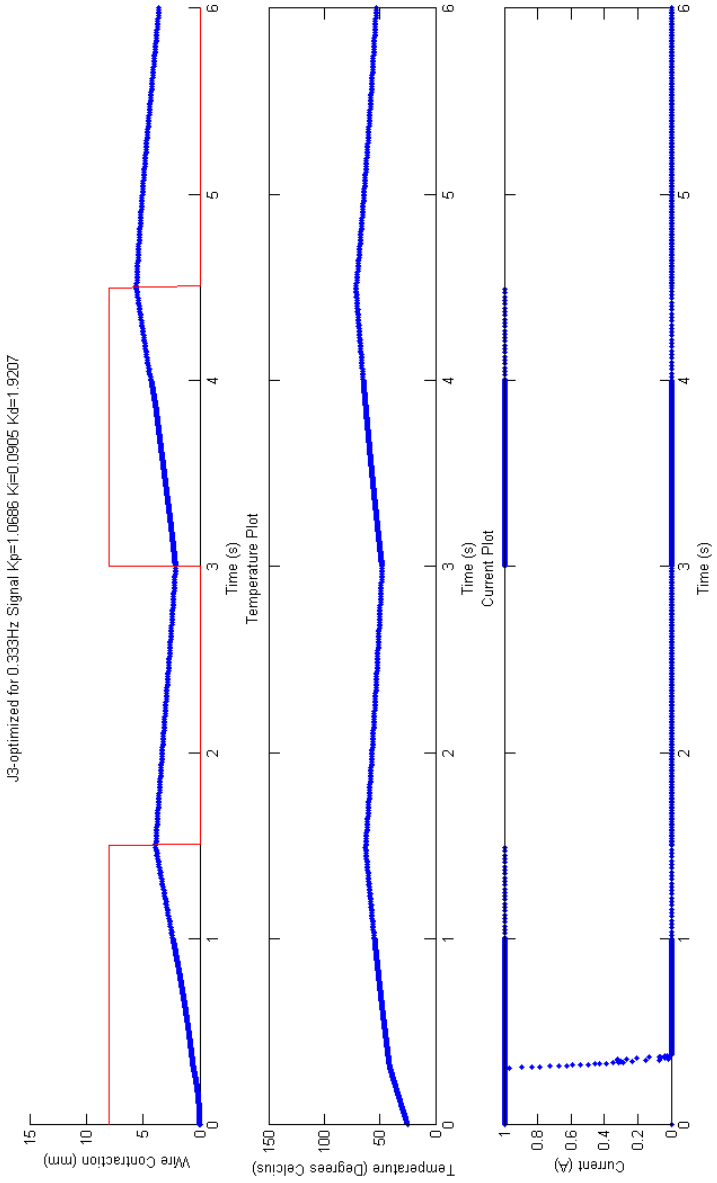

Figure 6.26: J3-Optimized for 0.333Hz Signal

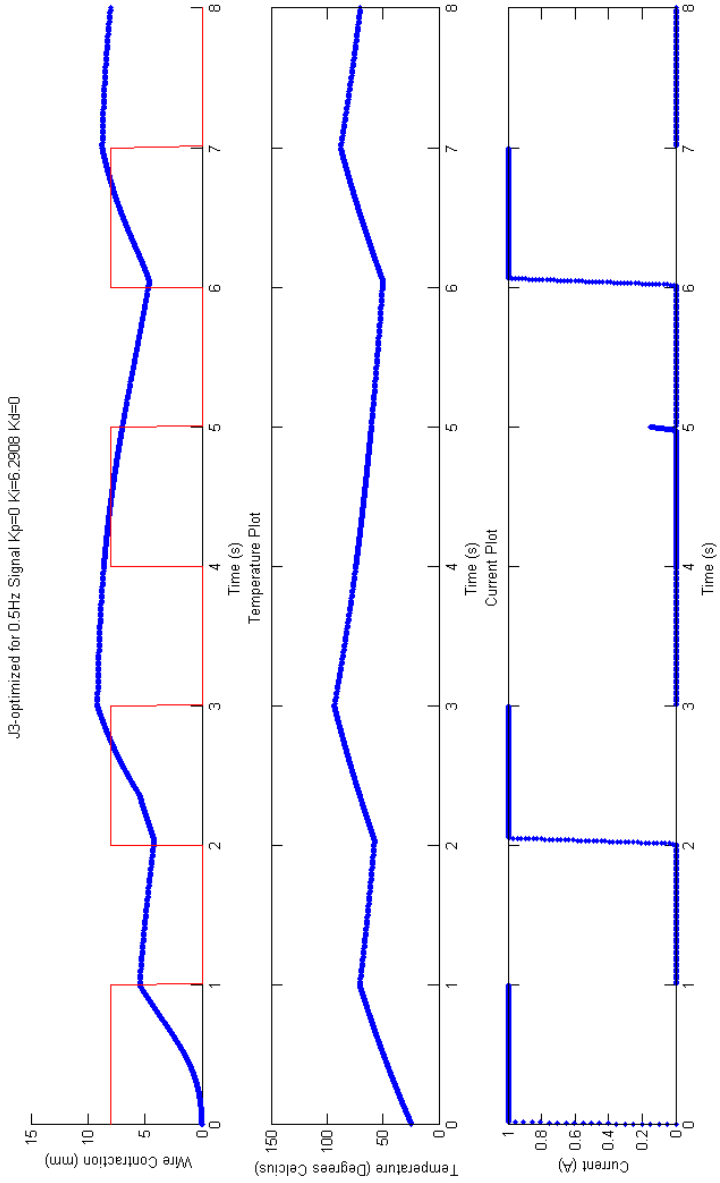

Figure 6.27: J3-Optimized for 0.5Hz Signal

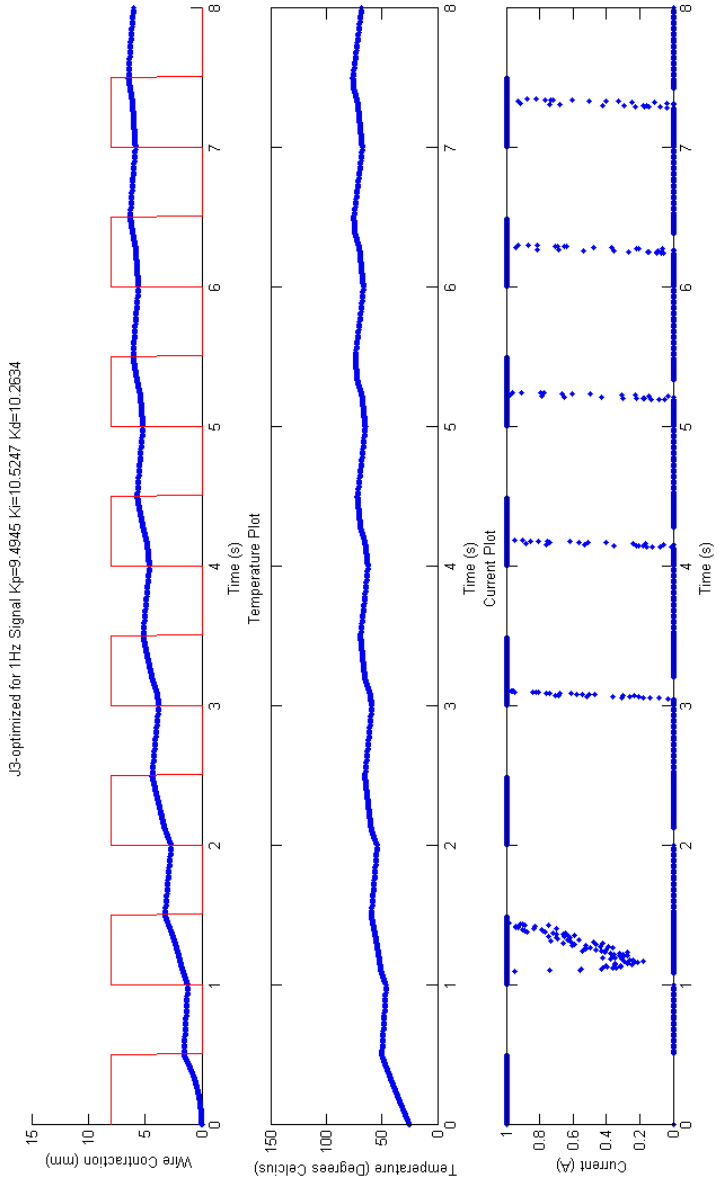

Figure 6.28: J3-Optimized for 1Hz Signal

Cost Function	K_p	K_i	K_d
J_1	0.116	0.2216	0
J_2	10.6033	9.2409	10.5912
J_3	10.7056	9.0962	10.4593

Table 6.4: Controller Gains for Different Cost Functions at 0.08Hz

6.4 Comparing Cost Functions

The three cost functions investigated have different applications, but the objectives are quite similar. The wire contraction and temperature plots of all three cost functions for the optimal controllers at each frequency are shown in Figures 6.29 to 6.30. In most cases, the system outputs are very similar for all three cost functions. At 0.5Hz, the output for J_3 is very different from the other cost functions. This is because the cost function J_3 only includes the displacement over consecutive switching times. Therefore, the system does not have to follow the 'zig-zag' pattern of other outputs to achieve the objective.

The system outputs at 0.125Hz are quite different for the three cost functions. This is because at this frequency, the system has enough time for cooling. Therefore, it is able to reach each target more effectively. Finally, at 0.08Hz, the system output for J_2 and J_3 are very similar. This is because by reaching close to 0mm and 8mm at the switching times for J_2, implies that it has also reached a spread of 7.6mm.

In Figure 6.29, the system output for all three cost functions are very similar at 0.08Hz. The outputs are almost identical when the wire contraction increases from 0 to 10mm initially. The optimal controller gains for the two cost functions at 0.08Hz are given in Table 6.4. While the outputs are similar, the controller gains for J_2 are much larger than the gains for J_1. This is a result of the input saturation. The magnitude of the gains is less significant when the input reaches positive saturation. It can be seen in the input plots of Figures 6.8, 6.19 and 6.23 that during the initial contraction from 0 to 10mm, the input is at positive saturation almost the entire time. Therefore, input saturation is a possible reason for the existence of multiple near-optimal solutions.

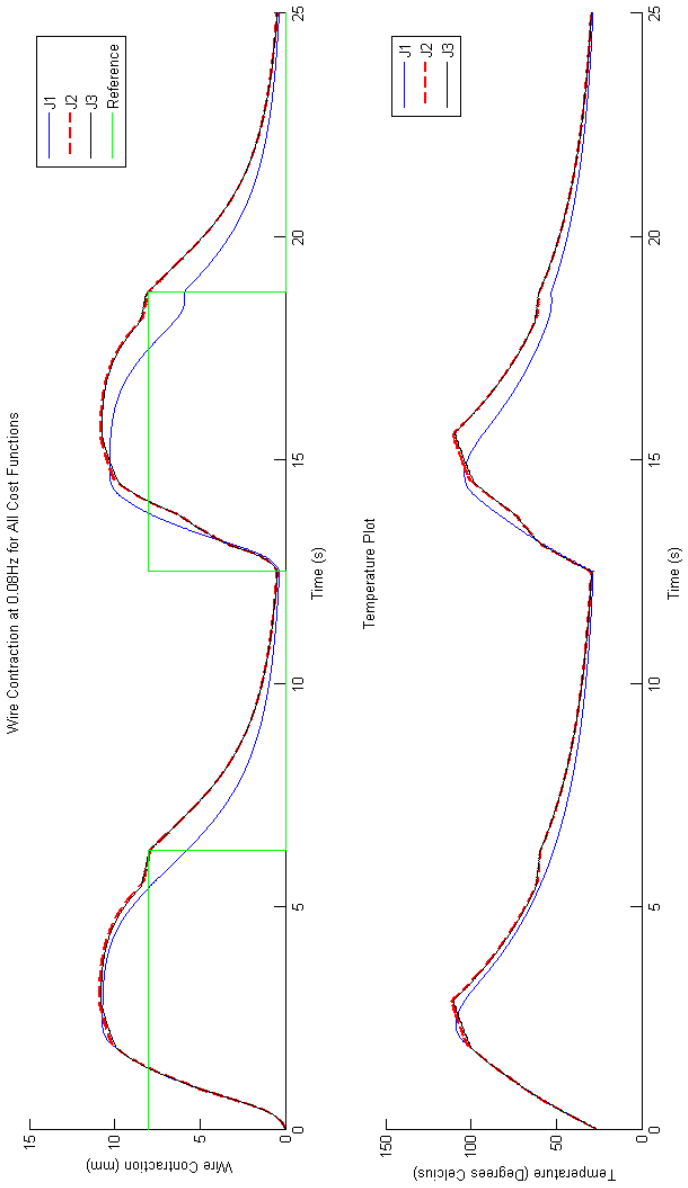

Figure 6.29: Comparison of Cost Functions for 0.08Hz Reference Signal

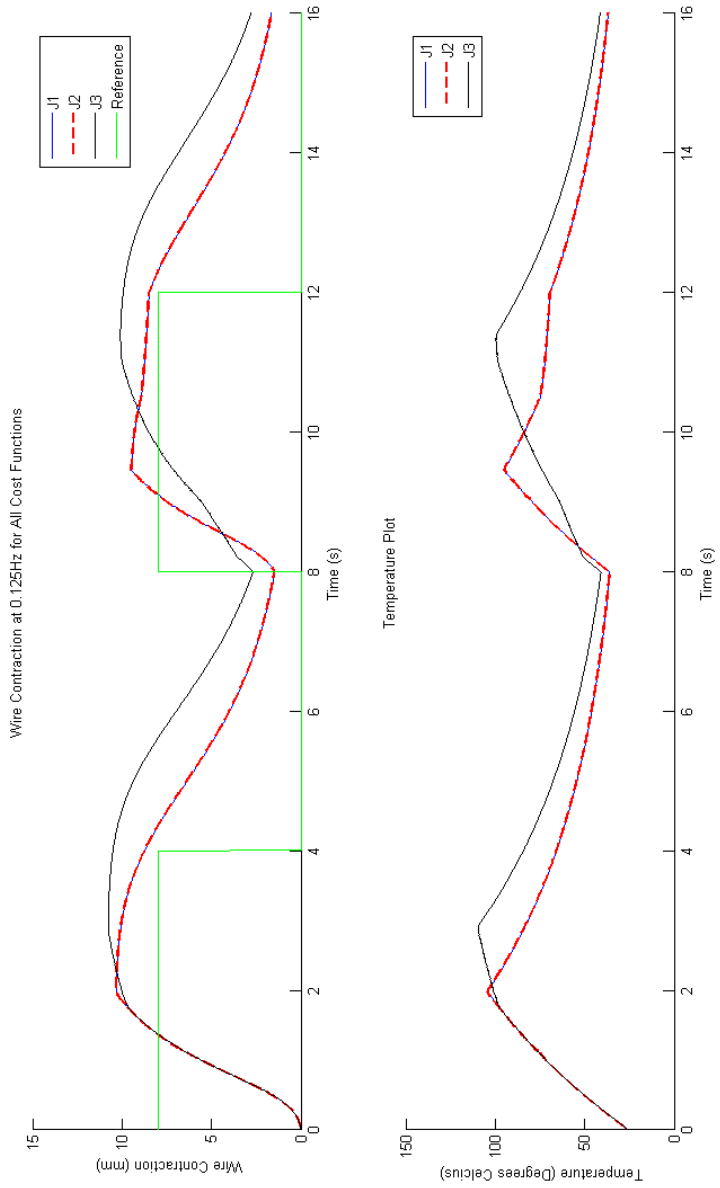

Figure 6.30: Comparison of Cost Functions for 0.125Hz Reference Signal

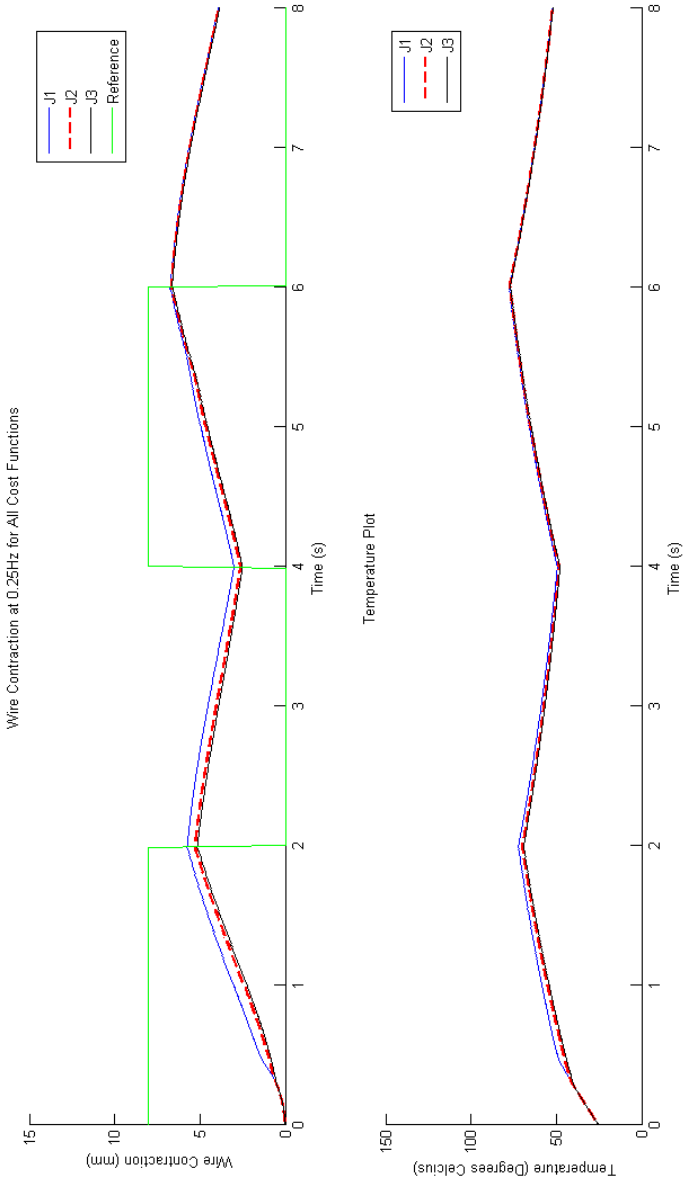

Figure 6.31: Comparison of Cost Functions for 0.25Hz Reference Signal

Chapter 7

Conclusions and Future Research

The research in this thesis examined the effect of PID-controller gain optimization on SMA wire control at different frequencies of operation. A constant-load SMA wire actuator with a PID-controller is used in the study. Heat is applied to the wire using an input electric current. The system is cooled through convection with the surrounding area. The lack of active cooling prevents the system from operating at high frequencies. By optimizing the controller gain values, the bandwidth of the system is improved over the controller optimized from the step response.

Three different cost functions are proposed for various applications. A square wave is used as the reference signal. The Preisach model is chosen to model the hysteretic behaviour of the SMA wire contraction. The system is implemented in MATLAB for simulations. Varying material properties such as electrical resistance and heat capacities are modelled to give a more accurate representation of the system's physical behaviour.

Using each cost function, optimal PID-controller gain values are obtained for a set of reference signal frequencies. The performance of the controllers at different reference frequencies are compared. Results show that the optimal controllers also give good performance in a range of frequencies near the frequency at which they are optimized. This allows the use of one controller for applications that involves a reference signal of varying frequencies as opposed to changing the controller gains whenever the reference frequency changes slightly.

An interesting observation is made in the heating cycle of the SMA wire. In order to

achieve faster cooling, overshoot is observed at low frequencies. This is a result of the system hysteresis. The system hysteresis allows different input signals to achieve the same output value. Since the rate of cooling is proportional to the temperature above ambient, better cooling is achieved by reaching a higher temperature. The error caused by the overshoot is compensated by the better cooling phase, which is not actively controlled.

7.1 Summary of Contributions

The main contribution of this research is summarized below.

- Improvement in performance by optimizing controller gain values over controller designed from the step response for periodic reference signals. The optimal controllers take into account the lack of control in the cooling phase to obtain better tracking results.

7.2 Future Research Directions

Ideas for some future projects are outlined below.

- This work is based on a PID-controller in feedback with an SMA wire. Other controllers such as variable structure control and inverse models can be used to study the bandwidth of the system through optimization. More sophisticated control schemes may provide better tracking results.

- Different actuator configurations such as a differential or a spring-biased type as well as other smart materials can be used in a similar study. Results may improve the performance of actuators where a cooling mechanism is costly to implement.

- The third cost function defined in Section 4.1 suggested that the reference signal amplitude can be used as an optimization parameter. For certain applications, the

system output is not required to track a specific reference signal. Only the relative spread of the output signal is of interest. The control signal and hence the system output depends on the reference signal. Therefore, by adjusting the reference signal amplitude, the cost function may be reduced for a fixed controller. At the same time, the additional design parameter increases the complexity of the optimization problem. Therefore, more effective optimization methods are needed to investigate this problem.

- The major obstacle in the optimization problem is the non-convexity of the solution space. Therefore, different optimization algorithms may be explored to obtain the global optimal solution.
- Finally, it is observed that in some cases, overshoot improves the system performance during the cooling phase. In some applications, overshoot is not desired. Different cost functions can be investigated to overcome this problem. On the other hand, for applications where overshoot is not a major concern, it might be worthwhile to study how this overshooting behaviour can be used to improve overall system performance.

Appendix A

State-space Representation for Preisach Model

This section provides the state-transition and read-out operators for the Preisach model as described in [15]. This state-space representation is shown to satisfy the dynamical systems definition in Section 2.4.

A.1 State-Transition and Read-Out Operators

The *state-transition operator* is an operator that describes the change of state when an input is applied to some initial state. We first introduce an intermediate space:

Definition A.1.1 *The* space of reduced memory sequences S *is defined as the set of reduced memory sequences* s_n *where* $|s_n| \leq u_{sat}\ \forall n$.

The idea of the intermediate space is to first convert the input signal into a reduced memory sequence, and from this sequence, we construct the boundary, or the state ψ. Since ψ uniquely defines the regions $\mathcal{P}_+(t)$ and $\mathcal{P}_-(t)$, we can then obtain the output using equation (2.2).

Next we define the mappings that relate the spaces B, S, U and Y.

Definition A.1.2 *The mapping $F_\tau : U \rightarrow S$ takes an input signal $u(t)$and creates a reduced memory sequence up to time $t = \tau$ as defined in Section 2.4.2.*

Since the reduced memory sequence does not contain all of the input history, the map F_τ does not have an inverse. But since inputs that share the same reduced memory sequence produce the same output value, we can treat these inputs as an equivalence class of inputs.

Definition A.1.3 *The mapping $F_\tau^r : S \rightarrow U$ takes a reduced memory sequence up to time $t = \tau$ and recreate a possible input signal $u(t)$ as follows [15]:*

Choose an arbitrary $t_0 < \tau$. Partition the interval $[t_0, \tau]$ by

$$t_i = t_0 + \frac{\tau - t_0}{2} \sum_{k=0}^{i-1} \frac{1}{2^k}, \ \forall i \geq 1.$$

Assign values of u in the following manner: $u(t) = 0$ for $t \leq t_0$and $u(t_i) = s_i$ for $i \geq 1$. Connect the discrete points $u(t_i)$ with straight lines.

Definition A.1.4 *The mapping $G_\tau : S \rightarrow B$ takes a reduced memory sequence up to time $t = \tau$ and create the corresponding boundary in the Preisach plane $\mathcal{P}_r$.*

Definition A.1.5 *The mapping $G_\tau^{-1} : B \rightarrow S$ takes a boundary in the Preisach plane $\mathcal{P}_r$ and reconstruct a corresponding reduced memory sequence up to time $t = \tau$.*

The inverse of G exists since the spaces B and S share the wiping out property, thus any reduced memory sequence uniquely defines a boundary and vice versa.

Definition A.1.6 *The concatenation operator $\Diamond$ is defined by*

$$u_{(t_0,t_1]} \Diamond v_{(t_1,t_2]} = \begin{cases} u, t_0 < t \leq t_1 \\ v, t_1 < t \leq t_2 \end{cases}$$

We are now ready to define the state-transition operator.

Definition A.1.7 *The* state-transition operator *is the mapping ϕ defined by*

$$\psi_1 = \phi(t_1, t_0, \psi_0, u) = G_{t_1} F_{t_1}((F_{t_0}^r G_{t_0}^{-1} \psi_0) \Diamond u_{(t_0,t_1]})$$

The state-transition operator takes the present boundary ψ at time $t = t_0$, first convert it to a corresponding reduced memory sequence via $G_{t_0}^{-1}$. Then an equivalent input is reconstructed using $F_{t_0}^r$. This 'past input' is then concatenated with the new input u. The new joint input is then used to construct a new reduced memory sequence, and in turn produces the new boundary ψ_1.

Note that $(F_{t_0}^r G_{t_0}^{-1}\psi_0)\Diamond u_{(t_0,t_1]}$ need not be a continuous function. Since there is no continuity requirement in the construction of the reduced memory sequence, the problem created by a discontinuous input is avoided.

Lastly, we define the read-out operator.

Definition A.1.8 *The* read-out operator r *is defined as*

$$y(t) = r(\psi) = \iint_{\mathcal{P}_+(\psi)} \mu(r,s)drds - \iint_{\mathcal{P}_-(\psi)} \mu(r,s)drds \tag{A.1}$$

where $\mathcal{P}_+(\psi)$ and $\mathcal{P}_-(\psi)$ denotes the regions $\mathcal{P}_+(t)$ and $\mathcal{P}_-(t)$ defined by the boundary ψ at time t respectively.

Note that equation (A.1) is a modified version of equation (2.2). Since the relays in the region $\mathcal{P}_+(t)$ and $\mathcal{P}_-(t)$ have output values of +1 and -1 respectively, the function $\gamma(r,s)$ in equation (2.2) can be accounted for by splitting into two integrals. The regions $\mathcal{P}_+(t)$ and $\mathcal{P}_-(t)$ are defined by the evolution of the boundary ψ. Since the boundary contains continuous line segments of slope +1 or -1, the integrals in equation (A.1) are well-defined.

We now show that the state space formulation described above indeed satisfies the axioms in Definition 2.4.1. Axioms (i)-(iii), (v) and (vi) are satisfied by construction.

$(iv)_a$ (consistency): For any $t_0 \in \mathbb{R}$, $\psi_0 \in B$ and an input $u \in \mathcal{U}$, we have

$$\begin{aligned}
\phi(t_0,t_0,\psi_0,u) &= G_{t_0}F_{t_0}((F_{t_0}^r G_{t_0}^{-1}\psi_0)\Diamond u_{(t_0,t_0]}) \\
&= G_{t_0}F_{t_0}(F_{t_0}^r G_{t_0}^{-1}\psi_0) \\
&= G_{t_0}G_{t_0}^{-1}\psi_0 \\
&= \psi_0
\end{aligned}$$

$(iv)_b$ (determinism): For any $t_1 \geq t_0 \in \mathbb{R}$, $\psi_0 \in B$ and inputs $u_1, u_2 \in \mathcal{U}$ satisfying $u_1(t) = u_2(t)$ for $t_0 \leq t \leq t_1$, we have

$$\begin{aligned} \phi(t_1, t_0, \psi_0, u_1) &= G_{t_1} F_{t_1}((F^r_{t_0} G^{-1}_{t_0} \psi_0) \Diamond u_{1(t_0,t_1]}) \\ \phi(t_1, t_0, \psi_0, u_2) &= G_{t_1} F_{t_1}((F^r_{t_0} G^{-1}_{t_0} \psi_0) \Diamond u_{2(t_0,t_1]}) \end{aligned}$$

if we let $u_0 = F^r_{t_0} G^{-1}_{t_0} \psi_0$ then

$$\begin{aligned} \phi(t_1, t_0, \psi_0, u_1) &= G_{t_1} F_{t_1}(u_0 \Diamond u_{1(t_0,t_1]}) \\ \phi(t_1, t_0, \psi_0, u_2) &= G_{t_1} F_{t_1}(u_0 \Diamond u_{2(t_0,t_1]}) \end{aligned}$$

since the mapping F_τ and G_τ are deterministic, equal inputs create the same reduced memory sequence and thus the same boundary; also since $u_{1(t_0,t_1]} = u_{2(t_0,t_1]}$, hence

$$\phi(t_1, t_0, \psi_0, u_1) = \phi(t_1, t_0, \psi_0, u_2)$$

$(iv)_c$ (semi-group property) For any $t_0 \leq t_1 \leq t_2 \in \mathbb{R}$, $\psi_0 \in B$ and input $u \in \mathcal{U}$, we need to show that

$$\phi(t_2, t_0, \phi_0, u) = \phi(t_2, t_1, \phi(t_1, t_0, \psi_0, u), u)$$

Left hand side: if we let $u_0 = F^r_{t_0} G^{-1}_{t_0} \psi_0$ then

$$\begin{aligned} \phi(t_2, t_0, \psi_0, u) &= G_{t_2} F_{t_2}((F^r_{t_0} G^{-1}_{t_0} \psi_0) \Diamond u_{(t_0,t_2]}) \\ &= G_{t_2} F_{t_2}(u_0 \Diamond u_{(t_0,t_2]}) \\ &= G_{t_2} F_{t_2} \tilde{u} \end{aligned}$$

where $\tilde{u} = u_0 \Diamond u_{(t_0,t_2]}$.

Right hand side: if we let $u_0 = F^r_{t_0} G^{-1}_{t_0} \psi_0$ then

$$\begin{aligned}
\phi(t_2, t_1, \phi(t_1, t_0, \psi_0, u), u) &= G_{t_2} F_{t_2}((F^r_{t_1} G^{-1}_{t_1} \phi(t_1, t_0, \psi_0, u)) \Diamond u_{(t_1,t_2]}) \\
&= G_{t_2} F_{t_2}((F^r_{t_1} G^{-1}_{t_1} G_{t_1} F_{t_1}((F^r_{t_0} G^{-1}_{t_0} \psi_0) \Diamond u_{(t_0,t_1]}) \Diamond u_{(t_1,t_2]}) \\
&= G_{t_2} F_{t_2}(F^r_{t_1} F_{t_1}((F^r_{t_0} G^{-1}_{t_0} \psi_0) \Diamond u_{(t_0,t_1]}) \Diamond u_{(t_1,t_2]}) \\
&= G_{t_2} F_{t_2}((F^r_{t_1} F_{t_1}(u_0 \Diamond u_{(t_0,t_1]})) \Diamond u_{(t_1,t_2]}) \\
&= G_{t_2} F_{t_2}((F^r_{t_1} F_{t_1} \tilde{u}_{(-\infty,t_1]}) \Diamond u_{(t_1,t_2]}) \\
&= G_{t_2} F_{t_2}(\tilde{u'}_{(-\infty,t_1]} \Diamond u_{(t_1,t_2]}) \\
&= G_{t_2} F_{t_2} \tilde{u'}_{(-\infty,t_2]} \\
&= G_{t_2} F_{t_2} \tilde{u}
\end{aligned}$$

The last equality holds since the inputs create the same reduced memory sequences, and hence the same output values. Hence we have

$$\phi(t_2, t_0, \psi_0, u) = \phi(t_2, t_1, \phi(t_1, t_0, \psi_0, u), u)$$

as required.

$(iv)_d$ (stationarity) For any $t_1 \geq t_0 \in \mathbb{R}$,$T \in \mathbb{R}$, $\psi_0 \in B$ and inputs $u, u_T \in \mathcal{U}$ satisfying $u_T(t) = u(t+T)\ \forall t \in \mathbb{R}$, we need to show that

$$\begin{aligned}
\phi(t_1 + T, t_0 + T, \psi_0, u) &= \psi(t_1, t_0, \psi_0, u_T) \\
G_{t_1+T} F_{t_1+T}((F^r_{t_0+T} G^{-1}_{t_0+T} \psi_0) \Diamond u_{(t_0+T,t_1+T]}) &= G_{t_1} F_{t_1}((F^r_{t_0} G^{-1}_{t_0} \psi_0) \Diamond u_{T(t_0,t_1]})
\end{aligned}$$

Left hand side: since the boundary and reduced memory sequences do not contain information in time, i.e. $G_{t_1+T} F_{t_1+T} = G_{t_1} F_{t_1}$ and $F^r_{t_0+T} G^{-1}_{t_0+T} = F^r_{t_0} G^{-1}_{t_0}$. Using the fact that $u_T(t) = u(t+T)\ \forall t \in \mathbb{R}$, we have

$$G_{t_1+T} F_{t_1+T}((F^r_{t_0+T} G^{-1}_{t_0+T} \psi_0) \Diamond u_{(t_0+T,t_1+T]}) = G_{t_1} F_{t_1}((F^r_{t_0} G^{-1}_{t_0} \psi_0) \Diamond u_{T(t_0,t_1]})$$

as required.

Thus, we have shown that the state space representation presented in the above sections defines a dynamical system.

Appendix B

MATLAB Codes

B.1 Main Simulation File

```
function [f]=SMA(K)

%initialization
temp_amb=25;

load sim_data.mat load refv.mat
%define parameters
L=0.380;        % meters, measured
d=12/1000*2.54/100; % meters, from 12mil specifications
rho=6500;       % kg/m^3, from Madill's thesis, confirmed in D&P paper
c_p=460;            % J/kgC,  from Madill's thesis
h=75;           % W/m^2C, from Madill's thesis
H=0.0618    % latent heat

%transformation temperatures
A_s=53
A_f=80
M_s=47
```

```
M_f=19


V=pi*(d/2)^2*L;     % m^3, wire volume calculation
A=pi*d*L;       % m^2, wire surface area for cooling purposes
Ra=4.5;
Rm=5;

for i=1:3
    if K(i)<0
        K(i)=0
    end
end

Ki=K(1);
Kp=K(2);
Kd=K(3);

%define variables
t_it=length(t) in=zeros(1,t_it); out=zeros(1,t_it);
temp=zeros(1,t_it); temperature = temp_amb;

R=Ra;
e1=ref(1);
e2=ref(1);
int_e=0;


[y]=pm(fod,index,0,fab_file,fab_coeffs,1)

in(1)=0;
```

```
out(1)=y;
temp(1)=temperature;

%loop over time interval

for i=2:length(t)
dt=t(i)-t(i-1)

u=Ki*int_e+Kp*e2+Kd*(e2-e1)/dt;

%input restrictions

if u>1
    u=1
end if u<0
    u=0
end if ref(i)==0
    int_e=0
    u=0
end

%varying heat capacities

if u>0
    if (temp(i-1)>=A_s) &&(temp(i-1)<=A_f)
        c=c_p+H*(log(100)/abs(A_s-A_f))*exp(-2*(log(100)/abs(A_s-A_f))*...
            abs(temp(i-1)-0.5*(A_s+A_f)))
    else
        c=c_p
    end
else
```

```
    if (temp(i-1)>=M_s) &&(temp(i-1)<=M_f)
        c=c_p+H*(log(100)/abs(M_s-M_f))*exp(-2*(log(100)/abs(M_s-M_f))*...
            abs(temp(i-1)-0.5*(M_s+M_f)))
    else
        c=c_p
    end
end

temperature=dt*(((R*(u^2)))-h*A*(temperature-25))/(rho*c*V)+temperature

%access Preisach Model file to obtain output
[y]=pm(fod,index,(temperature-temp_amb),fab_file,fab_coeffs,0)

output=y; in(i)=u; out(i)=output; temp(i)=temperature; e1=e2;
e2=ref(i)-output; int_e=int_e+.5*(e1+e2)*dt;
phase_frac=output/(index(length(index))-index(1));
R=phase_frac*Ra+(1-phase_frac)*Rm;
end %t loop

%display results
subplot(3,1,1);
hold on
plot(t,out,'.');
plot(t,ref,'-r');
hold off
xlabel('Time (s)')
ylabel('Wire Contraction (mm)')
title('Hysteresis
Output - Ref=0.5, Ki=0, Kp=3, Kd=0')
subplot(3,1,2);
plot(t,temp,'.');
```

```
xlabel('Time (s)')
ylabel('Temperature (Degrees
Celcius)')
title('Temperature Plot')
subplot(3,1,3);
plot(t,in,'.');
xlabel('Time (s)')
ylabel('Current (A)')
title('Current Plot')

%error output
f=trapz(t,(out-ref).^2)

end
```

B.2 Preisach Simulator

```
function [y]=pm(fod,index,u,fab_file,fab_coeffs,rr)
% Declare "memory" variables as global
global rms yo

% Size of identification grid (nxn)
n  = length(index)

% Some useful variables
umax = index(n); umin = index(1); ymax = fod(n,n); ymin = fod(n,1);
hmax=umax; hmin=umin;
% Initialize "memory" variables if this is the first call
% Assume P- = P, P+ is empty, and u=umin
if (rr==1)
    rms=[umax umin;umin umin];
```

```
    yo=ymin;
end evalstr=[fab_file '(fab_coeffs,']; h_plus=0;
% Get the value of the last input
uo = rms(1,end);

% First, a few special cases which are easy enough to handle seperately

if (u==uo)          % input hasn't changed since last time
    y=yo;
elseif (u>=umax)    % positive saturation
    rms = [umax umax;umin umax];
    y = ymax;
elseif (u<=umin)    % negative saturation
    rms = [umax umin;umin umin];
    y = ymin;

else % Not a special case: determine the new RMS & output
    if (u>uo) % increasing input
        cut = max(find(rms(1,:)>u));
        rms = [rms(:,1:cut) [u;rms(2,cut)] [u;u]];
        rms1=rms;

    elseif (u<uo) %decreasing input
        cut = max(find(rms(2,:)<u));
        rms = [rms(:,1:cut) [rms(1,cut);u] [u;u]];
        rms1=rms;
    end

    while (length(rms(2,:))>1),
        if (rms(2,1)~=rms(2,2))
        %use rms for the corners of the trapezoids
```

```
            trapezoid=0.5*(eval([evalstr num2str(rms(1,2))...
                ',num2str(rms(2,2)) ')']-eval([evalstr num2str(rms(1,1))...
                ',' num2str(rms(2,1)) ')'])  );
                    h_plus = h_plus + trapezoid;
        end
        [n1,n2]=size(rms)
        rms=rms(:,2:n2);    % pop the trapezoid index off the top
    end;
%
% Compute the output
%

big_triangle = 0.5*( eval([evalstr 'hmax , hmax)'])- ...
                     eval([evalstr 'hmax , hmin)'])  );

offset = 0.5*( eval([evalstr 'hmax,hmax)'])+ ...
               eval([evalstr 'hmax,hmin)'])  );


y = 2*h_plus - big_triangle + offset; rms =rms1 end

% Save output state for next time

yo = y; end
```

Bibliography

[1] H. Ashrafiuon, M. Eshraghi and M. H. Elahinia. Position Control of a Three-link Shape Memory Alloy Actuated Robot. *Journal of Intelligent Material Systems and Structures*, 17:381–392, May 2006.

[2] H. T. Banks, R. C. H. del Rosario and R. C. Smith. Reduced-Order Model Feedback Control Design: Numerical Implementation in a Thin Shell Model. *IEEE Transactions on Automatic Control*, 45(7):1312–1324, July 2000.

[3] M. Baulac, J. Defrance and Philippe Jean. Optimization of Multiple Edge Barriers with Genetic Algorithms Coupled with a Nelder-Mead Local Search. *Journal of Sound and Vibration*, 300:71–87, 2007.

[4] A. Bhattacharyya, D. C. Lagoudas, Y. Wang and V. K. Kinra. On the Role of Thermoelectric Heat Transfer in the Design of SMA Actuators: Theoretical Modeling and Experiment.

[5] M. Brokate and J. Sprekels. *Hysteresis and Phase Transitions*. Springer-Verlag, New York, 1996.

[6] G. Černivec, J. Krč, F. Smole and M. Topič. Band-gap Engineering in CIGS Solar Cells using Nelder-Mead Simplex Optimization Algorithm. *Thin Solid Films*, 511-512:60–65, 2006.

[7] S. H. Chen, X. W. Song and Z. J. Yang. Modal Optimal Control of Cabin Noise of Vehicles. *Smart Materials and Structures*, 14(1):257–264, 2005.

[8] W. Chen, M. Buehler, G. Parker and B. Bettig. Optimal Sensor Design and Control fo Piezoelectric Laminate Beams. *IEEE Transactions on Control Systems Technology*, 12(1):148–155, January 2004.

[9] E. de Klear, C. Roos and T. Terlaky. *Nonlinear Optimization: Course notes for CO 367*. University of Waterloo, 2004.

[10] S. S. Fan, Y. Liang and Erwie Zahara. A Genetic Algorithm and a Particle Swarm Optimizer Hydridized with Nelder-Mead Simplex Search. *Computers & Industrial Engineering*, 50:401–425, 2006.

[11] M. I. Frecker. Recent Advances in Optimization of Smart Structures and Actuators. *Journal of Intelligent Material Systems and Structures*, 14:207–216, May 2003.

[12] P. Ge and M. Jouaneh. Generalized Preisach Model for Hysteresis Nonlinearity of Piezoceramic Actuators. *Precision Engineering*, 20:99–111, 1997.

[13] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.

[14] R. B. Gorbet. *Control of Hysteretic Systems with Preisach Representations*. PhD thesis, University of Waterloo, Waterloo, ON, Canada, 1997.

[15] R. B. Gorbet, K. A. Morris and D. W. L. Wang. Control of Hysteretic Systems: A State-Space Approach. *Learning, Control and Hybrid Systems*, pages 432–451, 1998.

[16] R. B. Gorbet, K. A. Morris and D. W. L. Wang. Passivity-based Stability and Control of Hysteresis in Smart Actuators. *IEEE Transaction on Control Systems Technology*, 9(1):377–382, January 2001.

[17] R. B. Gorbet and D. W. L. Wang. A Dissipativity Approach to Stability of a Shape Memory Alloy Position Control System. *IEEE Transactions on Control Systems Technology*, 6(4):554–562, July 1998.

[18] D. Grant and V. Hayward. Variable Structure Control of Shape Memory Alloy Actuators. *IEEE Control Systems*, pages 80–88, 1997.

[19] C. Haag, M. D. Tandale and J. Valasek. Characterization of Shape Memory Alloy Behavior and Position Control Using Reinforcement Learning. pages 1–9, Sept. 2005 2005.

[20] A. G. Hatch, R. C. Smith, T. De and M. V. Salapaka. Construction and Experimental Implementation of a Model-Based Inverse Filter to Attenuate Hysteresis in Ferroelectric Transducers. *IEEE Transactions on Control Systems Technology*, 14(6):1058–1069, November 2006.

[21] Y. Y. He, S. Oi, F. L. Chu and H. X. Li. Vibration Control of a Rotor-bearing System using Shape Memory Alloy: I. Theory. *Smart Materials and Structures*, 16:114–121, 2007.

[22] Y. Y. He, S. Oi, F. L. Chu and H. X. Li. Vibration Control of a Rotor-bearing System using Shape Memory Alloy: II. Experimental Study. *Smart Materials and Structures*, 16:122–127, 2007.

[23] O. Heintze, S. Seelecke and C. Büskens. Modelling and Optimal Control of Microscale SMA Actuators. In R. C. Smith, editor, *Smart Structures and Materials 2003: Modelling, Signal Processing and Control*, volume 5049, pages 495–505. SPIE, 2003.

[24] D. J. Hill and P. J. Moylan. Stability Results for Nonlinear Feedback Systems. *Automatica*, 12:377–382, 1977.

[25] D. J. Hill and P. J. Moylan. Stability Criteria for Large-Scale Systems. *IEEE Transactions on Automatic Control*, 23(2):143–149, 1978.

[26] D. Huges and J. T. Wen. Preisach Modelling of Piezocramic and Shape Memory Alloy Hysteresis. *Proceedings of the 1995 IEEE Control Conference on Applications*, September 1995.

[27] K. Ikuta. Micro/miniature Shape Memory Alloy Actuator. *IEEE International Conference on Robotics and Automation*, 3:2156–2161, 1990.

[28] R. V. Iyer, X. Tan and P. S. Krishnaprasad. Approximate Inversion of the Preisach Hysteresis Operator with Application to Control of Smart Actuators. *IEEE Transactions on Automatic Control*, 50(6):798–810, June 2005.

[29] H. Janocha. *Adaptronics and Smart Structures*. Springer-Verlag, Berlin Heidelberg, 1999.

[30] D. C. Jiles and D. L. Atherton. Theory of Ferromagnetic Hysteresis. *Journal of Magnetism and Magnetic Materials*, 61:48–60, 1986.

[31] H. K. Khalil. *Nonlinear Systems, Third Edition*. Prentice Hall, Upper Saddle River, NJ, 2002.

[32] T. G. Kolda, R. M. Lewis and V. Torczon. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Review*, 45(3):385–482, 2003.

[33] B. M. Kolundzija and D. I. Olcan. Antenna Optimization Using Combination of Random and Nelder-Mead Simplex Algorithms. *2003 IEEE Propagation Society International Symposium*, 1:185–188, June 2003.

[34] J. C. Lagarias, J. A. Reeds, M. H. Wright and P. E. Wright. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998.

[35] R. Lashlee, V. Rao and F. Kern. H_∞ Optimal Control of Smart Structures. *SPIE*, 2192:156–167.

[36] S. Majima, K. Kodama and T. Hasegawa. Modeling of Shape Memory Alloy Actuator and Tracking Control System with the Model. *IEEE Transactions on Controll Systems Technology*, 9(1):54–59, January 2001.

[37] A. Martinez, F. A. Agelet, L. J. Alvarez-Vázquez, J. M. Hernando and D. Mosteiro. Optimal Transmitter Location in an Indoor Wireless System by Nelder-Mead Method. *Microwave and Optical Technology Letters*, 27(2):146–148, October 2000.

[38] I. D. Mayergoyz. *Mathematical Models of Hysteresis*. Springer-Verlag, New York, 1991.

[39] K. I. M. McKinnon. Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimization*, 9:148–158, 1998.

[40] K. A. Morris and J. N. Juang. Dissipative Controller Designs for Second-Order Dynamic Systems. *Fields Institute Communications*, 2:71–90, 1993.

[41] J. Nealis and R. C. Smith. $\mathcal{H}_\infty$ Control Design for a Magnetostrictive Transducer. *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 1801–1806, December 2003.

[42] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *Computer Jornal*, 7:308–313, 1965.

[43] P. Petrovic, N. Mitrovic and M. Stevanovic. A Hysteresis Model for Magnetic Materials Using the Giles-Atherton Model. *IEEE Transactions on Magnetics MAG*, pages 803–808, 1999.

[44] E. Polak. *Optimization: Algorithms and Consistent Approximations*. Springer-Verlag, New York, 1997.

[45] H. Prahlad and I. Chopra. Modeling and Experimental Characterization of SMA Torsional Actuators. *Journal of Intelligent Material Systems and Structures*, 18:29–38, January 2007.

[46] D. Ribbenfjärd and G. Engdahl. Modeling of Dynamic Hysteresis with Bergqvist's Lag Model. *IEEE Transactions on Magnetics*, 42(10):3135–3137, October 2006.

[47] E. Rustighi, M. J. Brennan and B. R. Mace. Real-time Control of a Shape Memory Alloy Adaptive Tuned Vibration Absorber. *Smart Materials and Structures*, 14:1184–1195, 2005.

[48] S. Seelecke, C. Büskens, I. Müller and J. Sprekels. *Online Optimization of Large Systems: State of the Art*, chapter Real-Time Optimal Control of Shape Memory Alloy Actuators in Smart Structures, pages 93–104. Springer-Verlag, 2001.

[49] S. Seelecke and I. Müller. Shape Memory Alloy Actuators in Smart Structures: Modeling and Simulation. *Applied Mechanics Reviews*, pages 23–46, 2004.

[50] Y. Shen, E. Winder, C. A. Pomeroy and U. C. Wejinya. Closed-Loop Optimal Control-Enabled Piezoelectric Microforce Sensors. *IEEE/ASME Transactions on Mechatronics*, 11(4):420–427, August 2006.

[51] R. Smith, S. Seelecke, Z. Ounaies and J. Smith. A Free Energy Model for Hysteresis in Ferroelectric Materials. *Journal of Intelligent Material Systems and Structures*, 14:719–739, November 2003.

[52] G. Song, V. Chaudhry and C. Batur. A Neural Network Inverse Model for a Shape Memory Alloy Wire Actuator. *Journal of Intelligent Material Systems and Structures*, 14:371–377, June 2003.

[53] V. Torczon. *Multi-directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Rice University, Houston, Texas, USA, 1989.

[54] A. van der Schaft. *L_2-gain and Passivity Techniques in Nonlinear Control*. Springer-Verlag, London, 1996.

[55] J. C. Willems. Dissipative Dynamical Systems, Part I: General Theory. *Archives Rational Mechanics Anal.*, 45:321–351, 1972.

[56] M. H. Wright. Direct Search Methods: Once Scorned, Now Respectable. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1995: Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis*, pages 191–208. Addison Wesley Longman,Harlow, UK, 1995.