2015-06-01

# Temperature Control in Friction Stir Welding Using Model Predictive Control

Brandon Scott Taysom
*Brigham Young University*

# Temperature Control in Friction Stir Welding

# Using Model Predictive Control

Brandon Scott Boyer Taysom

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Carl D. Sorensen, Chair
John D. Hedengren
Tracy W. Nelson

Department of Mechanical Engineering

Brigham Young University

June 2015

ABSTRACT

Temperature Control in Friction Stir Welding
Using Model Predictive Control

Brandon Scott Boyer Taysom
Department of Mechanical Engineering, BYU
Master of Science

Temperature is a very important process parameter in Friction Stir Welding (FSW), but until lately active control of temperature has not been practiced. Recently, temperature control via a PID controller has proven to be effective. Model Predictive Control (MPC) is a control method that holds promise, but has not been attempted in FSW before.

Two different model forms are developed for MPC and are evaluated. The first is a simple first-order plus dead time (FOPDT) model. The second is the Hybrid Heat Source model and is more complex; it combines the heat source method and a 1D discretized thermal model of the FSW tool. Model parameters were determined by fitting model predictions to actual weld data. The models were evaluated for their performance in modeled and unmodeled disturbances once the process was already at a quasi steady state condition and also were evaluated for control immediately after plunge. The FOPDT based MPC controller has very good performance and was comparable in performance to previously proven and well-tuned PID controllers.

For small modeled disturbances the FOPDT controller settled within 1°C of the setpoint in 10s with almost no oscillations and only 2°C of overshoot. For large unmodeled disturbances, the FOPDT controller settled within 1°C of the setpoint in 30s with no oscillations and 16°C of overshoot. For the same disturbances, the PID servo controller settled in 30s with no oscillations and 9°C of overshoot, and the PID regulator controller settled in 15s but had almost a full oscillation and 13°C of overshoot.

The Hybrid Heat Source MPC controller and the PID regulator controller were also able to control temperature within 5°C of the setpoint immediately after the plunge during the highly transient portion of the weld, which previously had been assumed to be too difficult to control. The PID regulator controller had a high degree of variability between the two runs (a settling time of 10s and 30s, and .5 and 4.5 oscillations before settling), but settled quickly and once settled was able to hold the temperature within 2°C of the setpoint. The HHS MPC controller on the other hand had far fewer oscillations (0 and 1 oscillation) before settling, but could only hold the temperature within 5°C of the setpoint. Both of these controllers performed far better than the FOPDT MPC and PID servo controllers.

Keywords: MPC, model predictive control, FSW, friction stir welding, heat source method, temperature control, PID, process control

# ACKNOWLEDGEMENTS

I wish to express my appreciation to my committee members: Dr. Carl D. Sorensen for his mentoring both in this research and for his sharing of knowledge in FSW and many other topics over the last five years, to Dr. John D. Hedengren for his advice and intimate knowledge of control theory and optimization, and to Dr. Tracy W. Nelson for his mentoring in FSW and metallurgy over the last five years. I thank CFSP for funding my research and graduate education. I thank my parents for raising me, for continued mentoring, for encouraging me to obtain a quality education, and for significant financial support as I have been in the process of obtaining my education, particularly during my undergrad years. I thank my peers for collaborating on many and varied projects which led to learning and insight for all. And lastly, but certainly not least, I thank God who has helped me to have these and other uplifting experiences and has given me life and with it the opportunity to learn and grow.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# 1 Introduction

Friction Stir Welding (FSW) is a solid-state hot-deformation process developed in 1991 at TWI in the UK and is usually used to join two pieces of metal. In this process a tool is rotated and pushed into the seam of two work pieces. This action creates heat and softens the metal. Once the metal is soft enough, the tool travels along the seam of the two pieces joining them by stirring the metal together. Another process which is very similar to FSW is Friction Stir Processing (FSP). In FSP, the tool travels through a single work piece instead of along the seam of two pieces. While operating parameters may differ slightly between FSW and FSP, the fundamental physics are the same.

FSW has several advantages over traditional welding that make it attractive in many circumstances. FSW does not melt the basemetal, and thus microstructure is altered much less than traditional welding. Where normal welding often weakens the basemetal down to 40-70% of the original metal's strength, FSW results in a product that usually has about 70-85% of the strength of the original metal (Lakshminarayanan 2007; Mahoney 1997). Because of the lower temperatures and lack of melting, the cracking, porosity, residual stresses and post-weld deflections in FSW are much lower than in traditional welding (Lakshminarayanan 2007; Mahoney 1997).

## 1.1 A Brief History of FSW Control and Current Control Capabilities

FSW was first implemented on modified milling machines, and consequently movement along the three primary axes and spindle rotation were the first parameters to be controlled. FSW was primarily done by selecting a depth, travel speed, and spindle rotation speed. This method and other variants of it (for instance, controlling z force instead of z position, or by using pseudo heat indices) have been used successfully for many years (Chimbli 2007; Ross 2012).

FSW is an inherently temperature-dependent process. If the temperature is too high or too low, the strength and quality of the weld will be affected because microstructure will be negatively altered by the temperature. In some cases, the welded piece will be completely unusable if the temperature control is poor (Cederqvist 2011). However, if a weld is run at constant input parameters, the temperature in the weld may change over time due to transients and disturbances, which can result in nonoptimal material properties.

Ross did work on FSW system identification and control (Ross 2012). He identified the system primarily as a first-order plus dead time (FOPDT) system, and used a PID controller to control the system. Ross's PID controller is a single-input single-output (SISO) controller, with the control variable (CV) being temperature, and the manipulated variable (MV) being power. Ross was able to control weld temperatures in both aluminum and steel within one degree Celsius. A block diagram of a PID controller for FSW is shown in Figure 1-1.

Dustin Marshall built upon Ross's work and refined the process of system identification by using a relay step test to determine the time constant, time delay, and gain of the system (Marshall 2013). Marshall then used these better system parameters to determine PID gains for regular welding, disturbance rejection, and other things as well. In aluminum, Marshall was able to produce welds that had (on average) about 5°C overshoot, a settling time of slightly over 12

seconds, and no steady state error. However, using a PID controller immediately after start-up during the transient portion of the weld has not been successfully implemented because current PID controllers do not respond well to these large initial error and heavy transient situations. Consequently, keeping the temperature steady at the set point soon after start-up has been problematic.

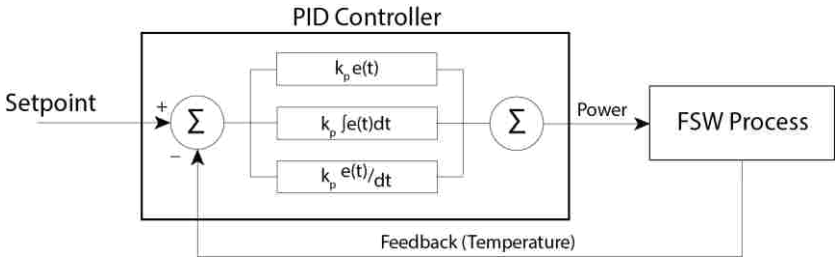



Figure 1-1: Block diagram of a PID temperature controller for FSW

## 1.2 MPC Overview

Model predictive control, or MPC, is a control method that predicts future responses of the system to various inputs based upon a model of the system. The form of a model is either determined from physics principles, derived from system response to input variable changes, or both. Likewise, parameter values are typically calculated from fundamental material or system properties, by fitting this model to actual data sets, or a combination of the two. Finally, during implementation, MPC is set within an optimization loop that optimizes a set of future moves to bring the output variable as close to the setpoint as is theoretically possible within constraints that the user specifies. This is shown as a schematic in Figure 1-2.

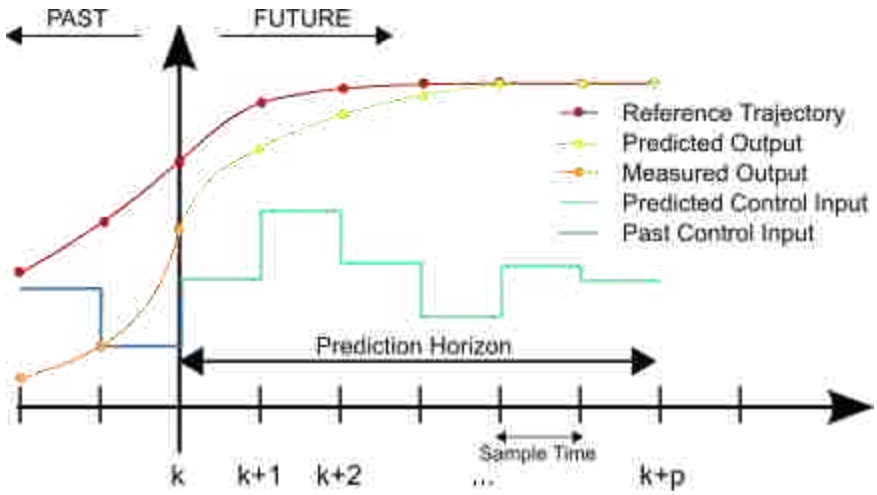



Figure 1-2: Schematic of MPC inputs and outputs (Behrendt 2009)

In Figure 1-2 the red line shows the reference trajectory, or the goal output. This can be a constant value or can be a curve to try to ease the CV to the desired set point as shown in the above schematic. A prediction function uses the past and current state of the system and a set of proposed future inputs to create a predicted future output. An optimization engine then alters the proposed future inputs or MVs to create future predicted outputs (the CVs) that match the reference trajectory, as much as possible within constraints. The optimization may be subject to constraints on the inputs or outputs. Constraints are usually for the purposes of product quality, to reduce machine wear, human safety, environmental considerations, and other factors.

For FSW, power is usually the MV and temperature the CV, and this is the same control architecture that will be used in this thesis. It is possible to change variables other than power as well (such as the traverse speed) within MPC and have a multi-input single-output (MISO) or multi-input multi-output (MIMO) system. For MPC control within this thesis, FSW is a SISO process. A block diagram for a FSW MPC controller is shown in Figure 1-3.

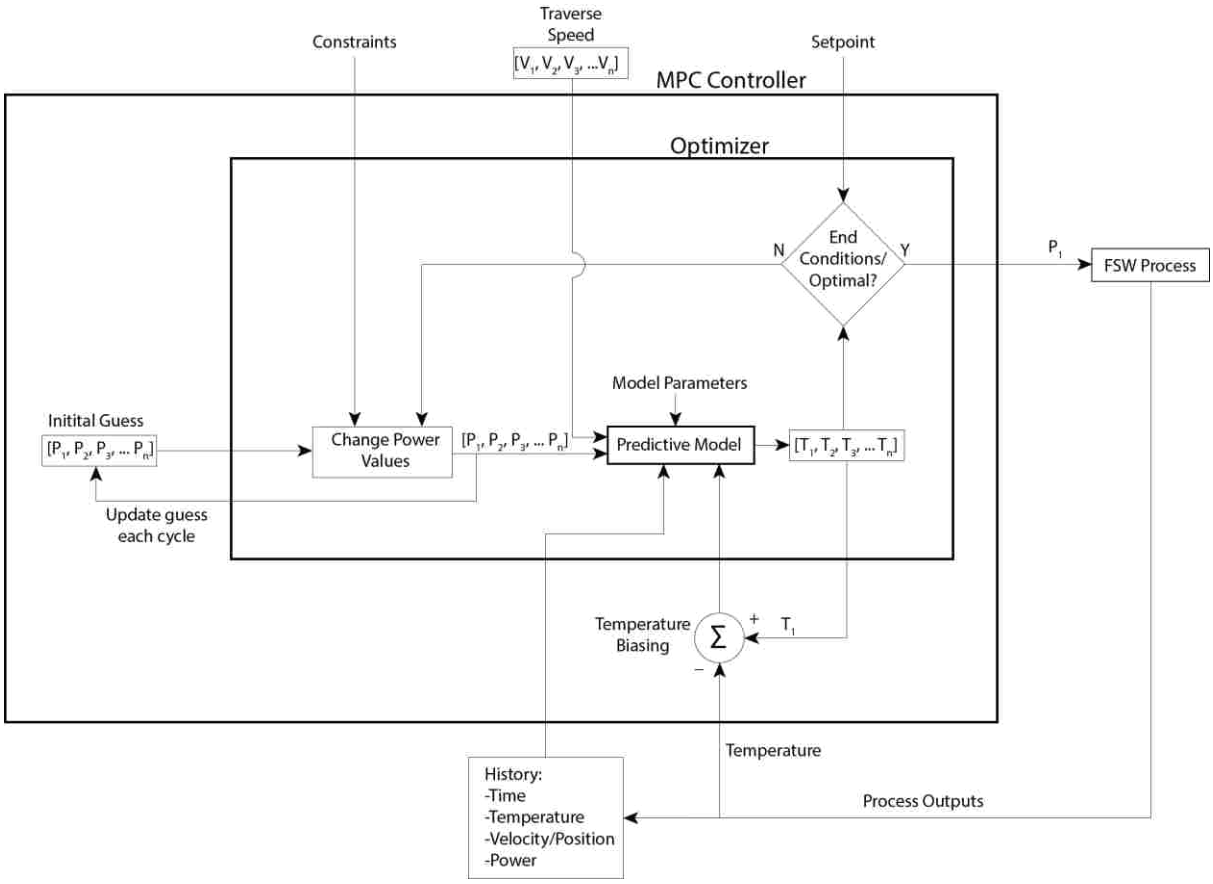



Figure 1-3: Block diagram of a SISO MPC temperature controller for FSW

## 1.3 Potential Benefits of MPC, Potential Pitfalls, Parameters, and Requirements

In an ideal world, MPC gives an "optimal" control where the process is as the closest it can be to its desired state within user-defined constraints. This optimal control may be effectively achievable when the process model matches the actual process very well and when disturbances are favorable. Disturbances are favorable when there are no disturbances, the disturbances are minor, or the disturbances can be measured and accounted for in the process model. When the process model is not as accurate and disturbances are more egregious, controller performance is degraded. As is similar with other controllers, MPC can be worse than no control, manual control, or a very simple controller if the model is very inaccurate.

All MPC controllers use a prediction horizon over which they will predict results (see Figure 1-2). This horizon is discretized into time steps. These time steps can be uniform in spacing. Alternatively, the horizon can be divided unequally with fine resolution in the near future and coarser at further time; this usually results in fewer total time steps for the same total time horizon. The latter may be more difficult to implement at first, but has the advantage of being computationally efficient while still retaining good prediction capabilities. The possible input at each individual time step is a variable that the optimizer must change and alter. In many optimizers, the relationship between number of variables to optimize and the optimization time is at least linear, if not quadratic or higher. Thus, by lessening the number of time steps by using a nonuniform time horizon, the computation time can be drastically decreased. This is especially important when the predictive function – and consequently the objective function – is computationally expensive. The length and discretization of the prediction horizon are important parameters in an MPC controller.

How aggressively the MPC controller accounts for disturbances or discrepancies is also important. This is often done with a biasing parameter. Aggressive biasing can lead to quick recovery from a disturbance, but in some cases an aggressive bias can lead to excessive overshoot or instability. Other important parameters are the limits. Oftentimes, there are high and low limits on the inputs, either for machine safety or regularity of the process. Rate of change limits for the inputs may be added as well. If a limit can be mathematically defined, it can be implemented, although numerous or very complex limits may slow down an optimizer.

In order to implement MPC, a computer must be sufficiently powerful to run the optimization quickly enough. For an unconstrained FOPDT SISO system with a small time horizon that only needs to update at 1 Hz, the .043 MHz computer on the first moon lander

would likely be powerful enough (Saran 2009) if the entire routine were coded efficiently. As the system becomes more complex and the model becomes less reflective of reality, an increasingly powerful computer is needed, and for some real world applications supercomputers may be needed to solve the problem rapidly enough (Zitney 1999).

## 1.4 Current State of MPC in FSW Research

Nielsen did theoretical work of implementing MPC for the FSW of large copper canisters (Neilsen 2012). He developed a complex physics-based model, and in simulation his MPC controller consistently outperformed other controllers that he was comparing against, even while staying within user-defined limits of depth, force, and temperature. However, his work was theoretical and has not yet been applied to an actual welding process.

## 1.5 Thesis Objectives

The purpose of this thesis is to implement MPC for FSW, to determine its suitability for FSW, and to compare it against the current best controllers used in FSW. In addition to good disturbance rejection and tight control to a set point, it is especially hoped that implementing MPC in FSW will reduce start-up transients and errors, which most current controllers have not been able to do well.

# 2 Models

A model is necessary for an MPC controller to make predictions. An accurate model will lead to accurate predictions made by the controller during optimization, and good control will follow. As models become less accurate and capture fewer of the system dynamics, controller performance will consequently degrade.

CFD methods have been used to model the basemetal as a complex non-Newtonian fluid with heat generated due to metal deformation and tool-basemetal slip. Even with these complex simulations which took hours to solve, FSW has proven difficult to model accurately (Posada 2012). Thus, extremely accurate models of FSW are not likely to be available within the foreseeable future. However, in order to control temperature, the model which the MPC controller uses only needs to predict the temperature reasonably well. As long as the major elements of the process are captured, small errors can be acceptable as they can be compensated via biasing, on-the-fly model corrections, or other methods.

In this investigation, two primary models will be considered, both of which are based upon heat transfer. One is a simple first-order model. Both Ross and Marshall noted that FSW was nearly a FOPDT system, and thus a first-order model holds promise. Another model is a modified heat source model with a tool FEA. The heat source method uses a history of point heat sources to calculate the temperature at any point and time in a semi-infinite solid, and for the hybrid model this is coupled with a 1-D thermal FEA of the tool.

## 2.1 First-Order Plus Dead Time Model

The first-order plus dead time model is based upon a simplification of the process with different regions of the weld, as shown in Figure 2-1. Each region is assumed to be isothermal and these regions provide for major heat transfer modes via conduction and advection. Naturally these regions do not exactly match reality, but a rough model is likely acceptable because model parameters will be fit from data instead of derived from relevant theoretical material properties, geometry, and heat transfer equations.

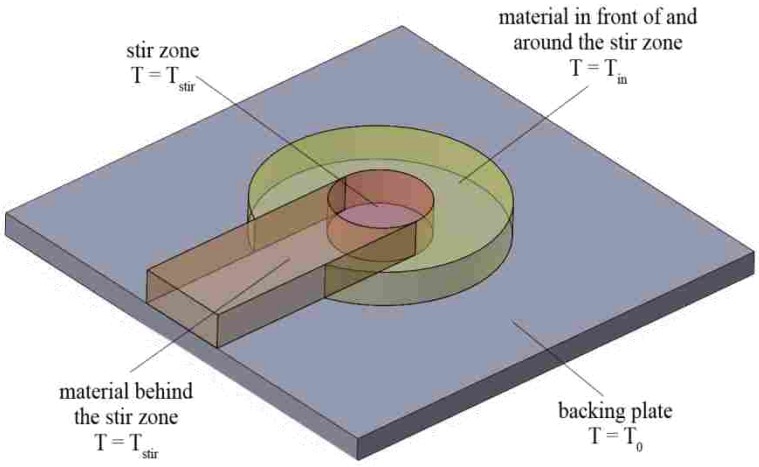



Figure 2-1: Regions of the first-order model that interact with, or are, the stir zone.

For this model the temperature of the stir zone will be controlled. The basic dynamic temperature equation of the stir zone is:

$$\frac{dT_{stir}}{dt} = \frac{E_{in}}{m \cdot c_p} \tag{2-1}$$

which (with some substation of terms for $E_{in}$) is equivalent to:

$$\frac{dT_{stir}}{dt} = \frac{a \cdot \mathrm{Q}_{power} + b \cdot \mathrm{Q}_{\mathrm{cond}} + c \cdot \mathrm{Q}_{\mathrm{adv}}}{\tau} \tag{2-2}$$

with $\tau$ being the time constant of the system. The locations for these major energy transfers are shown in Figure 2-2.

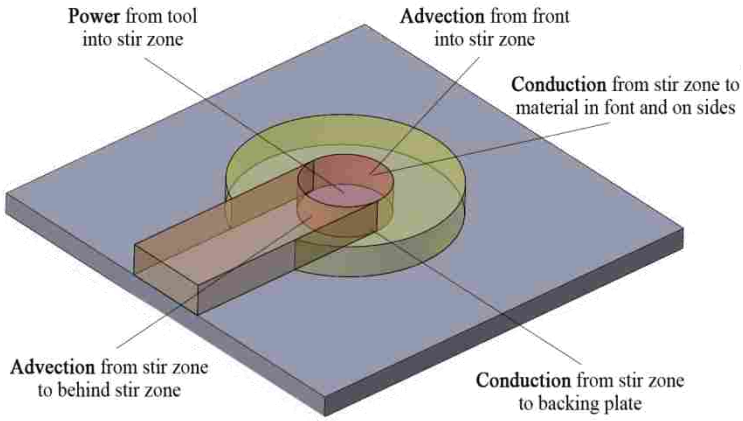



Figure 2-2: Major modes and locations of energy transfer in the FOPDT model.

### 2.1.1 Power Term

In this model power is assumed to be input into the stir zone from the rotation of the tool. Whether this happens at the top of the stir zone at tool-plate interface or inside the stir zone is inconsequential because all of the regions of this model are assumed to be isothermal. Thus power into the stir zone is:

$$Q_{power} = P \tag{2-3}$$

### 2.1.2 Conduction Term

Conduction occurs from the stir zone to both the material in front of and to the side of the stir zone and to the backing plate. Conduction of heat in this model is not assumed to flow to the

region behind the tool because as the tool progresses, stir zone material is left in the region behind it. Thus the area immediately behind the stir zone initially has the same temperature as the stir zone itself and no conduction will occur. It is assumed that conduction does not happen up the tool as previous studies have estimated heat loss up the tool to be minimal. Only about 3-5% of the energy during the weld is lost by conduction up the tool if there is a linear thermal gradient up the tool. Alternatively, the tool can be considered to be actively cooled and in contact with the large FSW machine. The machine also has a temperature of $T_0$, and thus can be lumped together with the backing plate. The net conduction of heat *into* the stir zone is:

$$Q_{cond} = h_1 \cdot A_1 \cdot (T_{in} - T_{stir}) + h_2 \cdot A_2 \cdot (T_0 - T_{stir}) \quad (2\text{-}4)$$

### 2.1.3 Advection Term

Advection occurs due to the transport of material in and out of the stir zone. However, since the material behind the stir zone is assumed to be the same temperature as the stir zone, this term will be 0. The advection term thus only comes from the cold material in front of the weld:

$$Q_{adv} = \dot{m} \cdot c_p(T_{in} - T_{stir}) \quad (2\text{-}5)$$

When taking into account that mass flow rate is directly dependent upon the traverse speed, *v*, (2-5) becomes:

$$Q_{adv} = v \cdot A \cdot \rho \cdot c_p(T_{in} - T_{stir}) \quad (2\text{-}6)$$

### 2.1.4 The $T_{in}$ Term

The material in front of the stir zone is heated as a result of the conduction from the stir zone and (at steady state) will be somewhere between $T_{stir}$ and $T_0$. It should not be a constant value; as $T_{stir}$ increases, $T_{in}$ should increase as well. Also, as travel speed increases, temperature

from the stir zone has less of a chance to conduct forward and preheat that material, and thus $T_{in}$ should decrease.

A simple 1-D thermal FEA was conducted to determine an appropriate relationship between the travel speed, $v$, and the temperature of the metal entering the stir zone, $T_{in}$. For this simulation, the tool temperature was kept constant, the nodes in the stir zone were constrained to be equal to the tool temperature, and any other heat transfer was due to conduction of heat from the stir zone. This simulation was run to steady state for a range of travel speeds that are appropriate to FSW. Figure 2-3 shows the results of these simulations.

In this simulation the nodes under the tool were constrained to the tool temperature; it is therefore pointless to look at the temperature of a node which has just entered the stir zone. Instead, the thermal gradient of the nodes in front of the tool should give a good indication of how much lower in temperature a point a small distance in front of the tool is than the stir zone. The results from Figure 2-3 suggest a linear relationship between travel speed and how much lower the temperature in front of the tool is, and by extension a linear relationship between travel speed and $T_{in}$. Thus, the relationship for $T_{in}$ should be of the form:

$$T_{in} = T_{stir} - c \cdot v \tag{2-7}$$

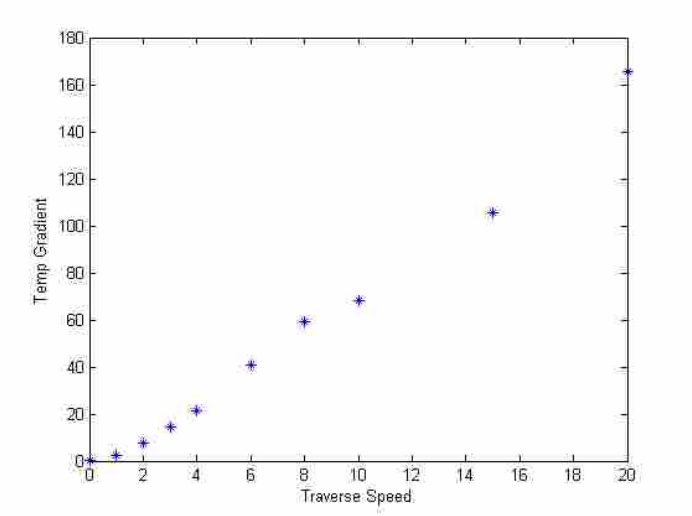



Figure 2-3: Results from 1-D FEA simulations showing the relative thermal gradient ahead of the tool vs. the travel speed. There is a nearly linear relationship between travel speed and the thermal gradient in front of the tool.

### 2.1.5 Complete First-Order Model

Substitution of Equations (2-3), (2-4), and (2-6) into (2-2) results in:

$$\frac{dT_{stir}}{dt} = \frac{a \cdot P + b \cdot v \cdot A \cdot \rho \cdot c_p \cdot (T_{in} - T_{stir}) + c \cdot [h_1 \cdot A_1 \cdot (T_{in} - T_{stir}) + h_2 \cdot A_2 \cdot (T_0 - T_{stir})]}{\tau} \quad (2\text{-}8)$$

Substituting (2-7) into (2-8), and condensing the parameters, material properties, and weighting terms into single variables results in a model of the form:

$$\frac{dT_{stir}}{dt} = \frac{c_1 \cdot P + c_2 \cdot v + c_3 \cdot v^2 + c_4 \cdot T_0 - T_{stir}}{\tau} \quad (2\text{-}9)$$

The resulting model is first-order with respect to power which is in harmony with what others have observed (Ross 2012). Equation (2-9) can also be written such that $\tau$ is not in the denominator and instead $T_{stir}$ is multiplied by a constant. This has the advantage that the parameters to be determined by curve fitting do not at all interrelate (which is easier for an optimization routine), but the disadvantage that the time constant of the model is no longer explicitly stated. However, as long as the curve fitting or optimization routine works correctly,

both expressions should be mathematically equivalent and the model should behave in exactly the same way. Parameters for Equation (2-9) were determined and are shown in Table 4-1.

### 2.1.6 Assumptions

As with all models, this model has several assumptions inherent to it. All regions are assumed to be isothermal. The backing plate reservoir temperature, $T_0$, is assumed to be constant and will therefore not heat up as the weld progresses. Either heat loss up the tool is not important *or* the tool and backing plate can be considered as one object. The geometry and thermal properties of the regions do not change with temperature or travel speed, and thus the model parameters do not change with these. As the tool changes depth in a weld, no system parameters are affected. The temperature of the thermocouple in the tip of the tool is the temperature of the stir zone.

Naturally the above-listed assumptions are wrong to varying degrees. However, if they are either negligible or can be compensated for via the fitting of model parameters, then the model can be useful for predictions in an MPC controller.

## 2.2 Heat Source Model

### 2.2.1 The Heat Source Method

The heat source model is based on the heat source method, which is a solution to an instantaneous point heat source in an infinite solid (Carslaw 1959, 256; Hou 2000). This solution is:

$$\theta = \frac{Q_{pt}}{8c\rho(\pi a\tau)^{3/2}} \cdot e^{-R^2/4a\tau} \tag{2-10}$$

Refer to Table 2-1 for the definitions of parameters of the heat source model used in this section and Section 2.3. The rise in temperature of a semi-infinite solid for the same amount of energy input would be exactly twice the temperature rise of the infinite solid as listed above.

Table 2-1: Definitions of Terms in the Heat Source Method

| Term | Definition | Units |
|---|---|---|
| $Q_i$ | Rate of heat entering the solid over a given time period | W |
| $Q_{pt}$ | Amount of heat entering the solid over a given time period/instant | J |
| $R$ | Distance between the point heat source and observation point | m |
| $a$ | Thermal diffusivity of the solid. Equal to k/ρc. | $m^2/s$ |
| $c$ | Heat capacitance of the solid | J/Kg°C |
| $k$ | Thermal conductivity of the solid | W/m°C |
| $t$ | Time relative to observation – zero at the time of observation, negative for the past, and positive for events in the future of the observation | s |
| $t_0$ | Total amount of time that the process has been happening | s |
| $\theta$ | Temperature rise at point of observation | °C |
| $\rho$ | Density of material | $Kg/m^3$ |
| $\tau$ | Time elapsed between heat entering the solid and observation | s |

Equation (2-10) can be transformed to a moving and variable heat source by integrating a series of instantaneous heat source points over the history of the heat source. For a semi-infinite plate, this results in:

$$\theta = \frac{1}{4c\rho(\pi a)^{3/2}} \int_{-t_0}^{0} \frac{Q_i}{\tau_i^{3/2}} \cdot e^{-R_i^2/4a\tau_i}\, dt \tag{2-11}$$

By evaluating Equation (2-11) over the history of the potentially variable and moving point heat source, a total rise in temperature, $\theta$, can be calculated at any past, present, or future time and at any point in the semi-infinite solid. By specifying a point in space and time that is in

the stir zone, predictions of future temperatures can be made and this can be used as a model for an MPC controller.

When evaluating Equation (2-11), it is important to note a few things. First, in Equation (2-11), time = 0 is the current time or the time at which the change in temperature is desired. Thus, if the process has been happening for a total time of $t_{total}$, then one must integrate from back in time to the present, or from $-t_{total}$ to 0. For simplicity, this is usually written as $-t_0$ to 0. Second, the substitution $t = -\tau$ should be used when evaluating (2-11). Thus, $\tau$ will have a small positive value for events in the recent past, and $\tau$ will have a large positive value for events in the distant past. Third, (2-11) becomes indeterminate at $\tau = 0$, so the equation should be evaluated until $\tau$ is a very small number, such as .00001. Unless the point of desired temperature is extremely close in both space and time to the point heat source, not evaluating the last .00001 have virtually no impact on the results. If time or space is micro or nano-scale, the small number may simply be reduced further, but not so far that the computer will round or truncate the number to 0. Fourth, events that are distant both in time and space have an extremely small impact. For a quasi steady state process, events occurring before $20a/v^2$ can be ignored as the error from ignoring them will be inconsequential (Hou 2000). This however assumes consistent linear movement, and if for instance the heat source was to take a circular path and come close to a previous position, ignoring what happened in the distant past may no longer be inconsequential. This may also be invalid for points that are extremely deep in a semi-infinite solid as heat far in the past will take a long time to conduct to the point of interest.

By nature, this model does not have any adjustable parameters per se. In practice, the depth of the point at which the temperature is desired can be changed, and either an additive or multiplicative factor can be added to the power term of the model. This will change the result

and these parameters can be chosen to help fit the predicted temperatures to actual temperatures experienced in the weld. The depth parameter should be somewhat close to the actual depth of the thermocouple tip within the plate, and preferably the power should be close (presumably not less than 50% or greater than 200%) to the actual power so that the model has a wide range of applicability and is close to the fundamental physics. If a very small or large value is chosen, it may perhaps fit a specific circumstance well, but will most likely be much less accurate for a wider range of circumstances.

### 2.2.2 Assumptions

The heat source method assumes that material properties are constant regardless of the temperature of the solid. There is no material flow anywhere in the plate – energy transfer only happens by convection and never advection. No heat flows up the tool, which at steady state is estimated to be about 3-5% of the total power. The tool does not occupy any space within the plate. The point of observation is equivalent to the point at which the thermocouple is measuring temperature.

## 2.3 Hybrid Heat Source FEA Model

The hybrid model is the heat source model with a 1-D discretized tool that interfaces with the plate and where heat can transfer between the plate and the nodes of the tool that are in contact with the plate. Additional terms are needed in excess of the heat source model and are defined in Table 2-2.

Table 2-2: Definitions of Additional Terms in the Hybrid Heat Source Model

| Term | Definition | Units |
|---|---|---|
| $A_n$ | Area of tool node *"n"* which is in contact with the plate | $m^2$ |
| $A_x$ | Area at the boundary of the element and a neighbor/interacting element, element *"x"* (usually w, e, 0 or c) | $m^2$ |
| $Q_i$ | Power dissipated into the plate at a given time | W |
| $Q'_i$ | Total power dissipated into the tool/plate system at a given time | W |
| $Q_{i,tool}$ | Power dissipated into the tool at a given time | W |
| $T_{plate,0}$ | Initial temperature of plate | °C |
| $T_x$ | Temperature of node/element *"x"* (usually w, e, 0 or c) | °C |
| $V$ | Volume of material in a tool node | $m^3$ |
| $c$ | Heat capacitance of the tool at the node in question | J/Kg°C |
| $h_0$ | Convection coefficient between the plate and tool nodes | $W/m^2°C$ |
| $h_n$ | Convection coefficient between the point of observation in the plate and tool node *"n"* | $W/m^2°C$ |
| $h_x$ | Convection coefficient between the element and a neighbor/interacting element, element *"x"* (usually w, e, 0 or c) | $W/m^2°C$ |
| $k_x$ | Thermal conductivity of the tool at the boundary of the element and a neighbor/interacting element, node *"x"* (usually w, e, 0 or c) | W/m°C |
| $p_{mult}$ | Scaling factor for artificially increasing/decreasing power values | unitless |
| $z_{pt}$ | Depth of the point of observation in the plate; this affects the value of R | m |
| $\delta_x$ | Distance between centers of neighbor/interacting node, node *"x"* (usually w, e, 0 or c) | s |
| $\Delta t$ | Total time duration of the current step | s |
| $\Theta_0$ | Rise of temperature at point of observation due to known events from the distant past up until the somewhat recent past | °C |
| $\Theta_1$ | Rise of temperature at point of observation due to events from the recent past due to energy dissipated into the plate | °C |
| $\Theta'_1$ | Rise of temperature at point of observation due to events from the recent past if 100% of that energy were dissipated into the plate instead of some into the plate and some into the tool | °C |
| $\Theta_{plate}$ | Same as θ | °C |
| $\rho$ | Density of tool at the node in question | $Kg/m^3$ |

### 2.3.1 Motivation for Developing the Hybrid Model

The hybrid model was originally developed because while in some circumstances the effect of the tool on the stir zone temperature is largely negligible, in other circumstances it can have a very large effect.

For a H13 tool welding aluminum, at steady state the system may apply about 2500 W to the plate with a tool tip temperature of about 450°C. If the tool is about 10 cm (4 inches) long and 2.54cm (1 inch) in diameter, and the thermal conductivity is 16 W/m·K, and a linear profile is assumed from the tip of the tool to a reservoir at the other end of 25°C, the tool will remove heat at a rate of 135 W, which is about 5% of the total heat being input to the system.

However, in order to get up to this steady state condition, assuming a specific heat of about 500 J/Kg·K and a density of 7800 Kg/m$^3$, about 45 KJ of thermal energy must be put into the tool. This is very close to the energy spent during the plunge of a weld, which was calculated from several different welds to be about 43 KJ (see Appendix D.1). Thus, omitting the amount of energy that travels up the tool during the start-up of a weld will lead to greatly over-predicted plate temperatures during the plunge (oftentimes peaking at about 600 - 700°C for an aluminum weld), and this was consistently observed while using the regular heat source model in an MPC controller.

### 2.3.2 Heat Transfer Equations for FEA of Tool Nodes

Before an FEA of the tool can occur, the tool must be first discretized. An example of a 1-D discretized tool is shown in Figure 2-4. The tool is discretized finer at the tip because this will help to capture quick changes much more accurately than large elements would.

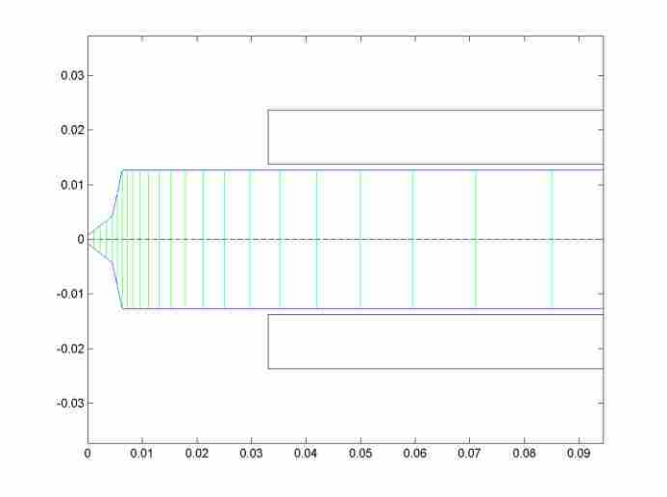



Figure 2-4: Example of a meshed tool in preparation of FEA analysis. The scale is in meters.

The derivation will start with an energy balance on a given element and will derive the standard form of the implicit Euler transient heat transfer equations for a node *p*. The implicit Euler form will be first-order accurate in time, and has the advantage of being unconditionally stable for any size of time step. A given element may touch nodes on its two sides, traditionally called "west" and "east" or *w* and *e*, and in the case of FSW may also touch the plate or tool collar, which are nodes *0* and *c*, respectively.

For a node, the temperature change over a given time period is:

$$\Delta T_p = \frac{q_{in} \cdot \Delta t}{\rho \cdot c \cdot V} \tag{2-12}$$

where $q_{in}$ is the sum of energy into the node due to neighboring node conduction and conduction/ convection of source terms. This can be expressed as:

$$q_{in} = \sum a_x (T_x - T_p) \tag{2-13}$$

where $a_x$ is either $k_x \cdot A_x/\delta_x$ for the conduction of a neighboring node *x* with the primary node or can be $h_x \cdot A_x$ for the convection of node/space *x* with the primary node, and where $T_x$ and $T_p$ are

the temperatures of the *x* and primary node, respectively. Splitting (2-13) for the explicit summation for nodes *w*, *e*, *0*, and *c*, and combining with (2-12) results in:

$$\Delta T_p \cdot \left(\frac{\rho \cdot c \cdot V}{\Delta t}\right) = a_w(T_w - T_p) + a_e(T_e - T_p) + a_0(T_0 - T_p) + a_c(T_c - T_p) \qquad (2\text{-}14)$$

By substituting $a_p^0 = \frac{\rho \cdot c \cdot V}{\Delta t}$ and $\Delta T_p = \left(T_p - T_p^0\right)$, where $T_p^0$ is the previous temperature of the node, the standard form of the equation as it relates node *p* to *w*, *e*, *0*, and *c* is:

$$a_p^0(T_p - T_p^0) = a_w(T_w - T_p) + a_e(T_e - T_p) + a_0(T_0 - T_p) + a_c(T_c - T_p) \qquad (2\text{-}15)$$

If the node *p* is not in contact with either the plate or the tool collar, then the coefficients $a_0$ and $a_c$ are zero and those terms drop out of the equation. Equation (2-15) can be rearranged for simultaneous solution via a matrix to:

$$(a_p^0 + a_w + a_e + a_0 + a_c)T_p - a_w T_w + a_e T_e + a_0 T_0 = a_p^0\, T_p^0 + a_c\, T_c \qquad (2\text{-}16)$$

where the temperatures on the left hand side of the equation will be solved for in the matrix solution, and everything else is a constant that goes into either the A or b matrix. Table 2-3 shows a summary of the $a_x$ values in (2-16). Since individual nodes may have different geometries, the $a_x$ values may be different for different nodes. For example, the value of $a_w$ for node #2 may be 1.4, but might be 1.8 for node #3. If a node is not in contact with the plate and/or the collar, then $a_0$ and/or $a_c$ is 0, respectively.

Table 2-3: Values of Parameters in Equation (2-16) for an Individual Node

| | |
|---|---|
| $a_p^0$ | $\rho \cdot c \cdot V/\Delta t$ |
| $a_w$ | $k_w\, A_w/\delta_w$ |
| $a_e$ | $k_e\, A_e/\delta_e$ |
| $a_0$ | $h_0 A_0$ |
| $a_c$ | $h_c A_c$ |

### 2.3.3 Modification of Heat Source Equation and Coupling with Tool FEA Equations

The heat source equations need to be slightly modified in order to be properly coupled with the tool equations. This is because the plate will be losing energy to the tool and consequently this energy will not heat the plate.

To more easily discuss terms, (2-11) will be simplified such that:

$$\int_a^b \psi_i(t)dt \overset{\text{def}}{=} \frac{1}{4c\rho(\pi a)^{3/2}} \int_a^b \frac{Q_i}{\tau_i^{3/2}} \cdot e^{-R_i^2/4a\tau_i} dt \tag{2-17}$$

In this and all other equations, $Q_i$ is the rate of energy into the plate. The term $Q_i'$ shall be defined as the total rate of energy put into the system at a given time (in other words, the spindle power) and is the sum of the rate of energy that goes into the plate and rate of energy into the tool:

$$Q_i' \overset{\text{def}}{=} Q_i + Q_{i,tool} \tag{2-18}$$

It then follows that the temperature rise over a period of time if all of the energy from the spindle entered the plate (instead of some entering the plate and some the tool) would be:

$$\int_a^b \psi_i'(t)dt = \frac{1}{4c\rho(\pi a)^{3/2}} \int_a^b \frac{Q_i'}{\tau_i^{3/2}} \cdot e^{-R_i^2/4a\tau_i} dt \tag{2-19}$$

Based upon (2-11) and (2-17) the temperature rise at a given point in the plate up to the current time is thus:

$$\theta = \int_{-t_0}^{0} \psi_i(t)dt \tag{2-20}$$

which can easily be split into two separate integrals:

$$\theta = \int_{-t_0}^{-\delta t} \psi_i(t)dt + \int_{-\delta t}^{0} \psi_i(t)dt \tag{2-21}$$

For past values up until the most recent time step ($\delta t$), the amount of energy $Q_i$ into the plate vs. into the tool will already be known and calculated, and thus the first integral of (2-21)

can be completely calculated. However, this is unknown for the most recent time step until it is calculated. If $Q_{i,tool}$ and $Q_i'$ were known for the most recent time step, then $Q_i$ could easily be calculated.

Three more terms will be defined for simplicity of discussion: $\Theta_0, \Theta_1$, and $\Theta_1'$. The first is the rise of temperature of the plate due to events in the distant past, the second is the rise of temperature due to events in the recent past, and the third is what the rise in temperature due to events in the recent past would be if the entire spindle energy was transferred 100% into the plate.

$$\Theta_0 \stackrel{\text{def}}{=} \int_{-t_0}^{-\delta t} \psi_i(t)dt \tag{2-22}$$

$$\Theta_1 \stackrel{\text{def}}{=} \int_{-\delta t}^{0} \psi_i(t)dt \tag{2-23}$$

$$\Theta_1' \stackrel{\text{def}}{=} \int_{-\delta t}^{0} \psi_i{}'(t)dt \tag{2-24}$$

Equation (2-21) is rewritten with substitution from (2-22) and (2-23) to more explicitly show that each of the parts directly constitutes a discrete change in temperature of the plate:

$$\Theta_{plate} = \Theta_0 + \Theta_1 \tag{2-25}$$

Since $\theta$ is linearly proportional to Q as shown by Equation (2-1), the following is true:

$$\Theta_1 = \Theta_1' \left(1 - \frac{Q_{tool}}{Q_i'}\right) \tag{2-26}$$

Thus, if the entire amount of power going into the system went into the tool, then (2-26) would evaluate to be 0, and if no power going into the system went into the tool, then (2-26) would evaluate as if $Q_i = Q_i'$, as it should. Substituting (2-26) into (2-25) gives:

$$\Theta_{plate} = \Theta_0 + \Theta_1' \left(1 - \frac{Q_{tool}}{Q_i'}\right) \tag{2-27}$$

with $T_0$ being the temperature of the plate and $T_1$ , $T_2$ , etc being nodes of the tool potentially in contact with the plate. We can then say that the energy flowing up into is the tool is:

$$Q_{i,tool} = [\, h_1 A_1 (T_0 - T_1) + h_2 A_2 (T_0 - T_2) + \ \ldots ] \tag{2-28}$$

Substitution of (2-28) into (2-27) results in:

$$\Theta_{plate} = \Theta_0 + \Theta_1' \cdot \{1 - [\, h_1 A_1 (T_0 - T_1) + h_2 A_2 (T_0 - T_2) + \ \ldots ] / Q_i' \} \tag{2-29}$$

If the entire plate is at nonzero initial temperature, $T_{plate,0}$ , then:

$$\Theta_{plate} = T_0 - T_{plate,0} \tag{2-30}$$

Substituting (2-30) into (2-29) and rearranging so that each of the temperature terms only appears once in the equation results in:

$$\alpha_0 T_0 - \alpha_1 T_1 - \ \alpha_2 T_2 - \ \ldots - \alpha_n T_n = \beta \tag{2-31}$$

where the Table 2-4 lists values for (2-31).

Table 2-4: Values for Parameters in Equation (2-31)

| | |
|---|---|
| $\alpha_0$ | $1 + (\, h_1 A_1 + \ h_2 A_2 + \ \ldots + \ h_n A_n) \cdot \Delta T_1' / Q_i'$ |
| $\alpha_1$ | $h_1 A_1 \cdot \Theta_1' / Q_i'$ |
| $\alpha_2$ | $h_2 A_2 \cdot \Theta_1' / Q_i'$ |
| $\alpha_n$ | $h_n A_n \cdot \Theta_1' / Q_i'$ |
| $\beta$ | $T_0 + \Theta_1' + T_{plate,0}$ |

With Equations (2-16) and (2-31), the A and b matrices can then be assembled to allow simultaneous solving of the new temperatures in the plate and tool. The values for each individual line come from Table 2-3 and Table 2-4.

The A matrix is:

$$= \begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & & & & \\ -a_0 & a_p & -a_e & 0 & \cdots & & 0 & \\ -a_0 & -a_w & a_p & -a_e & & & & \\ -a_0 & 0 & -a_w & a_p & & & & \\ & \vdots & & & \ddots & & \vdots & \\ & & & & & a_p & -a_e & 0 \\ & & 0 & & \cdots & -a_w & a_p & -a_e \\ & & & & & 0 & -a_w & a_p \end{bmatrix} \tag{2-32}$$

In Equation (2-32) the $\alpha_n$ term on the top row will technically continue all the way to the right-hand side, but it will be 0 when there is no contact between a node in the tool and the plate (as is the case for the latter nodes). The same is true of the $a_0$ terms in the first column of the matrix in (2-32). It should also be noted that the $a_x$ values in each row will likely be different from each other, as explained below.

The b vector is:

$$\vec{b} = \begin{bmatrix} \beta \\ a_p^0\, T_p^0 + a_c\, T_c \\ a_p^0\, T_p^0 + a_c\, T_c \\ \vdots \\ a_p^0\, T_p^0 + a_c\, T_c \end{bmatrix} \tag{2-33}$$

Similarly to the $a_0$ term in (2-32), if there is no contact between a node and the tool holder/collar, then the $a_c$ term in (2-33) equals 0. Alternately, if it is desired to constrain the end of the tool to the temperature of a chiller or other thermal heat sink, then the last row of (2-32) becomes all zeros except for the lower right-hand corner which becomes 1, and the bottom row in (2-33) becomes the temperature to which one wishes to constrain the end node, such as the $T_c$.

For Equations (2-32) and (2-33), the terms in each row are respective to the principle node of that row. As an example, the terms $a_p$ on rows 3 and 4 are for the tool nodes 2 and 3, respectively. Therefore, if the volume of those nodes is different, then the $a_p$ terms on each row

will be different as well. As another example, if the distance between nodes is nonuniform, then the $a_e$ and $a_w$ terms may also be different both within a row and from row to row.

The solution of the system of equations $A \cdot \vec{T} = \vec{b}$ will return the resulting temperatures for the end of the time step. This can be accomplished by matrix inversion or other matrix solution methods.

Some of the variables in this model will be changed from their theoretical values in this thesis in order to account for errors in the model and/or because some the actual values are unknown. These variables are (1) the depth of the point in the plate at which the heat source method is used and which couples the tool to the plate, $z_{pt}$, (2) the convection coefficient between the tool and the plate, $h_0$, and (3) the power multiplication factor which simply multiplies the spindle power by a factor to artificially increase/decrease the power into the system, $p_{mult}$.

### 2.3.4 Assumptions

The Hybrid Heat Source model has many of the same assumptions as the regular heat source model. Material properties are constant regardless of the temperature of the solid. There is no material flow anywhere in the plate – energy transfer only happens by convection and never advection. Heat flows up and down the tool, but only in one dimension. The tool does not occupy any space within the plate. The point of observation is equivalent to the point at which the thermocouple is measuring temperature. Nevertheless, all of the nodes of the tool that should be in contact with the plate contact the plate at the same point of a specified depth within the plate.

# 3 Model Accuracy and Reduction of Computational Time

Methods that use discretization of space and/or time to arrive at results should be evaluated for grid independence and validated. A grid independent solution is a solution where the results are no longer dependent upon the fineness of the gird or step size. If the solution is not independent, then the numerical solution changes as the grid changes. A validated model is one where the solution matches the solutions of other already existing and validated solutions, exact analytical solution, professional codes, experimental data, etc.

In grid independence, the step in space or time is usually reduced by half until the results change by a negligible amount. If the discretization is too coarse, results may not be as accurate as needed. If the discretization is too fine, while the results will usually be internally accurate and fully converged, the computational time is excessive. Furthermore, a model cannot be more accurate than the inputs to the model. For this work, 1% accuracy is acceptable, but .1% is desired if the computational cost is not prohibitive.

For a model to be validated, the results of the model must predict the same as a solution that is known to be correct. This is because even if the model is independent of its grid, if the results are completely wrong, then fully converged and grid independent results are useless. In FSW, no model exists that has been verified to accurately model the full process, and several different models which have been used to model the entire process all predict nontrivially different results and do not fully match real FSW data. Furthermore, measurement of the active

stir zone has proven to be extremely difficult. Thus, the models evaluated in this study will not be rigorously validated as there is nothing reliable to validate against. In absence of full validation, the results of the models will be compared against measured temperatures in the tip of the tool to make sure that they are reasonably correct.

## 3.1 Time Grid Independence of the Heat Source Model

The heat source model was studied for grid independence in time. This is because changes in the time spacing can have very large impacts on the end results. Data of a constant 3.81 mm/s (9 ipm) and 2900 W was fed into the model until it reached steady state, and the grid was altered. Since the tool moves along in space and time simultaneously, reducing the time steps also reduced the physical steps as well.

Two different schemes were used in time grid independence studies: a uniform and a nonuniform time discretization. These two schemes are discussed further in Section 3.3 and in Appendix A. At fine resolution, both schemes predicted a steady state temperature of 452.9°C. The absolute error percent vs. number of time divisions is shown in Figure 3-1. Using 100 divisions in time, the variable discretization scheme was within about .2% or 1°C accurate, which is probably sufficient for most purposes, although increasing the discretization to about 300 divisions would bring about .1°C accuracy. In contrast, the uniform time discretization scheme requires about 8,000 and 20,000 time divisions to achieve the same accuracy, respectively. Unless otherwise stated, all calculations based upon the heat source or Hybrid Heat Source models in this thesis use at least 200 discretizations for the nonuniform time step scheme.

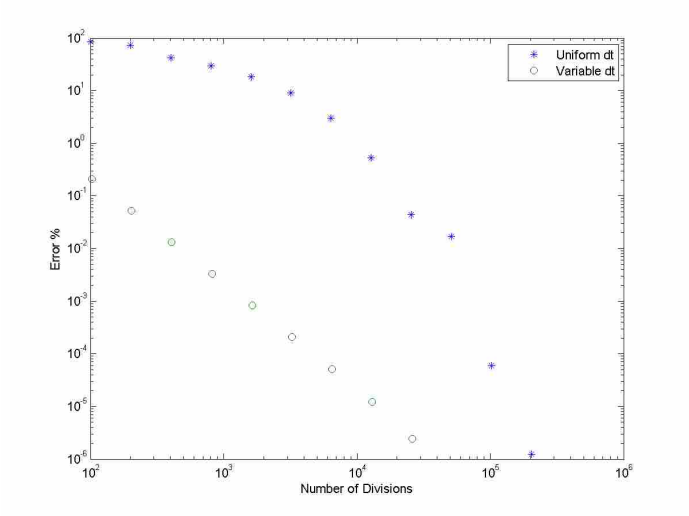



Figure 3-1: The percent error vs. number of time divisions for two different schemes of discretizing time. The uniform steady state temperature was calculated at 452.9°C. Thus, a 1% error is almost 5°C inaccurate. Note that for the uniform scheme, the error is exceptionally high at about 84% error for 100 nodes.

## 3.2 Spatial Grid Independence of the Heat Source Model

The spatial grid independence of the heat source model as it is applied to FSW is also important. This is because the tool is not in actuality a point heat source; on the contrary, it is similar to a distributed heat source. In order to determine the gird independence of the tool spatially, the tool was meshed symmetrically in both radial and circumferential directions to different fidelities; this is shown in Figure 3-2. The cross-section of a semi-infinite plate was examined for a few cases of coarse to extremely fine meshes. Plots of the temperature in the cross-section are shown in Appendix B. They show dramatic changes in predicted temperatures for coarse meshes and very similar results for the finest meshes. There is clearly a point of diminishing returns.

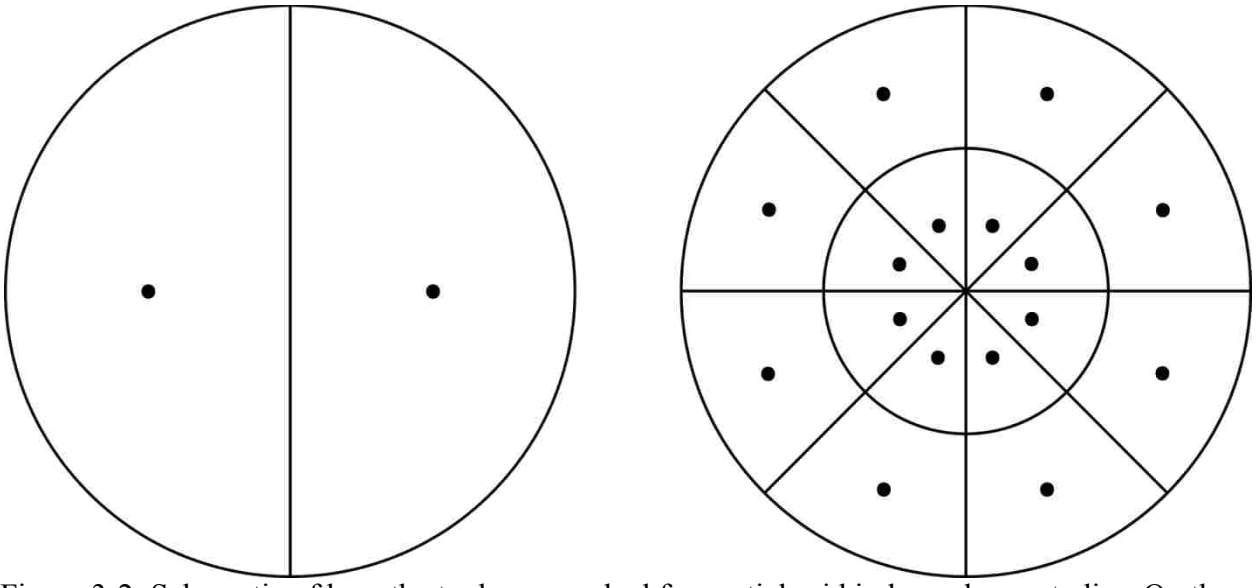


Figure 3-2: Schematic of how the tool was meshed for spatial grid independence studies. On the left is a tool with one radial division and two circumferential divisions; on the right the tool has two radial divisions and eight circumferential divisions. Dots show where the heat is applied for the entire section area that it is in. Thus, both locations in the left figure apply 1/2 of the total heat from the tool; for the right figure each of the outer dots apply 3/32 of the total heat and the inner dots apply 1/32 each. As a special case, a tool with only one division both radially and spatially would have a heat application site residing in the center of the tool.

In particular, a point .1875" deep was studied, as this point is about the depth of the thermocouple in the tool with respect to the plate. The effects of both radial and circumferential mesh density were studied with the two factors varying independent from each other. The error in temperature is shown in Figure 3-3. A single point has very high error as compared to distributed points. It is interesting that there is a clear increase in accuracy due to some grid refinements, and in other cases it appears to be wasted effort. In short there are many dominated meshes and a Pareto frontier appears for low error and low number of nodes. In general, meshing the tool radially gives much greater returns in accuracy than circumferential divisions. For most cases, increasing circumferential divisions had almost no effect on the model accuracy. However, at a certain point, in order to become more accurate, the circumferential mesh must be refined. This point of diminishing returns for radial divisions can be seen as the blue line curves

into the green line, and the green line into the red line. To improve graph clarity, the blue and green lines and their corresponding points were not plotted after they curved into the field of the other points.

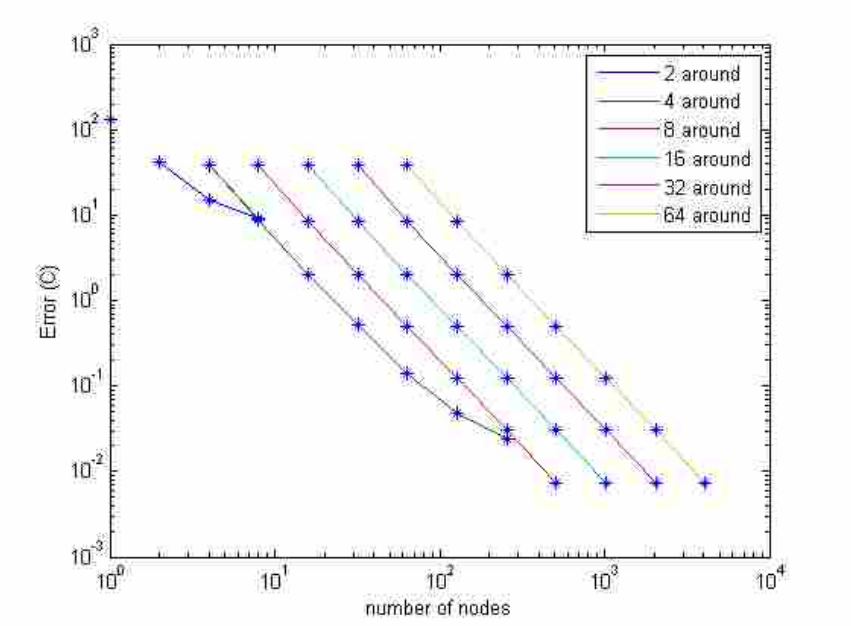



Figure 3-3: Error in temperature vs. number of nodes used to make up the tool as a heat source. Lines are drawn through individual configurations that are of the same number of circumferential mesh densities, but varying radial mesh densities. Except for where the points in the blue and green lines start to curve inwards, increasing the radial density lowers error more than increasing the circumferential density. Configurations closest to the lower-left corner lie along the Pareto front and are preferred designs.

In order to achieve a temperature accuracy on the order of 10°C a minimum of about 10 nodes are needed, and in order to achieve 0.1°C accuracy the number of required nodes rises to almost 100. However, in order be able to solve this problem quickly enough for real-time optimization, this was not able to be implemented, and a single point heat source was used. Nevertheless, because of the inherent parallelizable nature of the heat source method, a machine with many computational cores (such as offloading to a capable GPGPU or computing on a dual-

chip workstation/server) and parallel code could perhaps make this increase in computation time a nonissue. Since this level of parallelization was not available, for the purposes of this thesis, the heat source based methods were unfortunately not spatially grid independent.

Despite the single point heat source model not being grid independent, the affect of this on the performance of the controller may be minimal. By the methods in section 4.3, model parameters were determined by fitting the a grid independent Hybrid Heat Source model to actual weld data. The grid independent model had a very similar fit to the data as the grid dependent model (which was actually used in this thesis). Thus it is likely that using a grid independent controller would result in either small controller performance gains or possibly none at all.

### 3.3 Computational Time Reduction

In order to successfully implement the heat source and hybrid methods in real time, the time required to perform predictions had to be dramatically reduced. The optimization can easily call the objective function several hundred times, the objective function may have to call the predictive functions dozens of times, and the objective function must compute a temperature rise which may require the evaluation of tens to thousands of individual points in time. Thus a small increase or decrease in computational time for the predictive functions adds up very quickly. Several computational techniques were used to accomplish this. Details of these schemes are in Appendix A.1. A summary of what they are and their results are briefly presented below.

First, time was discretized in a nonuniform manner. For the heat source method, the majority of the rise in temperature is due to the actions within the most recent several seconds. The most recent several seconds also have the greatest nonuniformity, first derivative, and

second derivative as well. Thus, the recent past was discretized very finely, and the distant past was discretized coarsely. Below in Figure 3-4 is shown the error and calculation time for the two schemes at various meshes. The error can be reduced by almost two orders of magnitude while still taking the same amount of time to calculate by using a nonuniform time discretization, or alternatively, the calculation time is about an order of magnitude lower while still retaining the same accuracy.

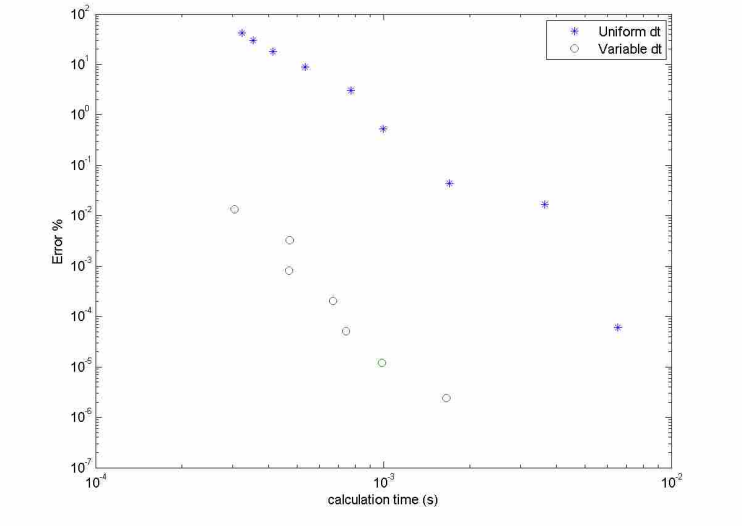



Figure 3-4: Absolute percent error vs. calculation time for uniform and variable time discretization schemes. The calculation time for each point varied slightly each time this study was redone, but the overall trends were about the same each time.

Even with using fewer discrete time segments, the time to calculate the objective function may still be unacceptably high. Prior to performing the above calculations, the data history must be interpolated/searched against to find data points that correlate these to the desired actual data point in time. The exact code that performs this function can drastically affect speed. For a very fine grid with and using a simple function that the author wrote, the calculations took a total time

of 36.22 seconds to simulate a full weld, of which 35.39 seconds were spent on the interpolation. By using Matlab's "interp1" function that uses a fast sorting algorithm, these times for the entirety of the weld were reduced to 1.28 seconds and .37 seconds, respectively. Matlab's function performed the same sorting task 95 times faster.

Another technique that sped up the process significantly was to precalculate the $\Delta\theta$ terms of the heat source model. Assuming a constant travel speed, it can be anticipated where the tool will be at future times. Thus, the rise in temperature due to events that have already taken place can be calculated to instances in the future since times and locations are known. Since Equation (2-17) and other heat source equations are linear with respect to the energy entering the system, $Q$, a unit $\Delta\theta$ can be precalculated from a unit $Q$ before the optimization. Thus, instead of calculating an expensive function the optimizer can simply linearly scale the precalculated results with a simple multiplication.

While not fully implemented here, parallelizing code can help tremendously. While some things must be done serially, many of the calculations can occur simultaneously. On a multi-core computer, this can significantly reduce the time to execute several sections of the code.

# 4 Dynamic Test Data and Parameter Estimation of Models

## 4.1 Physical and Thermal Setup for Experiments

Unless otherwise stated, all welds performed in this thesis have the following physical setup. The tool was a CS4 scroll style tool for .25" thick plate (see Appendix D.2). A thermocouple was placed in the tip of the tool approximately 2.5mm from the end of the tip of the tool. All welds were performed in 6.3 mm x 150 mm x 1.22 m (.25" x 6" x 4') Al 7075-T7 plate. The plates were not cut in half and then welded together; rather welds were run down the middle of the plate. Thus all welds are properly FSP, not FSW. This was done to reduce variability and to save time. To efficiently use material, three welds were run next to one another on the same plate. No area of the plate had more than a single pass, except where a previous weld segment was purposely welded over to create a disturbance. The plate was secured on top of 6.2 mm (.25") thick A-36 steel as a backing plate, which was on top of a steel anvil 51 mm (2") thick and .51 m (20") wide. See Appendix D.3 for a picture of the setup.

The following thermal conditions also apply to all welds in this thesis, unless stated otherwise. Room temperature was approximately 24°C. All welds were run with active water cooling for the tool holder. The chiller and water circulation were allowed to run for about 30 minutes or more before the start of any weld. This cooling chilled the tip of the tool (as measured by the thermocouple) to a steady state temperature of approximately 10°C. Due to the energy expended by the welding process, the tool and plate-anvil system expectedly increased in

temperature after multiple welds. When multiple welds were run within a short amount of time, the tool was allowed to cool within 10°C of its steady state low temperature, and the start and end locations of the new weld were not allowed to be higher than 10°C above room temperature.

After the plunge, the traverse speed was ramped from 0 m/s (0 ipm) up to 3.8 mm/s (9 ipm) over a distance of 5.1 cm (2 inches).

## 4.2 Dynamic Test Data for Parameter Estimation

In order to determine constants for the models, several welds were performed in 2.44 m (8') long material. After the plunge and a short velocity ramp, the heat input and traverse speed were abruptly and randomly changed to one of several possible predetermined levels and was held at that level for a random length of time within predetermined upper and lower bounds. This approach is similar to a Pseudo Random Binary Sequence (PRBS), but is different in a potentially important way. For these welds, there were three levels that the heat input and traverse speed could be set to, whereas in a PRBS there are only two levels. A PRBS uses the high and low binary value to obtain maximum excitation of the system, but because there are only two levels it cannot readily detect nonlinearity. While adding a middle level will reduce the magnitude of some of the excitations, it enables the detection of nonlinearity. One weld is shown below in Figure 4-1. Process data from all of the PRBS welds are in Appendix B.1.

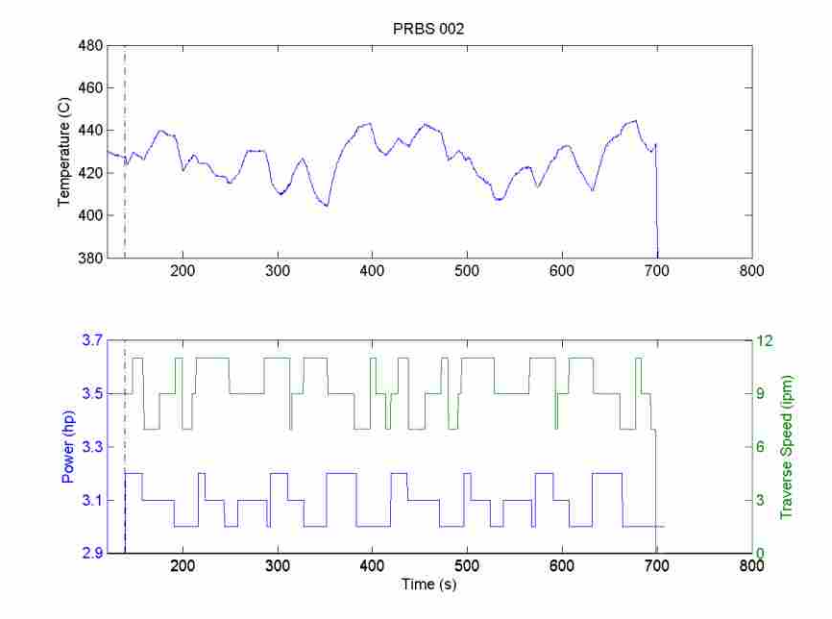



Figure 4-1: Temperature, power, and traverse speed for the PRBS 002 weld that was performed for parameter estimation. Each PRBS weld was 88 inches in total length. Welds 002 and 003 have segments that were repeated twice; weld 004's segment was repeated four times.

## 4.3 Determining Model Parameters

Constants for both the FOPDT and the Hybrid Heat Source Models were determined by optimization. The optimizer varied the constants in order to minimize the SSE of the predicted versus actual temperatures. The data sets used for this optimization are those presented in the preceding section. Additionally, constants were determined manually for the models due to the SSE objective function not being the best measure of optimality of parameters in the MPC controllers.

### 4.3.1 FOPDT Model Parameters

For the FOPDT model, the constants $c_1$, $c_2$, $c_3$, $c_4$, and $\tau$ in Equation (2-9) were determined. The values are presented below in Table 4-1 and the predictions for all three data sets are shown in Appendix B.7. All three data sets were weighted equally to their time; i.e. a

data set that took place for 12 seconds is weighted 20% more than a set that took place for 10 seconds. For this model, the prediction capabilities were started at only 2 seconds before the PRBS sequence started instead of from the beginning of the weld. This was done in order to avoid the futility of trying to fit models to heavy transients that immediately follow the plunge. Furthermore, trying to fit models to this portion of the data will likely introduce extra error into the model parameters. As is shown later in Section 5.3, the FOPDT model is of an incompatible form to be able to model the behavior of a weld before the quasi steady state.

Table 4-1: Parameters for the First-Order Model Determined by Optimization

| $c_1$ (°C/kW) | $c_2$ (°C·s /m) | $c_3$ (°C·s$^2$/m$^2$) | $c_4$ (°C) | $\tau$ (s) |
|---|---|---|---|---|
| 122.1 | -54.27 | 20.57 | 18.70 | 18.19 |

### 4.3.2 Hybrid Heat Source Model Parameters

The constants that were determined by optimization for the hybrid model are $z_{pt}$, $h_0$, and $p_{mult}$, and are shown below in Table 4-2. In addition, some constants such as $h_{collar}$ were determined manually. This was done because experience had shown that small to medium changes in the values of these constants did not have large impacts on the predictions of the model. Furthermore, several other constants had a similar effect on the model and optimizing their values was sufficient. Other constants which are necessary in order to implement the model are shown in Table 4-3. Other parameters which are not constants per se (such as the discretization of time and the tool geometry) are specified in Appendix D. The predictions for all three data sets are shown in Appendix B.9. Unlike the FOPDT model predictions which were only started after the PRBS commenced, the hybrid model predictions included data from the

very beginning of the weld. The error of the plunge was not included in the objective function, but error immediately after the weld started to traverse was included by the objective function.

Table 4-2: Parameters for the Hybrid Heat Source Model Determined by Optimization

| $z_{pt}$ (mm) | $h_0$ (W/m$^2$ K) | $p_{mult}$ (unitless) |
|---|---|---|
| 1.09 | 4613 | .6011 |

Table 4-3: Other Values and Parameters Needed for the Hybrid Heat Source Model

| $h_{collar}$ (W/m$^2$ K) | $k_{plate}$ (W/m K) | $\rho_{plate}$ (Kg/m$^3$) | $c_{plate}$ (J/Kg K) |
|---|---|---|---|
| 200 | 140 | 2800 | 1100 |

### 4.3.3 Manually Determining Model Parameters

Parameters were also manually determined for the first-order model. This is because an objective function equal to the SSE between the actual temperature and model's predicted temperature might not actually result in the best model parameters for optimal control. This can be demonstrated with a representative graphic, Figure 4-2.

Figure 4-2 shows a fictional actual temperature (blue line) and two models' predicted temperatures based upon parameters for good offset and good dynamics (green and red lines, respectively). Of the two proposed models, the good offset will have a much lower SSE than the red, and consequently would be picked out of the two if an optimizer's objective function was the SSE between the model and actual temperature. In terms of control, the good dynamics model would be a much better candidate for MPC control than the good offset model. This is because the good dynamics model has very few disadvantages and the good offset model has a very large disadvantage.

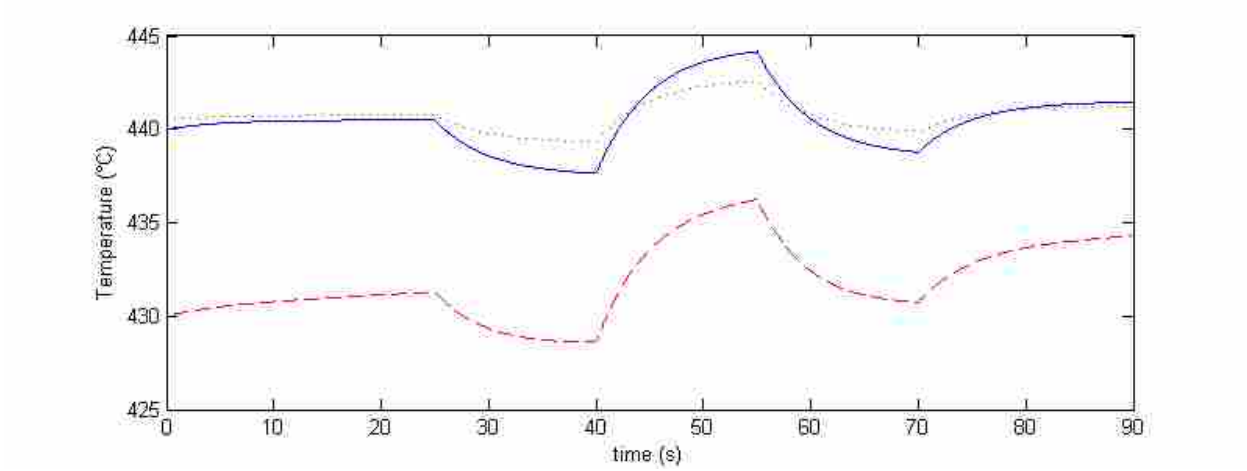



Figure 4-2: Representative plot of temperature over time and two proposed models to predict the temperature. The solid blue line is the actual temperature, and the dotted green and dashed red lines are the results of two proposed models for this process: a good offset model and good dynamics model, respectively. The models and input data for this plot are fictional and based upon a strictly first-order model and five different input levels.

By using model biasing, it is very easy for the MPC controller to bring the model temperature up or down by a set amount, and the offset of the good dynamics model quickly becomes a nonissue during control. This is especially true if the controller starts biasing the model before the MPC controller actually takes control from a preceding controller (as is the case for the welds in this thesis). Even if the model drifts slowly over time (as the red model does), it is easy to compensate for this gradual change via biasing.

The good offset model captures the dynamics of the process much worse than the good dynamics model does. Because the good offset model greatly under predicts the results of a given input, in order to achieve a given change in temperature to compensate for either modeled or unmodeled disturbances, an MPC controller using the green model will predict that a much larger input is needed to overcome a disturbance. With a larger input than is needed, overshoot will likely happen. On the other hand, if a model were to greatly overpredict the effects of a given input, the control would be sluggish since the model would predict that a small is needed

to correct the temperature, when in reality a much larger action is needed to correct the temperature.

Ideally, a model exactly predicts the output of a model based upon inputs. However, this almost never happens. First, for this to be correct, the form of the model must be exactly correct, i.e. a strictly first-order model can never capture all of the dynamics of a FOPDT system with a nonzero time delay because the time delay term is not in a simple first-order model. Second, any variance in the system will lead to incorrect predictions. As can be seen by the temperature output of each of the PRBS welds (see Appendix B.1), the inputs repeat throughout the weld, but the temperature is not exactly the same over each repeat interval. Also, sometimes large temperature variations can occur. For instance, the PRBS 003 and PRBS 003 v2 welds have practically identical inputs, but the temperature of PRBS 003 is approximately 40°C higher than PRBS 003 v2, despite taking precautions to ensure that the inputs and setup for the welds were as identical as possible.

An optimizer cannot perfectly fit two different outputs with nearly identically inputs. In the case of identical inputs but two outputs that are offset by a constant amount, in order to reduce the SSE, the optimizer will settle on a model which does not predict extremely well for either scenario but which has a somewhat low SEE for both. To do this, the optimizer will focus on reducing offset more than matching dynamics because the offset will have a greater impact on the SSE than the dynamics will. However, as discussed above, those are exactly the opposite priorities of what is needed for a model that will be used in an MPC context. A different kind of objective function would likely result in better model parameters. However, since manually fitting the model parameters resulted in sufficient controller performance, an in-depth study and analysis of optimal objective functions was not undertaken.

Finding suitable manual parameters does require manually increasing and decreasing model parameter values until the temperature prediction is suitable, as described above. However, additional knowledge and data can greatly help this process.

Knowledge of the model can be effectively used. For instance, in the case of the FOPDT model, the value of $\tau$ ought to be changed to fix problems with the response to the step changes being either too linear or too curved. If the approximate change in steady state temperature from a step change in power is too high, then the gain on the power of the model ($c_1$) should be decreased. Lastly, any parameter that affects the offset ($c_4$) is changed to minimize the offset of the model as much as is possible. The Hybrid Heat Source model's parameters are much more interrelated, but the same strategy can be taken for it as well. Generally speaking the $z_{pt}$ and $h_0$ variables were adjusted to change how quickly the system responded and the relative size of input changes to output changes, and $p_{mult}$ was then adjusted to eliminate as much of offset error as possible. However, since all of these parameters are much more interconnected than FOPDT model parameters, adjusting them is more difficult.

Previous weld data that used the same model can be used beneficially in determining new parameters for that model. As an example, the FOPDT 001 weld had poor performance when the traverse speed was changed. This suggests that the parameters that related the traverse speed to the temperature ($c_2$ and $c_3$) are not correct. Those parameters likely need to be changed to fix that problem. Additionally, the data from that mediocre weld can now also be estimated against, and new model parameters should produce changes in temperature similar to what actually happened during the weld.

Parameters for the FOPDT model were determined manually by fitting the predicted dynamics to the actual dynamics of the system as much as possible. The temperature vs. time

predictions for the optimizer determined and manually determined models are shown in Figure 4-3 and Figure 4-4, and the predictions for the rest of the PRBS welds are shown in Appendix B.8. The manually determined parameters are shown below in Table 4-4. These parameters are worse at modeling the offset of the PRBS welds, but better at modeling the dynamics.

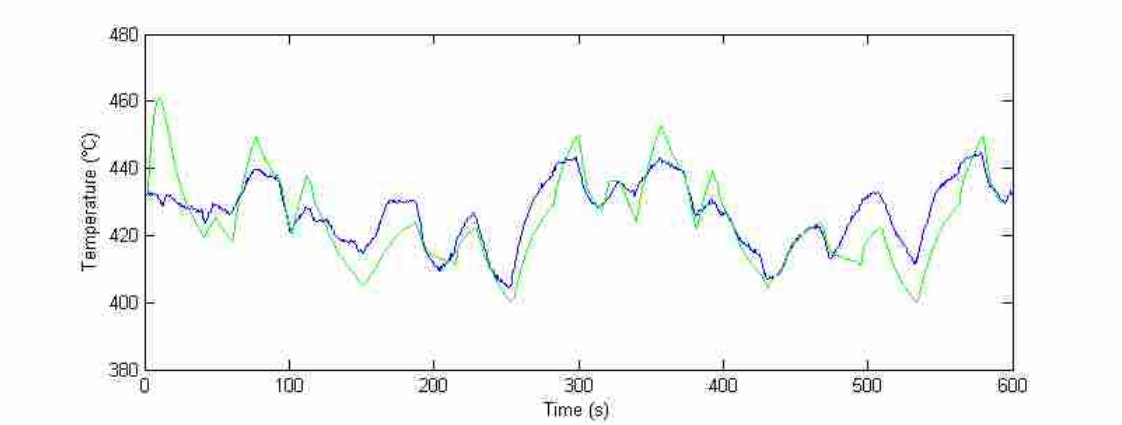



Figure 4-3: Temperature vs. time data for weld PRBS 002 (blue) with the predicted temperature (green) based upon the FOPDT model with optimizer determined parameters.

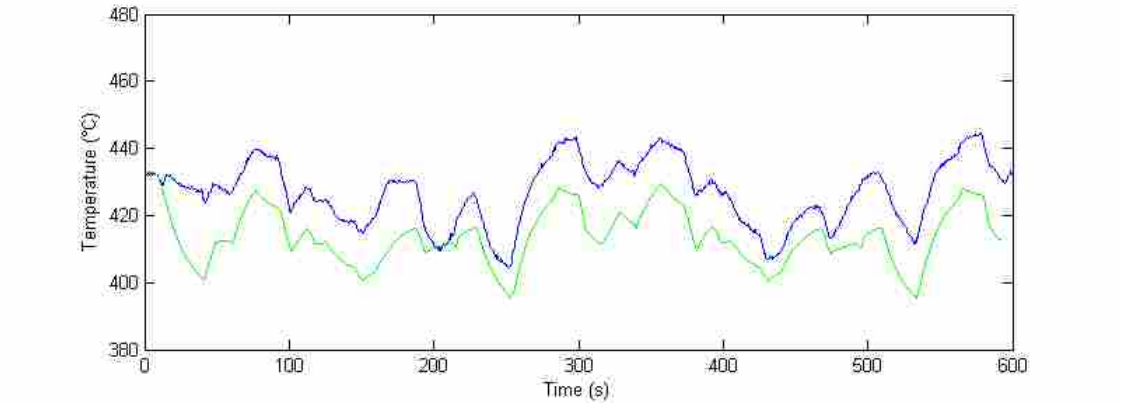



Figure 4-4: Temperature vs. time data for weld PRBS 002 (blue) with the predicted temperature (green) based upon the FOPDT model with manually determined parameters.

Table 4-4: Manually Determined Parameters for the First-Order Model

| $c_1$ (°C/kW) | $c_2$ (°C·s /m) | $c_3$ (°C·s$^2$/m$^2$) | $c_4$ (°C) | $\tau$ (s) |
|---|---|---|---|---|
| 158.9 | -7.040 | -.0967 | 4.918 | 16.39 |

Parameters were also manually determined for the Hybrid Heat Source model, due to the same reasons and in the same manner as the FOPDT model. The temperature vs. time predictions for the optimizer determined and manually determined parameters for PRBS 002 are shown in Figure 4-5 and Figure 4-6, and the predictions for the rest of the PRBS welds are shown in Appendix B.10. The manually determined parameters are shown in Table 4-5.

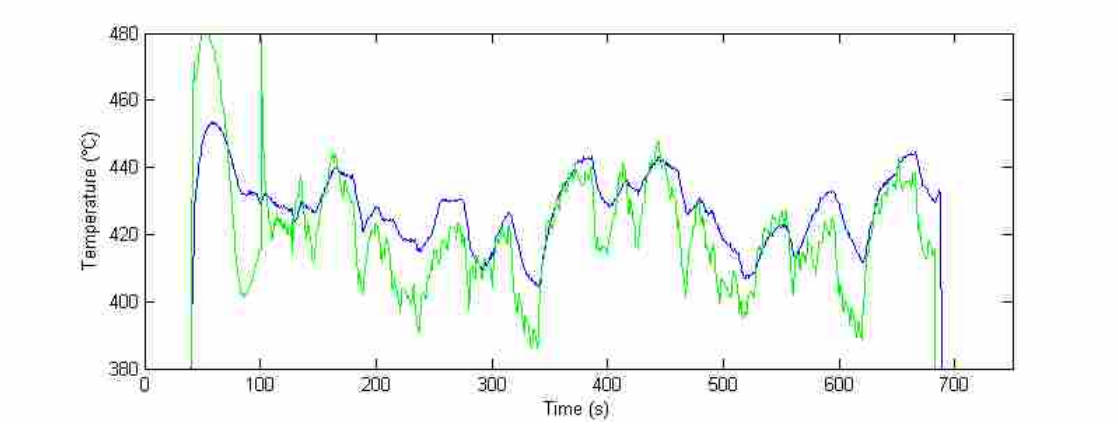



Figure 4-5: Temperature vs. time data for weld PRBS 002 (blue) with the predicted temperature based upon the Hybrid Heat Source model with optimizer determined parameters (green).

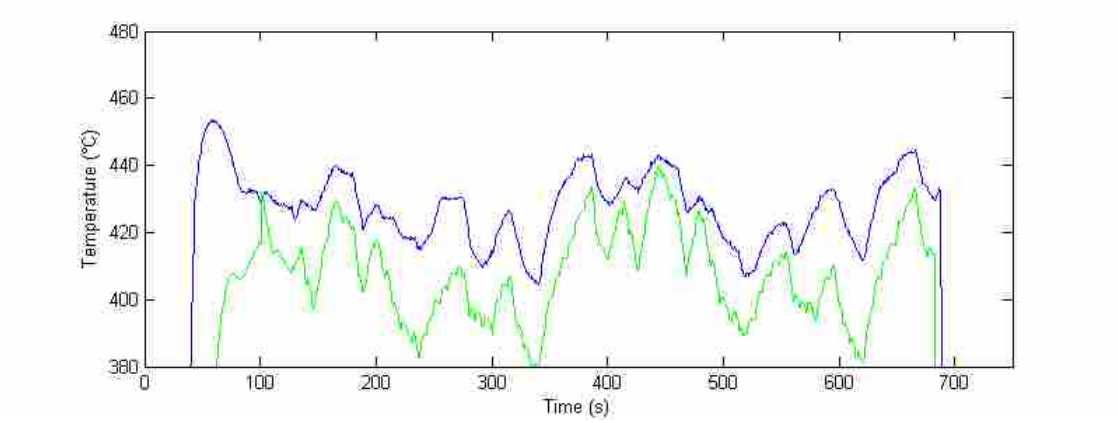



Figure 4-6: Temperature vs. time data for weld PRBS 002 (blue) with the predicted temperature based upon the Hybrid Heat Source model with manually determined parameters (green).

Table 4-5: Manually Determined Parameters for the Hybrid Heat Source Model

| $z_{pt}$ (inches) | $h_0$ (W/m$^2$ K) | $p_{mult}$ (unitless) |
|---|---|---|
| 4.57 | 600 | 1.4 |

Further, parameters were also manually determined for the Hybrid Heat Source model for the transient portion of the weld. The parameters are shown below in Table 4-6.

Table 4-6: Manually Determined Parameters for the Hybrid Heat Source Model for the Transient Portion of a Weld

| $z_{pt}$ (inches) | $h_0$ (W/m$^2$ K) | $p_{mult}$ (unitless) | $k_{plate}$ (W/m K) |
|---|---|---|---|
| 4.57 | 4000 | 1.4 | 190 |

## 4.4 Determining Gains for PID controllers

Two separate sets of gains were determined for PID controllers. This was done so that the MPC controllers can be compared to the best current FSW controllers. A PID controller is a simple closed-loop controller which has been used for many years in many different fields. The CV commanded by the PID controller at any time, *u(t)*, is:

$$u(t) = u_{start} + k_p\, e(t) + k_i \int e(t)\, dt + k_p \frac{d(e(t))}{dt} \tag{4-1}$$

where $u_{start}$ is the value of the MV immediately before the PID controller takes control, *e(t)* is the error, or the setpoint minus the CV, and the *k* terms are the PID gains. In practice, the error's derivative term is usually replaced by the negative of CV's derivative. When a setpoint change happens, this causes a discontinuity in the error and, therefore, a very large derivative at that point. Making the above mentioned change eliminates this spike in the derivative, and is mathematically the exact same for setpoints which are constant. Furthermore, depending on how noisy the CV is, oftentimes the derivative term is averaged to help with smoothness and control.

There is an infinite set of possible PID gains. The two sets that are used in this thesis are a 0% overshoot servo set proposed by Chien (O'Dwyer 2006), and a regulator gain set purposed by Murrill (O'Dwyer 2006). These were successfully used in FSW in the past (Marshall 2013).

The 0% overshoot servo gains rules are:

$$k_p = \frac{.6\tau}{K_m \theta} \tag{4-2}$$

$$k_i = \frac{k_p}{\tau} \tag{4-3}$$

$$k_d = .5 k_p \theta \tag{4-4}$$

The regulator gains rules are:

$$k_p = \frac{1.357}{K_m} \left(\frac{\tau}{\theta}\right)^{.947} \tag{4-5}$$

$$k_i = \frac{k_p}{\frac{\tau}{.842}\left(\frac{\theta}{\tau}\right)^{.738}} \tag{4-6}$$

$$k_d = k_p \tau \left(\frac{\theta}{\tau}\right)^{.995} \tag{4-7}$$

where $\theta$ is the time constant of the system, $\tau$ is the time delay of the system, and $K_m$ is the model gain of the system. These three parameters can be readily determined using the relay feedback method, as is detailed in the work by Marshall. Other system identification methods could be used to identify the system parameters as well.

A relay feedback test was performed at a nominal temperature of 440°C. The actual temperature when the relay test took control was 439.7°C with a power of 2.35kW (3.15 hp). The results are shown in Figure 4-7.

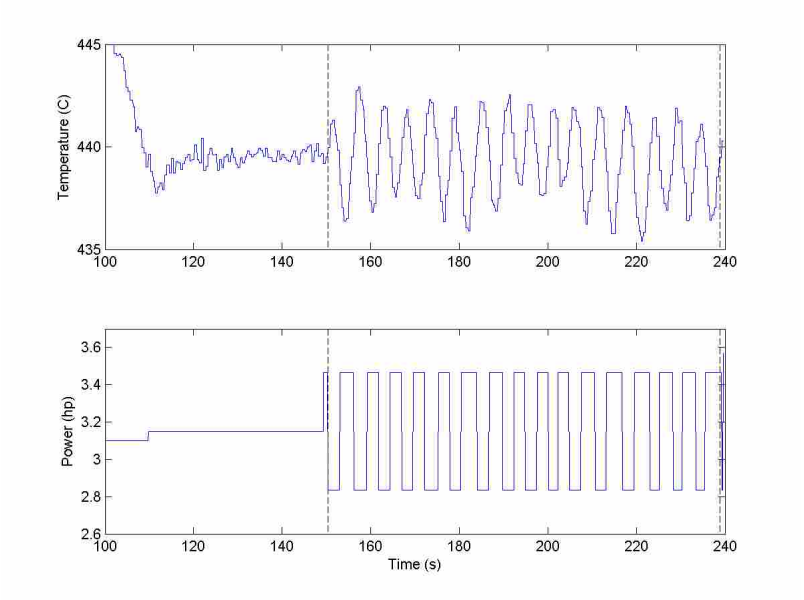



Figure 4-7: Relay feedback test results for a nominally 440°C weld

The first and last steps in power were excluded from analysis because the test was not yet at steady state or the step could not finish before the weld began to extract. From this test, the following system parameters were determined as shown in Table 4-7. A further description of these parameters and how to determine them can be found in Marshall's work and in many process controls texts.

Table 4-7: System Parameters as Determined by the Relay Test Method

| $K_u$ (°C/kW) | $P_u$ (s) | $\omega_u$ (Hz) | $K_m$ (°C/kW) | $\tau$ (s) | $\theta$ (s) |
|---|---|---|---|---|---|
| .2242 | 5.521 | 1.145 | 185.6 | 20.18 | 1.38 |

The last three of these system parameters were used to determine PID gains according to the two previously stated tuning rules and are shown in Table 4-8.

Table 4-8: PID Gains for the Servo and Regulator Tuning Rules at 440°C

| | $k_p$ (hp/°C) | $k_i$ (hp/s°C) | $k_d$ (hp·s/°C) |
|---|---|---|---|
| Servo | .0478 | .0023 | .0330 |
| Regulator | .0929 | .0283 | .0500 |

# 5 Evaluation of Controller Performance

## 5.1 Experimental Setup

Unless otherwise stated, the same physical and thermal conditions as stated in Section 4.1 apply to these welds as well. The same model parameters and gains as were determined in Sections 4.3 and 4.4 are used here unless otherwise stated.

All welds were plunged to a final depth 6.04mm (.238 inches) at a rate of 0.15 mm/s (.35 in/min). Following this, the traverse speed was increased from 0 to 3.8 mm/s (0 to 9 ipm) over a distance of 51 mm (2 inches). After these transient parts of the weld, traverse speed and depth were kept constant unless otherwise specified. The controller taking control of a weld is shown by a black vertical dashed line in figures in this chapter.

For welds at quasi steady state conditions, two different disturbance patterns were used. The first round of testing had both modeled and unmodeled disturbances which are: initially coming to temperature, two temperature setpoint changes, three traverse speed changes, and welding over a preexisting weld. Exact details for these disturbances can be found in Appendix C.1. A second round of testing only incorporated wholly unmodeled disturbances and included initially coming to temperature, welding over a weld, and a vertical position change for the tool. With fewer steps, the longer term behavior of the controller reacting to these disturbances was able to be observed. For the MPC controllers, power rate change limits of $dP/dt = \pm 370$ W/s ($\pm 0.5$ hp/s) were set, and an $\alpha = 0.5$ was used for all biasing.

For the transient welds, the first round of welds were only run through the end of the traverse speed ramp, which is the same as described above. However during this transient period, the controllers were actively controlling power. The second round of transient welds also had a 230 mm (9 inch) or 60 second steady segment after the traverse speed ramp. The MPC *dP/dt* and $\alpha$ terms were doubled. No external disturbances were added for the transient welds.

Full data for all welds can be found in Appendix B. Additional analysis of all the welds which were performed and different types of changes which were made are found in Appendix C, whereas only the best version of each controller is shown in this section. The exact parameters of all welds and the times, locations, and types of disturbances are also detailed in Appendix C.

## 5.2 Definition of Metrics

An evaluation of performance is only as valid as the metrics used to evaluate it. If the metric in question was whether a controller could hold temperature within 100°C or not, then all controllers in this thesis would appear equally optimal. The metrics that are used in this thesis to judge controller performance are as follows:

- *Settling Time:* How long it took for the weld segment to maintain temperature within a temperature tolerance, which is defined as within ±2°C of the setpoint for quasi steady state welds (see Figure 5-1). During transient testing, this tolerance was changed to ±5°C because ±2°C was extremely hard to achieve. Marshall used the same ±2°C temperature tolerance in his work, and his data showed that a good controller can easily hold temperature with 2°C of the setpoint even after a major disturbance. A tighter tolerance such 1°C may sporadically artificially increase the settling time for a controller which is

in fact exceptional. For instance, if a controller encounters a small disturbance coupled with the noise of the temperature reading, this can easily briefly push the temperature outside of a 1°C tolerance window. When a segment did not settle within the time of a segment, the estimated settling time was either (1) extrapolated based upon the exponential decay rate of oscillations of that segment, or (2) an extrapolated time of intercept to the setpoint if there were no oscillations present. For the initial round of testing, the segments were nominally 30-40 seconds long, and so any settling time that is considerably over the segment time (about 10s) is a rough estimate and may be somewhat inaccurate.

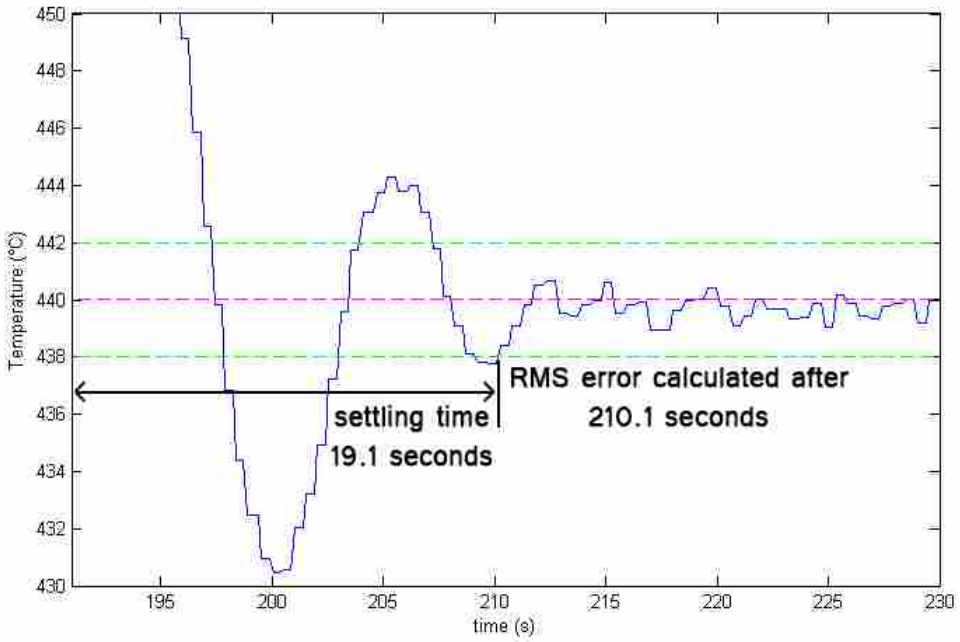



Figure 5-1: Schematic showing settling time and RMS error. Data from the FOPDT 001 weld. A temperature setpoint change from 460°C to 440°C happened at 191s. The setpoint is shown by the dashed magenta line. The setpoint tolerance is shown by the two dashed green lines.

- *RMS Error*: The average root mean square error after the temperature is within the settling temperature tolerance (see Figure 5-1). This is a metric of how well the controller can hold temperature. If a weld or segment of a weld did not settle within the allotted time, then a value of 4.8°C was used. This default number was used instead of NaN for plotting and comparison purposes. This value of 4.8°C was used because it is twice of 2.4°C, which is the maximum RMS value calculated for any weld which did settle.

- *Overshoot*: How much the controller overshoots the target setpoint (see Figure 5-2 ). In the case of changes in the traverse speed, since the controllers were normally already at or about at temperature, it is how far the temperature was pulled away from the setpoint. In the case of welding over welds, these disturbances caused the temperature to quickly drop, and the overshoot is how much it overshot the setpoint while attempting to bring the temperature up to the setpoint – not how much the temperature initially dropped due to the disturbance.

- *Oscillations*: How many oscillations a given controller had in a weld segment (see Figure 5-2). This is *not* the magnitude or severity of the oscillations. In this analysis, a half oscillation was classified as the temperature passing entirely through the settling temperature band/tolerance about the temperature setpoint. Therefore, oscillations that lie completely within the tolerance band are not considered oscillations for this criteria.

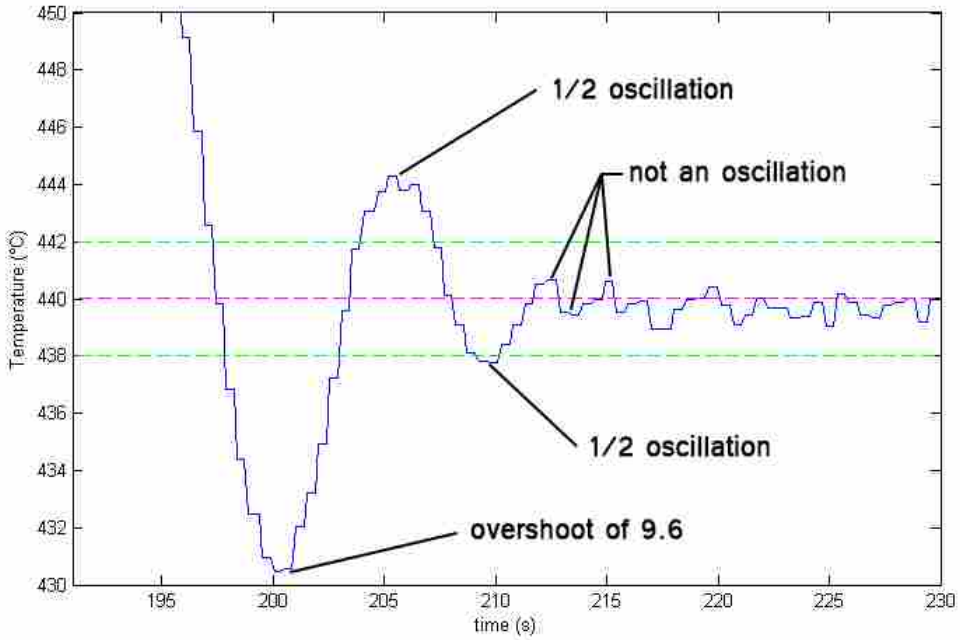



Figure 5-2: Schematic showing overshoot and oscillations error. Data from the FOPDT 001 weld. A temperature setpoint change from 460°C to 440°C happened at 191s. The setpoint is shown by the dashed magenta line. The setpoint tolerance is shown by the two dashed green lines.

## 5.3 FOPDT MPC Controllers

The FOPDT controller with parameters determined by optimization was able to overcome both unmodeled disturbances and modeled disturbances within about 10-50 seconds depending upon the type of disturbance, and usually with minor oscillatory action preceding the steady state control. The results for this controller are shown in Figure 5-3. In this and other quasi steady state welds shown in this section, the vertical dashed black line indicates the start of the temperature controller. The three traverse speed changes are shown by the three magenta diamond symbols. The last two unmodeled disturbances happen at 336 and 365 seconds and are immediately followed by a sharp drop in temperature.

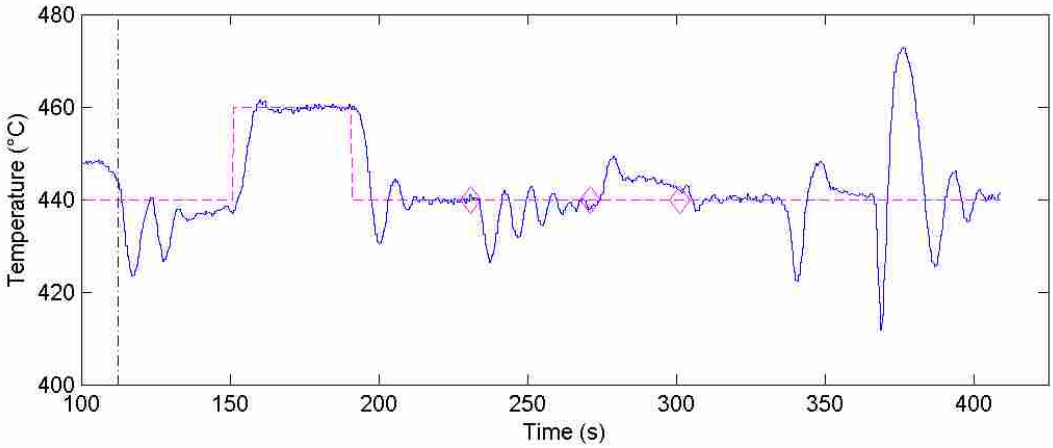



Figure 5-3: Temperature of the FOPDT 001 weld. The controller eventually reached or approached a steady condition after each disturbance, but oscillations were often observed.

Using manual parameters for the FOPDT controller resulted in significant benefits to controller performance. The reasons for using manual parameters and methods for determining them is covered in Section 4.3.3. The results for this controller are shown in Figure 5-4. The controller handled modeled disturbances (which were changes in the temperature setpoint and traverse speed) extremely well. For example, at about 271 seconds, there was a traverse speed jump from 2.54 mm/s to 5.1 mm/s (6 ipm to 12 ipm). The controller correctly anticipated the corrective action to take and changed the power from an average of about 2.4 kW to 2.75 kW (3.2 hp to 3.7 hp) within 2 seconds. So, instead of a 40°C drop in temperature occurring (the approximate temperature change without controller action), the temperature only momentarily dropped to a maximum of 3°C from the setpoint and quickly recovered to the setpoint. Overall, this controller exhibited exceptional control with low overshoot, oscillations, steady state RMS error, and settling time.

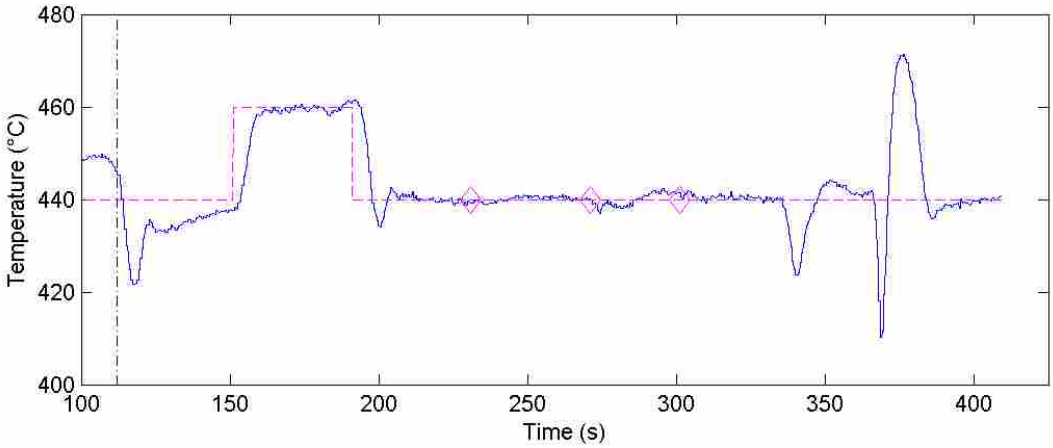



Figure 5-4: Temperature of the FOPDT 002 weld at quasi steady state conditions. The controller had very good control, especially for the modeled disturbances/state changes of a change in traverse speed.

The FOPDT controller did not do nearly as well during the transient portion of the weld. It was never able to settle during the start-up transient. See Figure 5-5 for the performance of the FOPDT controller. Changing the parameters did not help, and actually hurt performance. This is likely because the model is completely incapable of capturing some of the trends in the transient portion of a weld. For a constant spindle speed weld, immediately after the plunge the power drops and the traverse speed increases while the temperature stays level or even slightly increases. However, the FOPDT model predicts the exact opposite: an increase in traverse speed and decrease in power leads to a dramatic drop in temperature. Therefore, based upon observed performance and theory the FOPDT MPC controller is ill-suited to control immediately after the plunge.

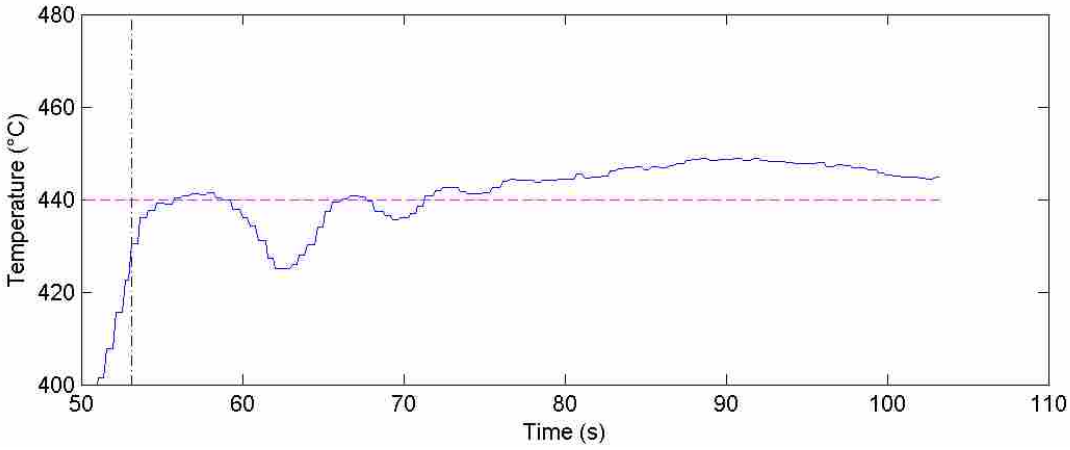



Figure 5-5: Temperature of the FOPDT 101 weld during transient conditions. The controller was not effective at controlling the temperature to the setpoint during transient conditions.

## 5.4 Hybrid Heat Source MPC Controllers

Similar to the FOPDT controller, the Hybrid Heat Source controller greatly benefited from using manually determined parameters over the optimizer determined parameters. Additionally, adding other things to the controller such as a minor time delay and a MV move penalty aided controller performance. The stock controller and final controller are shown in Figure 5-6 and Figure 5-7

As can be seen, a great deal of performance gains were achieved through the above-mentioned means. In some cases, such as the two temperature setpoint changes, the performance is very similar to the FOPDT MPC controller. However, the best Hybrid Heat Source controller still did worse for the other modeled and unmodeled disturbances than did the best FOPDT controller.

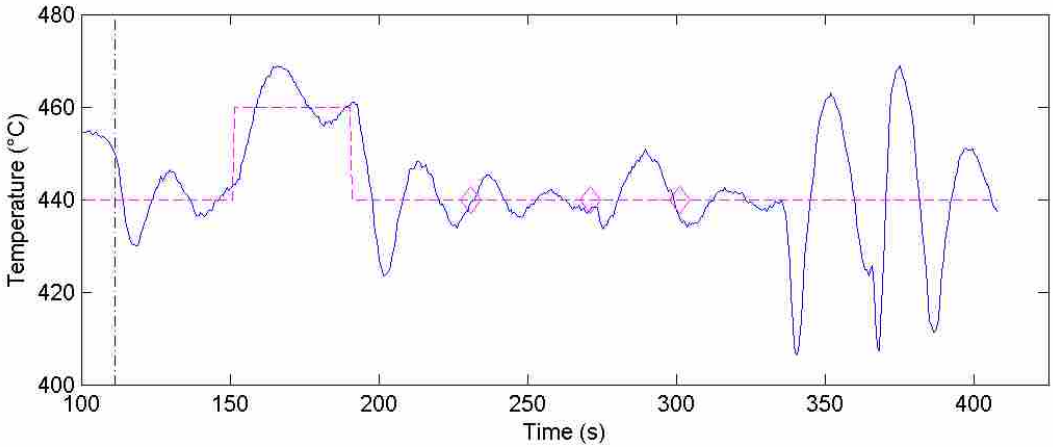



Figure 5-6: Temperature of the HS 001 weld at quasi steady state conditions. The controller had very poor temperature control performance

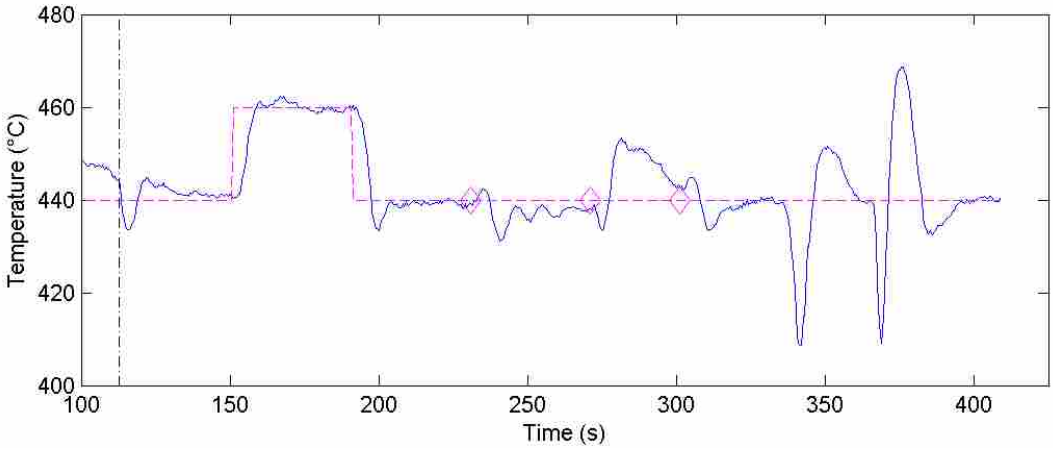



Figure 5-7: Temperature of the HS 005v2 weld at quasi steady state conditions. This controller had much better control than the unmodified controller, but was still not as good as the FOPDT controller.

Using the previously determined manual parameters, control immediately after the plunge was also tried. The control was poor with these parameters (see Figure 5-8). However,

with appropriate model parameter changes the weld performance was very good during the post-plunge transient (see Figure 5-9).

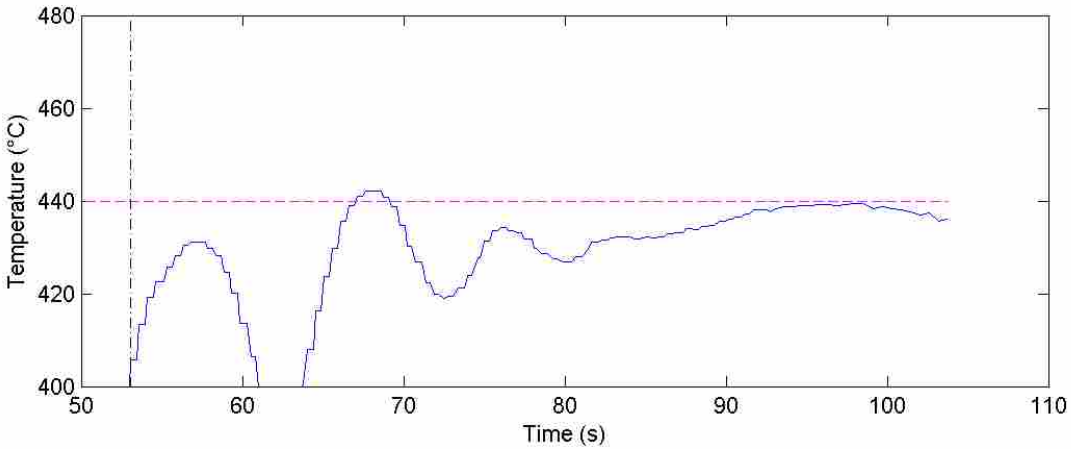



Figure 5-8: Temperature of the HS 101 weld during transient conditions. The controller was not very effective at controlling the temperature to the setpoint in transient conditions.

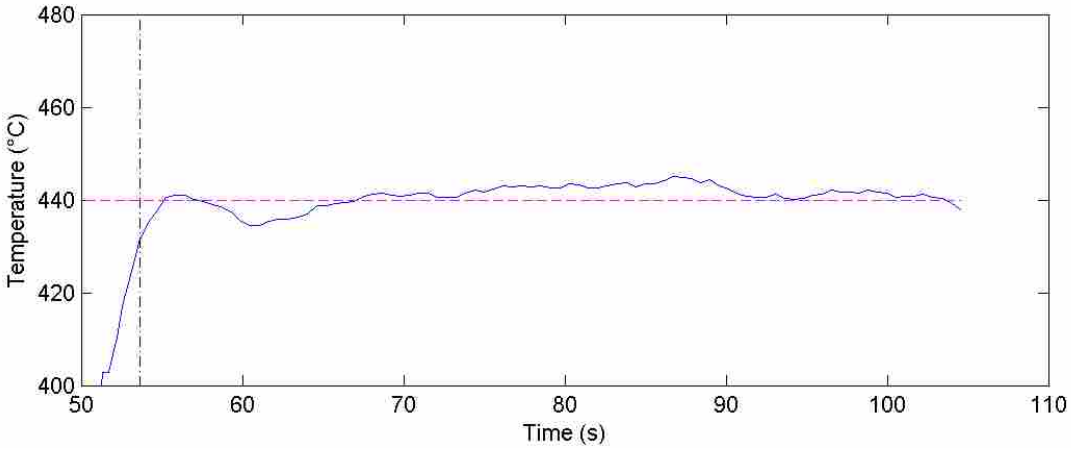



Figure 5-9: Temperature of the HS 101v2 weld during transient conditions. With the new model, this controlled temperature much better than any of the other MPC controllers.

One of the model parameters that was not changed before, but was changed for the HS 101v2 and HS 102 welds, was the assumed thermal conductivity of the plate. Previously it was

assumed to be 140 W/m·K, but for these welds a value of 180 W/m·K was used. This was done for two reasons. First, this change led to a much better predicting model when analyzing previous welds. Second, while some sources report relatively constant thermal conductivities for 2000 and 7000 series aluminum over temperatures (Taylor 1998), others report values that are highly temperature dependent (Incropera 2007). Also, even at the same temperature, reported values vary widely. The value for 7075 at room temperature has been reported from about 130 - 155 W/m·K by material suppliers (Matweb 2015a; Matweb 2015b). However, at elevated temperatures, values have been reported in the 160-180 W/m·K range (Taylor 1998). The heat source method assumes and needs constant thermal properties, which of course is not true when large temperature variants are present. At high traverse speeds, only a small amount of material is at highly elevated temperatures, and so a thermal conductivity value closer to room temperature will be most representative of the system. However, during and immediately after the plunge, heat is input at roughly the same location and (assuming a constant tool temperature) the temperatures in the plate at a given distance from the tool will be higher than at the same distance from the tool when at a fast traverse speed. Because more of the material surrounding the tool is at a higher temperature, using material properties that are for material at this state should result in better predictions.

## 5.5 PID Controllers

The performance of two PID controllers, one with regulator gains and the other with servo gains, was also determined by having both controllers try to control temperature through the exact same setpoint changes and disturbances as the Hybrid Heat Source and FOPDT MPC controllers. The results from these welds is shown in Figure 5-10 and Figure 5-11.

As can be seen, both have good control but different strengths. The regulator PID controller was very aggressive. It controlled to the new state quickly and tended to be very true to the setpoint, but often had high overshoot, many large oscillations while settling to the setpoint, and many small and persistent oscillations at the setpoint. On the other hand, the servo PID controller did not get to the setpoint as fast and did not hold to it as aggressively, but had very smooth action and no overshoots or oscillations.

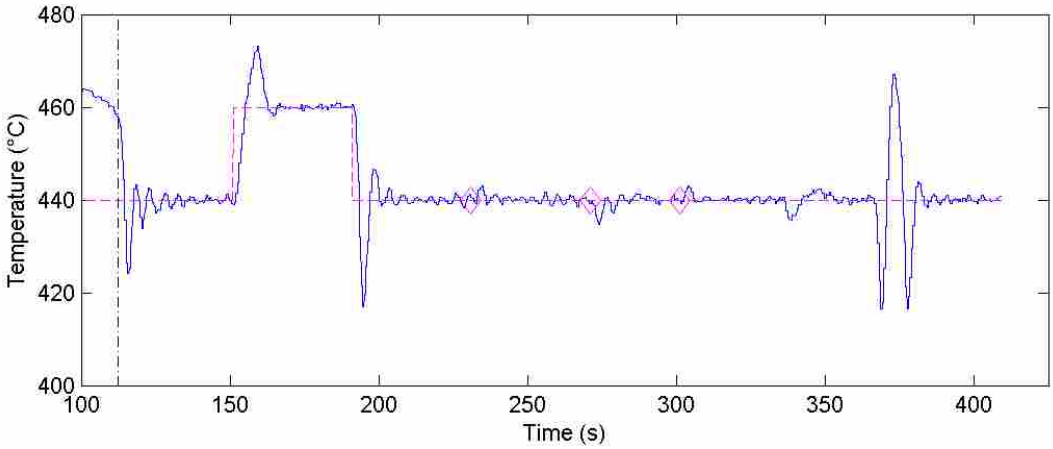



Figure 5-10: Temperature of the PID reg 001 weld at quasi steady state conditions. The controller was very aggressive and oscillated a lot, but reached the setpoint very quickly. Magenta diamonds have been inserted in order to identify the locations of the traverse speed changes.

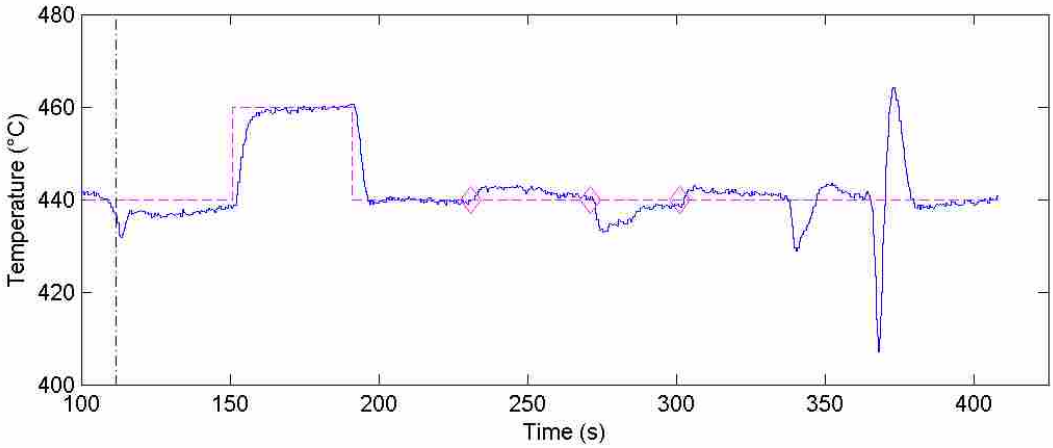



Figure 5-11: Temperature of the PID servo 001 weld at quasi steady state conditions. This controller was very smooth and had no overshoot, but did not hold to the setpoint as well as the PID regulator gains controller did.

The PID regulator controller performed acceptably well, as can be seen in Figure 5-12. During the transient after the plunge, the servo PID controller performed poorly, as shown in Figure 5-13. The P and I terms were not aggressive enough to counteract the continuously changing thermal transient. The entirety of the weld is much closer to the setpoint than the servo controller. There are many oscillations; however, the controller reaches steady state by the end of the transient period.

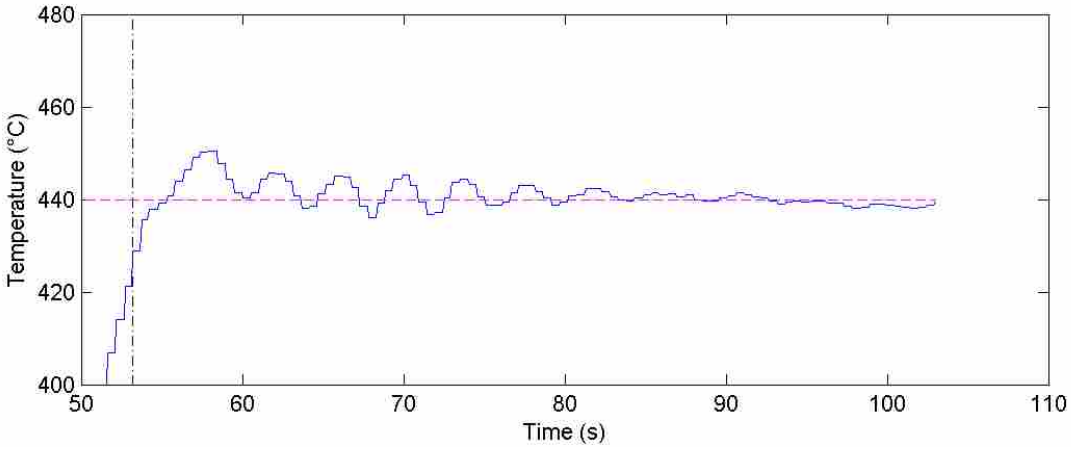



Figure 5-12: Temperature of the PID reg 101 weld during transient conditions. The controller was still oscillatory in nature and did not close in on the setpoint as quickly as it did for quasi steady state conditions, but reached the setpoint after about 30 seconds and then held relatively true to it.

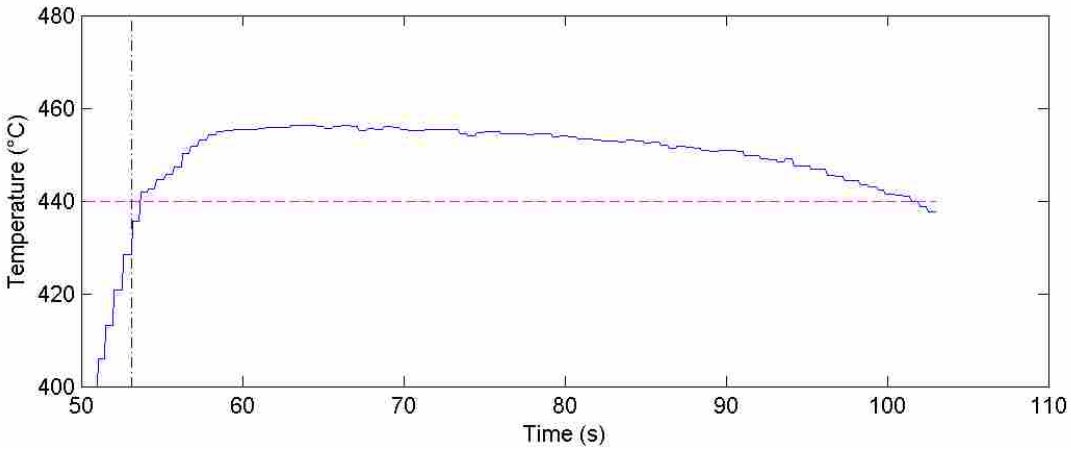



Figure 5-13: Temperature of the PID servo 101 weld during transient conditions. This controller did not have large enough P and/or I terms, and therefore was not able to reach the setpoint until the very end of the weld.

It is interesting that the PID regulator controller had nearly sustained oscillations during the transient portion of the weld (see Figure 5-12), but had quickly decaying oscillations in previous welds (see Figure 5-10). Regardless of transients or disturbances, it is believed that this

is due to the system dynamics changing during the traverse speed ramp and regulator gains which are tuned too aggressively for the system at the beginning of the plunge. When Marshall applied PID regulator gains determined from a high time constant system (475°C) to a medium time constant system (450°C), he observed sustained oscillations, and when applying them to a low time constant system (425°C), he observed increasing oscillations (Marshall 2013, 55-56).

While the time constant of the FSW system has not been explicitly studied with respect to traverse speed, there is good reason to assume that it should increase with traverse speed. Looking at the system through the lens of the FOPDT model, the time constant of the system is equivalent to the mass of the stir zone times its thermal capacitance – see Equations (2-1) and (2-2). At low traverse speeds, little metal is flowing through the stir zone. It should thus be relatively easy to change the temperature of the metal, which would correlate to a low time constant of the system. Conversely, a high traverse speed would have a high time constant.

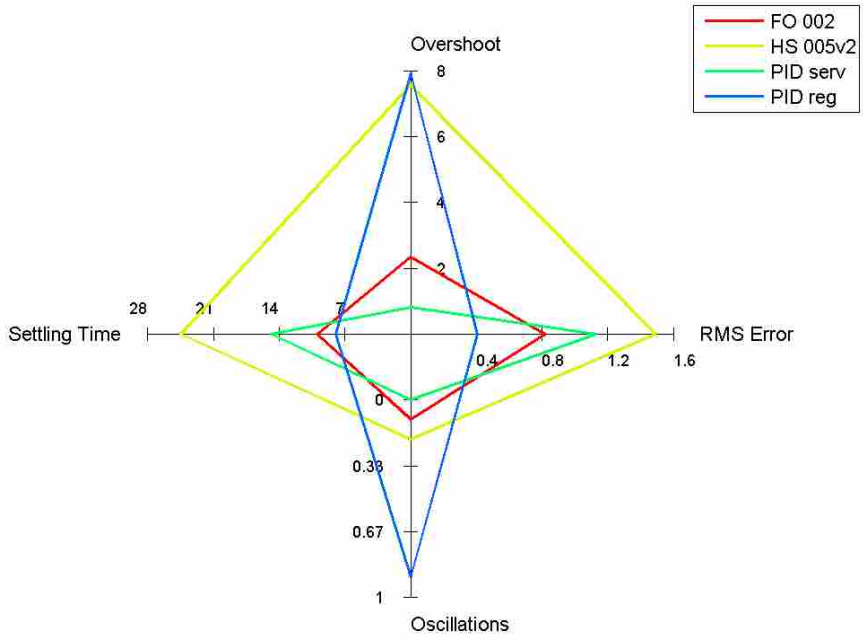
If this is in fact true, then using system parameters determined at 3.81 mm/s (9 ipm) for a weld at 0.4 – 1.3 mm/s (1-3 ipm) would likely result in an overly aggressive system with sustained or unbounded oscillations. As the traverse speed of the weld increases during the traverse speed ramp, the FSW system would have a longer and longer time constant and oscillations should decay. This matches what was seen during the transient welds using the aggressive regulator gains.

## 5.6 Comparison of Controller Types

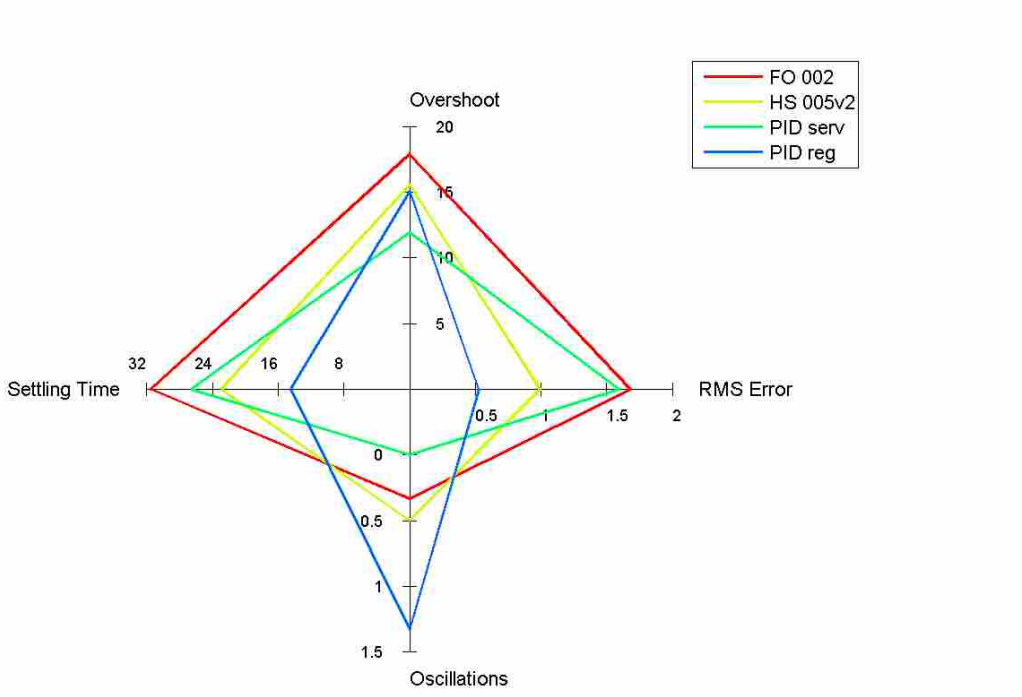
The weld performance for both modeled and unmodeled disturbances was analyzed for the best FOPDT and Hybrid Heat Source controllers and the two PID controllers for the previously presented standard disturbance pattern. The modeled disturbances were changes in the

temperature setpoint and in traverse speed. The unmodeled disturbances occurred when the controllers were initially coming to temperature once the controller took control and when welding over previous welds. These results are shown in Figure 5-14.

For the modeled disturbances, the FOPDT MPC controller has very good performance. It has generally smooth performance and is the second best in each category. The Hybrid Heat Source MPC controller is worst in three of the four metrics, and second worst in oscillations. The PID regulator controller is best in RMS error and settling time, but the worst in overshoot and oscillations. Conversely, the PID servo controller is the best in terms of overshoot and oscillation, but second worse in RMS error and settling time.




(a)




(b)

Figure 5-14: Performance of controllers for (a) modeled disturbances and (b) unmodeled disturbances within the first round of quasi steady state testing.

A second round of testing was done to more closely look at the controllers' unmodeled disturbance behavior. Each of the steps were allowed to run longer in order to more carefully observe their settled behavior after each disturbance. The results are shown in Figure 5-15. The temperature and power vs. time plots can be found in Appendix B.3.

For this testing, the Heat Source MPC controller was dominated in all aspects by each of the other controllers. The FOPDT MPC controller had very good performance and was comparable to the PID controllers, although all three had different strengths and weaknesses. The FOPDT controller had no oscillations, medium settling time and RMS error, and high overshoot. The PID servo controller was nearly identical to the FOPDT controller in oscillations and settling time, but had much better overshoot behavior and slightly worse RMS error. The PID regulator

had the best settling time and RMS error of the four controllers, but had medium overshoot and high oscillation.

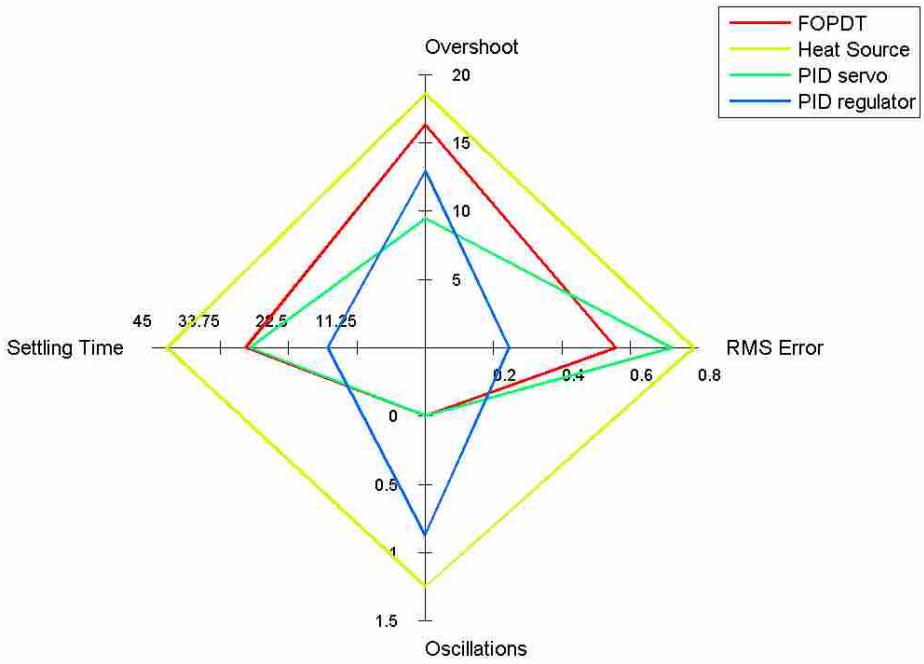



Figure 5-15: Unmodeled disturbance performance of the welds in the second round of quasi steady state. The HS MPC controller is a dominated controller design.

During the transient immediately after the plunge, the FOPDT and PID servo controllers could not obtain good temperature control. The PID regulator and Hybrid Heat Source controllers were able to have decent control. However, the length of those tests was very short, and so the long-term settling behavior as the welds progressed into steady state was not able to be observed. Because of this, two more welds were run for both controllers. The performance metrics are shown in Figure 5-18. The best control of each controller is shown in Figure 5-16 and Figure 5-17 .

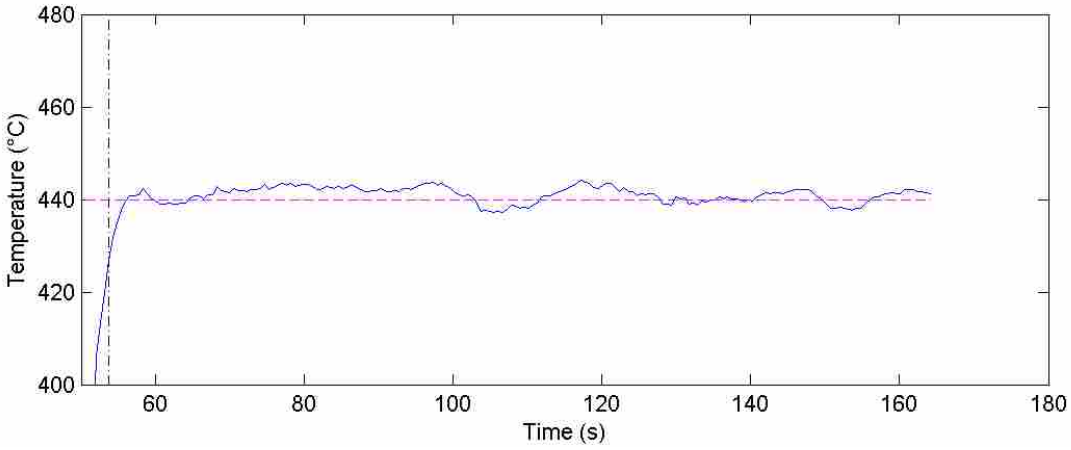



Figure 5-16: Temperature of the HS 103 v2 weld during transient conditions and into steady state. The controller was very steady in nature and was close to the setpoint the entire time, but had a relatively high RMS error after settling compared to other welds. It should be noted that this controller used the same parameters the entire time and did not revert back to the preferred manual parameters after the majority of the initial thermal transient had passed.

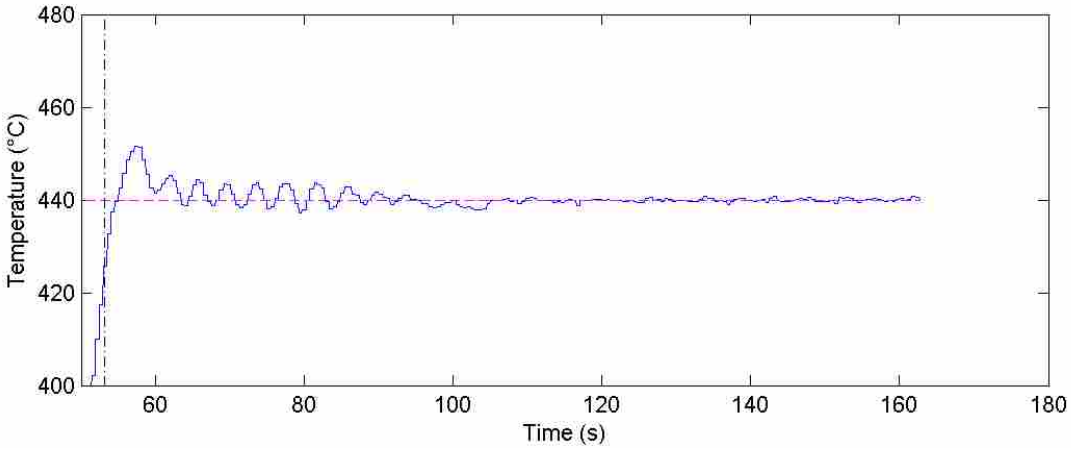



Figure 5-17: Temperature of the PID regulator 102v2 weld during transient conditions and into steady state. The controller overall had excellent control, but did have a large amount of overshoot and oscillations at the beginning.

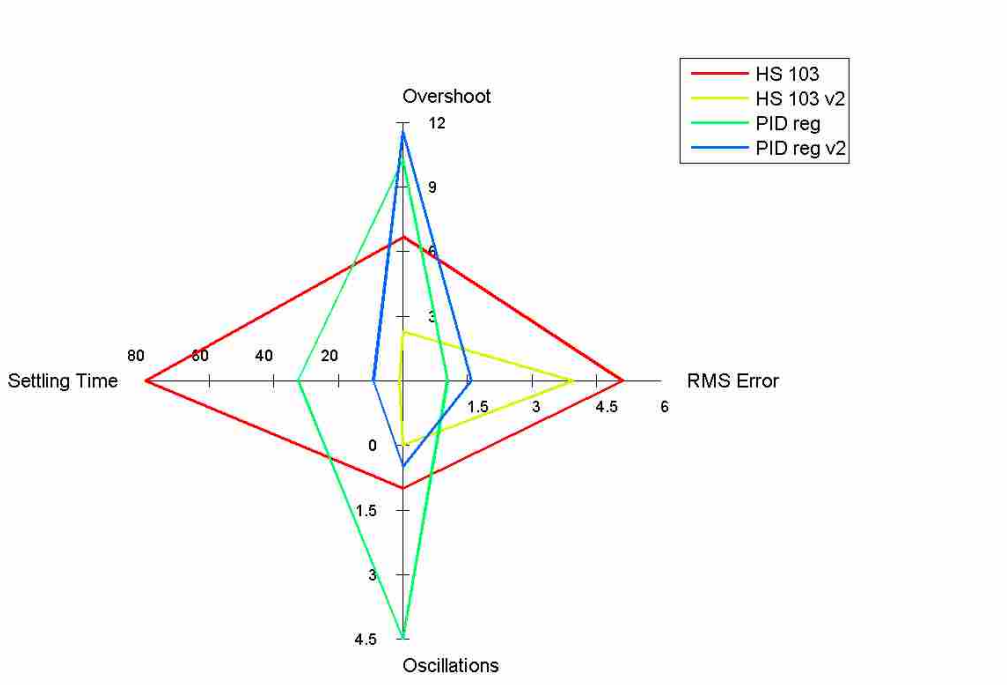



Figure 5-18: Performance metrics for all of the welds in the second round of transient testing. Because it is harder to control temperature during this part of the weld, the tolerance for defining settling and oscillations was raised to ±5°C.

As can be seen in Figure 5-18, the performance for each controller – even with the same exact settings – may vary from weld to weld. Generally, the PID regulator controller had higher overshoot and oscillations, but lower RMS error. Conversely, the Hybrid Heat Source controller had slightly better oscillation and much better overshoot characteristics. The settling time of the two Hybrid Heat Source controllers was both the best and the worst of all the controllers, so it is hard to make even general conclusions with only this data.

Regardless of the controller used, doing a quicker plunge would likely help to mitigate the effect of the transient as the plate will accumulate less heat.

Generally speaking, because MPC uses a model of the process, it deals with setpoint changes relatively well, especially if the model is very accurate. However, it relies completely on

model biasing to overcome unmodeled disturbances, and so its performance here is very much affected by how good of a biasing routine is used. On the other hand, PID has no model and therefore does not have the advantage of foresight when a setpoint change occurs. The PID controllers usually reacted better to unmodeled disturbances than MPC controllers did.

The following table qualitatively compares the controllers in several areas.

Table 5-1: Qualitative Comparison of Controllers. Positive marks mean better, not necessarily larger, and negative marks are the opposite.

| | | PID - Servo | PID - Regulator | MPC - FOPDT | MPC - HS |
|---|---|---|---|---|---|
| Performance | Mid-weld Performance | ++ | ++ | ++ | + |
| | Initial Transient Performance | -- | + | -- | + |
| | Modeled Disturbances | ++ | + | ++ | + |
| | Unmodeled Disturbances | +/- | + | + | +/- |
| | Stability in Various Conditions | ++ | +/- | + | + |
| Implementation | Ease of Implementation | + | + | - | -- |
| | Computational Expense | ++ | ++ | - | -- |
| | Parameter/Gain Determination | + | + | - | -- |

# 6 Conclusions, Future Work, and Applications of MPC for FSW

## 6.1 Conclusions

Model Predictive Control was successfully implemented for FSW. With the correct controller parameters and settings, an MPC controller is comparable in many aspects to a PID controller. The FOPDT MPC controller is a viable quasi steady state controller that works well after the large initial transient stage of the weld. The FOPDT model is of the wrong form to perform well immediately post plunge. The Hybrid Heat Source model, which models the intrinsic heat transfer of the plate and tool, performed much better at the start-up of the weld.

MPC controllers inherently react better to modeled vs. unmodeled disturbances – if they are tuned correctly. The FOPDT MPC controller settled within 15 seconds, had almost no oscillations, and had only 2°C overshoot. Thus, if a welding application demanded a rapidly changing temperature or traverse speed, MPC would be a good candidate for that application. Even when reacting to severe unmodeled disturbances, the FOPDT MPC controller settled within 30 seconds, had no oscillations, had 16°C overshoot, and had an RMS error of .5.

This thesis also confirms that PID is a very good temperature control method for FSW. Two different tuning rules/controllers were revisited in this work. When used at a setpoint temperature similar to that for which parameters were determined, control is very good. The two controllers analyzed in this thesis had different strengths and weaknesses. When reacting to

severe unmodeled disturbances the PID servo controller settled within 30 seconds, had no oscillations, had 9°C overshoot, and had a RMS error of .7

For the first time, MPC and PID controllers were used immediately after the plunge during the heavily transient part of the weld. The PID regulator controller had a high degree of variability between the two runs (a settling time of 10 s and 30 s, and .5 and 4.5 oscillations before settling), but settled quickly and once settled was able to hold the temperature within 2°C of the setpoint. The HHS MPC controller on the other hand had far fewer oscillations (0 and 1 oscillation) before settling, but could only hold the temperature within 5°C of the setpoint. While control is not as good as during the quasi steady state portion of the weld, certain MPC and PID controllers were nevertheless able to control the temperature within about 5°C of the setpoint and transitioned well into the quasi steady state portion of the weld. This is excellent compared to a non-temperature-controlled start-up.

## 6.2 Future Work on MPC Temperature Control

One of the current challenges with MPC is accounting for unmodeled disturbances. It is already known that changes in the vertical position of the tool of thousandths or hundredths of an inch can have large impacts on the temperature of the weld, and that many welding surfaces are not flat. Thus, an MPC model could be created from first principles and/or empirical data that accounts for a vertical position change. With this model and some way to measure the tool-plate distance during a weld, this potentially large source of disturbances could be almost eliminated or at least mitigated.

Generally speaking, other model changes may be beneficial as well. The two MPC models presented here were shown to work well, but no effort was taken to ensure that they were

the optimal models. Thus, changing or removing terms or even combining multiple completely different models into a hybrid model may have a benefit. Also, it is quite probable that the same holds true for PID controllers and that a hybrid controller or a simple PID controller based upon different tuning rules could have superior performance. This is viewed as highly promising because the regulator gains produce a sometimes overly aggressive controller, and the servo gains often result in an overly relaxed controller.

MPC is well suited for on-the-fly model correction and self-tuning methods. As long as there is enough computational power in excess of that required for the normal MPC optimization-based solution, the model could either be corrected on the fly, or model parameters could be determined while doing normal welds instead of having to perform potentially expensive parameter estimation welds. On-the-fly parameter correction was successfully performed in preliminary versions of the FOPDT controller, but was not implemented in results shown in this thesis because of adequate FOPDT MPC performance, a desire to avoid accidentally fitting disturbances, and a lack of time and aluminum plate of the same heat to perform these unneeded welds. Self-tuning methods are already extensively used in the petroleum industry so that a plant model can be determined without having to take the plant off-line for days or weeks to determine system parameters.

Some challenging but possibly quite beneficial features that MPC could enable are MIMO or MISO systems. It is well established in FSW literature that as long as limits of relevant parameters (temperature, force, traverse speed, etc) are not exceeded which produce weld defects, a faster traverse speed can produce better material properties. Also, plant productivity can be improved by performing welds that are faster but still in-spec. Thus, MPC could control both spindle power and traverse speed and control to a fastest possible traverse speed while still

maintaining temperature and staying within limits that will avoid defects. The MPC architecture already provides for constrained optimization. Constraints can be specified which will avoid weld defects so long as that window and the desired factor of safety is already known. The objective function could either be tiered or weighted to attempt to maintain temperature *and* have a high traverse speed. Due to the speed-ups that were implemented for the Hybrid Heat Source model, simultaneously changing both power and traverse speed is not possible within the current controller architecture for that model. However, to get around this a sufficiently accurate look-up table for both past and future moves could be created with interpolation capabilities. A look-up table could also be used to in order to get the grid independence and accuracy of a multipoint tool without incurring that extra cost every single optimization loop. See Appendix A.1.4 for details on this.

## 6.3 Applications of MPC and Other Temperature Control Methods in FSW

Based upon the current performance of well-tuned PID controllers in MPC, there is not a sufficiently large benefit to necessitate MPC in most SISO FSW applications if PID or another reliable temperature control method is already used. However, there are some applications where MPC ought to be seriously considered.

If the temperature of a weld needs to be controlled as soon as possible, using an MPC controller with a model that accurately captures the initial heat transfer can result in a weld being close to the desired temperature relatively quickly. This can include, but is not necessarily limited to, the Hybrid Heat Source method. It may be possible to develop a simpler model that captures the initial behavior well enough for MPC, and then transition to a FOPDT-type model or a PID controller.

Another place where the Hybrid Heat Source MPC controller may be very applicable is in pipe welding. In pipe welding as the tool comes back around to the initial plunge, the plate is already partially preheated and the tool will experience an increase in temperature if the same power is maintained. Using a mirror heat source of the same exact magnitude – but ahead of the tool by the circumferential distance of the pipe – the model should be able to predict the approximate rise of temperature in the pipe and accommodate, and consequentially not overshoot in temperature too much. The 2D version of the heat source method (Carlsaw and Jaeger 1959, 258) should probably be used here instead of, or in conjunction with, the 3D version that is used as the basis for the Hybrid Heat Source model. It may also be possible to use a FOPDT model with a feed-forward disturbance to capture the latent heat already present as the tool re-approaches the initial plunge location. A feed forward method could likely also be used within another temperature control scheme.

## REFERENCES


Behrendt, Martin. 2009. "Model Predictive Control." Wikipedia.org. Last Modified October 2, 2009. Accessed May 29, 2013. http://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg

Carslaw, H. S. and Jaeger, J.C. 1959. *Conduction of Heat in Solids: Second Edition*. pg 258. New York: Oxford University Press.

Cederqvist, L., Garpinger, O., Hagglund, T., and Robertsson, A., 2011. *Reliable Sealing of Copper Canisters through Cascaded Control of Power Input and Tool Temperature.* John Wiley & Sons, Inc.

Chimbli, S.K., Medlin, D.J., and Arbegast, W.J. 2007. "Minimizing Lack of Consolidation Defects in Friction Stir Welds." *The Minerals, Metals, and Materials Society: 135-142.*

Nielson, Issac. 2012. "Modeling and Controlling Friction Stir Welding in 5 cm thick Copper Canisters." Master's thesis, Linköpings Unversitet.

Hou, Z. B. and Komanduri, R. 2000. "General Solutions for Stationary/Moving plane Heat Source Problems in Manufacturing and tribology." *International Journal of Heat and Mass Transfer*, no. 43: 1679-1698.

Incropera, Frank P., Dewitt, David P., Bergman, Theodore L., and Lavine, Adrienne S. 2007. *Fundamentals of Heat and Mass Transfer. 6th Edition.* United States, John Wiley & Sons.

Lakshminarayanan, A.K., Balasubramanian, V., and Elangovan, K. 2007. "Effect of Welding Process on Tensile Properties of AA6061 Aluminum Alloy Joints." *International Journal of Advanced Manufacturing Technology*, no. 40: 286-296.

Mahoney, M. W., Rhodes, C. G., Flintoff, J. G., Spurling, R.A., and Bingel, W. H. 1998. "Properties of Friction-Stir-Welded 7075 T651 Aluminum." *Metallurgical and Materials Transactions A* 29A (July): 1955-1964.

Marshall, Dustin. 2013. "An Alternative System Identification Method for Friction Stir Processing." Master's thesis, Brigham Young University.

Matweb. 2015a. "Aluminum 7075-T6; 7075-T651" Accessed January 6, 2015.http://matweb.com/search/DataSheet.aspx?MatGUID=4f19a42be94546b686bbf43f79c51b7d&ckck=1

Matweb. 2015b. "Alclad Aluminum 7075-T6; 7075-T651" Accessed April 2, 2015. http://matweb.com/search/DataSheet.aspx?MatGUID=8b681656bc654fa8ba6cdb0f802e6468

O'Dwyer, Aidan. 2006. *Handbook of Pi and PID Controller Tunning Rules: Aidan O'Dwyer.* Imperial College Press.

Posada, Maria. 2012. "Evaluation of 3-D Friction Stir Welding Viscoplastic Finite Element Model." PhD diss., Brigham Young University.

Ross, Ken. 2012. "Investigation and Implementation of a Robust Temperature Control Algorithm for Friction Stir Welding." Master's thesis, Brigham Young University.

Saran, Cliff. 2009. "Apollo 11: The computers that put man on the moon." *ComputerWeekly.com*, July. Accesses January 31, 2015. http://www.computerweekly.com/feature/Apollo-11-The-computers-that-put-man-on-the-moon

Taylor, R., Groot, H., Goerz, T., Ferrier, J., and Taylor, D. 1998. "Thermophysical Properties of Molten Aluminum Alloys." *High Temperature – High Pressures* 30, no. 3: 269-275.

Zitney, Stephen E. 1999.. "Chemical Process Engineering on Supercomputers: Capturing the Value." Cray Research, Inc. Last modified March 4, 1999. Accessed February 2, 2015. ftp://ftp.cray.com/applications/news/Chem_Process_value.9312.txt.

# Appendix A. Computational and Numerical Details

## A.1 Computational Time Reduction for the Heat Source and Hybrid Models

Presented in the following subsections are details which either were or could be implemented in order to reduce the computational time of calculating temperatures via the heat source method. Unless otherwise stated, all simulations will be for 3.8 mm/s (9 ipm), 2.9kW, and 3.2 mm (.125 inches) deep. All methods use a center-point integration scheme.

### A.1.1 Non-uniform Time Discretization

If a time step of 1 second is used, the calculated temperature error is about 193°C, or about 42.5% off. In order to achieve an accuracy of about 1°C by using a uniform time step-size, a step size of about .02 seconds is necessary. For a weld taking about 400 seconds, by the end of the weld about 20,000 points must then be used for the integration. This is because at the most recent times, the curvature of $d\theta/dt$ is very high and requires a fine resolution in order to capture the curve well. The curve which must be integrated is shown in two different forms below.
As can be seen, the curve has both a very high slope and curvature in the beginning, and this levels out relatively quickly. Thus, events in the distant past to not need to be calculated with a time spacing as fine as events in the near past do. In order to overcome this, a non-uniform time spacing was created that used very fine spacing in the near past and very wide spacing in the far

past. The base configuration is shown below in Table A-1. The optimal spacing and resolutions would certainly depend upon the traverse speed of the heat source and depth of the query point.

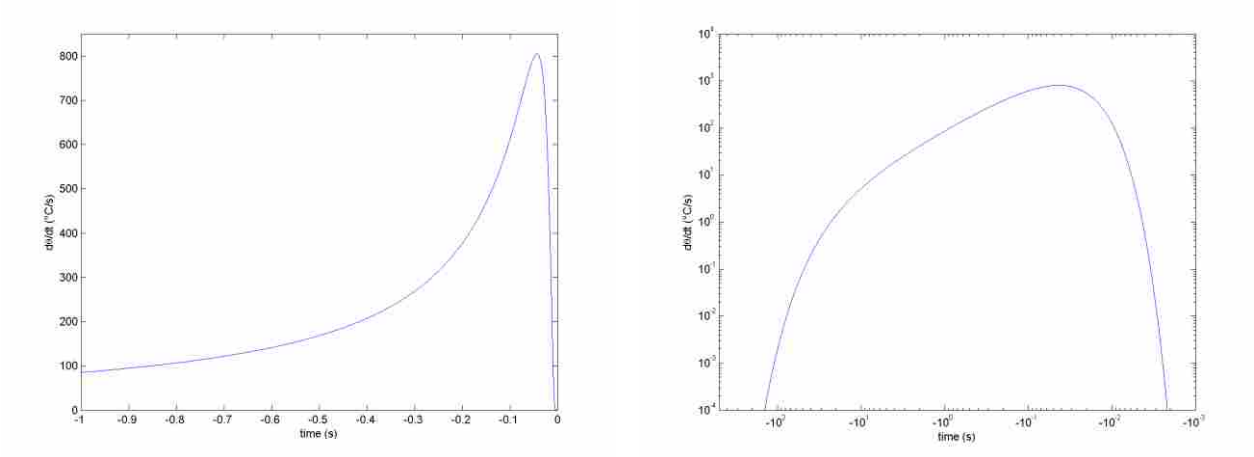



Figure A-1: Differential increases in temperature vs. time for the heat source method.

Furthermore, the above spacing was determined manually and does not necessarily represent an optimal integration scheme for the given parameters. For grid independence studies, the bounds were kept the same but the resolutions were all quadrupled, doubled, halved, quartered, etc.

Table A-1: Standard Time Discretization Scheme for the Non-Uniform Time Spacing of Data for the Heat Source Method.

| Upper Bound (s) | 0 | -.03 | -.05 | -.1 | -.4 | -1 | -5 | -10 | -50 |
|---|---|---|---|---|---|---|---|---|---|
| Lower Bound (s) | -.03 | -.05 | -.1 | -.4 | -1 | -5 | -10 | -50 | $-\infty$ |
| Resolution (s) | .005 | .01 | .025 | .1 | .2 | .5 | 1 | 5 | 25 |

This was very effective at more accurately calculating the temperature rise due to the heat source method with fewer time divisions for the integration, as can be seen in the below figure. On average, the same accuracy could be achieved by using one to two orders of magnitude less points, or by using the same number of points, the variable time division scheme achieved results about two to three orders of magnitude more accurate.

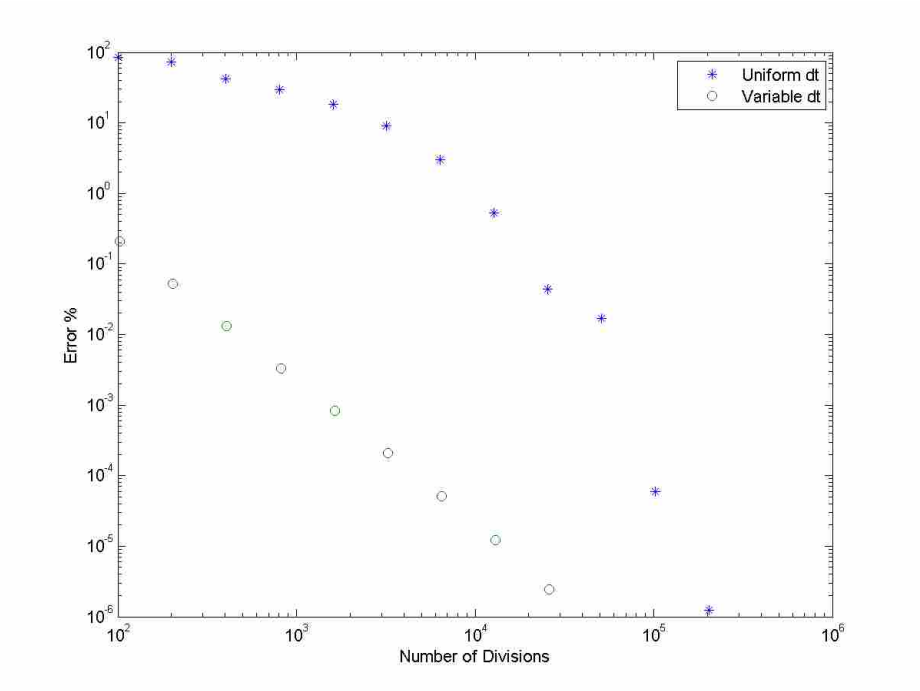



Figure A-2: Absolute temperature error percent vs. number of divisions for several different grid resolutions for both the uniform and variable time-spacing schemes.

Because fewer points are needed, there is also a reduced calculation cost associated with this. This is shown below. Much more accurate results can be obtained in the same time, or the same accuracy can be obtained in about one-tenth of the time.

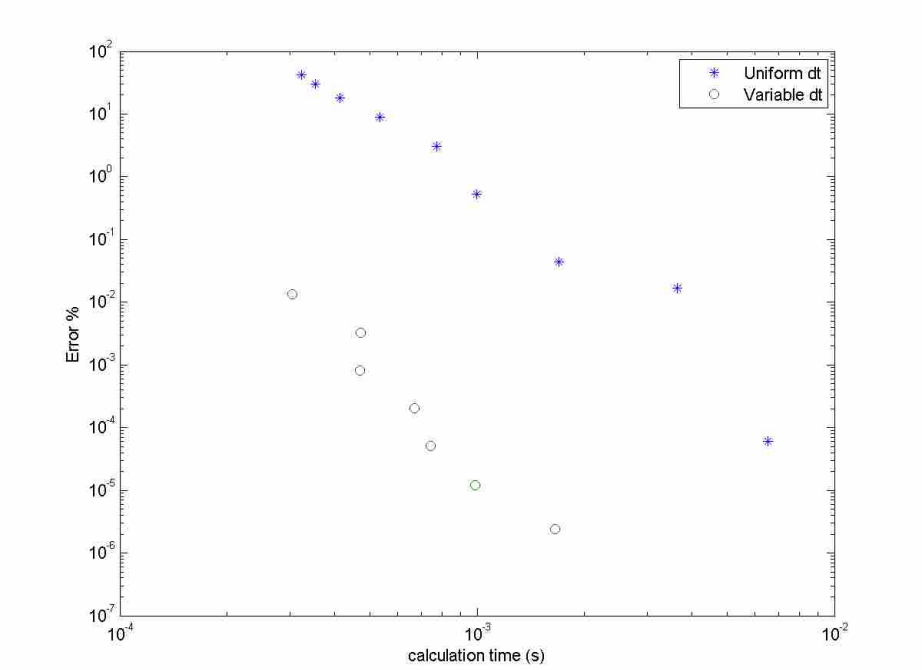



Figure A-3: Absolute temperature error percent vs. calculation time for several different grid resolutions for both the uniform and variable time-spacing schemes.

### A.1.2 Precalculating Segmented Temperature Rises Based Upon Unit Power Inputs

For convenience, Equation (2-11) is copied below:

$$\theta = \frac{1}{4c\rho(\pi a)^{3/2}} \int_{-t_0}^{0} \frac{Q_i}{\tau_i^{3/2}} \cdot e^{-R_i^2/4a\tau_i}\, dt \tag{2-11}$$

If the heat input over a given segment of time is constant, then the equation can be written as:

$$\theta = \frac{Q_i}{4c\rho(\pi a)^{3/2}} \int_{-t_0}^{0} \frac{e^{-R_i^2/4a\tau_i}}{\tau_i^{3/2}} dt \tag{A-1}$$

Thus, even if the heat/power is unknown, the integral and denominator can still be calculated as long as (1) the bounds on the time are known and (2) the positions (and therefore the $R_i$ term) at those times are also know. Thus, $R_i$ can be thought of as a function of $t$.

As long as the traverse speed is kept constant, then the above conditions are met, and the integral can be calculated in advance. This has two direct applications, both for when the power is known and unknown, and which are both applicable to the MPC application.

When the power is known – such as for system inputs that have already happened – then the temperature rise over the segment can be fully calculated. When the power is predicted, such as for future events, the temperature rise over that segment can be calculated based upon a unit $Q_i$, and then scaled appropriately. An example is now given. For simplicity's sake, the following is defined:

$$Q_i \int_{-t_0}^{0} X\, dt = \frac{Q_i}{4c\rho(\pi a)^{3/2}} \int_{-t_0}^{0} \frac{e^{-R_i^2/4a\tau_i}}{\tau_i^{3/2}} dt \tag{A-2}$$

For this example, a history of powers, positions, and time is known for all past events and the future velocity is known, which means that the integral portion can be calculated. This example will have three future points that the optimizer will optimize for: going from $t = 0$ to .5, .5 to 1.5, and 1.5 to 3. It must also be remembered that the relative $t = 0$ is at the end of each step. Thus, for the end of the third step (which is actually 3 seconds away from the present) $t = 0$. From the perspective of the end of the third step, the "actual" present time would then be $t = -3$ (from that perspective), and an event which is currently 5 seconds in the past (from the actual current time) will then be $t = -8$ from the perspective of the end of the third step. In other words, if an event happened 5 seconds in the past, and its effect at 3 seconds in the future is desired, then $\tau$ would be 8 seconds. Therefore, one can say that:

$$\theta_{past,1} = \frac{1}{4c\rho(\pi a)^{3/2}} \int_{-t_0}^{-.5} \frac{Q_i}{\tau_i^{3/2}} \cdot e^{-R_i^2/4a\tau_i}\, dt \tag{A-3}$$

$$\theta_{past,2} = \frac{1}{4c\rho(\pi a)^{3/2}} \int_{-t_0}^{-1.5} \frac{Q_i}{\tau_i^{3/2}} \cdot e^{-R_i^2/4a\tau_i}\, dt \tag{A-4}$$

$$\theta_{past,3} = \frac{1}{4c\rho(\pi a)^{3/2}} \int_{-t_0}^{-3} \frac{Q_i}{\tau_i^{3/2}} \cdot e^{-R_i^2/4a\tau_i}\, dt \tag{A-5}$$

The portion of temperature rise for instances in the future due to past events is completely known and does not need to be calculated again.

The portion of temperature rise for instances in the future due to events in the future still needs to be determined, and expressed through matrix multiplication is:

$$\begin{bmatrix} \theta_{future,1} \\ \theta_{future,2} \\ \theta_{future,3} \end{bmatrix} = \begin{bmatrix} \int_{-.5}^{0} X\, dt & 0 & 0 \\ \int_{-1.5}^{-.5} X\, dt & \int_{-.5}^{0} X\, dt & 0 \\ \int_{-3}^{-1.5} X\, dt & \int_{-1.5}^{-.5} X\, dt & \int_{-.5}^{0} X\, dt \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{A-6}$$

In effect, the future powers, $Q_1$, $Q_2$, and $Q_3$ simply become scaling factors for the integrals, which can be precalculated. Therefore, end temperatures are simply:

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \theta_{past,1} \\ \theta_{past,2} \\ \theta_{past,3} \end{bmatrix} + \begin{bmatrix} \int_{-.5}^{0} X\, dt & 0 & 0 \\ \int_{-1.5}^{-.5} X\, dt & \int_{-.5}^{0} X\, dt & 0 \\ \int_{-3}^{-1.5} X\, dt & \int_{-1.5}^{-.5} X\, dt & \int_{-.5}^{0} X\, dt \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} \tag{A-7}$$

The heat source problem then becomes a linear problem and can be evaluated extremely quickly by the optimizer, so long as the number of future command points is not unreasonably large. The integrals could be calculated each time as new data comes is, or could be found based upon a look-up table. If the velocity is constant, then all of the entries along a diagonal should be the same. If not, then every value may be different.

### A.1.3 Parallelizing Code

The heat source method is particularly well suited for parallelized methods. This is because partial results can be calculated and then simply added together at the very end. For instance, if the data is extremely finely meshed, and if using a single point heat source and a single point of inquiry, then the integration can be split up among multiple threads and the results then added together. However, in most cases this is not necessary and may even result in no computational benefit due to the cost of setting up multiple threads.

Parallelizing can be particularly helpful when the heat source is a distribution of points and/or the temperature rise at multiple points is desired. In these cases, everything is the same except for the distance, or $R_i$, terms. Each thread can calculate one temperature rise and then move onto a different source/desired point. This type of analysis is particularly well suited not only to multi-core CPUs, but also GPUs. GPUs do not perform very well for multiple instruction type problems, but do very well for same instruction multi-dataset problems, such as this.

A potential drawback to using GPUs is that many can perform teraflops of single precisions calculations for lower dollar and thermal costs than CPUs, but they lack in double precision capabilities. For a GPU architecture, a GPU with good double-precisions capabilities is most likely needed. Many/most GPUs have a 1:16 or 1:32 penalty when performing double precision calculations, however some do not. As of 2015, examples of high performance GPUs that only suffer a 1:4 or less penalty include: R9 280 & 280X and Fire Pro W8100, W9100 & S9150 from AMD; and GeForce GTX Titan, Titan Black & Titan Z and Tesla S2050, K20, K20X, K40 & K80 from Nvidia. These provide in the range of .8 to 2.9 double-precision TFLOPs depending on the particular card for about $200 - $4,000. As a comparison, even a motherboard with two 2.3 GHz hyperthreaded 18 core Xeon E5-2699v3 server-grade processors

will not be able to produce more than a few hundred GFLOPs of double-precision calculation, and at a very high cost of about $8,000. This being said, in most cases this level of processing power is not needed unless a multinode heat source is being computed at multiple points in real time.

### A.1.4 Look-up Table Based Upon Section A.1.2

In order to calculate the temperature rises based upon a unit heat input, the start and end times and tool positions must be known. Therefore, an exhaustive look-up table would have to be four-dimensional. In order to avoid situations that would be completely useless (such as an end time/position being before the start), instead of absolute end times and positions, those two dimensions should be time duration and distance traveled during said duration.

One problem with a resolution full look-up table, however, is that many values will never need to be accessed and the time creating the table and reading the table may become nontrivial. In order to avoid this, the table can be split into two separate three-dimensional tables, and these tables may only need to be accessed two dimensions at a time. The two tables would be for the past and future values.

The future values table is easier to set up, and most likely smaller in size, and should be quicker for accessing values. Even though the positions and powers are unknown, both start and end times are known for each segment because the time horizon spacing is known. Thus, the combining of the two time dimensions into one time dimensions shrinks the table from 4D to 3D.

The first dimension of the look-up table will need as many entries as there are points in the MPC horizon. For example, if the time horizon had four entries corresponding to 0 to .5, .5 to

1.5, 1.5 to 3, and 3 to 5, a look-up table for this problem would have 4 entries along the first dimension.

The second dimension is the tool position in the weld direction at the end of each step. The third dimension is the distance the tool traveled during that step. Both of these dimensions will need to be fine enough for accurate look-up, and luckily both have limits on their values. The end tool position must start at 0 and go to the maximum velocity that is allowed multiplied by the total time in the time horizon up until the last entry. If, for example, the maximum velocity was 5.1 mm/s (12 ipm), then the maximum value that this dimension must go up to would be 15.3mm (6 inches), corresponding to 5.1 mm/s (12 ipm) for 3 seconds. The second dimension must start at the minimum velocity (usually 0) multiplied by the minimum time step (.5 seconds in this example), and go up to the maximum velocity (5.1 mm/s or 12 ipm) multiplied by the maximum time covered by a single horizon segment (2 seconds), and therefore would be .4 seconds for this example. For both of these dimensions, finer resolution should be used for the smaller entries. This is because the value of the integral for events that are physically closer both change quicker and have a larger impact than events that have taken place far away.

In use, the optimizer would have a set of powers and velocities for the future values. For the $n^{th}$ horizon step, the $n^{th}$ value along the first dimension is selected, and then the 2D data in that dimension is interpolated against to obtain the value of the unit-power integral for that $n^{th}$ horizon step. The results from these are loaded into the matrix as detailed previously and these values are scaled for the power.

The methodology for the past data's look-up table is similar, but the limits for the three dimensions are not defined quite as well as for the future values.

## A.2 Time Grid Independence of the Heat Source Model

The time-based grid independence for the heat source method is already shown in Appendix A.1.1. As can been seen in the figures in that section, as the grid is continually refined the accuracy is increased. Unless otherwise stated, all calculations in this thesis are based upon a variable time spacing as described in that section with at least 400 nodes in that spacing. This results in an accuracy of about .01%, or roughly .05°C. This is viewed to be more than sufficient as the temperature in the tip of the tool is not even measured with that degree of accuracy or precision.

## A.3 Spatial Grid Independence of the Heat Source Model

Below are several different cross sections of predicted temperature for the heat source model based upon several different distributions of the heat source at the top of the plate. The results are shown in Figure A-4 through Figure A-11; for these figures, the color axis is in °C, and the spatial axis is in inches. The tool is traveling from left to right in all of these figures. Also, the maximum temperature that is represented by a color is 500°C. This was done to assist readability of the cooler locations. The below table shows different configurations that were used for these studies. Eight nodal configurations were used, each with different divisions both radially and circumferentially. The finest resolution had 4096 nodes.

Table A-2: Nodal Configurations for Spatial Grid Independence Studies of the Heat Source Method.

| Configuration # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Radial divisions | 1 | 1 | 1 | 2 | 4 | 8 | 16 | 32 |
| Circumferential divisions | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |

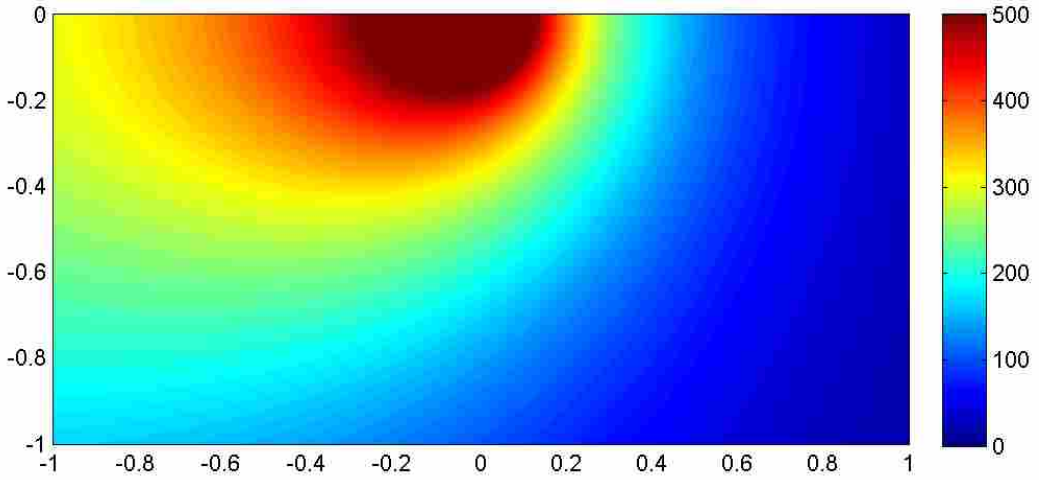



Figure A-4: Thermal profile for heat source tool node configuration #1

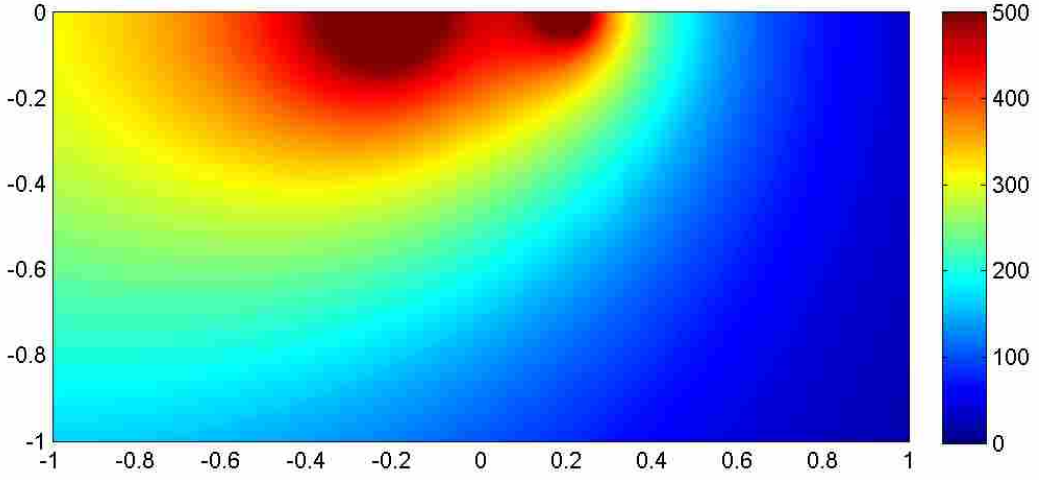



Figure A-5: Thermal profile for heat source tool node configuration #2

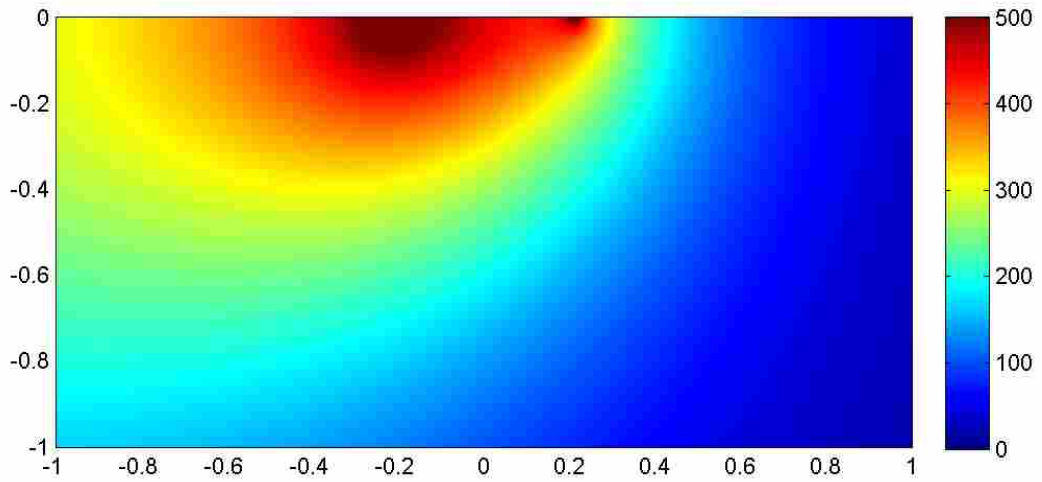



Figure A-6: Thermal profile for heat source tool node configuration #3

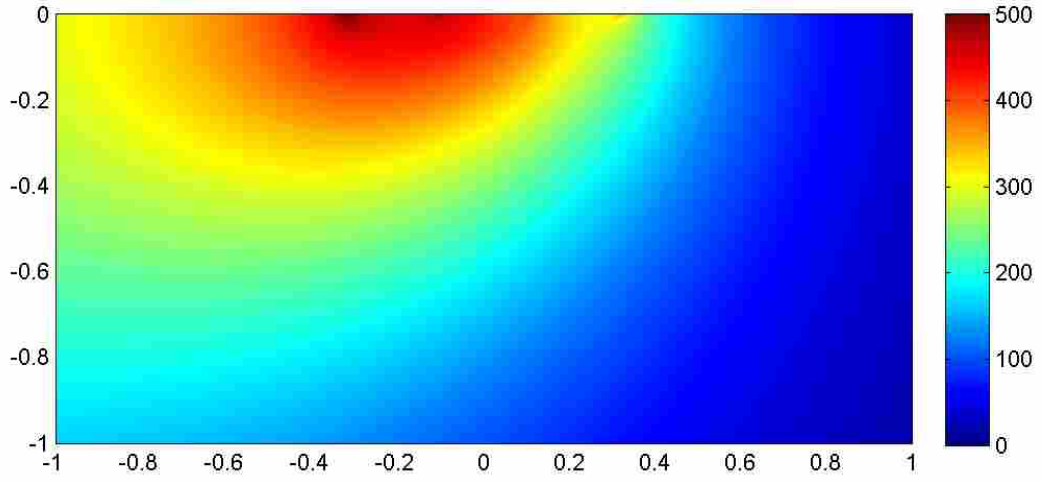



Figure A-7: Thermal profile for heat source tool node configuration #4

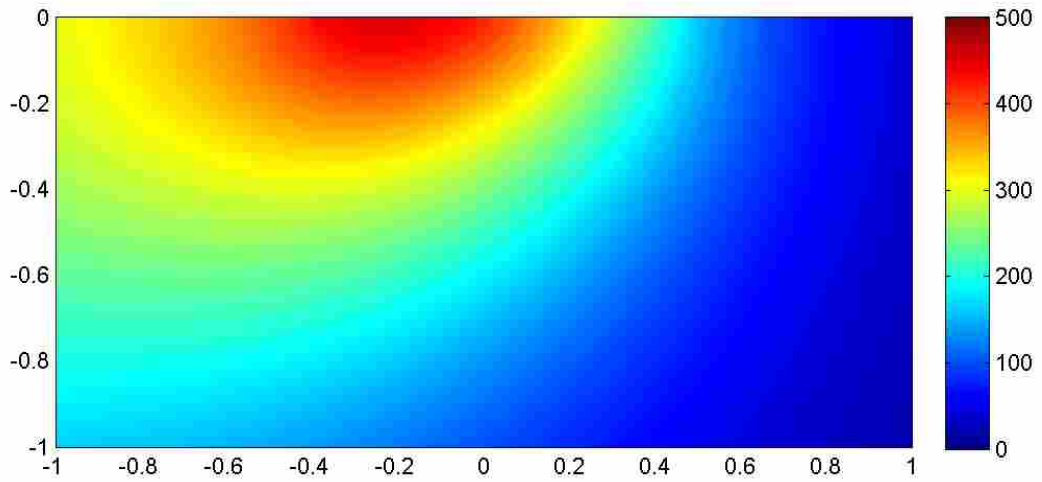



Figure A-8: Thermal profile for heat source tool node configuration #5

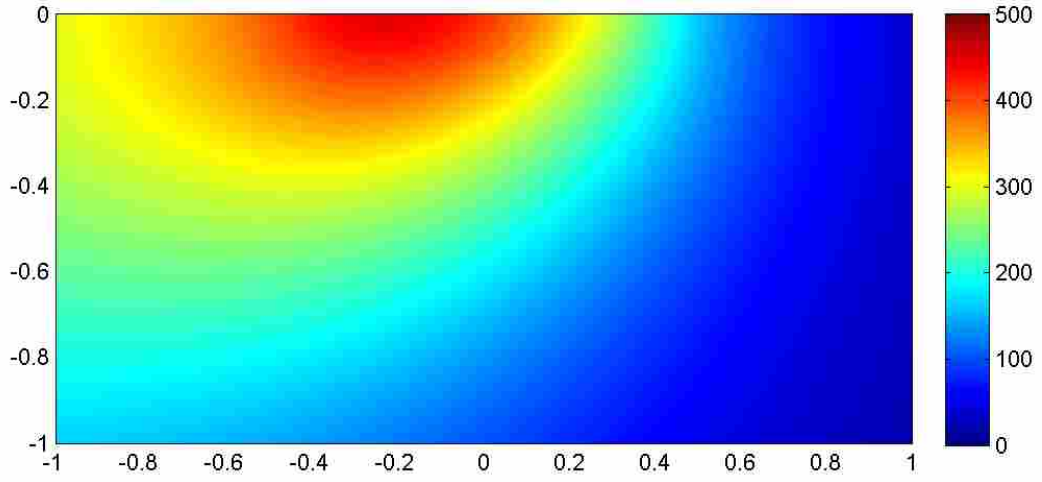



Figure A-9: Thermal profile for heat source tool node configuration #6

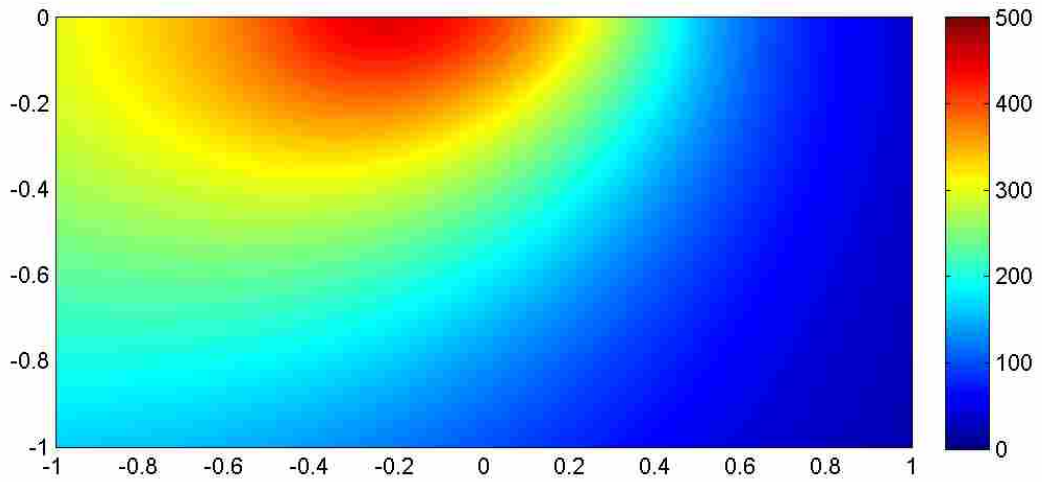



Figure A-10: Thermal profile for heat source tool node configuration #7

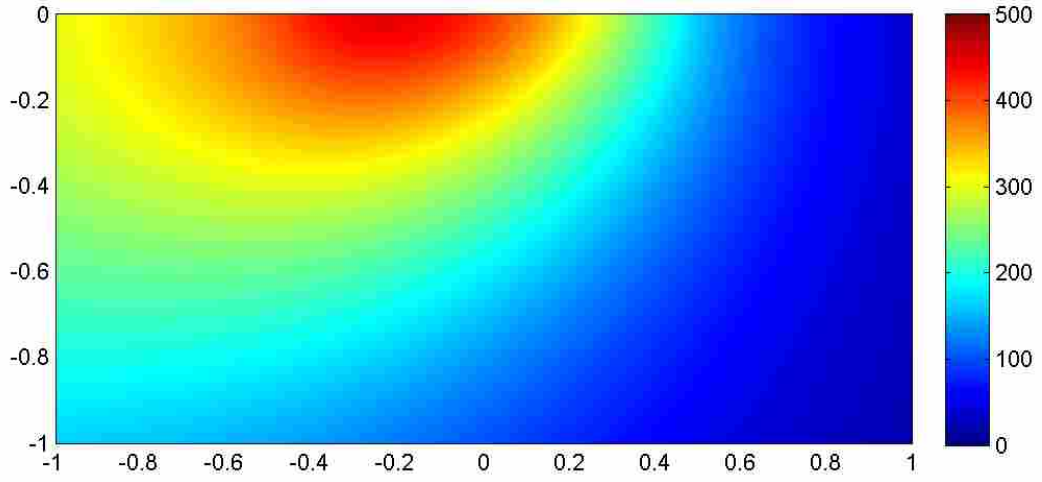



Figure A-11: Thermal profile for heat source tool node configuration #8

As can be seen, the temperature profile in the plate changes as the point heat source become more distributed. At first, the peak temperatures are very high directly under each node. However, as more nodes are added, each node has less intensity and thus temperature peaks due to a given individual node are less intense, and the temperature eventaully evens out into a smooth curve. Furthermore, as more nodes in the tool are added, the peak temperatures shift farther back. Also, as the point depth is made deeper, the difference between the different nodal configurations becomes less. These trends are more clearly seen by looking at the streamlines at specific depths.

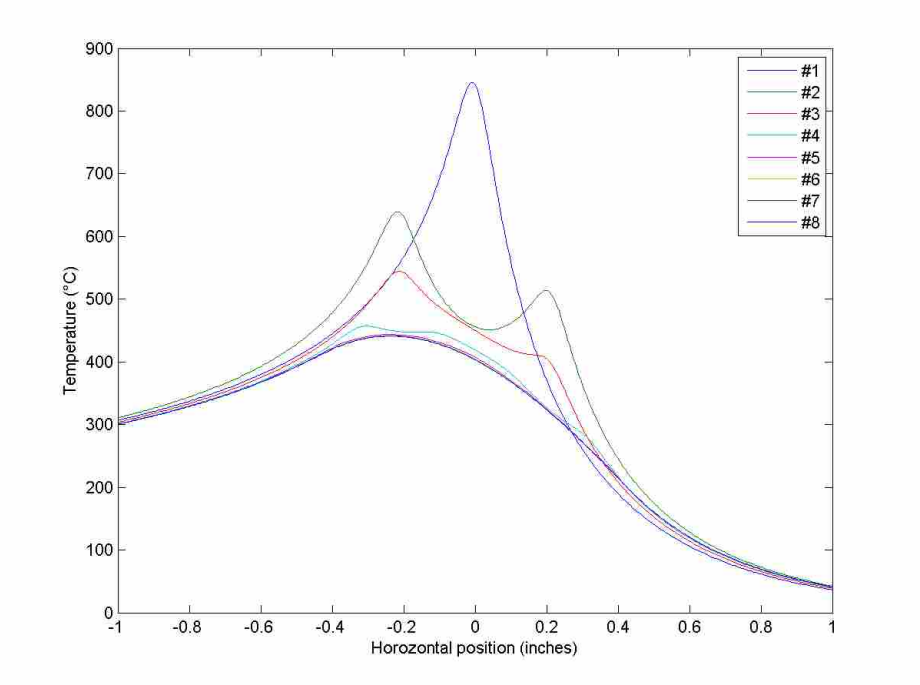



Figure A-12: Temperature streamlines at a depth of .05" for various node configurations.

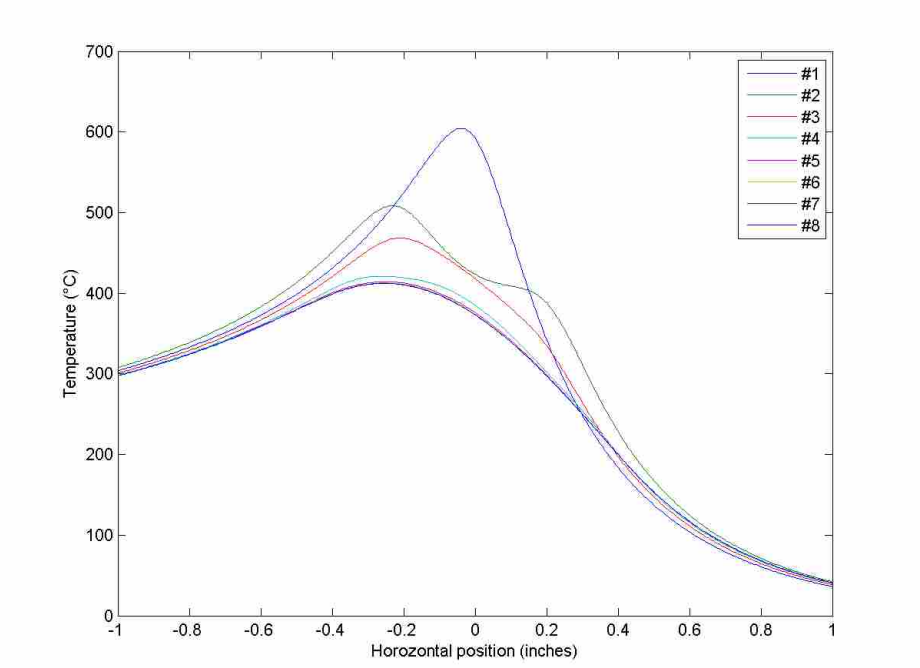



Figure A-13: Temperature streamlines at a depth of .125" for various node configurations.

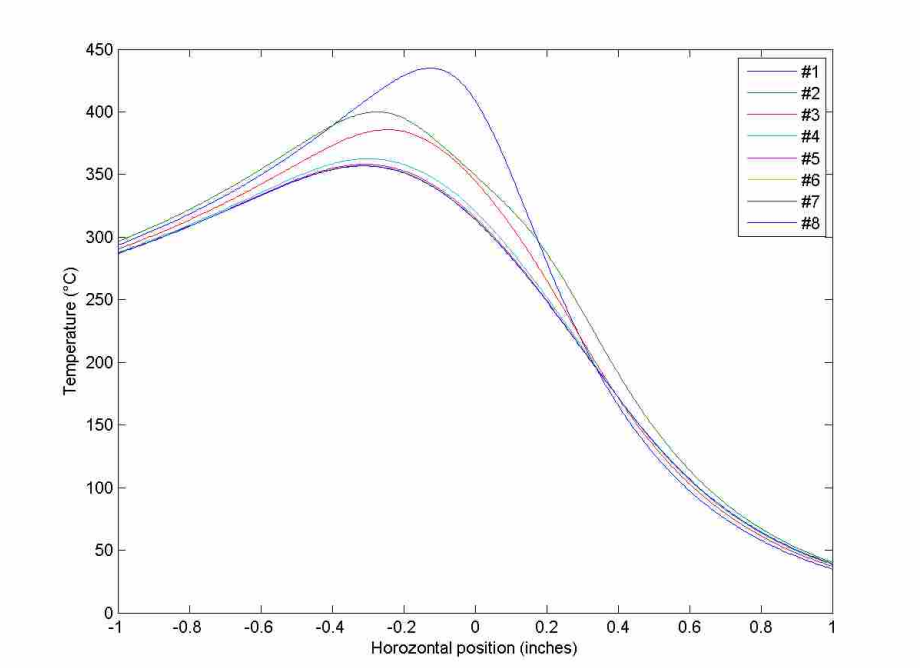



Figure A-14: Temperature streamlines at a depth of .25" for various node configurations.

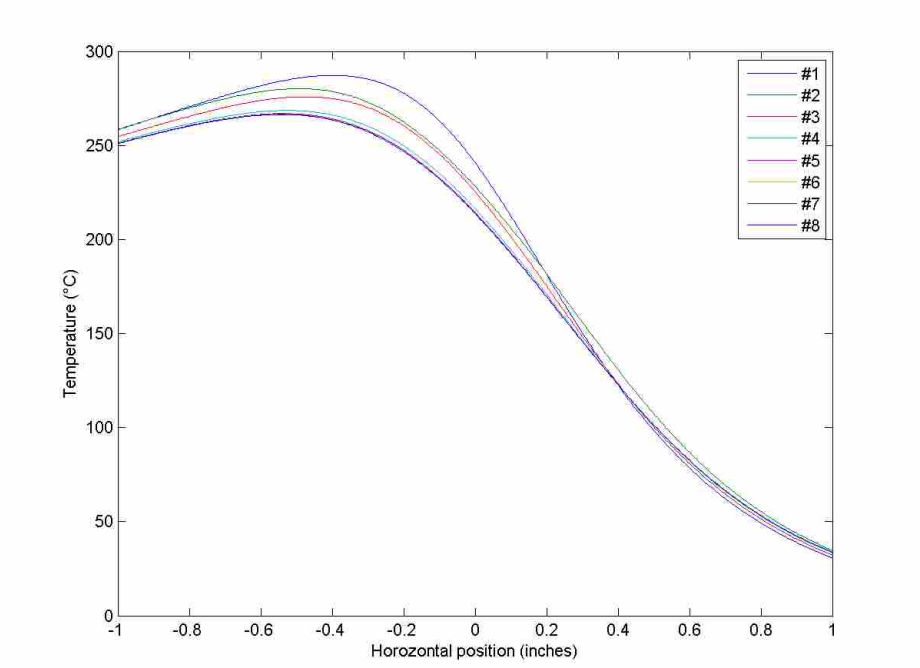



Figure A-15: Temperature streamlines at a depth of .5" for various node configurations.

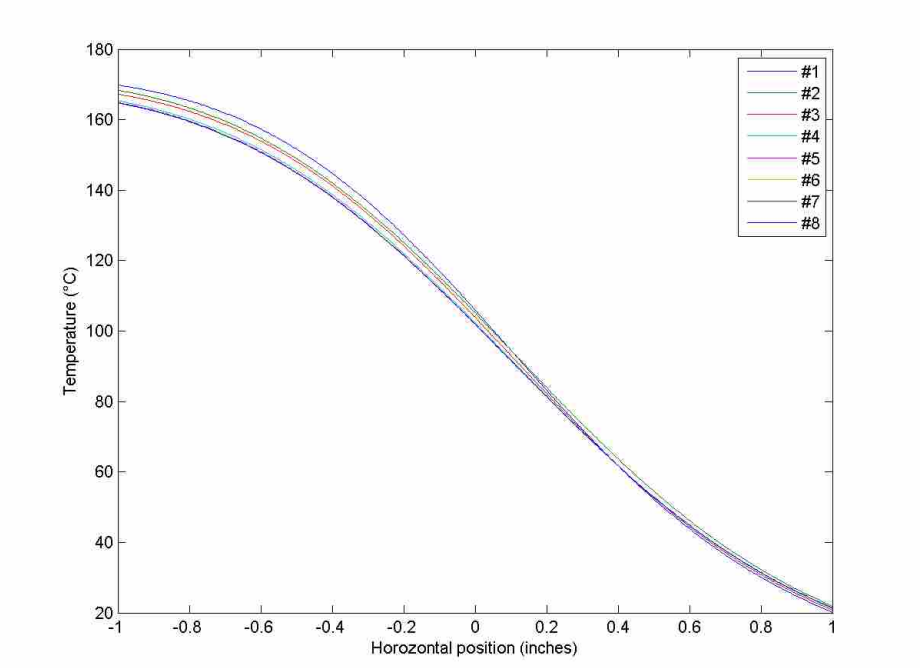



Figure A-16– Temperature streamlines at a depth of 1" for various node configurations.

## Appendix B. Weld Data

Presented below are figures that show the full data for each weld in this thesis. An example is shown below with a legend for the line types. These are consistent throughout this section but not necessarily between this section and other sections. Not all line types may be present in all figures. For example, since PID controllers do not use a predictive model, no predicted temperatures are shown.

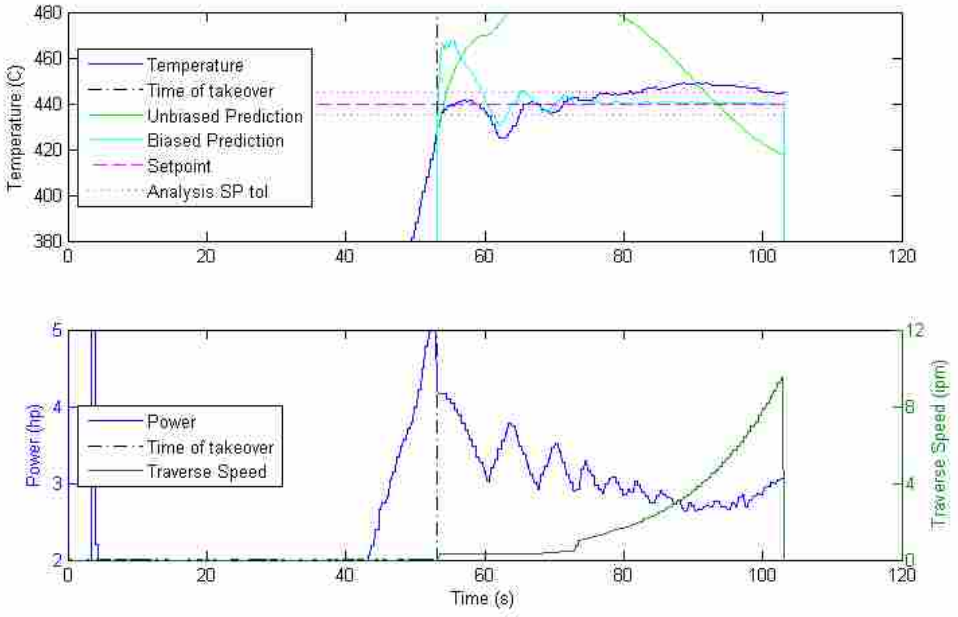



Figure B-1: Example of common line types in plots as used in Appendix B.

## B.1 Weld Data for Initial Parameter Estimation/PRBS Welds

For these welds, the 002 and 003 welds had the power and traverse speed segments repeated twice for the entire weld, and in the 004 welds the segments were repeated four times. Also, duplicates of the 003 and 004 welds were run. Note how there are significant differences between welds and weld segments, despite nearly identical input waveforms.

For example, the minimum temperature of 003 is 430°C, but is 378°C in 003 v2. Also, in the 004 and 004 v2 welds the inputs at times of about 160, 300, 440 and 580 seconds correspond to each other, but each of these segments has a different magnitude of temperature drop within each weld, even though the two welds are very similar to each other. However, at the times of 240, 380, 520, and 660 seconds, the temperature is roughly the same, with the exception of the last instance which is a bit hotter in both welds.

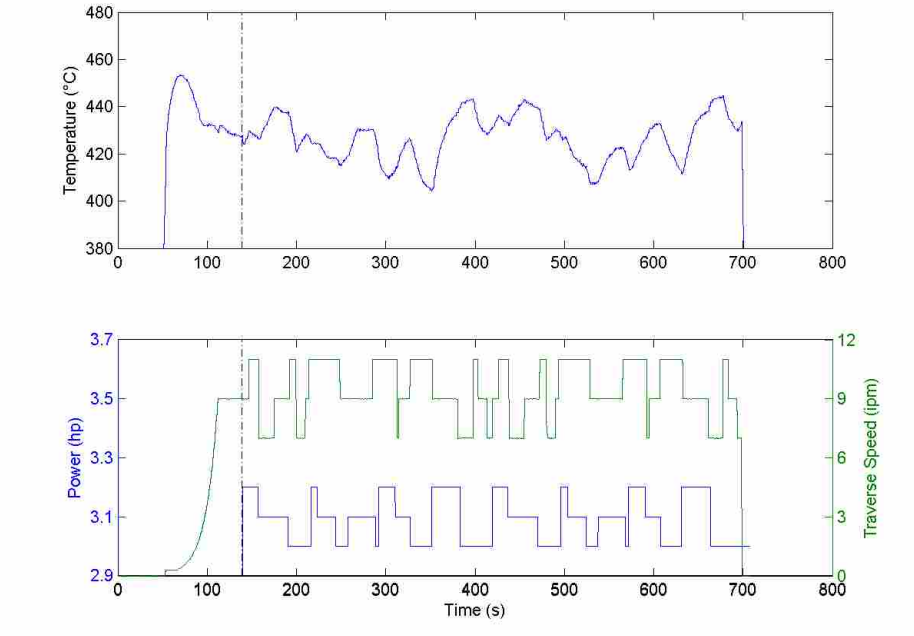



Figure B-2: Temperature and power data for the PRBS 002 weld.

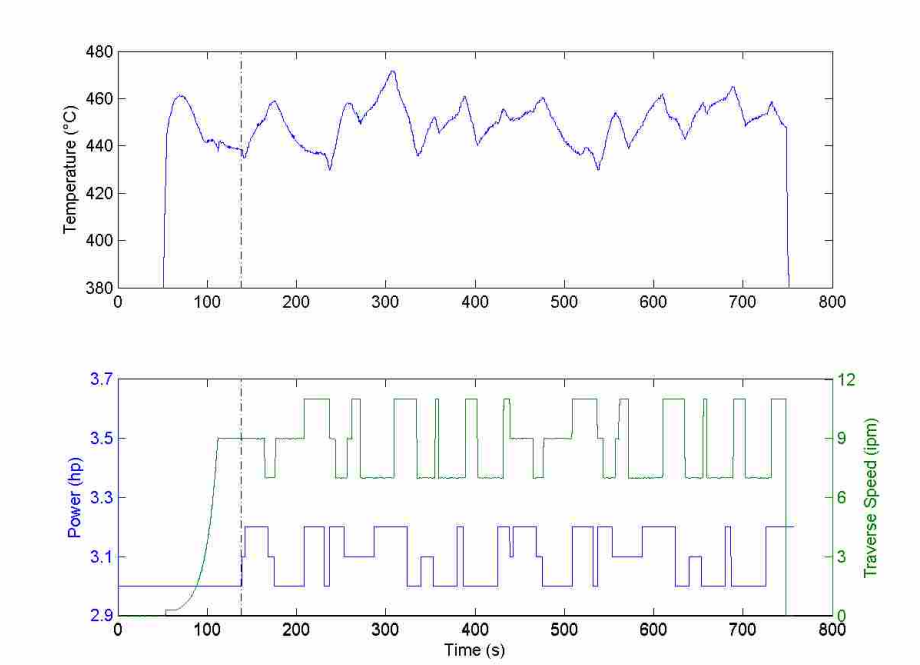



Figure B-3: Temperature and power data for the PRBS 003 weld.

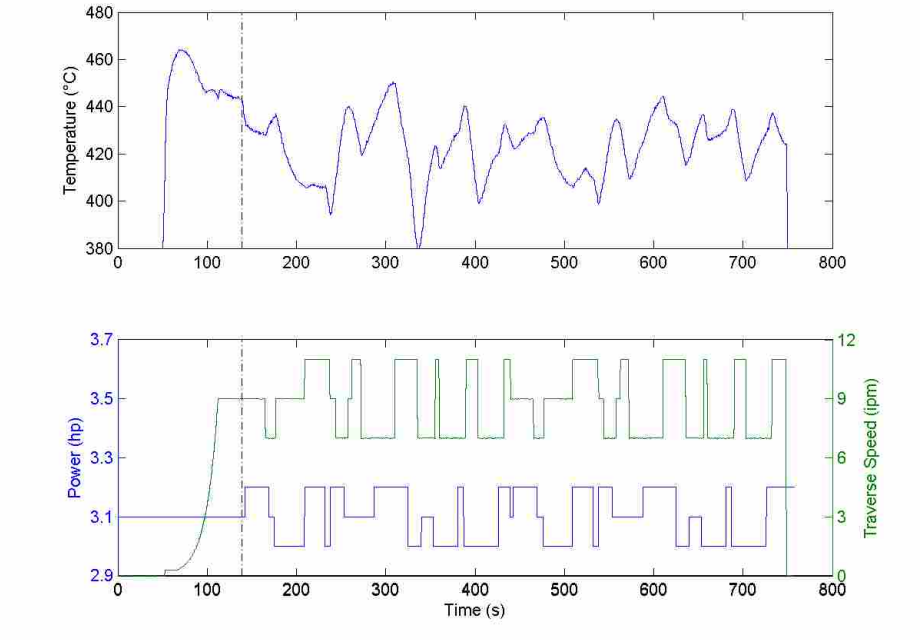



Figure B-4: Temperature and power data for the PRBS 003 v2 weld.

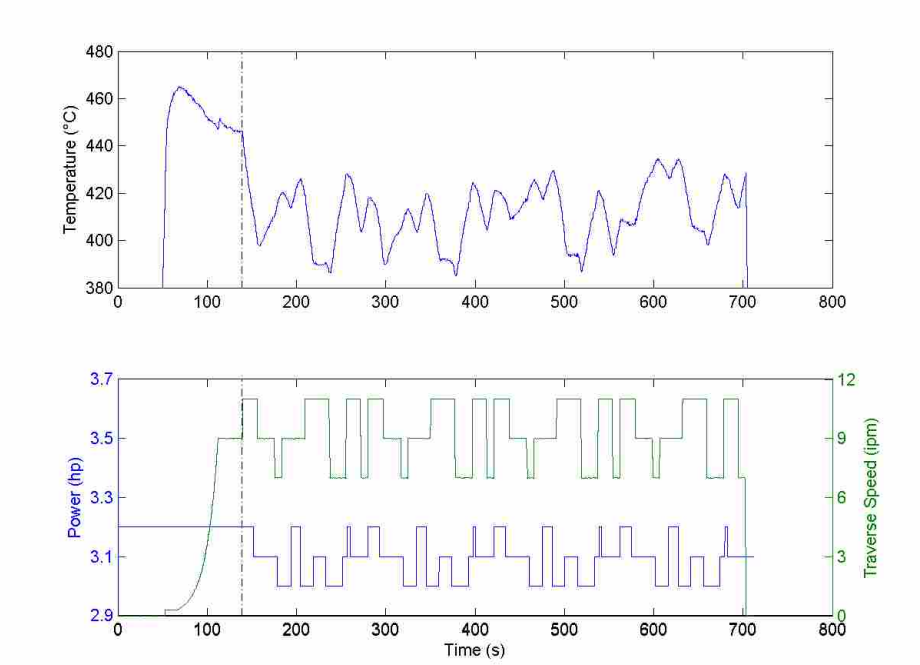



Figure B-5: Temperature and power data for the PRBS 004 weld.

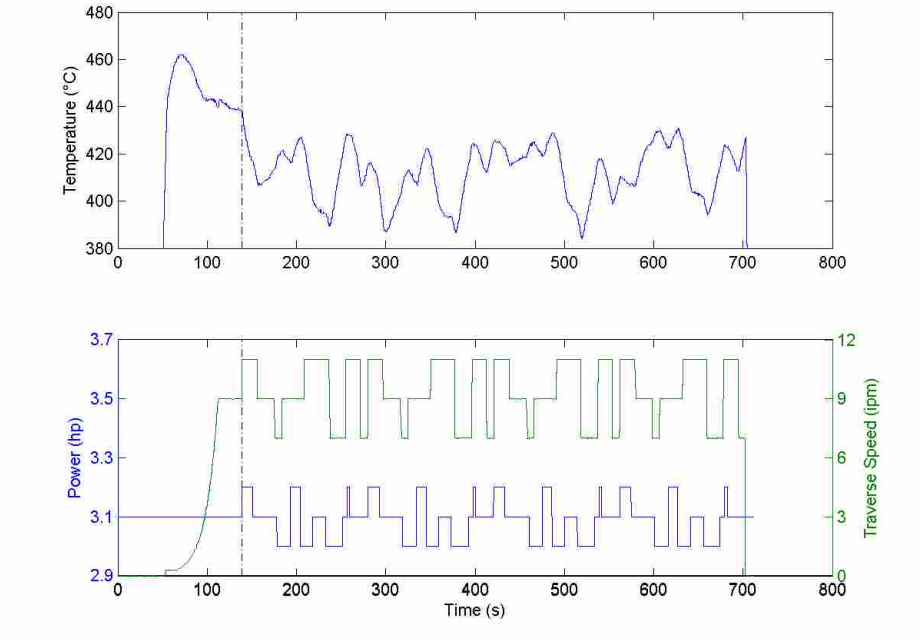



Figure B-6: Temperature and power data for the PRBS 004 v2 weld.

## B.2 Weld Data for the First Round of Quasi Steady State Testing

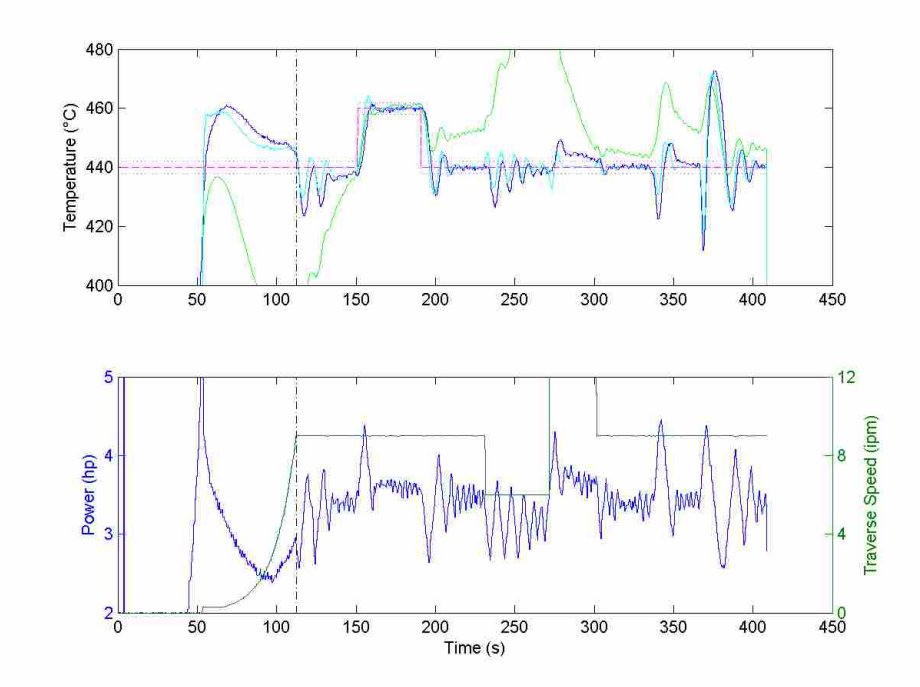



Figure B-7: Temperature and power data for the FOPDT 001 weld.

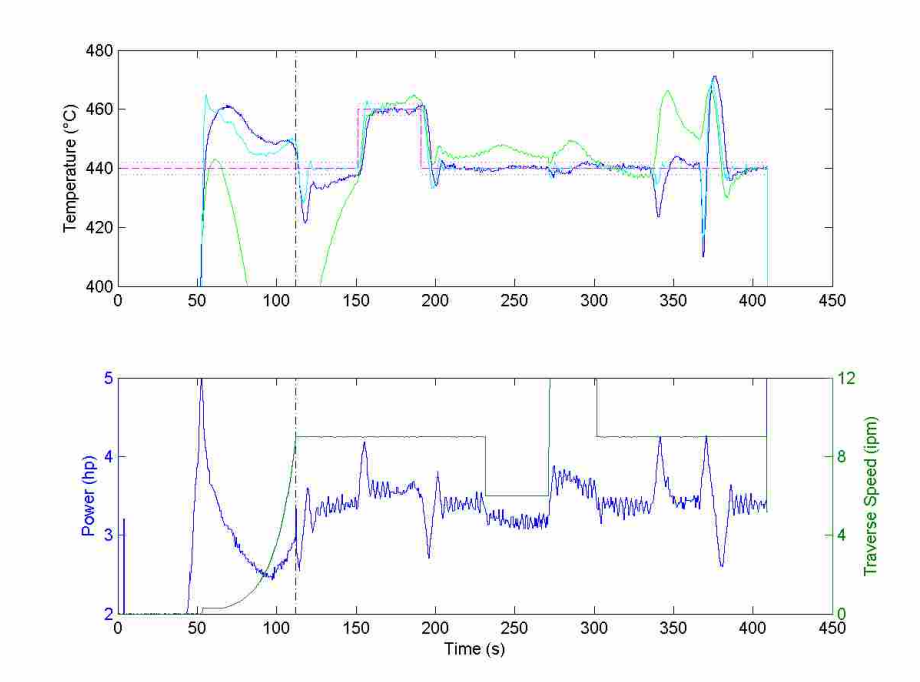



Figure B-8: Temperature and power data for the FOPDT 002 weld.

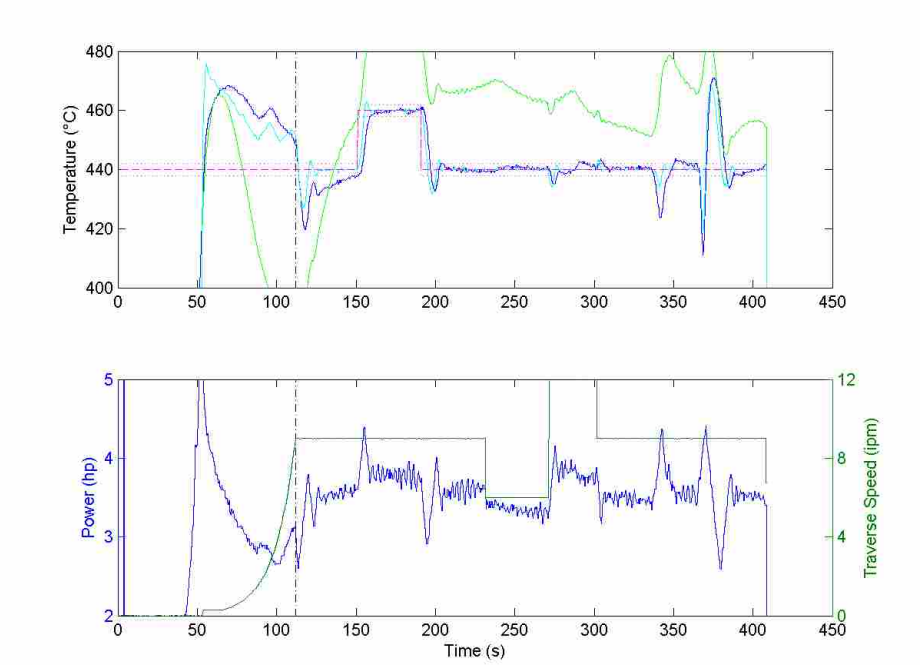



Figure B-9: Temperature and power data for the FOPDT 003 weld.

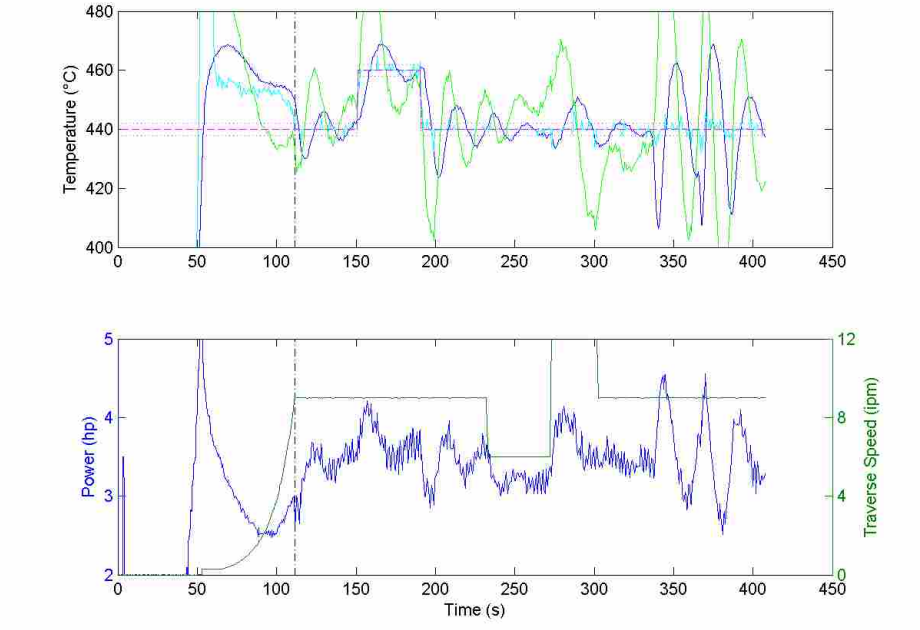



Figure B-10: Temperature and power data for the HS 001 weld.

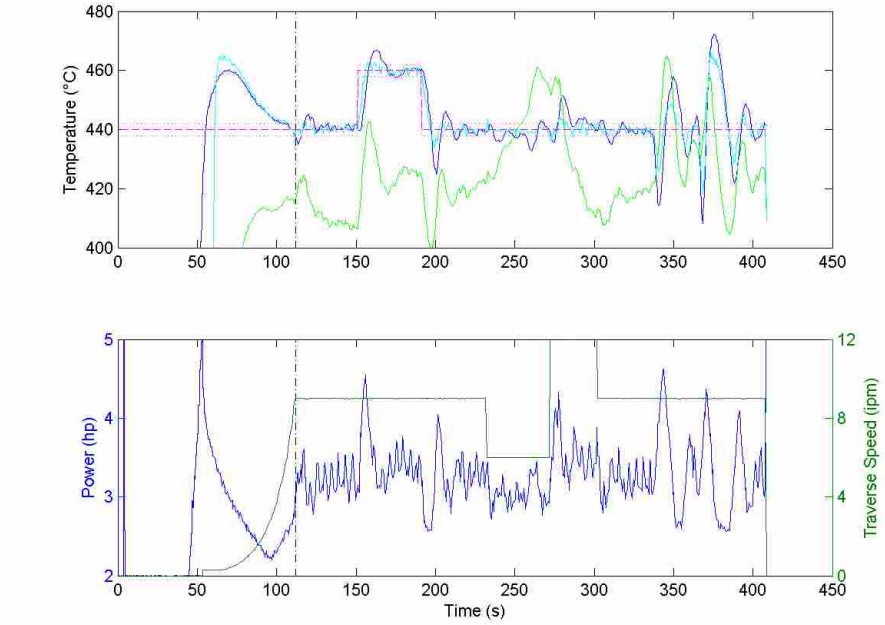



Figure B-11: Temperature and power data for the HS 003 weld.

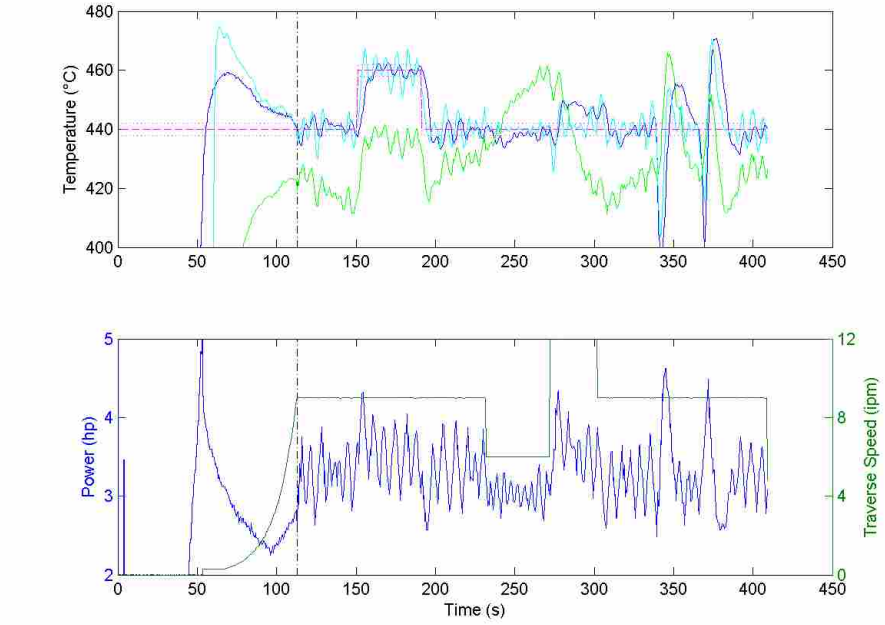



Figure B-12: Temperature and power data for the HS 004 weld.

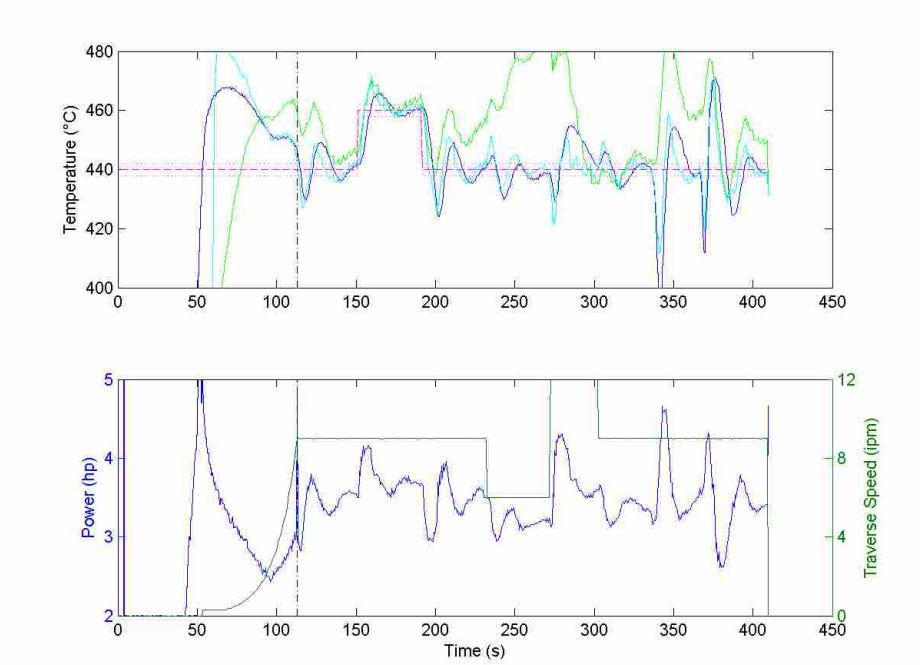



Figure B-13: Temperature and power data for the HS 005 v1 weld.

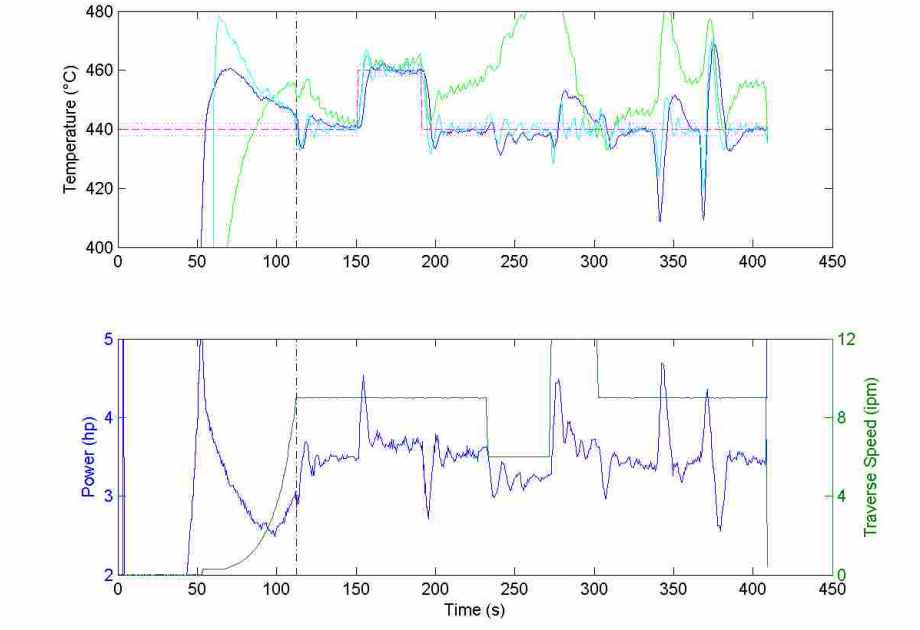



Figure B-14: Temperature and power data for the HS 005 v2 weld.

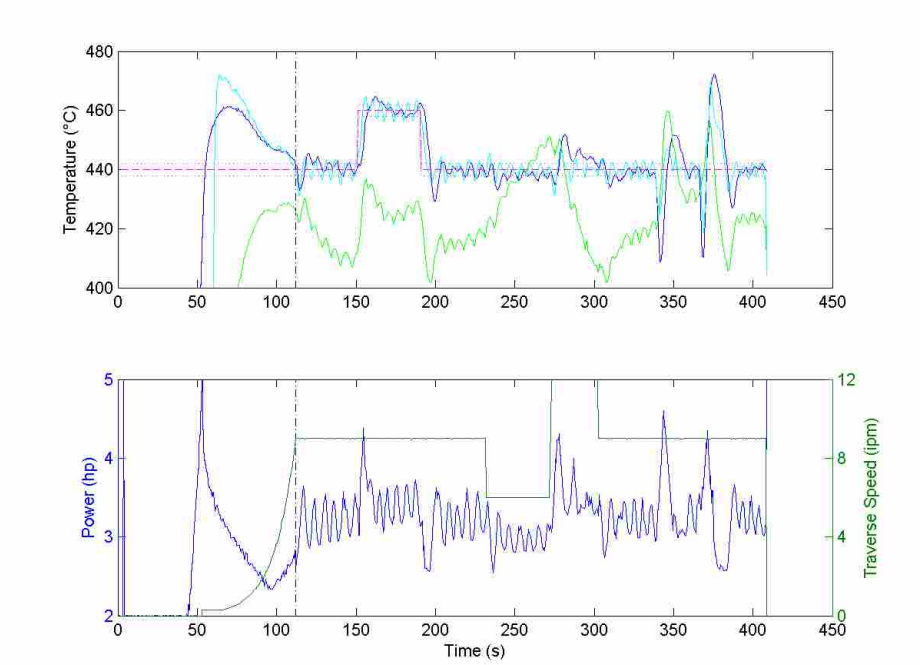



Figure B-15: Temperature and power data for the HS 005 v3 weld.

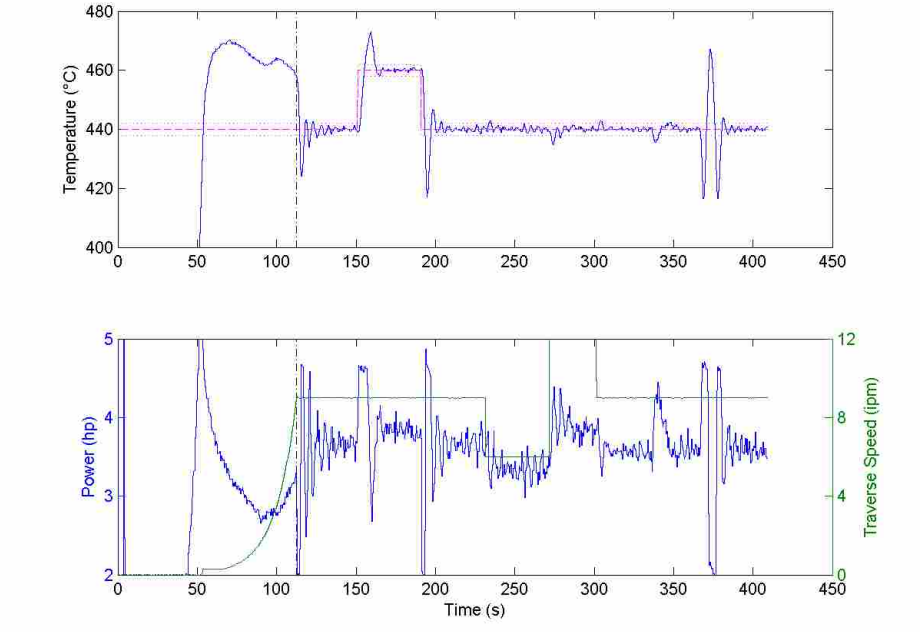



Figure B-16: Temperature and power data for the PID regulator weld.

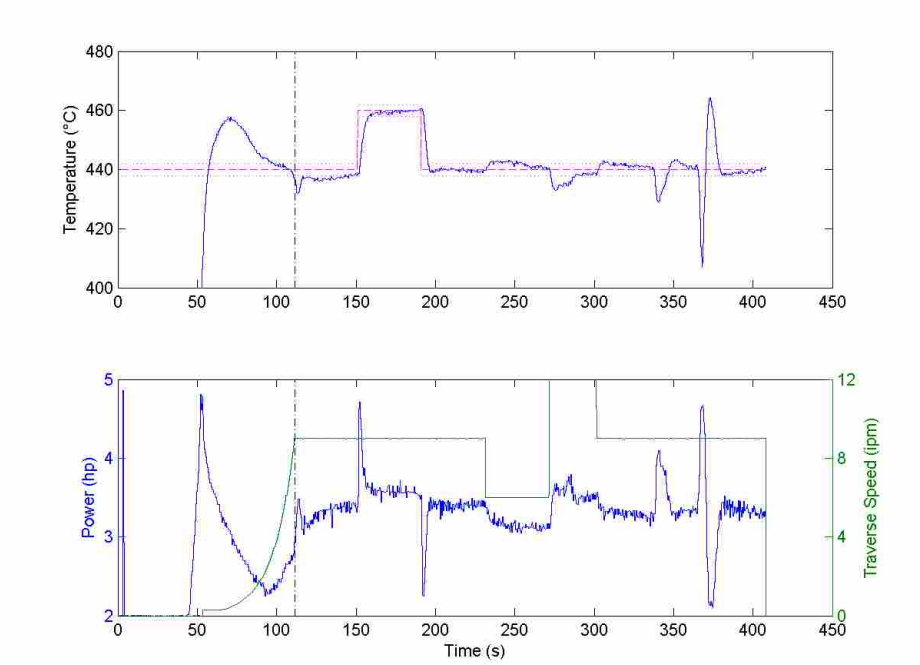



Figure B-17: Temperature and power data for the PID servo weld.

## B.3 Weld Data for the Second Round of Quasi Steady State Testing

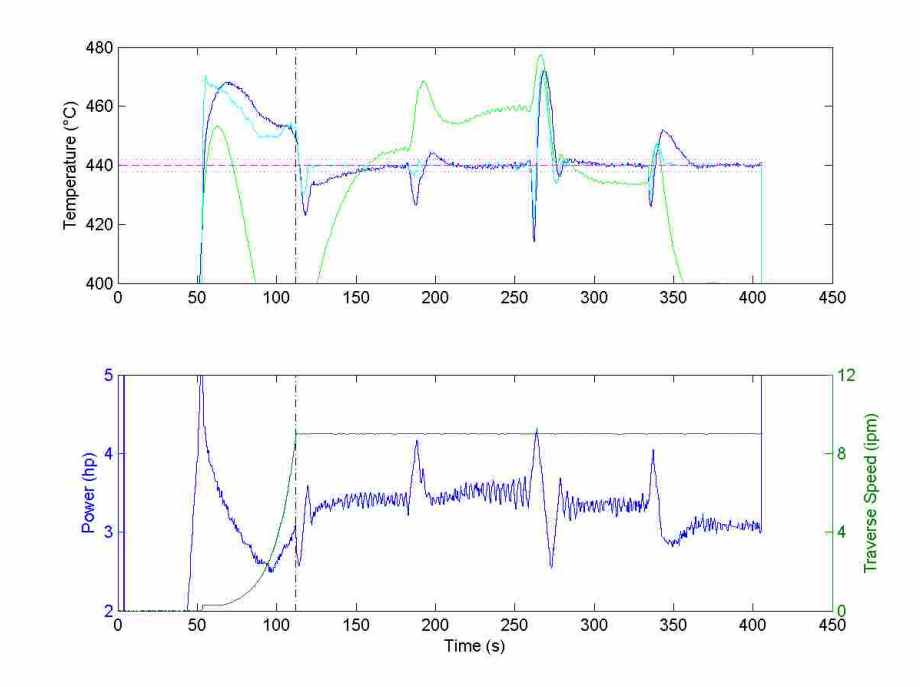



Figure B-18: Temperature and power data for the FOPDT 005 weld.

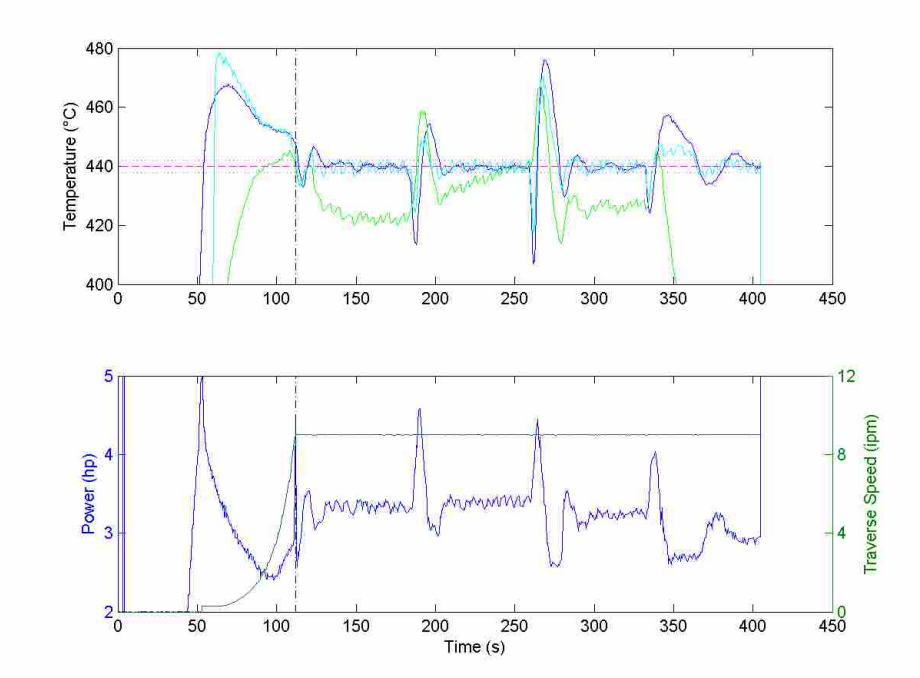



Figure B-19: Temperature and power data for the HS 007 weld.

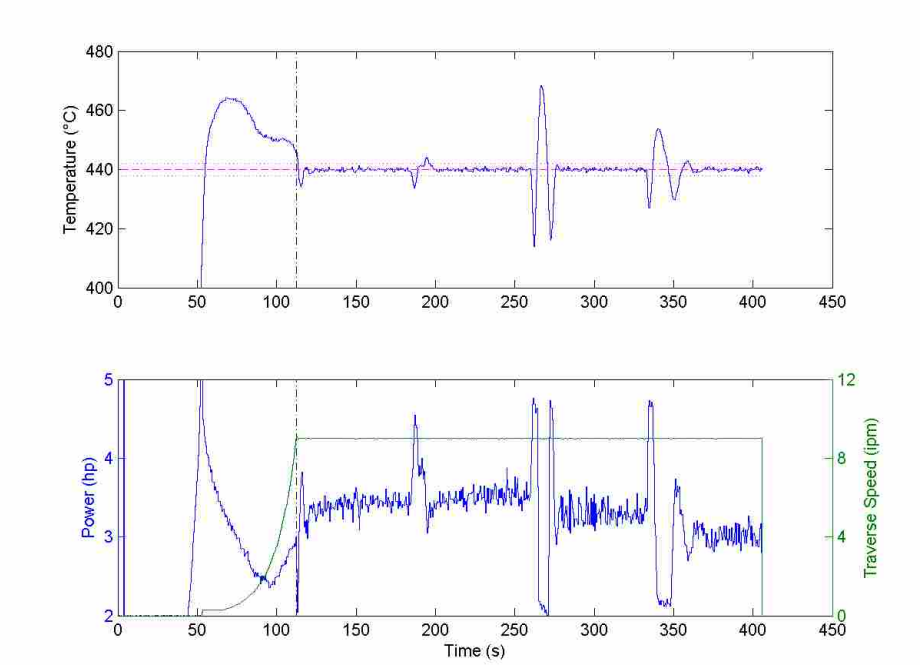



Figure B-20: Temperature and power data for the PID regulator (2) weld.

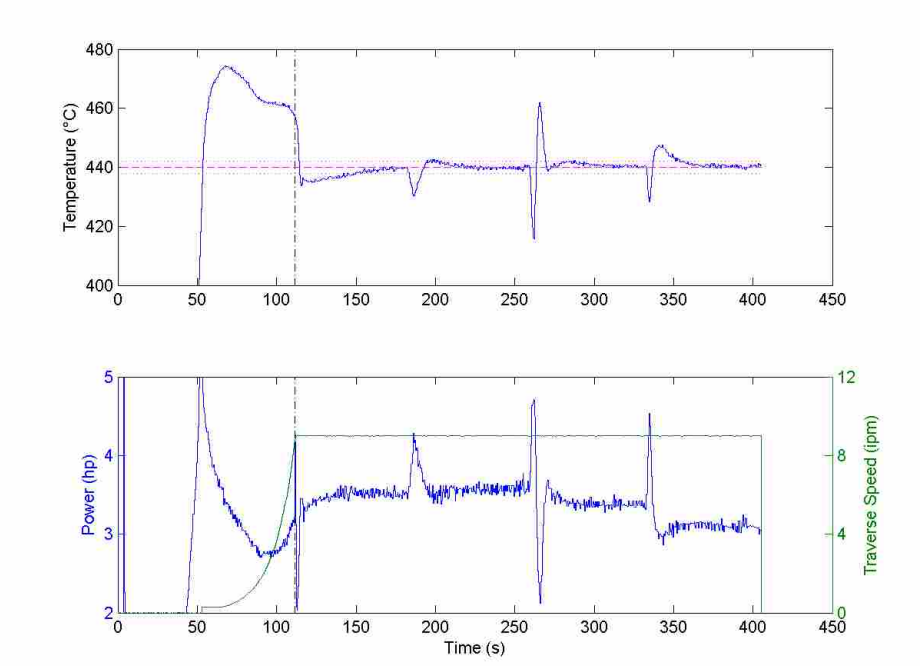



Figure B-21: Temperature and power data for the PID servo (2) weld.

## B.4 Weld Data for the First Round of Transient Testing

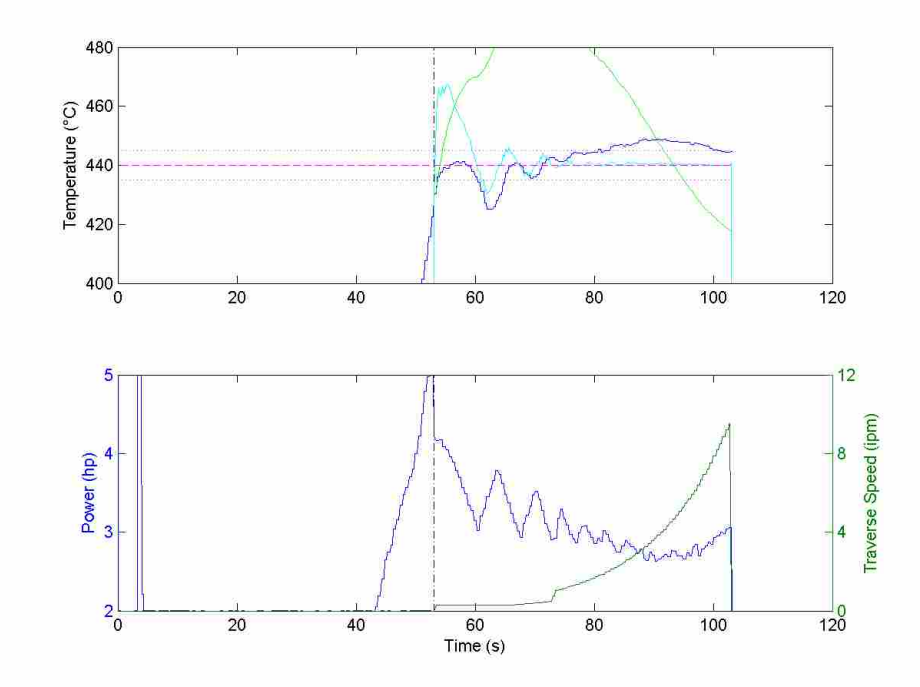



Figure B-22: Temperature and power data for the FOPDT 101 weld.

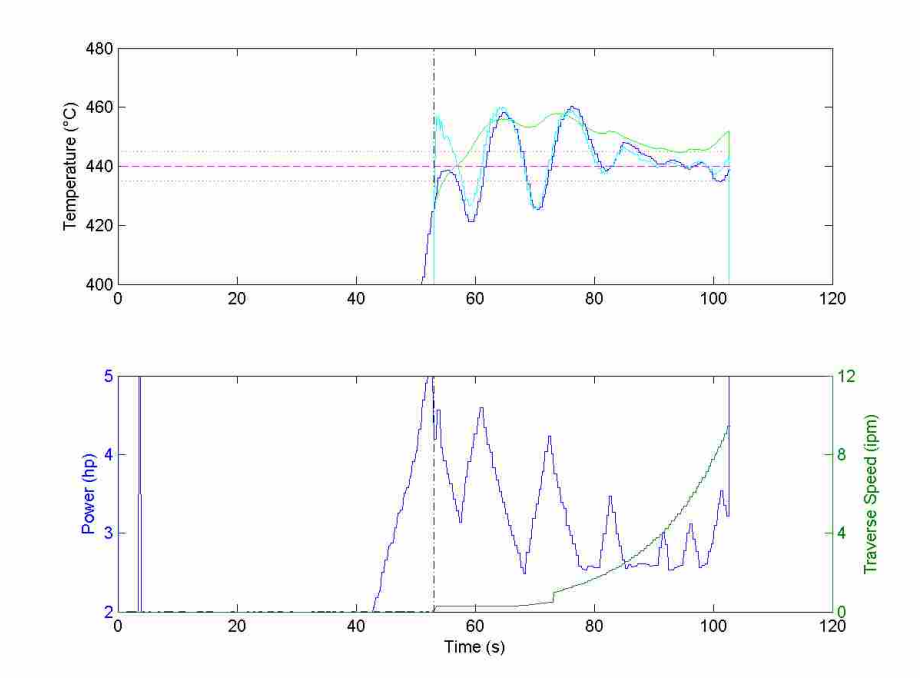



Figure B-23: Temperature and power data for the FOPDT 102 weld.

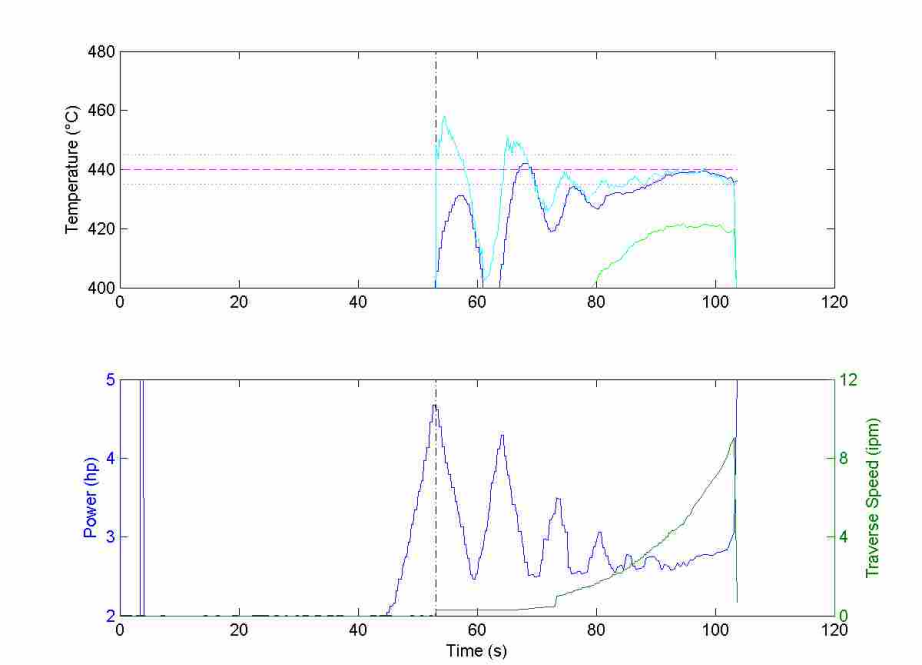



Figure B-24: Temperature and power data for the HS 101 weld.

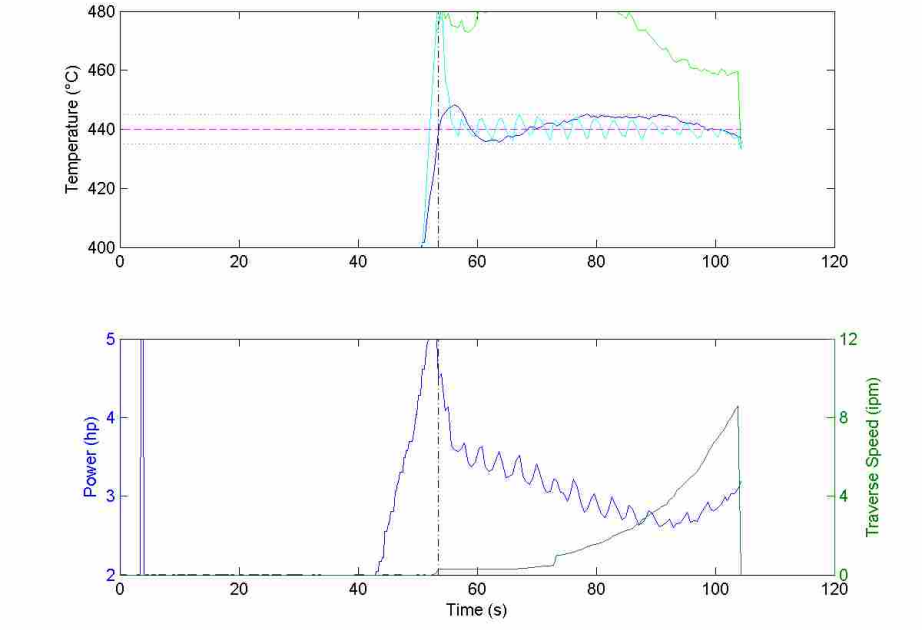



Figure B-25: Temperature and power data for the HS 102 weld.

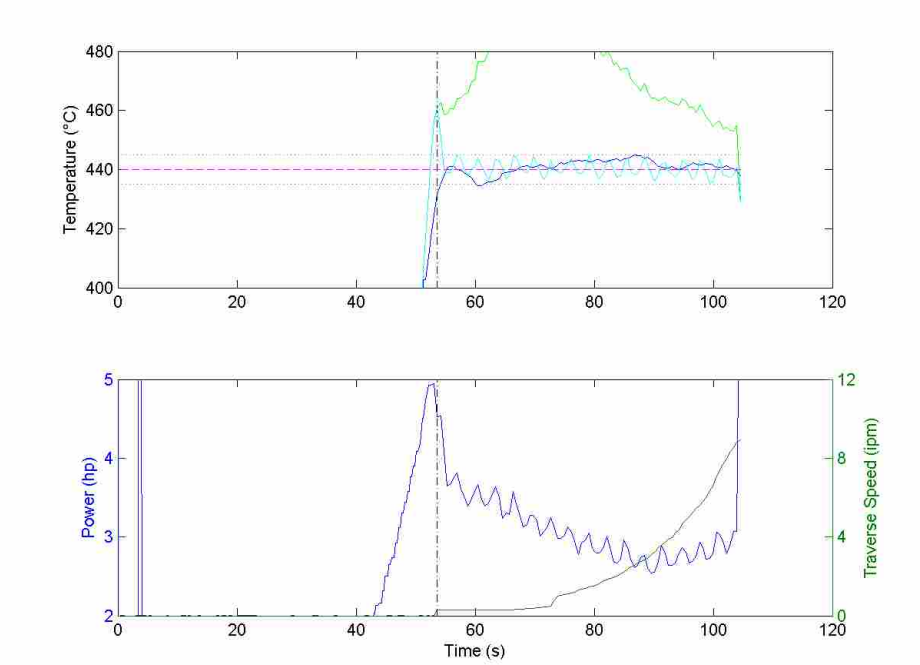



Figure B-26: Temperature and power data for the HS 102 v2 weld.

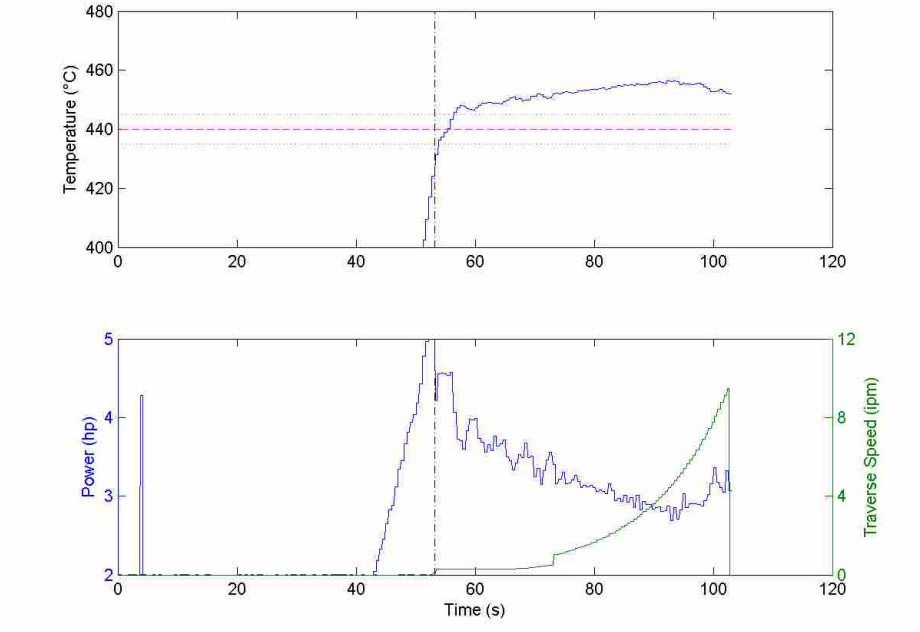



Figure B-27: Temperature and power data for the PD regulator weld.

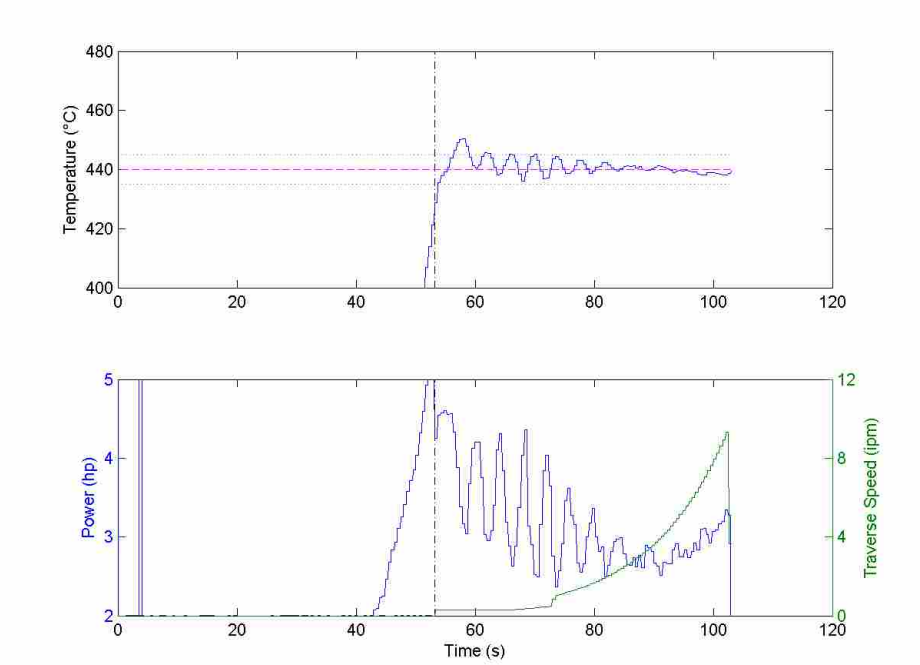



Figure B-28: Temperature and power data for the PID regulator (3) weld.

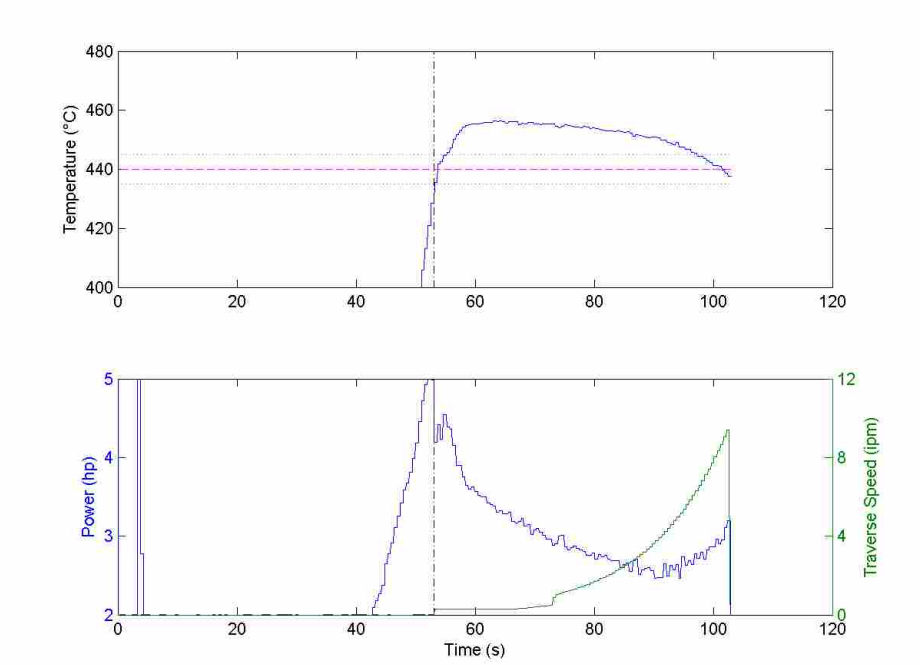



Figure B-29: Temperature and power data for the PID servo (3) weld.

## B.5 Weld Data for the Second Round of Transient State Testing

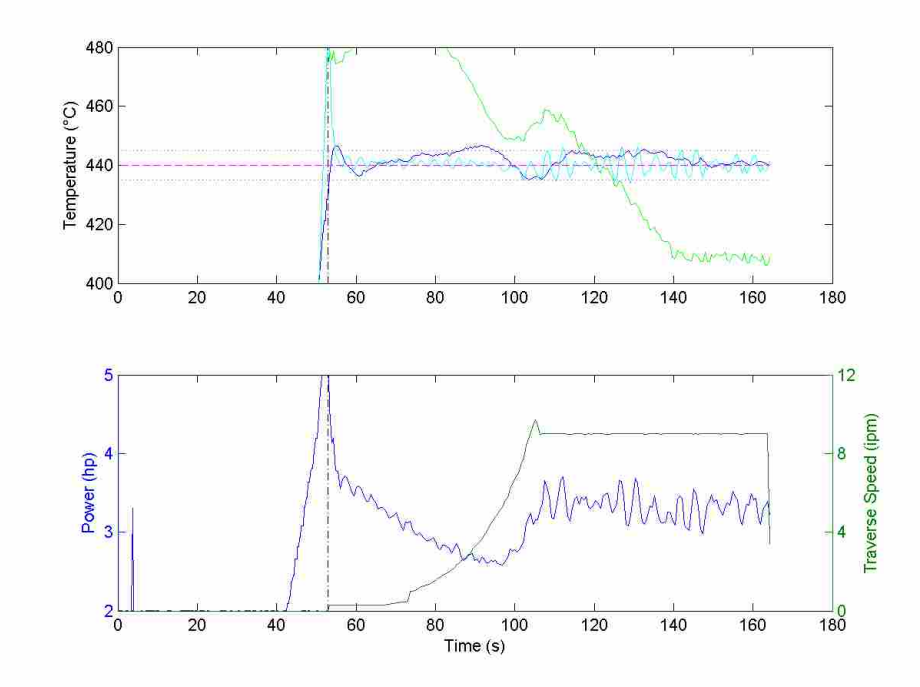



Figure B-30: Temperature and power data for the HS 103 weld.

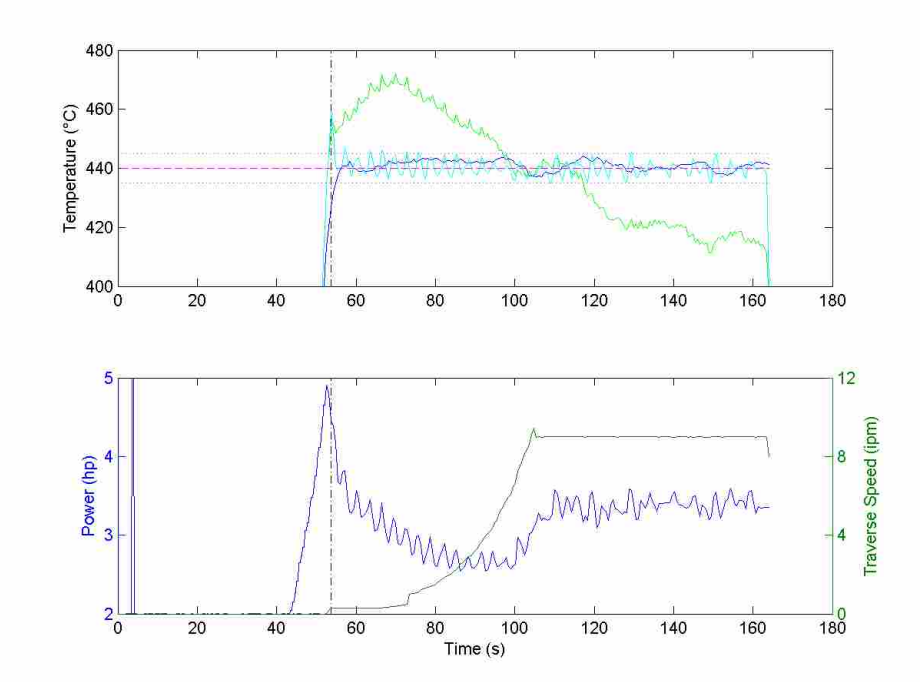



Figure B-31: Temperature and power data for the HS 103 v2 weld.

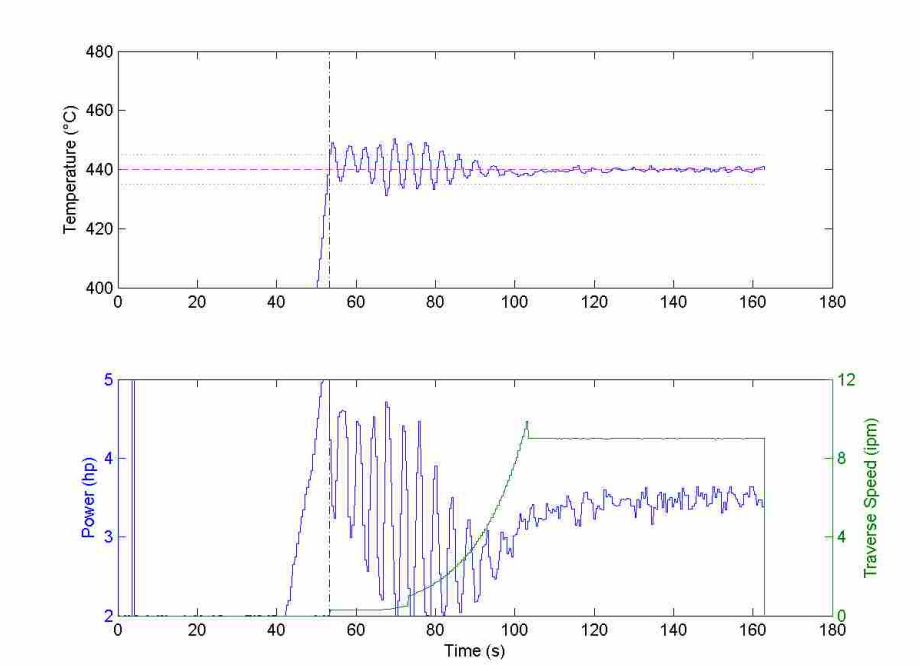



Figure B-32: Temperature and power data for the PID regulator (4.1) weld.

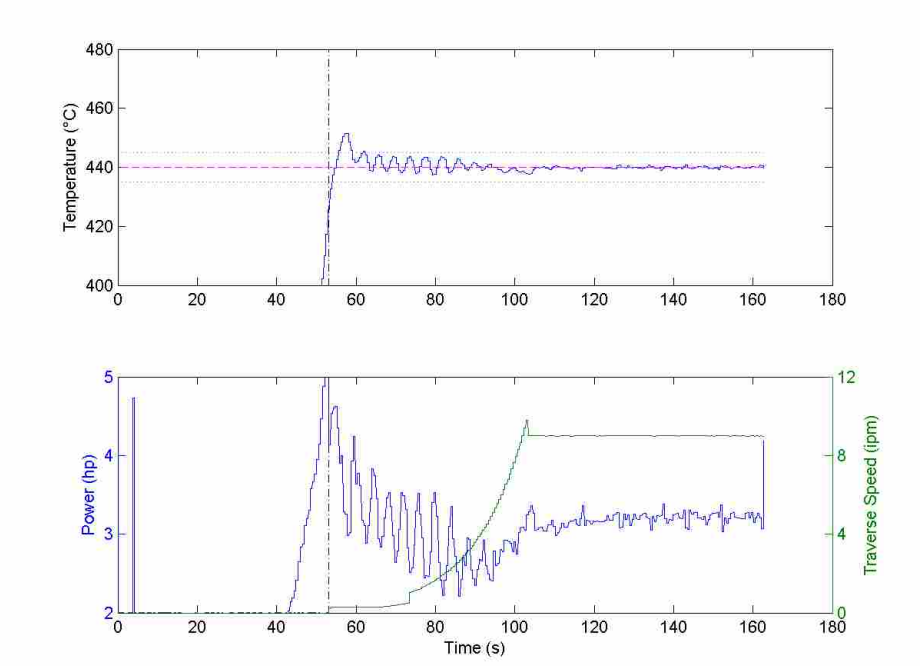



Figure B-33: Temperature and power data for the PID regulator (4.2) weld.

## B.6 Weld Data for PID Relay Tests

For these three figures, the dashed black lines indicate the period of analysis for the relay feedback tests. Sometimes the first or last few cycles need to be omitted because of irregular behavior. All tests were performed at 9 ipm. The 440 test was used for determining the PID parameters which were used in this thesis. The other two were simply used for determining the time constant of the system with respect to temperature.

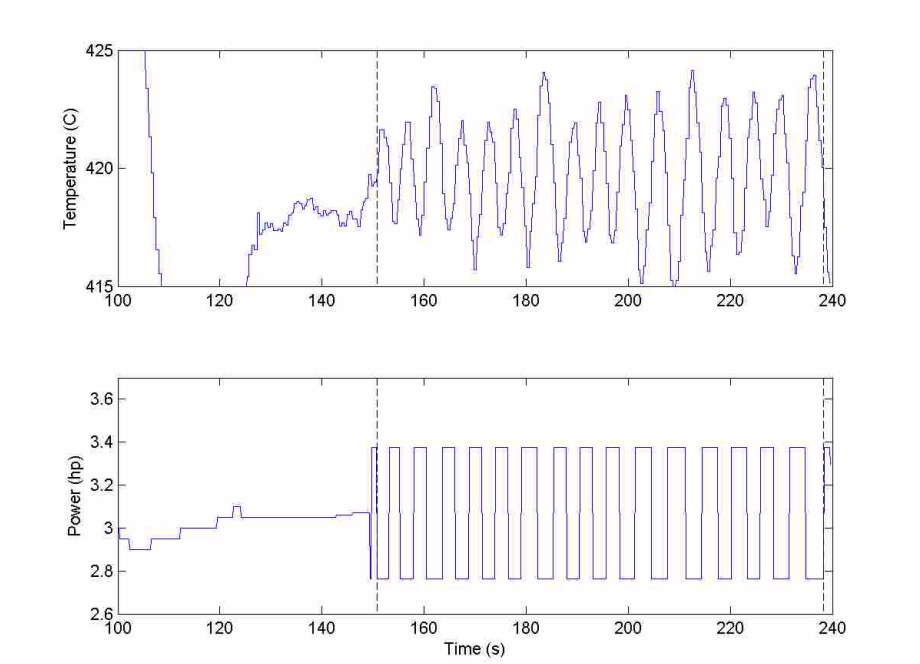



Figure B-34: Temperature and power data for the Relay 420 weld. Actual midpoint temperature was 419.8°C.

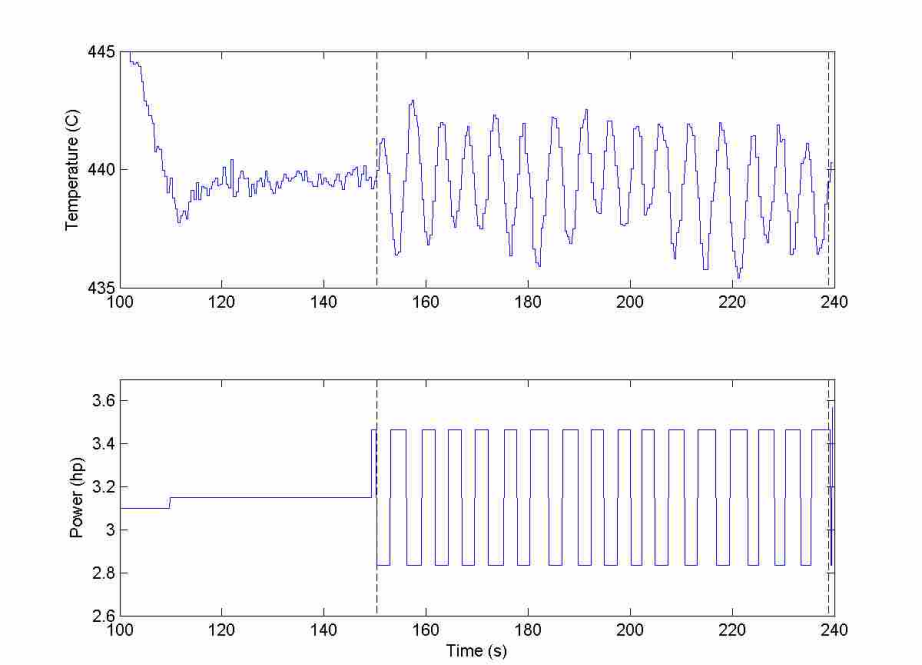



Figure B-35: Temperature and power data for the Relay 440 weld. Actual midpoint temperature was 439.7°C.

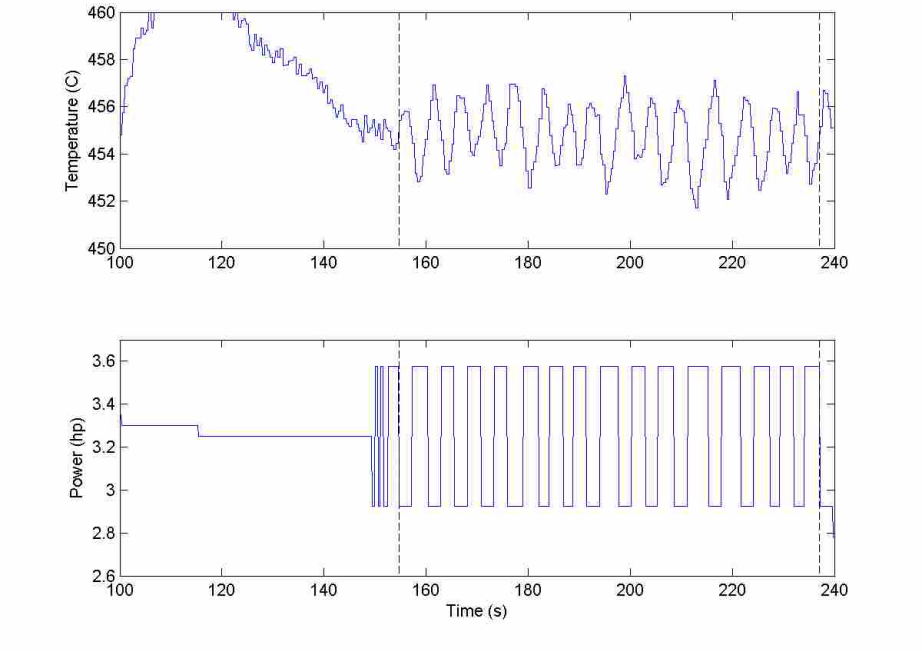



Figure B-36: Temperature and power data for the Relay 460 weld. Actual midpoint temperature was 455.1°C.

## B.7 Fit of PRBS Data Against FOPDT Model with Optimized Parameters

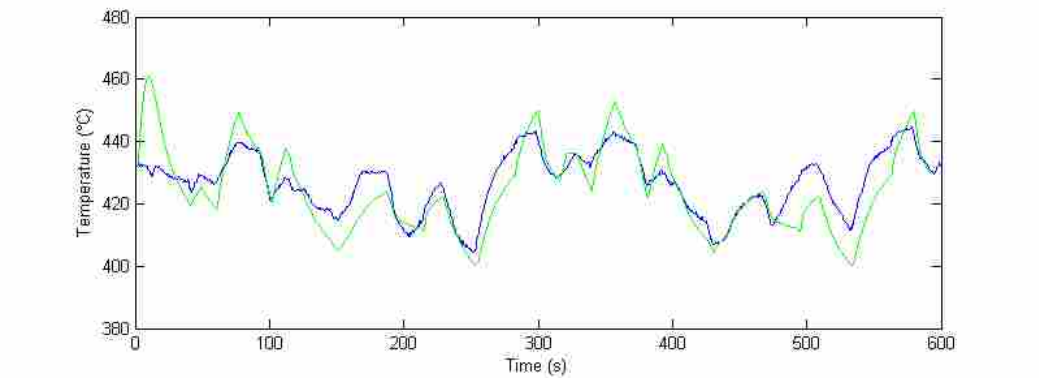



Figure B-37: Temperature vs. time data for the PRBS 002 weld with the predicted temperature based upon the FOPDT model with optimizer-determined parameters.

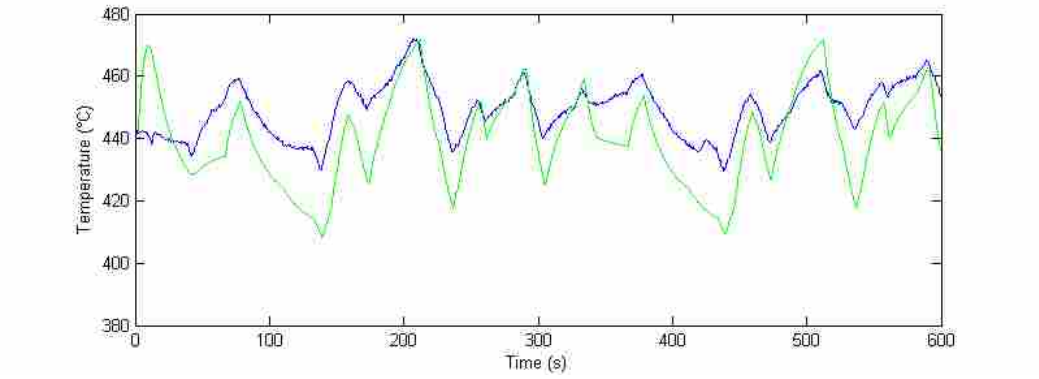



Figure B-38: Temperature vs. time data for the PRBS 003 weld with the predicted temperature based upon the FOPDT model with optimizer-determined parameters.

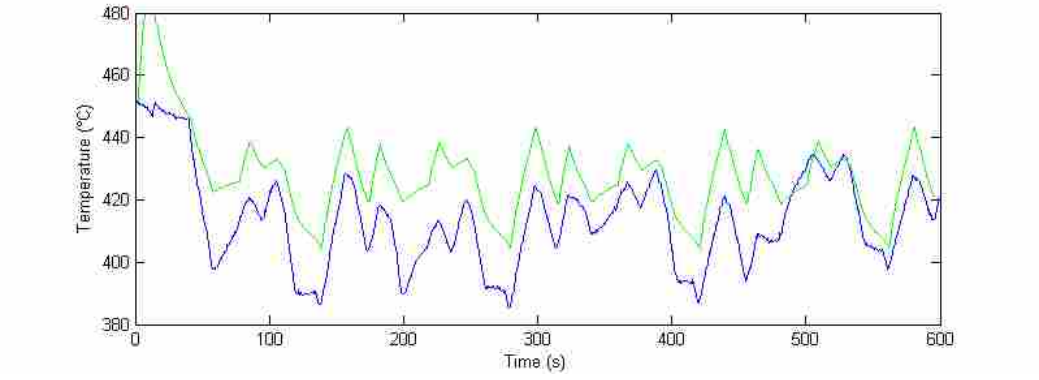



Figure B-39: Temperature vs. time data for the PRBS 004 weld with the predicted temperature based upon the FOPDT model with optimizer-determined parameters.

## B.8 Fit of PRBS Data Against FOPDT Model with Manual Parameters

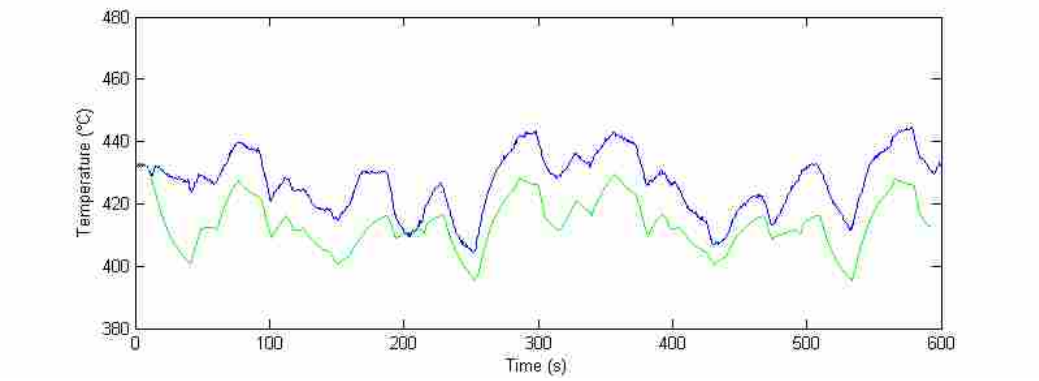



Figure B-40: Temperature vs. time data for the PRBS 002 weld with the predicted temperature based upon the FOPDT model with manually determined parameters.

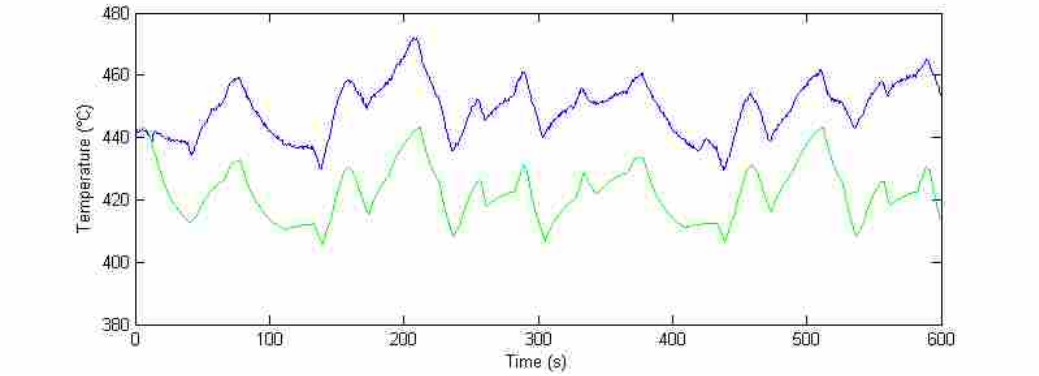



Figure B-41: Temperature vs. time data for the PRBS 003 weld with the predicted temperature based upon the FOPDT model with manually determined parameters.

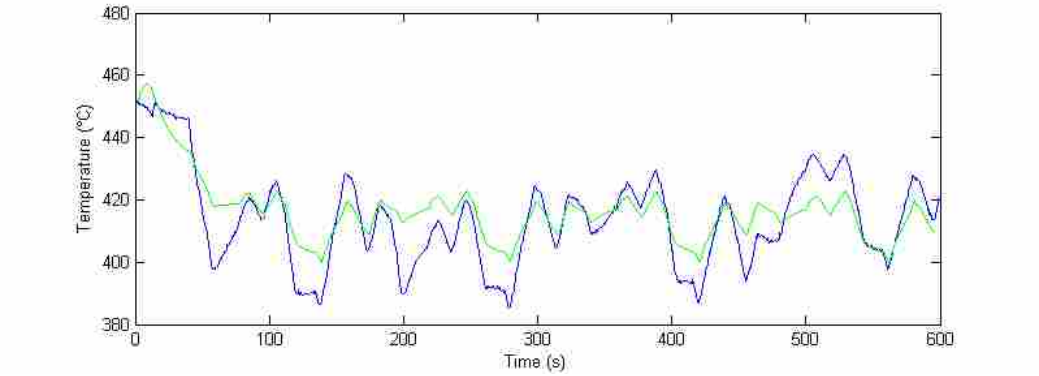



Figure B-42: Temperature vs. time data for the PRBS 004 weld with the predicted temperature based upon the FOPDT model with manually determined parameters.

## B.9 Fit of PRBS Data Against Hybrid Heat Source Model with Optimized Parameters

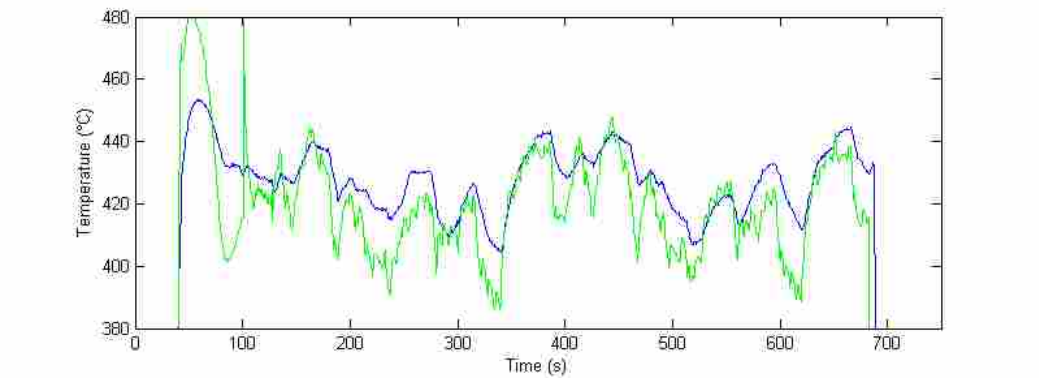



Figure B-43: Temperature vs. time data for the PRBS 002 weld with the predicted temperature based upon the Hybrid Heat Source model with optimizer-determined parameters.

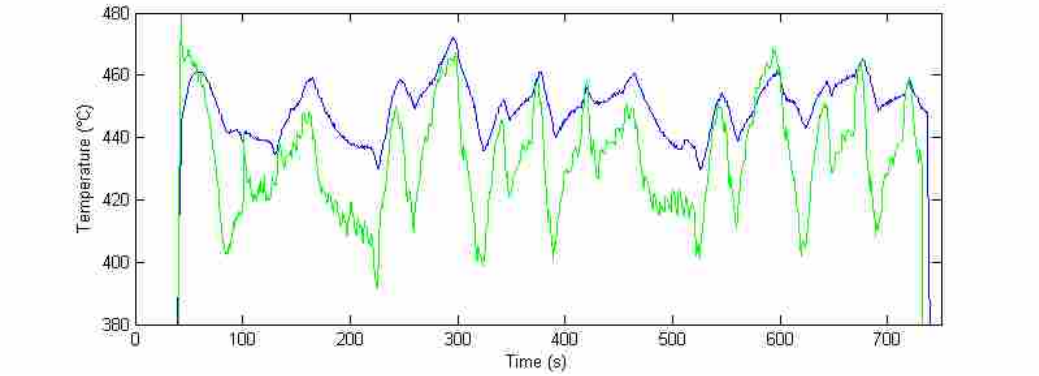



Figure B-44: Temperature vs. time data for the PRBS 003 weld with the predicted temperature based upon the Hybrid Heat Source model with optimizer-determined parameters.

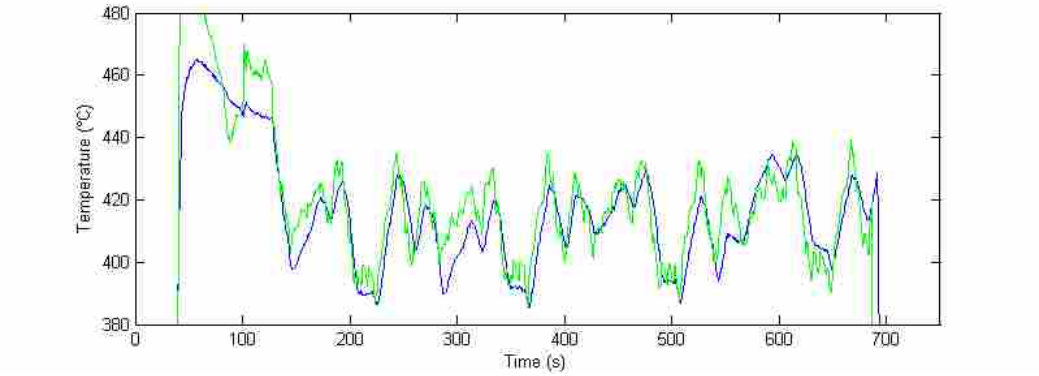



Figure B-45: Temperature vs. time data for the PRBS 004 weld with the predicted temperature based upon the Hybrid Heat Source model with optimizer-determined parameters.

## B.10 Fit of PRBS Data Against Hybrid Heat Source Model with Manual Parameters

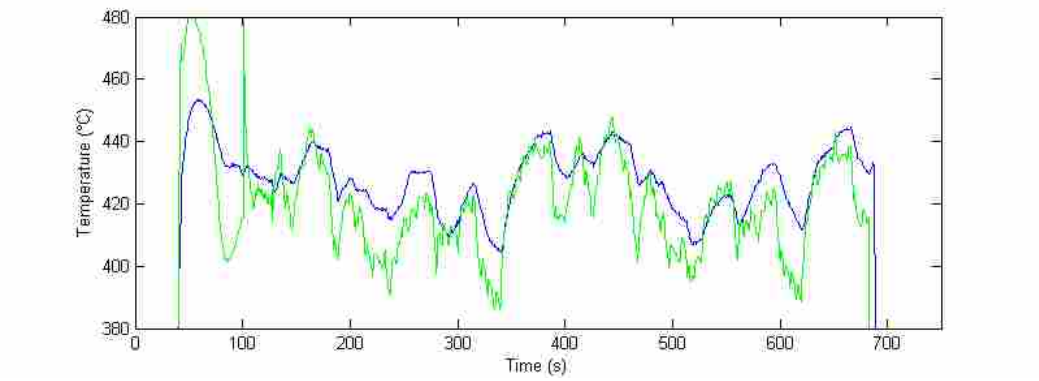



Figure B-46: Temperature vs. time data for the PRBS 002 weld with the predicted temperature based upon the Hybrid Heat Source model with manually determined parameters.

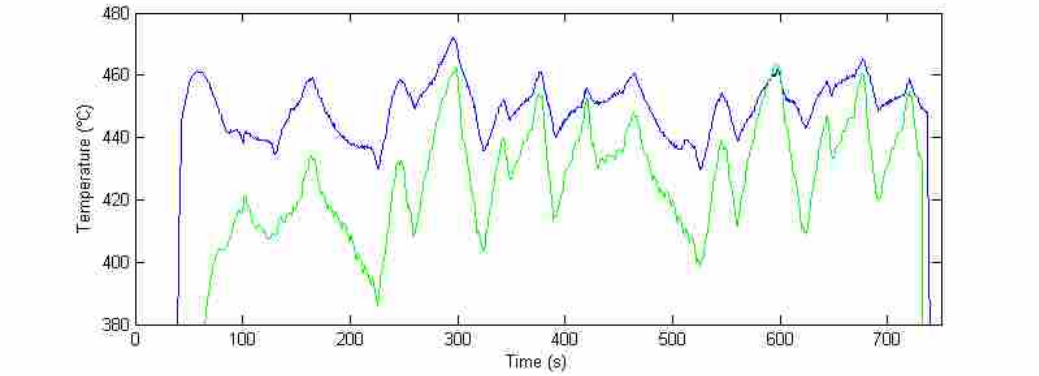



Figure B-47: Temperature vs. time data for the PRBS 003 weld with the predicted temperature based upon the Hybrid Heat Source model with manually determined parameters.

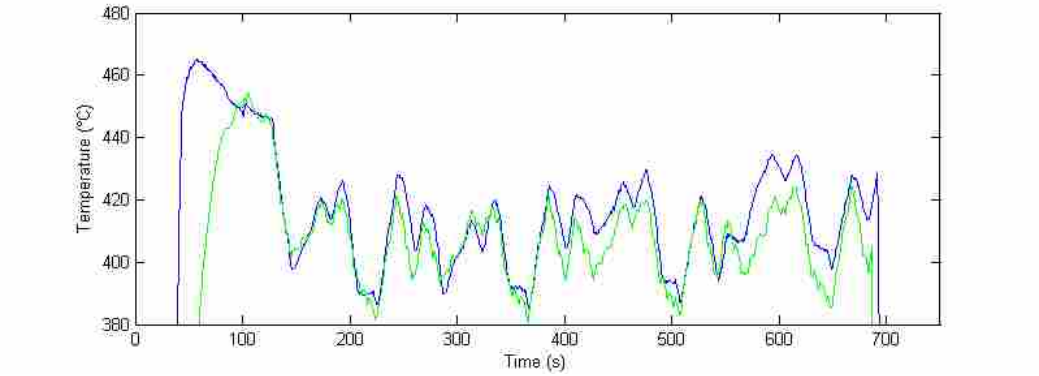



Figure B-48: Temperature vs. time data for the PRBS 004 weld with the predicted temperature based upon the Hybrid Heat Source model with manually determined parameters.

## Appendix C. Detailed Controller Performance Metrics, Analysis, and Discussion

This section covers all of the welds in this thesis to some extent or another. Rather than the final results based approach in Section 5, this appendix approaches the controller performance from the perspective of the chronological development of the controllers. As such, the rational for and precisely *how* the controllers were changed to get their performance is laid out explicitly.

The discussion and figures in this section are based upon the exact same metric definitions and data as is found in Section 5.2. Some of the text and several of the figures in this appendix appear in similar/exact form in the body of the thesis as well. To maintain readability of this appendix, instead of constant references back to the body, content is often duplicated here.

In the first round of quasi steady state testing, the controllers tried to keep the temperature constant over a series of both modeled and unmodeled disturbances (temperature setpoint changes, traverse speed step changes, and physical disturbances). Controller parameters and settings were changed so that the effects of these changes could be studied. This round had two primary goals: (1) to determine which controllers were generally better, and (2) to determine how the controllers reacted to both modeled unmodeled disturbances.

The second round of quasi steady state testing used fewer changes/disturbances to allow for a longer evaluation period for controller performance and stability per change. Two MPC

controllers and two PID controllers were compared to each other. This round used only unmodeled disturbances/changes as those are more common in applications of FSW.

The first round of transient testing was done to determine controller performance immediately post plunge and as the weld increased in traverse speed. The controllers from the second round of quasi steady state testing were used and some additional modifications were done to those controllers to make them more suitable for the initial transient of the weld. This round of testing was performed primarily to determine which controllers were suitable for control during the initial transient part of the weld.

The second round of transient testing used the best controllers from the first round of transient testing. The process was allowed to run longer than in the first round so that the long-term behavior of the controllers could be observed during the initial weld transient.

## C.1 Initial Round of Quasi steady State Testing

All welds performed in the initial round of testing encountered the same changes at the points in time and space for each weld. These are shown graphically in Figure C-1 with the details of the events and weld segments given in Table C-1. Several different kinds and magnitudes of changes were performed to allow a more comprehensive testing of the controllers. This was done as it is possible that controller A may perform well in one scenario, but in another scenario controller B may greatly outperform controller A.

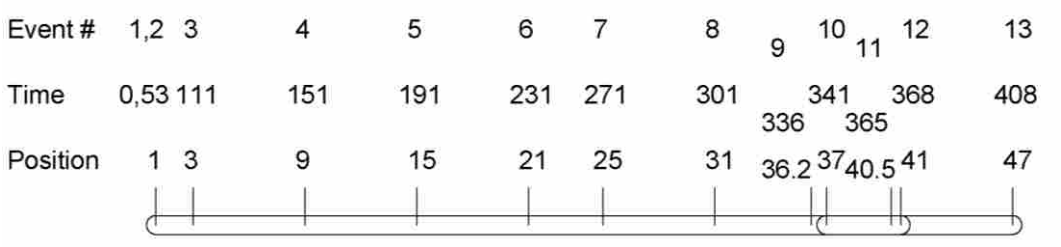



Figure C-1: The standard setpoint and disturbance sequence used for the initial round of controller testing.

Table C-1: Details of the Standard Disturbance Weld Sequence Used for the Initial Round of Testing.

| Event/ Segment # | Starting Time (s) | Duration (s) | Starting Position (in) | Traverse speed (ipm) | Temp Setpoint (C) | Description |
|---|---|---|---|---|---|---|
| 1 | 0 | 53 | 1 | 0 | n/a | Plunge |
| 2 | 53 | 58 | 1 | var | n/a | Velocity ramp from 0 to 9 ipm |
| 3 | 111 | 40 | 3 | 9 | 440 | Temperature control engaged |
| 4 | 151 | 40 | 9 | 9 | 460 | Temperature setpoint raised to 460°C |
| 5 | 191 | 40 | 15 | 9 | 440 | Temperature setpoint lowered to 440°C |
| 6 | 231 | 40 | 21 | 6 | 440 | Traverse speed changed to 6 ipm |
| 7 | 271 | 40 | 25 | 12 | 440 | Traverse speed changed to 12 ipm |
| 8 | 301 | (35) | 31 | 9 | 440 | Traverse speed changed to 9 ipm |
| 9 | 336 | (5) | 36.2 | 9 | 440 | Tool begins to contact flash of previous weld |
| 10 | 341 | (44) | 37 | 9 | 440 | Location of previous weld's plunge |
| 11 | 365 | (3) | 40.5 | 9 | 440 | Tool begins to interact with previous pin exit hole |
| 12 | 368 | 40 | 41 | 9 | 440 | Location of previous weld's extract |
| 13 | 408 | n/a | 47 | n/a | n/a | Extract of weld from plate |

First, the plunge and initial segment of the weld was designed so that the temperature would be higher than desired upon controller takeover. This allowed observation of the controller as it initially brought the weld under control during an extremely transient environment. As

expected, many of the controllers had a hard time keeping temperature during a continuously changing transient.

Second, the temperature setpoint was changed up and then back down. This was done twice so that both up and down behavior could be observed.

Third, the traverse speed of the weld was changed three times to allow observation of changes both up and down and and of a medium and large magnitude.

Lastly, the weld was run over an existing weld. At the first of these two, the welder encountered the previous welding flash and had to accommodate to different properties of the already-welded segment of the plate. At the end, the welder encountered a pin hole that was left from the previous weld. This disturbance had large and opposite impacts on temperature as the tool approached and exited the pinhole location.

### C.1.1 Effect of Parameters

Several parameters, in the controllers were varied for two primary reasons. First, in order to determine what factors have an impact on the various MPC controllers. Second, in order to find good versions of both MPC controllers. These welds were run against the same exact disturbance pattern as described in the preceding section and evaluated based upon the same metrics as the original models. The plots of their power and temperature vs. time can be found in Appendix B.2.

#### C.1.1.1 Effect of Parameters on FOPDT MPC Controllers

For the FOPDT MPC controllers, (1) the model parameters were changed, and (2) with these new model parameters the time constant was changed to a function of temperature.

#### C.1.1.1.1 Changing Model Parameters

Parameters were initially determined by optimization. As detailed in Section 4.3.3, there are several reasons that an optimizer's model parameters may not in fact be the best for a controller.

In order to correct this, parameters were first re-determined via optimization which included the PRBS welds and the data from the first FOPDT weld, FO 001. Then, the parameters were manually altered further to better match certain – even at the expense of temperature offset. These new parameters are reported below.

Table C-2: Original and New Manual Parameters for the FOPDT Model

| | $c_1$ (hp/°C) | $c_2$ (m/s°C) | $c_3$ ($m^2/s^2$°C) | $c_4$ (°C) | $\tau$ (s) |
|---|---|---|---|---|---|
| Original | 91.02 | -54.27 | 2.567 | 18.70 | 18.19 |
| Manual | 118.48 | -7.039 | -.0967 | 4.918 | 16.39 |

A weld was run with the new parameters. The performance metrics, averaged for each of the eight segments, are shown below in Figure C-2.

As can be seen in Figure C-2, every single metric improved and the original controller is dominated by the new controller. It is particularly notable how the average oscillations per segment for the optimization determined parameters was five times worse than manually determined parameters – 15/16 vs. 3/16. Manually changing the parameters greatly helped the controller performance. It is likely that they could be further refined and optimized for a specific situation.

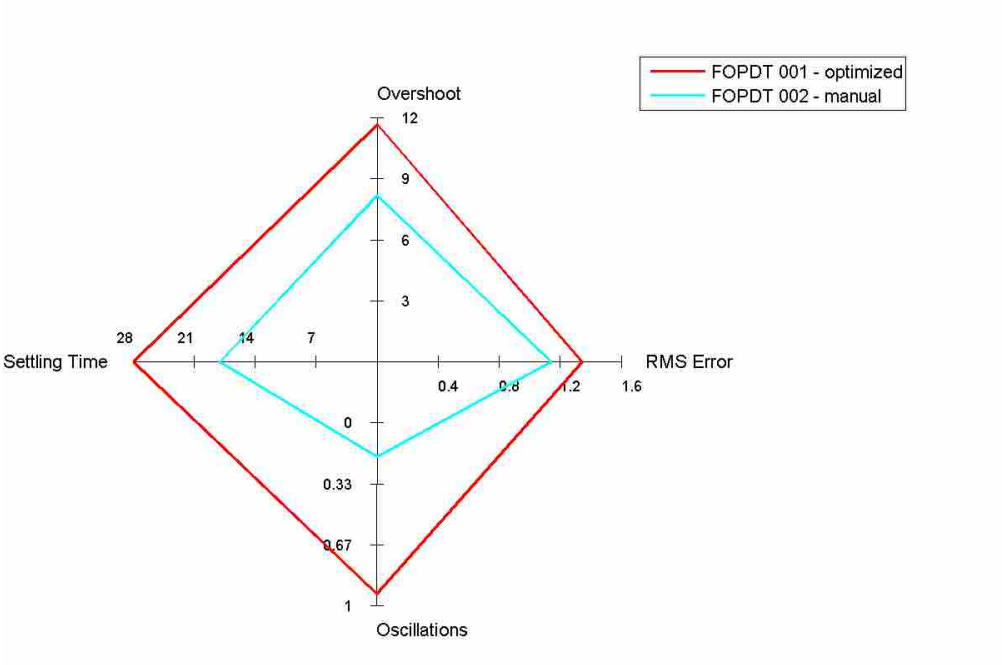



Figure C-2: Effect of parameters determined by optimization and manual fitting on the FOPDT MPC controller

#### C.1.1.1.2 Time Constant as a Function of Temperature

Previously, data by Marshall had suggested that the time constant of the FSW system increased with temperature. One possible physical explanation is that as the temperature lowers, less material is soft enough to be caught up in the stir zone. Based upon the FOPDT model presented in Equation (2-1) and then (2-2), the time constant of the system is the mass of the stir zone times the heat capacitance. Thus, if the stir zone is smaller, the time constant should be lower; the converse is also true.

If in fact the time constant of the system is variable but is modeled as a constant value, then controller performance could be negatively affected. As the controller attempts to raise the temperature, it predicts that its actions will take effect quicker than they actually do and the controller will be more sluggish than desired. Conversely, as it attempts to lower the

temperature, it will predict that the actions will have less of an effect than they do in reality and the controller will be too aggressive.

The time constants were by the relay method of system identification as explained in Section 4.4. These numbers were fit to a linear profile based upon the actual midpoint temperature of the relay test. The equation for the fit is given below:

$$\tau(T) = -169 + .4369\,T \qquad \text{(C-1)}$$

where the time constant is in seconds, and temperature in degrees Celsius. The actual data and fit from Equation (C-1) is shown below in Figure C-3.

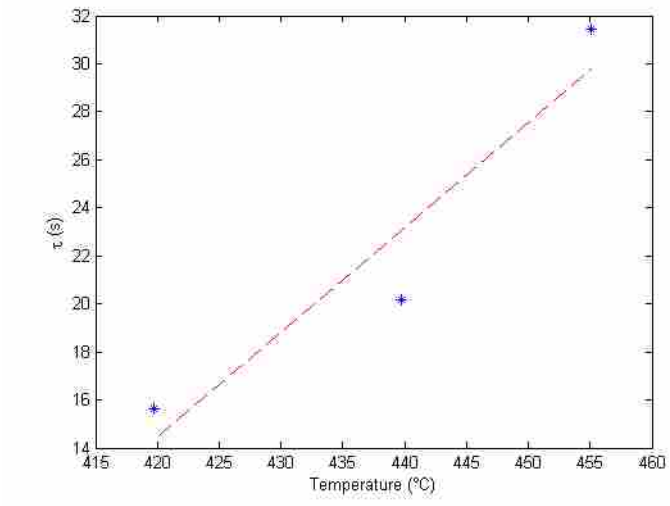



Figure C-3: The time constant vs. temperature of the system, determined from relay test data. The linear best fit line is also shown.

For implementation, upper and lower bounds were put on the variable time constant of 10s and 40s. This was done for stability since below 387°C the time constant would be predicted to be negative based upon (C-1), and it is particularly important that the time constant is a positive value. While a linear fit may not be the optimal fit, and while velocity or other factors may also affect the time constant, the purpose of this experiment was to investigate the effect of

a variable time constant as it might apply to control theory, and not to determine a robust model that is extremely accurate in all conceivable circumstances.

A weld was run with the same manual parameters as before, except that the time constant is variable as described by (C-1). The results are shown below in Figure C-4.

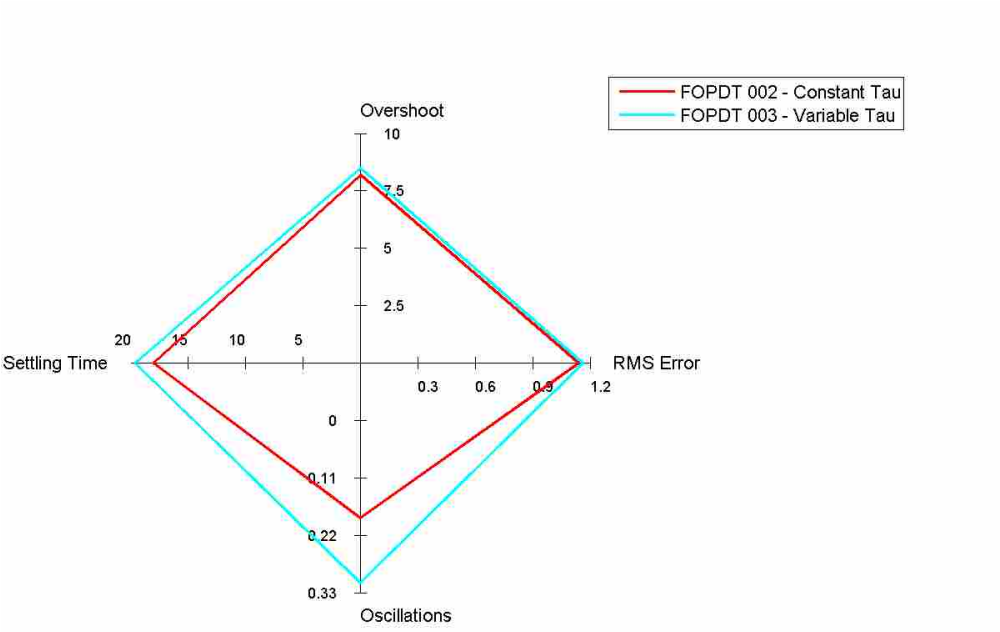



Figure C-4: Effect of making the time constant a function of temperature.

As can be seen in Figure C-4, adding a variable tau slightly decreased the performance of the controller in all circumstances. The variable time constant controller is thus a barely dominated design, both in the four calculated metrics and in model complexity. It is unknown why the variable time constant controller performed worse. Considering the new model was very close in most metrics, it is possible that a better formulation of the variable time constant may do better. Further work would be needed to explore the possible benefits and how/if this feature can improve controller performance.

### C.1.1.2 Effect of Parameters on Hybrid Heat Source MPC Controllers

Welds were performed to evaluate the effects of changing model parameters, adding a time delay, and adding a MV penalty term to the Hybrid Heat Source controller.

#### C.1.1.2.1 Changing Model Parameters

As discussed above in Sections 4.3.3 and C.1.1.1.1, there may be theoretical and practical benefits resulting from changing parameters from those determined by an optimizer. This was also shown to have a positive benefit in the case of the FOPDT controller. Model parameters were also changed for the Hybrid Heat Source model, and several of the parameters such as the point depth and power multiplication factor were also brought closer to their theoretical values. The optimizer and manually determined values are shown below in Table C-3.

Table C-3: Original and New Parameters for the Hybrid Heat Source Model.

| | $z_{pt}$ (inches) | $h_0$ (W/m$^2$ K) | $p_{mult}$ (unitless) |
|---|---|---|---|
| Original | .0428 | 4613 | .6011 |
| Manual | .18 | 600 | 1.4 |

These values were then used in the MPC controller. The results of this are shown in Figure C-5. As can be seen, using manually determined parameters improved the performance of all metrics. The controller with parameters strictly determined via optimization is a dominated design.

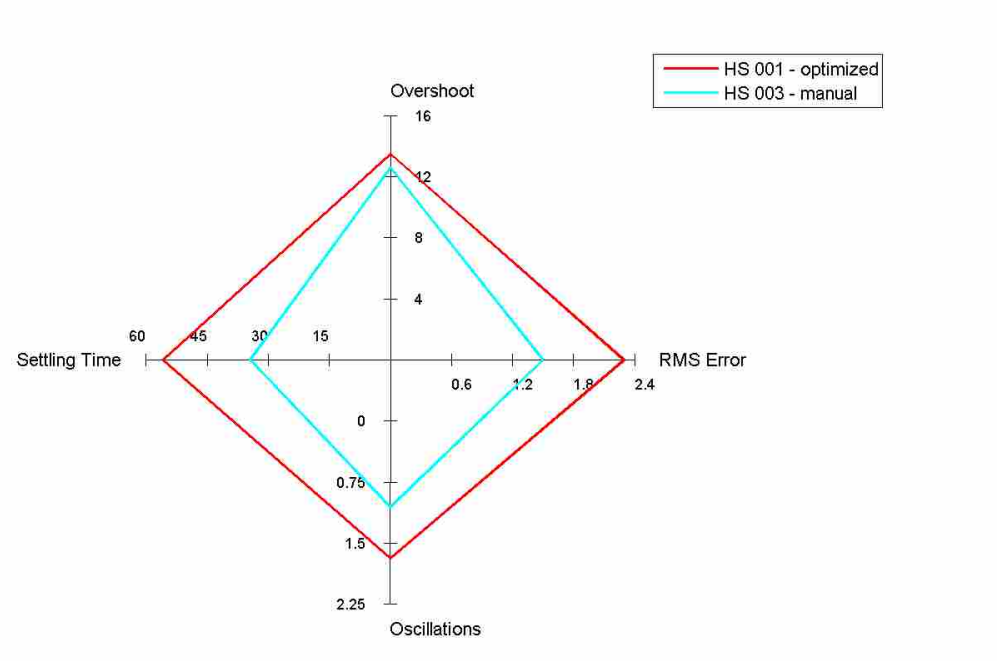



Figure C-5: Effect of parameters determined by optimization vs. manual fitting on the Hybrid Heat Source MPC controllers' performance

#### C.1.1.2.2 Model Time Delay

It was noticed that the time delay of actual temperature response was not captured well by the stock Hybrid Heat Source model. The actual temperature response had a time delay of approximately 1-2 seconds, whereas the model predicted a nearly zero time delay. While the Hybrid Heat Source model has parameters that can effectively constitute a time delay, they are coupled with other model characteristics and a time delay is not explicit in the Hybrid Heat Source formulation.

If a system has a non-inconsequential time delay, adding a time delay to the model can theoretically help when the controller is approaching a setpoint. Without a modeled time delay, the controller will try to keep changing the MV higher or lower until CV is at the setpoint. However, even with perfect gains, due to the time delay the CV will continue increasing/decreasing past the setpoint despite the throttling of the MV. With a time delay in the

model, the controller can better predict when the CV will reach the setpoint within the near future – even though the CV feedback has not yet indicated as such. It can then start to throttle the MV before the setpoint is reached, and thus prevent or mitigate overshoot and therefore improve controller performance.

A variant of Hybrid Heat Source model was developed that had a time delay. This was set conservatively at 1 second. The same manually determined constants as presented in Table C-3 were used for this controller. The effects on the overshoot at each of the eight different weld segments for the controller with and without the time delay are shown below in Figure C-6.

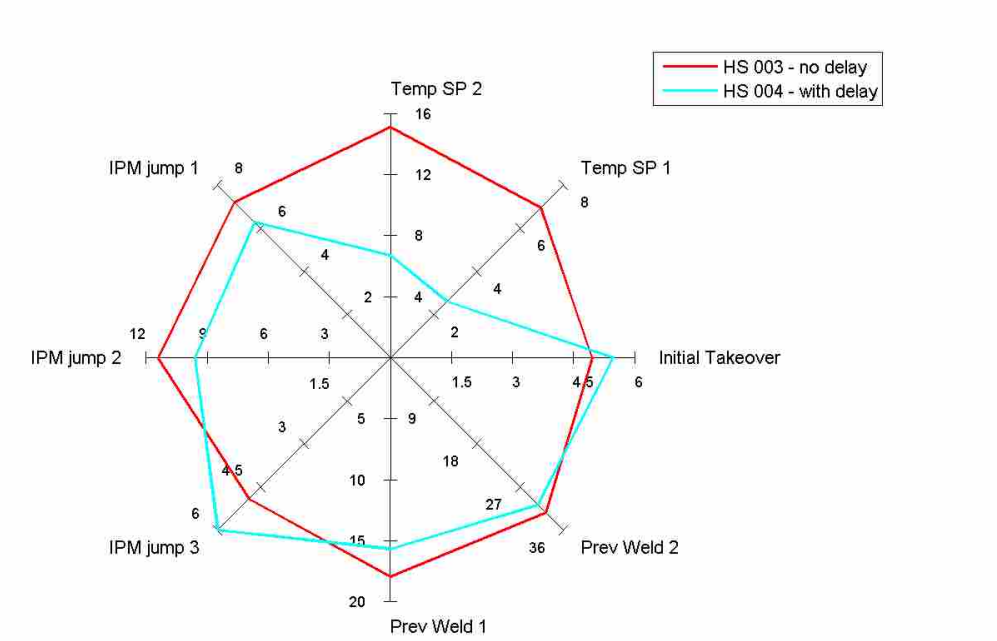



Figure C-6: Effect of adding a time delay on the overshoot of the weld.

In most of the cases, the overshoot of the weld was reduced as expected. However, this did not necessarily translate into improved performance in other categories as seen in Figure C-7.

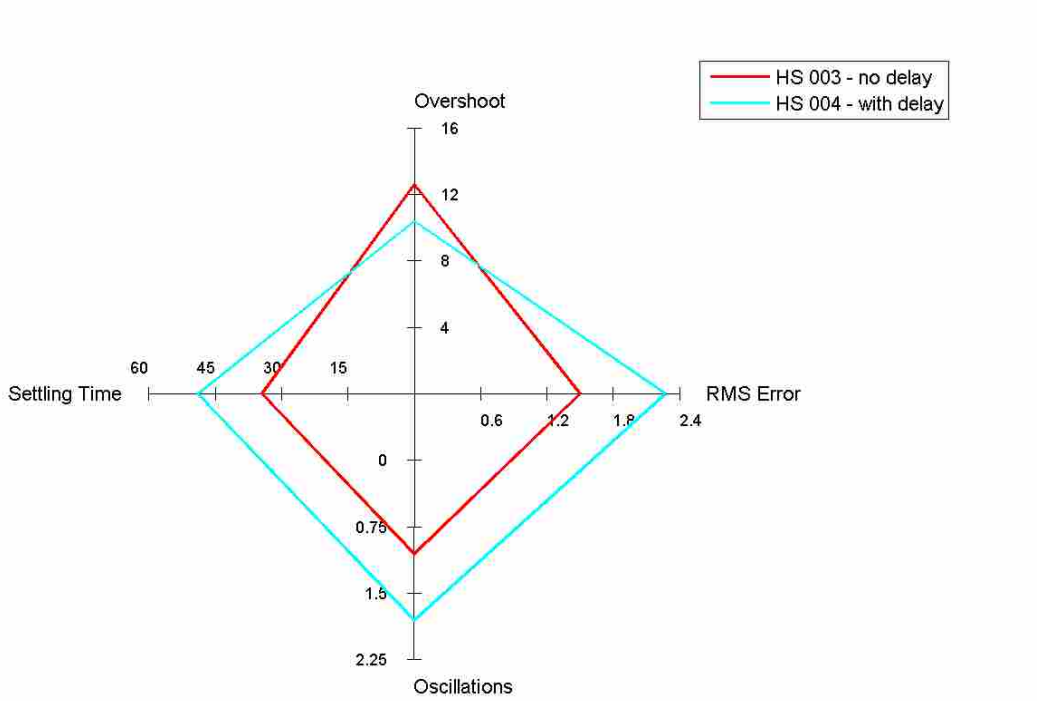



Figure C-7: Effect of adding a time delay on the performance of the Hybrid Heat Source MPC controller.

It is believed that the reason that this created more oscillations (which led to a longer settling time and higher RMS error) is that because the process model was still inaccurate, the controller was overreacting and ended up pushing the temperature beyond the setpoint, then overcontrolled and pushed the temperature the other way, etc. However, the controller was approximately correct in the level of power necessary for a given temperature, but it was slightly too aggressive. The oscillatory/saw-toothed nature of the MV and oscillatory behavior of the CV as shown in Figure C-8 supports the observation that the controller is overcompensating and is too aggressive.

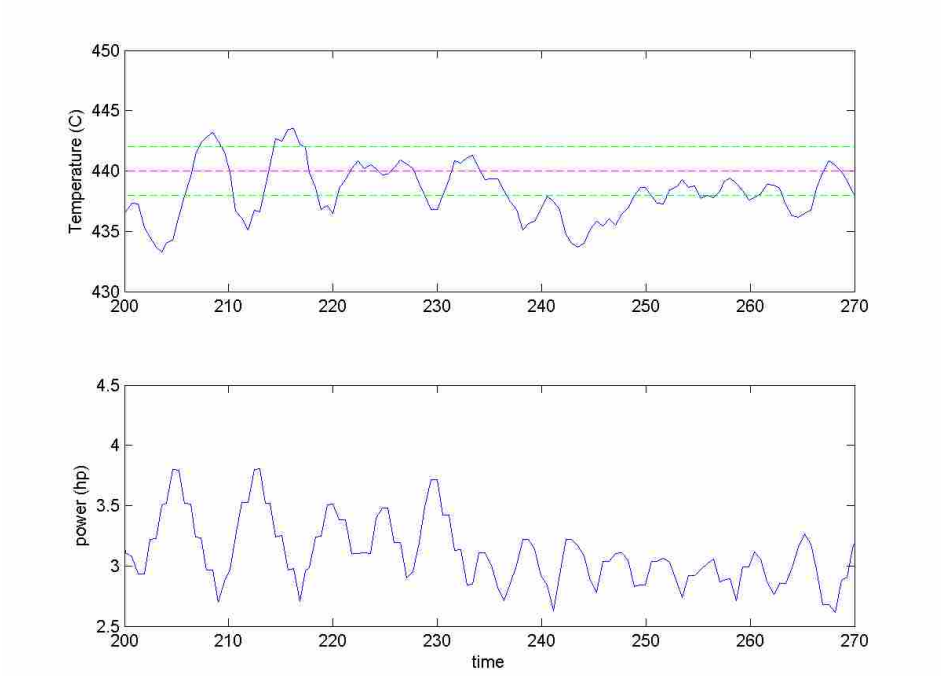



Figure C-8: Temperature and power during the middle of the HS 004 weld. The sawtoothing of power is evident, and creates a nonconstant temperature.

#### C.1.1.2.3 MV Move Penalty

One way to make a MPC controller more or less aggressive is to raise the gain on the MV so that the optimizer predicts that smaller inputs are needed to achieve the desired result, and this in turn makes the controller less aggressive. However, unlike the FOPDT model, there is no explicit gain on the power in the Hybrid Heat Source model per se as the power multiplication variable will also have a large impact on the weld offset temperature. Thus, a simple gain cannot be changed to reduce the aggressiveness of the controller.

Another way to limit the aggressiveness of a controller – especially when it is creating sustained oscillations about the setpoint – is to introduce a MV move penalty. For this method, a penalty is added for movement in the MV variable. The optimizer then tries to keep this low while also keeping the predicted temperature as close to the setpoint as possible. In practice a

multi-objective optimization in the MPC engine is rarely used as this (1) is much more computationally expensive than single-objective optimization, and (2) multi-objective optimization routines usually produce a variety of results along the Pareto frontier, and from these a single candidate must still be chosen to implement on the FSW machine. Instead of doing multiobjective optimization, a weighting function is used to add the MV move penalty to the CV's SSE penalty (temperature minus setpoint). At very large temperature differences, the temperature difference should compose the grand majority of the objective function's value, and the controller will thus aggressively try to achieve the setpoint. At small temperature differences, the temperature portion of the objective function is comparable to the MV move penalty, and so the controller will sacrifice aggressiveness of reducing the temperature error in order to keep the MV smooth.

A few welds were run with the MV penalty weighting factors of 10e-5, 3e-5, and 1e-5. At these weights, at large temperature differences the MV penalty portion was orders of magnitude smaller than the temperature difference portion, but when the temperature was within a few degrees of the setpoint, the two portions of the objective function were approximately within an order of magnitude of each other. The results of these three welds with a MV penalty vs. a no MV penalty weld is shown in Figure C-9. As can be seen, a middle value provided the best control, dominated the two controllers with higher and lower MV penalty weights, and almost dominated the controller with no MV move penalty. A further refinement of this weighting factor may very well lead to further improvements.

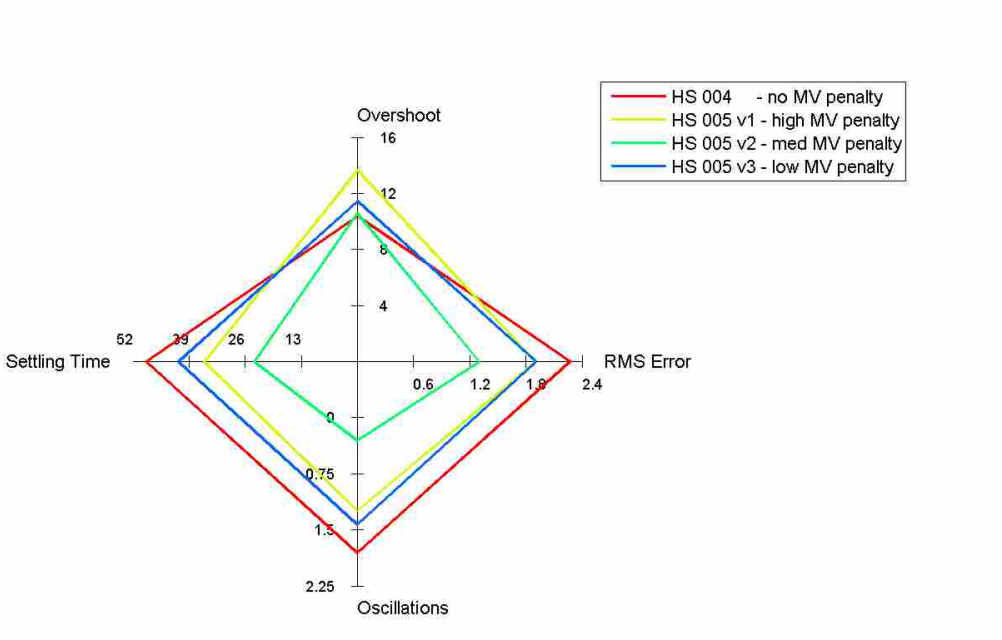



Figure C-9: Effect of adding a MV penalty with varying degrees of weighting relative to the CV SSE penalty. The controller with a medium amount of MV penalty almost dominates all other controllers.

Looking at a portion of the actual MV signals and CV responses for these particular welds is especially instructive (Figure C-10). For the weld with a low MV move penalty (HS 005 v3), the CV is nearly as jagged and as sawtoothed as a weld with no MV move penalty (HS 004). On the other hand, if the MV move penalty is too high (HS 005 v1), then the MV signal becomes very smooth, but at the expense of good of temperature control. A medium amount (HS 005 v2) does a good job at regulating some of the unnecessary MV movement when the CV is near the setpoint, but still allows the MV to move and respond when there are changes to the system.

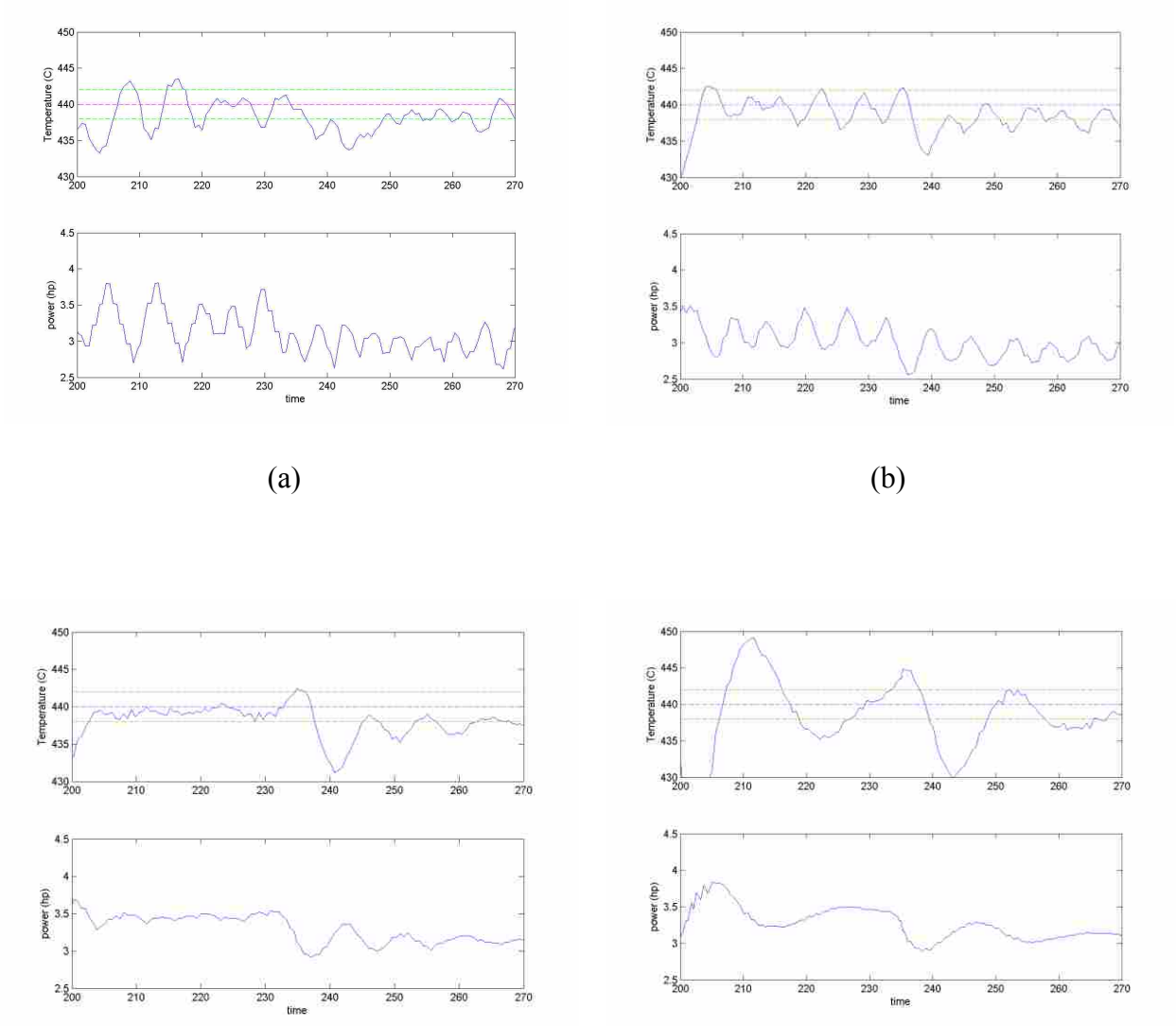



Figure C-10: MV (power) and CV (temperature) for welds with different levels of MV move penalty in their objective function, shown in order of lowest to highest MV move penalty weight. (a) HS 004 – no MV penalty, (b) HS 005 v3 – low MV penalty, (c) HS 005 v2 – medium MV penalty, (d) HS 005 v1 – high MV penalty.

### C.1.2 Comparison of all Controllers' Performance for the Initial Round of Testing

Each performance metric (RMS error, overshoot, settling time, and oscillations) was compared in two ways: each metric per segment type and averaged over all of the segments.

### C.1.2.1 Performance for Each Metric per Segment Type

The first metric of performance analyzed is the RMS error after settling which was defined in Section 5.2. The RMS error post settling was plotted against the average of each of the weld disturbance types and is shown in Figure C-11. The PID with regulator gains dominates all other controllers. Among the FOPDT models, 002 and 003 are very similar, as was previously noted in Section C.1.1.1.1. The HS 005 v2 controller dominates all of the other heat source type models except HS 003 in a couple categories.

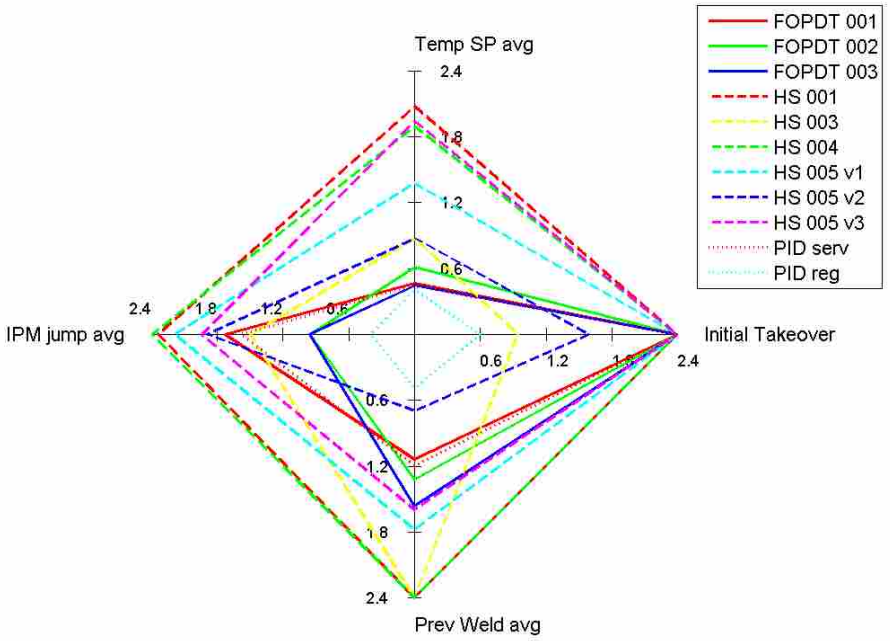



Figure C-11: Average RMS error for the initial round of testing, plotted for each of the weld segment types.

For overshoot, it is not surprising that the PID controller with servo gains, which are based upon a 0% overshoot rule, dominate the other PID controller and also do very well overall. Again, the FOPDT 002 and 003 controllers are very similar. The HS 003 controller is very good for the heat source controllers, with HS 005 v2 also scoring relatively well.

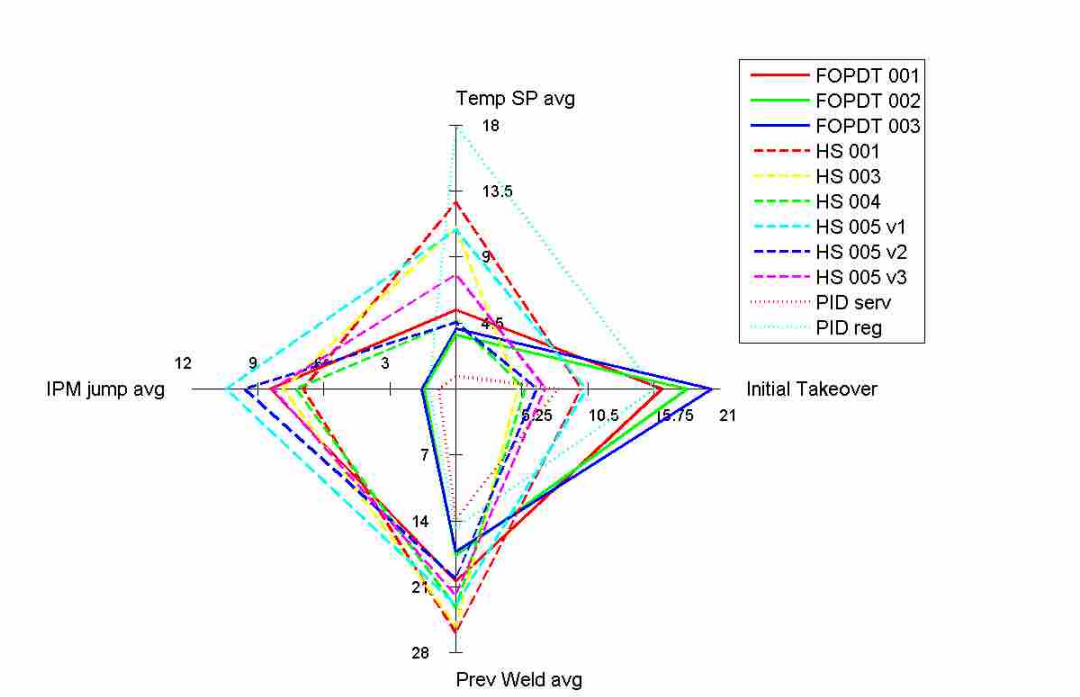



Figure C-12: Average overshoot for the initial round of testing, plotted for each of the weld segment types.

For settling time, the PID with regulator gains dominates most other controllers and is better than the servo PID controller, except in the temperature setpoint change segments. Again, the FOPDT 002 and 003 controllers are very similar. The HS 005 v2 controller appears to dominate all other heat source based controllers.

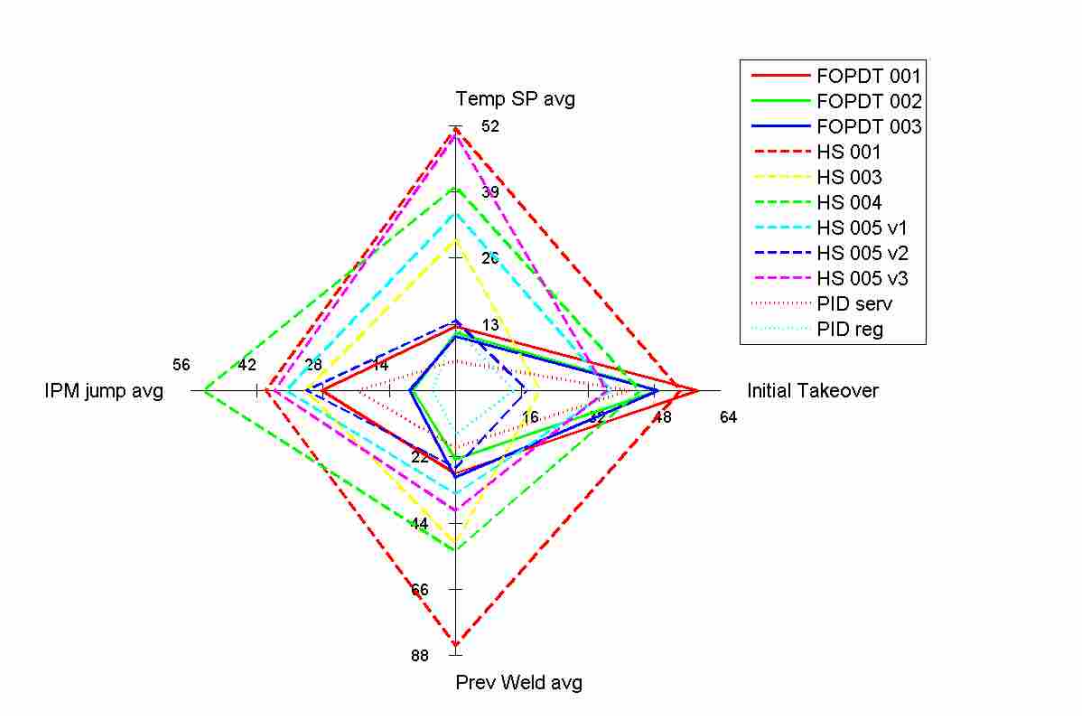



Figure C-13: Average settling time for the initial round of testing, plotted for each of the weld segment types.

For the number of oscillations per segment, the PID with servo gains dominates all other controllers and has no oscillations in any of the situations. This is unsurprising as the servo gains are based upon no overshoot and low oscillations criteria. Again, the FOPDT 002 and 003 controllers are very similar. The HS 005 v2 controller appears to dominate all other heat source based controllers and performs very similar to the two good FOPDT controllers.

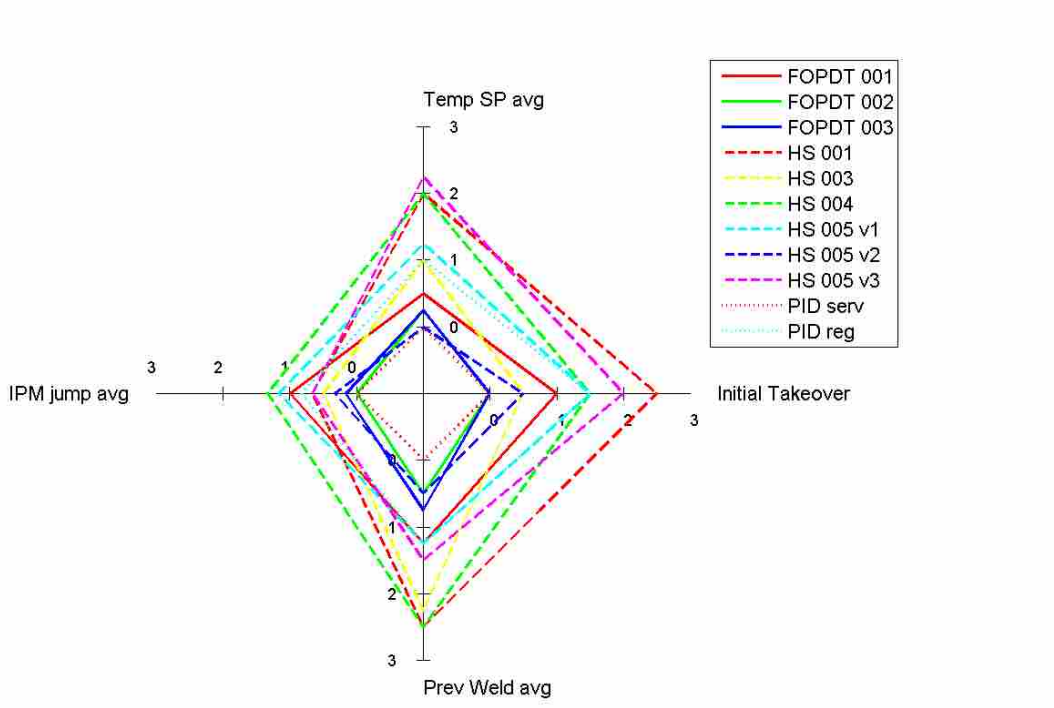



Figure C-14: Average number of oscillations for the initial round of testing, plotted for each of the weld segment types. Note that for readability purposes the 0 for each of the axes is not at the center of the plot.

### C.1.2.2 Overall Performance per Metric

The performance metrics were averaged over each of the eight weld segments and the results were plotted per metric. The results are shown in Figure C-15. Both of the PID controllers are very different from each other. The regulator gains PID controller is by far the best of any of the controllers in terms of RMS error and settling time, but has very poor overshoot and oscillation qualities. In contrast, the servo gains PID controller has the best overshoot and oscillation characteristics, as well as decent RMS error and settling time characteristics. The FOPDT 002 and 003 controllers are very similar to each other, with the 002 controller barely dominating. The HS 005 v2 controller appears to dominate all other heat source based

controllers, but is nonetheless dominated by the two good FOPDT controllers and the PID with servo gains.

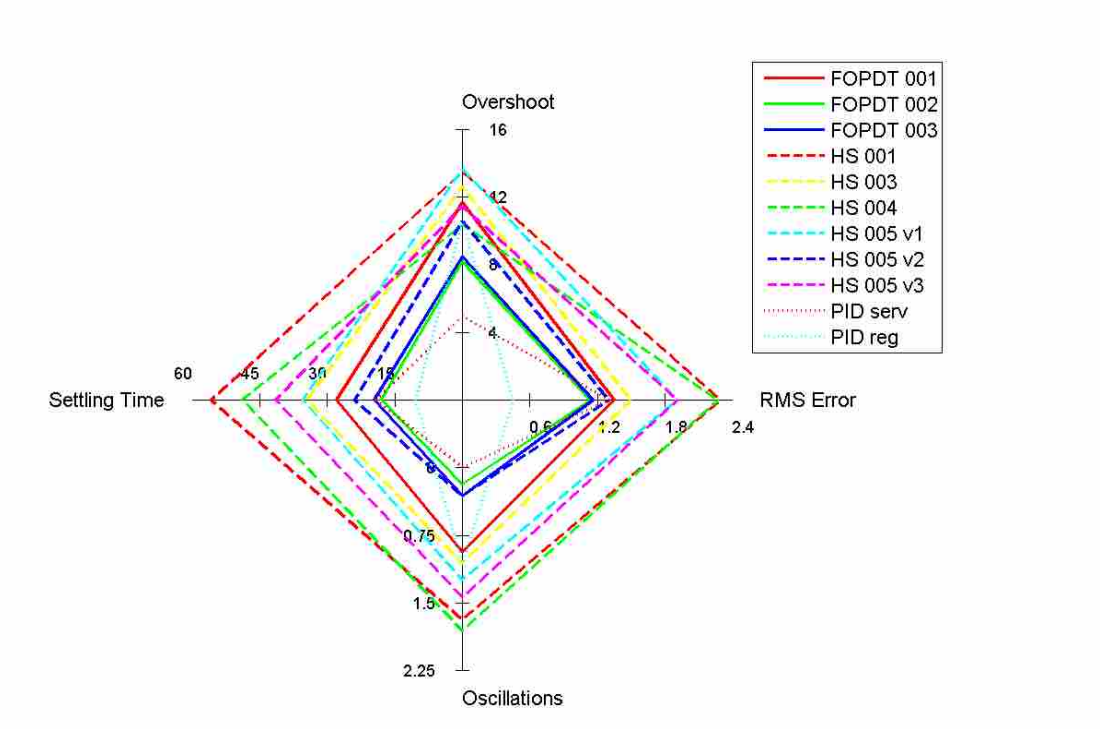



Figure C-15: Performance metrics for all of the welds in the initial round of testing, averaged per metric.

## C.2 Second Round of Quasi Steady State Testing

For the second round of testing, only the nondominated of each type of controller from the initial testing were selected for evaluation. These are the FOPDT 002, HS 005 v2, PID servo, and PID regulator controllers. In order to allow a longer term of evaluation for these four controllers, the number of weld segments and disturbances were reduced so that the full settling of the controllers could be observed and performance metrics calculated. The disturbances were chosen to be ones that were unmodeled. This was done to more rigorously test the MPC controllers and because jumps in temperature or traverse speed are rarely desired in actual industrial FSW applications.

The initial plunge and 2 inch traverse speed ramp were performed at a slightly higher spindle speed in order to raise the temperature further from the temperature setpoint, which allowed for a higher initial controller excitation. Subsequently, four weld segments were performed, each at 9 ipm, a temperature setpoint of 440°C, and for about 11 inches: (1) the initial controller takeover, (2) entering a previously welded region, (3) running over a weld pinhole and re-entering normal material, and (4) a rise in vertical position change of .01 inches. The fourth step was chosen because FSW is very dependent upon depth and welding plates, backing plates, and welding anvil surfaces often have variations in height which when compounded can easily exceed several thousandths of an inch. The plots of their power and temperature vs. time can be found in Appendix B.3.

### C.2.1 Performance for Each Metric per Segment

The PID controller with regulator gains nearly dominates all other controllers. The FOPDT controller is better than the PID servo and heat source controller in three of four segments, and PID servo controller is also better than the heat source controller in three of four segments.

For overshoot, it is not surprising that the PID controller with servo gains again nearly dominates all the other controllers, as it did for the initial round of testing. The heat source controller is also dominated by the other PID controller as well.

For settling time, the PID with regulator gains nearly dominates all other controllers. The FOPDT and PID servo controllers are extremely similar in each category. The heat source controller is almost dominated by all three other controllers.

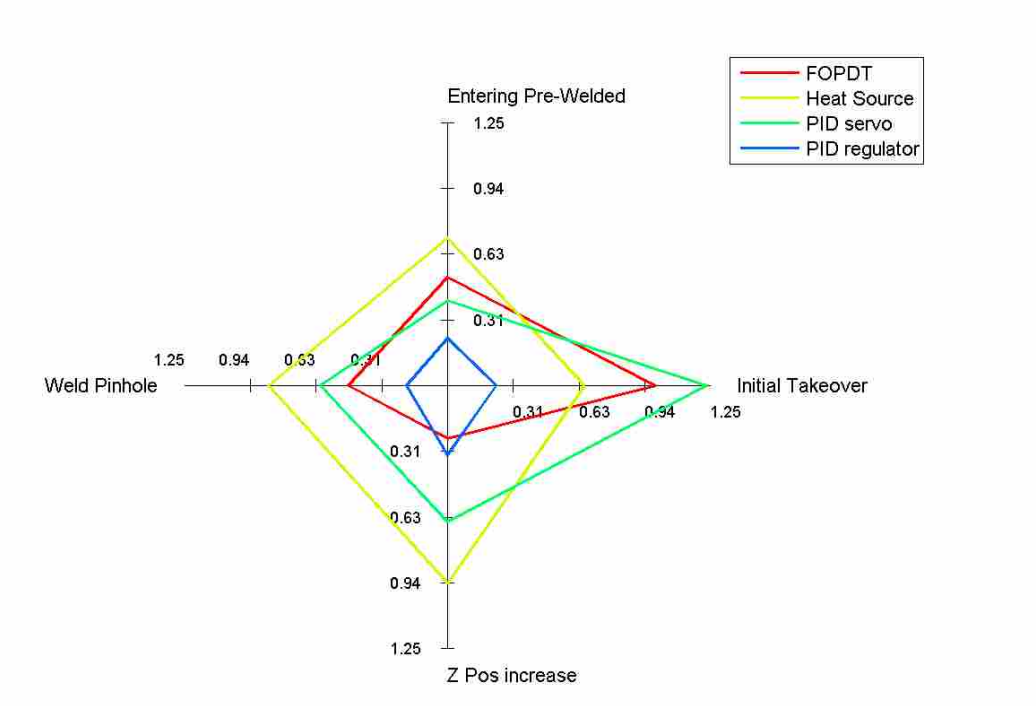



Figure C-16: RMS Error for the final round of testing, plotted for each of the weld segments.

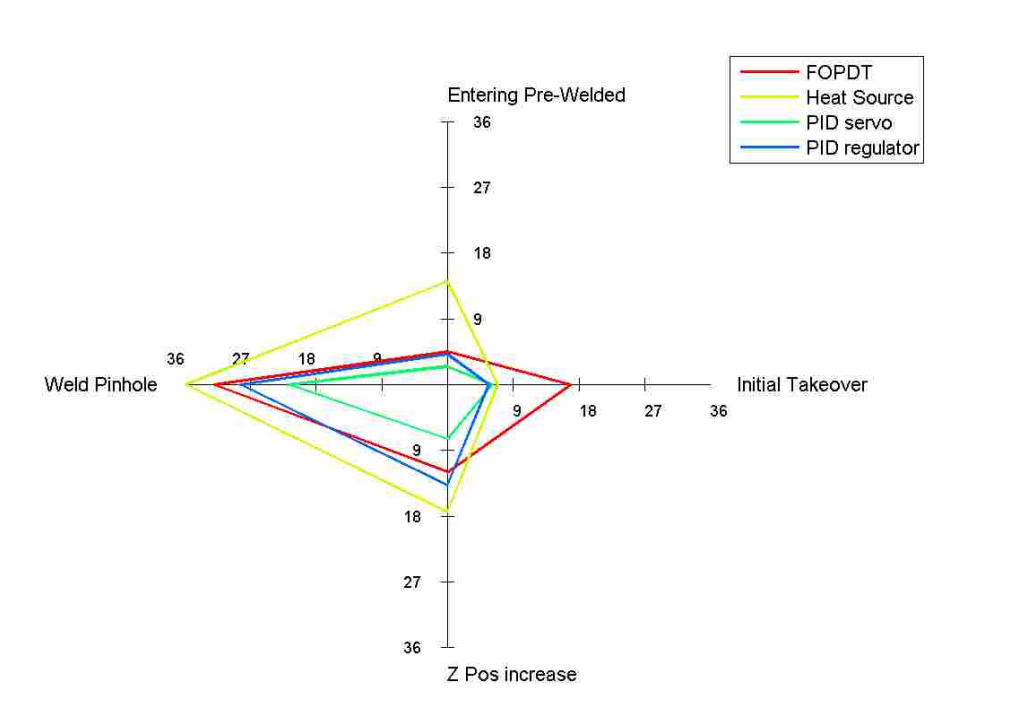



Figure C-17: Overshoot for the final round of testing, plotted for each of the weld segments.

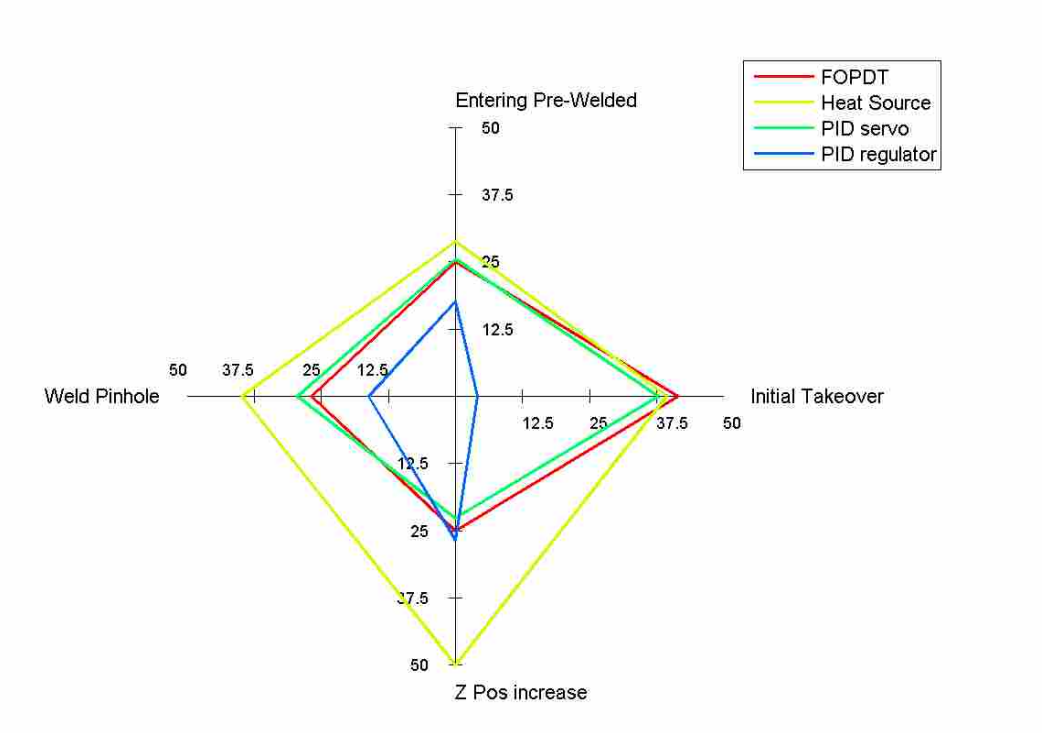



Figure C-18: Settling time for the final round of testing, plotted for each of the weld segments.

For the number of oscillations per segment, the PID servo controller is again a dominating controller, but it matched in performance by the FOPDT controller. These two dominate the PID regulator controller, and the heat source controller is dominated by the other three controllers.

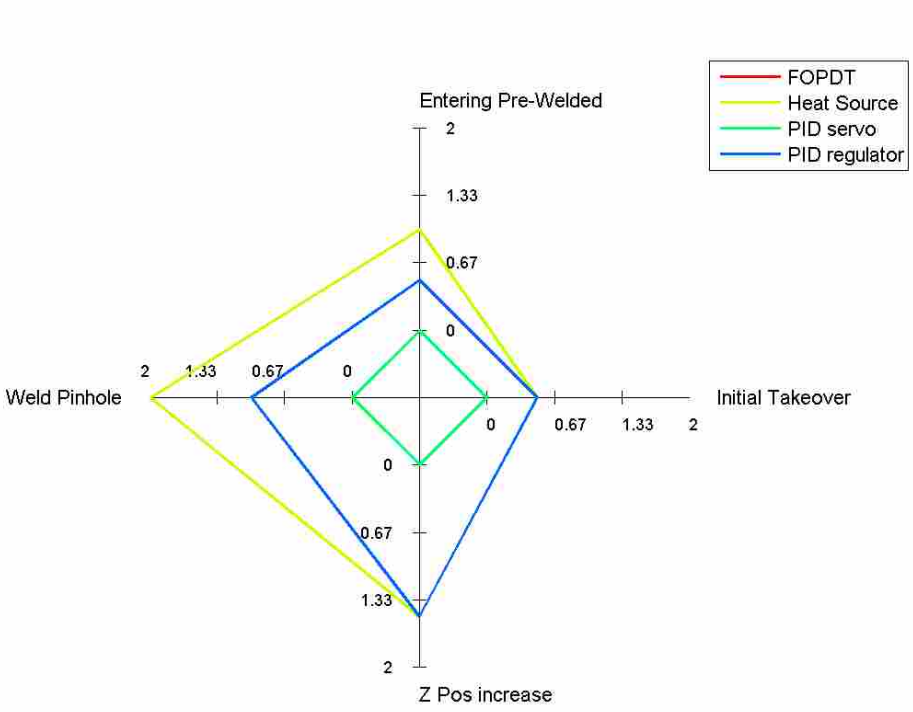



Figure C-19: Number of oscillations for the final round of testing per weld segment. Note that for readability purposes, the 0 for each of the axes is not at the center of the plot. Furthermore, the FOPDT and PID servo controllers each had exactly 0 oscillations, and so thus lie on top of each other in this plot and are visually indistinguishable.

### C.2.2 Overall Performance per Metric

The performance metrics were averaged over each of the four weld segments and the results were plotted per metric. The results are shown in Figure C-20.

The heat source controller is dominated in all categories by the other controller, and is also dominated in terms of the unplotted metrics of controller complexity and computational power required as well. As was the case with the initial round of testing, the FOPDT and PID servo gains controllers were extremely similar in their settling time and number of oscillations. Again, the PID servo controller is better in terms of overshoot, but the FOPDT controller has lower RMS error. Again, the PID with regulator gains has the best RMS error and settling time characteristics, but poorer overshoot and oscillation performance.

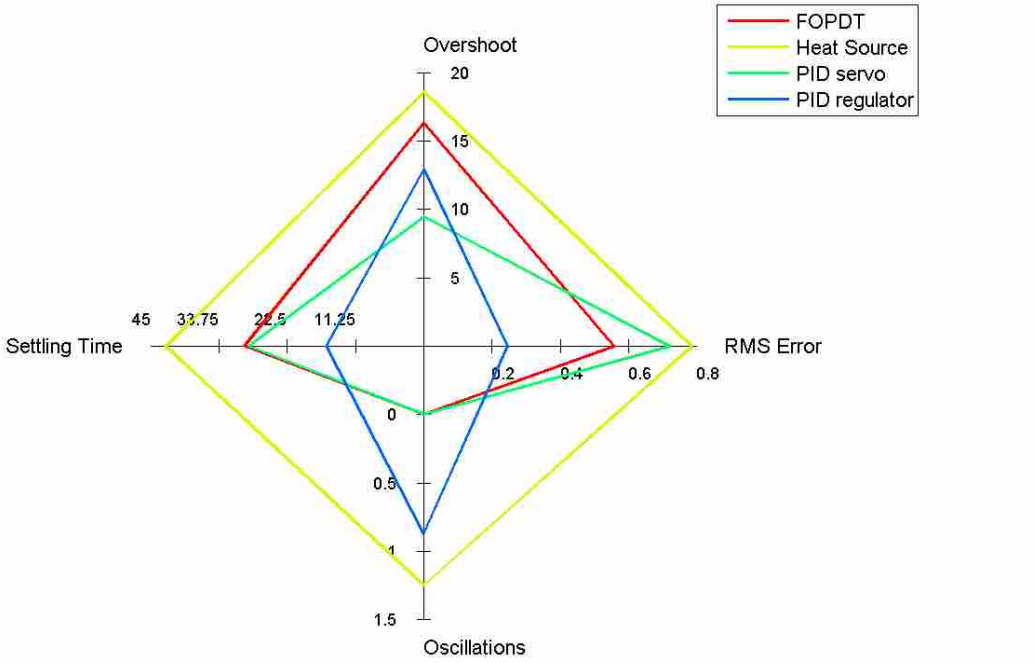



Figure C-20: Performance metrics for all of the welds in the final round of testing, averaged over the four segments.

## C.3 First Round of Transient Testing

In order to test the performance of the controllers during the initial transient, all controllers took control of the weld power immediately after the plunge, while the weld was still increasing in traverse speed from 0 to 9 ipm. Controllers with the same parameters as in the second round of quasi steady state testing were used. In addition to controllers with the same parameters as the second round of quasi steady state testing, FOPDT and Hybrid Heat Source based controllers with parameters manually fit to the initial transient portion of the weld were evaluated, and a PD regulator controller was used with the hopes that the elimination of the integrator term would help with weld control using the transient portion by eliminating oscillations. The PD regulator was exactly the same as the PID regulator controller, but with the integral term set to 0. The model parameters of the Hybrid Heat Source and FOPDT controllers

are shown below in Table C-4 and Table C-5. In the MPC controllers the dP/dt limit was raised to 1 hp/s and the $\alpha_{bias}$ was raised to 1 to help them be more aggressive.

Table C-4: Parameters for the First-Order Model for Weld FOPDT 102

| $c_1$ (hp/°C) | $c_2$ (m/s°C) | $c_3$ ($m^2/s^2$°C) | $c_4$ (°C) | $\tau$ (s) |
|---|---|---|---|---|
| 20 | -13.23 | 2.589 | 15.659 | 10 |

Table C-5: Parameters for the Hybrid Heat Source Model for Welds HS 102 & HS 102 v2

| $z_{pt}$ (inches) | $h_0$ ($W/m^2$ K) | $p_{mult}$ (unitless) |
|---|---|---|
| .18 | 4000 | 1.4 |

The results of these welds are shown in Figure C-21. It should be noted that in the short amount of time taken for these welds, the settling time and RMS error metrics are not necessarily relevant or even accurate as the welds may not have fully settled at termination.

Neither of the FOPDT controllers worked well; the second one is dominated by almost every other controller. It is initially surprising that the second FOPDT model with parameters that were manually fit to attempt to model the transient did worse than the stock model. The form of the FOPDT model is completely incapable of capturing the trends in the initial portion of the weld. With the power decreasing and traverse speed increasing, the FOPDT model predicts a dramatic drop in temperature, but in actuality the temperature is somewhat steady. Thus, manually fitting the FOPDT model didn't actually help to capture the trends that are in the transient portion of the weld. Because the manual parameters for the FOPDT model did not capture the transient well and significantly degraded the prediction post-transient, the parameters are not reported. A discussion on why manual parameters did not help is found in the body of the thesis.

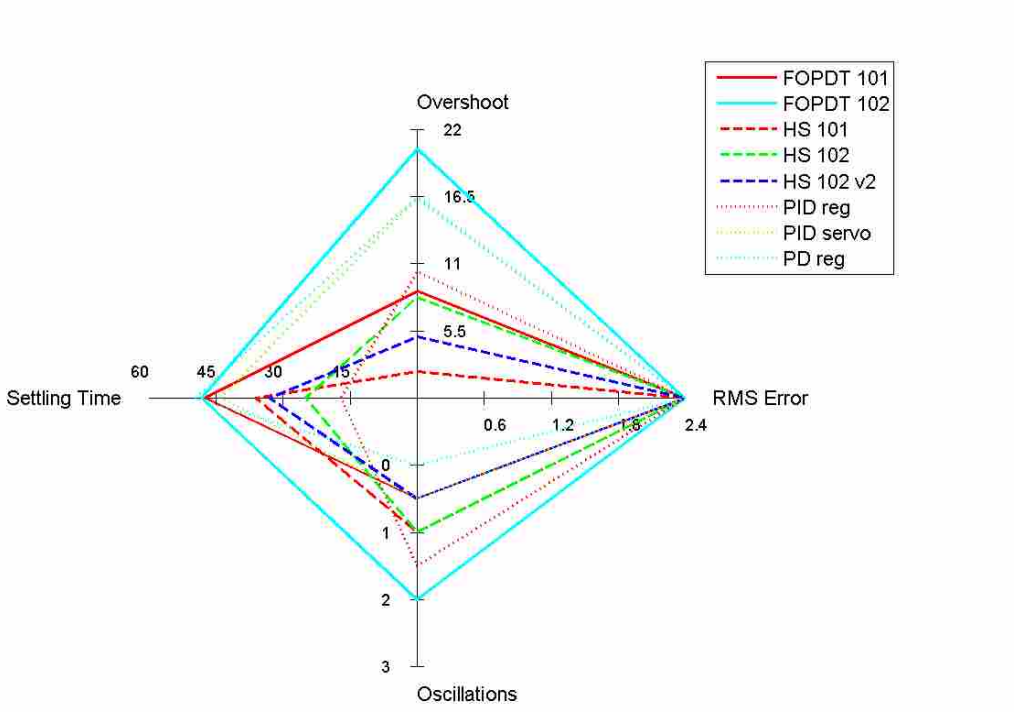



Figure C-21: Performance metrics for all of the welds in the first round of transient testing.

The MPC controllers did much better than the FOPDT controllers with the 102 v2 weld dominating all of the FOPDT controllers. Using manual parameters did improve some aspects of the weld, but others were slightly degraded. However, inspection of the temperature profiles (see Appendix B.4) shows that the two controllers with manual parameters are much better and settle very quickly. However, both briefly go outside of the settling band as it was defined, and thus the settling times appear to be similar to the original heat source weld. The manual parameters are presented in the body of the thesis in Table C-4and Table C-4.

The PID servo and PD regulator controllers did poorly because they did not have enough integrator action to bring the controller to the setpoint during the continuously changing

transient. The PID regulator controller, as is typical, had high overshoot and oscillations and low settling time.

Surveying the four main types of controllers, the FOPDT and PID servo controllers seem ill-suited for control during the transient part of the weld. Conversely, the heat source and PID regulator controllers had relatively good control.

## C.4 Second Round of Transient Testing

Welds were performed the same as in the first round of transient testing, but after the traverse speed reached the desired 9 ipm, the weld was allowed to continue at that state for 9 inches. The heat source controller with manually determined parameters was used, and for the second weld the controller was allowed to be more aggressive by reducing the time delay and the MV move penalty. A high degree of variability in the results was observed in other preliminary tests for this controller in a transient environment. For this reason, the same exact PID regulator controller was run twice. Results from these welds are shown in Figure C-22. The temperature and power curves for these welds can be found in Appendix B.5.

As can be seen by Figure C-22, the PID regulator controllers have relatively good RMS error and settling time characteristics, but poor oscillation and overshoot characteristics. The HS 103v2 weld has an atypically low settling time, but because the temperature in the HS 103 weld momentarily goes outside of the temperature band, its calculated settling time is slightly inflated over what it effectively is.

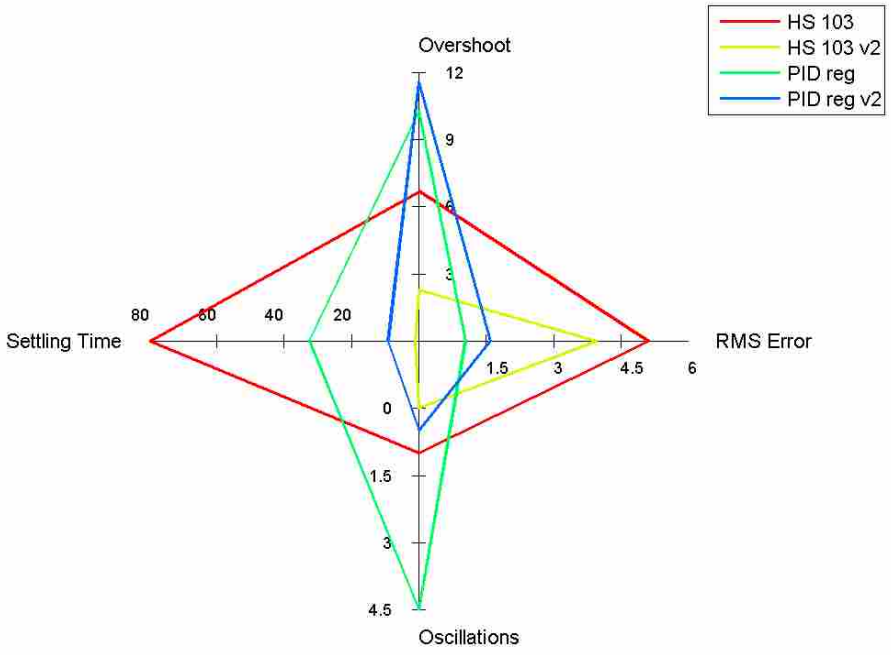



Figure C-22: Performance metrics for all of the welds in the second round of transient testing.

Generally speaking, for the highly transient portion of the welds, the heat source based controllers have more regular performance, whereas the PID regulator's performance may vary significantly. The heat source controller has much smoother temperature and power over the course of the weld, but it is less true to the temperature setpoint. Conversely, the PID regulator controller has large oscillations in both power and temperature in the beginning of the weld, but once the weld nears the steady state portion of temperature it becomes extremely steady and true to the setpoint.

## Appendix D. Miscellaneous Supporting Figures and Data

### D.1 Energy During the Plunge

During the plunge a significant amount of energy is put into the plate/tool system. However, immediately after the plunge the tool is still not at a steady state operating condition. This is because the energy expended during the plunge goes into both the tool and the plate, and this total amount is comparable to the energy required to heat the tool up to steady state, and more energy is therefore required to heat the plate to steady state as well. As such, neither the plate nor the tool will be at a steady state condition by the end of the plunge.

The energy required to heat a H13 tool that is 4" in length and 1" in diameter to steady state (assuming a linear thermal gradient from 450°C at one end to a cold thermal reservoir at the other) is about 45 kJ. Reviewing data from several welds, the total energy expended during the plunge ranged from about 41.5 kJ to 44.5 kJ, averaging about 43 kJ. Naturally, a faster plunge (which is commonplace in FSW of aluminum) would required higher peak power, but dissipate less energy and would heat the tool and plate less. A slower plunger would do the opposite. In Figure D-1 the energy vs. time profile for an average weld in this thesis is shown. The first vertical line shows when the tool comes in contact with the plate (it starts .05" above the plate), and the second line shows the end of the plunge and the start of the traverse.

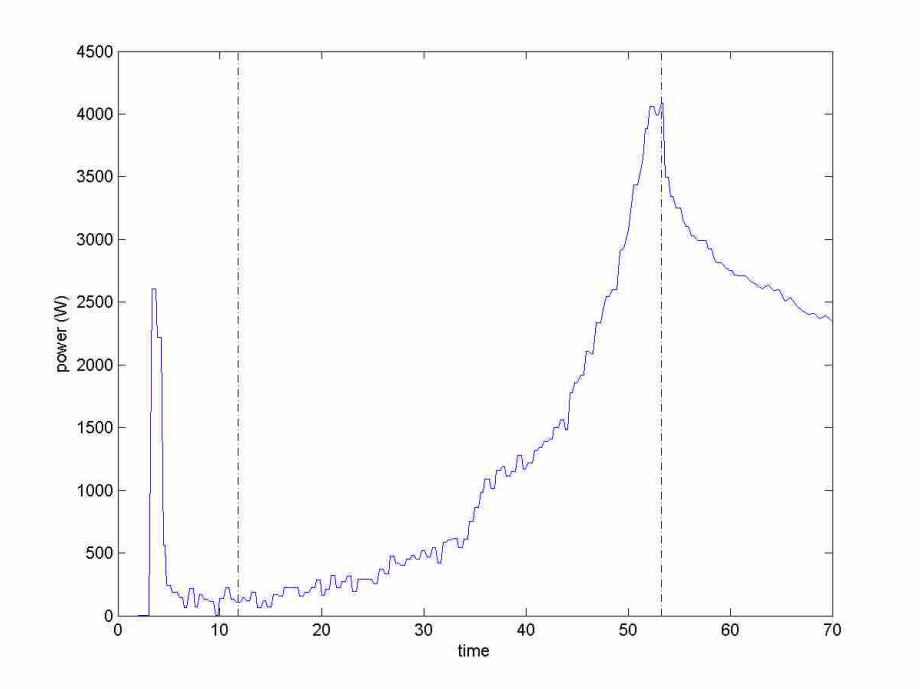



Figure D-1: Power vs. time curve during the plunge of a typical weld. The area under this curve between the two dashed black lines is about 44 kJ.

## D.2 CS4 Tool Geometry

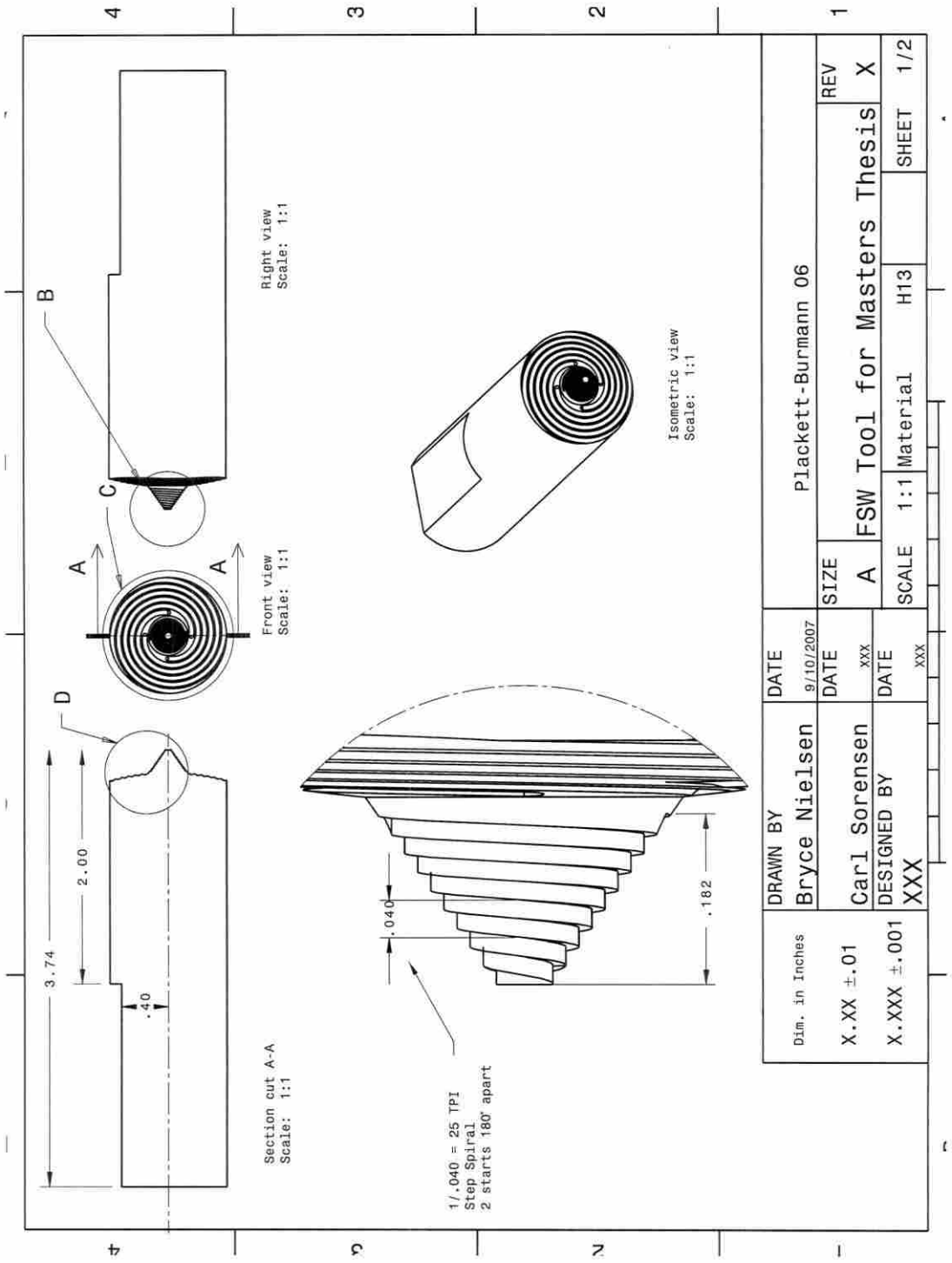



Figure D-2: CS4 tool geometry used in this thesis. Page 1 of 2.

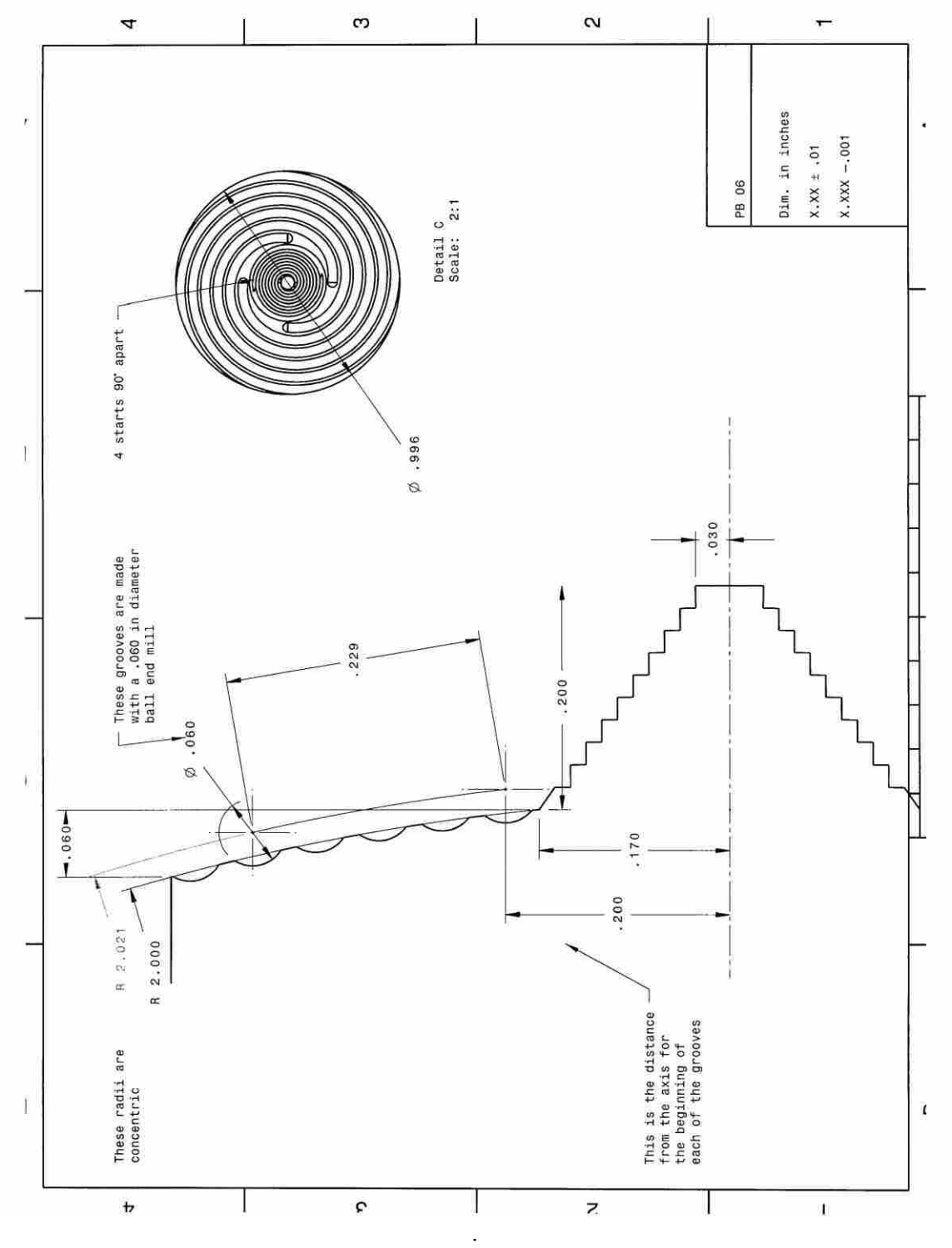



Figure D-3: CS4 tool geometry used in this thesis. Page 2 of 2.

## D.3 Picture of Plate Setup

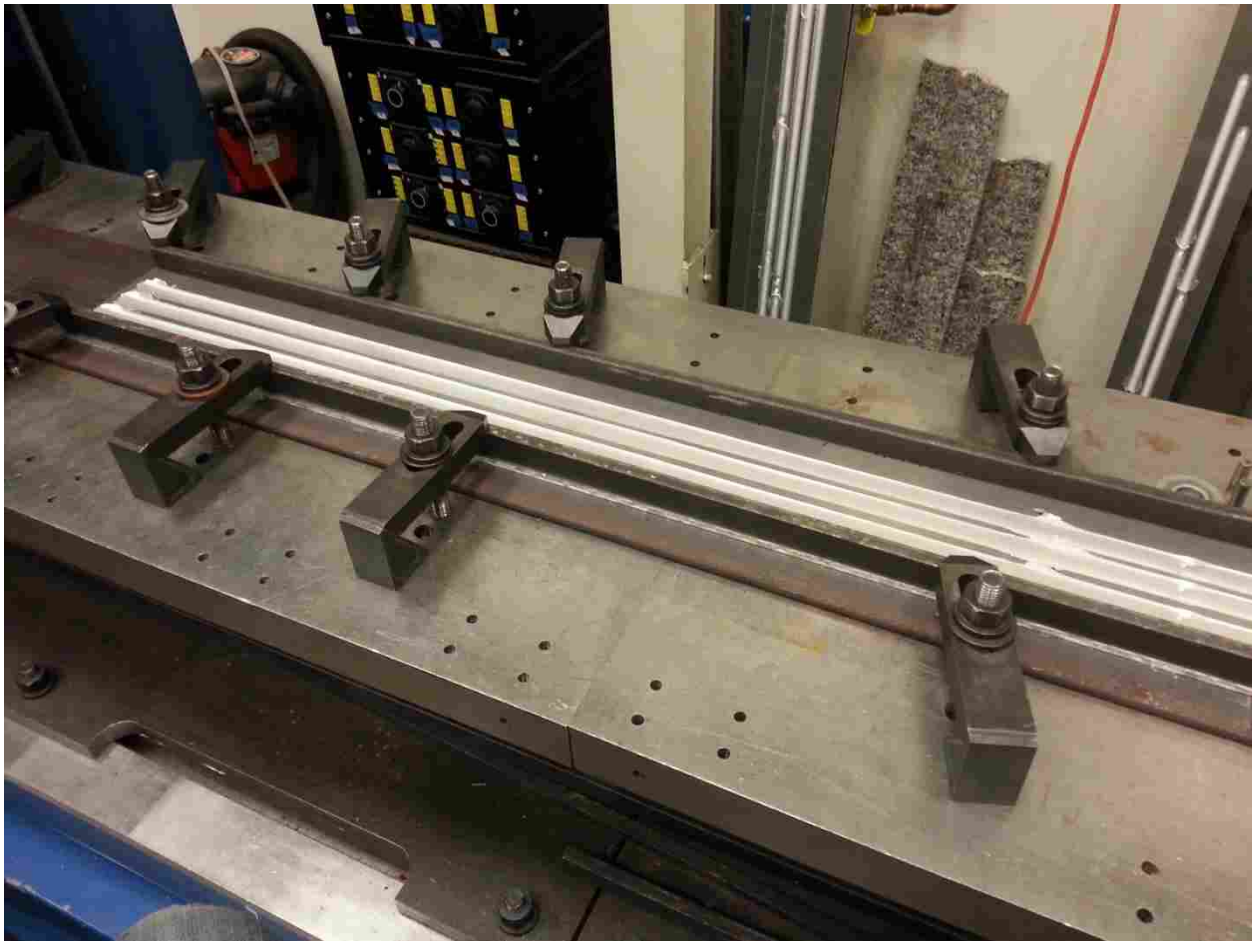


Figure D-4: Picture of setup for a 4' plate. Up to 3 welds were run next to each other on the same plate after the plate had cooled sufficiently.

## D.4 Time Horizon for the MPC Controllers

The time horizon for the MPC controller was specified in order to (1) allow sufficient resolution in the immediate future, (2) have predictive abilities that are sufficiently far in the future, and (3) have the ability for the optimizer to solve the problem quickly. A goal of about 2 Hz or better for the MPC controller was desired. In order to achieve this, the Hybrid Heat Source MPC controllers' time horizons had to be discretized more coarsely than the FOPDT MPC controllers' time horizons.

The spacing of steps for the FOPDT controller was: .5, .5, 1, 1, 2, 2, 3, 3, 3, 3, 3, 3, which totals to 25 seconds for the 12 increments.

The spacing of steps for the Hybrid Heat Source controller was: .5, .5, 1, 2, 3, 4, 6, 8, which totals to 25 seconds for the 8 increments.

## Appendix E. Computer Code

Code is provided for the Hybrid Heat Source model for both parameter estimation and the MPC controller. Code for the FOPDT model is simpler, and it is assumed that the step down to the FOPDT model can easily be done with this provided code and therefore separate code is not provided.

In addition to Matlab 2013 or later, the OPC and Optimization toolboxes are needed.

## E.1 PRBS Files

### E.1.1 PRBS_generator. m

```
%PRBS generator
clc
clear all
format compact
%*************************************************************************
%***********INPUTS:
filename = 'PRBS_21.mat'
savefile = 1;

%time
Hz = 10; total_time = 500;
t = 0:1/Hz:total_time; %Calculated

%for the highs & lows:
time_range = [5 30];%the computer will chose a value in between these 2
%Power:
P_range = [3.05 3.2 3.35];%the computer will chose one of these each time
%ipm:
ipm_range = [2 3 4 5 6];%the computer will chose one of these each time
%*************************************************************************
P_num = length(P_range); ipm_num = length(ipm_range);
%set up the intervals to be 0 at first
t_interval_P = 0;
t_interval_ipm = 0;
%set the P and ipm to values which will allow a new one to be generated
P_index = 1;
ipm_index = 1;
P_index_new = 0;
ipm_index_new = 0;
%initialize variables:
power = zeros(length(t),1);
ipm = power;
distance = ipm;
total_distance = 0;

for i = 1:length(t)
    %if we are starting from scratch or have come to the end of a segment,
    %chose a new time interval and command value for that duration
    if t_interval_P <= 0
        %calculate a new time interval for the next P
        t_interval_P = rand(1)*(time_range(2)-time_range(1)) + time_range(1);
        %chose an index of P to get the power from
        P_index_new = round((P_num-1)*rand) + 1;
        %keep on choosing a new index if it was the same as the last one
        while P_range(P_index_new) == P_range(P_index);
            P_index_new = round((P_num-1)*rand) + 1;
        end
        %set up the indices for the next time
        P_index = P_index_new;
```

```
        %set the power
        P_set = P_range(P_index_new);
    end

    if t_interval_ipm <= 0
        %calculate a new time interval for the next ipm
        t_interval_ipm = rand(1)*(time_range(2)-time_range(1)) +
time_range(1);
        %chose an index of ipm to get the power from
        ipm_index_new = round((ipm_num-1)*rand) + 1;
        %set up the indices for the next time
        ipm_index = ipm_index_new;
        %set the power
        ipm_set = ipm_range(ipm_index_new);
    end
    %set the values
    power(i) = P_set;
    ipm(i) = ipm_set;
    total_distance = total_distance + ipm(i)/(60*Hz);
    distance(i) = total_distance;
    %reduce the time interval by one clock cycle
    t_interval_P = t_interval_P - 1/Hz;
    t_interval_ipm = t_interval_ipm - 1/Hz;
end

total_distance
average_power = mean(power,1)
average_ipm = mean(ipm,1)

figure(1)
clf
plot(t,power,'--',t,ipm,'b')
xlabel('time (s)')
ylabel(' IPM & Power (hp)')
legend('Power','IPM','location','southeast')
axis([0 max(t) min(min(ipm_range),min(P_range)) - .5,...
    max(max(ipm_range),max(P_range)) + .5])

figure(2)
clf
plot(distance,power,'--',distance,ipm,'b')
xlabel('tdistance (inches)')
ylabel(' IPM & Power (hp)')
legend('Power','IPM','location','southeast')
axis([0 total_distance min(min(ipm_range),min(P_range)) - .5,...
    max(max(ipm_range),max(P_range)) + .5])

exportpath = 'PRBS_profiles';
exportname =  fullfile(exportpath,filename);

if savefile == 1

save(exportname,'t','power','ipm','total_distance','distance','average_power'
,'average_ipm')
end
```

### E.1.2 PRBS_welder.m

```
%PRBS

clc
clear all
format compact

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          Weld set-up, input parameters, extract data, etc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%INPUTS:
importname = 'PRBS_thesis_3.mat'
real_time_plotting = 1;%set if you want a real-time plot updated ~every x
seconds
plotting_frequency = .5;%in seconds
max_PRBS_time = 3600; %this will just put a fail safe
%save settings
newfilename = 'PRBS_new_Al7075_full_005.mat'
material = 'Al_7075_T7_new';
thickness = .25;
tool_geom = 'CS4';%the type of tool - usually the same unless custom tool
TC_location = 'tip';%location of TC in the tool
tool = 1;%number for the tool of specific geom
tool_setup = 1;%setup for that tool, as things may change
save_file = 1;%select whether or not to write the acquired data at the end or
not

%high/low override inputs
high_low_override = 0;%tell it if a real or fake weld
power_override_low = 2.9;
power_override_high = 3.3;

%air weld inputs
air_weld = 0;%tell it if a real or fake weld
power_air_low = .1;
power_air_high = .125;

importpath = 'PRBS_profiles';
filename =  fullfile(importpath,importname);
%set up to save to a specific directory
exportpath = 'PRBS_step_test_data';
exportname =  fullfile(exportpath,newfilename);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%check to see if that file is already there
current_names = dir(exportpath);
for i = 1:length(current_names)
    if strcmp(current_names(i).name, newfilename) == 1
```

```
        warning('The name you have selected to give to this weld is already
in use in the target folder. If you do not stop this file, the previous data
will be replaced')
        disp(' ')
        warning('If you dont want to erase the file or to quit this script,
then quickly navigate to the destination folder, and copy that file somewhere
else momentarily')
        warning(exportpath)
    end
end

%do a preview plot of what the chosen PRBS will be
C = open(filename);

figure(11)
clf
subplot(2,1,1)
plotyy(C.t,C.power,C.t,C.ipm)
xlabel('time (s)')
ylabel(' IPM & Power (hp)')
title('Preview of what the PRBS should look like')
legend('Power','IPM','location','southeast')

subplot(2,1,2)
plotyy(C.distance,C.power,C.distance,C.ipm)
xlabel('Distance (in)')
ylabel(' IPM & Power (hp)')
legend('Power','IPM','location','southeast')

total_distance = max(C.total_distance)

%extract values from the PSRB file to use more easily
time_goal = C.t';%rotate this vector
power_goal = C.power;
ipm_goal = C.ipm;
x_goal = C.distance;

%this is for air weld stuff only -- change the power to low levels
if air_weld == 1
    power_orig_max = max(power_goal);
    power_orig_min = min(power_goal);

    power_new = (power_goal - power_orig_min)*...
        (power_air_high-power_air_low)/(power_orig_max-power_orig_min)...
        + power_air_low;
    power_goal = power_new;

    warning('The computer is set for an Air Weld. If that was a mistake,
abort the weld NOW')
end

%this is for demanding artificial highs & lows
if high_low_override == 1
    power_orig_max = max(power_goal);
    power_orig_min = min(power_goal);
```

```
    power_new = (power_goal - power_orig_min)*...
        (power_override_high-power_override_low)/(power_orig_max-
power_orig_min)...
        + power_override_low;
    power_goal = power_new;

    warning('The computer is set to override the high & low power. If that
was a mistake, abort the weld NOW')
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%**************Create an opcda object, connect to the server, set up tags
opc_setup

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%********Do some settings and checks before commencing to the PRBS portion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%set min/max power based upon an air weld or regular weld
if air_weld == 1
    write(min_max_power_grp, {.05,.5})
else
    write(min_max_power_grp, {1,10})
end

%You need to make sure that it starts out NOT active so that is can switch
write(MPC_ACTIVE_itm_write, 0)

%**********Pause and do checks before starting up
disp('Matlab will now wait until the machine begins the weld')
disp(' ')
i4 = 0; welder_on = 0; %initialize counter and the holder
while welder_on == 0
    i4 = i4+1;
    weld_in_process_struct = read(weld_in_process_itm, 'device');
    weld_in_process  = weld_in_process_struct.Value;

    %if it is welding, then get out of this loop
    if weld_in_process == 1
        welder_on = 1;
        disp('The welder is now going')
        break
    end

    pause(.1)%pause for .1 seconds before going on -- no reason to loop like
crazy
end


disp(' ')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%*********start the loop that will actually do power control & stuff*******
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%now start the loop, start the timer, etc

date_time = clock;%this is used later; but take the date & time at start
i3 = 0;
time_current = 0;%initialize stuff
plot_toggle = 0;
take_over = 0;% = 1 when the CODE is actually writing stuff; like MPC_ACTIVE
PRBS_time = 0;%time measured from when PRBS takes over; before = 0;
tic
while welder_on == 1
    i3 = i3+1;
    loop_time = toc;

    %*****************************************************************
    %Read values from the PLC
    read_opc_group_values

    %*****************************************************************
    %Now check to see if we are traversing AND in PRBS mode
    if weld_traverse_current == 1 && MPC_enable_current == 1

        %The FIRST time we are good: go active, start timer, etc
        if take_over == 0;
            PRBS_timer_start = loop_time;%we need to know this
            write(MPC_ACTIVE_itm_write, 1)%write that it is active to PLC
            take_over = 1;%so that we won't come back into this loop again
            disp('The welder has now passed/tried to pass control to Matlab')
        end
        PRBS_time = loop_time - PRBS_timer_start;

        MPC_enable = MPC_enable_current;%update this or else the loop will
never end

        %Find the index of the data file that corresponds to current PRBS
time
        for i = 1:length(time_goal)
            if (time_goal(i) - PRBS_time) >= 0
                index = i;
                break
                %to make this better, I should get the first of the two
                %indices that straddle the time as there will be a bit
                %of time spent by matlab and the PLC implementing this plan
            end
        end

        %!!!I would like to add an ipm offset here too
        write(write_grp,
{power_goal(i)+RPM_offset_current/100,ipm_goal(i),1})%yes, this should be i
not i3
    else
        pause(.1)%otherwise do a small pause
    end
```

```
    %*********************************************************************
    %record data into a structure
    log_data

    %*********************************************************************
    %Plots

    %now plot stuff every X seconds of running -- this will slightly
    %affect performance though....
    do_i_plot = floor(loop_time/plotting_frequency);
    if do_i_plot > plot_toggle

        figure(12)

plot(data.time,data.MPC_pwr_cmd_matlab,data.time,data.MPC_pwr_cmd_matlab +
data.power_offset)
        title('Power - original PRBS profile, + offset')

        figure(3)
        plot(data.time,data.MPC_ipm_cmd_matlab,
data.time,data.MPC_ipm_cmd_matlab + data.x_vel_offset,
data.time,data.x_velocity_actual)
        title('X Velocity - original PRBS profile, + offset, & actual')

        figure(4)
        plot(data.time,data.tooltemp)
        title('Tool Temperature')
        axis([0, loop_time, 390, 480])

        figure(5)
        plot(data.time,data.spindle_power)
        title('Spindle Power')
        axis([0, loop_time, 2.9, 3.8])

        figure(6)
        plotyy(data.time,data.spindle_speed,data.time,data.motor_torque)
        title('Spindle Speed & Torque')

        plot_toggle = plot_toggle + 1;%set the toggle switch one higher
    end

    %*********************************************************************
    %Some end conditions

        %if it runs out of "goal" data too soon, or hits the hard end, then
set
    %the "MPC_active" back to 0 -- this will not show up in the data
    %structure created here as the data writing is already done at this
    %point
    if i3 == length(time_goal)
        write(MPC_ACTIVE_itm_write, 0)
    end
```

```
    if  loop_time > max_PRBS_time
        warning('The input "max PRBS_time" has been exceeded. Killing matlab
program...')
        write(MPC_ACTIVE_itm_write, 0)
        welder_on = 0;
        break
    end

    if weld_in_process_current == 0%if the welder has stopped, kill it
        welder_on = 0;
        break
    end
end
write(MPC_ACTIVE_itm_write, 0)%regardless of how it ended, write this again

average_loop_time = data.time(end)/i3
length_of_data = length(data.time)
data.average_loop_time = average_loop_time ;

%save the data collected
if save_file == 1

save(exportname,'data','date_time','material','thickness','tool_geom','TC_loc
ation',...
        'tool','tool_setup','air_weld')
else
    warning('Data has not been saved! Do this manually or it will be lost')
end

disconnect(da)
disp(' ')
disp('The script has finished, and has saved the file to the specified
folder')
```

## E.2 Parameter Estimation

### E.2.1 constant_finder_heat_source.m

[set the optimizer to use larger than standard minimum movements in the variables so that the output different will be big enough for the optimizer to work correctly via "diffminchange"; otherwise the noise of the actual data will overwhelm the optimizer. Set other options as desired]:

```
clc
clear all
format compact
format shortg

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global P %this is my global parameter to pass things back and forth
P.description = 'P is a variable that is used to easily pass stuff between
functions';
P.timer.FEA_solver = 0; P.timer.interp = 0; P.stop = 0;

param_loader %load tool, material, & other geometry into the P variable;
tool_mesher %call the function tool mesher

P.current.T_chiller = -4;
P.current.T_collar  = -4;
P.current.T_ambient = 12;
P.constants.constrain_back_end = 1;

P.solver.node_of_interest = 4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Select files to optimize against

filepaths{1} = fullfile('Compressed Files','PRBS_new_Al7075_full_002.mat');
filepaths{2} = fullfile('Compressed Files','PRBS_new_Al7075_full_003.mat');
filepaths{3} = fullfile('Compressed Files','PRBS_new_Al7075_full_004.mat');
P.current.multi_filepaths = filepaths;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

alg = 'sqp';
% alg = 'active-set';
alg = 'interior-point';);
options = optimset('Algorithm',alg,'maxiter',1000,...
    'tolfun',1e-4,'tolcon',1e-2,'diffminchange',1e-4,...
    'display','iter','MaxFunEvals',10000);
% the 'diffminchange' is *NECESSARY* to be higher than default, otherwise
noise of the data will overwhelm the optimizer
```

```
point_depth = .125; %in INCHES!
power_increase_ratio = 1;
h_tip = 3000;

k_cst = 140;
P.mat.k = k_cst;

tic

x0 = [point_depth;
    power_increase_ratio;
    h_tip];

P.lock_x0 = [0; 0; 0];
P.x0 = x0;

P.x0 = [0.042811
      0.60114
       4613.3];%optimized parameters

%
% P.x0 = [0.18
%         1.4
%         600.3];
x0 = P.x0;

lb = [.1 .5 500]';
ub = [.8 1.8 6000]';

feval('obj_funct_param_fitter_heat_source_multiple_files',x0)

x_new = fmincon('obj_funct_param_fitter_heat_source_multiple_files',...
     x0,[],[],[],[],lb,ub,[],options)

toc
% save('HS_constants_manual_v1','x0','h_collar','k_cst')

P.solver.node_of_interest
```

### E.2.2 function: obj_funct_param_fitter_heat_source_multiple_files.m

```
function [sse_total] = obj_funct_param_fitter_heat_source_multiple_files(x0)

global P %this is my global parameter to pass things back and forth

filepaths = P.current.multi_filepaths;
lf = length(filepaths);

%loop through the files and find the error for each
for i = 1:lf
    P.current.file_num = i;
    P.current.filepath = filepaths{i};
    sse(i) = feval('obj_funct_param_fitter_heat_source',x0);
end

%display stuff to the screen
x0'
sse_total = sum(sse)
toc
disp(' ')
```

### E.2.3 function: obj_funct_param_fitter_heat_source.m

```
function [sse] = obj_funct_param_fitter_heat_source(x0)

global P; %this is my global parameter to pass things back and forth

T_start = P.current.T_collar*ones(size(P.geo.dx,1)+1,size(P.geo.dx,2));
T_start(1) = P.current.T_ambient;

change_desired_z_pos = 0;%allows for a shallower HS calc during plunge
node_of_interest = P.solver.node_of_interest;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Optimizing variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

P.current.point_depth = x0(1);%in INCHES!
P.numerics.power_increase_ratio = x0(2);
P.constants.h_tip = x0(3);

ratio_of_2D = 0;
%I have this in so that I can do combo 2D & 3D methods, if desired

% P.constants.h_tip = 3000;
%wouldn't this be a function of fluid velocity (aka
% tool-to-plate velocity difference, and thus a function of the RPM & radius?
idea)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The actual function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%I will need to give it the full file path so it can gets its info...
% filepath = fullfile(foldername,filename);
filepath = P.current.filepath;
C = open(filepath);

%get the data, and cut off stuff before the tool even contacts the plate
z_pos_full = C.data.z_pos;
index = 1;%just in case it started mid-weld
for i=1:numel(z_pos_full)-1; if z_pos_full(i) > 0 && z_pos_full(i+1) < 0;
index = i; end; end;
t0 = C.data.time(index);
time = C.data.time(index:end-1) - t0;%otherwise matlab's built in
interpolation has problems
power = 1*(C.data.spindle_power(index:end-1)*765-0*765);%convert to watts;
subtract off the friction stuff first
power_cmd = 1*765*(C.data.MPC_pwr_cmd(index:end-1) +
C.data.power_offset(index:end-1));

tool_position = ( C.data.x_pos(index:end-1) - C.data.x_pos(index) )*.0254;
tool_position = position_corrector(time,tool_position);
x_vel = C.data.x_velocity_actual(index:end-1);
tool_temp =  C.data.tooltemp(index:end-1);
```

```
traverse = C.data.weld_traverse(index:end-1); engage =
C.data.weld_engage(index:end-1);
MPC_enable = C.data.MPC_enable(index:end-1);
MPC_PRBS = C.data.MPC_PRBS(index:end-1);

%initialize some things
dT0 = zeros(size(power)); dT1 = dT0; dT2 = dT1; dt = dT0; Q1_new =
power;%initialize some stuff
Q1 = power; T_prev = T_start; Time = time;
Theta_new = zeros(length(T_prev),length(tool_temp));
Theta_new(:,1) = T_start;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%check and see if it is a PRBS or not; if so I will want to cut it a bit
%more..
PRBS = 0;
traverse_index = [1, length(traverse)];
for i = 2:length(MPC_PRBS)
    if MPC_PRBS(i) == 1
       PRBS = 1;
    end
    if traverse(i-1) == 0 && traverse(i) == 1
        traverse_index(1) = i;
    elseif traverse(i-1) == 1 && traverse(i) == 0
        traverse_index(2) = i;
    end
end

% PRBS
%do a correction for one of the cases
if PRBS == 1
%     traverse_index = traverse_index + [200,-100];
    traverse_index = traverse_index + [2,-10];
else
    traverse_index = traverse_index + [0,-10];
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 2:traverse_index(2);%length(tool_temp)-200
    if mod(i,1000) == 0; disp(strcat('num: ', num2str(i)) ); end%disp iter

    %get & update data
    T_prev = Theta_new(:,i-1);
    P.current.tool_depth = -.0254*C.data.z_pos(i); %this is the *current*
tool depth
    if traverse(i) == 0 && engage(i) == 0; P.current.tool_depth = -.0254;
end; %so when it come out contact stops     if P.current.tool_depth <= 0;
power(i) = 0; end;%0 the power when we are not in contact;
    tool_contact_area%for the tool tip contact calculation

    %adjust desired depth if this fucntionality is turned on
    if change_desired_z_pos == 1 &&...
            C.data.z_pos(i) > -point_depth && C.data.z_pos(i) < 0
```

```
       P.current.point_depth = -C.data.z_pos(i) + .02;
    end
    dt(i) = time(i) - time(i-1);

    Q1_new(i) = Q1_new(i-1);

    dT0(i) = heat_source_model_3D_funct(time,Q1_new,tool_position,...
        time(i),tool_position(i)/.0254, dt(i), time(i) );%convert the desired
position to meters!
    dT1(i) = heat_source_model_3D_funct(time,Q1_new,tool_position,...
        time(i),tool_position(i)/.0254,  0,  dt(i)     );%convert the desired
position to meters!

%     if ratio_of_2D == 0
%         twoD0 = 0;
%         twoD1 = 0;
%     else
%     twoD0 = heat_source_model_2D_funct(time,Q1_new,tool_position,...
%         time(i),tool_position(i)/.0254, dt(i), time(i) );
%     twoD1 = heat_source_model_2D_funct(time,Q1_new,tool_position,...
%         time(i),tool_position(i)/.0254,  0,  dt(i)     );
%     end
%
%     if ratio_of_2D == 1
%         threeD0 = 0;
%         threeD1 = 0;
%     else
%     threeD0 = heat_source_model_3D_funct(time,Q1_new,tool_position,...
%         time(i),tool_position(i)/.0254, dt(i), time(i) );
%     threeD1 = heat_source_model_3D_funct(time,Q1_new,tool_position,...
%         time(i),tool_position(i)/.0254,  0,  dt(i)     );
%     end
%
%     dT0(i) = twoD0*ratio_of_2D + threeD0*(1-ratio_of_2D);
%     dT1(i) = twoD1*ratio_of_2D + threeD1*(1-ratio_of_2D);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %Solve the problem for that step
    [temp_vec, Q1_val] =
FEA_heat_source_solver(T_prev,dT0(i),dT1(i),Q1(i),dt(i) );
    Theta_new(:,i) = temp_vec;
    Q1_new(i) = Q1_val;%maybe I need to index it back a bit more???
%     Q1_new(i) = Q1(i);%turn this on to un-do the affect of the previous...

end

figure(100+ P.current.file_num)
clf
% subplot(2,1,1)
plot(time,tool_temp, Time,Theta_new(1,:),':',...
    Time,Theta_new(P.solver.node_of_interest,:),'g:', Time,Theta_new(end,:) )
% legend('Actual','Prediction Plate','Prediction Tip','Prediction End
Tool','location','southeast')
h = axis;
h(3) = 350; h(4) = 550; h(1) = 0;
axis(h);
```

```
% subplot(2,1,2)
% plotyy(time,power, time,x_vel)
% axis([0 h(2) 2000 4000])

actual = tool_temp(traverse_index(1):traverse_index(2));
pred2 = Theta_new(P.solver.node_of_interest,:)';
pred = pred2(traverse_index(1):traverse_index(2));
error = sum( (actual - pred).^2 );
sse = double(error);

fh = figure(200+ P.current.file_num)
clf
% subplot(2,1,1)
plot(time,tool_temp,    Time,Theta_new(P.solver.node_of_interest,:),'g')
axis([0 750 380 480])
ylabel(strcat('Temperature (', char(176),'C)'))
xlabel('Time (s)')
p1 = get(fh,'position');
set(fh,'position',[p1(1) p1(2) 800 300])

% toc

end
```

## E.3 MPC Controller Files

### E.3.1 MPC_welder_heat_source.m

```
clc
clear all
format compact

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%**************Set up the parameters for the MPC
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%*********set up my global parameter for neat and easy passing of stuff
global P
param_loader %load tool, material, & other geometry
tool_mesher %mesh the tool

%*******************give a temp setpoint & IPM
P.solver.setpoint0 = 440;                ipm_setpoint0 = 9;
P.solver.setpoint = P.solver.setpoint0;

%********select the save file name
savefilename =
'thesisweldstransient_HS_003_transientparamas_longerweld_v3.mat'
savefile = 1;
notes = 'MV move penalty = 1e-5; theta = 0; bias = 1; dPdt = 1';
wait_until_weld_begins = 1;%if this = 1, the file will wait for the welder
simulation_mode = 0;%if = 0; it will NOT actually command a weld

%********************MPC inputs/constants
theta = 1;%time delay; only used in biasing
dPdt_max = 1*745;           tau = .005;
P_min_MPC = 2.5*745;          P_max_MPC = 4.5*745;
dt = [.6 .6  1  1.8  3 4 6 8 ]; %define the dt horizion:
P.current.point_depth = .18;%in INCHES!
% P.current.point_depth = .0428;%in INCHES!
P.constants.h_collar = 200;
P.constants.h_tip = 600;%wouldn't this be a function of fluid velocity (aka
%tool-to-plate velocity difference, and thus a function of the radius? idea)
% P.constants.h_tip = 4613;
P.solver.node_of_interest = 4;
P.numerics.power_increase_ratio = 1.4;
% P.numerics.power_increase_ratio = .6011;
P.numerics.MV_move_penalty = 3e-5;

theta = .01;%time delay; only used in biasing
P.mat.k = 190;
P.solver.node_of_interest = 4;
point_depth = .18 %in INCHES!
P.numerics.power_increase_ratio = 1.4;
P.constants.h_collar = 500;
P.constants.h_tip = 4000;%wouldn't this be a function of fluid velocity (aka
```

```
%******************define the initial state of the system
P.current.T_chiller = -1;%8;%
P.current.T_collar  = P.current.T_chiller;%-4;%8;%
P.current.T_ambient = 12+0;%as the NON CHILLED TOOL measures room temp!
P.constants.constrain_back_end = 1;% if = 1, then back node's Temp =
T_chiller

%******************Biasing and Model correction
%BIASING:
alpha_bias = 1;%0 = no biasing, 1 = *super* aggressive biasing
alpha_bias_SS = alpha_bias;%.5;%for when temp < 5 degrees off
if alpha_bias > 1; warning('Bias is >1 and is thus too high...'); end
bias_time_start = 45;%time to start doing the bias at

%******************Automatic setpoint and jumps
jump_active = 0;%if 0, then it will not do this; if 1 then it will
jump_time_profile =   [-111 40 80 120 160 190 1000]+111;%this must be ONE
longer than the next, and go from 0 to a high number
jump_temp_setpoint =  [0 20  0   0   0   0];%compared to nominal value; deg C
jump_traverse_speed = [0  0  0  -3   3   0];%compared to nominal value; ipm

%******************Automatic ramp
ramp_active = 1;%if 0, then it will not do this; if 1 then it will
ramp_pos = [0 2];
ramp_vel = [.5 9];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%**************Do weld setup items (calculate stuff, connect to opc, etc)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%******************Set up/define other stuff
dP_max = dPdt_max.*dt;%this will need to change to vary for the different dt
pausetime = .02;%if you want to slow down the MPC loop; %actually, this des
something else now...

%make a vector for assumed initial temperatures before the weld start
T_start = P.current.T_collar*ones(size(P.geo.dx,1)+1,size(P.geo.dx,2));
T_start(1) = P.current.T_ambient;

%do some calculations
n = length(dt);
P.quick.dT_hist = dt*0; total_horizion = sum(dt);
dt_sum = dt; P.quick.dt = dt; P.quick.dt_sum = dt_sum;
for i = 1:numel(dt);     dt_sum(i) = sum(dt(1:i)); end;

%**************Set up save file path and also precheck for overwriting
path_gen %the exisiting file check is not working... :-(

%**************Create an opcda object, connect to the server, set up tags
if simulation_mode == 1 warning('You are connecting to a FAKE opc server!')
else opc_setup
end
```

```
%************** wait for weld to begin before jumping in
if wait_until_weld_begins == 1 && simulation_mode ~= 1
    disp('Matlab will now wait until the machine begins the weld')
    disp(' ')
    i4 = 0; welder_on = 0; %initialize counter and the holder
    while welder_on == 0
        i4 = i4+1;
        weld_in_process_struct = read(weld_in_process_itm, 'device');
        weld_in_process  = weld_in_process_struct.Value;

        %if it is welding, then get out of this loop
        if weld_in_process == 1
            welder_on = 1;
            disp('The welder is now going')
            break
        end

        pause(.1)%pause for .1 seconds before going on -- no reason to loop
like crazy
    end
end


funct = 'obj_funct_heat_source';%function the the MPC will minimize

%********************give starting values**********************
x0 = 3.5*745*ones(n,1);%give it a starting guess for power

%*************************Set up constraints for MPC**********************

%lower and upper bounds on optimization variables (ie: high & low power)
lb = zeros(size(x0)) + P_min_MPC;
ub = lb - P_min_MPC + P_max_MPC;

%set up constraint matrix b for the dPdt < |value|
b_top = [dP_max(1:end-1), dP_max(1:end-1)]' .* ones(2*n -2,1);%for most of
them

%set up constraint matrix A for the dTdt < |value|
diag = eye(n-1);
diag1 = cat(2,diag,zeros(n-1,1));
diag2 = cat(2,zeros(n-1,1),-diag);
diag_top = diag1+diag2;

%add the same constraint but to the previous input (aka link them together)
prev_input_constraint = cat(2,[1;-1],zeros(2,n-1));
A = cat(1,diag_top,-diag_top,prev_input_constraint);

%********************set up parameters for the MPC optimizer***************
alg = 'sqp';
% alg = 'active-set';
% alg = 'interior-point';
%'diffminchange',1e-3,%this can help it pick out the derivatives that
```

```
%actually matter
options = optimset('Algorithm',alg,'maxiter',1000,'tolfun',1e-4,...
    'tolcon',1e-3,'diffminchange',1e-3,'display','off','MaxFunEvals',400);
%maybe I could have MaxFunEvals change so that if the weld is undergoing a
%period of a massive prediction change then it can take a bit more time,
%but I am not sure if that is really desired as we want quick data feedback

bias = 0;%zeros(n,1);
current_MPC_bias = 0;%initialize

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%**************Do the actual predictions & control
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%initialize this
vel_horizion = ipm_setpoint0*.0254/60*ones(size(dt));%in metric!
dp = dt.*vel_horizion;
dp_sum = dp;
for i = 1:numel(dt);        dp_sum(i) = sum(dp(1:i));            end;

%turn it on by setting to active (send this to the PLC):
if simulation_mode ~= 1 write(MPC_ACTIVE_itm_write, 1);
else next_MPC_pwr = 0; next_MPC_ipm = 0;
end

MPC_solve = 1;%we will need a switch later to un-toggle it once the MPC
section is finished
MPC_go = 0; MPC_kill = 0;
T_prev = T_start;
T_calc_hist = T_prev;
i2 = 0;

tic%do NOT put another "tic" from here on out as it will really mess with
things
while MPC_solve == 1;
    disp(' ')
    i2 = i2+1

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %**************Get the loop time and current tag values
    loop_time(i2,1) = toc;%get the time of the loop
    if simulation_mode == 1; fake_opc_group_values;
    else read_opc_group_values%this script reads a group of values from the
PLC
    end

    %check if we need to do jumps in temp setpoint or in imp
    if jump_active == 1
        for i4 = 1:length(jump_temp_setpoint)
            if jump_time_profile(i4) <= loop_time(i2,1) && ...
                    jump_time_profile(i4+1) >= loop_time(i2,1)
                ipm_setpoint = ipm_setpoint0 + jump_traverse_speed(i4);
                temp_setpoint_jump = jump_temp_setpoint(i4);
```

```
            end
        end
    else
        temp_setpoint_jump = 0;
        ipm_setpoint = ipm_setpoint0;
    end

    %check if we need to do ramps in imp
    if i2 == 1
        xpos0 = x_pos_current;%get a zero position
    end

    if ramp_active == 1;%if 0, then it will not do this; if 1 then it will
        if x_pos_current <= ramp_pos(2) + xpos0;

            frac = (x_pos_current - xpos0)/(ramp_pos(2) - ramp_pos(1) )
            ipm_setpoint = ramp_vel(1) + frac*(ramp_vel(2) + ramp_vel(1))

        else%if we are past the ramp, then command like normal
            ipm_setpoint = ramp_vel(2);
        end

    end

    P.solver.setpoint = P.solver.setpoint0 + double(RPM_offset_current) +
temp_setpoint_jump;
    P.solver.setpoint_trajectory = P.solver.setpoint + exp(-
dt_sum/tau)*(tooltemp_current - P.solver.setpoint);
    data.x_pos(i2,1) = x_pos_current;%I need to get this now so I can have
the first one


    if i2 == 1;
        loop_dt_est = .2;%this can/should be updated as I go; maybe via
biasing method?
        dt_loop = 0;       loop_time(1,1) = 0;
    else
        dt_loop = loop_time(i2,1) - loop_time(i2-1,1)
        loop_dt_est = loop_dt_est*.8 + .2*dt_loop;
    end
    data.dt_loop(i2,1) = dt_loop;


    %update the velocity horizion based upon actual velocity
    if weld_traverse_current == 1
        vel_horizion = (ipm_setpoint +
x_vel_offset_current)*.0254/60*ones(size(dt));%in metric!
        dp = dt.*vel_horizion;
        dp_sum = dp;
        for i = 1:numel(dt);        dp_sum(i) = sum(dp(1:i));            end;
    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    %**************calculate the nodes & nodal area in contact with the plate

    P.current.tool_depth = -.0254*z_pos_current; %tool depth & z_pos are
different sign conventions
    if weld_traverse_current == 0 && weld_engage_current == 0;
        P.current.tool_depth = -.0254*1;%say that it is pulled out by 1"
    end; %so when it comes out contact stops
    %     if P.current.tool_depth <= 0;  powerb(i) = 0; end;%0 the power when
we are not in contact;

%     P.current.tool_depth = .0254*.21;%ONLY for debuging purposes
    tool_contact_area

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %**************adjust previous cmd pwr values before the MPC takes over

    %if the MPC has not taken over yet, approximate the cmd pwr
    if MPC_enable_current == 0
        power_current = spindle_power_current*745 * .98;%
    else %if we are the in the MPC section, then use the latest power cmd
        power_current = MPC_pwr_cmd_current*745;
    end

    if power_current < .001; %we can't have negative power or 0...
        power_current = .001;
    end

    P.current.power = power_current;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %*****************************Do biasing*****************************

    %this assumes that the matlab code wrote & implemented what it actually
    %wanted to
    alpha = alpha_bias;
%     if tooltemp_current <= (P.solver.setpoint + 5) && tooltemp_current >=
(P.solver.setpoint - 5)
%         %if withith 5 degrees, then a tighter bias
%         alpha = alpha_bias_SS;
%     end
    if i2 >= 2 && weld_traverse_current == 1 && loop_time(i2,1) >=
bias_time_start %check to make sure we are far enough along
%         raw_bias (i2,1) = tooltemp_current -
P.solver.future_temps_raw(P.solver.node_of_interest);

%         previous_predicted_temperature =
T_calc_hist(P.solver.node_of_interest,i2);

%if the theta is too low, then I may have to do some adjustments....

neg_time = data.time(1:i2-1)-loop_time(i2,1);
if neg_time(end) >= -theta
        previous_predicted_temperature = interp1(data.time(1:i2-1)-
loop_time(i2,1),...
            T_calc_hist(P.solver.node_of_interest,1:i2-1),-theta);
else
```

```
    previous_predicted_temperature = interp1(data.time(1:i2-1)-
loop_time(i2,1),...
            T_calc_hist(P.solver.node_of_interest,1:i2-1), neg_time(end) );
end
        raw_bias (i2,1) = tooltemp_current - previous_predicted_temperature;
        bias(i2,1) = alpha*raw_bias(i2,1) + (1-alpha)*bias(i2-1,1);

    else
        raw_bias(i2,1) = 0;
        bias(i2,1) = 0;
    end
    P.current.bias = bias(i2,1);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %************************get MPC constraints ready*******************

    %re-do the previous constraint
    x_previous = power_current;
    b_bottom = [x_previous + dP_max(1)*loop_dt_est/dt(1);...
        - x_previous + dP_max(1)*loop_dt_est/dt(1)];%link 1st of my solution
x to previous x
    b = cat(1,b_top,b_bottom);
    b = double(b);

    if weld_traverse_current == 0%so that for the first steps it is not
unsolvable
        lb_current = lb*0+.25*745;
    else
        lb_current = lb;
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %************************get data ready for predictions***************

    %extend values a bit into the future for my heat source method
    time2(i2) = loop_time(i2,1);
    time2(i2+1) = time2(i2) + loop_dt_est;

    power2plate(i2) = power_current;
    power2plate(i2+1) = power_current;%this must be power *into* the plate,
    %except of course for the last 2 values!

    tool_pos2(i2) = x_pos_current*.0254 - data.x_pos(1,1)*.0254;
    tool_pos2(i2+1) = tool_pos2(i2) + x_vel_current*.0254/60*loop_dt_est;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %*****************calculate the effect of the last step***************

    %do the heat source & FEA for the last step through when we expect to
    %implement the next step
    if i2 == 1%if so, then there is nothing going into the far history
anyways...
        dT0 = 0;
    else
        dT0 = heat_source_model_3D_funct(time2,power2plate,tool_pos2,...
```

```
            time2(i2+1),tool_pos2(i2+1)/.0254, loop_dt_est, time2(i2+1)
);%convert the desired position to meters!
    end
    dT1 = heat_source_model_3D_funct(time2,power2plate,tool_pos2,...
        time2(i2+1),tool_pos2(i2+1)/.0254,  0,  loop_dt_est     );%convert
the desired position to meters!

    [Theta_new, Q1_val] = FEA_heat_source_solver...
        (T_prev, dT0, dT1, power2plate(i2+1), loop_dt_est);

    P.current.T_prev  = Theta_new;
    power2plate(i2) = Q1_val;     power2plate(i2+1) = Q1_val;
    T_calc_hist(:,i2+1) = Theta_new;
    T_prev = Theta_new;


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %*****************Precalculate things for the obj funct***************

    %calculate the influence of the historical stuff
    for i = 1:numel(dt)

        P.quick.dT_hist(i) = heat_source_model_3D_funct(time2, power2plate,
tool_pos2,...
            time2(i2+1) + dt_sum(i),    tool_pos2(i2+1)/.0254 +
dp_sum(i)/.0254,...
            dt_sum(i),     time2(i2+1) + dt_sum(i) );
    end

    %do the "unit Q" estimations for the entire horizon
    P.quick.Q_2_dT_mat = zeros(n, n);
    unit_power = 1;
    for i = 1:length(dt)
        for j = 1:1%the first row needs to be set up slightly differently
            P.quick.Q_2_dT_mat(i,j) = heat_source_model_3D_funct...
                ([0; dt_sum(j)], [unit_power; unit_power], [0; dp_sum(j)],...
                dt_sum(i), dp_sum(i)/.0254, ...
                -dt(j)+dt_sum(i), 0+dt_sum(i)  );
        end
        for j = 2:i
            P.quick.Q_2_dT_mat(i,j) = heat_source_model_3D_funct...
                ([dt_sum(j-1); dt_sum(j)], [unit_power; unit_power],
[dp_sum(j-1); dp_sum(j)],...
                dt_sum(i), dp_sum(i)/.0254, ...
                -dt_sum(j)+dt_sum(i), -dt_sum(j-1)+dt_sum(i) );
        end
    end
    P.quick.Q_2_dT_diag = P.quick.Q_2_dT_mat.*eye(length(dt));
    P.quick.Q_2_dT_mat_no_diag = P.quick.Q_2_dT_mat - P.quick.Q_2_dT_diag;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %**************Do the MPC calculations & implementation solution*******


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    %****************************Solve the Problem************************
    MPC_go = 1;

    %Optimize the horizon ; everything into fmincon MUST be a double; no
singles
    MPC_horizion =
fmincon(funct,x0,A,b,[],[],lb_current,ub,[],options);%optimize
    MPC_power(i2,:) = MPC_horizion';%get the first value; this is sort of a
storage array
    P.solver.future_temps_raw;
    P.solver.future_temps_biased;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %**********************implement the MPC's solution*******************

    write_time = .015;%this is the estimated delay in writing
    loop_time2(i2,1) = toc;

    %this is the time from which's value I want to now command
    time_diff = loop_time2(i2,1) - loop_time(i2,1) + write_time;

    if time_diff < loop_dt_est
        pause( min(loop_dt_est - time_diff, pausetime) )
    end

    %Ready the values to write
    MPC_pwr_val_to_write = MPC_horizion(1);
    MPC_ipm_val_to_write = ipm_setpoint;%do NOT add in the velocity offset
    %here; the HMI already does that, so adding here would be twice as much
    %as is actually needed

    %Implement the first step (or skip if in simulation mode)
    if simulation_mode == 1;
        next_MPC_pwr = MPC_pwr_val_to_write/745;
        next_MPC_ipm = MPC_ipm_val_to_write;
    else
        write(write_grp, {MPC_pwr_val_to_write/745, MPC_ipm_val_to_write,1})
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %****************Re-do the initial guess for the next loop*************

    %now re-do the initial guess so that is is super close to what it
    %should be
%     for i = 1:n-1
%         x0(n,1)
%        dt_sum
%     end

    x0(1:n-1,1) = MPC_horizion(2:n,1);
    x0(n,1) = MPC_horizion(n,1);%duplicate the last one as a guess

    log_data
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %**********************Do Plots***************

    %Figure 1
    %**********************Temperature sub-Plot***************
    figure(1)
    subplot(2,1,1)
    plot(data.time, data.tooltemp,'b',...
        data.time, data.temp_calc_node,'g',...
        dt_sum+data.time(i2,1), P.solver.future_temps_raw,'r*',...
        dt_sum+data.time(i2,1), P.solver.future_temps_biased,'r--',...
        data.time, data.setpoint,'m--',...
        data.time(i2,1), P.solver.setpoint,'ro',...
        (data.time(i2,1) + dt_sum), P.solver.setpoint_trajectory,'m:')
    if data.tooltemp(i2) > 400 && data.temp_calc_node(i2) > 400 &&...
          data.tooltemp(i2) < 480 && data.temp_calc_node(i2) < 480
    axis([0, data.time(i2,1)+ dt_sum(end)+ 1 , 400, 480])
    elseif data.tooltemp(i2) > 380 && data.temp_calc_node(i2) > 380 &&...
          data.tooltemp(i2) < 500 && data.temp_calc_node(i2) < 500
    axis([0, data.time(i2,1)+ dt_sum(end)+ 1 , 380, 500])
    elseif data.tooltemp(i2) > 350 && data.temp_calc_node(i2) > 350 &&...
          data.tooltemp(i2) < 530 && data.temp_calc_node(i2) < 530
    axis([0, data.time(i2,1)+ dt_sum(end)+ 1 , 350, 530])
    end
    title('Tool Temperature')
%     legend('Temperature History','Temp Predicted','Temp Predicted +
Bias',...
%         'Setpoint','location','southoutside')

   %**********************Power sub-Plot***************
    subplot(2,1,2)
    plot(data.time,data.MPC_pwr_cmd*745,'g',...
         data.time,data.spindle_power*745,'b',...
         data.time(i2,1)+[dt_sum; [0,dt_sum(1:end-1)]], [MPC_horizion,
MPC_horizion]','r' )
    if data.MPC_pwr_cmd(i2) > 2500 && data.MPC_pwr_cmd(i2) < 3500
    axis([0, data.time(i2,1)+ dt_sum(end)+ 1 , 2500, 3500])
    else
    axis([0, data.time(i2,1)+ dt_sum(end)+ 1 , 2*745, P_max_MPC*1.1])
    end
    title('Previous & Future commanded powers')



    figure(6)
    plotyy(data.time,data.spindle_speed,data.time,data.motor_torque)
    title('Spindle Speed & Torque')

    if alpha_bias > 0
        figure(3)
        plot(data.time, data.bias)
        title('Bias')
    end

%     toc
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %**********************Kill the loop after MPC ends***************
```

```
    %if it WAS enabled and now it isn't, then break the loop
    if i2 > 3 %on the first loop i-1 = 0, and so will mess up the below logic
        if data.MPC_enable(i2-1,1) == 1 && data.MPC_enable(i2,1) == 0
            MPC_kill = 1;
        end
    end

    if MPC_kill == 1
        break
    end


end
disp(strcat('Total Time = ',num2str(toc) ))


% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% disp('File has run up to the "return" stop');
% return
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%**********************Save the data***************

%put the MPC parameters into a structure
controller_params.temp_setpoint = P.solver.setpoint;
controller_params.ipm_setpoint = ipm_setpoint;
controller_params.dPdt_max = dPdt_max;
controller_params.P_max_MPC = P_max_MPC;
controller_params.P_min_MPC = P_min_MPC;
controller_params.n = n;
controller_params.dt = dt;
% controller_params.theta = theta;
controller_params.alpha_bias = alpha_bias;
controller_params.pausetime = pausetime;
controller_params.notes = notes;
controller_params.power_increase_ratio = P.numerics.power_increase_ratio;
controller_P = P;
controller_params.alpha = alpha;
% controller_params.error_power = error_power;
% controller_params.MV_change_penalty = MV_change_penalty;
% controller_params.model_correct_limits = model_correct_limits;
% controller_params.model_correct_time = model_correct_time;
% controller_params.model_correct_penalty = model_correct_penalty;

date_time = clock;

%save the data collected
if savefile == 1
    save(exportname,'data','controller_params','notes','date_time','P')
```

```
else
    warning('Data has not been saved! Do this manually or it will be lost.
Remember to change the save file names or it will overwrite previous data')
end

figure(10)
plot(data.time,data.tooltemp,data.time,data.MPC_pwr_cmd*5+460)
title('Powers & Temperatures')

figure(11)
plot(data.time, data.x_velocity_actual)
title('Traverse Speed')

figure(12)
plot(data.time, data.dt_loop)
title('Loop Time')

figure(13)
clf
plot(data.time, data.tooltemp,'b',...
    data.time, data.temp_calc_node,'g',...
    dt_sum+data.time(i2,1), P.solver.future_temps_raw,'r*',...
    dt_sum+data.time(i2,1), P.solver.future_temps_biased,'r--',...
    data.time, data.setpoint,'m--',...
    data.time(i2,1), P.solver.setpoint,'ro')
hold on
plot(data.time,(data.MPC_pwr_cmd*745-3200)/50 + 450,'r')
axis([0 data.time(end) 420 480])

disp('File Finished')
```

### E.3.2 heat_source_model_3D_funct.m

```
function [Theta] =
heat_source_model_3D_funct(time_hist,power_hist_W,pos_hist_m, ...
    time_desired,pos_desired_inch, t1,t2)
%this function will predict the temp at a point in time & space given
%historical heat intensities, times, and locations, all of which must be
%METRIC!

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%load variables from global and rename ones that are sent in
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%global variables
global P; k = P.mat.k; ro = P.mat.ro; cp = P.mat.cp; a = P.mat.a;
a = k/(ro*cp);
%historical values (re-name them)
time = time_hist; tool_position = pos_hist_m;
power = power_hist_W*P.numerics.power_increase_ratio; %boost the power here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%end the section where it will get it's data from
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%I want properties at: (x,y,z,t) (this is for .125 depth)
M = [pos_desired_inch,0,-
P.current.point_depth,time_desired].*[.0254,.0254,.0254,1];

% %with this, I will both cut the time and I will properly and fix the zeros
% t_steady_state = P.numerics.heat_source_integration_t_steady_state;
%
% if t_steady_state > time_desired;   time_total = time_desired;
% else                                time_total = t_steady_state;
% end

[dt, t_i, time_dividers] = dt_time_spacer(t1,t2);

%get some uncut data values
t_i_uncut = time - M(4);
q_i_uncut = power;

%extend these just a tiny to hopefully avoid NaN errors during
%interpolation later on...
t_i_uncut(1) = t_i_uncut(1) - .0001;
t_i_uncut(end) = t_i_uncut(end) + .0001;

%interpolate the data to where I want it
%(having extra data that I want is ok; ie I can feed the function times from
%0 to 250, shift that to -150 to 100, and still get the -150 to 0 range that
I want)

toc1 = toc;
q_i = interp1(t_i_uncut,q_i_uncut,-t_i);
tool_pos = interp1(t_i_uncut,tool_position,-t_i);
P.timer.interp = P.timer.interp + toc - toc1;
```

```
R_i_sq = ( (M(1)-tool_pos).^2 + M(2)^2 + M(3)^2);
% keyboard
%now do temp prediction at the point of interest
Theta = 1/(.5*cp*ro*(4*pi*a)^1.5)*( sum(q_i.*dt./t_i.*exp(-
R_i_sq./(4*a*t_i))) );

if P.stop == 1
    keyboard
end
% Q = q_i(1)
% T = t_i(1)
% M1 = M(1)
% TP = tool_pos(1)
% M_TP = M(1) - tool_pos(1)
% R = R_i_sq(1)
%
%
% keyboard
end
```

### E.3.3 FEA_heat_source_solver.m

```
function [Theta_new, Q1_new] = FEA_heat_source_solver(T_prev,dT0,dT1,Q1,dt)
%FEA solver for the Heat Source problem coupled with a tool

%pass this the previous temps, dT1, dT0, and dt
%it will return a new set of temps

%declare P to be global to that is can pass & receive variables
global P

toc1 = toc;
%unpack some stuff that will be used here
dx = P.geo.dx;      V = P.geo.V;
h_collar = P.constants.h_collar*ones(size(P.geo.dx));
h_tip = P.constants.h_tip*ones(size(P.geo.dx));

%now we get into stuff that would usually be in the solver
k_node = 26*ones(size(P.geo.dx));%this could be variable for a PCBN tool
k_w = k_node;       k_e = k_node;

ro = 7750*ones(size(P.geo.dx));%this could be variable for a PCBN tool
c = 460*1*ones(size(P.geo.dx));%this could be variable for a PCBN tool
% c = 46*1*ones(size(P.geo.dx));%this could be variable for a PCBN tool

%calculate the stuff to put into the solver for all of the nodes
a_p_0 = ro.*c.*V./dt;
a_w = k_w./P.geo.del_x_w.*P.geo.A_w;
a_e = k_e./P.geo.del_x_e.*P.geo.A_e;
a_0 = h_tip.*P.current.contact_metal_area;%this goes in the first column
a_c = h_collar.*P.geo.contact_collar_area;

%do some boundary conditions
a_w(1) = 0;
a_e(end) = 0;

%calcuate the a_p coefficients which should now include the correct a_w etc
a_p = a_p_0 + a_e + a_w + a_0 + a_c;
b = a_c*P.current.T_collar  +  a_p_0.*T_prev(2:end);

%**************************************************************************

%now all the indexing changes...we will now add another row of stuff
A = zeros(P.geo.num_nodes + 1,P.geo.num_nodes + 1);

%pack the matrix
%remember that we have different lengths here, so a_e & a_w of element 20
%goes in spot #21
for i = 1:P.geo.num_nodes
    A(i+1,i+1) = a_p(i);
end
for i = 2:P.geo.num_nodes
```

```
    A(i+1,i) = -a_w(i);
    A(i,i+1) = -a_e(i-1);
end

%assign values to the left-hand side of the equation...
B(2:P.geo.num_nodes+1,1) = b;

%if we are constraining the back node to T_chiller, change A & b's bottom row
if P.constants.constrain_back_end == 1
    A(end,:) = 0;
    A(end,end) = 1;
    B(end) = P.current.T_chiller;
end
%**************************************************************************
%now add in heat-source method terms

%create the stuff for the first row/the in-plate calculations
A(1,1) = 1 + dT1/Q1*sum(h_tip' .*P.current.contact_metal_area');
A(1,2:end) = -dT1/Q1 *h_tip' .*P.current.contact_metal_area' ;
B(1,1) = dT0 + dT1 + P.current.T_ambient;

%now link the other rows (ie the tipe nodes) to the metal also.
A(2:end,1) = -a_0;

%**************************************************************************
%Solve the problem
% keyboard
Theta_new = A\B;
T_0 = Theta_new(1);

Q1_new = Q1 - sum(a_0.*(T_0 - Theta_new(2:end)));

if P.stop == 1
    keyboard
end

P.timer.FEA_solver = P.timer.FEA_solver + toc - toc1;
```

## E.4 Common Files

### E.4.1 opc_setup.m

```
%this connects to the OPC server and creates some read & write groups

%this is NOT to be a function; otherwise it will not share the same
%workspace as the driver function

%having this in another file instead of written inline ensures that the same
%setup will be logged & in the same way from one way welding file to
%another

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%**************Create an opcda object, connect to the server, set up tags
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Create an opcda object associated with the server and connect to the server
da = opcda('localhost','Matrikon.OPC.AllenBradleyPLCs.1');
connect(da); disp(' ');
matrikonInfo = opcserverinfo(da);
disp(' ')

%Create group objects to manage the required items.
read_grp = addgroup(da, 'Matlab_PRBS_read_Group');
write_grp = addgroup(da, 'Matlab_PRBS_write_Group');
min_max_power_grp = addgroup(da, 'Matlab_min_max_power_Group');
weld_stage_grp = addgroup(da, 'Matlab_weld_stage_Group');
%I should create one for the weldstage & stuff to get from the machine off
%of the PLC data and such

%Add items to the read group -- if anything is re-ordered, change that below
in the loop too
% A = serveritems(da); %use to check to see what ALL the tags which are on
the PLC are(this is a huge and takes a while...):
MPC_PRBS_itm = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:BOOL:MPC_PRBS.VALUE');
MPC_ENABLE_itm = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:BOOL:MPC_ENABLE.VALUE');
MPC_ACTIVE_itm = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:BOOL:MPC_ACTIVE.VALUE');
MPC_pwr_cmd_itm_read = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-
L55-A 1756-M12-A LOGIX5555:REAL:MPC_PWR_CMD.VALUE');
MPC_ipm_cmd_itm_read = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-
L55-A 1756-M12-A LOGIX5555:REAL:MPC_IPM_CMD.VALUE');

ToolTemp_itm = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:Temperature:TOOLTEMPERATURE.FA.VALUE');
Spindle_power_itm = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-
L55-A 1756-M12-A LOGIX5555:REAL:SPINDLEPOWER.VALUE');
RPM_vel_offset_read = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-
L55-A 1756-M12-A LOGIX5555:REAL:WELDOFFSET_RPM.VALUE');
```

```
motor_speed = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:REAL:SPINDLEMOTORSPEEDFB.VALUE');
motor_torque_mtr = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-
A 1756-M12-A LOGIX5555:REAL:SPINDLEMOTORTORQUEFB.VALUE');

X_vel_itm_read = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:AxisPosition:XPOS.VELOCITY_INCH.VALUE');
X_vel_offset_itm_read = additem(read_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A LOGIX5555:REAL:WELDOFFSET_VEL.VALUE');
X_pos_rel_itm = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:AxisPosition:XPOS.INCH_REL.VALUE');
Z_pos_rel_itm = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:AxisPosition:ZPOS.INCH_REL.VALUE');
X_force_lbf = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:AxisForce:XFORCE.LBF.VALUE');
Y_force_lbf = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:AxisForce:YFORCE.LBF.VALUE');
Z_force_lbf = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:AxisForce:ZFORCEAVG.LBF.VALUE');
Z_vel_itm_read = additem(read_grp, 'Allen Bradley via Ethernet/IP:1756-L55-A
1756-M12-A LOGIX5555:AxisPosition:ZPOS.VELOCITY_INCH.VALUE');

%add to the write group
MPC_pwr_cmd_itm_write = additem(write_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A LOGIX5555:REAL:MPC_PWR_CMD.VALUE');
MPC_ipm_cmd_itm_write = additem(write_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A LOGIX5555:REAL:MPC_IPM_CMD.VALUE');
MPC_ACTIVE_itm_write = additem(write_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A LOGIX5555:BOOL:MPC_ACTIVE.VALUE');

%add to the min/max power group
min_power_itm_write = additem(min_max_power_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A
LOGIX5555:REAL:MINIMUMALLOWEDSPINDLEPOWERHP_CMD.VALUE');
max_power_itm_write = additem(min_max_power_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A
LOGIX5555:REAL:MAXIMUMALLOWEDSPINDLEPOWERHP_CMD.VALUE');

%add to the weld stage
weld_in_process_itm = additem(weld_stage_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A LOGIX5555:BOOL:WELDINPROCESS.VALUE');
weld_engage_itm = additem(weld_stage_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A LOGIX5555:BOOL:WELD_ENGAGE.VALUE');
weld_traverse_itm = additem(weld_stage_grp, 'Allen Bradley via
Ethernet/IP:1756-L55-A 1756-M12-A LOGIX5555:BOOL:WELD_TRAVERSING.VALUE');

%now do a read to get some initial values so that I can start out the MPC
multi_value = read(read_grp); %perform a multi-item read from the group
%store the values now:
tooltemp_current = multi_value(6).Value;
x_vel_current = multi_value(10).Value*60;
x_vel_offset_current = multi_value(11).Value;
RPM_offset_current = multi_value(8).Value;
initial_temp = tooltemp_current;
```

### E.4.2 read_opc_group_values

```
%this reads a group of values from the OPC server and splits up the

%this is NOT to be a function; otherwise it will not share the same
%workspace as the driver function; do NOT put "clc" or "clear all" at the
%top

%having this in another file instead of written inline ensures that the same
%setup will be logged & in the same way from one way welding file to
%another

%first get the most current data from the PLC
multi_value = read(read_grp); %perform a multi-item read from the group
MPC_PRBS_current = multi_value(1).Value; %store the values somewhere
MPC_enable_current = multi_value(2).Value;
MPC_active_current = multi_value(3).Value;
MPC_pwr_cmd_current = multi_value(4).Value;
MPC_ipm_cmd_current = multi_value(5).Value;

tooltemp_current = multi_value(6).Value;
spindle_power_current = multi_value(7).Value;
RPM_offset_current = multi_value(8).Value;
motor_speed_current = multi_value(9).Value;
motor_torque_current = multi_value(10).Value;

x_vel_current = multi_value(11).Value*60;
x_vel_offset_current = multi_value(12).Value;
x_pos_current = multi_value(13).Value;
z_pos_current = multi_value(14).Value;
x_force_lbf_current = multi_value(15).Value;
y_force_lbf_current = multi_value(16).Value;
z_force_lbf_current = multi_value(17).Value;
spindle_speed_current = motor_speed_current/2.5;
z_vel_current = multi_value(18).Value*60;

%get the weld stage
multi_value2 = read(weld_stage_grp);
weld_in_process_current = multi_value2(1).Value;
weld_engage_current = multi_value2(2).Value;
weld_traverse_current = multi_value2(3).Value;
```

### E.4.3 log_data.m

```
%this put logged data into a nice structure to later save it

%this is NOT to be a function; otherwise it will not share the same
%workspace as the driver function; do NOT put "clc" or "clear all" at the
%top

%having this in another file instead of written inline ensures that the same
%setup will be logged & in the same way from one way welding file to
%another

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%**********************Set up data for saving & plotting***************

%record data into a structure
%record what this loops says
data.time(i2,1) = loop_time(i2,1);
data.time2(i2,1) = loop_time2(i2,1);
data.MPC_pwr_cmd_matlab(i2,1) = MPC_pwr_val_to_write;
data.MPC_ipm_cmd_matlab(i2,1) = MPC_ipm_val_to_write;

%and record what the PLC/OPC sever told me for this loop
data.MPC_PRBS(i2,1) = MPC_PRBS_current;
data.MPC_enable(i2,1) = MPC_enable_current;
data.MPC_active(i2,1) = MPC_active_current;
data.MPC_pwr_cmd(i2,1) = MPC_pwr_cmd_current;
data.MPC_ipm_cmd(i2,1) = MPC_ipm_cmd_current;

data.tooltemp(i2,1) = tooltemp_current;
data.spindle_power(i2,1) = spindle_power_current;
data.power_offset(i2,1) = RPM_offset_current/100;
data.spindle_speed(i2,1) = spindle_speed_current;
data.motor_torque(i2,1) = motor_torque_current;

data.x_velocity_actual(i2,1) = x_vel_current;
data.x_vel_offset(i2,1) = x_vel_offset_current;
data.x_pos(i2,1) = x_pos_current;
data.z_pos(i2,1) = z_pos_current;
data.x_force_lbf(i2,1) = x_force_lbf_current;
data.y_force_lbf(i2,1)= y_force_lbf_current;
data.z_force_lbf(i2,1) = z_force_lbf_current;

data.weld_in_process(i2,1) = weld_in_process_current;
data.weld_engage(i2,1) = weld_engage_current;
data.weld_traverse(i2,1) = weld_traverse_current;

%     data.temps_calculated(:,i2) = P.solver.future_temps_raw;
data.T_calc_hist(:,i2) = T_calc_hist(:,i2);
```

### E.4.4 param_loader.m

```
function [] = param_loader
%Loads geometric & material parameters into the P variable

%If I want the intermediate/non-global variables to exist outside of this
%then it should not be a function; otherwise a function is nice and clean

%the point of this function is to load stuff into the variable workspace
%and not have that clutter up your main code/re-define the same stuff
%everywhere

global P %this is my global parameter to pass things back and forth
P.description = 'P is a variable that is used to easily pass stuff between
functions';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P.constants.description = 'P.constants is for constants that I
determine/specify';
P.constants.h_collar = 100;
P.constants.h_tip = 2000;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%tool geometry

%dimensions of the tool
d1 = 1.5e-3; d2 = 8.5e-3; d3 = 25.4e-3;
h1 = 0e-3; h2 = 4.5e-3; h3 = 1.8e-3;
l_total = 94.5e-3;
l_to_tool_holder = 25e-3;%distance until the tool holder (ie when the
convection starts)
%node
tip_nodes = 4; shoulder_nodes = 2;
after_shoulder_node = 1;%the thickness of the first node after the shoulder
relative to the last should node
node_growth = 1.2;%this is an exponential growth function

P.geo.description = 'P.geo is for all geometry related items that DO NOT
Change during a weld';
P.geo.d1 = d1; P.geo.d2 = d2; P.geo.d3 = d3;
P.geo.h1 = h1; P.geo.h2 = h2; P.geo.h3 = h3;
P.geo.l_total = l_total; P.geo.l_to_tool_holder = l_to_tool_holder;
P.geo.tip_nodes = tip_nodes; P.geo.shoulder_nodes = shoulder_nodes;
P.geo.after_shoulder_node = after_shoulder_node; P.geo.node_growth =
node_growth;
P.geo.collar_start = 1.3*.0254;%inches from tip of the pin;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%mat properties
P.mat.description = 'P.mat is for material properties of the plate and plate
parameters';
% P.mat.a = 70/ 10^6;%thermal diffusivity in m^2/s
P.mat.k = 180;%thermal conductivity in W/m*K
```

```
P.mat.k = 140;%thermal conductivity in W/m*K
%http://matweb.com/search/DataSheet.aspx?MatGUID=4f19a42be94546b686bbf43f79c5
1b7d
P.mat.ro = 2800;%density of material in kg/m^3
P.mat.cp = 1100;%thermal capacity in J/kg*K
P.mat.a = P.mat.k/(P.mat.ro*P.mat.cp);
P.mat.thickness = 1/4*.0254;
P.mat.h_conv_plate = 20;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%numeric parameters (for various things & algorithms) than could be changed
P.numerics.description = 'P.numerics is for random numeric constants that
influence the behavior of the code numerically';
P.numerics.heat_source_integration_dt_set = [.005 .01 .025 .1 .2 .5 1 5
25]/2^3;
P.numerics.heat_source_integration_dt_time_set = [0 .03 .05 .1 .4 1 5 10 50
1000];
P.numerics.heat_source_integration_t_steady_state = 500;%in seconds
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P.quick.description = 'P.quick is to precalculate stuff that is needed in the
optimization function that can be calculated in advance';

P.tool_props.description = 'P.tool_props is for properties of the tool';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P.current.T_chiller = -4;
P.current.T_collar  = -4;
P.current.T_ambient = 12;
P.current.bias = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P.timer.description = 'This is for using timers to analyze code speed within
functions';
P.timer.FEA_solver = 0;
P.timer.interp = 0;

P.stop = 0;

end
```

### E.4.5 tool_mesher.m

```
function [] = tool_mesher
%this function will be called ONCE at the very beginning of each weld, and
%data from it will be used every time the FEA routines are called up. It
%both meshes the tool and calculates where the nodes contact the collar

%If I want the intermediate/non-global variables to exist outside of this
%then it should not be a function; otherwise a function is nice and clean

%**************************************************************************
% Unpack the variables to use them; OR I could re-name the entire code
%**************************************************************************
global P %specify as a global so that I can easily pass stuff

%dimensions of the tool
d1 = P.geo.d1; d2 = P.geo.d2; d3 = P.geo.d3;
h1 = P.geo.h1; h2 = P.geo.h2; h3 = P.geo.h3;
l_total = P.geo.l_total; l_to_tool_holder = P.geo.l_to_tool_holder;

%node
tip_nodes = P.geo.tip_nodes; shoulder_nodes = P.geo.shoulder_nodes;
after_shoulder_node = P.geo.after_shoulder_node;
node_growth = P.geo.node_growth;

%**************************************************************************
%create the mesh here
%**************************************************************************

%create "dx," which is the width of all of the nodes
dx(1:tip_nodes,1) = h2/tip_nodes;
dx(tip_nodes+1:tip_nodes+shoulder_nodes,1) = h3/shoulder_nodes;

j = 0;
for i = tip_nodes+shoulder_nodes+1:1000%arbitrary max
    dx(i,1) = h3/shoulder_nodes*after_shoulder_node*node_growth^j;
    j = j + 1;

    if sum(dx) >= l_total%if it is now too long
        dx(i) = l_total - sum(dx(1:i-1));%then cut the last element to size
        num_nodes = i;
        break%and terminate the loop
    end
end
P.geo.dx = dx;
P.geo.num_nodes = num_nodes;

del_x_w = zeros(size(dx)); del_x_w(1) = dx(1)/2;
del_x_e = zeros(size(dx)); del_x_e(end) = dx(end)/2;
for i = 1:length(dx) - 1
    del_x_w(i+1,1) = (dx(i) + dx(i+1))/2;
    del_x_e(i,1) = (dx(i) + dx(i+1))/2;
end
P.geo.del_x_w = del_x_w; P.geo.del_x_e = del_x_e;
```

```
position = zeros(size(dx));
position(1) = dx(1)/2;
for i = 2:length(dx)
    position(i) = position(i-1) + del_x_w(i);
end
position_w = position - dx/2;
position_e = position + dx/2;
P.geo.position = position; P.geo.position_w = position_w; P.geo.position_e =
position_e;

V = dx*(pi*d3^2/4);%we will need to re-do the tip & shoulder
dia_w = ones(size(dx))*d3;%we will need to re-do the tip & shoulder
dia_e = dia_w;%we will need to re-do the tip & shoulder

%*********************now do geometry stuff for the tip
m12 = h2/(d2-d1);
h12 = m12*d2;
h01 = h12 - h2;
P.geo.m12 = m12; P.geo.h12 = h12; P.geo.h01 = h01;

m23 = h3/(d3-d2);
h23 = m23*d3;
h03 = h23 - h3;
P.geo.m23 = m23; P.geo.h23 = h23; P.geo.h03 = h03;

for i = 1:tip_nodes
    h_tip_to_base = h01 + sum(dx(1:i));
    h_tip_to_top = h01 + sum(dx(1:i-1));

    dia_w(i) = h_tip_to_top/m12;
    dia_e(i) = h_tip_to_base/m12;

    V(i) = pi*m12/12*(h_tip_to_base^3 - h_tip_to_top^3);
    V(i) = pi*m12/12*(dia_e(i)^3 - dia_w(i)^3);
end
A_w = dia_w.^2/4*pi;
A_e = dia_e.^2/4*pi;
P.geo.A_w = A_w; P.geo.A_e = A_e; %I may not actually need these...

for i = tip_nodes+1:tip_nodes+shoulder_nodes
    h_tip_to_base = h03 + sum(dx(tip_nodes+1:i));
    h_tip_to_top = h03 + sum(dx(tip_nodes+1:i-1));

    dia_w(i) = h_tip_to_top/m23;
    dia_e(i) = h_tip_to_base/m23;

    V(i) = pi*m23/12*(h_tip_to_base^3 - h_tip_to_top^3);
    V(i) = pi*m23/12*(dia_e(i)^3 - dia_w(i)^3);
end

P.geo.V = V; P.geo.dia_w = dia_w; P.geo.dia_e = dia_e;

%*************************************************************************
```

```
%get a vector for which nodes may contact the collar/tool holder
%************************************************************************
P.geo.contact_collar = zeros(size(dx));

for i = 1:num_nodes
    if position_w(i) >= P.geo.collar_start
    %the node is all the way up on the collar
    P.geo.contact_collar(i) = 1;

    elseif position_e(i) <= P.geo.collar_start
    %the node is 0% in the collar
    %it is already 0, so do nothing
    else
    %collar_start > position_w(i) && collar_start < position_e(i)
    %the node is partially in/on the collar
    P.geo.contact_collar(i) =  (position_e(i) - P.geo.collar_start)/
(position_e(i) - position_w(i));
    end
end
%now attach an area to it
P.geo.contact_collar_area = P.geo.contact_collar .* (dia_w + dia_e)/2*pi .*
dx;
%************************************************************************
%Do plotting stuff to create a visual
%************************************************************************

%create a 5 point box (4 lines) for the collar/tool holder
c_plot_x = [P.geo.collar_start,P.geo.collar_start,
position_e(num_nodes),position_e(num_nodes) , P.geo.collar_start];
c_plot_y = [d3/2 + .001, d3/2 + .011, d3/2 + .011, d3/2 + .001, d3/2 + .001];

%modify the stuff so that it can be plotted well
position_boundary = position_w;
position_boundary(num_nodes+1) = position_e(num_nodes);
dia_boundary = dia_w;
dia_boundary(num_nodes+1) = dia_e(num_nodes);

% figure(1001)
% clc
% plot(position_boundary,dia_boundary/2,'b',position_boundary,-
dia_boundary/2,'b',[position_boundary(1),position_boundary(end)],[0,0],'k--')
% hold on
% title('Meshed Tool')
% for i = 1:num_nodes+1
%    plot( [position_boundary(i),position_boundary(i)], [dia_boundary(i)/2,-
dia_boundary(i)/2],'g')
% end
% plot(c_plot_x,c_plot_y,'k',c_plot_x,-c_plot_y,'k')
%
% hold off
% axis equal

disp('The tool has now been meshed')
disp('Nodes contacting the collar/tool holder have now been calculated')
```

### E.4.6 tool_contact_area.m

```
function [] = tool_contact_area
%Calculates which nodes contact the plate & what the contact area is; this
%is run once every cycle

%If I want the intermediate/non-global variables to exist outside of this
%then it should not be a function; otherwise a function is nice and clean

global P %Load up the current pack of variables

%P.current.tool_depth is positive when the tool is in the plate, and
%negative when it is out of the plate

if P.current.tool_depth > P.geo.h2 + P.geo.h3
    warning('Your tool appears to be deeper than the input shoulder
geometry!')
end
%get a vector for which nodes may contact the metal (ie: tip & shoulder)
P.current.contact_metal_area = zeros(size(P.geo.dx));

for i = 1:P.geo.tip_nodes+P.geo.shoulder_nodes
    %select if we want the first slope or the second one (tip or sholder
slope):
    if i <= P.geo.tip_nodes
        m_ab = P.geo.m12;%the slope ratio of diameter to
    else
        m_ab = P.geo.m23;
    end

    %calculate/determine the area in contact with the metal
    if P.current.tool_depth <= P.geo.h1
        return %if the tool is not even penetrated yet, then no need to run
anymore
    elseif P.current.tool_depth > P.geo.position_w(i) && P.current.tool_depth
< P.geo.position_e(i)
        %if the tool-metal interface is somewhere in the current node
        %then do part of the cone's area
        r_a = P.geo.dia_w(i)/2;
        r_b = P.geo.dia_w(i)/2 + ( P.geo.dia_e(i)-P.geo.dia_w(i) )/2 *...
            ( (P.current.tool_depth - P.geo.position_w(i))/ P.geo.dx(i) );
        P.current.contact_metal_area(i) = pi*(r_b^2 - r_a^2)*sqrt(4*m_ab^2
+1); %calculate by interpolation
    elseif P.current.tool_depth > P.geo.position_e(i)
        %if the tool-metal interface is past the current node (ie fully in
the metal)
        %then do the entire portion of the cone's area
        r_a = P.geo.dia_w(i)/2;
        r_b = P.geo.dia_e(i)/2;
        P.current.contact_metal_area(i) = pi*(r_b^2 - r_a^2)*sqrt(4*m_ab^2
+1);
    else
        %if the tool-metal interface is not to the node (ie 0% in the metal)
        %do nothing as it is already 0;
    end
```

```
end

%if it has contacted the metal
if P.current.tool_depth >= 0
    P.current.contact_metal_area(1) = P.current.contact_metal_area(1) +
pi*P.geo.d1^2/4;%this is the flat tip of the pin
end

P.current.contact_metal_fraction =
P.current.contact_metal_area./sum(P.current.contact_metal_area);
```