The Pennsylvania State University

The Graduate School

A SPLINE-BASED METHOD FOR SOLVING THE INVERSE HEAT CONDUCTION PROBLEM IN SLABS AND THICK-WALLED CYLINDERS UNDER COMPLEX THERMAL LOADING

A Thesis in

Engineering Science and Mechanics

by

Grant D. Klinger

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2023

The thesis of Grant D. Klinger was reviewed and approved by the following:

Albert E. Segall
Professor of Engineering Science and Mechanics
Thesis Co-Advisor

Corina S. Drapaca
Associate Professor
Thesis Co-Adviser

Matthew H. Lear
Assistant Research Professor
Thesis Co-Adviser

ABSTRACT

A spline solution to the inverse heat conduction problem using a thermal spline is explored for a finite plate/slab and a thick-walled cylinder with axial symmetry undergoing thermal loading at a surface. The surface thermal loading is assumed to take the form of a cubic polynomial with four unknown coefficients, with convection allowed to occur on the opposite surface where no thermal loading is applied. A direct heat conduction solution is defined as a function of time in terms of the cubic polynomial surface excitation and unit response solution using Duhamel's integral. Response temperatures were then theoretically constructed at a point on the convective surface of the slab and cylinder at discrete times. The direct thermal solution was substituted into the cubic spline process to fit the data by finding the four coefficients of the inverse solution between each set of data points. An inverse solution was then constructed piecewise for each interval with these four coefficients, allowing for much higher accuracy than that of the least squares method. Furthermore, response data that oscillates can easily be modeled by the thermal spline and produces a dependable inverse solution, whereas the least squares method is not capable of modeling such complex data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to sincerely thank Drs. A. E. Segall, C. S. Drapaca, and M. H. Lear for their advice, support, and patience throughout the entirety of this project. I truly appreciate the chance to work with such a wonderful faculty and I am grateful for all of the engineering and mathematical knowledge they have passed onto me.

I would like to specifically thank Dr. A. E. Segall for generating the response temperature data used in the test cases in this paper.

Research funding contributed by the Penn State ARL Walker Assistantship was immensely helpful and very much appreciated. Results and conclusions of this paper do not reflect the views of PSU/ARL.

Finally, I would like to thank my mother and father, Kristie and Neil, for supporting my decision to continue my education. Their love and support over the years has helped me overcome all obstacles.

Chapter 1

Introduction to the Direct and Inverse Heat Conduction Problems

1.1 Background

Understanding heat transfer through a solid body is of great importance in engineering. Thermal energy changes the way a material behaves, induces deformation, and can affect longevity. When a surface temperature history or a heat flux is known at one surface of a material, engineers are able to predict how temperatures will change at any point inside the material or on its opposite surface over time due to conduction. This is called the direct heat conduction problem and can be seen in Figure 1 for two very common geometries in engineering scenarios: the plate/slab and the thick-walled cylinder with axial symmetry. In many situations, the thermal loading is unknown, and cannot easily be measured due to restricted or no accessibility to the surface where the loading takes place. This is known as the inverse heat conduction problem and can also be seen in Figure 1.

A solution to the inverse problem is applicable in many manufacturing processes and beyond. Machining techniques can be improved, as well as cutting tool lifespan and product quality. The thermal loading for milling and hot forming metals can be optimized along with improvements in quenching, spray, and jet impingement processes that are needed to cool materials. In the aerospace industry, the surface temperature history can be determined on the outer surface of a shuttle or missile due to air resistance as the object returns to Earth. In biomedicine, excess internal heat distributed throughout the human body caused by diseases can be measured, and the heating process in hyperthermia can be improved upon to effectively destroy cancer cells without causing thermal damage elsewhere. Electronics can be cooled more efficiently by knowing the intensity of the heat generated by their parts, and non-destructive testing of materials under intense thermal loads, such as the electronic components, can be performed. These are just a few examples from the extensive list provided by Woodbury et al. [1].

1.2 Direct Heat Conduction Problem

The direct problem has been explored by many researchers over the decades with ample solutions discussed. The solutions generated are predominately for slab and cylinder geometries, with a few pertaining to other geometries such as the sphere. While all can be useful in their own rights depending on the problem being explored, adiabatic boundary conditions, assumed heat flux/surface temperature history functions, and time constraints can limit their applications.

In terms of direct solutions, Chen [2] proposed a model for a vehicle re-entering the Earth's atmosphere in terms of an arbitrary, time-dependent heat flux $q(t)$ acting upon the outer surface denoted as $x = a$. The problem's geometry was simplified to that of a slab, and it was assumed that the initial temperature was constant and the inner surface $x = 0$ was considered to be insulated. After applying Laplace transforms to the heat conduction equation and its initial and boundary conditions, a direct solution was obtained. However, the solution is for large time only due to series convergence issues and a different solution was proposed for small time values.

Nied [3][4] considers thermal shock on both an axisymmetric, hollow cylinder with a circumferential crack on its inner surface and an elastic strip (slab) with an edge-crack. The primary goal of these analyses was to determine the thermal stress distribution at the cracks, which relied heavily on the solution to the transient temperature distribution in the uncracked geometries. For the cylinder, the outer surface at $r = b$ was assumed to be insulated. The boundary condition on the inner surface at $r = a$ was modelled using a Heavyside function. For the slab, the side opposite that of the heat source, denoted $y = 0$, is again assumed to be insulated. On the surface $y = L$ where the heat flux is applied, a convective boundary condition was used to describe the heat loss to the surrounding environment. In both geometries, by solving the heat conduction equation with their respective prescribed boundary conditions, the direct solution was determined.

Austin [5] derived a solution for solid bodies undergoing heating whose surface temperature can be expressed as $\theta = \theta'\left(1 - e^{-\beta t}\right)$, where θ is the temperature at the surface at time t, θ' is the equilibrium temperature of the surface, and β is a constant that must be determined experimentally. This approach was designed to tackle problems with heat sources that cannot be expressed linearly or instantaneously. The surface boundary condition was defined as $\theta = \theta'\left(1 - e^{-\beta t}\right)$, and an initial condition of $\theta(0) = 0$ specified temperature at any point in the solid body at $t = 0$. A modified version of Duhamel's integral, dependent on a function $F(x, y, z, t)$ that represents temperature at a point (x, y, z) at time t and complies with the boundary conditions already mentioned, allows the generalized form of the direct solution to be determined. The solution was applied to many different, simple, solid-body geometries including a slab, rectangular parallelopiped, rod, cylinder, and sphere in multiple dimensions with known $F(x, y, z, t)$ found in previous research.

Finally, Pisarenko et al. [6] found a direct solution to the problem of a linear heating inside of refractory, nonmetallic hollow cylinders thermally loaded by an electric heater on the inner surfaces. Boundary conditions were imposed so that convection may occur on the outer radius $\rho = R$, and the thermal loading on the inner surface $\rho = r$ must change at a constant rate through time t. To justify the linear load, it was reasoned that if the thermal loading was described by the sum of a linear and nonlinear term; the nonlinear term would tend to zero as time increased under the prescribed formulation of the problem, thus leaving only the linear term.

1.3 Inverse Heat Conduction Problem

The inverse problem does not have as many applicable solutions as the direct problem due to it being ill-posed and very sensitive to fluctuations in data measurements. Regardless, there have been attempts made by researchers to establish some form of a solution, although they too suffer from limiting assumptions, impractical data collection techniques, time constraints, and lack of

versatility across geometries. Indeed, it is the lack of versatility for relevant problems that represents a major limitation to widespread applicability.

An early study by Imber [7] proposed a backward extrapolation technique for the inverse solution of a hollow cylinder with inner and outer radii r_i and r_0, respectively. Two thermocouples at different interior positions, r_1 and r_2, were used in this technique, with $r_2 >, r_1 > r_i$. A Laplace transform was used to solve the heat conduction equation with two interior conditions describing the temperature at each thermocouple as functions of time, $T_1(t)$ and $T_2(t)$, and a constant initial temperature. T_1, T_2, and the asymptotic series expansions of the modified Bessel functions made up the Laplace solution. A function in the real domain was obtained with the conversion of the Laplace solution, valid for any $r_i < r < r_1$ and $r_2 - r_1 \geq r_1 - r_i$. This solution is limited to short times due to the convergence issues with the asymptotic series expansions of the modified Bessel functions at later times. It was noted that a similar approach can be used for forward extrapolation for $r < r_2$.

Imber M. [8] also proposed an inverse solution for a solid cylinder, with outer radius r_0, using a single thermocouple located at r_1. Again, only one interior condition is prescribed defining the temperature at r_1 to be a function of time $T_1(t)$. The other boundary condition considers the thermal symmetry of the problem and allows no change in temperature to occur at the very center where $r = 0$. The Laplace transform of the problem was used to derive a s-domain solution in terms of T_1 and a ratio of modified Bessel functions. If the modified Bessel function ratio was replaced by the Bessel functions' asymptotic series expansions, a real-domain solution was obtained that was useful for small time under the condition that $\frac{r_1}{r_0} \leq \frac{r_1}{r} \leq 1$. However, for a solution for all time, a function in the s-domain, $F(s)$, is used to approximate the modified Bessel function ratio at small and large times. With this function, a solution is developed and transformed back to the real domain for $\frac{r_1}{r} \geq 0.5$ under the same stipulation that $\frac{r_1}{r_0} \leq \frac{r_1}{r} \leq 1$.

Blanc and Raynaud [9] took a unique approach in finding the inverse thermal solution for a hollow, axisymmetric cylinder using strain measurements induced by the heat flux $q(t)$ inside. A quasi-static state was assumed for the cylinder, with thermal and mechanical loading and a pressure differential between its inner and outer surfaces. By minimizing the sum of the squared differences between the measured strain X_{ll} and total calculated strain ε_{ll} in the l direction, the heat flux was determined in a recurrence relation provided heat flux, strains, and strain sensitivity coefficients S_ε are known at the previous time step, and heat flux remains constant over the number of future time strains. Initial temperature of the cylinder was set to zero, heat flux was applied on the inner surface, and temperature was measured on the outer surface, replacing the original insulated boundary condition. It was revealed that the sensitivity coefficients related to the heat flux through partial differentiation, that is $S_\varepsilon = \frac{\partial \varepsilon}{\partial q}$, and a system of equations provided a means for solving for the coefficients. It was also shown that by measuring the internal and external pressures, and considering that the circumferential displacement in the cylinder is zero, the total strain of the cylinder is only dependent on the temperature field caused from the heat flux inside as well as two simple parameters. This strain was called the inversion strain (ε_i), and through the use of equilibrium equations and stress-strain and strain-displacement relations, the temperature field was determined.

Yang et al. [10] used a similar approach to Blanc G. and Raynaud M. in finding the inverse solution. Heat flux was determined by minimizing the integral from time 0 to t_f of the squared differences

between the estimated displacement $u(r_2, t)$ and measured displacement $Y(r_2, t)$ at the strain gage location for the functional $J(q)$. This was done through an iterative process to find the heat flux function $q(t)$ to minimize $J(q)$.

Burggraf [11] used a series expansion composed of time derivatives of temperature distribution $T_0(t)$ and heat flux $q_0(t)$ with accompanying functions $f_n(r)$ and $g_n(r)$ to solve the inverse problem. This solution is applicable to adiabatic and isothermal situations for the slab as well as solid or hollow cylinders and spheres. However, due to the fact that higher order time derivatives in experimental data are not known accurately, the series solution for the slab must converge rapidly or else error from the higher order derivatives will cause problems in the solution. This means that for any practical application, the slab must be thin enough for convergence to occur. The solution for the hollow cylinder geometry is able to be applied to any wall thickness as long as slow changes occur in $T_0(t)$ and $q_0(t)$ due to abbreviating the series of time derivatives to only the first two terms. Thus, the solution is better suited for hollow cylinders with thin walls.

A least squares method was used by Taler [12] to solve the overdetermined inverse problem when multiple temperature sensors are used at different thicknesses to capture temperature data of a slab and hollow cylinder and sphere. It was assumed that perfect insulation was achieved on the outer surface of the material. A polynomial of pth degree with unknown coefficients a_i was assumed to describe the temperature change over time on the outer surface. It was then concluded that the temperature distribution $T(R, Fo)$ was the sum of the same unknown coefficients a_i multiplied by their known respective functions ϕ_i. By fitting the sensor data in the least squares sense, the coefficients a_i could be calculated and the inverse solution could be found.

Segall et al. [13] provided an elegant inverse solution that can be generalized for both a slab and an axisymmetric, thick-walled cylinder for any type of thermal loading that can be modeled using a polynomial. This is similar to the way Irving [14] defined the heat flux $q(t)$ in his article, with two key differences: surface temperature history $T_s(\omega_s, t)$ is used instead of heat flux, and both integer- and half-order powers of time are considered. Convection is assumed on the opposite surface of the thermal loading, so the solution is not limited to adiabatic boundaries. A direct thermal response solution is first formed using Duhamel's integral on the derivative of the polynomial describing the surface temperature excitation multiplied with the unit response. From there, the direct thermal response is fit to the temperature data gathered on the opposite surface of the excitation measured with only one sensing device. The fit is done using the least squares method, and thus the coefficients a_i are solved for and substituted back into the Nth order thermal loading polynomial for the inverse solution.

The issue with the least squares approach is the underlying assumption that the excitation can be described using a polynomial. When it can be, a fit to the temperature data results in a very reasonable inverse solution. However, the least squares method cannot approximate excitations of more complex forms. Therefore, the work by Segall et al. is enhanced by modifying the inverse solution approach to use a spline method, called the thermal spline, rather than a least squares fit to the temperature data collected on the back or outer surface of the slab or cylinder. This way, the excitation can be approximated piecewise and can handle more complex shapes rather than being limited to just a polynomial.

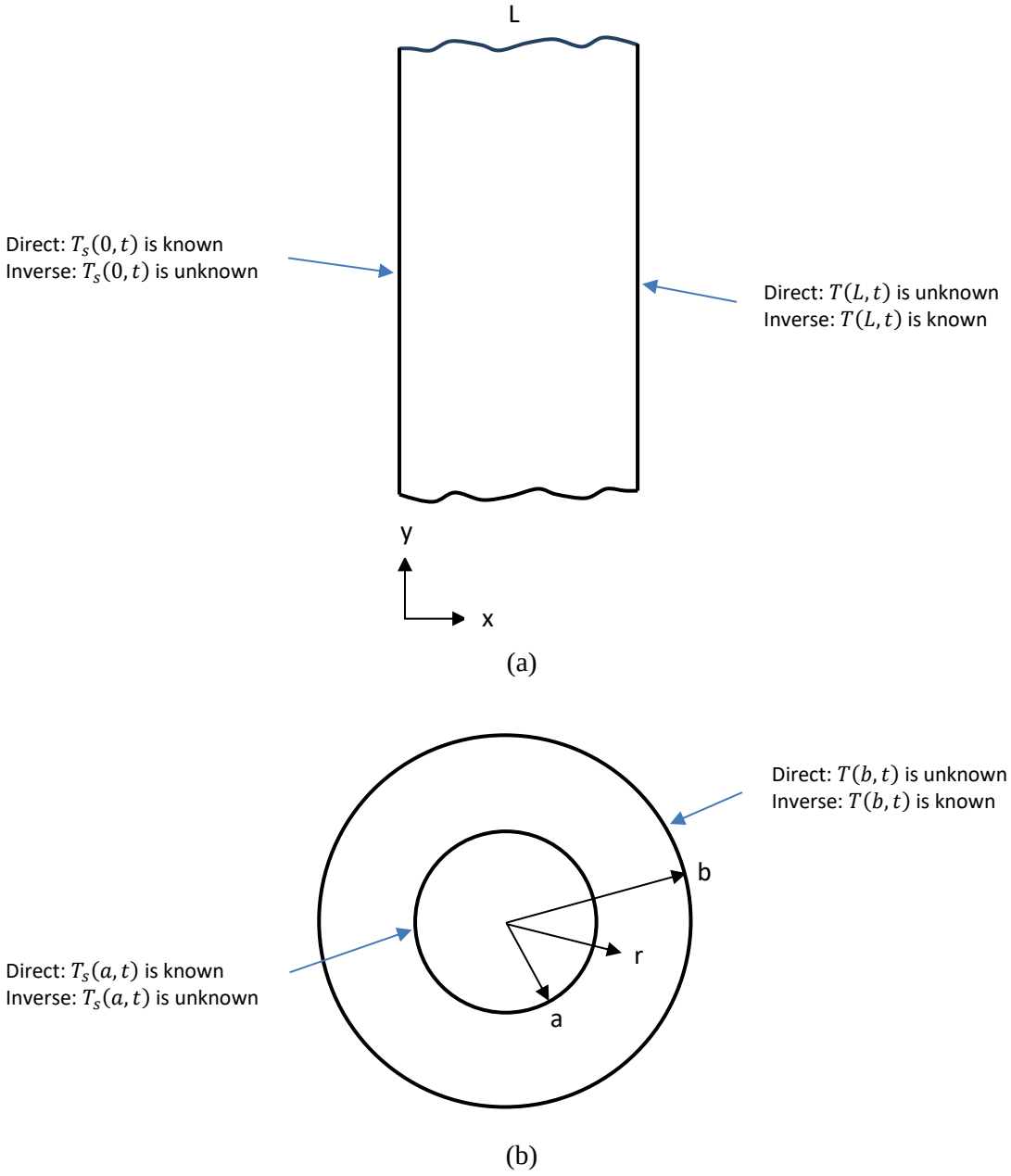

Figure 1 Direct and inverse problems in a (a) plate/slab and an (b) axisymmetric, thick-walled cylinder.

Chapter 2

Defining the Direct and Inverse Heat Conduction Solutions for a Plate/Slab and a Thick-Walled Cylinder

2.1 Heat Conduction Equation and Surface Temperature Excitation

All theoretical and practical description in Chapter 2 is mostly based on the work in [12] unless otherwise noted. To begin, the generalized partial differential equation describing one-dimensional heat conduction is defined as

$$\frac{1}{\omega^q}\frac{\partial}{\partial\omega}\left[k\omega^q\frac{\partial T}{\partial\omega}\right]=\rho C_p\frac{\partial T}{\partial t}, \tag{1}$$

where k is the thermal conductivity, ρ the density, C_p the specific heat of the material, t the time, and T the temperature. In Eq. (1), ω denotes the spatial coordinate that varies with the integer exponent q. When $q=0$, the coordinate system is rectangular ($\omega \equiv x$) and is used for the planar geometry found in a slab. When $q=1$, the coordinate system is polar ($\omega \equiv r$) and is used for the axisymmetric geometry of a thick-walled cylinder.

$T_s(\omega_s, t)$ represents the thermal loading at the surface of the slab or the cylinder, also known as the surface temperature history or surface temperature excitation, with the value ω_s being the coordinate of the surface where the thermal loading occurs. $T_s(\omega_s, t)$ can be represented as the following polynomial:

$$T_s\left(\omega_s,t\right)=a_0+a_1t^{\frac{1}{2}}+a_2t+a_3t^{\frac{3}{2}}+\cdots=\sum_{n=0}^{N}a_nt^{\frac{n}{2}}, \tag{2}$$

where t is time and the a_n terms are coefficients. For reasons that will be revealed later, the half-order terms are ignored and Eq. (2) is truncated at $N=6$ to obtain a standard cubic polynomial to describe the surface temperature excitation as

$$T_s(\omega_s,t)=a_0+a_2t+a_4t^2+a_6t^3. \tag{2a}$$

The coefficients a_n are reindexed so that Eq. (2a) becomes

$$T_s(\omega_s,t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3. \tag{2b}$$

Herein, $T(\omega,t)$ will be used to denote the direct thermal response to the surface temperature excitation at any given point in the geometry of the slab or the cylinder; this will be considered the direct solution in this paper and will be defined later in the chapter.

2.2 Thermal Conduction in a Plate/Slab

Substituting $q = 0$ into Eq. (1) results in the following heat conduction differential equation for a slab:

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{D}\frac{\partial T}{\partial t}, \tag{3}$$

where D is the thermal diffusivity of the material defined as $D = \frac{k}{\rho C_p}$, and x is the spatial coordinate within the thickness of the slab measured from the surface where the temperature excitation is applied. The initial and boundary conditions imposed include

$$T(x,0) = 0, \tag{3a}$$

$$-k\frac{\partial T}{\partial x}(L,t) = hT, \tag{3b}$$

and

$$T(0,t) = T_s(0,t), \tag{3c}$$

where L is the total thickness of the slab, h is the convective coefficient, and $T_s(0,t)$ is the thermal loading at the surface $x = 0$ in Eq (2b). When $T_s(0,t) = 1$, it has been shown that the following unit response is the solution to Eq. (3):

$$\phi(x,t) = \frac{1 + Bi\left(1 - \frac{x}{L}\right)}{1 + Bi} - \sum_{m=1}^{\infty} \frac{2\left(\beta_m^2 + Bi^2\right)\sin\left(\beta_m^2 \frac{x}{L}\right)}{\beta_m\left[Bi + Bi^2 + \beta_m^2\right]} e^{-\gamma_m^2 t}. \tag{4}$$

For brevity, the steady-state term that appears first in Eq. (4) is defined as

$$F\left(x\right)=\frac{1+Bi\left(1-\frac{x}{L}\right)}{1+Bi}, \tag{5}$$

and the quotient term inside of the summation that appears second in Eq. (4) is defined as

$$G_m\left(x\right)=-\frac{2\left(\beta_m^2+Bi^2\right)\sin\left(\beta_m^2\frac{x}{L}\right)}{\beta_m\left[Bi+Bi^2+\beta_m^2\right]}, \tag{6}$$

so that Eq. (4) can be compactly written as

$$\phi(x,t)=F(x)+\sum_{m=1}^{\infty}G_m(x)e^{-\gamma_m^2 t}. \tag{7}$$

Bi is the Biot number defined by

$$Bi=\frac{hL}{k}, \tag{8}$$

γ_m is defined by

$$\gamma_m=\frac{\beta_m\sqrt{D}}{L}, \tag{9}$$

and β_m can be found by solving for the positive real roots of the equation

$$\beta\cot\left(\beta\right)+Bi=0. \tag{10}$$

However, the roots of Eq. (10) can be accurately approximated for each β_m using the following equation used in [13]:

$$\beta_m\approx d_m+\frac{\pi}{2}\psi_m\Theta_m, \tag{11}$$

where

$$d_m = \pi\left(k - \frac{1}{2}\right), \tag{11a}$$

$$\psi_m = \frac{Bi}{Bi + \frac{\pi}{2} d_m}, \tag{11b}$$

and

$$\Theta_m \approx 1 + \psi_m\left(1 - \psi_m\right)\left[1 - \frac{0.85}{m} + \left(0.6 - \frac{0.71}{m}\right)\left(1 + \psi_m\right)\left(0.6 - \frac{0.245}{m} - \psi_m\right)\right]. \tag{11c}$$

All calculations in this paper use the aforementioned approximation to calculate the necessary β_m terms used in the transient term $G_m(x)$ of the unit response.

2.3 Thermal Conduction in a Thick-Walled Cylinder

Using $q = 1$ for the thick-walled cylinder, Eq. (1) reduces to the following heat conduction differential equation:

$$\frac{1}{D}\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r}, \tag{12}$$

with the thermally loaded surface at $r = a$ and the convection occurring at $r = b$. Boundary conditions as outlined in Eq. (3a) – (3c) are imposed for the cylinder, with r and b replacing x and L respectively in Eq. (3a) and (3b), and a replacing 0 in Eq. (3c). The unit response solution to Eq. (12) found in [15] is defined as

$$\phi\left(r,t\right) = \frac{1 - Bi\ln\left(\frac{r}{b}\right)}{1 - Bi\ln\left(\frac{a}{b}\right)} + \pi\sum_{m=1}^{\infty}\frac{C_0\left(r,\beta_m\right)C_1^2\left(b,\beta_m\right)}{C_2\left(a,b,\beta_m\right)}e^{-\gamma_m^2 t}. \tag{13}$$

Again, for brevity, the steady-state term that appears first in Eq. (13) is defined as

$$F\left(r\right) = \frac{1 - Bi\ln\left(\frac{r}{b}\right)}{1 - Bi\ln\left(\frac{a}{b}\right)}, \tag{14}$$

with the summation term redefined as

$$G_m(r) = \pi \frac{C_0(r,\beta_m) C_1^2(b,\beta_m)}{C_2(a,b,\beta_m)}, \tag{15}$$

thus, reducing Eq. (13) to

$$\phi(r,t) = F(r) + \sum_{m=1}^{\infty} G_m(r) e^{-\gamma_m^2 t}. \tag{16}$$

The Biot number Bi is modified from Eq. (8) so that b replaces L as follows:

$$Bi = \frac{hb}{k}, \tag{17}$$

and γ_m now excludes L from Eq. (9) to become

$$\gamma_m = \beta_m \sqrt{D}. \tag{18}$$

$C_0(r,\beta_m)$, $C_1(b,\beta_m)$, and $C_2(a,b,\beta_m)$ are functions expressed in terms of Bessel functions of the first kind, J_0 and J_1, and of the second kind, Y_0:

$$C_0(r,\beta_m) = J_0(r\beta_m) Y_0(a\beta_m) - Y_0(r\beta_m) J_0(a\beta_m), \tag{19}$$

$$C_1(b,\beta_m) = k\beta_m J_1(b\beta_m) - hJ_0(b\beta_m), \tag{20}$$

and

$$C_2(a,b,\beta_m) = \left(k^2\beta_m^2 + h^2\right)\left[J_0(a\beta_m)\right]^2 - \left(C_1(b,\beta_m)\right)^2. \tag{21}$$

Lastly, each β_m is a real root of the characteristic equation

$$J_0(a\beta)\left[\beta Y_1(b\beta) - \frac{h}{k} Y_0(b\beta)\right] - Y_0(a\beta)\left[\beta J_1(b\beta) - \frac{h}{k} J_0(b\beta)\right] = 0. \tag{22}$$

2.4 A Generalized Solution for the Direct Thermal Response for a Plate/Slab and Thick-Walled Cylinder using Duhamel's Integral

It is useful to generalize the unit response for a slab and cylinder in Eq. (4) and (13) respectively as

$$\phi(\omega,t) = F(\omega) + \sum_{m=1}^{\infty} G_m(\omega) e^{-\gamma_m^2 t}. \tag{23}$$

Duhamel's integral can be used to obtain the following relationship between the direct thermal response $T(\omega,t)$ and the surface temperature excitation $T_s(\omega_s,t)$ using the unit response $\phi(\omega,t)$ as a kernel as follows:

$$T(\omega,t) = T_s(\omega_s,0)\phi(\omega,t) + \int_0^t \frac{\partial T_s(0,\tau)}{\partial \tau}\phi(\omega,t-\tau)\,d\tau, \tag{24}$$

where the first term represents the contribution of a unit-type loading at the start of the transient heating phase. In addition, τ is the dummy integration variable. For a slab, $\omega_s = 0$, and for a cylinder, $\omega_s = a$. Evaluating $T_s(\omega_s,0)$ and $T_s(0,\tau)$ using Eq. (2b) produces

$$T_s(\omega_s,0) = a_0 + a_1(0) + a_2(0)^2 + a_3(0)^3 = a_0 \tag{25}$$

and

$$T_s(0,\tau) = a_0 + a_1\tau + a_2\tau^2 + a_3\tau^3. \tag{26}$$

Eq. (24) can be rewritten using Eq. (25) and (26) as follows:

$$T(\omega,t) = a_0\phi(\omega,t) + \int_0^t \frac{\partial\left(a_0 + a_1\tau + a_2\tau^2 + a_3\tau^3\right)}{\partial \tau}\phi(\omega,t-\tau)\,d\tau. \tag{27}$$

Taking the partial derivative with respect to τ in the integral of Eq. (27) provides

$$T(\omega,t) = a_0\phi(\omega,t) + \int_0^t \left(a_1 + 2a_2\tau + 3a_3\tau^2\right)\phi(\omega,t-\tau)\,d\tau. \tag{28}$$

Lastly, by substituting in Eq. (23) for $\phi(\omega,t)$ and $\phi(\omega,t-\tau)$ into Eq. (28), the following formulation is the result:

$$\begin{aligned} T(\omega,t) &= a_0\left[F(\omega) + \sum_{m=1}^{\infty} G_m(\omega) e^{-\gamma_m^2 t}\right] \\ &+ \int_0^t \left[\left(a_1 + 2a_2\tau + 3a_3\tau^2\right)\left(F(\omega) + \sum_{m=1}^{\infty} G_m(\omega) e^{-\gamma_m^2 (t-\tau)}\right)\right] d\tau. \end{aligned} \tag{29}$$

Through integration, Eq. (29) can be evaluated for the generalized direct thermal response as

$$T(\omega,t)=a_0\left[F(\omega)+\sum_{m=1}^{\infty}G_m(\omega)e^{-\gamma_m^2 t}\right]+a_1\left[F(\omega)t+\sum_{m=1}^{\infty}\left(G_m(\omega)\left(\frac{1}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t}\right)\right)\right)\right]$$
$$+a_2\left[F(\omega)t^2+\sum_{m=1}^{\infty}\left(2G_m(\omega)\left(\frac{t}{\gamma_m^2}-\frac{1}{\gamma_m^4}\left(1-e^{-\gamma_m^2 t}\right)\right)\right)\right] \quad (30)$$
$$+a_3\left[F(\omega)t^3+\sum_{m=1}^{\infty}\left(3G_m(\omega)\left(\frac{t^2}{\gamma_m^2}-\frac{2t}{\gamma_m^4}+\frac{2}{\gamma_m^6}\left(1-e^{-\gamma_m^2 t}\right)\right)\right)\right].$$

As long as one knows some basic material properties of the slab or cylinder, the only unknown values in Eq. (30) are the direct thermal response $T(\omega,t)$ and the surface temperature loading coefficients $a_0 - a_3$. Fortunately, $T(\omega,t)$ can be easily measured on the back surface $x = L$ of a slab, or the outer radial surface $r = b$ of a cylinder at discrete points in time. This allows for the cubic spline process to be adopted to solve for the coefficients $a_0 - a_3$.

Chapter 3

Mimicking the Cubic Spline to Solve the Inverse Heat Conduction Problem

3.1 Cubic Spline

The cubic spline works on a set of data points $(t_0, y_0), (t_1, y_1), \dots, (t_n, y_n)$, where n is the total number of data points, also known as knots. These knots are discrete points that lie on the curve $y = f(t)$, where $f(t)$ is unknown and does not follow the shape of any traditional function in mathematics. Any function such as $f(t)$ can be approximated by using a cubic spline consisting of third-degree piecewise polynomials defined as

$$S_i(t) = a_i + b_i t + c_i t^2 + d_i t^3, \tag{31}$$

where $a_i, b_i, c_i,$ and d_i are the coefficients for the i^{th} polynomial on the interval $[t_i, t_{i+1}]$, $i = 0,1, \dots n-1$. Each piecewise polynomial interpolates the data, meaning they must pass through each of the knots. Furthermore, each polynomial is considered to be a C^2 function, that is, they have continuous zeroth, first, and second derivatives. Thus, the entire cubic spline is said to be C^2 smooth.

All of the coefficients $a_i, b_i, c_i,$ and d_i must be found for each of the $n-1$ polynomials, so $4(n-1)$ equations are needed. The equations can be found by taking advantage of the interpolating nature of the spline and its C^2 smoothness. To begin, each polynomial must pass through the knots (t_i, y_i) and (t_{i+1}, y_{i+1}) such that:

$$S_i(t_i) = y_i$$

$$a_i + b_i t_i + c_i t_i^2 + d_i t_i^3 = y_i, \tag{32}$$

and

$$S_i(t_{i+1}) = y_{i+1}$$

$$a_i + b_i t_{i+1} + c_i t_{i+1}^2 + d_i t_{i+1}^3 = y_{i+1}. \tag{33}$$

Both Eq. (32) and (33) provide $n-1$ equations. Next, the first and second derivatives of the two polynomials joining at the knots (t_{i+1}, y_{i+1}) must be the same to guarantee C^2 smoothness. That is,

$$S_i'(t_{i+1}) = S_{i+1}'(t_{i+1})$$

$$b_i + 2c_i t_{i+1} + 3d_i t_{i+1}^2 = b_{i+1} + 2c_{i+1} t_{i+1} + 3d_{i+1} t_{i+1}^2, \tag{34}$$

and

$$S_i''(t_{i+1}) = S_{i+1}''(t_{i+1})$$

$$2c_i + 6d_i t_{i+1} = 2c_{i+1} + 6d_{i+1} t_{i+1}. \tag{35}$$

Eq. (34) and (35) each provide $n-2$ equations since the last node t_n cannot use these conditions. The number of equations now totals $4n-6$, with the final two equations usually generated from the boundary conditions of the spline; most popular choices are the natural, not-a-knot, and clamped boundaries. For the purposes of this paper, the natural boundary conditions are used for the final two equations. Natural boundary conditions state that the first and last polynomial's second derivative at the first and last point are zero respectively.

$$S_0''(t_0) = 0$$

$$2c_0 + 6d_0 t_0 = 0. \tag{36}$$

and

$$S_{n-1}''(t_n) = 0$$

$$2c_{n-1} + 6d_{n-1} t_n = 0. \tag{37}$$

Using this approach, the system now has the required $4(n-1)$ equations with the addition of the boundary conditions. Eq. (38) shows the resulting system of $4(n-1)$ equations in matrix form, where the last two rows are the natural boundary conditions.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & t_1 & t_1^2 & t_1^3 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 2t_1 & 3t_1^2 & 0 & -1 & -2t_1 & -3t_1^2 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 6t_1 & 0 & 0 & -2 & -6t_1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & t_{n-1} & t_{n-1}^2 & t_{n-1}^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & t_n & t_n^2 & t_n^3 \\ 0 & 0 & 2 & 6t_0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 2 & 6t_n \end{bmatrix} \begin{bmatrix} a_0 \\ b_0 \\ c_0 \\ d_0 \\ \vdots \\ a_{n-1} \\ b_{n-1} \\ c_{n-1} \\ d_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ 0 \\ 0 \\ \vdots \\ y_{n-1} \\ y_n \\ 0 \\ 0 \end{bmatrix}. \quad (38)$$

The coefficients of each piecewise polynomial can now be solved for and, thus, a polynomial $S_i(t)$ can be constructed between all of the knots.

3.2 Thermal Spline

By modifying $S_i(t)$ to become Eq. (30), the thermal spline is defined as

$$\begin{aligned} S_i(t) &= a_{0_i}\left[F(\omega)+\sum_{m=1}^{\infty}G_m(\omega)e^{-\gamma_m^2 t}\right]+a_{1_i}\left[F(\omega)t+\sum_{m=1}^{\infty}\left(G_m(\omega)\left(\frac{1}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t}\right)\right)\right)\right] \\ &+a_{2_i}\left[F(\omega)t^2+\sum_{m=1}^{\infty}\left(2G_m(\omega)\left(\frac{t}{\gamma_m^2}-\frac{1}{\gamma_m^4}\left(1-e^{-\gamma_m^2 t}\right)\right)\right)\right] \\ &+a_{3_i}\left[F(\omega)t^3+\sum_{m=1}^{\infty}\left(3G_m(\omega)\left(\frac{t^2}{\gamma_m^2}-\frac{2t}{\gamma_m^4}+\frac{2}{\gamma_m^6}\left(1-e^{-\gamma_m^2 t}\right)\right)\right)\right]. \end{aligned} \quad (39)$$

Substituting Eq. (39) into Eq. (32) – (35), and replacing y_i with $T_i(\omega, t_i)$, results in

$$a_{0_i}\left[F(\omega)+\sum_{m=1}^{\infty}G_m(\omega)e^{-\gamma_m^2 t_i}\right]+a_{1_i}\left[F(\omega)t_i+\sum_{m=1}^{\infty}\left(G_m(\omega)\left(\frac{1}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_i}\right)\right)\right)\right]$$
$$+a_{2_i}\left[F(\omega)t_i^2+\sum_{m=1}^{\infty}\left(2G_m(\omega)\left(\frac{t_i}{\gamma_m^2}-\frac{1}{\gamma_m^4}\left(1-e^{-\gamma_m^2 t_i}\right)\right)\right)\right] \tag{40}$$
$$+a_{3_i}\left[F(\omega)t_i^3+\sum_{m=1}^{\infty}\left(3G_m(\omega)\left(\frac{t_i^2}{\gamma_m^2}-\frac{2t_i}{\gamma_m^4}+\frac{2}{\gamma_m^6}\left(1-e^{-\gamma_m^2 t_i}\right)\right)\right)\right]=T_i(\omega,t_i),$$

$$a_{0_i}\left[F(\omega)+\sum_{m=1}^{\infty}G_m(\omega)e^{-\gamma_m^2 t_{i+1}}\right]+a_{1_i}\left[F(\omega)t_{i+1}+\sum_{m=1}^{\infty}\left(G_m(\omega)\left(\frac{1}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right]$$
$$+a_{2_i}\left[F(\omega)t_{i+1}^2+\sum_{m=1}^{\infty}\left(2G_m(\omega)\left(\frac{t_{i+1}}{\gamma_m^2}-\frac{1}{\gamma_m^4}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right] \tag{41}$$
$$+a_{3_i}\left[F(\omega)t_{i+1}^3+\sum_{m=1}^{\infty}\left(3G_m(\omega)\left(\frac{t_{i+1}^2}{\gamma_m^2}-\frac{2t_{i+1}}{\gamma_m^4}+\frac{2}{\gamma_m^6}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right]=T_{i+1}(\omega,t_{i+1}),$$

$$a_{0_i}\left[-\sum_{m=1}^{\infty}G_m(\omega)\gamma_m^2 e^{-\gamma_m^2 t_{i+1}}\right]+a_{1_i}\left[F(\omega)+\sum_{m=1}^{\infty}\left(G_m(\omega)e^{-\gamma_m^2 t_{i+1}}\right)\right]$$
$$+a_{2_i}\left[2\left(F(\omega)t_{i+1}+\sum_{m=1}^{\infty}\left(\frac{G_m(\omega)}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right]$$
$$+a_{3_i}\left[3\left(F(\omega)t_{i+1}^2+\sum_{m=1}^{\infty}\left(\frac{2G_m(\omega)}{\gamma_m^2}\left(t_{i+1}-\frac{1}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right)\right]$$
$$=a_{0_{i+1}}\left[-\sum_{m=1}^{\infty}G_m(\omega)\gamma_m^2 e^{-\gamma_m^2 t_{i+1}}\right]+a_{1_{i+1}}\left[F(\omega)+\sum_{m=1}^{\infty}\left(G_m(\omega)e^{-\gamma_m^2 t_{i+1}}\right)\right] \tag{42}$$
$$+a_{2_{i+1}}\left[2\left(F(\omega)t_{i+1}+\sum_{m=1}^{\infty}\left(\frac{G_m(\omega)}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right]$$
$$+a_{3_{i+1}}\left[3\left(F(\omega)t_{i+1}^2+\sum_{m=1}^{\infty}\left(\frac{2G_m(\omega)}{\gamma_m^2}\left(t_{i+1}-\frac{1}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right)\right],$$

and

$$
\begin{aligned}
&a_{0_i}\left[\sum_{m=1}^{\infty} G_m(\omega)\gamma_m^4 e^{-\gamma_m^2 t_{i+1}}\right] + a_{1_i}\left[-\sum_{m=1}^{\infty}\left(G_m(\omega)\gamma_m^2 e^{-\gamma_m^2 t_{i+1}}\right)\right] \\
&+a_{2_i}\left[2\left(F(\omega)+\sum_{m=1}^{\infty}\left(G_m(\omega)e^{-\gamma_m^2 t_{i+1}}\right)\right)\right] \\
&+a_{3_i}\left[6\left(F(\omega)t_{i+1}+\sum_{m=1}^{\infty}\left(\frac{G_m(\omega)}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right] \\
&=a_{0_{i+1}}\left[\sum_{m=1}^{\infty} G_m(\omega)\gamma_m^4 e^{-\gamma_m^2 t_{i+1}}\right] + a_{1_{i+1}}\left[-\sum_{m=1}^{\infty}\left(G_m(\omega)\gamma_m^2 e^{-\gamma_m^2 t_{i+1}}\right)\right] \\
&+a_{2_{i+1}}\left[2\left(F(\omega)+\sum_{m=1}^{\infty}\left(G_m(\omega)e^{-\gamma_m^2 t_{i+1}}\right)\right)\right] \\
&+a_{3_{i+1}}\left[6\left(F(\omega)t_{i+1}+\sum_{m=1}^{\infty}\left(\frac{G_m(\omega)}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_{i+1}}\right)\right)\right)\right].
\end{aligned}
\tag{43}
$$

Using the natural boundary conditions defined in Eq. (36) and Eq. (37), the final two equations become

$$
\begin{aligned}
&a_{0_0}\left[\sum_{m=1}^{\infty} G_m(\omega)\gamma_m^4 e^{-\gamma_m^2 t_0}\right] + a_{1_0}\left[-\sum_{m=1}^{\infty}\left(G_m(\omega)\gamma_m^2 e^{-\gamma_m^2 t_0}\right)\right] \\
&+a_{2_0}\left[2\left(F(\omega)+\sum_{m=1}^{\infty}\left(G_m(\omega)e^{-\gamma_m^2 t_0}\right)\right)\right] \\
&+a_{3_0}\left[6\left(F(\omega)t_0+\sum_{m=1}^{\infty}\left(\frac{G_m(\omega)}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_0}\right)\right)\right)\right]=0
\end{aligned}
\tag{44}
$$

and

$$
\begin{aligned}
&a_{0_{n-1}}\left[\sum_{m=1}^{\infty} G_m(\omega)\gamma_m^4 e^{-\gamma_m^2 t_n}\right] + a_{1_{n-1}}\left[-\sum_{m=1}^{\infty}\left(G_m(\omega)\gamma_m^2 e^{-\gamma_m^2 t_n}\right)\right] \\
&+a_{2_{n-1}}\left[2\left(F(\omega)+\sum_{m=1}^{\infty}\left(G_m(\omega)e^{-\gamma_m^2 t_n}\right)\right)\right] \\
&+a_{3_{n-1}}\left[6\left(F(\omega)t_n+\sum_{m=1}^{\infty}\left(\frac{G_m(\omega)}{\gamma_m^2}\left(1-e^{-\gamma_m^2 t_n}\right)\right)\right)\right]=0.
\end{aligned}
\tag{45}
$$

Using Eq. (40) – (43) for each piecewise function and the two natural boundary conditions in Eq. (44) and (45), $4(n-1)$ equations can be solved using the following system:

$$\begin{bmatrix} A_{0_0} & A_{1_0} & A_{2_0} & A_{3_0} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ A_{0_1} & A_{1_1} & A_{2_1} & A_{3_1} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ A_{0_1}^{'} & A_{1_1}^{'} & A_{2_1}^{'} & A_{3_1}^{'} & -A_{0_1}^{'} & -A_{1_1}^{'} & -A_{2_1}^{'} & -A_{3_1}^{'} & \cdots & 0 & 0 & 0 & 0 \\ A_{0_1}^{''} & A_{1_1}^{''} & A_{2_1}^{''} & A_{3_1}^{''} & -A_{0_1}^{''} & -A_{1_1}^{''} & -A_{2_1}^{''} & -A_{3_1}^{''} & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A_{0_{n-1}} & A_{1_{n-1}} & A_{2_{n-1}} & A_{3_{n-1}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A_{0_n} & A_{1n} & A_{2_n} & A_{3_n} \\ A_{0_0}^{''} & A_{1_0}^{''} & A_{2_0}^{''} & A_{3_0}^{''} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A_{0_n}^{''} & A_{1_n}^{''} & A_{2_n}^{''} & A_{3_n}^{''} \end{bmatrix} \begin{bmatrix} a_{0_0} \\ a_{1_0} \\ a_{2_0} \\ a_{3_0} \\ \vdots \\ a_{0_{n-1}} \\ a_{1_{n-1}} \\ a_{2_{n-1}} \\ a_{3_{n-1}} \end{bmatrix}$$

$$= \begin{bmatrix} T_0(\omega, t_0) \\ T_1(\omega, t_1) \\ 0 \\ 0 \\ \vdots \\ T_{n-1}(\omega, t_{n-1}) \\ T_n(\omega, t_n) \\ 0 \\ 0 \end{bmatrix}. \tag{46}$$

where the leftmost matrix contains the expressions that multiply coefficients $a_{0_i} - a_{3_i}$ defined as follows:

$$A_{0_j} = \left[F(\omega) + \sum_{m=1}^{\infty} G_m(\omega) e^{-\gamma_m^2 t_j} \right] \tag{46a}$$

$$A_{1_j} = \left[F(\omega) t_j + \sum_{m=1}^{\infty} \left(G_m(\omega) \left(\frac{1}{\gamma_m^2} \left(1 - e^{-\gamma_m^2 t_j} \right) \right) \right) \right] \tag{46b}$$

$$A_{2_j} = \left[F(\omega) t_j^2 + \sum_{m=1}^{\infty} \left(2G_m(\omega) \left(\frac{t_j}{\gamma_m^2} - \frac{1}{\gamma_m^4} \left(1 - e^{-\gamma_m^2 t_j} \right) \right) \right) \right] \tag{46c}$$

$$A_{3_j} = \left[F(\omega) t_j^3 + \sum_{m=1}^{\infty} \left(3G_m(\omega) \left(\frac{t_j^2}{\gamma_m^2} - \frac{2t_j}{\gamma_m^4} + \frac{2}{\gamma_m^6} \left(1 - e^{-\gamma_m^2 t_j} \right) \right) \right) \right] \tag{46d}$$

$$A'_{0_j} = \left[-\sum_{m=1}^{\infty} G_m(\omega)\gamma_m^2 e^{-\gamma_m^2 t_j} \right] \tag{46e}$$

$$A'_{1_j} = \left[F(\omega) + \sum_{m=1}^{\infty} \left(G_m(\omega) e^{-\gamma_m^2 t_j} \right) \right] \tag{46f}$$

$$A'_{2_j} = \left[2\left(F(\omega) t_j + \sum_{m=1}^{\infty} \left(\frac{G_m(\omega)}{\gamma_m^2} \left(1 - e^{-\gamma_m^2 t_j} \right) \right) \right) \right] \tag{46g}$$

$$A'_{3_j} = \left[3\left(F(\omega) t_j^2 + \sum_{m=1}^{\infty} \left(\frac{2G_m(\omega)}{\gamma_m^2} \left(t_j - \frac{1}{\gamma_m^2} \left(1 - e^{-\gamma_m^2 t_j} \right) \right) \right) \right) \right] \tag{46h}$$

$$A''_{0_j} = \left[\sum_{m=1}^{\infty} G_m(\omega)\gamma_m^4 e^{-\gamma_m^2 t_j} \right] \tag{46i}$$

$$A''_{1_j} = \left[-\sum_{m=1}^{\infty} G_m(\omega)\gamma_m^2 e^{-\gamma_m^2 t_j} \right] \tag{46j}$$

$$A''_{2_j} = \left[2\left(F(\omega) + \sum_{m=1}^{\infty} G_m(\omega) e^{-\gamma_m^2 t_j} \right) \right] \tag{46k}$$

$$A''_{3_j} = \left[6\left(F(\omega) t_j + \sum_{m=1}^{\infty} \left(\frac{G_m(\omega)}{\gamma_m^2} \left(1 - e^{-\gamma_m^2 t_j} \right) \right) \right) \right] \tag{46l}$$

where $j = i$ or $i + 1$ depending on the condition. At this point, the inverse solution can be found by constructing the surface temperature excitation in a piecewise fashion by substituting each set of coefficients back into the following equation on each interval $[t_i, t_{i+1}]$, $i = 0,1,\ldots n-1$:

$$T_{s_i}(\omega_s, t) = a_{0_i} + a_{1_i} t + a_{2_i} t^2 + a_{3_i} t^3 \tag{47}$$

Eq. (47) is the piecewise representation of Eq. (2b), where all of the data points at which the cubic piecewise polynomials join will still be referred to as the knots.

3.3 Smoothing Noisy Data

As mentioned earlier, a spline is designed to interpolate data and, if the data measurements contain errors, the resulting spline will go through the erroneous data points by adjusting slope and concavity. This creates a slight wiggle in the spline through the response temperature data and a very drastic wiggle in the inverse solution. To account for the random error that occurs within the measured temperature data on the convective surface of the cylinder, a combination of both pre- and post-processing can be used.

To pre-process the response temperature data, a moving Least Squares (LS) regression approach over one or more specified window sizes on different sections of the data can be used. By implementing the LS fit at each point, the data can be adjusted to a more accurate value based on the curve fit to the points within the window. The function used in the LS process is that of the direct thermal response defined in Eq. (30), and can be done in MATLAB using the *lsqcurvefit()* function.

Since the inverse solution is just comprised of piecewise cubic polynomials, i.e., a cubic spline, standard smoothing algorithms can be used to post-process the inverse solution. This can be implemented in MATLAB using the *csaps()* function, where f is defined as the smoothing spline that minimizes the functional shown in Eq. (48).

$$p\sum_{k=1}^{n} w_k \left| y_k - f\left(x_k\right)\right|^2 + (1-p)\int \lambda(t)\left| f^{"}(t)\right|^2 dt \tag{48}$$

The first term consists of a LS fit for error measurement over n points, while the second is the roughness penalty for smoothness. A tuning parameter, p, with optimal range $0 \le p \le 1$, provides a balance between the two criteria. All error measure weights w_j and the piecewise constant weight function $\lambda(t)$ are kept at values of 1 for the purposes of this paper.

Chapter 4

Results and Discussion

4.1 Preliminary Information for the Plate/Slab

Following analyses use theoretical heating situations to display the effectiveness of the thermal spline method when calculating the inverse solution. The LS method is used for comparison with the thermal spline and was implemented using *lsqcurvefit()* in MATLAB with Eq. (30) as the function for the data to fit. Furthermore, to illustrate the benefit of the smoothing techniques previously mentioned for the thermal spline, a randomly generated error in the range of $\pm 2\%$ was applied on the response temperature data in each scenario to simulate realistic noise in the measurements. Pre-processing window size and post-processing smoothing parameters were manually selected for each case, as these will change based on the dataset and user preferences.

4.2 Inverse Solution for the Plate/Slab

For all heating scenarios for the slab, the following properties were used: $h = 6.75 \times 10^{-3}\ \frac{J}{s \cdot cm^2 \cdot °C}$, $k = 0.0417\ \frac{J}{s \cdot cm \cdot °C}$, $C_p = 0.169\ \frac{J}{g \cdot °C}$, $\rho = 0.066\ \frac{g}{cm^3}$, $L = 3\ cm$, $x = 3\ cm$. The summation upper bound was capped at a value of $m = 100$, and the response temperature data was generated at the back surface of the slab at $x = 3\ cm$; only the surface temperature excitation at $x = 0\ cm$ was changed in each situation.

4.2.1 Asymptotic Exponential Excitation

In this example, the excitation was modeled using an asymptotic exponential of the form: $T_s(x = 0, t) = 1 - e^{-\frac{1}{2}t}$ as shown in Figure 2 along with the response temperature data it generated listed in Table 1.

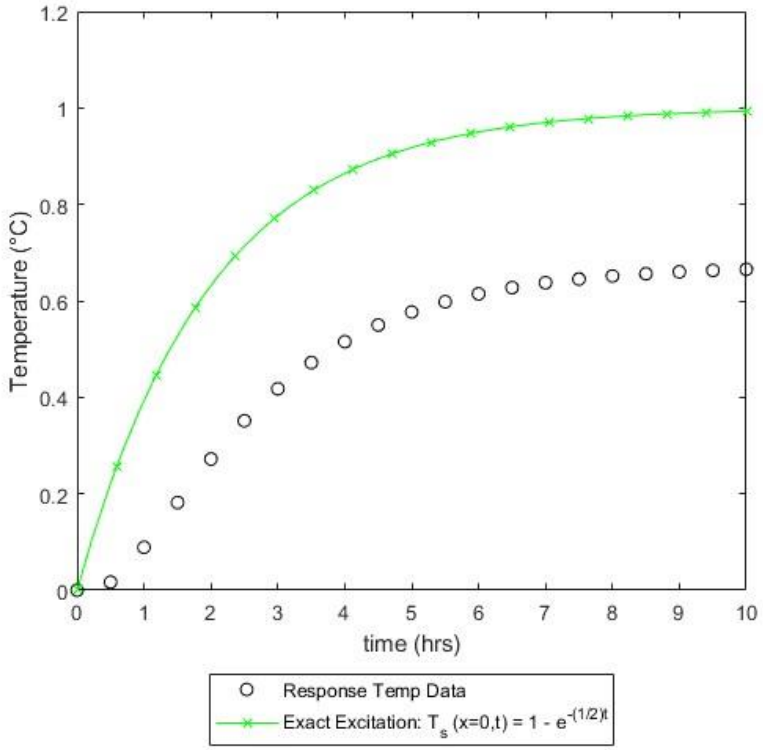

Figure 2 The exact surface temperature excitation defined by $T_s(x=0,t) = 1 - e^{-\frac{1}{2}t}$ along with the corresponding response for the slab.

Table 1 The response temperature data generated on the back surface of the slab at $x = 3$ as a result of the surface temperature excitation defined by $T_s(x = 0, t) = 1 - e^{-\frac{1}{2}t}$.

Time (hrs)	*Temperature (°C)*
0.001	0.0000
0.5	0.0167
1	0.0890
1.5	0.1817
2	0.2719
2.5	0.3512
3	0.4177
3.5	0.4720
4	0.5154
4.5	0.5499
5	0.5770
5.5	0.5981
6	0.6146
6.5	0.6274
7	0.6374
7.5	0.6452
8	0.6514
8.5	0.6563
9	0.6601
9.5	0.6632
10	0.6654

Figure 3 presents the inverse solution obtained by fitting a thermal spline to the response temperature data in Table 1 along with the exact surface temperature excitation for comparison. Figure 4 presents the same image using the LS approach, which does an adequate job fitting the response temperature data and, thus, the polynomial approximating the surface temperature loading is rather close. However, the thermal spline method creates a more accurate fit of the data, leading to an even better inverse solution.

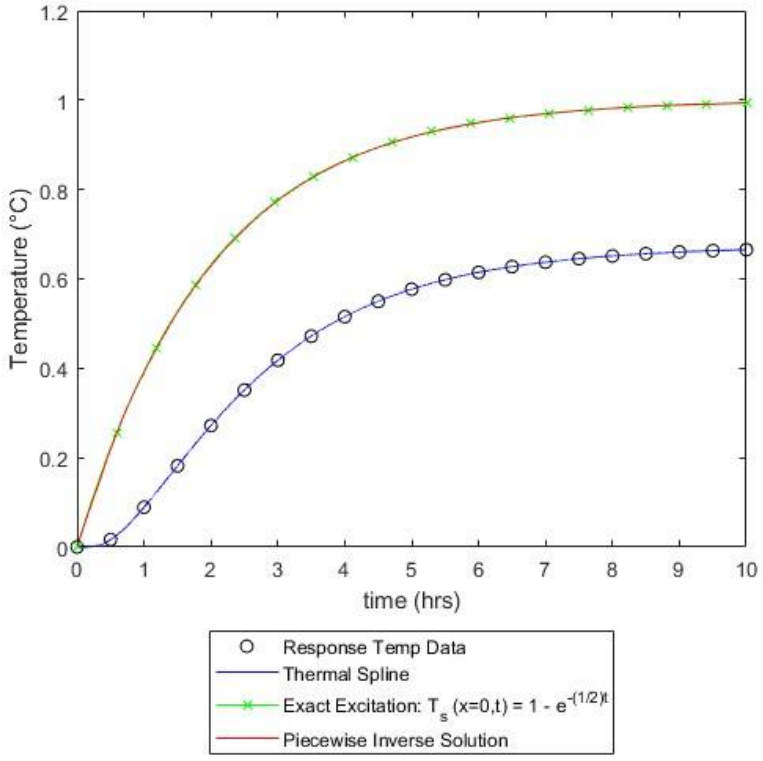

Figure 3 Response temperature data fitted with a thermal spline, the resulting piecewise inverse solution, and the exact surface temperature excitation $T_s(x=0,t) = 1 - e^{-\frac{1}{2}t}$ for the slab.

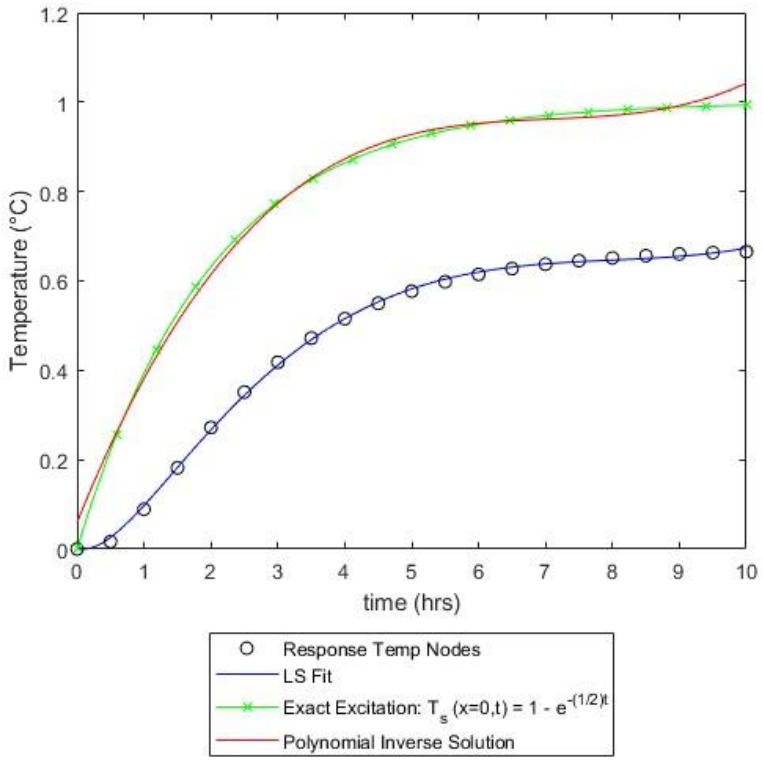

Figure 4 Response temperature data fitted with LS, the resulting polynomial inverse solution, and the exact surface temperature excitation $T_s(x = 0, t) = 1 - e^{-\frac{1}{2}t}$ for the slab.

Table 2 displays sampled points at and midway between the knots of the LS polynomial and thermal spline piecewise inverse solutions and compares them to the exact inverse solution. The percent error is calculated to show the increased accuracy of the thermal spline over the LS method in more detail. It must be noted, however, that due to numerical roundoff error during calculations of the coefficients for the piecewise functions used in the thermal spline, the coefficients themselves inherit error. Therefore, the temperature values of the piecewise inverse solution at interior knots are slightly different when calculated using the cubic polynomial on the left side versus the cubic polynomial on the right side of the knot. This difference creates a minor discontinuity as can be seen in Figure 5. Thus, Table 2 uses the mean temperature value at the interior knot locations.

Table 2 The percent error of the thermal spline and LS method inverse solutions as compared with the exact surface temperature excitation $T_s(x=0,t)=1-e^{-\frac{1}{2}t}$ for the slab.

Time (hrs)	*Exact Excitation* $T_s(x=0,t)=1-e^{-\frac{1}{2}t}$. *(°C)*	*Piecewise Inverse Solution (°C)*	*Percent Error (%)*	*LS Method Inverse Solution (°C)*	*Percent Error (%)*
0.001	0.0005	0.0000	103.3747	0.0599	11886.8370
0.2505	0.1177	0.1118	5.0260	0.1493	26.8073
0.5	0.2212	0.2190	1.0050	0.2324	5.0823
0.75	0.3127	0.3130	0.0866	0.3098	0.9398
1	0.3935	0.3899	0.9048	0.3813	3.0850
1.25	0.4647	0.4610	0.7985	0.4473	3.7424
1.5	0.5276	0.5253	0.4492	0.5080	3.7122
1.75	0.5831	0.5804	0.4690	0.5637	3.3399
2	0.6321	0.6291	0.4761	0.6144	2.8003
2.25	0.6753	0.6725	0.4188	0.6605	2.1914
2.5	0.7135	0.7112	0.3189	0.7023	1.5722
2.75	0.7472	0.7451	0.2693	0.7398	0.9804
3	0.7769	0.7753	0.2069	0.7735	0.4400
3.25	0.8031	0.8018	0.1630	0.8034	0.0330
3.5	0.8262	0.8253	0.1117	0.8298	0.4291
3.75	0.8466	0.8459	0.0846	0.8529	0.7426
4	0.8647	0.8642	0.0548	0.8731	0.9716
4.25	0.8806	0.8802	0.0434	0.8904	1.1168
4.5	0.8946	0.8943	0.0298	0.9052	1.1810
4.75	0.9070	0.9067	0.0311	0.9176	1.1692
5	0.9179	0.9176	0.0292	0.9279	1.0879
5.25	0.9276	0.9272	0.0378	0.9363	0.9453
5.5	0.9361	0.9357	0.0416	0.9431	0.7508
5.75	0.9436	0.9431	0.0519	0.9484	0.5153
6	0.9502	0.9497	0.0559	0.9526	0.2506
6.25	0.9561	0.9555	0.0631	0.9558	0.0303
6.5	0.9612	0.9606	0.0627	0.9582	0.3135
6.75	0.9658	0.9652	0.0633	0.9601	0.5842
7	0.9698	0.9693	0.0561	0.9618	0.8268
7.25	0.9734	0.9729	0.0489	0.9634	1.0249
7.5	0.9765	0.9761	0.0348	0.9651	1.1615
7.75	0.9792	0.9790	0.0207	0.9673	1.2190
8	0.9817	0.9817	0.0014	0.9701	1.1791
8.25	0.9838	0.9840	0.0136	0.9738	1.0231
8.5	0.9857	0.9860	0.0292	0.9785	0.7318
8.75	0.9874	0.9879	0.0451	0.9846	0.2857
9	0.9889	0.9895	0.0596	0.9922	0.3353
9.25	0.9902	0.9906	0.0384	1.0016	1.1516
9.5	0.9913	0.9913	0.0031	1.0130	2.1836
9.75	0.9924	0.9926	0.0262	1.0266	3.4524
10	0.9933	0.9943	0.1095	1.0427	4.9790

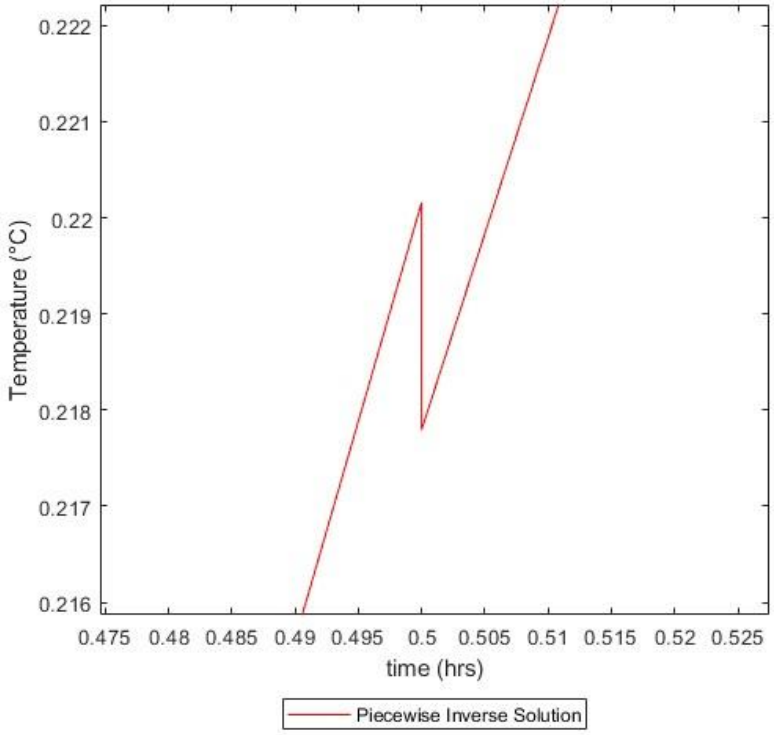

Figure 5 A magnified example of a minor discontinuity in the piecewise inverse solution at $t = 0.5\ hrs$ for the slab.

When random noise was applied to the response data, no smoothing was used for discrete time points $0.001 \leq t < 3$, while a smoothing window of 6 was used for $t \geq 3$. A post-processing smoothing parameter of $p = 0.008$ was used in combination with the pre-processing to create the image in Figure 6. It can be seen that the thermal spline produces a reasonable fit through the data and the inverse solution stays close to the exact excitation.

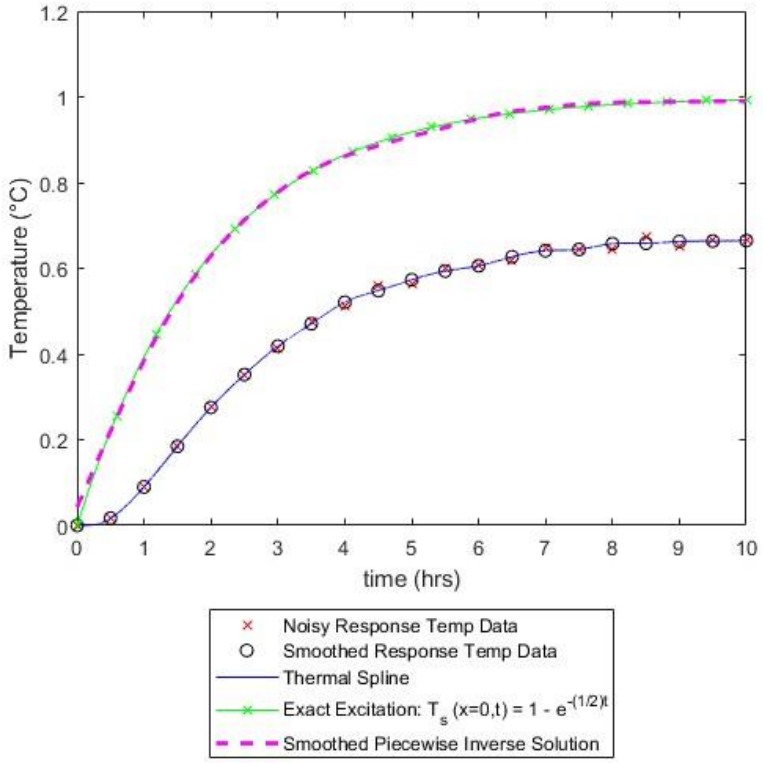

Figure 6 Thermal spline fit to pre-processed response temperature data with a window size of 6 for discrete time points $t \geq 3$, the resulting post-processed inverse solution using $p = 0.008$, and the exact surface temperature excitation: $T_s(x = 0, t) = 1 - e^{-\frac{1}{2}t}$ for the slab.

4.2.2 Sinusoidal Excitation

The LS method is comparable to the thermal spline method when calculating an inverse solution modeled after a decaying, asymptotic exponential. However, the LS method breaks down when the data takes on an oscillating shape. For this case, the excitation was modeled as $T_s(x = 0, t) = \sin(t)$. Figure 7 shows this excitation and the response temperature data listed in Table 3.

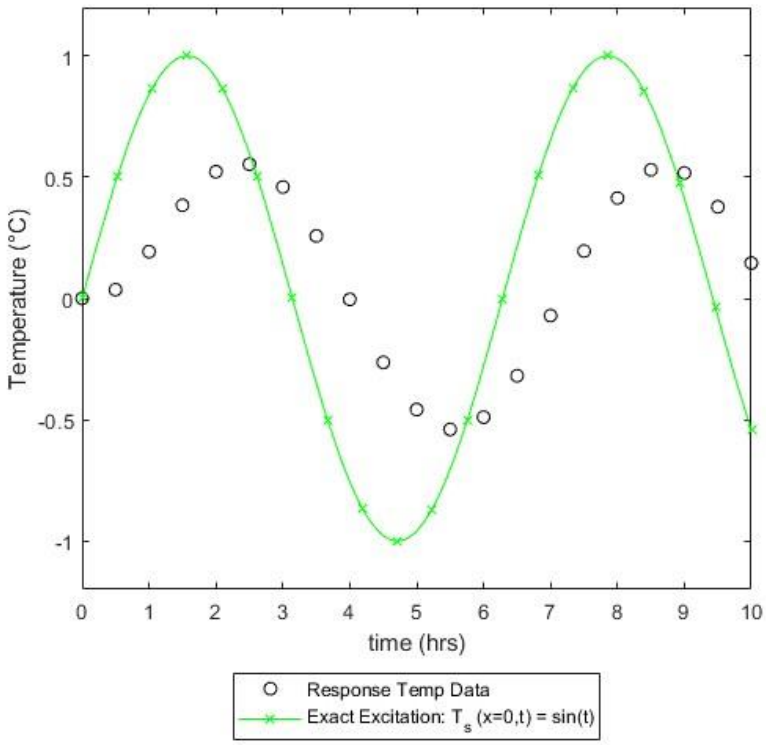

Figure 7 Exact surface temperature excitation defined by $T_s(x = 0, t) = \sin(t)$ for the slab.

Table 3 Response temperature data generated on the back surface of the slab at $x = 3$ as a result of the excitation defined by $T_s(x = 0, t) = \sin(t)$.

Time (hrs)	*Temperature (°C)*
0.001	0.0000
0.5	0.0360
1	0.1928
1.5	0.3841
2	0.5225
2.5	0.5535
3	0.4594
3.5	0.2580
4	-0.0041
4.5	-0.2638
5	-0.4582
5.5	-0.5402
6	-0.4897
6.5	-0.3193
7	-0.0706
7.5	0.1954
8	0.4135
8.5	0.5304
9	0.5175
9.5	0.3778
10	0.1457

Figure 8 displays the inverse solution obtained by fitting a thermal spline to the response temperature data in Table 3 and is again overlaid with the exact inverse solution for comparison. A very accurate inverse prediction results from the capability of the thermal spline to capture the true shape of the data. Slight deviation from the shape of the exact excitation does occur in the last cubic polynomial segment in the piecewise inverse solution. This can be attributed to the boundary condition choice at the end point of the spline. If a more suitable boundary condition is known at the last point of the response data, the final cubic polynomial in the inverse solution will fit more appropriately. Data in Table 3 was fitted with the LS method as well. This time, the LS fit to the data is very poor, and results in inaccurate coefficients that produce a polynomial inverse solution that is just as poor seen in Figure 9.

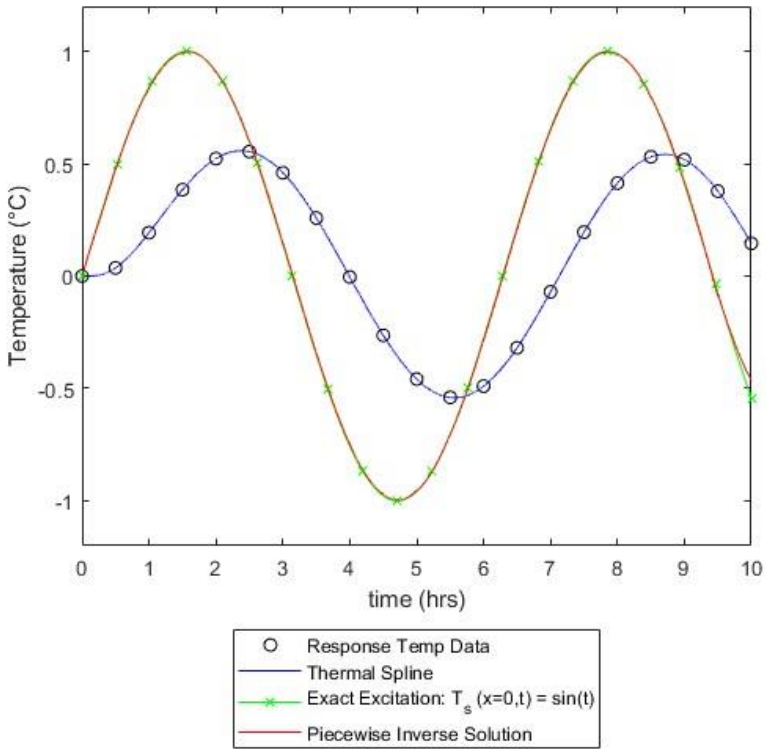

Figure 8 Response temperature data fitted with a thermal spline, the resulting piecewise inverse solution, and the exact surface temperature excitation $T_s(x = 0, t) = \sin(t)$ for the slab.

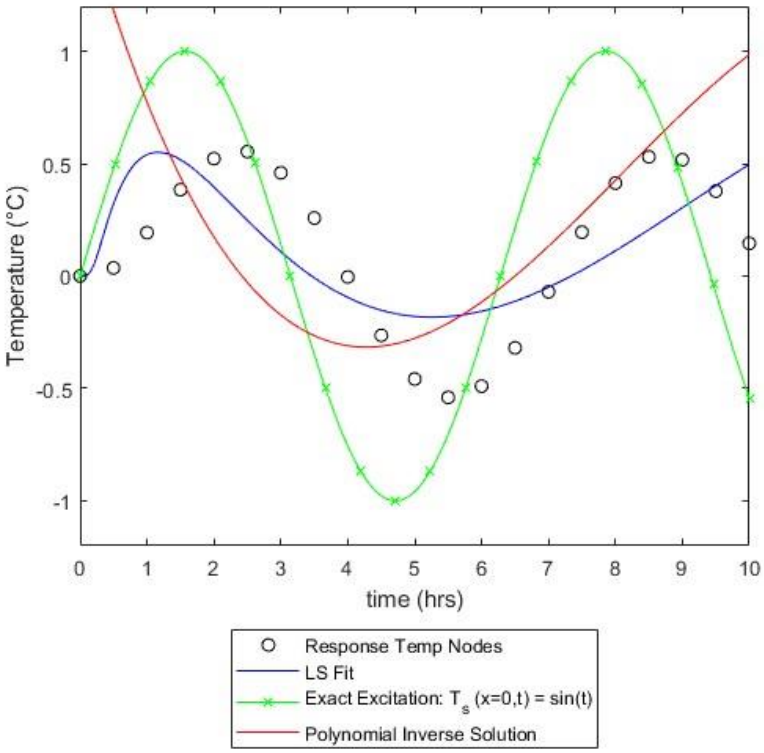

Figure 9 Response temperature data fitted with LS, the resulting polynomial inverse solution, and the exact surface temperature excitation $T_s(x = 0, t) = \sin(t)$ for the slab.

After adding noise to the response data, only a post-processing smoothing parameter of $p = 0.1$ was required to smooth the inverse solution. Figure 10 shows how well the thermal spline can fit the noisy data and how the post-processed inverse solution still remains very accurate in approximating the exact excitation.

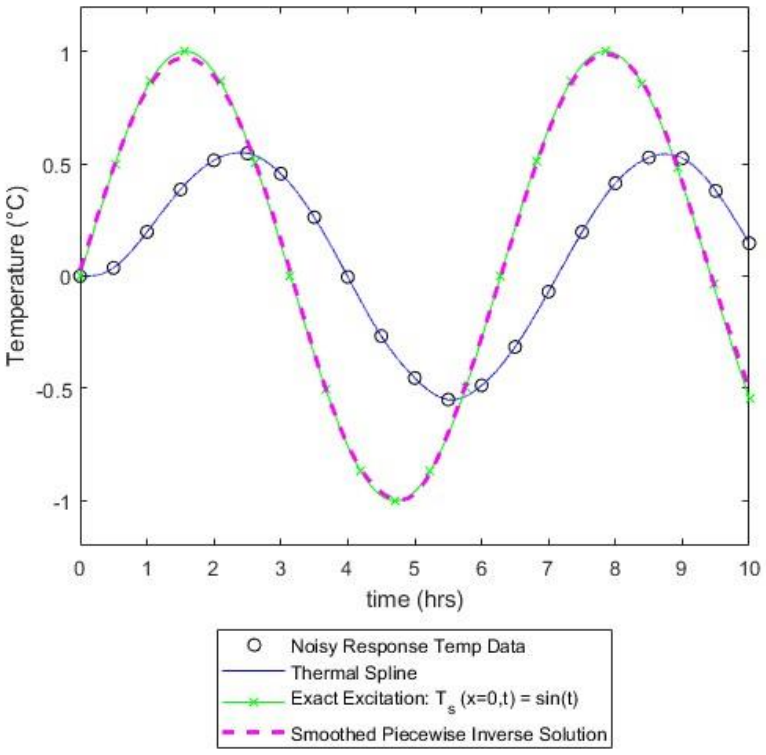

Figure 10 Thermal spline fit to response temperature data with error, the resulting post-processed inverse solution using $p = 0.1$, and the exact surface temperature excitation $T_s(x = 0, t) = \sin(t)$ for the slab.

4.3 Inverse Solution for the Thick-Walled Cylinder

The thermal spline method was expected to work exactly the same for the cylinder geometry. However, mild to severe oscillations occur during the transient phase of heating in the spline and the inverse solution it produces. To showcase this, two heating scenarios were examined with the following properties: $h = 0.121\ \frac{J}{s \cdot cm^2 \cdot °C}$, $k = 0.1\ \frac{J}{s \cdot cm \cdot °C}$, $C_p = 0.5\ \frac{J}{g \cdot °C}$, $\rho = 0.241\ \frac{g}{cm^3}$, $a = 4\ cm,\ b = 6\ cm,\ r = 6\ cm$, with the summation upper bound set at $m = 100$. Response temperature data was generated at the outer surface of the cylinder at $r = 6\ cm$, while only the thermal excitation at $r = 4\ cm$ was changed in each situation.

The first example is shown in Figure 11 and illustrates the excitation modeled using $T_s(r = 4, t) = 1 - e^{-\frac{1}{2}t}$. Response temperature data generated at the outer radius by this excitation is also shown in the figure as well as listed in Table 4. As with other test cases, the thermal spline method was used to fit the data and then generate the inverse solution as shown in Figure 11. While the piecewise inverse solution is still a relatively good fit, small oscillations can be seen in the very early time segments during the transient heating of the cylinder. Such results are most likely the

cause of an incorrect fit of the spline to the beginning data points that is not visible to the eye in the plot.

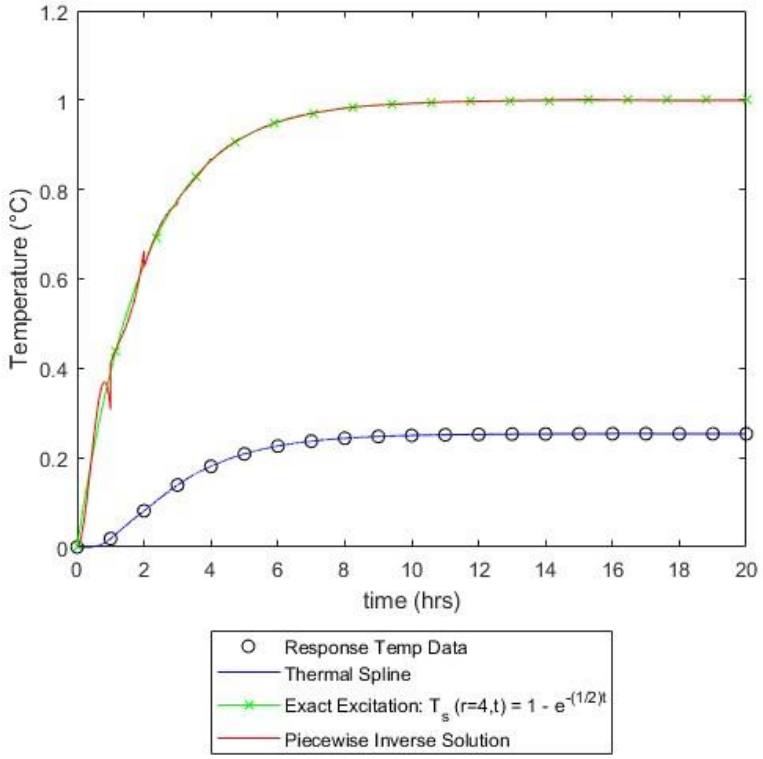

Figure 11 Response temperature data fitted with the thermal spline, the resulting piecewise inverse solution, and the exact surface temperature excitation of $T_s(r = 4, t) = 1 - e^{-\frac{1}{2}t}$ for the cylinder.

Table 4 The response temperature data generated on the outer surface of the cylinder at $r = 6$ as a result of the surface temperature excitation defined by $T_s(r = 4, t) = 1 - e^{-\frac{1}{2}t}$.

Time (hrs)	*Temperature (°C)*
0.001	0.0000
1	0.0189
2	0.0810
3	0.1388
4	0.1806
5	0.2084
6	0.2260
7	0.2369
8	0.2434
9	0.2474
10	0.2497
11	0.2511
12	0.2521
13	0.2527
14	0.2531
15	0.2534
16	0.2536
17	0.2536
18	0.2535
19	0.2534
20	0.2534

To support this hypothesis, a thermal spline was fitted to the response data in Table 5 produced by an excitation modeled using $T_s(r = 4, t) = \sin(t)$. Figure 12 shows much more extreme oscillations that are visible in the spline during the transient phase. These oscillations become much more amplified in the beginning of the piecewise inverse solution and render it useless. Figure 12 cannot show the inverse solution due to the extremely large scale of the plot to capture its wide range of oscillations, so only the spline fit to the response temperature data is displayed. Further research is needed to isolate the root cause of the issue in the cylindrical geometry.

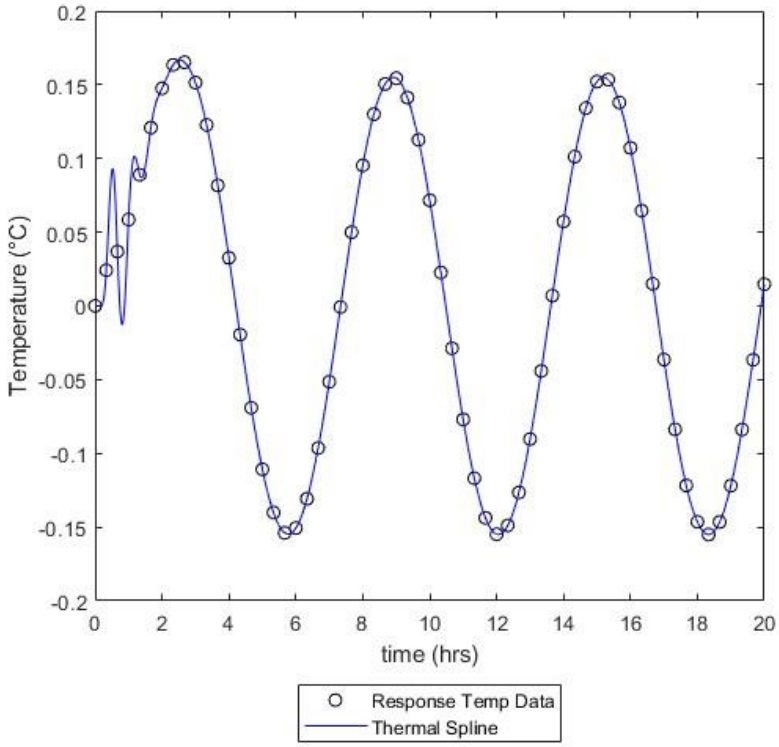

Figure 12 Response temperature data fitted with the thermal spline for the cylinder.

Table 5 Response temperature data generated on the outer surface of the cylinder at $r = 6$ as a result of the exact surface temperature excitation defined by $T_s(r = 4, t) = \sin(t)$.

Time (hrs)	*Temperature (°C)*
0	0.0000
0.3333	0.0242
0.6667	0.0369
1	0.0586
1.3333	0.0889
1.6667	0.1209
2	0.1476
2.3333	0.1635
2.6667	0.1653
3	0.1514
3.3333	0.1227
3.6667	0.0818
4	0.0326
4.3333	-0.0195
4.6667	-0.0691
5	-0.1108
5.3333	-0.1401
5.6667	-0.1539
6	-0.1506
6.3333	-0.1307
6.6667	-0.0964
7	-0.0514
7.3333	-0.0007
7.6667	0.0500
8	0.0953
8.3333	0.1301
8.6667	0.1505
9	0.1544
9.3333	0.1413
9.6667	0.1127
10	0.0716
10.3333	0.0227
10.6667	-0.0288
11	-0.0770
11.3333	-0.1168
11.6667	-0.1437
12	-0.1548
12.3333	-0.1489
12.6667	-0.1266
13	-0.0903
13.3333	-0.0441

Time (hrs)	*Temperature (°C)*
13.6667	0.0070
14	0.0573
14.3333	0.1013
14.6667	0.1341
15	0.1522
15.3333	0.1535
15.6667	0.1379
16	0.1072
16.3333	0.0646
16.6667	0.0150
17	-0.0364
17.3333	-0.0837
17.6667	-0.1218
18	-0.1465
18.3333	-0.1550
18.6667	-0.1466
19	-0.1219
19.3333	-0.0839
19.6667	-0.0366
20	0.0147

Chapter 5

Conclusion

5.1 Summary

The surface temperature excitation was assumed to be in the form of a cubic polynomial with four unknown coefficients. Duhamel's integral was used to convolve the surface temperature excitation and the unit response to obtain a function for the direct thermal response for a plate/slab or an axisymmetric, thick-walled cylinder. The cubic spline process was then adopted to construct the thermal spline, which allowed for the calculation of the four unknown coefficients in the direct thermal response by fitting to the response temperature data points gathered on the convective surface. Once the coefficients were known on each interval, the inverse solution was constructed piecewise.

The thermal spline method ultimately offers a new way to solve the inverse heat conduction problem without the need for limiting assumptions or boundary conditions that may not always be realistic in every type of scenario. Indeed, the thermal spline is a simple and elegant one-size-fits-all approach for thermal loads of varying complexity that only requires one measurement source, on the back side of the slab or outer surface of the cylinder, to accurately predict the behavior of the excitation. Though small discontinuities appear at the knots of the inverse solution, they are insignificant and are often undetectable on large time intervals like in the test cases shown. Due to the fact that the thermal spline interpolates data, a least squares smoothing technique over variable window sizes was developed to reduce any inherent noise and help the thermal spline achieve a better fit. The cubic smoothing spline also offered a way to post-process the inverse solution for further improvement. However, this does require manual intervention to set the smoothing criteria in each case and may involve trial and error. Nonetheless, once the appropriate criteria are found, the thermal spline creates an accurate fit to the response data, and a reliable inverse solution can still be built.

5.2 Future Work

Unfortunately, the thermal spline is currently only effective in predicting the surface temperature excitation in the slab geometry. In the beginning transient heating portion, the cylinder inverse solution faces problems in the form of oscillations at small time values, which are due to wiggle in the beginning piecewise functions of the spline. Recent research suggests that there were errors in the process that generated the theoretical response temperature data for both the asymptotic exponential and sinusoidal excitations used for the cylinder. New data was generated in a MATLAB process that allowed for the thermal spline to produce very accurate inverse solutions in both heating scenarios, with no oscillations present. Further investigation into the original data generation process is ongoing to find and verify the error(s).

Smaller time increments could potentially be a problem as more discontinuities in the inverse solution will most certainly be produced. Employment of post-processing may be useful to smooth

the inverse solution on small time steps. Thus, a definition for “fine data” should be determined, and the threshold be implemented in the code in order to prompt the user for a cubic smoothing spline parameter in these scenarios. Furthermore, if an optimal smoothing parameter value could be found, the post-processing for small time steps could be completely automated without any user input required. This technique could be used for large time steps as well when the user requests post-process smoothing.

Producing an inverse solution under complex loading was the primary objective of the thermal spline, so more exploration into these situations for the slab, and potentially the cylinder, should be investigated. Theoretical thermal loads with discontinuities, such as a triangular loading, and more oscillatory loadings, such as Bessel functions, are great candidates to analyze the effectiveness of the thermal spline’s ability to predict the inverse solution. Perhaps more importantly, experimental response temperature data needs to be used with the thermal spline in order to determine the applicability of this method in realistic scenarios. More test cases would help determine if any further development is needed for the thermal spline, and strengthen the argument for the spline’s use for the inverse problem.

Appendix

Thermal_Spline MATLAB Code

Please note, the Thermal_Spline MATLAB code reads the material properties, time, and temperature data from two sheets in an Excel workbook named “Data File.xlsx” located in the same folder as the Thermal_Spline.m file. The Excel workbook sheets are named “Slab” and “Cylinder,” and are respectively formatted as shown below:

	A	B	C	D	E	F	G	H	I	J
1	**Properties**				**Time**	**Temperature**	**Window Size (Pre-Processing)[1][2]**	**Cubic Smoothing Spline Parameter (Post-Processing)[1]**		[1]Leave blank if not needed
2	Convective Coefficient (h)	0.00675			0.001	0				
3	Thermal Conductivity (k)	0.0417			0.5	0.016685058				[2]Enter an integer greater than or equal to 5 in each desired row for smoothing, or enter 0 for no smoothing. The specified window size is used for the data pair it is placed next to and for all pairs below unless a new window size is specified. All data must have a window size specified if pre-processing is necessary.
4	Specific Heat (Cp)	0.169			1	0.089016619				
5	Density (rho)	0.066			1.5	0.181697073				
6	Thickness (x)	3			2	0.27188734				
7	Slab Thickness (L)	3			2.5	0.351240279				
8					3	0.417741116				
9	Summation Upper Bound	100			3.5	0.471969198				
10					4	0.515440637				
11					4.5	0.543887528				
12					5	0.576958631				
13					5.5	0.598110511				
14					6	0.614580396				
15					6.5	0.627393339				
16					7	0.637381378				
17					7.5	0.645205284				
18					8	0.6513751				
19					8.5	0.656268226				
20					9	0.660144864				
21					9.5	0.663161053				
22					10	0.665379657				
23										

	A	B	C	D	E	F	G	H	I	J
1	**Properties**				**Time**	**Temperature**	**Window Size (Pre-Processing)[1][2]**	**Cubic Smoothing Spline Parameter (Post-Processing)[1]**		[1]Leave blank if not needed
2	Convective Coefficient (h)	0.121			0.001	0				
3	Thermal Conductivity (k)	0.1			1	0.0189				[2]Enter an integer greater than or equal to 5 in each desired row for smoothing, or enter 0 for no smoothing. The specified window size is used for the data pair it is placed next to and for all pairs below unless a new window size is specified. All data must have a window size specified if pre-processing is necessary.
4	Specific Heat (Cp)	0.5			2	0.08098				
5	Density (rho)	0.241			3	0.1388				
6	Inner Radius (a)	4			4	0.1806				
7	Outer Radius (b)	6			5	0.2084				
8	Radius (r)	6			6	0.226				
9					7	0.2369				
10	Summation Upper Bound	100			8	0.2434				
11					9	0.2474				
12					10	0.2497				
13					11	0.2511				
14					12	0.2521				
15					13	0.2527				
16					14	0.2531				
17					15	0.2534				
18					16	0.2536				
19					17	0.2536				
20					18	0.2535				
21					19	0.2534				
22					20	0.2534				
23										

```
%% CLEARING EVERYTHING FROM THE PREVIOUS RUN

clear


%% GETTING THE GEOMETRY FROM THE USER

% Prompting the user to select slab or cylinder geometry
geoPrompt = 'Enter 1 for Slab or 2 for Cylinder:  ';
geo = input(geoPrompt);

% Keep prompting the user for geometry if it is not a 1 or 2
while geo ~= 1 && geo ~= 2
    disp('INVLAID RESPONSE');
    geo = input(geoPrompt);
```

```
end


%% GETTING AND CALCULATING PROPERTIES FOR THE SLAB OR CYLINDER
% Slab
if geo == 1
    %Reading the data file
    Data = readtable('Data File.xlsx', 'Sheet', 'Slab', 'HeaderLines', 1);

    % Initializing time and temperature arrays
    t = table2array(Data(1:height(Data), 5)); % Time
    t(isnan(t)) = []; % Cleaning NaN values from time

    TempData = table2array(Data(1:height(Data), 6)); % Temperature
    TempData(isnan(TempData)) = []; % Cleaning NaN values from temperature

    % Initializing/calculating properties independent of m
    h = table2array(Data(1, 2)); % Convective Coefficient
    k = table2array(Data(2, 2)); % Thermal Conductivity
    Cp = table2array(Data(3, 2)); % Specific Heat
    p = table2array(Data(4, 2)); % Density
    x = table2array(Data(5, 2)); % Variable Thickness
    L = table2array(Data(6, 2)); % Slab Thickness
    D = k / (Cp * p); % Materials Thermal Diffusivity
    Bi = (h * L) / k; % Biot number
    F = (1 + (Bi * (1 - (x / L)))) / (1 + Bi); % Steady State Term

    mUpper = table2array(Data(8, 2)); % Summation Upper Bound

    win = table2array(Data(:, 7)); % Window Size
    winSt = find(~isnan(win)); % Finding the associated row index where the
window size starts
    win(isnan(win)) = []; % Cleaning NaN values from window sizes

    % Checking to see if pre-processing is necessary based on user input
    if isempty(win)
        preProc = 'N';
    else
        preProc = 'Y';

        % Killing the program if the user forgot to define the window size for
the first section of data
        if winSt(1) ~= 1
            disp('ERROR: The beginning window size is not defined.');
            return;
        end
    end

    sp = table2array(Data(1, 8)); % Smoothing Parameter

    % Checking to see if post-processing is necessary based on user input
    if isnan(sp)
        postProc = 'N';
```

```
    else
        postProc = 'Y';
    end

    %Initializing the Beta (B), Gamma (Y), and Time-independent summation term
arrays
    B = zeros(mUpper, 1);
    Y = zeros(mUpper, 1);
    G = zeros(mUpper, 1);

    % Calculating properties dependent on m
    for m = 1:mUpper
        d = pi * (m - 0.5);
        psi = Bi / (Bi + ((pi / 2.0) * d));
        theta = 1.0 + (psi * (1.0 - psi)) * (1.0 - (0.85 / m) + ((0.6 - (0.71
/ m)) * (1.0 + psi) * (0.6 - (0.245 / m) - psi)));
        B(m) = d + ((pi / 2.0) * psi * theta); % Beta
        Y(m) = (B(m) * sqrt(D)) / L; % Gamma
        G(m) = -(((2.0 * ((B(m)^2) + Bi^2)) * sin(B(m) * (x / L))) / (B(m) *
(Bi + Bi^2 + B(m)^2))); % Time-independent summation term in the unit response
    end

%Cylinder
else
    %Reading the data file
    Data = readtable('Data File.xlsx', 'Sheet', 'Cylinder', 'HeaderLines', 1);

    % Initializing time and temperature arrays
    t = table2array(Data(1:height(Data), 5)); % Time
    t(isnan(t)) = []; % Cleaning NaN values from time

    TempData = table2array(Data(1:height(Data), 6)); % Temperature
    TempData(isnan(TempData)) = []; % Cleaning NaN values from temperature

    % Initializing/calculating properties independent of m
    h = table2array(Data(1, 2)); % Convective Coefficient
    k = table2array(Data(2, 2)); % Thermal Conductivity
    Cp = table2array(Data(3, 2)); % Specific Heat
    p = table2array(Data(4, 2)); % Density
    a = table2array(Data(5, 2)); % Inner Radius
    b = table2array(Data(6, 2)); % Outer Radius
    r = table2array(Data(7, 2)); % Radius
    D = k / (Cp * p); % Materials Thermal Diffusivity
    Bi = (h * b) / k; % Biot number
    F = (1 - (Bi * log(r / b))) / (1 - (Bi * log(a / b))); % Steady State Term

    mUpper = table2array(Data(9, 2)); % Summation Upper Bound

    win = table2array(Data(:, 7)); % Window Size
    winSt = find(~isnan(win)); % Finding the associated row of the window size
    win(isnan(win)) = []; % Cleaning NaN values from window sizes

    % Checking to see if pre-processing is necessary based on user input
```

```
    if isempty(win)
        preProc = 'N';
    else
        preProc = 'Y';

        % Killing the program if the user forgot to define the window size for
the first section of data
        if winSt(1) ~= 1
            disp('ERROR: The beginning window size is not defined.');
            return;
        end
    end

    sp = table2array(Data(1, 8)); % Smoothing Parameter

    % Checking to see if post-processing is necessary based on user input
    if isnan(sp)
        postProc = 'N';
    else
        postProc = 'Y';
    end

    % Calculating properties dependent on m
    inc = 0.01; % Increment value
    pt = 0.001 + inc; % Current point being evaluated
    prevPt = 0.001; % Previous point that was evaluated
    rtPos = 1; % Index to put the rtFunc value in the Beta array when a root
is found
    rtCt = 0; % Roots that were found
    rtFunc = @(B) (besselj(0, a*B) .* ((B.*bessely(1, b*B)) - ((h/k) .*
bessely(0, b*B)))) - (bessely(0, a*B) .* ((B*besselj(1, b*B)) - ((h/k) .*
besselj(0, b*B)))); % Root function

    %Initializing the Beta (B) array
    B = zeros(mUpper, 1);

    % Searching for real roots of rtFunc to populate the Beta array
    while rtCt < mUpper

        % Searching for a sign change between the current point and previous
point and then looking for the zero between those two points when a sign
change is found
        % The root is ignored if the current point equals zero and there is a
sign change between it and the previous point. The root will be captured the
next time around when the previous point equals zero. This is to make sure
there are no duplicate roots.
        if sign(rtFunc(prevPt)) ~= sign(rtFunc(pt)) && sign(rtFunc(pt)) ~= 0
            B(rtPos) = fzero(rtFunc, [pt prevPt]);
            rtPos = rtPos + 1;
            rtCt = rtCt + 1;

        % The root is captured at the previous point if both points equal
zero. This is to make sure there are no duplicate roots.
```

```
        elseif sign(rtFunc(prevPt)) == 0 && sign(rtFunc(pt)) == 0
            B(rtPos) = prevPt;
            rtPos = rtPos + 1;
            rtCt = rtCt + 1;
        end

        % Setting the previous point to equal the current point and
incrementing the current point
        prevPt = pt;
        pt = pt + inc;
    end

    % Gamma (Y) array
    Y = B * sqrt(D);

    % Recurring terms in the unit response summation
    C0 = (besselj(0, r*B) .* bessely(0, a*B)) - (bessely(0, r*B) .* besselj(0,
a*B));
    C1 = (k * B .* besselj(1, b*B)) - (h * besselj(0, b*B));
    C2 = (((k^2 * B.^2) + h^2) .* ((besselj(0, a*B)).^2)) - (((k * B .*
besselj(1, b*B)) - (h * besselj(0, b*B))).^2);

    % Time-independent summation term (G) array
    G = pi * ((C0 .* C1.^2) ./ C2);
end

% Defining the total number of knots
n = length(t);


%% PRE-PROCESSING THE TEMPERATURE DATA IF REQUIRED, ELSE JUST USING THE
ORIGINAL TEMPERATURE DATA FROM THE DATA FILE

% Pre-processing the data if required
if preProc == 'Y'

    T = zeros(n, 1); % Pre-processed temperature data array
    TIdx = 1; % Pre-processed temperature data index counter

    lastWin = length(win); % Last Window Index

    % Looping through all window sizes
    for w = 1:lastWin

        winSize = win(w); % Window Size

        %Finding the indices of the points selected for the window
        if w < lastWin
            ptIdx = winSt(w):(winSt(w) + (winSt(w + 1) - winSt(w)) - 1);
        else
            ptIdx = winSt(w):n;
        end
```

```
        % Pre-processing if the window size is not 0 (0 = No Smoothing)
        if winSize ~= 0
            tSub = zeros(1, winSize); % Time Subset Array
            TSub = zeros(1, winSize); % Temperature Subset Array

            % Finding half the window size and rounding down in case it is odd
            winHalf = floor(winSize / 2);

            %Finding upper and lower bounds to the window
            % Lower Bound subtracts half the window size from the entire
window size to find the first point it can center on
            % Upper Bound subtracts half the window size from the total points
to find the first point it can center on
            % Odd
            if mod(winSize, 2) == 1
                lb = winSize - winHalf; % Window Lower Bound
                ub = n - winHalf; % Window Upper Bound
            % Even
            else
                % Need -1 here since the "center" of an even interval is
defined at the first point to the right of the true center
                lb = winSize - (winHalf - 1); % Window Lower Bound
                ub = n - (winHalf - 1); % Window Upper Bound
            end

            % Finding the location of the point and creating the window
            for i = 1:length(ptIdx)
                % If the current point does not have enough points on the
left, then the window is centered on the upper bound
                if ptIdx(i) < lb && ptIdx(i) <= ub
                    for j = 1:(winSize)
                        tSub(j) = t(lb - winHalf + (j - 1));
                        TSub(j) = TempData(lb- winHalf + (j - 1));
                    end

                    findPt = (tSub == t(ptIdx(i)));
                    pt = tSub(findPt);

                % If there are an equal amount of points on both sides of the
current point, then the window is centered on the current point
                elseif ptIdx(i) >= lb && ptIdx(i) <= ub
                    for j = 1:(winSize)
                        tSub(j) = t(ptIdx(i) - winHalf + (j - 1));
                        TSub(j) = TempData(ptIdx(i) - winHalf + (j - 1));
                    end

                    findPt = (tSub == t(ptIdx(i)));
                    pt = tSub(findPt);

                % If the current point does not have enough points on the
right, then the window is centered on the upper bound
                elseif ptIdx(i) >= lb && ptIdx(i) > ub
                    for j = 1:(winSize)
```

```
                        tSub(j) = t(ub - winHalf + (j - 1));
                        TSub(j) = TempData(ub- winHalf + (j - 1));
                    end

                    findPt = (tSub == t(ptIdx(i)));
                    pt = tSub(findPt);
                end

                % Defining the function for the curve fit
                fun = @(c, tSub) c(1) * (F + sum(G .* exp(-Y.^2 .* tSub)))...
                    + c(2) * ((F * tSub) + sum(G .* ((1 / Y.^2) * (1 - exp(-
Y.^ 2 * tSub)))))...
                    + c(3) * ((F * tSub.^2) + sum(2 * G .* ((tSub ./ Y.^2) -
((1 / Y.^4) * (1 - exp(-Y.^ 2 .* tSub))))))...
                    + c(4) * ((F * tSub.^3) + sum(3 * G .* ((tSub.^2 ./ Y.^2)
- ((2 * tSub ./ Y.^4) + ((2 / Y.^6) * (1 - exp(-Y.^ 2 .* tSub)))))));

                % Creating a cubic polynomial fit to the subset to get an
initial guess for the coefficients
                pf = polyfit(tSub, TSub, 3);

                % Using the initial guess to start the curve fit
                c0 = [pf(4) pf(3) pf(2) pf(1)];
                c = lsqcurvefit(fun,c0,tSub,TSub);

                % Populating the pre-processed temperature data array
                T(TIdx) = fun(c, pt);
                TIdx = TIdx + 1;
            end
        %Using original temperature data if window size is 0
        else
            for i = 1:length(ptIdx)
                T(i) = TempData(i);
                TIdx = TIdx + 1;
            end
        end
     end
% Using original temperature data if pre-processing is not required
else
    T = TempData;
end


%% CALCULATING THE SUMMATION TERMS THAT MULTIPLY EACH OF THE COEFFICIENTS AT
THE KNOTS FOR THE 0TH, 1ST, AND 2ND DERIVATIVES

% 0th derivative summation terms
a0Sum = zeros(n, 1);
a1Sum = zeros(n, 1);
a2Sum = zeros(n, 1);
a3Sum = zeros(n, 1);

% 1st derivative summation terms
```

```
a0Sum1D = zeros(n, 1);
a1Sum1D = zeros(n, 1);
a2Sum1D = zeros(n, 1);
a3Sum1D = zeros(n, 1);

% 2nd derivative summation terms
a0Sum2D = zeros(n, 1);
a1Sum2D = zeros(n, 1);
a2Sum2D = zeros(n, 1);
a3Sum2D = zeros(n, 1);

for i = 1:n
    for j = 1:mUpper

        a0Sum(i) = a0Sum(i) + (G(j) * exp(-Y(j)^2 * t(i)));
        a1Sum(i) = a1Sum(i) + (G(j) * ((1 / Y(j)^2) * (1 - exp(-Y(j)^ 2 *
t(i)))));
        a2Sum(i) = a2Sum(i) + (2 * G(j) * ((t(i) / Y(j)^2) - ((1 / Y(j)^4) *
(1 - exp(-Y(j)^ 2 * t(i))))));
        a3Sum(i) = a3Sum(i) + (3 * G(j) * ((t(i)^2 / Y(j)^2) - ((2 * t(i)) /
Y(j)^4) + ((2 / Y(j)^6) * (1 - exp(-Y(j)^ 2 * t(i))))));

        a0Sum1D(i) = a0Sum1D(i) + (-G(j) * Y(j)^2 * exp(-Y(j)^2 * t(i)));
        a1Sum1D(i) = a1Sum1D(i) + (G(j) * exp(-Y(j)^2 * t(i)));
        a2Sum1D(i) = a2Sum1D(i) + (((2 * G(j)) / Y(j)^2) * (1 - exp(-Y(j)^ 2 *
t(i))));
        a3Sum1D(i) = a3Sum1D(i) + (((6 * G(j)) / Y(j)^2) * (t(i) - ((1 /
Y(j)^2) * (1 - exp(-Y(j)^ 2 * t(i))))));

        a0Sum2D(i) = a0Sum2D(i) + (G(j) * Y(j)^4 * exp(-Y(j)^2 * t(i)));
        a1Sum2D(i) = a1Sum2D(i) + (-G(j) * Y(j)^2 * exp(-Y(j)^2 * t(i)));
        a2Sum2D(i) = a2Sum2D(i) + (2 * G(j) * exp(-Y(j)^2 * t(i)));
        a3Sum2D(i) = a3Sum2D(i) + (((6 * G(j)) / Y(j)^2) * (1 - exp(-Y(j)^2 *
t(i))));

    end
end


%% SELECTING THE APPROPRIATE CONDITIONS FOR THE iTH FUNCTION

% Sizing the A and b matrices that are used in Ax = b to solve for the
coefficients
AMat = zeros((4*(n - 1)), (4*(n - 1)));
bMat = zeros((4*(n - 1)), 1);

% Declaring the row/column position
row = 1;
col = 1;

% Assigning and evaluating conditions for each piecewise function
cond = [];
for i = 1:(n - 1)
```

```
    % The ith piecewise function is the first but not the only piecewise
function
    if (i == 1) && (i ~= (n - 1))
        cond = [1, 2, 3, 4, 5];

    % The ith piecewise function is an intermediate piecewise function
    elseif (1 < i) && (i < (n - 1))
        cond = [1, 2, 3, 4];

    % The ith piecewise function is the last but not the only piecewise
function
    elseif (i == (n - 1)) && ((n - 1) ~= 1)
        cond = [1, 2, 6];

    % The ith piecewise function is the only piecewise function
    else
        cond = [1, 2, 5, 6];
    end

    for c = 1:length(cond)

        switch cond(c)

            case 1
            % I.  S_i(t_i) = T_i
            AMat(row, col) = F + a0Sum(i);
            AMat(row, (col + 1)) = (F * t(i)) + a1Sum(i);
            AMat(row, (col + 2)) = (F * t(i)^2) + a2Sum(i);
            AMat(row, (col + 3)) = (F * t(i)^3) + a3Sum(i);
            bMat(row, 1) = T(i);

            row = row + 1;

            case 2
            % II.  S_i(t_i+1) = T_i+1
            AMat(row, col) = F + a0Sum(i + 1);
            AMat(row, (col + 1)) = (F * t(i + 1)) + a1Sum(i + 1);
            AMat(row, (col + 2)) = (F * t(i + 1)^2) + a2Sum(i + 1);
            AMat(row, (col + 3)) = (F * t(i + 1)^3) + a3Sum(i + 1);
            bMat(row, 1) = T(i + 1);

            row = row + 1;

            case 3
            % III.  S_i'(t_i+1) - S_i+1'(t_i+1) = 0
            AMat(row, col) = a0Sum1D(i + 1);
            AMat(row, (col + 1)) = F + a1Sum1D(i + 1);
            AMat(row, (col + 2)) = (2 * F * t(i + 1)) + a2Sum1D(i + 1);
            AMat(row, (col + 3)) = (3 * F * t(i + 1)^2) + a3Sum1D(i + 1);
            AMat(row, (col + 4)) = -a0Sum1D(i + 1);
            AMat(row, (col + 5)) = -(F + a1Sum1D(i + 1));
            AMat(row, (col + 6)) = -((2 * F * t(i + 1)) + a2Sum1D(i + 1));
```

```
            AMat(row, (col + 7)) = -((3 * F * t(i + 1)^2) + a3Sum1D(i + 1));
            %bMat(row, 1) = 0;

            row = row + 1;

            case 4
            % IV.  S_i''(t_i+1) - S_i+1''(t_i+1) = 0
            AMat(row, col) = a0Sum2D(i + 1);
            AMat(row, (col + 1)) = a1Sum2D(i + 1);
            AMat(row, (col + 2)) = (2 * F) + a2Sum2D(i + 1);
            AMat(row, (col + 3)) = (6 * F * t(i + 1)) + a3Sum2D(i + 1);
            AMat(row, (col + 4)) = -a0Sum2D(i + 1);
            AMat(row, (col + 5)) = -a1Sum2D(i + 1);
            AMat(row, (col + 6)) = -((2 * F) + a2Sum2D(i + 1));
            AMat(row, (col + 7)) = -((6 * F * t(i + 1)) + a3Sum2D(i + 1));
            %bMat(row), 1) = 0;

            row = row + 1;

            case 5
            % V.  S_0''(t_0) = 0
            AMat(row, col) = a0Sum2D(i);
            AMat(row, (col + 1)) = a1Sum2D(i);
            AMat(row, (col + 2)) = (2 * F) + a2Sum2D(i);
            AMat(row, (col + 3)) = (6 * F * t(i)) + a3Sum2D(i);
            %bMat(row, 1) = 0;

            row = row + 1;

            case 6
            % VI.  S_n-1''(t_n) = 0
            AMat(row, col) = a0Sum2D(i + 1);
            AMat(row, (col + 1)) = a1Sum2D(i + 1);
            AMat(row, (col + 2)) = (2 * F) + a2Sum2D(i + 1);
            AMat(row, (col + 3)) = (6 * F * t(i + 1)) + a3Sum2D(i + 1);
            %bMat(row, 1) = 0;

            row = row + 1;
        end
    end

    col = (4 * i) + 1;
end

%% SOLVING FOR ALL THE COEFFICIENTS a_i, b_i, c_i, d_i

% Solving the linear system Ax = b for the coefficients
coeff = linsolve(AMat, bMat);

% Initializing the sample increment size and the start and end indexes that
will be used to popluate the sampled time, termperature, and inverse
temperature arrays
sampleInc = 0.001;
```

```
% Initializing the start and finish indexes that will be used to determine
where to popluate the sampled time, termperature, and inverse temperature
arrays for each piecewise function
start = 1;
finish = 0;

% Finding the size needed for the sampled time, termperature, and inverse
temperature arrays
intSize = zeros((n - 1), 1);
for i = 1:(n - 1)
    intSize(i) = (((t(i + 1) - t(i)) * sampleInc));
end

arraySize = ((1 / sampleInc) + 1) * (n - 1); % +1 is needed to include the end
point

% Initializing the sampled time, termperature, and inverse temperature arrays
tsArray = zeros(arraySize, 1);
TsArray = zeros(arraySize, 1);
TinvArray = zeros(arraySize, 1);

% Looping through all the piecewise function coefficients
for i = 1:(n - 1)

    % Assigning the appropriate coefficients to the function from the coeff
vector
    a0 = coeff((4 * i) - 3);
    a1 = coeff((4 * i) - 2);
    a2 = coeff((4 * i) - 1);
    a3 = coeff((4 * i));

    % Using a fine sampling of the interval for each piecewise function
    tsam = t(i):intSize(i):t(i + 1);

    % Setting the finish index for the next instance
    finish = finish + length(tsam);

    % Recalculating the 0th derivative summation terms for each time sample
point tsam
    a0SumT = 0;
    a1SumT = 0;
    a2SumT = 0;
    a3SumT = 0;

    for j = 1:mUpper
        a0SumT = a0SumT + (G(j) * exp(-Y(j)^2 * tsam));
        a1SumT = a1SumT + (G(j) * ((1 / Y(j)^2) * (1 - exp(-Y(j)^ 2 *
tsam))));
        a2SumT = a2SumT + (2 * G(j) * ((tsam / Y(j)^2) - ((1 / Y(j)^4) * (1 -
exp(-Y(j)^ 2 * tsam)))));
        a3SumT = a3SumT + (3 * G(j) * ((tsam.^2 / Y(j)^2) - ((2 * tsam) /
Y(j)^4) + ((2 / Y(j)^6) * (1 - exp(-Y(j)^ 2 * tsam)))));
```

```
    end

    % Calculating the temperature point at each time sample point
    Tsam = (a0 * (F + a0SumT)) + (a1 * ((F * tsam) + a1SumT)) + (a2 * ((F *
tsam.^2) + a2SumT)) + (a3 * ((F * tsam.^3) + a3SumT));

    % Calculating the inverse temperature point at each time sample point
    Tinv =  a0 + (a1 * tsam) + (a2 * tsam.^2) + (a3 * tsam.^3);

    % Storing time, temperature, and inverse temperature in their respective
arrays
    % Shared node values will be repeated in the arrays due to each inverse
piecewise polynomial producing slightly different inverse temperature values
due to error
    tsArray(start: finish) = tsam;
    TsArray(start: finish) = Tsam;
    TinvArray(start: finish) = Tinv;

    % Setting the next starting index to be where this instance finished
    start = finish + 1;
end

% Plotting the thermal spline and inverse solution
plot(t, T,'ko');
hold on
plot(tsArray, TsArray,'b');
hold on;
plot(tsArray, TinvArray,'r');


% Post-processing the Inverse data if required
if postProc == 'Y'
    hold on
    pp = csaps(tsArray, TinvArray, sp);
    fnplt(pp, '--m');
end

% Adding a title, axis labels, and a legend
title('Thermal Spline and Inverse Solution');

xlabel('time');
ylabel('Temperature');

legend('Response Temp Data', 'Thermal Spline', 'Piecewise Inverse Solution');
```

References

[1] Woodbury, K., Najafi, H., De Monte, F., & Beck, J. V. (2023). Inverse Heat Conduction: Ill-Posed Problems (2nd ed.). *John Wiley & Sons*, 7-14.

[2] Chen, S. Y. (1961). One-dimensional heat conduction with arbitrary heating rate. *Journal of the Aerospace Sciences*, *28*(4), 336-337.

[3] Nied, H. F. (1987). Thermal shock in an edge-cracked plate subjected to uniform surface heating. *Engineering Fracture Mechanics*, *26*(2), 239-246.

[4] Nied, H. F., & Erdogan, F. (1983). Transient thermal stress problem for a circumferentially cracked hollow cylinder. *Journal of thermal stresses*, *6*(1), 1-14.

[5] Austin, J. B. (1931). Temperature distribution in solid bodies during heating or cooling. *Physics*, *1*(2), 75-83.

[6] Pisarenko, G. S., Gogotsi, G. A., & Grushevskii, Y. L. (1978). A method of investigating refractory nonmetallic materials in linear thermal loading. *Strength of Materials*, *10*(4), 406-413.

[7] Imber, M. (1973). A temperature extrapolation method for hollow cylinders. *AIAA Journal*, *11*(1), 117-118.

[8] Imber, M. (1979). Inverse problem for the solid cylinder. *AIAA Journal*, *17*(1), 91-94.

[9] Blanc, G., & Raynaud, M. (1996). Solution of the inverse heat conduction problem from thermal strain measurements.

[10] Yang, Y. C., Chen, U. C., & Chang, W. J. (2002). An inverse problem of coupled thermoelasticity in predicting heat flux and thermal stresses by strain measurement. *Journal of thermal stresses*, *25*(3), 265-281.

[11] Burggraf, O. R. (1964). An exact solution of the inverse problem in heat conduction theory and applications.

[12] Taler, J. (1997). Analytical solution of the overdetermined inverse heat conduction problem with an application to monitoring thermal stresses. *Heat and mass transfer*, *33*(3), 209-218.

[13] Segall, A. E., Schoof, C. C., & Yastishock, D. E. (2020). Thermal solutions for a plate with an arbitrary temperature transient on one surface and convection on the other: direct and inverse formulations. *Journal of Pressure Vessel Technology*, *142*(5), 051301.

[14] Frank, I. (1963). An application of least squares method to the solution of the inverse problem of heat conduction.

[15] Vedula, V. R., Segall, A. E., & Rangarajan, S. K. (1998). Technical Note Transient analysis of internally heated tubular components with exponential thermal loading and external convection. *International journal of heat and mass transfer*, *41*(22), 3675-3678.