Brigham Young University
BYU ScholarsArchive

All Theses and Dissertations

2012-11-27

Measurement of Thermal Diffusivities Using the Distributed Source, Finite Absorption Model

James B. Hall
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Mechanical Engineering Commons

BYU ScholarsArchive Citation

Hall, James B., "Measurement of Thermal Diffusivities Using the Distributed Source, Finite Absorption Model" (2012). *All Theses and Dissertations*. 3467.
https://scholarsarchive.byu.edu/etd/3467

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

Measurement of Thermal Diffusivities Using the Distributed Source, Finite Absorption Model

James B. Hall

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

Master of Science

Matthew R. Jones, Chair
W. Jerry Bowman
Steven E. Gorrell

Department of Mechanical Engineering

Brigham Young University

November 2012

Copyright © 2012 James B. Hall

All Rights Reserved

ABSTRACT

Measurement of Thermal Diffusivites Using the
Distributed Source, Finite Absorption Model

James B. Hall
Department of Mechanical Engineering, BYU
Master of Science

Thermal diffusivity in an important thermophysical property that quantifies the ratio of the rate at which heat is conducted through a material to the amount of energy stored in a material. The pulsed laser diffusion (PLD) method is a widely used technique for measuring thermal diffusivities of materials. This technique is based on the fact that the diffusivity of a sample may be inferred from measurement of the time-dependent temperature profile at a point on the surface of a sample that has been exposed to a pulse of radiant energy from a laser or flash lamp.

An accepted standard approach for the PLD method is based on a simple model of a PLD measurement system. However, the standard approach is based on idealizations that are difficult to achieve in practice. Therefore, models that treat a PLD measurement system with greater fidelity are desired. The objective of this research is to develop and test a higher fidelity model that more accurately represents the spatial and temporal variations in the input power. This higher fidelity model is referred to as Distributed Source Finite Absorption (DSFA) model.

The cost of the increased fidelity associated with the DSFA model is an increase in the complexity of inferring values of the thermal diffusivity. A new method of extracting values from time dependent temperature measurements based on a genetic algorithm and on reduced order modeling was developed. The primary contribution of this thesis is a detailed discussion of the development and numerical verification of this proposed new method for measuring the thermal diffusivity of various materials.

Verification of the proposed new method was conducted using numerical experiments. A detailed model of a PLD system was created using advanced engineering software, and detailed simulations, including conjugate heat transfer and solution of the full Navier-Stokes equations, were used to generate multiple numerical data sets. These numerical data sets were then used to infer the thermal diffusivity and other properties of the sample using the proposed new method. These numerical data sets were also used as inputs to the standard approach. The results of this verification study show that the proposed new method is able to infer the thermal diffusivity of samples to within 4.93%, the absorption coefficient to within 10.57 % and the heat capacity of the samples to within 5.37 %. Application of the standard approach to these same data sets gave much poorer estimates of the thermal diffusivity, particularly when the absorption coefficient of the material was relatively low.

Keywords: heat transfer, thermal diffusivity, reduced order modeling, genetic algorithm

ACKNOWLEDGMENTS

I would like to thank Dr. Matthew Jones for his constant counsel, guidance, and help in my returning to school after working in industry. His help in this project and many other academic ventures have been critical in my educational pursuits. I appreciate his patience and willingness in the process that this research has taken. I acknowledge Dr. Vladimir Solovjov and his assistance in the mathematical realm of thermal transport. I also thank Dr. Steven Gorrell and Dr. Jerry Bowman for their gracious help in this pursuit. I am also very grateful to my fellow BYU students Jeremy LeFevre, Trevor Terrill and Joshua Thornock for their help in developing and testing the FLUENT solutions that became the basis for validation for the DSFA model and created blind tests. Without their selfless efforts, this project could never have been completed.

Many times, this research was made possible by the loving support of those who are around me. I would like to thank the support of my family and friends, especially my wife Melony. Their patience and support during this work will not soon be forgotten. I also recognize the hand of my Heavenly Father, as there have been many answers to prayers in regards to making my research successful.

I would like to thank the Mechanical Engineering Department and Brigham Young University for giving me the opportunity to continue my educational pursuits and expand my learning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Latin Characters

A – Solution set matrix

A – Nondimensional equation

A_s – Exposed surface area

a – Nondimensional diffusivity

B – Expansion coefficient matrix

B – Nondimensional equation

b – Expansion coefficient vector

b – Expansion coefficient

Bi – Biot number

C – Coefficient matrix

C – Nondimensional equation

c – Heat capacity

D – Nondimensional equation

F – Interpolation matrix

F – Nondimensional equation

f – Interpolation vector

f – Laser power duration function

G – Nondimensional equation

g – Thickness energy is absorbed in

H – Incident irradiation from source

h – Convection coefficient

$\mathbf{I}$ – Identity matrix

i_{Gen} – Generation number index

J – Bessel function

K – Coefficient corresponding to rise time

$\mathbf{k}$ – Arbitrary case of parameters

k – Thermal conductivity

L – Sample thickness

M – Number of time steps

m_c – Matrix cutoff value

N – Number of parameter sets

N_{Gen} – Number of generations

N_T – Number of terms

n – Index

P – Laser power

parameter – Input value from parameter set

Q – Incident energy

$\dot{q}$ – Volumetric heat generation

R – Outside radius of sample

R_n – Expansion factor

r – Radial coordinate

r_0 – One standard deviation from center of laser power

r_A – Rank of $\mathbf{A}$

r_F – Rank of $\mathbf{F}$

r_m – Initial probability of mutation

$\mathbf{S}$ – Diagonal matrix of singular values

S – Optical depth

T – Temperature

t – Time

$\mathbf{U}$ – Orthogonal matrix

$\mathbf{V}$ – Orthogonal matrix

V – Normalized temperature rise

V_s – Sample volume

x_1, x_2, x_3 – Governing parameters

y – Output value for parameter set

$\mathbf{Z_k}$ – Vector of temperature profile

Z_m – Expansion function

z – Vertical coordinate

Greek Characters

α – Thermal diffusivity

β – Time to maximum power in laser pulse

β_m – Eigenvalue

ε – Convergence parameter

ζ – Nondimensional thickness

θ – Nondimensional temperature

κ – Absorption coefficient

λ_n – Eigenvalue

ν – Nondimensional radius

ν_0 – Nondimensional one standard deviation from center of laser power

ρ – Density

$\mathbf{\Sigma}$ – Diagonal matrix

σ – Singular values

τ – Nondimensional time

Φ – Basis set matrix

ϕ – Angular coordinate

χ – Integration parameter

ω – Normalized rise time

Subscripts

$_0$ – Zero index

$_{0.5}$ – Temperature half rise time

$_1$ – First parameter set

$_2$ – Second parameter set

$_A$ – From **A** matrix

$_{DSFA}$ – From DSFA values

$_F$ – From **F** matrix

$_i$ – Index

$_{initial}$ – Initial state

$_{input}$ – Input parameter

$_M$ – Maximum

$_m$ – Index

$_{maxrise}$ – Point of maximum temperature

$_n$ – Index

$_p$ – End of pulse

$_x$ – Percent rise of temperature

$_\beta$ – Point of maximum pulse power

$_\infty$ – Initial or surrounding

Superscripts

^{i} – Index

^{j} – Index

^{m} – Index

^{n} – Index

$^\top$ – Transpose

$^+$ – Moore-Penrose Pseudoinverse

1 INTRODUCTION

1.1 Motivation

The thermal diffusivity of a material

$$\alpha = \frac{k}{\rho c} \qquad \textbf{(1-1)}$$

is an important physical property that quantifies the ratio of the rate at which heat is conducted through the material relative to the amount of energy that is stored in the material. A material with a large diffusivity will more quickly come to thermal equilibrium when the temperature of its surroundings fluctuates, while materials that have small diffusivities take longer to come to equilibrium. Values for thermal diffusivities vary from 10^{-3} m^2/s for metallic materials to 10^{-7} m^2/s for nonmetallic materials.

Accurate knowledge of the thermal diffusivity of a material is critical in selecting materials to meet design specifications, in modeling transient thermal transfer, and in calibrating temperature sensors and heat flux gauges [Hay et al]. In addition to their intrinsic worth, measurements of diffusivity are frequently used in conjunction with independent measurements of the materials density and specific heat to calculate the thermal conductivity of materials [ASTM]. The thermal conductivity of a material is an important thermal transport property, and accurate thermal conductivity measurements are essential to the analysis of conduction heat transfer.

Since the thermal diffusivity quantifies the rate at which a material comes to thermal equilibrium with its surroundings, a logical approach to measuring α is to perturb the temperature of a sample and to monitor its time-dependent temperature as it returns to equilibrium. The Flash Method [Hay et al., ASTM] is a common implementation of this approach. Initially proposed by Parker in 1961 [Parker], the underlying premise of the flash method is very straightforward. One face of a thin sample is exposed to a pulse of radiant energy which is emitted by a laser or by a flash lamp. The resulting time-dependent temperature profile at a point on the opposite face is recorded, and comparison of this measured profile with the temperature profile generated by a thermal model of the process is used to infer the diffusivity of the material.

Over the years, the theory and application of the flash method have been refined and improved by a number of investigators [Hay et al., Maglić, Vozar, Baba]. In particular, non-ideal phenomena occurring in practical measurement systems have been treated with greater fidelity through the use of increasingly complex thermal models. However, the complexity and computational expense of these algorithms used to infer the thermal diffusivity from transient temperature measurements have increased in parallel with the increased complexity of the thermal models.

Methods such as the flash method fall into a class of problems known as parameter estimation [Tarantola], which is a type of inverse problem. Problems of this class rely on a model of a dynamic process, which is generally expressed as a partial differential equation. This governing equation involves a dependent variable and one or more independent variables. There are sources or sinks and properties of the system control how variations in these and the independent variables influence the dependent variable. The dependent variable in the governing equation is often referred to as the state variable and the properties are generally referred to as parameters. State variables are observable, meaning they can be measured directly, whereas parameters are generally not observable. Therefore, estimation of a parameter requires postulating a relationship between the state variable and the targeted parameter. This relationship is then used to infer the parameter based on measurements of the state variable [Beck and Arnold]. In the flash method, the state variable is the time-dependent temperature profile at some location within the sample, and the targeted parameter is the thermal diffusivity.

The objective of the research presented in this thesis is to develop a parameter estimation algorithm for a pulsed laser diffusivity measurement (PLD) system based on reduced order modeling (ROM) and a Genetic Algorithm (GA). Tests of the proposed data reduction algorithm are conducted using simulated measurements. The proposed algorithm is shown to be superior to the parameter estimation method recommended in the most recent ASTM standard [ASTM].

1.2 **Outline**

The model currently implemented in the ASTM standard for PLD measurement systems is developed and described in Chapter 2. Limitations associated with the ASTM standard approach are clearly revealed by thoroughly examining its theoretical basis and derivations of its fundamental equations.

Chapter 3 presents a more physically realistic thermal model which has been developed for this study. This model is called the Distributed Source Finite Absorption (DSFA) model. This model incorporated the effects of three phenomena neglected by the ASTM standard – heat loss off the top surface of the sample, finite laser pulse durations, and nonuniform heating of the sample. These three phenomena have been identified as the most significant by Vozar and Hohenauer [Vozar]. Although an analytical model is obtained, evaluation of this high fidelity model is computationally expensive, requiring approximately 0.85 seconds using a desktop PC with 2.30 GHz processor and 2.00 GB of RAM. It is also not possible to develop a simple relationship between measured temperatures and thermal diffusivities based on the DSFA model. A random search procedure utilizing a genetic algorithm (GA) is described and implemented to solve the inverse problem. Due to the computational requirements of the GA, a reduced order model (ROM) has been developed to use in conjunction with the GA, resulting in a more computationally efficient approach. The primary result described in Chapter 3 is the demonstration that the inverse problem of evaluating properties from a temperature profile can be solved efficiently using ROM with a GA.

Chapter 4 contains results of five blind test cases in which temperature profiles generated by FLUENT were used as inputs. Thermal diffusivity values were obtained using the proposed method are compared with the actual values and to values obtained using the ASTM standard. The results of these blind test cases indicate that the DSFA model based ROM with GA is more robust and accurate than the ASTM standard.

Chapter 5 summarizes the work done and gives recommendations for further investigations. Several appendices follow the list of references. The appendices contain the derivation of the Parker model – the base of the ASTM method, complete derivation of the DSFA, Matlab source codes that were written to run the DSFA model, the GA, and the ROM.

2 PREVIOUS PULSED LASER DIFFUSION MODELS

2.1 **Introduction**

Thermal diffusivity measurements based on the laser flash method, also known as Pulsed Laser Diffusion (PLD), were first introduced in 1961 by Parker et al [Parker]. It is a method to find the thermal diffusivity with use of a laser pulse on the surface of a material sample. Parker et al. theorized that one could pulse a laser on a material sample and based on the thermal response of the rear surface calculate the diffusivity of the sample. PLD is currently being used widely in material science and composites [Vozar], carbon nanotubes [Haydari], and the electronics industry [Fullem].

Over the years, many researchers have improved the model originated by Parker. There have been modifications to the analytical solutions to try and improve accuracy and reliability of the models [Maglić]. There is also discussion in the literature of the effects of the assumptions of the Parker model [Hay], [McMasters]. With all the work that has been done, the Parker model is still the backbone of the ASTM standard method used today [ASTM].

2.2 **Parker Model Development**

The approach developed by Parker is based on a simple model [Parker]. The thermal model used is a one dimensional model of temperature profile in the material sample. A schematic of this model is shown in Figure 2-1.

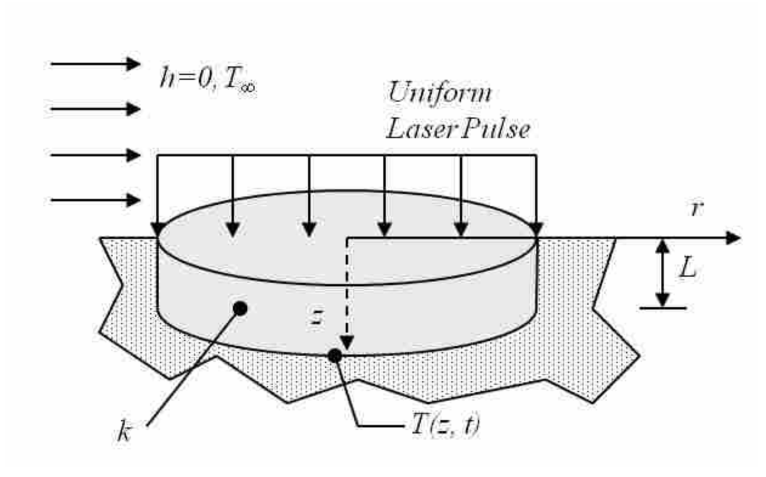


Figure 2-1: Parker Model Schematic

The development of the thermal model starts with the heat equation. The heat equation with volumetric heat generation in radial coordinates is shown in Eq. (2-1).

$$\frac{1}{r}\frac{\partial}{\partial r}\left(kr\frac{\partial T}{\partial r}\right)+\frac{1}{r^2}\frac{\partial}{\partial \phi}\left(k\frac{\partial T}{\partial \phi}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+\dot{q}=\rho c\frac{\partial T}{\partial t} \tag{2-1}$$

Based on symmetry, Parker's model neglects temperature variations in the the r and ϕ directions. Parker also assumes no internal heat generation as all the energy is absorbed in a very small thickness on top of the sample designated by δ. These assumptions reduce the heat equation from Eq. (2-1) to Eq. (2-2).

$$\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)=\rho c\frac{\partial T}{\partial t} \tag{2-2}$$

Further simplifications are achieved by assuming constant properties such that *k*, *ρ* and *c* are uniform and independent of temperature giving Eq. (2-3).

$$\alpha \frac{\partial^2 T}{\partial z^2} = \frac{\partial T}{\partial t} \tag{2-3}$$

Assuming that the pulse is absorbed instantaneously at the surface leads to the assumption that the initial temperature profile may be approximated by the step function given in Eq. (2.4).

$$T(z, t=0) = \begin{cases} T_\infty + \dfrac{Q}{\rho c \delta A} & 0 \le z \le \delta \\ T_\infty & \delta < z \le L \end{cases} \tag{2-4}$$

The solution to this partial differential equation is an infinite series solution. The derivation of the solution is shown in Appendix A. The solution in infinite series form is given in Eq. (2-5).

$$T(L,t) = \frac{Q}{\rho c L}\left(1 + 2\sum_{n=1}^{\infty}(-1)^n \exp\left(\frac{-n^2\pi^2}{L^2}\alpha t\right)\right) \tag{2-5}$$

There are two dimensionless parameters that are introduced, *V* and *ω*, as shown in Eq. (2-6) and Eq. (2-7) where T_M is the maximum temperature from the laser pulse that is reached by the rear face.

$$V(L,t) = T(L,t)/T_M \quad \textbf{(2-6)}$$

$$\omega = \pi^2 \alpha t / L^2 \quad \textbf{(2-7)}$$

Combining Eq. (2-5), Eq. (2-6), and Eq. (2-7) yields Eq. (2-8)

$$V = 1 + 2\sum_{n=1}^{\infty}(-1)^n \exp\left(-n^2\omega\right) \quad \textbf{(2-8)}$$

This yields the common form that is used for determining the diffusivity based on the time that is takes the sample to go half way to its maximum value after the laser pulse, $t_{0.5}$, as shown in Eq. (2-9) as the nondimensional temperature rise, V, and nondimensional time, ω, are shown in Figure 2-2.

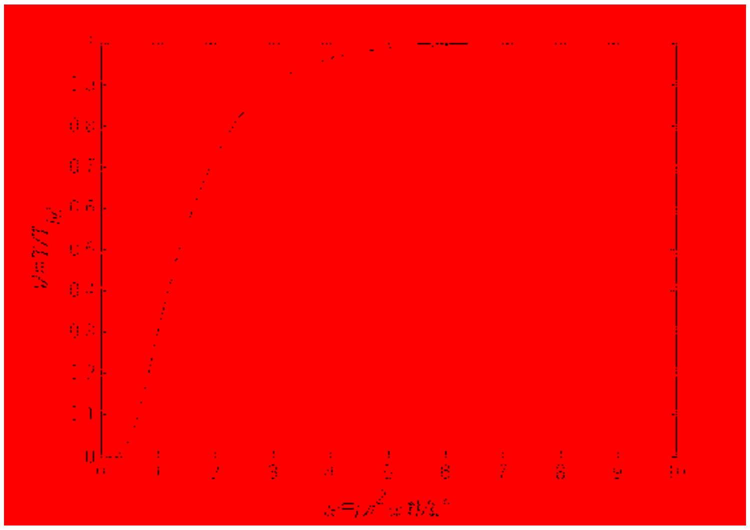


Figure 2-2: Nondimensional Temperature Rise from Parker Equations

As is shown in Appendix A, when *V* is set to be 0.5, or half the maximum temperature rise, the value for ω is 1.3698. Once a temperature profile of a sample of thickness *L* is known in time, one just needs to determine the time taken to reach the midpoint temperature, and α may be directly evaluated with Eq. (2-9).

$$\alpha = \frac{1.3698L^2}{\pi^2 t_{0.5}} \tag{2-9}$$

2.3 **Maglić Method**

The method introduced by Maglić in 1992 [Maglić] is based on the same model as Parker [Parker]. Due to having the same base model, all the same assumptions carry over from the Parker model. The novelty of this method is that it does not restrict the calculation of only the time at half the temperature rise. It allows for finding the time to reach a number of different percent rises along the profile and averaging the returned values of α, which gives a more accurate measurement of the thermal diffusivity. With the increase in flexibility of the method, the equation for α is slightly different than that of Parker in Eq. (2-9). The Maglić method equation is shown in Eq. (2-10) Where K_x is a constant that corresponds to an x percent rise, and t_x is the time for the temperature to get to that x percent rise.

$$\alpha = \frac{K_x L^2}{t_x} \tag{2-10}$$

Values of K_x and percent temperature rise, x, are shown in Table 2-1 [Maglić]. It is proposed by Maglić to not just use one single point, but to use as many as possible to reduce the error in the system and provide for more accurate evaluations of the diffusivity.

Table 2-1: Values of K_x for Eq. (2-10) for Constant Laser Power [Maglić]

x (%)	K_x	*x (%)*	K_x
10	0.066108	60	0.162236
20	0.084251	66	0.181067
25	0.092725	70	0.191874
30	0.101213	75	0.210493
33	0.106976	80	0.233200
40	0.118960	90	0.303520
50	0.138785		

2.4 ASTM Method

The ASTM method also uses the same model as Parker [ASTM]. Similarly with the Maglić method, the same base assumptions from the Parker model also carry into the ASTM method. The calculation that it uses for determining α is the same as Eq. (2-10) with the values as shown in Table 2-2.

Table 2-2: Values of K_x for Eq. (2-10) for Constant Laser Power in ASTM Method

x (%)	K_x
25	0.092725
50	0.138785
75	0.210493

It is clear that the ASTM method is a balance between the Parker and Maglić methods. The value of 0.138785 is the same as the Parker where $0.138785 = 1.36975/\pi^2$. The value of 0.138785 corresponds to the value of the 50% temperature rise as is seen in Table 2-1. The

ASTM method finds the balance of the Parker and Maglić methods by stating that to check the validity of the measurement, use the equation developed by Maglić, Eq. (2-10), at a minimum of two other points along the profile of the rise curve, generally the 25% and 75% rise times as shown in Table 2-2. With the three measurements it allows a moderate level of confidence that the value returned is the diffusivity if they correlate within 5%. If the values do not correlate within the given level, then the response curve is to be analyzed to determine what effects are causing the error in the solution. The ASTM standard mentions that if errors are large, look at effects of a non-infinitesimally short laser pulse, radiant effects, and non-uniform heating [ASTM]. There are a number of corrections that are made to the model to account for these influences as stated previously in efforts to improve the solution.

2.5 Summary of Previous Models

It is seen that the Parker [Parker] model is the basis for PLD system calculation. The temperature of the rear surface is measured after a laser pulse. The time required for the measurement to get to a predetermined percent of the temperature rise is then used in an equation to return the diffusivity of the substance. This can be done with one point in time, or many points in time, but the core idea is the same.

This base model involves a number of simplifying assumptions. There are assumptions about the laser pulse power being constant and infinitesimally short. There are assumptions about the material sample absorbing all the energy uniformly in a thin layer on the top surface, that the heat transfer in the sample is one-dimensional, and that the properties of the material are constant. There is also the assumption that convective and radiative heat losses from the top surface are negligible. Each of these assumptions simplifies the physics of the problem and removes some of the physical realism of the model. A new model for finding α is desired that does not use all the assumptions that are used in the current models so that the accuracy of the system can be improved. With a new higher-fidelity model, the temperature profiles are simulated more accurately. More accurate temperature profiles should allow for more accurate measurement of the diffusivity of the material.

3 PARAMETER IDENTIFCATION BASED ON AN IMPROVED PLD MODEL

3.1 Parameter Identification

The objective of this study is to develop an improved PLD model that may be used to determine the thermal diffusivity and possibly other material parameters based on measurements of the time-dependent temperature profile on the surface of the sample. The proposed new model is referred to as the Distributed Source Finite Absorption (DSFA) model, because it accounts for spatial variations in the pulse and in the absorption of the pulse within the sample. The DSFA model also accounts for convective and radiative heat losses from the uninsulated top surface of the sample, as illustrated in Figure 3-1. Note that for convenience the origin of the coordinate system is placed at the center of the lower surface instead of at the center of the top surface, where it was located in developing the Parker model as shown in Figure 2-1.

The DSFA model requires four inputs. In additions to the heat transfer coefficient, h, which quantifies both convective and radiative heat loss from the uninsulated upper surface, the DSFA model depends on the following material properties of the sample:

- The thermal diffusivity, α
- The heat capacity, ρc
- The absorption coefficient, κ

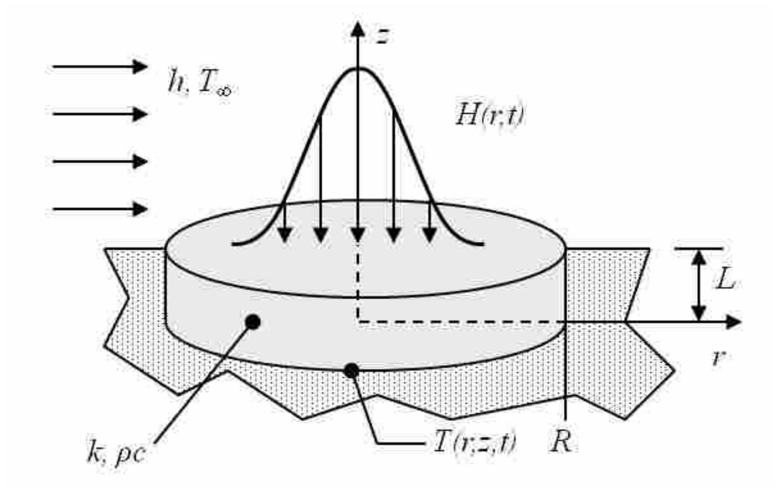


Figure 3-1: DSFA Model Schematic

The DSFA model produces the time-dependent, axisymetrical temperature profile in the sample, *T(r,z,t)*. However, it is desirable to use the time-dependent temperature profile at a simple point on the lower surface of the sample as data that may be used to measure the thermal diffusivity and other input parameters. For convenience, the origin will be used as the data acquisition point. A schematic of the forward problem is shown in Figure 3-2. The coordinate system used in this work is also shown in Figure 3-1.

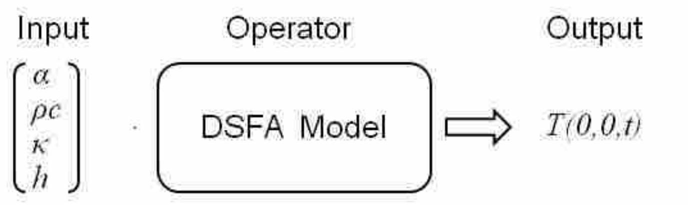


Figure 3-2: Forward Model Schematic

With fewer assumptions, the DSFA model more accurately represents the heat transfer process occurring in an actual PLD system.

It is desirous to be able to take the output from the model, *T(0,0,t)*, and perform the inverse operation to return results of the thermal diffusivity, heat capacity, absorption coefficient and heat transfer coefficient. The inverse operation is illustrated schematically in Figure 3-3.

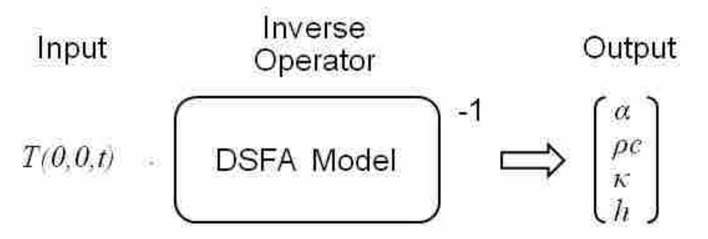


Figure 3-3: Inverse Model Schematic

3.2 **Inversion of the DSFA Model**

Because the DSFA model represents a PLD system with greater fidelity, this improved model does not result in a simple relationship between the targeted parameter, thermal diffusivity, and the measured temperature profile as was the case for the simplified Parker model. Therefore, an iterative, random search process based on a genetic algorithm (GA) [Ostrowski] is proposed. A schematic representation of the inversion of the DSFA model is shown in Figure 3-4.

Due to the complexity of the DSFA model, each run is computationally expensive – a typical MATLAB run requires approximately 0.85 seconds on a 2.3 GHz processor. In running a GA, the model is called many times. Solution of the inverse problem using a GA typically requires that the DSFA model can be executed more than 7500 times. Performing the operation from Figure 3-4 would require more than 17 hours to complete.

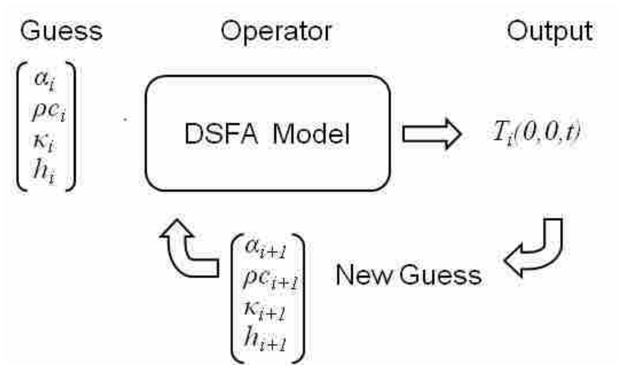

Guess
Operator
Output
α_i ρc_i κ_i h_i
DSFA Model
$T_i(0,0,t)$
α_{i+1} ρc_{i+1} κ_{i+1} h_{i+1}
New Guess

Figure 3-4: Inversion of the DSFA Model

To speed the process of the GA, a highly-accurate reduced order model (ROM) is developed. The ROM is able to simulate the DSFA model with high levels of accuracy over the entire profile and is much faster than the DSFA model. The modified process takes the computationally expensive DSFA process that is shown in Figure 3-4 and replaces it with a ROM in order to solve the inverse problem, as shown in Figure 3-5.

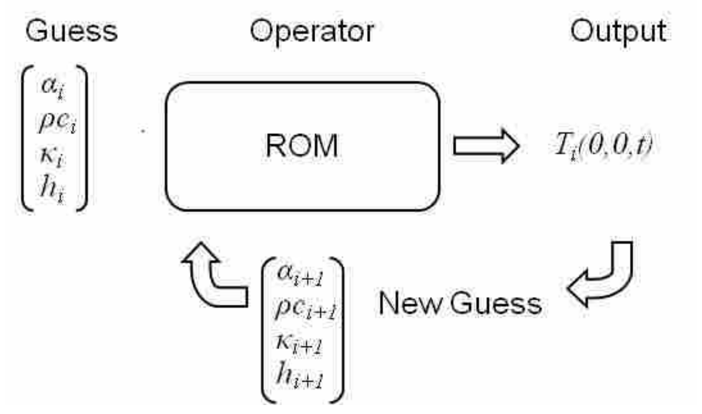

Guess
Operator
Output
α_i ρc_i κ_i h_i
ROM
$T_i(0,0,t)$
α_{i+1} ρc_{i+1} κ_{i+1} h_{i+1}
New Guess

Figure 3-5: Inversion of the DSFA Model using ROM

Using the ROM in place of the DSFA allows the inverse problem to be solved in a fraction of the time. A typical MATLAB run of the ROM on a 2.3 GHz processor is only 0.0025 seconds. Since running the DSFA requires approximately 0.85 seconds, the inverse problem with ROM can be solved in 0.3 percent of the time that would be required to solve the problem using the DSFA, or about three minutes. Since the ROM is so much faster than the DSFA model, it is more efficient to utilize the ROM as the simulation technique, although either model can return accurate solutions.

3.3 **Model Definition**

The DSFA model captures more of the true physics of the problem by the use of more appropriate assumptions. Using assumptions that more closely match the actual physics should allow for more accurate data. As shown in Figure 3-6, the top surface of the sample is not insulated, so there will be heat loss due to convection and radiation from this surface. These losses are modeled using *h* as the heat transfer coefficient. The bottom and outside circumferential surfaces are well insulated and assumed to be adiabatic. The laser irradiation is modeled as a Gaussian profile in *r* and as a triangle pulse in *t*. This model of the temporal variation in the pulse follows the recommendations of ASTM [ASTM].

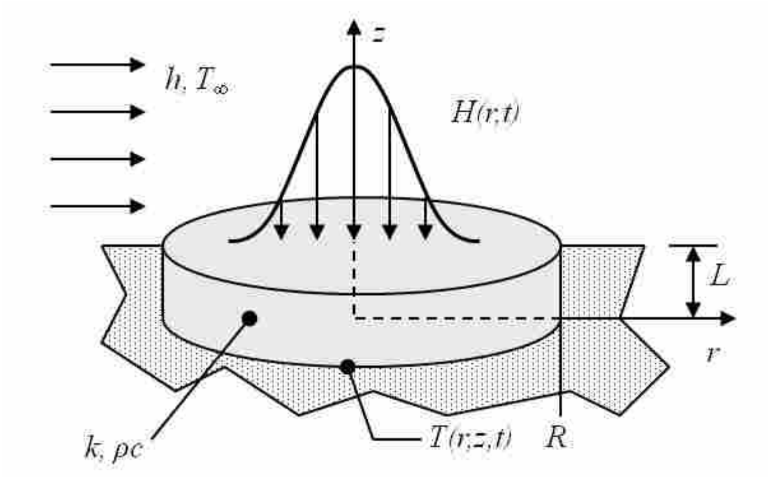


Figure 3-6: DSFA Model of PLD Schematic

As in other PLD models, the thermal and physical material properties are assumed to be uniform constants.

3.4 **DSFA Model Equation Development**

The equation that determines the spatial and time dependent temperature profiles in the sample are shown in Eq. (3-1).

$$\frac{1}{r}\frac{\partial}{\partial r}\left(kr\frac{\partial T}{\partial r}\right)+\frac{1}{r^2}\frac{\partial}{\partial \phi}\left(k\frac{\partial T}{\partial \phi}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+\dot{q}=\rho c\frac{\partial T}{\partial t} \qquad \textbf{(3-1)}$$

Since the model is axisymetric, temperature variations in the ϕ direction are neglected. The constant property assumption is also implemented to yield Eq. (3-2).

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right)+\frac{\partial^2 T}{\partial z^2}+\frac{\dot{q}}{k}=\frac{1}{\alpha}\frac{\partial T}{\partial t} \qquad \textbf{(3-2)}$$

The volumetric heat generation, $\dot{q}$, is dependent on the laser pulse size, radial position, and the axial position into the material due to optical depth. Eq. (3-3) gives the function of $\dot{q}$ where *P* is the laser power, κ is the absorption coefficient, r_o is a measure of the radius of the beam, and *f(t)* is the laser power duration function.

$$\dot{q}=\frac{P\kappa}{\pi r_o^{\,2}}\exp\left(\frac{-r^2}{r_o^{\,2}}-\kappa(L-z)\right)f(t) \qquad \textbf{(3-3)}$$

The function, *f(t)* represents how the strength of the pulse varies in time. It is suggested in the literature that a triangle pulse be used [ASTM]. The strength of the pulse is assumed to increase linearly from zero at $t = 0$ to a maximum, at $t = t_m$. From $t = t_m$, the strength of the pulse decreases linearly back to zero at $t = t_p$. The values for t_m and t_p may be determined by measuring the time dependence of the laser pulse power with an optical detector. This is dependent on the system that is being used as not all lasers will have the same temporal characteristics. Eq. 3-4 gives *f(t)* for the recommended profile and *f(t)* is plotted in Figure 3-7.

$$f(t) = \left\{ \begin{array}{cc} \dfrac{t}{t_m} & 0 < t \le t_m \\ \dfrac{t_p - t}{t_p - t_m} & t_m < t \le t_p \\ 0 & t > t_p \end{array} \right\} \qquad \textbf{(3-4)}$$

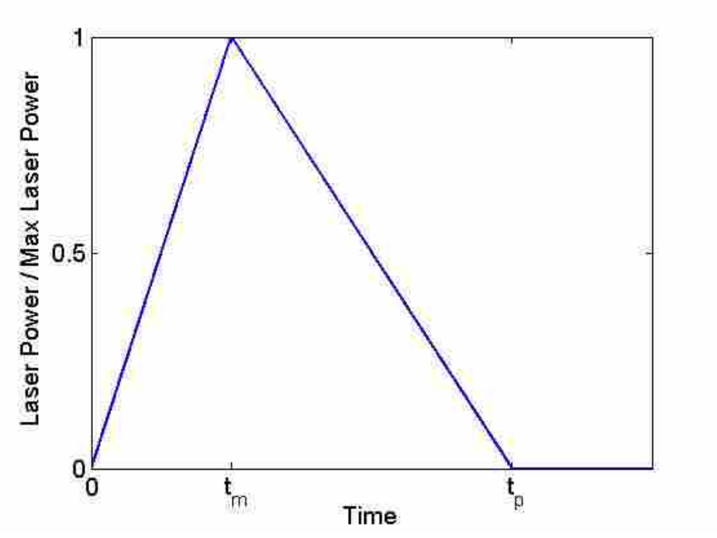


Figure 3-7: Triangle Pulse of the Laser

For a second order partial differential equation in two dimensions with a time dependence, four boundary conditions are required and also one initial condition. The boundary conditions and initial condition are shown in Eq. (3-5) and Eq. (3-6).

$$T(r,z,0) = T_\infty \tag{3-5}$$

$$\begin{aligned}
\left.\frac{\partial T}{\partial r}\right|_{r=0} &= 0 \\
\left.\frac{\partial T}{\partial r}\right|_{r=R} &= 0 \\
\left.\frac{\partial T}{\partial z}\right|_{z=0} &= 0 \\
-k\left.\frac{\partial T}{\partial z}\right|_{z=L} &= h\left(T(r,L,t) - T_\infty\right)
\end{aligned} \tag{3-6}$$

In order to simplify the solution, the problem is nondimensionalized as outlined in the Appendix B. The nondimensional equation is Eq. (3-7).

$$\frac{1}{\upsilon}\frac{\partial}{\partial \upsilon}\left(\upsilon\frac{\partial \theta}{\partial \upsilon}\right) + a^2\frac{\partial^2 \theta}{\partial \zeta^2} + \exp\left(\frac{-\upsilon^2}{\upsilon_o^{\ 2}} - S(1-\zeta)\right) f(\tau) = \frac{\partial \theta}{\partial \tau} \tag{3-7}$$

The boundary conditions, initial condition, and time dependent laser pulse function are also nondimensionalized. The nondimensional boundary condition is Eq. (3-8), the initial condition is Eq. (3-9), and the laser pulse function in Eq. (3-10). The Biot number, *Bi*, is given by *Bi=hL/k*.

$$\left.\frac{\partial\theta}{\partial\upsilon}\right|_{\upsilon=0}=0$$
$$\left.\frac{\partial\theta}{\partial\upsilon}\right|_{\upsilon=1}=0$$
$$\left.\frac{\partial\theta}{\partial\zeta}\right|_{\zeta=0}=0 \qquad \textbf{(3-8)}$$
$$\left.\frac{\partial\theta}{\partial\zeta}\right|_{\zeta=1}=-Bi\,\theta(\upsilon,1,\tau)$$

$$\theta(\upsilon,\zeta,0)=0 \qquad \textbf{(3-9)}$$

$$f(\tau)=\begin{cases}\dfrac{\tau}{\tau_m} & 0\le\tau\le\tau_m\\[2ex] \dfrac{\tau_p-\tau}{\tau_p-\tau_m} & \tau_m<\tau\le\tau_p\\[2ex] 0 & \tau>\tau_p\end{cases} \qquad \textbf{(3-10)}$$

3.4.1 Eigenfunction Expansion

With the equation nondimensionalized, the Eigenfunction expansion method can be performed to obtain the nondimensional temperature profiles, θ. The expansion of θ can be done according to Eq. (3-11).

$$\theta(\upsilon,\zeta,\tau)=\sum_{n=0}^{\infty}\sum_{m=0}^{\infty}b_{nm}(\tau)R_n(\upsilon)Z_m(\zeta) \qquad \textbf{(3-11)}$$

The functions $R_n(\upsilon)$ and $Z_m(\zeta)$ are obtained by the solution of the associated Sturm-Louisville problems. $R_n(\upsilon)$ is found to be $J_0(\lambda_n \upsilon)$ and $Z_m(\zeta)$ is found to be $\cos(\beta_m \zeta)$. J_i is the Bessel Function of the first kind, i is the order of the Bessel Function. The expanded solution of θ is shown in Eq. (3-12).

$$
\begin{aligned}
&\theta(\rho,\xi,\tau)=\sum_{q=1}^{\infty}\sum_{p=0}^{\infty} b_{qp}(\tau)\cos\left(\beta_q \xi\right)J_0\left(\lambda_p \rho\right)\\
&\beta_q \tan\left(\beta_q\right)=Bi\\
&J_1\left(\lambda_p\right)=0\\
&b_{qp}(\tau)=F_p G_q \mathrm{T}_{qp}(\tau)\\
&F_p=\frac{2}{J_0^{\,2}\left(\lambda_p\right)}\int_0^1 \rho e^{-\rho^2 / \rho_o^{\,2}} J_0\left(\lambda_p \rho\right) d\rho\\
&G_q=\frac{2}{1+\left(S / \beta_q\right)^2}\,\frac{(Bi+S)\cos\left(\beta_q\right)-Se^{-S}}{Bi(Bi+1)+\beta_q^{\,2}}\\
&\mathrm{T}_{qp}(\tau)=\begin{cases}
\dfrac{e^{-\gamma_{qp}\tau}}{\tau_m \gamma_{qp}^{\,2}}\left(1-e^{\gamma_{qp}\tau}+\tau\gamma_{qp}e^{\gamma_{qp}\tau}\right) & 0\le\tau\le\tau_m\\[2ex]
\dfrac{e^{-\gamma_{qp}\tau}}{\tau_m \gamma_{qp}^{\,2}\left(\tau_m-\tau_p\right)}\begin{pmatrix}\tau_m-\tau_p+\tau_p e^{\gamma_{qp}\tau_m}-\tau_p\tau_m\gamma_{qp}e^{\gamma_{qp}\tau}\\ -\tau_m e^{\gamma_{qp}\tau}+\tau_m\tau\gamma_{qp}e^{\gamma_{qp}\tau}\end{pmatrix} & \tau_m<\tau\le\tau_p\\[2ex]
\dfrac{e^{-\gamma_{qp}\tau}}{\tau_m \gamma_{qp}^{\,2}\left(\tau_p-\tau_m\right)}\left(\tau_p-\tau_m-\tau_p e^{\gamma_{qp}\tau_m}+\tau_m e^{\gamma_{qp}\tau_p}\right) & \tau>\tau_p
\end{cases}\\
&\gamma_{qp}=\lambda_p^{\,2}+a^2\beta_q^{\,2}\\
&Bi=\frac{hL}{k}=\frac{hL}{\rho_m c\alpha}\\
&S=\kappa L,\ \rho={r}/{L},\ \xi={z}/{L},\ \tau={\alpha t}/{R^2},\ \rho_o={r_o}/{R},\ a={R}/{L}\\
&\theta=\left(T-T_\infty\right)\frac{\pi r_o^{\,2}k}{P\kappa R^2}
\end{aligned}
\qquad \textbf{(3-12)}
$$

The values of β_m are the roots of the equation $Bi = \beta_m \tan(\beta_m)$. The values of λ_n are the roots of the equation $J_1(\lambda_n) = 0$. Whereas there is no zero root of the β_m equation, the infinite summation starts at one. There is, however, a zero root of the λ_n equation, and thus the infinite sum is started at $n = 0$. The coefficient b is modified for the zero value of n, and thus it is separated from the double infinite sum. The rest of the development of the Eigenfunction expansion is mathematically complex and the entire derivation is covered in Appendix B. Eq. (3-12) is returned from the derivation in Appendix B.

3.4.2 Infinite Series Truncation

With increased amount of terms, the computational time increases geometrically as each point is calculated from the truncated series. With the truncation at N_T terms on both indices, Eq. (3-12) becomes Eq. (3-13). The other constants and functions previously defined in Eq. (3-12) stay the same and are not repeated for simplicity.

$$\theta(\upsilon,\zeta,\tau)=\sum_{m=1}^{N_T} b_{0m}(\tau)\cos(\beta_m\zeta)+\sum_{n=1}^{N_T}\sum_{m=1}^{N_T} b_{nm}(\tau)\cos(\beta_m\zeta)J_0(\lambda_n\upsilon) \tag{3-13}$$

This equation is a mathematical solution to the original heat equation and boundary conditions. The accuracy of this equation in comparison to the analytical model can only be shown by a large number of terms in the truncated series.

To determine that there were sufficient values being used; a study was performed using 5, 10, 15, 30 and 50 terms. There was no change in any of the profiles when using 10 terms or more, and thus the 10 term solution was determined to be used as the base. Having decided on 10 terms for the study, the first 10 eigenvalues were needed. Table 3-1 is an example of what one of the eigenvalue sets contains when the Biot number is 8.081 E -4.

Table 3-1: First 10 Eigenvalues Including 0 for *Bi=8.081 E -4*

Eigenvalue	λ_n	B_m
1	0	0.0284
2	3.8317	3.1418
3	7.0156	6.2833
4	10.1735	9.4249
5	13.3237	12.5664
6	16.4706	15.7080
7	19.6159	18.8496
8	22.7601	21.9912
9	25.9037	25.1328
10	29.0468	28.2744

3.4.3 DSFA Model Validation

The DSFA validation was completed by comparing the results from the model from Eq. (3-13) to the ASTM method for PLD and comparing the results of diffusivity to the predicted results of the diffusivity. The DSFA model that has been developed for this analysis uses a spatially varying Gaussian laser intensity profile with a triangle pulse. For model validation, the radius was made to be much larger than the sample. The distance to one sigma from center was twice that of the sample size. This resulted in the laser power that was incident on the top surface of the sample to be very nearly constant. For the large laser radius, the measurement location on the back of the sample does not affect the results. This is due to the fact that the laser pulse is approximately uniform, having no radial dependence. The laser power time dependence was also reduce to be more close to infinitesimally short. The convective heat coefficient was set to be 1 as values of zero would alter the equations and to protect against divide by zero. The value for the absorption coefficient was set to be much higher, simulation an even larger optical depth. This simulates more closely the behavior of all the energy absorbed in a very thin layer on top of the sample. The values returned by the DSFA model were all similar and the averages of the values have small error, approximately 5%, between these values in Table 2-1 and any of the cases analyzed. As the ASTM standard is accurate within 5% [ASTM], this is an acceptable range. As is shown in Figure 3-8, the non-dimensional temperature profiles of the DSFA and the Parker are very similar.

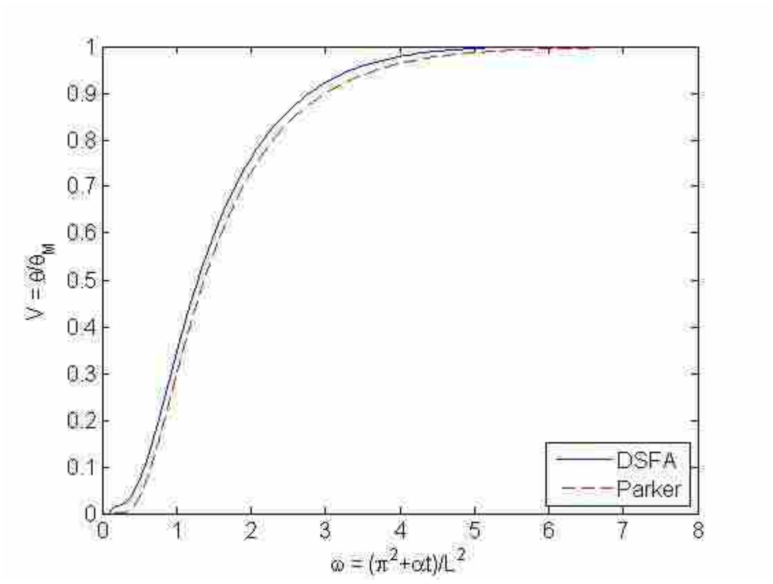


Figure 3-8: Comparison of Parker and DSFA

The differences in diffusivities between the tested cases are up to orders of magnitude different and this verifies that the values in Table 2-1 themselves are not dependent on the material. As the Eq. (2-8) and Eq. (2-9) include the thickness term, the values given are also independent of thickness of the sample.

3.5 **Genetic Algorithm Development and Implementation**

Due to the complexity of the DFSA model, it cannot be inverted such that inputting a temperature profile will return a value for α. The DSFA model can only solve the forward problem when diffusivity is known. Thus, the DSFA model developed in this work needs a search technique to be able to determine α of the tested material. To solve the inverse problem, a search technique needs to run the forward problem many times to compare the experimental profile to the analytical profiles calculated by the DSFA model. A GA can be used as the search technique for the method.

The solutions to inverse problems are often found by repeated tests of parameter cases until the desired solution is returned. This method of iterative process solution is computationally intensive. It must perform many simulations in an organized search so that the desired result may be found. The process returns the value for the time-dependent temperature, $T_i(0,0,t)$ for each of the parameter sets generated. It then compares the returned temperature profile to the measured temperature profile and calculates a fitness value. After all parameter sets have been created for the generation, convergence is tested using the fitness values. Convergence of a GA is when all the parameter sets in a population consist of similar individuals [Davis]. When convergence is achieved, the process is stopped and the parameter set of α, κ, ρc, and h is returned as the solution to the inverse problem. If convergence is not reached in the current generation, the parameter sets in the population are modified using tournament selection, crossover, and mutation. This process is done for each of the population members by the GA. New generations are created until the GA has converged.

Simple GAs are widely used in many practical problems. A GA has its foundations based on the concept of diversity, inheritance, and fitness pressure. The GA randomly selects the values from within the parameter bounds to make up the first group of parameter sets for the generation. It then calculates the fitness values of the members of the population and the generational progression can begin. Diversity is introduced into the procedure by the modification of the parameters in the sets. Inheritance is the concept of having the changes made from one generation to the next where new parameter sets gain the positive traits of the previous generation. The fitness pressure is what allows the algorithm to achieve a desirable solution. Fitness values are calculated by the program and compared one with another to determine the best values.

The GA for the DSFA model was run with the fitness function that uses an average error of the temperature values along temperature profile of length M, the absolute error in the maximum temperature, and a penalty function. The penalty function utilizes the larger of $t_{maxrise,input}/t_{maxrise,DSFA}$ and $t_{maxrise,DSFA}/t_{maxrise,input}$. The maximum of the two values is used as it gives the largest penalty for variations in the curve. This penalty function only is used when the values for $t_{maxrise,input}$ and $t_{maxrise,DSFA}$ are not the same, when they are the same, the penalty function returns zero. The total fitness function with the penalty function is shown in Eq. (3-14).

$$Fitness = -\left(\frac{100 * \sum_{i=1}^{M} \left| \frac{T_{i,input} - T_{i,DSFA}}{T_{i,input}} \right|}{M} + \frac{10 * \left| T_{max,input} - T_{max,DSFA} \right|}{T_{max,input} - T_{\infty}} + Penalty \right) \cdot 100\%$$

$$Penalty = \begin{cases} 0 & t_{maxrise,input} = t_{maxrise,DSFA} \\ 10 * \left(maximum\left(\frac{t_{maxrise,input}}{t_{maxrise,DSFA}}, \frac{t_{maxrise,DSFA}}{t_{maxrise,input}} \right) - 1 \right) & t_{maxrise,input} \neq t_{maxrise,DSFA} \end{cases}$$

(3-14)

The absolute values in the fitness function allow the GA to find the difference and always ensure that the value is a positive number. The negative sign at the start flips the value which allows the fitness pressure to be maximized. The maximization procedure forces small error to bring the fitness value closer and closer to zero. This is done so that the fitness value is never being able to cross over zero and then increase the error.

For this case, there were 150 parameter sets generated randomly. The random distribution of the fitness values of the first population of a sample test is shown in Figure 3-9.

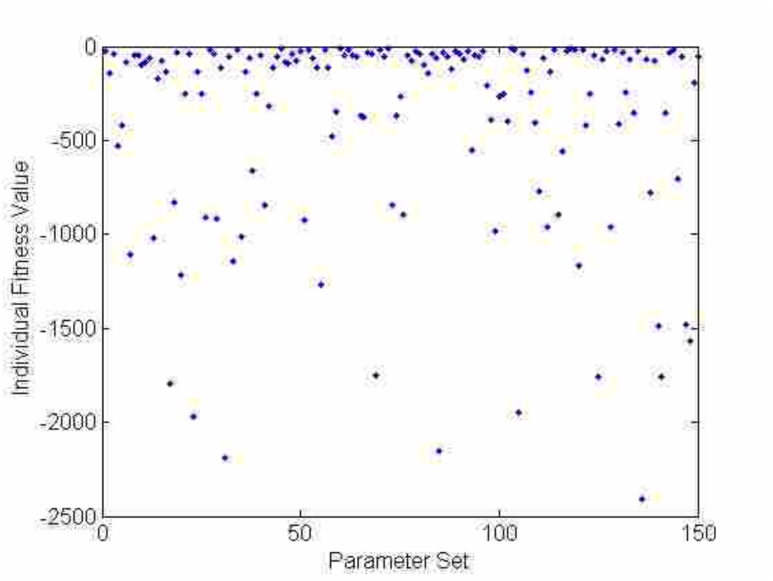


Figure 3-9: Distribution of Fitness Values at First Generation before the start of the GA

At the start of each generation, the parameter sets compete in pairs against one another to determine the better set. This competition is done using tournament selection. Inheritance of characteristics in the GA is due to the use of this selection technique. The fitness value of one parameter set is compared with the value of the other. The set with the higher value is then selected. This is done to populate the next generation with the parameter sets which have the highest fitness values. Each of the parameter sets in the population compete with the others and are combined to yield the highest possible fitness values for that population.

After the tournament selection, the GA performs crossover and mutation on the parameter sets. These processes allow for the introduction of diversity that leads to the improvements in fitness values [Goldberg].

For real valued and continuous GAs, blend crossover can assist in the obtaining a solution in an efficient manner. The crossover allows for changes in the parameter sets that may create better fitness values. Crossover pulls a random number and if it is above a certain predetermined threshold, then the crossover function is used. The threshold for the testing done with this GA was set to be 50%. This number was found to achieve convergence in relatively few generations. Blend crossover uses the value of two paired parameter sets and mixes them based on a blend fraction *r* that was also pulled as a random number.

The GA would select two parameter sets and pair them up. It would then take the first parameter of their sets and if the pulled random number was higher than the threshold it would mix them as shown in Eq. (3-15).

$$\begin{aligned} y_1 &= (r)\cdot parameter_1 + (1-r)\cdot parameter_2 \\ y_2 &= (1-r)\cdot parameter_1 + (r)\cdot parameter_2 \end{aligned} \qquad \textbf{(3-15)}$$

The values from the crossover y_1 and y_2 then replace the values for $parameter_1$ and $parameter_2$ respectively in their own parameter sets. If the random number was above the threshold, it would perform the crossover and continue to the next parameter. This was done until all four of the parameters were tested for the pair.

Mutation is a process where one of the parameters in the set is changed to a new, random value in the bounded set. This is also done in an effort to increase the diversity of the population. Opportunity for mutation is determined by a mutation parameter. The mutation parameter in this GA is a changing value that allows for high probability early on when the fitness values are not very good and as the generations pass, and the fitness values improve, the likelihood of mutation decreases. This type of mutation is called dynamic mutation. The mutation for this GA utilized a dynamic mutation rate where the likelihood of mutation would decrease according to Eq. (3-16).

$$Mutation = 0.75\left(1 - \frac{i_{Gen} - 1}{N_{Gen}}\right)^{4} \tag{3-16}$$

The constant in front of the exponential expression gives the starting probability of mutation and was also selected through testing to be an appropriate value for achieving convergence in a reasonable amount of runs. As is seen in Figure 3-10 this is a parabolic decrease in the percent chance for mutation starting at the given constant of 75% down to near zero chance when the generation number, i_{Gen}, is equal to a maximum number of generations plus one that the GA will run, N_{Gen}.

As the generation number increases, the stronger parameter sets begin to dominate the population and the mutation probability decreases. Mutation is less desirable as the fitness values are driven to zero. This is due to the parameter sets beginning to resemble one another and large changes are less likely to improve the fitness values of the population.

After the tournament selection, crossover, and mutation, the fitness values are found for each new parameter set. The values are compared and the best value is stored as the first value of the next generation. The mandatory continuation of a certain parameter set is called elitism and allows for the current best value to always continue [Davis]. This allows for sets with the best fitness values to positively affect the other sets in the population. This procedure continues for each generational step for as many steps as is set by the designer of the GA and convergence is reached.

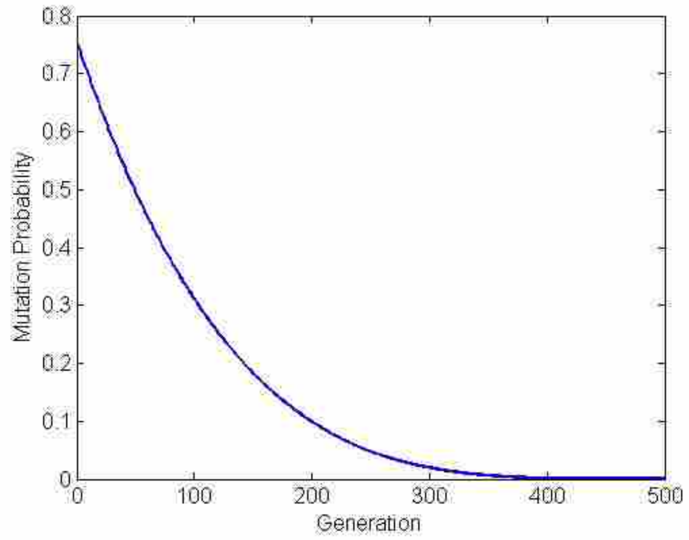


Figure 3-10: Change of Mutation Probability by Generation

The use of GA is broad and the values and procedure used for crossover, mutation, and elitism can be changed for each desired application of the GA [Goldberg], [Davis]. The increased diversity in the population from crossover and mutation can allow for the stronger fitness values to begin dominating the populations. When the mutation and crossover procedures create parameter sets with poor fitness values, the poor fitness value sets are removed from future populations by the tournament selection.

The generational progression of the best fitness value of the population is seen in Figure 3-11 and Figure 3-12. These figures show that there is an early jump in the fitness values, but there is still much progress made throughout the progression of the generations. As is seen in these figures, there are jumps where the GA finds a better solution than it previously had known.

The jumps are due to the new best case fitness value as found by the GA. At these times, the GA has come to an all-time better solution to what it previously had. That case is then propagated forward allowing all cases to improve and possibly find yet a better solution.

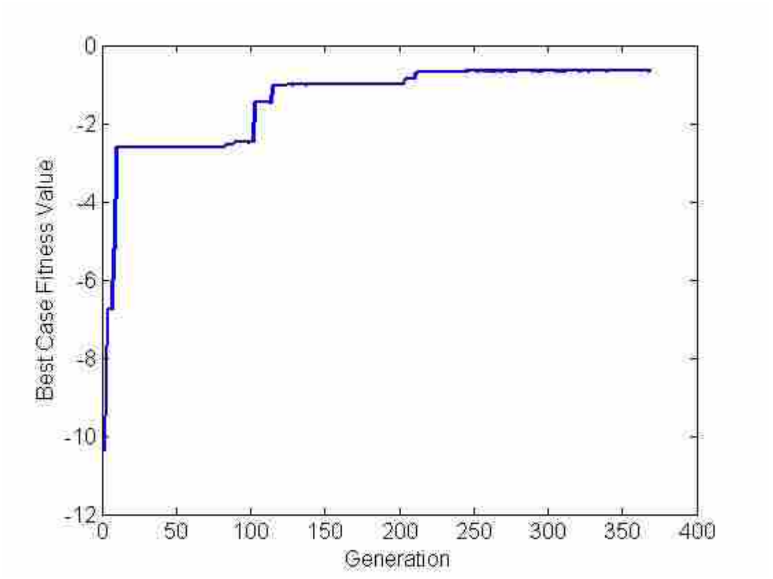


Figure 3-11: Progression of Best Case Fitness Value of the Population in the Generational Progression of the GA

The distribution of the fitness values for the 150 parameter sets in the final generation of the sample GA is shown in Figure 3-13. The outliers are due to mutations as all the other parameter sets are similar and thus blending would not change their values by much.

It is seen in Figure 3-13 that although the values look the same, they are slightly different. It is seen in the scaling of the figure that they are very close, but the parameter sets returned are slightly different from one another.

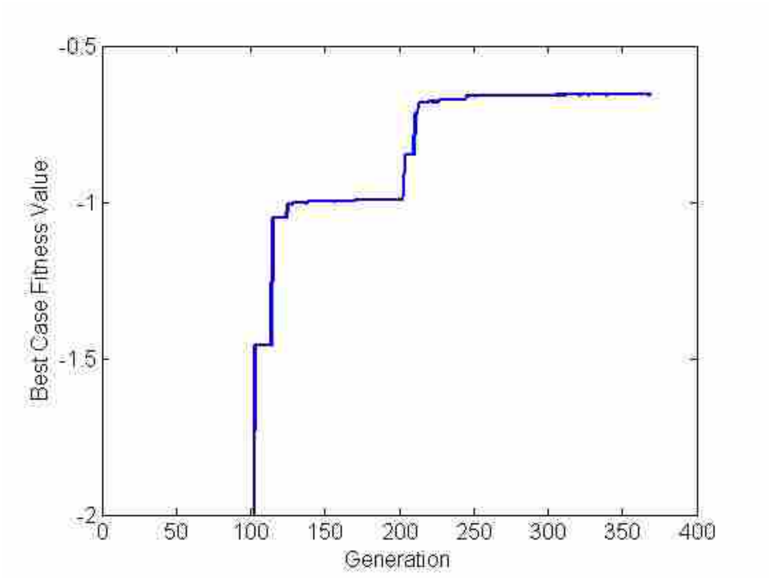


Figure 3-12: Zoomed in Progression of Best Case Fitness Value of the Population in the Generational Progression of the GA

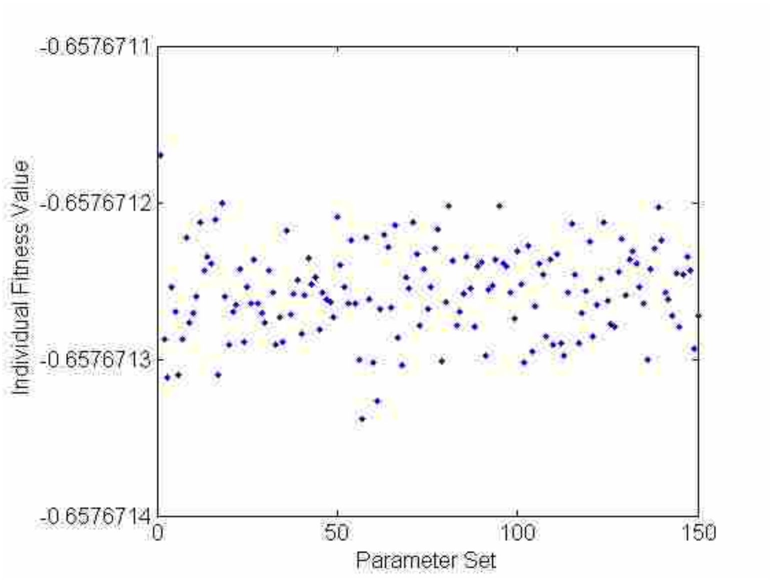


Figure 3-13: Distribution of Fitness Values after Completion, 369 Generations of the GA

Once parameters begin to become very similar, mutation is the only way to introduce diversity in the system to attempt to find a better solution. As there have been already 334 generations of possible 500 to get to this result, the likelihood of a much better solution due to random chance is low and so the mutation probability is also low.

Convergence determines when the GA will stop and return its solution. Convergence is defined as when all the parameter sets have become similar to each other [Davis]. Convergence is determined by calculating the standard deviation of the fitness values for each parameter set in the current population, and comparing it against a convergence parameter, ε. If the standard deviation of the fitness values in the population is below the value set for the convergence parameter, then the GA stops. The convergence criteria should be small, but setting is too low will cause the GA to run for more generations than is necessary. An appropriate value of ε needs to be determined for the case to ensure that the desired solution is to be met. This requires the testing of different values and determining the best value for the computational effort. For this testing that was performed, ε was set at 0.05. The set shown in Figure 3-13 terminated at 369 generations because the convergence criteria had been met.

Figure 3-14 shows the generational progression of the convergence criteria of the case as shown in the above figures. In Figure 3-14, it is seen that the initial values of the standard deviation are very high and sporadic. The values generally decrease as the parameters begin to become more similar and closer in range due to the efforts of the tournament selections removing the parameter sets with poor fitness values, crossover to blend the values to find potentially better values, and decreasing the mutational diversity with increasing generation number.

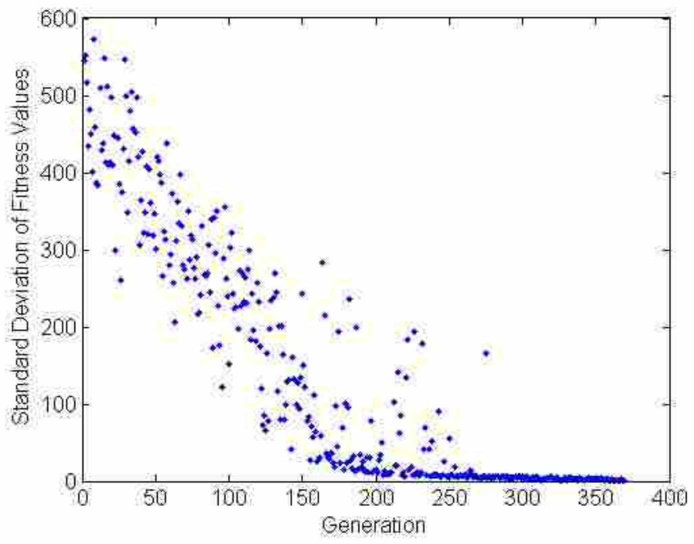
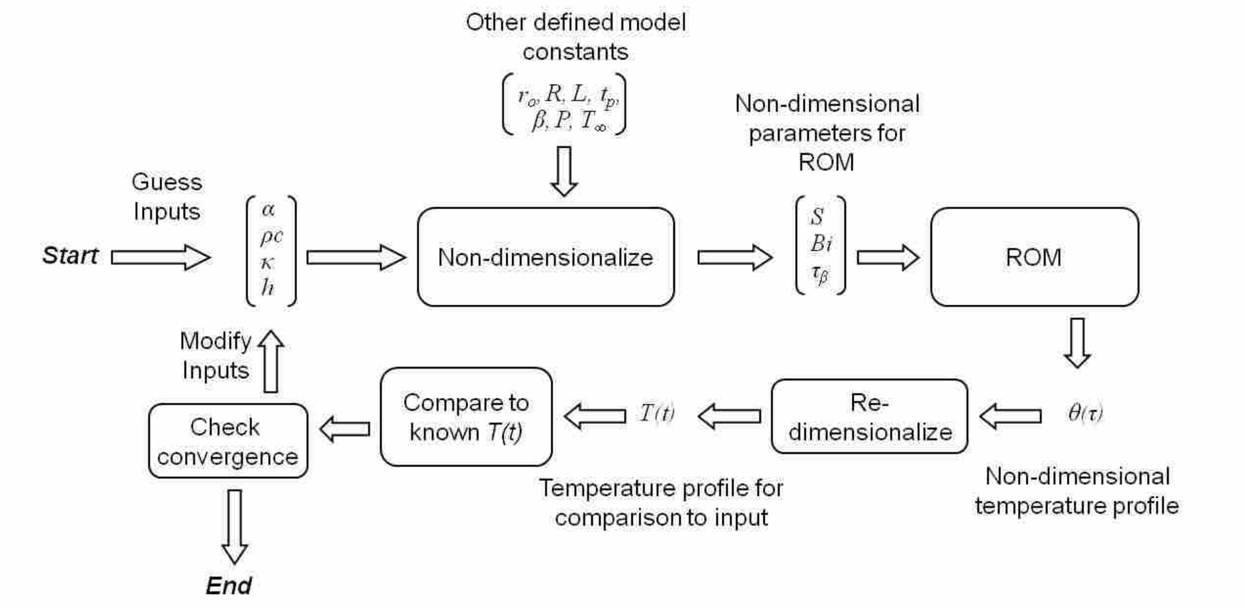


Figure 3-14: Generational Progression of Standard Deviation of Fitness Values

The flow of the iterative process is shown in Figure 3-15. In this case, it generated 150 parameter sets and continues processing on theses sets for up to 500 generations. For this GA to process there are up to 75,100 function calls to find the time dependant temperature profiles by the forward problem.



Other defined model constants
$r_o, R, L, t_p, \beta, P, T_\infty$
Non-dimensional parameters for ROM
Guess Inputs
Start
α
ρc
κ
h
Non-dimensionalize
S
Bi
τ_β
ROM
Modify Inputs
Check convergence
Compare to known $T(t)$
$T(t)$
Re-dimensionalize
$\theta(\tau)$
Temperature profile for comparison to input
Non-dimensional temperature profile
End

Figure 3-15: Flowchart for the GA Process

With all the efforts to speed the solution of the DSFA model, it still takes approximately 0.85 seconds to return the temperature profile. For the GA to make 75,100 function calls of the DSFA, it would require the GA program to run for more than 17 hours.

It is desired to have a method that can allow for faster solutions of the model to speed the iterative process. ROM is a technique that can evaluate models very quickly. Using a ROM instead of the DSFA model will speed the evaluation of the temperature profiles. This will allow the solution of the inverse problem to be found approximately 240 times faster.

3.6 **Reduced Order Model Development**

ROM is a technique that has been developed to allow simulations of engineering models to run in a more computationally efficient manner. ROM is based on the theory of proper orthogonal decomposition (POD). POD was developed in the early twentieth century for the manipulation of statistical data [Pearson]. Its use has been utilized by the ROM in allowing matrix manipulations to be performed on model data rather than requiring the direct solution [Ostrowski], [Rambo], [Bergman].

ROM uses interpolation functions on known solution sets from accurate engineering models to simulate the models with high levels of accuracy. The ROM performs matrix manipulations on the data sets to execute its simulations of the model rather than the complex equations from the thermal engineering model. The use of matrix manipulation allows the ROM simulations to be run at a high rate of speed.

ROM is based on Eq. (3-17) where **A** is an $N \times M$ matrix that contains the solution values based on the orthogonal basis set of governing parameters [Larson]. The matrix Φ is an $N \times m$

matrix that is formed from the basis set of solutions. **B** is an $m \times M$ matrix that contains the expansion coefficients for each set of governing parameters.

$$\mathbf{A} = \Phi\mathbf{B} \tag{3-17}$$

In solving the model for each set of governing parameters, the matrix **A** is built by each column being the values of the solution for each set of governing parameters. After all desired cases have been run, a singular value decomposition (SVD) is performed on the matrix **A**. Using the SVD, **A** is factored into an $N \times N$ orthogonal matrix **U**, an $N \times M$ diagonal matrix **Σ** and an $M \times M$ orthogonal matrix **V** as is shown in Eq. (3-18) [Strang].

$$\mathbf{A} = \mathbf{U}_A \Sigma_A \mathbf{V}_A^{\mathsf{T}} \tag{3-18}$$

The elements in $\mathbf{\Sigma}_A$ are called the singular values. They are sorted by the SVD algorithm from highest to lowest. The number of nonzero singular values of a matrix is the same as the rank of that matrix. The rank of the matrix **A** is given the symbol r_A. The non-zero singular values are represented by the symbols σ_{Ai}, $i=1,\ldots,r_A$ [Strang].

The rank defines the amount of columns in $\mathbf{U}_A$ that form an orthonormal basis for the column space of **A**. It is not necessary to use the entire basis set in order to represent the data, and so the matrix $\mathbf{U}_A$ is truncated at m to reduce the amount of computational effort in solving the matrix manipulations. This truncation gives a close approximation of the data from $\mathbf{U}_A$ required to produce the set of basis vectors $\mathbf{\Phi}$ required by Eq. (3-17). The cutoff, m, is dependent on the system and will vary from model to model. The decision of where to place the cutoff is to be made by the model designer. The value can be determined by looking at the singular values of **A** and choosing an appropriate value. The value can also be determined through trial and error in finding a solution that closely fits the desired or known data and is still computationally efficient. An example of a plot of the singular values of an A matrix is shown in Figure 3-16.

Evaluating **B** is easy by simply rearranging Eq. (3-17) and using the orthogonality characteristics of Φ yields Eq. (3-19).

$$\mathbf{B} = \Phi^{T}\mathbf{A} \qquad \textbf{(3-19)}$$

Estimating the expansion coefficients is performed by interpolation of the results given in Eq. (3-19). To do this, a coefficient matrix **C** is defined as shown in Eq. (3-20).

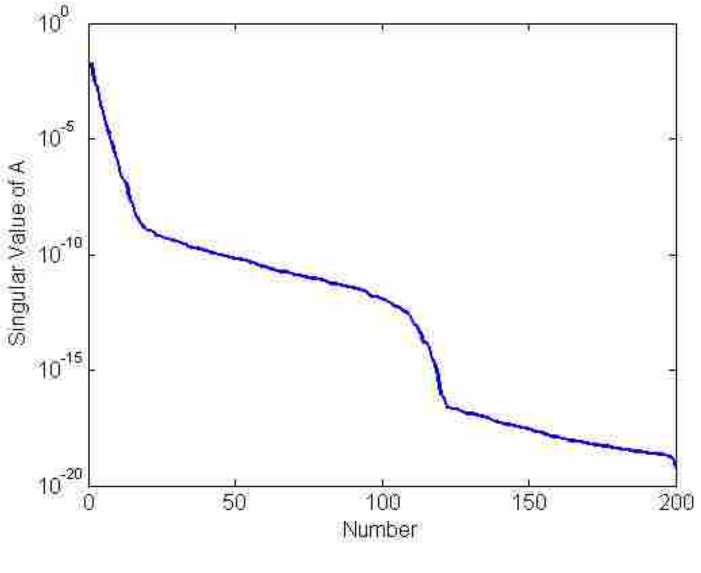


Figure 3-16: Singular Values of A

$$\mathbf{B} \equiv \mathbf{CF} \qquad \textbf{(3-20)}$$

F is an $M \times M$ matrix of the interpolating functions. Interpolation functions can vary from model to model and are to be used in such a way that fits best to the data and provides the best results. The designer of the ROM must determine the best function for their case. In the literature, it has been shown that inverse multiquadratic functions are useful in interpolating data in multi-dimensions [Hardy 1971], [Hardy 1990]. **F** is defined such that the i^{th} column is calculated by using the determined interpolation function chosen for each of the parameter sets that were used to create **A**. The modified multiquadratic function is shown in Eq. (3-21) where $\mathbf{f}_i$ represents the i^{th} column of **F**.

$$\mathbf{f}_i = \begin{bmatrix} \dfrac{1}{\sqrt{\left(\dfrac{x_1^i - x_1^1}{x_{1,\max}}\right)^2 + \left(\dfrac{\log_{10} x_2^i - \log_{10} x_2^1}{\log_{10} x_{2,\max}}\right)^2 + \left(\dfrac{\log_{10} x_3^i - \log_{10} x_3^1}{\log_{10} x_{3,\max}}\right)^2 + 1}} \\ \vdots \\ \dfrac{1}{\sqrt{\left(\dfrac{x_1^i - x_1^j}{x_{1,\max}}\right)^2 + \left(\dfrac{\log_{10} x_2^i - \log_{10} x_2^j}{\log_{10} x_{2,\max}}\right)^2 + \left(\dfrac{\log_{10} x_3^i - \log_{10} x_3^j}{\log_{10} x_{3,\max}}\right)^2 + 1}} \\ \vdots \\ \dfrac{1}{\sqrt{\left(\dfrac{x_1^i - x_1^m}{x_{1,\max}}\right)^2 + \left(\dfrac{\log_{10} x_2^i - \log_{10} x_2^m}{\log_{10} x_{2,\max}}\right)^2 + \left(\dfrac{\log_{10} x_3^i - \log_{10} x_3^m}{\log_{10} x_{3,\max}}\right)^2 + 1}} \end{bmatrix} \qquad \textbf{(3-21)}$$

The multiquadratic function had to be modified so that it could properly interpolate the logarithmically scaled parameters along with the linearly scaled parameters. The modification of the base of x_2 and x_3 into the log scale interpolates correctly with the logarithmic variation of the parameters. Each of the values x_1, x_2, x_3 correspond to the separate governing parameters that define the system **A**. The maximum values are the largest values of each of the governing parameters within the defined bounds.

F may be a singular matrix and thus cannot be inverted and multiplied to isolate and solve for **C**. The Moore-Penrose pseudoinverse is a method that can be used to isolate and solve for **C** in Eq. (3-22) [Strang], [Press]. The Moore-Penrose pseudoinverse, $\mathbf{F}^{+}$, is found based an SVD of the interpolation matrix **F** similar to Eq. (3-18) as shown in Eq. (3-22).

$$\mathbf{F} = \mathbf{U}_F \Sigma_F \mathbf{V}_F^{\mathsf{T}} \qquad \textbf{(3-22)}$$

The matrices $\mathbf{U}_F$, $\mathbf{\Sigma}_F$, $\mathbf{V}_F$ have the same meaning as described above with the **A** matrix. The pseudoinverse is post-multiplied onto both sides of Eq. (3-22) as shown in Eq. (3-23).

$$\mathbf{F}\mathbf{V}_F\mathbf{S}_F\mathbf{U}_F^{\mathsf{T}} = \mathbf{U}_F \Sigma_F \mathbf{V}_F^{\mathsf{T}} \mathbf{V}_F \mathbf{S}_F \mathbf{U}_F^{\mathsf{T}} = \mathbf{I} \qquad \textbf{(3-23)}$$

The matrix $\mathbf{S}_F$ is the diagonal matrix of the inverse of the singular values from the matrix $\mathbf{\Sigma}_F$. $\mathbf{S}_F$ is defined in Eq. (3-24).

$$\mathbf{S}_F = diag\left\{\frac{1}{\sigma_{F,i}}\right\} \qquad \textbf{(3-24)}$$

The values along the diagonal of $\mathbf{\Sigma}_F$ are defined as $\sigma_{F,i}$, $i=1,\ldots,r_F$ similar to what was done in the decomposition of the **A** matrix. To completely define the pseudoinverse, all values of $\sigma_{F,i} > 0$ must be used. However, a truncation can also be performed on the data for **F** similar to what was done on **A** to reduce the amount of data needed to represent the data accurately and efficiently. The modeler selects a value to be the minimum allowable value of $\sigma_{F,i}$. All values that fall below this value are set to zero. These singular values can also be plotted to allow the modeler to see the data in order to choose an appropriate value for the cutoff as is shown in Figure 3-17.

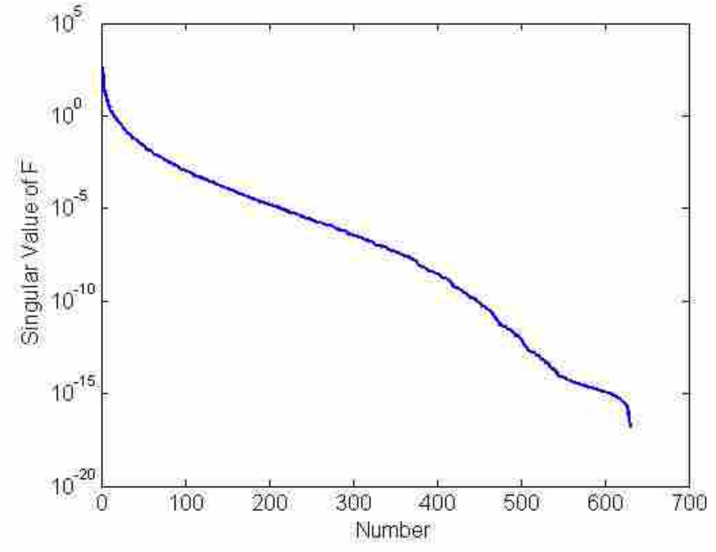


Figure 3-17: Singular Values of F

From Eq. (3-23), it is clear to see that the pseudoinverse of **F**, or $\mathbf{F}^{+}$, is given by Eq. (3-25).

$$\mathbf{F}^{+} = \mathbf{V}_{F}\mathbf{S}_{F}\mathbf{U}_{F}^{\mathsf{T}} \qquad \textbf{(3-25)}$$

With the pseudoinverse of **F** known, it can be used to isolate the coefficient matrix **C** in Eq. (3-20) by post-multiplying $\mathbf{F}^{+}$ to both sides to yield Eq. (3-26).

$$\mathbf{C} = \mathbf{B}\mathbf{F}^{+} \qquad \textbf{(3-26)}$$

The coefficient matrix is built by a bounded set of governing parameters based on the interpolation function matrix and is thus assumed to be valid for arbitrary parameters that are also bounded by the original set of parameters that built the model. The expansion coefficients for any arbitrary case **k** that lies within the bounds can also be found by multiplying the coefficient matrix with the interpolation function of those parameters as shown in Eq. (3-27).

$$\mathbf{b}(\mathbf{k}) = \mathbf{C}\mathbf{f}(\mathbf{k}) \qquad \textbf{(3-27)}$$

The vector for the interpolation function **f(k)** is found using the arbitrary parameters in the single column vector shown in Eq. (3-28).

$$\mathbf{f}(\mathbf{k}) = \begin{bmatrix} \dfrac{1}{\sqrt{\left(\dfrac{x_1^{\mathbf{k}} - x_1^1}{x_{1,\max}}\right)^2 + \left(\dfrac{\log_{10} x_2^{\mathbf{k}} - \log_{10} x_2^1}{\log_{10} x_{2,\max}}\right)^2 + \left(\dfrac{\log_{10} x_3^{\mathbf{k}} - \log_{10} x_3^1}{\log_{10} x_{3,\max}}\right)^2 + 1}} \\ \vdots \\ \dfrac{1}{\sqrt{\left(\dfrac{x_1^{\mathbf{k}} - x_1^j}{x_{1,\max}}\right)^2 + \left(\dfrac{\log_{10} x_2^{\mathbf{k}} - \log_{10} x_2^j}{\log_{10} x_{2,\max}}\right)^2 + \left(\dfrac{\log_{10} x_3^{\mathbf{k}} - \log_{10} x_3^j}{\log_{10} x_{3,\max}}\right)^2 + 1}} \\ \vdots \\ \dfrac{1}{\sqrt{\left(\dfrac{x_1^{\mathbf{k}} - x_1^m}{x_{1,\max}}\right)^2 + \left(\dfrac{\log_{10} x_2^{\mathbf{k}} - \log_{10} x_2^m}{\log_{10} x_{2,\max}}\right)^2 + \left(\dfrac{x_3^{\mathbf{k}} - \log_{10} x_3^m}{\log_{10} x_{3,\max}}\right)^2 + 1}} \end{bmatrix} \qquad \textbf{(3-28)}$$

The expansion coefficients **b(k)** returned from Eq. (3-27) for the arbitrary set of governing parameters are then used in Eq. (3-29) to return $\mathbf{Z}_\mathbf{k}$. $\mathbf{Z}_\mathbf{k}$ is the vector that contains the temperature values that are output by the ROM. In testing the ROM, the values of $\mathbf{Z}_\mathbf{k}$ for a given set of parameters that were used in the creation of the solution matrix **A** should match closely to the returned value of $\mathbf{Z_k}$ for that set.

$$\mathbf{Z_k} \approx \Phi \mathbf{b}(\mathbf{k}) \qquad \textbf{(3-29)}$$

Where the principle of ROM states from Eq. (3-29) that you can use the Φ matrix multiplied by the expansion coefficient vector **b(k)** to return a desired temperature profile, $\mathbf{Z_k}$, you cannot just invert the Φ matrix and multiply it by $\mathbf{Z_k}$ to get the coefficient vector **b(k)**. This is due to the singularity of the Φ matrix. When the inverse of Φ is multiplied by $\mathbf{Z_k}$, large errors in the calculation of the **b(k)** vector exist. Even if the vector **b(k)** was returned accurately, it still does not return the parameters, but the expansion coefficients. To get the parameters, you have to do the inverse of Eq. (3-27) to get the values for **f(k)**. As is seen in Eq. (3-28), **f(k)** is a function of the three governing parameters **k** and cannot be inverted for a unique solution that would return the parameters. There are many different sets of parameters that can create the same interpolation function values.

As the inverse solution of the ROM cannot be performed directly, an iterative search technique, such as a GA, must be performed to solve the inverse problem.

3.6.1 Defining Parameters

Once the model is defined and verified, a group of solution sets need to be created in order to run the ROM. The ROM used as governing parameters to be optical depth (*S*), Biot Number (*Bi*), and nondimensional time to peak laser power (τ_m) as the governing parameter set. These nondimensional parameters are obtained by varying dimensional parameters of absorption coefficient, κ, thermal diffusivity, α, heat capacity, ρc, and convection coefficient, h. Other non-varying system parameters are used in the determination of the nondimensional parameters. The other system parameters are r_o, R, L, t_p, t_m, P, and T_∞.

Absorption coefficients are selected to yield a broad range of temperature profiles for the laser power selected. The range was selected from a search of common semiconductor materials [Palik][Weber]. With a 0.002 m thickness the seven values for κ used and their corresponding rear face optical depths, S, are shown in Table 3-2.

Table 3-2: Parameters used for κ, S

Case	*κ (m^{-1})*	*S*
1	400	0.80
2	1000	2.00
3	1600	3.20
4	2200	4.40
5	2800	5.60
6	3400	6.80
7	4000	8.00

The convection coefficient is in the range of free convection [Incropera and DeWitt] and is taken to be the set as shown in Table 3-3.

Table 3-3: Parameters used for h

Case	*h (W/m^2K)*
1	20.0
2	30.0
3	40.0

The thermal diffusivity is dependent on material properties. The range of values is based on a general range of common materials taken from tables of properties of semiconductor materials [Dargys et al][Goldbery eta al][Yamaguchi et al]. The range of values is shown in Table 3-4.

Table 3-4: Parameters used for α

Case	*α (m^2/sec)*
1	1.00 E -6
2	3.00 E -6
3	1.00 E -5
4	3.00 E -5
5	1.00 E -4

Heat capacity was also utilized in building the parameter sets, as it was a required parameter for creating of the nondimensional parameters. The range of values for heat capacity was also taken from a range of common materials from tables of properties of semiconductor materials [Le-Ping et al][Goldbery et al][Yamaguchi et al]. The range of values for heat capacity is shown in Table 3-5.

Table 3-5: Parameters used for ρc

Case	*ρc (J/m^3K)*
1	0.50 E 6
2	1.50 E 6
3	2.50 E 6
4	3.50 E 6
5	4.50 E 6
6	5.50 E 6

The output values are the nondimensional temperature profiles at the point of the bottom surface of the sample directly below the center of the laser pulse with respect to nondimensional time, $\theta(0,0,\tau)$. The nondimensional temperature values then become a vector of length *M*, where *M* is the number of nondimensional time steps taken.

3.6.2 Creation of Solution Sets

Once the parameters are determined, the code runs the forward simulations of the high order equations. It stores the solutions in the matrix **A** that will be used in the ROM. It also stores the matrix **B** based on the input governing parameters of the model. The number of governing parameters that are used is *N*. This leaves **A** as an $M \times N$ matrix of solution sets where *M* is the number of time steps taken. A flowchart to the process of creating the basis set of values for $\theta(0,0,\tau)$ is shown in Figure 3-18.

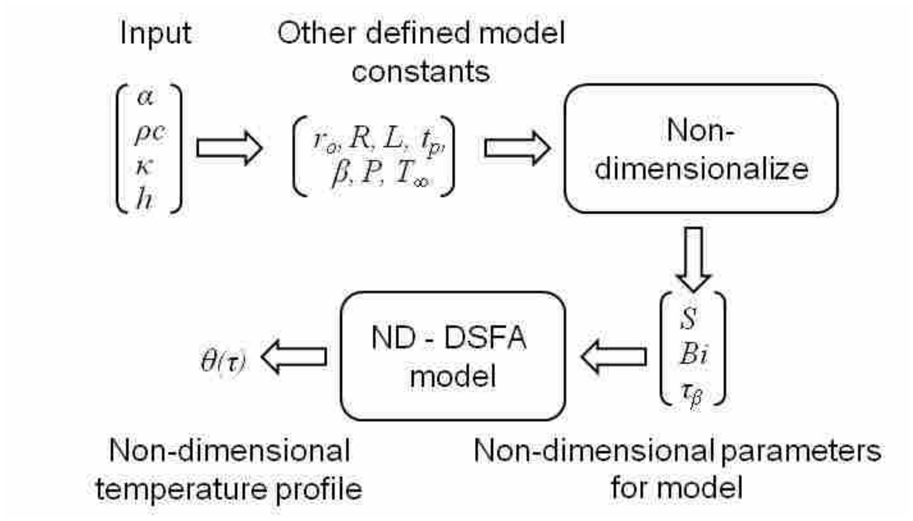

Input
α, ρc, κ, h
Other defined model constants
$r_o, R, L, t_p, \beta, P, T_\infty$
Non-dimensionalize
S, Bi, τ_β
ND - DSFA model
$\theta(\tau)$
Non-dimensional temperature profile
Non-dimensional parameters for model

Figure 3-18: Flowchart for Creation of Basis Set of Solutions for ROM

3.7 ROM Model Verification

Once the ROM is prepared and the required matrices are created, any bounded parameters can be input into the model. The ROM performs the matrix manipulations on the input parameter and returns the desired values. This process is very fast, due to the fact that all that is done is the calculation of the interpolation vector, a matrix manipulation of the coefficient matrix **C** to the vector, and multiplication of that product to the previously calculated Φ matrix.

The ROM for the DSFA model was modified so that it would match cases that would be input into the model. The number of significant eigenvalues that the ROM would use was set to be 25. This allowed for the fastest solution of the model while still maintaining acceptable accuracy of the profile. The singularity tolerance of the **F** matrix was set to be 1.0 E -13.

These parameters allowed the ROM to simulate the solution of the profile against a case known from the creation of the basis set of solutions – where the parameter set was α = 3.0 E -5, ρc = 3.5 E 6, h = 30, and κ = 2800. The ROM simulated this case with an average error of less than 9.439 E -7% with a maximum error of 3.762 E -6%. The error of the temperature value as reported at each step, $Error_i$, is calculated in Eq. (3-30).

$$Error_i = \left| \frac{T_{DSFA,i} - T_{ROM,i}}{T_{DSFA,i}} \right| \cdot 100\% \qquad \textbf{(3-30)}$$

As is seen in Figure 3-19, the ROM simulation matches well along the whole length of the profile. In this known case, it shows that the interpolation functions are able to recreate a known case with very high accuracy.

The case that is run and shown in Figure 3-19 is a case that is solved for by the coefficient builder program and thus the solution is known. The case in Figure 3-20 is a case with random parameters that are away from the known parameters. It is shown in Figure 3-20 that ROM still performs adequately when the parameter set is not from known parameters. This case has the parameter set of h = 22.28, κ = 1258, α = 1.432 E -5, and ρc = 2.986 E 6. The average error between the DSFA model solution and the ROM output solution is 0.225% with a maximum error of 0.821%.

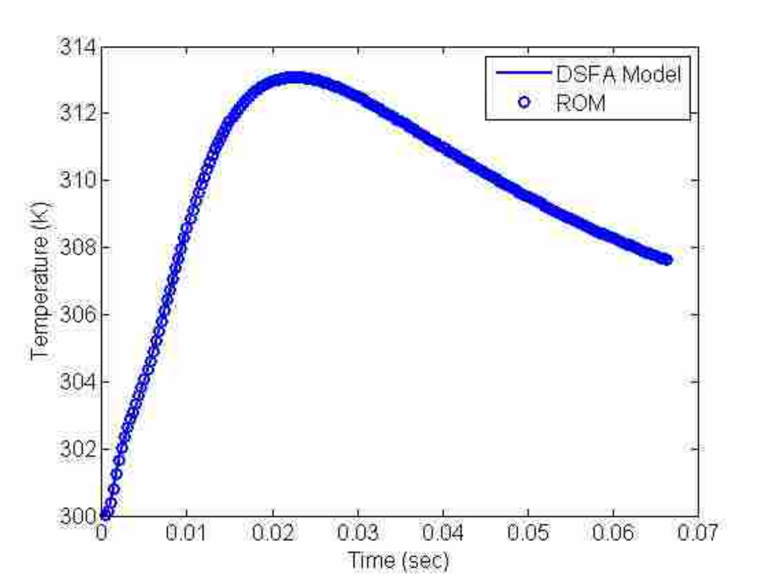


Figure 3-19: Example Case of Comparison of DSFA and ROM for Known Case

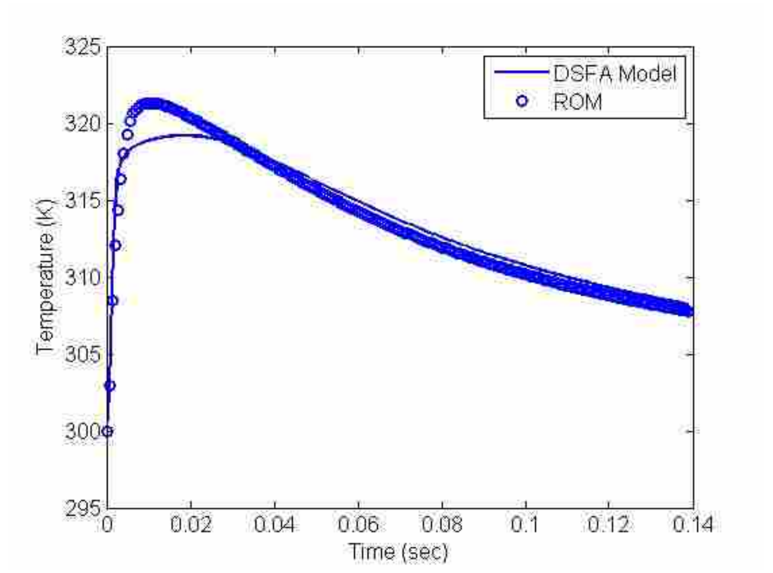


Figure 3-20: Example Case of Comparison of DSFA and ROM for Arbitrary Case 1

The largest errors in the cases exist in the early on areas of the model near the peak of the temperature. As is seen in Figure 3-21, the largest percentage in the error is 0.401%. This case from Figure 3-21 has the parameter set of h = 35.84, κ = 2168, α = 7.568 E -6, and ρc = 3.785 E 6. The average error between the DSFA model solution and the ROM output solution is 0.225% with a maximum error of 0.821%.

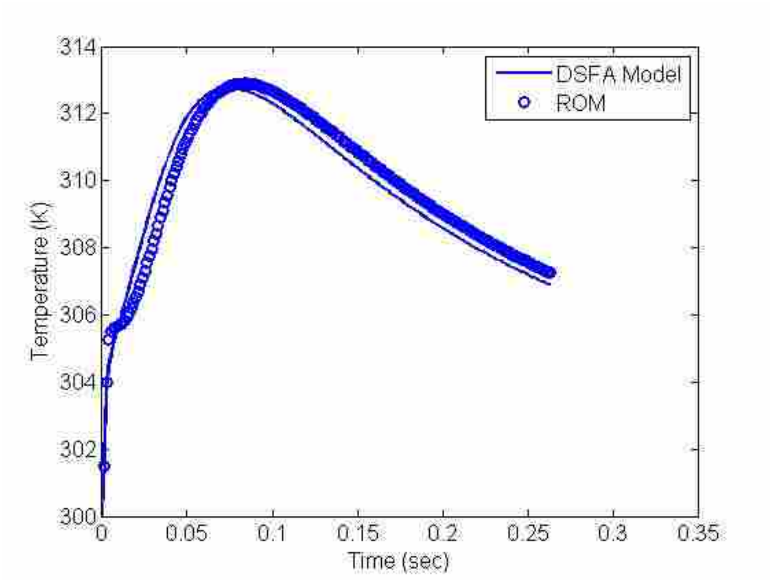


Figure 3-21: Example Case of Comparison of DSFA and ROM for Arbitrary Case 2

This shows that the ROM is able to simulate with moderately good accuracy the temperature profile for a previously unknown case. This shows that the closer the unknown parameter set is to a known solution, the better the interpolation will return an accurate temperature profile. Even when the profile is far from a known case, the solution is still fairly accurate throughout the majority of the profile. When looking at the dimensional parameters, the values are quite strong with the average errors and the maximum errors being much smaller.

The time that is required by the ROM to run the simulations is an average of 0.0026 seconds to return each profile. The complete DSFA model solution takes an average of 0.85 seconds to complete each profile. This means that the ROM is able to run approximately 340 times faster than the DSFA solution. As the average error is still small in the ROM cases, the ROM can be used to speed the GA solution.

3.8 **Example Case Testing**

To test the capability of the GA process with using ROM, sample cases were made from actual material properties and run in a FLUENT simulation using the same assumptions as the DSFA model. These profiles gave an independent profile for which to test the GA with ROM against. There were a number of cases developed and four of the test cases were used to refine the GA and ensure that it would be able to return accurate data.

3.8.1 Gallium Arsenide

The data for Gallium Arsenide was input the GA and run for five tests to ensure consistent convergence. The input profile was also used with the ASTM method to determine what it evaluates as the diffusivity. The input values for the Gallium Arsenide case are shown in Table 3-6 [Dargys et al][Carlson et al][Sharmin et al].

Table 3-6: Gallium Arsenide Input and Outputs

Parameter	*True Value*	*GA/ROM*	*ASTM*
α	3.133 E -5	3.097 E -5	6.224 E -5
κ	3400	3142	-
ρc	1.756 E 6	1.774 E 6	-
h	37.2	33.9	-

The comparisons of the temperatures from the input FLUENT profile, the parameter output from the GA/ROM as input into the DSFA, and the DSFA when using the true, known values directly as input into the DSFA are shown in Figure 3-22 for the bottom surface and Figure 3-23 for the top surface, both at the centerline of the sample. The figures show the high level of accuracy returned by both the GA/ROM solution and the accuracy of the DSFA with matching a profile with known parameters. In the figures, the correlation is seen by all three curves laying almost entirely on each other.

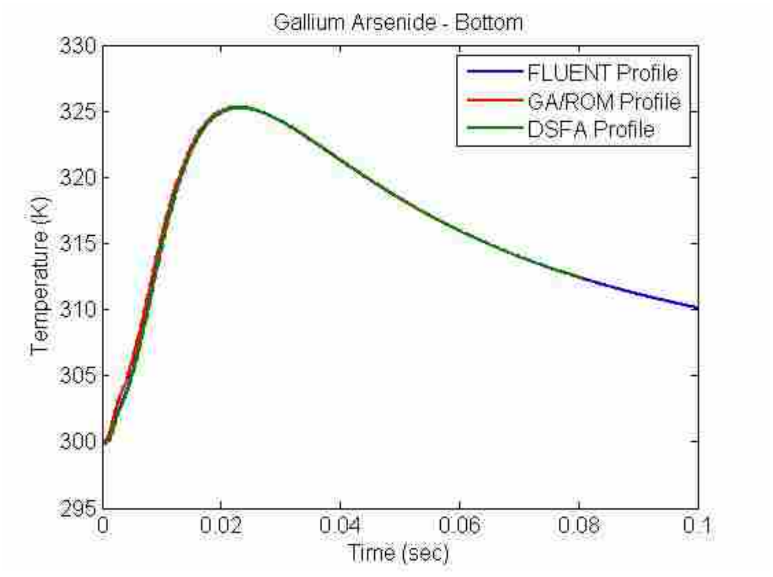


Figure 3-22: Gallium Arsenide Bottom Temperature Profile Comparisons

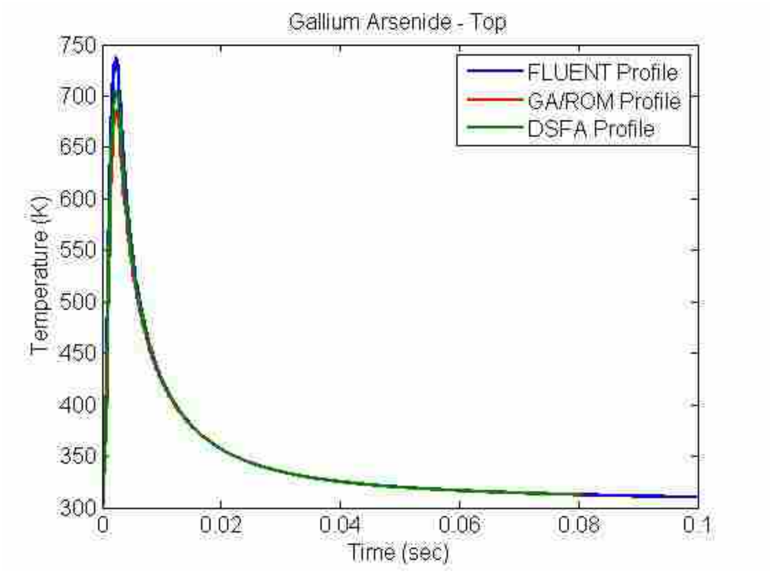


Figure 3-23: Gallium Arsenide Top Temperature Profile Comparisons

The errors for the parameters from GA/ROM are consistently good for this case. The error in *α* is 1.14%, *κ* is 7.59%, *ρc* is 1.05% and *h* is 2.54%. The returned error from the ASTM calculation is 98.67%.

3.8.2 Silicon

As with the data for the Gallium Arsenide case, the data from Silicon was input the GA and run for five tests to ensure consistent convergence. The input profile was also used with the ASTM method to determine what it evaluates as the diffusivity. The input values for the Silicon case are shown in Table 3-7 [Palik][Okhotin et al].

Table 3-7: Silicon Input and Outputs

Parameter	*True Value*	*GA/ROM*	*ASTM*
α	9.139 E -5	8.784 E -5	2.702 E -4
κ	2000	1734	-
ρc	1.630 E 6	1.715 E 6	-
h	37.1	27.41	-

The comparisons of the temperatures from the input FLUENT profile, the parameter output from the GA/ROM as input into the DSFA, and the DSFA when using the true, known values directly as input into the DSFA are shown in Figure 3-24 for the bottom surface and Figure 3-25 for the top surface. The figures show the high level of accuracy returned by both the GA/ROM solution and the accuracy of the DSFA with matching a profile with known parameters. In the figures, the correlation is seen by all three curves laying almost entirely on each other. The GA/ROM profile is just slightly off on both surfaces from the FLUENT and DSFA profiles which lay almost literally on top of each other on both sides.

The errors for the parameters from GA/ROM are consistently good for this case. The error in α is 3.89%, κ is 13.3%, ρc is 5.24% and h is 26.16%. The returned error from the ASTM calculation is 195%.

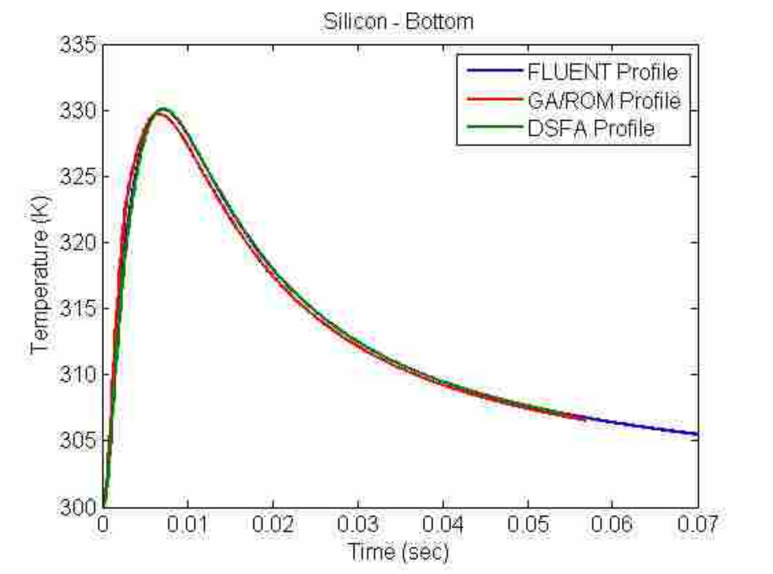


Figure 3-24: Silicon Bottom Temperature Profile Comparisons

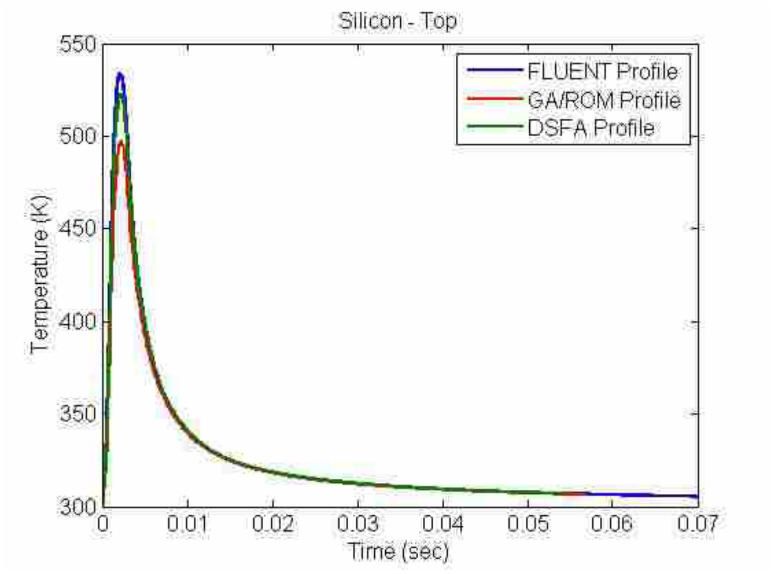


Figure 3-25: Silicon Top Temperature Profile Comparisons

3.8.3 Cupric Oxide

Again, as with the data for Gallium Arsenide and Silicon, the data from Cupric Oxide was input the GA and run for five tests to ensure consistent convergence. The input profile was also used with the ASTM method to determine what it evaluates as the diffusivity. The input values for the Cupric Oxide case are shown in Table 3-8 [Le-Ping et al][Palik].

Table 3-8: Cupric Oxide Input and Outputs

Parameter	*True Value*	*GA/ROM*	*ASTM*
α	3.959 E -6	3.425 E -6	4.254 E -4
κ	960.5	790	-
ρc	5.052 E 6	5.285 E 6	-
h	23.0	31.0	-

The comparisons of the temperatures from the input FLUENT profile, the result of the GA/ROM profile and the DSFA when using the true, known values directly as input into the DSFA are shown in Figure 3-26 for the bottom surface and Figure 3-27 for the top. Figure 3-26 shows very strong matching returned by the GA/ROM solution. Figure 3-27 shows that the profile is not as good of a match from the top surface as the DSFA. The matching by the DSFA with known parameters was not as strong at the peak temperature on the bottom surface, but did match better with the top temperature profile than did the GA/ROM profile.

The errors for the parameters from GA/ROM are consistently good for this case. The error in α is 13.48%, κ is 4.61%, ρc is 17.75% and h is 34.78%. The returned error from the ASTM calculation is two orders of magnitude.

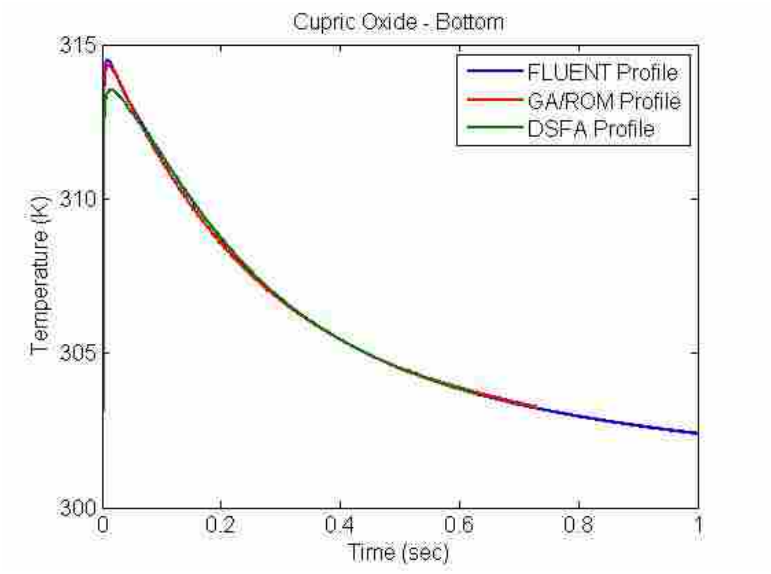


Figure 3-26: Cupric Oxide Bottom Temperature Profile Comparisons

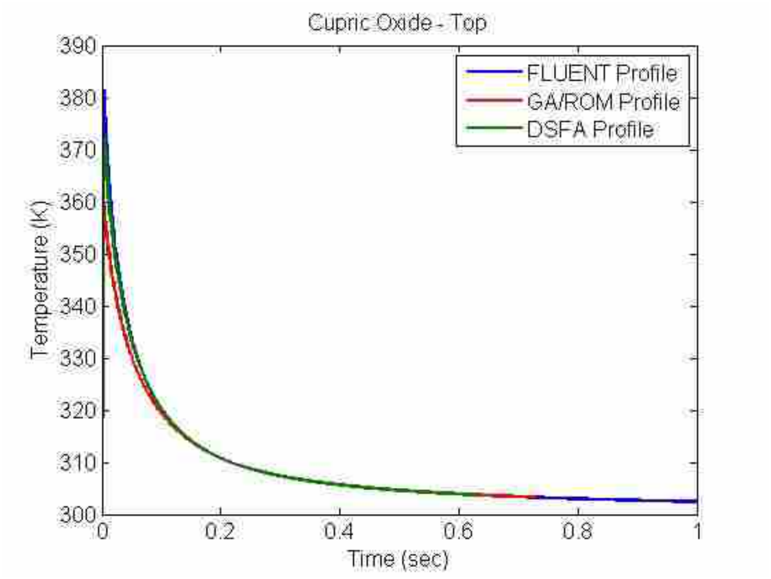


Figure 3-27: Cupric Oxide Top Temperature Profile Comparisons

3.8.4 Aluminum Antimony

Aluminum Antimony was run the same way as the three previous cases. The temperature profile was input the GA and run for five tests to ensure consistent convergence. The input profile was also used with the ASTM method to determine what it evaluates as the diffusivity. The input values for the Aluminum Antimony case are shown in Table 3-9 [Adachi][Chryssis et al].

The comparisons of the temperatures from the input FLUENT profile, the result of the GA/ROM profile input into the DSFA, and the DSFA when using the true, known values directly as input into the DSFA are shown in Figure 3-28 and Figure 3-29 for bottom and top surfaces respectively. The figures both show the high level of accuracy returned by both the GA/ROM solution and the accuracy of the DSFA with matching a profile with known parameters. In this case, the GA/ROM profile lays better against the FLUENT profile for the entire curve, with the DSFA profile once again slightly under at the peak temperature value on both top and bottom profiles.

Table 3-9: Aluminum Antimony Input and Outputs

Parameter	***True Value***	***GA/ROM***	***ASTM***
α	2.006 E -5	2.195 E -5	4.495 E -4
κ	486.0	476.2	-
ρc	1.346 E 6	1.260 E 6	-
h	36.6	23.5	-

The errors for the parameters from GA/ROM are consistently good for this case. The error in α is 9.44%, κ is 6.40%, ρc is 2.01% and h is 35.79%. The returned error from the ASTM calculation is more than 2000%.

The data show the strength of the GA/ROM in its capability to return accurate profiles along the bottom surface that can be utilized to accurately predict the key material properties of the known samples. The value for convective coefficient is just not strong enough to be able to force much effect and its accuracy is more of a case of luck in the GA/ROM more than the strength of the routine.

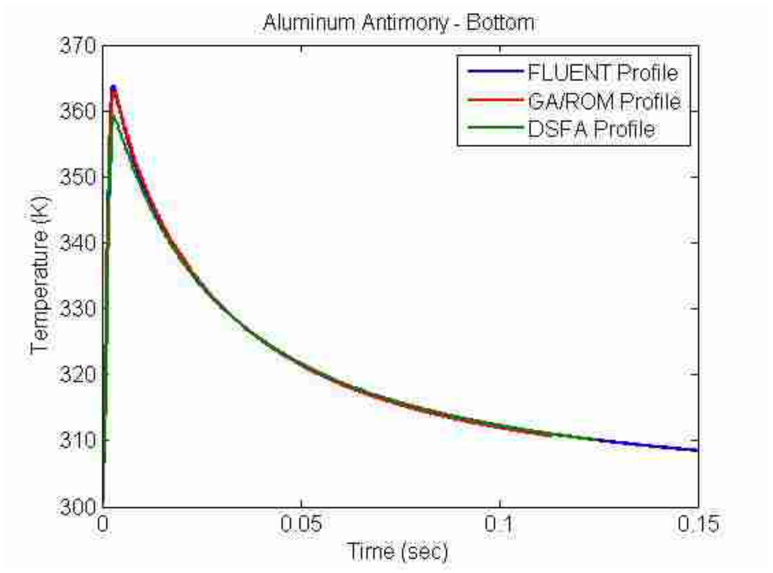


Figure 3-28: Aluminum Antimony Bottom Temperature Profile Comparisons

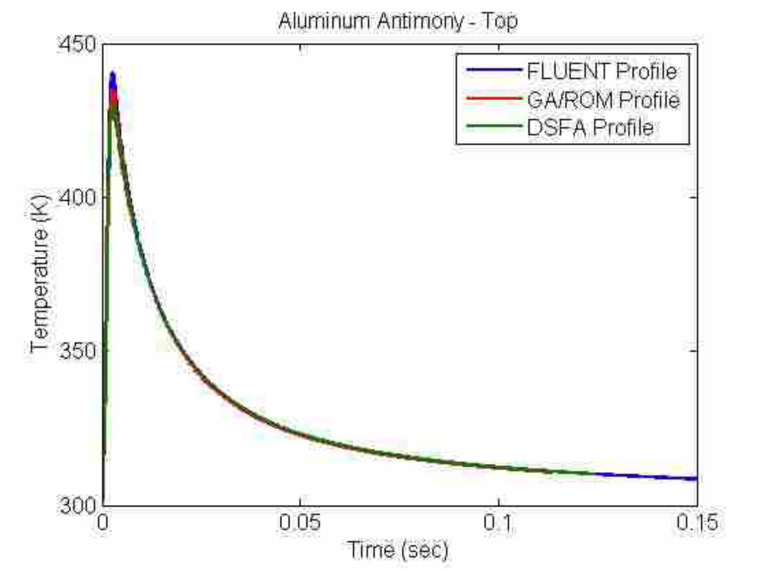


Figure 3-29: Aluminum Antimony Top Temperature Profile Comparisons

4 RESULTS

4.1 **Blind Test Cases**

Similar to what was done for the four cases in Section 3.8, cases were run from FLUENT and used as inputs to the GA/ROM program. In these cases, the parameters were not known beforehand. The data were run and the solutions returned back to the creator of the FLUENT profiles for checking. This process was done to avoid any potential contamination of the results and to exercise the capability to perform a true blind study.

Once the data were run, the true parameters were revealed for evaluation of error as is shown above and also so that the DSFA can be run with the true values to evaluate the accuracy of the DSFA for those parameters as well.

4.1.1 Iron Disilicide

The first full blind test was Iron Disilicide (FeSi2). The provided data did not have a convective coefficient, and as such, the error for h cannot be calculated for the case. The only data provided were arrays of time, bottom temperature, and top temperature. Once the data were run and the GA/ROM returned parameters, they were resent to the creator of the input temperature profiles in FLUENT. The results from the GA/ROM and ASTM methods for the case are shown in Table 4-1 [Milosavljević et al] [Kojima].

Table 4-1: Iron Disilicide Input and Outputs

Parameter	*True Value*	*GA/ROM*	*ASTM*
α	3.163 E -6	3.027 E -6	7.285 E -6
κ	3384	3138	-
ρc	2.846 E 6	2.882 E 6	-
h	-	27.3	-

The errors for the parameters from GA/ROM are consistently good for this case. The error in α is 4.29%, κ is 7.29%, ρc is 1.27% and h is unknown. The returned error from the ASTM calculation is more than 130%.

The plots of the top and bottom temperature profiles are shown in Figure 4-1 and Figure 4-2. In both the figures, the DSFA Profile lays directly on top of the FLUENT Profile. The accuracy of the GA/ROM parameters are clearly seen as the DSFA Profile lays very nearly with them as well. In Figure 4-2, it is impossible to see any differences in any of the three profiles.

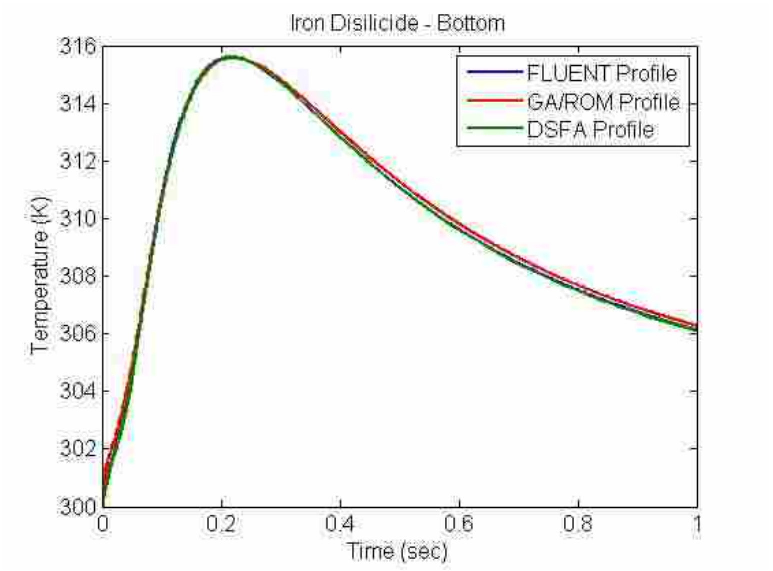


Figure 4-1: Iron Disilicide Bottom Temperature Profile Comparisons

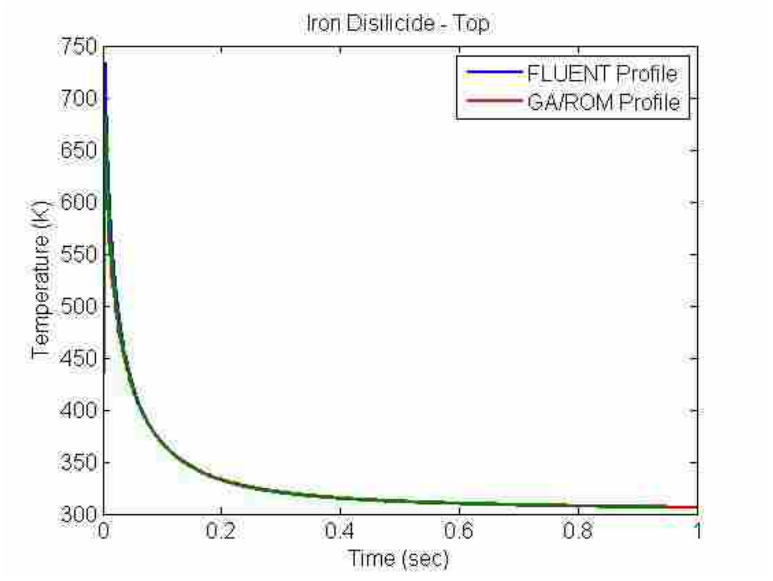


Figure 4-2: Iron Disilicide Top Temperature Profile Comparisons

The errors in the GA/ROM profile from Iron Disilicide are very small and show the accuracy of the entire process from DSFA solution sets through the ROM and GA. The returned data is significantly more accurate than the ASTM method would have predicted for this temperature sample.

4.1.2 Gallium Phosphide

For the blind test of Gallium Phosphide the only data provided were arrays of time, bottom temperature, and top temperature. Once the data were run and the GA/ROM returned parameters, they were resent to the creator of the input temperature profiles in FLUENT. The results from the GA/ROM and ASTM methods for the case are shown in Table 4-2 [Goldbery et al] [Aspnes and Studna]. Once the GA/ROM and ASTM data from Table 4-2 were returned, the parameters were returned to the creator of the profile for comparison to the true values that were used to create the temperature profiles.

Table 4-2: Gallium Phosphide Input and Outputs

Parameter	*True Value*	*GA/ROM*	*ASTM*
α	6.179 E -5	6.043 E -5	1.023 E -4
κ	3680	2826	-
ρc	1.780 E 6	1.913 E 6	-
h	37.0	28.3	-

The errors for the parameters from GA/ROM are consistently good for this case. The error in α is 2.20%, κ is 23.2%, ρc is 7.46% and h is 29.0%. The returned error from the ASTM calculation is 67.0%.

The plots of the top and bottom temperature profiles are shown in Figure 4-3 and Figure 4-4. In both the figures, the DSFA Profile lays directly on top of the FLUENT Profile. The accuracy of the GA/ROM parameters are clearly seen as the DSFA Profile lays very nearly with them as well.

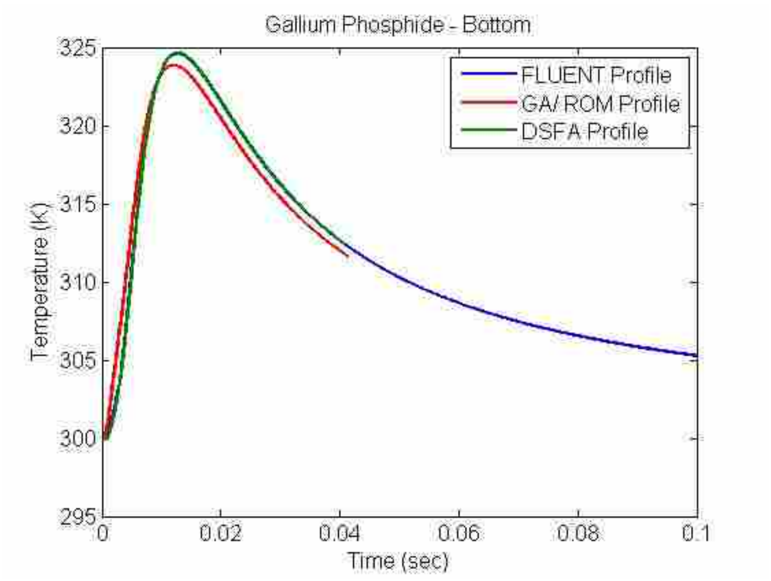


Figure 4-3: Gallium Phosphide Bottom Temperature Profile Comparisons

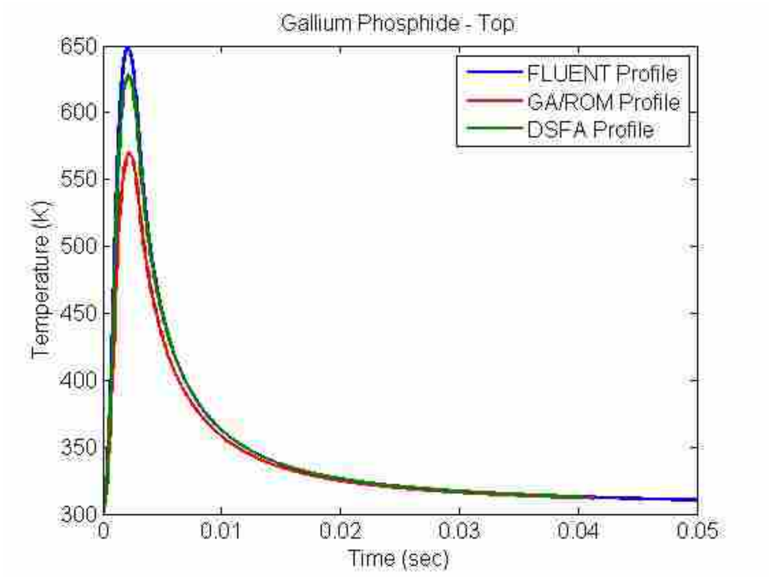


Figure 4-4: Gallium Phosphide Top Temperature Profile Comparisons

The errors in the GA/ROM bottom profile from case of Gallium Phosphide are small and show the accuracy of the entire process from DSFA solution sets through the ROM and GA. The top temperature profiles have a larger amount of error than the bottom and that is displayed in the error in the parameters. The DSFA parameters with the true values are much closer and so more accurate temperature profiles would return more accurate parameter sets. The GA/ROM for the five runs of Gallium Phosphide all successfully converged. The returned data is more accurate than the ASTM method would have predicted for this temperature sample.

4.1.3 Indium Phosphide

The only data provided for the Indium Phosphide case were arrays of time, bottom temperature, and top temperature. Once the data were run and the GA/ROM returned parameters, they were resent to the creator of the input temperature profiles in FLUENT. The results from the GA/ROM and ASTM methods for the case are shown in Table 4-3 [Dargus and Kundrotas][Aspnes and Studna]. Once the GA/ROM and ASTM data from Table 4-3 were returned, the parameters were returned to the creator of the profile for comparison to the true values that were used to create the temperature profiles.

Table 4-3: Indium Phosphide Input and Outputs

Parameter	***True Value***	***GA/ROM***	***ASTM***
α	4.560 E -5	3.066 E -5	4.424 E -4
κ	985.2	642.7	-
ρc	1.491 E 6	1.693 E 6	-
h	37.2	36.3	-

The errors for the parameters from GA/ROM are consistently good for this case. The error in α is 32.8%, κ is 34.8%, ρc is 13.5% and h is 2.41%. The returned error from the ASTM calculation is more than 870%.

The plots of the top and bottom temperature profiles are shown in Figure 4-5 and Figure 4-6. In both the figures, the DSFA Profile lays near to the FLUENT Profile. The accuracy of the GA/ROM parameters are clearly seen as the DSFA Profile lays nearly with them as well.

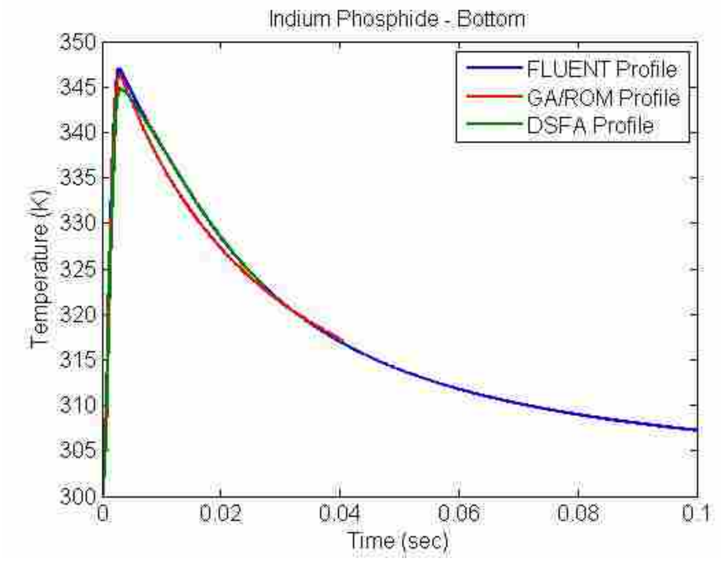


Figure 4-5: Indium Phosphide Bottom Temperature Profile Comparisons

In Figure 4-6, it is seen that there is a significant difference in the GA/ROM profile and the FLUENT profile. This is a case where the measurement of the top surface would be able to be used for calculating the material properties. If the data were taken on the top and utilized similarly to what is done by the bottom surface temperature, there would be a good probability that the inverse solution to the temperature curve would return better solutions.

The return of improved solutions would be to the increase in fitness values that would be returned by the top temperature profile being more accurate. As is shown in Figure 4-5, the data returned by GA/ROM did not lie perfectly on the profile, but it was near to it. With increased fitness pressure coming as well from the top profile, the likelihood of returning parameters with the same amount of error would be small.

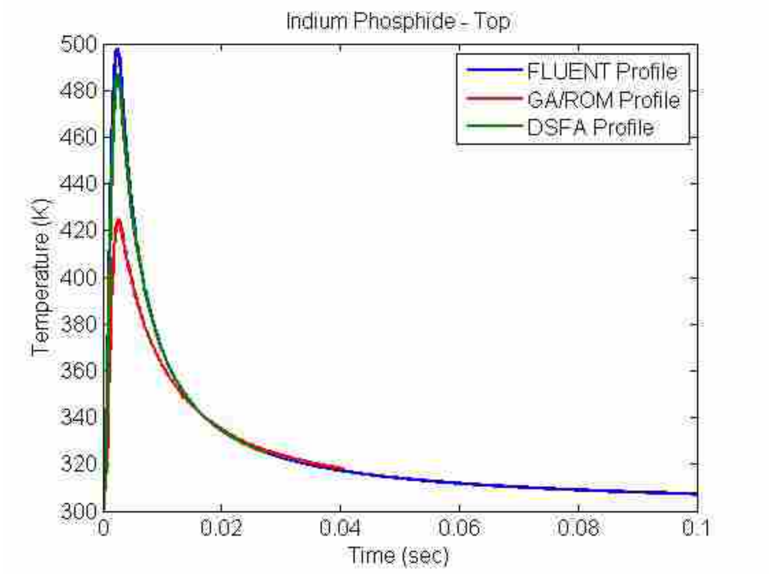


Figure 4-6: Indium Phosphide Top Temperature Profile Comparisons

The errors in the GA/ROM profile from Indium Phosphide are generally small for the bottom surface. The top profiles are not as similar and that is due to the errors in the parameters returned by the GA/ROM against the true values. The temperature profiles matching fairly well on the bottom show that there is a possibility for near non-unique solutions. It is not to be inferred that the profiles will match exactly with varying parameters, but that there can be large variations in parameters that can produce bottom profiles that are similar. The distinction can be made by investigating the top surface along with the bottom surface. The combination of top profiles and bottom profiles are unique. If there are profiles along the bottom that are near when the parameters are far, there will be large differences in the top profiles as shown in Figure 4-5 and Figure 4-6.

The returned data is still much more accurate than the ASTM method would have predicted for this temperature sample. The DSFA parameters with the true values are much closer and so more accurate temperature profiles would return more accurate parameter sets. The GA/ROM for the five runs of Indium Phosphide converged successfully each time without timing out of its 500 generation limit.

4.1.4 Zinc Selenide

For the case of Zinc Selenide, the only data provided were arrays of time, bottom temperature, and top temperature. Once the data were run and the GA/ROM returned parameters, they were resent to the creator of the input temperature profiles in FLUENT. The results from the GA/ROM and ASTM methods for the case are shown in Table 4-4 [Crystran]. Once the GA/ROM and ASTM data from Table 4-4 were returned, the parameters were returned to the creator of the profile for comparison to the true values that were used to create the temperature profiles.

The errors for the parameters from GA/ROM are consistently good for this case. The error in α is 0.54%, κ is 6.86%, ρc is 8.05% and h is 39.4%. The returned error from the ASTM calculation is more than 4000%.

Table 4-4: Zinc Selenide Input and Outputs

Parameter	*True Value*	*GA/ROM*	*ASTM*
α	1.008 E -5	1.013 E -5	4.147 E -4
κ	517.9	482.4	-
ρc	1.787 E 6	1.643 E 6	-
h	37.6	22.76	-

The plots of the top and bottom temperature profiles are shown in Figure 4-7 and Figure 4-8. In both the figures, the DSFA Profile lays directly on top of the FLUENT Profile. The accuracy of the GA/ROM parameters are clearly seen as the DSFA Profile lays very nearly with them as well. In Figure 4-8, it is impossible to see any differences in any of the three profiles.

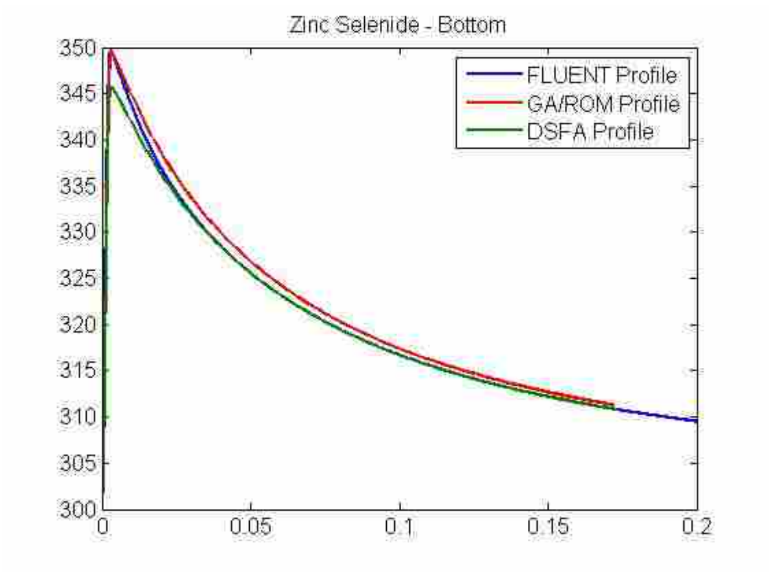


Figure 4-7: Zinc Selenide Bottom Temperature Profile Comparisons

The errors in the GA/ROM profile from Zinc Selenide are very small and show the accuracy of the entire process from DSFA solution sets through the ROM and GA. The top temperature profiles have a larger amount of error than the bottom and that is displayed in the error in the parameters. The DSFA parameters with the true values are much closer and so more accurate temperature profiles would return more accurate parameter sets.

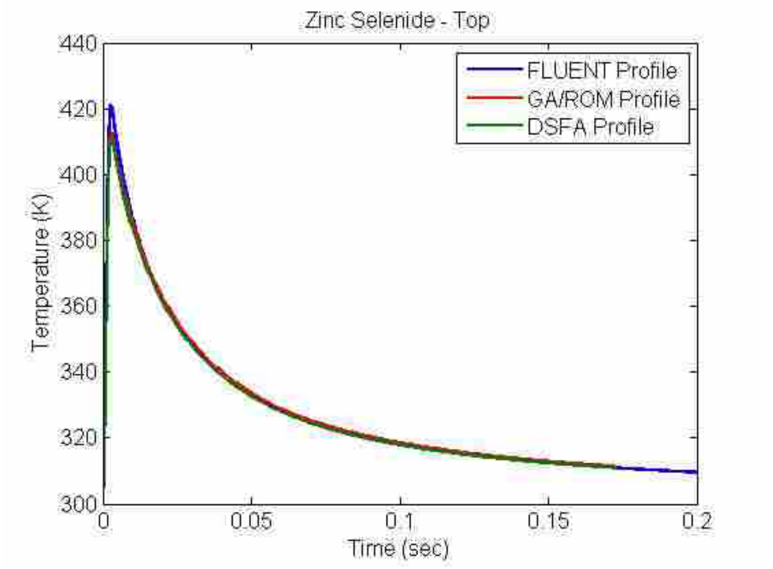


Figure 4-8: Zinc Selenide Top Temperature Profile Comparisons

The returned data is significantly more accurate than the ASTM method would have predicted for this temperature sample. For each test run with Zinc Selenide, the simulation was able to successfully converge before reaching its 500 generation limit.

4.1.5 Aluminum Gallium Arsenide

The last blind test case was Aluminum Gallium Arsenide. The only data provided were arrays of time, bottom temperature, and top temperature. Once the data were run and the GA/ROM returned parameters, they were resent to the creator of the input temperature profiles in FLUENT. The results from the GA/ROM and ASTM methods for the case are shown in Table 4-5 [Goldbery et al] [Kelso et al].

Once the GA/ROM and ASTM data from Table 4-5 were returned, the parameters were returned to the creator of the profile for comparison to the true values that were used to create the temperature profiles.

Table 4-5: Aluminum Gallium Arsenide Input and Outputs

Parameter	*True Value*	*GA/ROM*	*ASTM*
α	6.009 E -5	5.958 E -5	9.547 E -5
κ	4089.6	2877.7	-
ρc	1.771 E 6	1.925 E 6	-
h	12.1	25.3	-

The errors for the parameters from GA/ROM are consistently good for this case. The error in α is 6.84%, κ is 29.63%, ρc is 8.72% and h is 110%. The returned error from the ASTM calculation is 58.9%.

The plots of the top and bottom temperature profiles are shown in Figure 4-9 and Figure 4-10. In both the figures, the DSFA Profile lays directly on top of the FLUENT Profile. The accuracy of the GA/ROM parameters are clearly seen as the DSFA Profile lays very nearly with them as well. In Figure 4-10, it is impossible to see any differences in any of the three profiles.

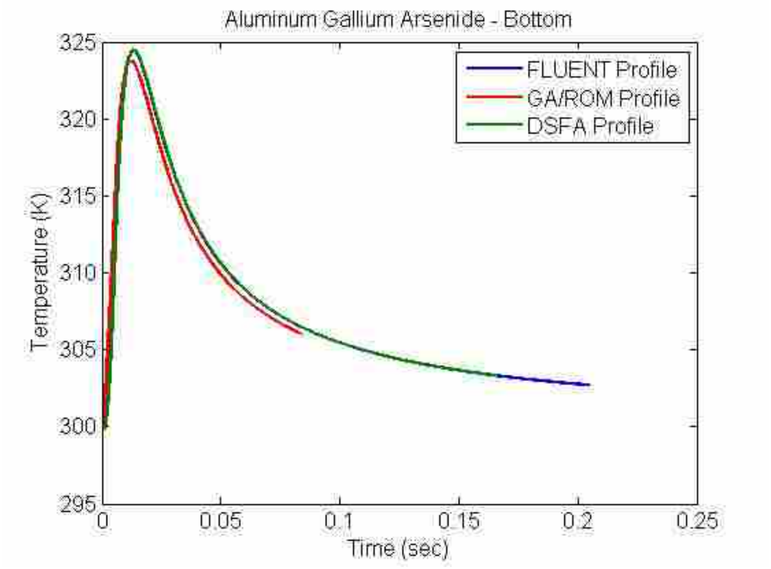


Figure 4-9: Aluminum Gallium Arsenide Bottom Temperature Profile Comparisons

The errors in the GA/ROM bottom profile from Aluminum Gallium Arsenide are small and show the accuracy of the entire process from DSFA solution sets through the ROM and GA. The top temperature profiles have a larger amount of error than the bottom and that is displayed in the error in the parameters. The DSFA parameters with the true values are much closer and so more accurate temperature profiles would return more accurate parameter sets. The GA/ROM for the five runs of Aluminum Gallium Arsenide all successfully converged. The returned data is more accurate than the ASTM method would have predicted for this temperature sample.

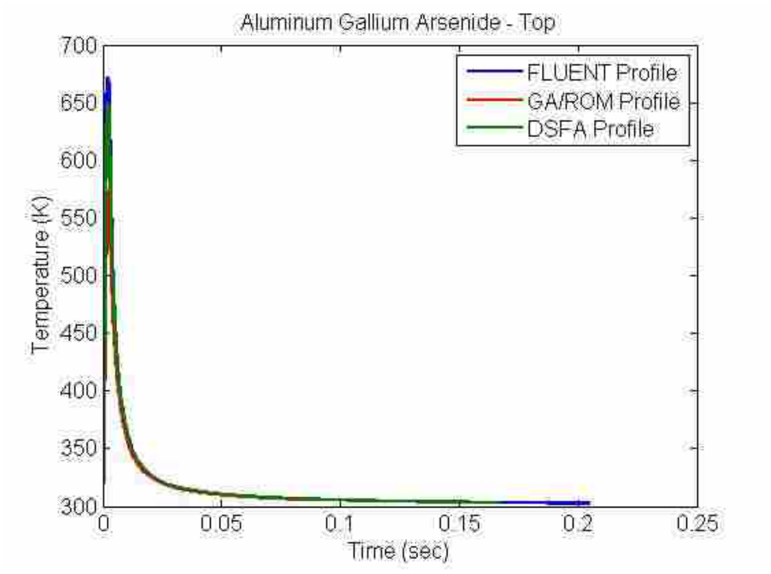


Figure 4-10: Aluminum Gallium Arsenide Top Temperature Profile Comparisons

For each of the blind case tests, the parameters for diffusivity returned by the GA were more accurate than the calculation using the ASTM method. The errors in the temperature profiles are generally quite small and thus GA/ROM is able to quickly and accurately determine the desired parameter of diffusivity. The procedure is also able to return parameters for heat capacity and optical depth of the unknown materials. The values for convective heat transfer are not strong enough to give a high level of confidence in their measurements, but the parameter is not a material property.

The convergence criterion was met for each run of the GA. There were no tests in which the convergence criterion was not able to be met in the 500 generations. The GA was run for each profile five times to ensure that the GA would be able to return consistent solutions as the GA process is inherently random and non-deterministic. The data was investigated and shown to be able to return results that were both accurate and consistent as their variations were small.

5 SUMMARY AND CONCLUSIONS

Modern engineering practice increasingly relies on the ability to perform precise numerical simulations in order to optimize the design of systems. The accuracy of these numerical simulations depends on the accuracy of the material properties that are required input parameters. Therefore, the ability to accurately measure material properties is critical in modern engineering practices.

The thermal diffusivity is a key material property needed to perform thermal analyses of engineered systems. Over the last few decades, pulsed laser diffusion (PLD) systems have become the method of choice for making measurements of thermal diffusivity. Previous PLD models have been based on a number of highly restrictive assumptions. The results presented in this thesis show how a higher fidelity PLD model may be implemented and used to more accurately measure the thermal diffusivity and other properties of various materials.. The Distributed Source – Finite Absorption (DSFA) model proposed in this thesis accounts for the most important effects that were neglected in previous DSFA models. The top surface was not taken to be adiabatic, but allowed for free convection to affect the material sample. The laser pulse power was not assumed to be completely absorbed at the top surface, but realistically modeled as being absorbed throughout the sample. The laser power was assumed to take on a Gaussian profile rather than a uniform profile across the sample surface. The laser pulse was not taken to be infinitesimally short, but to be a triangle pulse rising to a maximum temperature at a certain time and returning back to zero. As demonstrated through comparison with full CFD simulations of a PLD system, the DSFA results in a high fidelity model of a practical PLD system.

In this thesis, the development and performance of previous PLD models have been thoroughly reviewed. The method first established by Parker in 1961 was introduced and developed. The model was based on a number of assumptions about the physics of the process.

The Parker model assumes:

- one-dimensional heat transfer
- uniform heating in a thin layer on the top surface of the sample
- all surfaces are adiabatic
- infinitesimally short pulse time
- uniform energy on the top surface of the sample
- no radiative effects

Parker proposed that α can be found by inputting the thickness of the sample and the time that it takes for the rear face to get to half of its maximum value in a simple equation shown in Eq. (2-13).

Maglić utilized the same model development as Parker and used an expanded form of the Parker equation for finding α as shown in Eq. (2-14). He uses a set of values that correspond to the percent rise of the temperature as shown in Table 2-1. The values of α can then be averaged over the temperature rise curve for the α of the sample.

ASTM has accepted a method that also uses the same development from Parker and the same expanded equation from Maglić. It suggests that you compare the values of the 50% rise time to those of the 25% rise time and the 75% rise time. If they match within 5% of each other, then the value computed is the accepted α. The ASTM method is the generally accepted industry standard.

In order to avoid the use of the same assumptions as is done in the previous models, the DSFA model is proposed. The DSFA model returns the temperature at any point in the sample at any time. The new model allows for two-dimensional heat transfer, non-uniform heating effects in time and space, and heat loss from the exposed surface.

ROM is a fairly accurate technique that can be used to solve the new equation. The ROM is faster than the full computation solution of the new technique. The errors introduced by running the ROM are generally small. However, using the ROM developed in this work does add the possibility for non-unique solutions.

The data clearly shows a trend that as the noise in the parameter sets grows, the errors in the predictions for the individual parameters also grows. These tests show the robustness of the method of GA solving the inverse problem with ROM. In the presence of noise the GA with ROM can consistently perform and return an acceptable set of parameters. Due to the non-uniqueness of the solution sets, the returned solution can come to a number of different values.

The time required to return the nondimensional profile is minimized due to the use of the ROM being utilized. The GA was run for each case up to 500 generations. Running all 500 generations of a population set of 150 is 75100 function calls. The time for each function call of the ROM is 0.0025 sec making the GA return a value in approximately 187.8 seconds or about 3.13 minutes. For the GA to use the DSFA, which returns a profile in approximately 0.85 sec, the GA would take more than 17 hours to return a solution. The ROM allows for processing at a rate of 340 times faster.

Future work that can be done to improve the accuracy of the inverse operation could include measurement of the top surface of the sample that is being irradiated. The case that had larger errors were due to nonuniqueness of the solution for the bottom temperature. When the top surface is investigated, it is shown that the returned parameters created a profile that did not well match. If the top surface temperature errors were included in the fitness value, it stands to reason that the GA would have returned parameters with smaller errors. The inclusion of the top surface would likely also reduce the errors in the calculation when the errors were not large. The improvement in accuracy of the measurement technique could be a driving force for standard testing procedure modification.

REFERENCES

Adachi,S., (2004) *Handbook on Physical Properties of Semiconductors: III-V Compounds Semiconductors, Vol 2*., Kluwer Academic Publishers, Norwell, MA.

Aspnes, D. E., Studna, A. A., (1983). “Dielectric functions and optical parameters of Si, Ge, GaP, GaAs, GaSb, InP, InAs, and InSb from 1.5 to 6.0 eV,” *Physics Review B,* 27 (2), 985-1009.

ASTM E 1461-01 (2001). “Standard Test Method for Thermal Diffusivity by the Flash Method.” ASTM International, West Conshohocken, PA.

Baba, T., Ono, A. (2001). “Improvements of the laser flash method to reduce uncertainty in thermal diffusivity measurements.” *Measurement Science and Technology*, 12, 2046-2057.

Beck, J. V., Arnold, K. J. (1977). *Parameter Estimation in Engineering and Science*, John Wiley and Sons, Hoboken, NJ.

Bergman, M., Cordier, L., Brancher, J. P. (2005). “Optimal rotary control of the cylinder wake using proper orthogonal decomposition reduced-order model.” *Physics of Fluids*, 17, 097101 (21 pgs.).

Carlson, R. O., Slack, G. A., Silverman, S. J., (1965). “Thermal Conductivity of GaAs and GaAsP Laser Semiconductors”, *Journal of Applied Physics*, 36 (2), 505.

Carslaw, H. S., Jaeger, J. C. (1959). *Conduction of Heat in Solids, 2nd Edition*, Oxford University Press, New York, NY.

Crystran, “Zinc Selenide Data Sheet.” Crystran, Ltd., Poole, UK.

Chryssis, A., Ryu, G., Dagenais, M., (2010) “Thermal Impedance of Epi-Up and Epi-Down Interband Cascade Lasers.” *23rd Annual Meeting of the IEEE Photonics Society*, Denver, CO.

Davis, L. (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, NY.

Dargys A., Kundrotas, J. (1994). *Handbook on Physical Properties of Ge, Si, GaAs and InP*, Vilnius, Science and Encyclopedia Publishers, 1994

Fullem, T. Z., Rae, D. F., Sharma, A., Wolcott, J. A., Cotts, E. J. (2008). “Thermal characterization of thermal interface material bondlines.” *ITHERM 2008. 11th*

Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. 174-179.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Reading, MA.

Goldbery, Y. A., Levinshtein, M., Rumyantsev , S., Shur, M. (1996). *Handbook Series on Semiconductor Parameters, Vol 1.* World Scientific, London, UK.

Hardy, R. L. (1971). "Multiquadratic equations of topography and other irregular surfaces." *Journal of Geophysics Research*, 76, 1905-1915.

Hardy, R. L. (1990). "Theory and applications of the multiquadratic-biharmonic method: 20 years of discovery 1968-1988." *Computational Math Applications*, 19, 163-208.

Hay, B., Filtz, J.R., Hameury, J., Rongione, L. (2005). "Uncertainty of Thermal Diffusivity Measurements by Laser Flash Method." *International Journal of Thermodynamics*, 26, 1883-1898.

Haydaryi, M., Moskin, M. M., Yahya, N., Yunus, W. M. M., Grozescu, V. I. (2005). "Thermal Wave Study on Carbon Nanotube-filled Polymer Films at Low Temperatures by Using Flash Technique." *American Journal of Applied Science*, Special Issue, 49-52.

Incropera, F. P., DeWitt, D. P. (2002). *Fundamentals of Heat and Mass Transfer, Fifth Edition*, John Wiley and Sons, Hoboken, NJ.

Kelso, S. M., Aspnes, D. E., Logan, R. A., Bhat, R., (1986). "Optical Properties of AlGaAs," *Journal of Applied Physics*, 60 (2), 754-767.

Kojima, T., (2006). "Semiconducting and Thermoelectric Properties of Sintered Iron Disilicide," *Physica Status Solidi (a)*. 111, 233-242.

Larson, R. S. (2007). "Computationally Efficient Modeling of Transient Radiation in a Purely Scattering Foam Layer." Masters Thesis, Brigham Young University, Provo, UT.

Le-Ping Zhou, Bu-Xuan Wang, Xiao-Feng Peng, Xiao-Ze Du, and Yong-Ping Yang (2010). "On the Specific Heat Capacity of CuO Nanofluid," *Advances in Mechanical Engineering*, vol. 2010, Article ID 172085, 4 pages.

Maglić, K. D., Cazairliyan, A., Peletsky, V. E. (1992). *Compendium of Thermophysical Property Measurement Methods Vol. 2*, Plenum Press, New York, NY.

McMasters, R. L., Beck, J. V., Dinwiddie, R. B., Wang, H. (1999). "Accounting for Penetration of Laser Heating in Flash Thermal Diffusivity Experiments." *Journal of Heat Transfer*, 121, 15-21.

Milosavljević, M., Shao, G., Bibić, N., McKinty, C. N., Jeynes, C., Homewood, K. P., (2001). "Amorphous-iron disilicide: A promising semiconductor." *Applied Physics Letters*, 79, 10.

Ostrowski, Z., Bialecki, R. A., Kassab, A. J. (2005). "Estimation of constant thermal conductivity by use of Proper Orthogonal Decomposition." *Computational Mechanics*, 37, 52-59.

Ostrowski, Z., Bialecki, R. A., Kassab, A. J. (2007). "Solving inverse heat conduction problems using trained POD-RBF network inverse method." *Inverse Problems in Science and Engineering*, 16(1), 39-54.

Okhotin, A. S., Pushkarskii, A. S., Gorbachev, V. V. (1972). *Thermophysical Properties of Semiconductors*, "Atom" Publishing House, Moscow, Russia

Palik, E. D., (1985). *Handbook of Optical Constants of Solids*, Academic Press, San Diego, CA.

Parker, W. J., Jenkins, R. J., Butler, C. P., Abbot, G. L. (1961). "Flash Method of Determining Thermal Diffusivity, Heat Capacity, and Thermal Conductivity." *Journal of Applied Physics*, 32, 1679-1684.

Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space." *Philosophical Magazine*, 2, 559-572.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P. (1992). *Numerical Recipes in FORTRAN: The Art of Scientific Computing*. Cambridge University Press, New York, NY.

Rambo, J., Joshi, Y. (2007). "Reduced-order modeling of turbulent forced convection with parametric conditions." *International Journal of Heat and Mass Transfer*, 50, 539-551.

Sharmin, M., Choudhury, S., Akhtar, N., Begum, T., (2012). "Optical and Transport Properties of p-Type GaAs." *Journal of Bandladech Academy of Sciences*, 36 (1), 97-107.

Strang, G. (2006). *Linear Algebra and Its Application*, Thompson Learning/Brooks/Cole, Belmont, CA.

Tarantola, A. (2005). *Inverse Problem Theory and Methods for Model Parameter Estimation*, SIAM, Philadelphia, PA.

Vozar, L., Hohenauer, W. (2003/2004). "Flash method of measuring the thermal diffusivity. A review." *High Temperatures-High Pressures*, 35/36, 253-264.

Weber, M. J. (2003). *Handbook of Optical Materials*. CRC Press, Boca Raton, FL.

Yamaguchi, K., Itagaki, K., Yazawa, A. (1989) “High-Temperature Heat-Content Measurements of the AlN, AlP, AlAs, AlSb, GaN, GaP, GaAs, GaSb, InN, InP, InAs, InSb, Compounds.” *Journal of the Japan Institute of Metals, 53(8), 764-770.*

APPENDIX A. PARKER MODEL DEVELOPMENT

$$\frac{1}{r}\frac{\partial}{\partial r}\left(kr\frac{\partial T}{\partial r}\right)+\frac{1}{r^2}\frac{\partial}{\partial \phi}\left(k\frac{\partial T}{\partial \phi}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+\dot{q}=\rho_m c\frac{\partial T}{\partial t}$$

The one-dimensional analysis is such that

$$\frac{\partial T}{\partial \phi}=0, \frac{\partial T}{\partial r}=0$$

Assuming constant *k, ρ_m,* and *c* allows

$$\alpha=\frac{k}{\rho_m c}$$

No internal heat generation gives

$$\dot{q}=0$$

To return the simplified heat equation to

$$\frac{\partial^2 T}{\partial z^2}=\frac{1}{\alpha}\frac{\partial T}{\partial t}$$

With the adiabatic boundary conditions of

$$\left.\frac{\partial T}{\partial z}\right|_{z=0}=\left.\frac{\partial T}{\partial z}\right|_{z=L}=0$$

The initial condition is defined as

$$T(z,0)=\begin{cases}T_0+\dfrac{Q}{\rho c\delta A} & 0\le z\le\delta \\ T_0 & \delta<z\le L\end{cases}$$

Non-dimensionalization is done by allowing

$$\theta = \frac{T - T_0}{\left(Q \big/ \rho c L A\right)}, \zeta = \frac{z}{L}, \tau = \frac{\alpha t}{L^2}$$

The heat equation now non-dimensionalized is

$$\frac{\partial^2 \theta}{\partial \zeta^2} = \frac{\partial \theta}{\partial \tau}$$

With the non-dimensionalized boundary conditions of

$$\left.\frac{\partial \theta}{\partial \zeta}\right|_{\zeta=0} = \left.\frac{\partial \theta}{\partial \zeta}\right|_{\zeta=1} = 0$$

And the non-dimensionalized initial condition of

$$\theta(\zeta,0) = \begin{cases} \frac{L}{\delta} & 0 \le \zeta \le \frac{\delta}{L} \\ 0 & \frac{\delta}{L} < \zeta \le 1 \end{cases}$$

The equation can now be solved using the technique of separation of variables. The base equation is defined as

$$\theta = \mathrm{Z}(\zeta)\mathrm{T}(\tau)$$

The base definition is substituted into the differential equation

$$\mathrm{T}\frac{\partial^2 \mathrm{Z}}{\partial \zeta^2} = \mathrm{Z}\frac{\partial \mathrm{T}}{\partial \tau}$$

The equation is now separated with the eigenvalue defined

$$\frac{1}{\mathrm{Z}}\frac{\partial^2 \mathrm{Z}}{\partial \zeta^2} = \frac{1}{\mathrm{T}}\frac{\partial \mathrm{T}}{\partial \tau} = -\lambda^2$$

The individual equations can now be evaluated separately. First, the Z equation will be solved:

$$\frac{\partial^2 \mathrm{Z}}{\partial \zeta^2} + \lambda^2 \mathrm{Z} = 0$$

For the basic second order equation, we can solve using the sin and cos functions

$$\mathrm{Z} = C_1 \cos(\lambda\zeta) + C_2 \sin(\lambda\zeta)$$

Now, the T equation will be solved:

$$\frac{\partial \mathrm{T}}{\partial \tau} + \lambda^2 \mathrm{T} = 0$$

The first order equation can be solved with an exponential

$$\mathrm{T} = C_3 \exp(-\lambda^2 \tau)$$

The Z and T equation are recombined

$$\theta = C_3 \exp(-\lambda^2 \tau)[C_1 \cos(\lambda\zeta) + C_2 \sin(\lambda\zeta)]$$

The equation is differentiated so the boundary conditions can be applied to begin to solve for the constants

$$\frac{\partial \theta}{\partial \zeta} = C_3 \exp(-\lambda^2 \tau)\lambda[-C_1 \sin(\lambda\zeta) + C_2 \cos(\lambda\zeta)]$$

$$\left.\frac{\partial \theta}{\partial \zeta}\right|_{\zeta=0} = \left.\frac{\partial \theta}{\partial \zeta}\right|_{\zeta=1} = 0$$

The boundary conditions force $C_2 = 0$ as the value of cos(0) is not 0, the only way for the expression to be 0 is if the constant is 0. The value of sin(λ) is 0 when and only when λ=nπ

$$\cos(0) \neq 0 \rightarrow C_2 = 0$$
$$\sin(\lambda) = 0 \rightarrow \lambda = n\pi, \; when\; n = 0, 1, 2, \cdots$$

The equation is now a solved with an infinite sum starting at 0 with both constants C_1 and C_3 can be combined to C_n

$$\theta = \sum_{n=0}^{\infty} C_n \exp\left(-n^2\pi^2\tau\right)\cos(n\pi\zeta)$$

To solve for the values of C_n, orthogonality is used to isolate and solve in the summation

$$\int_0^1 \theta\cos(m\pi\zeta)d\zeta = \sum_{n=0}^{\infty} C_n \exp\left(-n^2\pi^2\tau\right)\int_0^1 \cos(n\pi\zeta)\cos(m\pi\zeta)d\zeta$$

Recall the orthogonality rules that

$$\int_0^1 \cos(n\pi\zeta)\cos(m\pi\zeta)d\zeta = \begin{cases} 0 & if \quad m \neq n \\ 1/2 & if \quad m = n \neq 0 \\ 1 & if \quad m = n = 0 \end{cases}$$

Applying the orthogonality rules

$$\frac{C_m}{2} = \int_0^1 \theta(\zeta,0)\cos(m\pi\zeta)d\zeta$$

$$C_0 = \int_0^1 \theta(\zeta,0)d\zeta$$

For $m = 0$

$$C_0 = \int_0^1 \theta(\zeta,0)d\zeta = \int_0^{\delta/L} \frac{L}{\delta}d\zeta = 1$$

For $m \neq 0$

$$C_m = 2\int_0^{\delta/L} \frac{L}{\delta}\cos(m\pi\zeta)d\zeta$$

$$C_m = \frac{2L}{\delta}\left[\frac{\sin(m\pi\zeta)}{m\pi}\right]_0^{\delta/L}$$

$$C_m = \frac{2L}{m\pi\delta}\sin\left(\frac{m\pi\delta}{L}\right)$$

As the radiation is assumed to be absorbed is a very small depth, the small angle approximation is used to remove the sin term

$$\sin\left(\frac{m\pi\delta}{L}\right) \approx \frac{m\pi\delta}{L}$$
$$C_m = 2$$

Substituting the constants back into the base equation,

$$\theta = 1 + 2\sum_{n=1}^{\infty} \exp\left(-n^2\pi^2\tau\right)\cos\left(n\pi\zeta\right)$$

Redimensionalize the equation

$$\frac{T(z,t) - T_o}{Q/\rho cLA} = 1 + 2\sum_{n=1}^{\infty} \exp\left(\frac{-n^2\pi^2\alpha t}{L^2}\right)\cos\left(\frac{n\pi z}{L}\right)$$

Solve for temperature at the bottom surface, *z* = *L*

$$\frac{T(L,t) - T_o}{Q/\rho cLA} = 1 + 2\sum_{n=1}^{\infty} \exp\left(\frac{-n^2\pi^2\alpha t}{L^2}\right)\cos\left(n\pi\right)$$

Solve for the ½ temperature rise on the bottom surface where at $t = t_{0.5}$

$$\frac{T(L,t_{0.5}) - T_o}{Q/\rho cLA} = 0.5$$

$$1/2 = 1 + 2\sum_{n=1}^{\infty} \exp\left(\frac{-n^2\pi^2\alpha t_{0.5}}{L^2}\right)\cos\left(n\pi\right)$$

As cosine is an odd function, the infinite sum of the *cos* term can be substituted for $(-1)^n$

$$1/2 = 1 + 2\sum_{n=1}^{\infty} (-1)^n \exp\left(\frac{-n^2\pi^2\alpha t_{0.5}}{L^2}\right)$$

Evaluate the expression at one term

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right)$$

Solving for the expression yields

$$\frac{-\pi^2 \alpha t_{0.5}}{L^2} = \ln(1/4) = -1.3863$$

Solving for *α* gives

$$\alpha = \frac{0.14046 L^2}{t_{0.5}}$$

Evaluating the expression at two terms

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right) + \exp\left(\frac{-4\pi^2 \alpha t_{0.5}}{L^2}\right)$$

Isolating a single exponential

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right)\left[1 - \exp\left(\frac{-3\pi^2 \alpha t_{0.5}}{L^2}\right)\right]$$

Using the single term approximation for the exponential term

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right)[1 - \exp(3(-1.3863))]$$

$$-\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right) = 0.25397$$

$$\frac{-\pi^2 \alpha t_{0.5}}{L^2} = -1.3705$$

Solving for *α* for two terms gives

$$\alpha = \frac{0.13887 L^2}{t_{0.5}}$$

Evaluating the expression and solving for three terms

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right) + \exp\left(\frac{-4\pi^2 \alpha t_{0.5}}{L^2}\right) - \exp\left(\frac{-9\pi^2 \alpha t_{0.5}}{L^2}\right)$$

Isolating a single exponential

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right)\left[1 - \exp\left(\frac{-3\pi^2 \alpha t_{0.5}}{L^2}\right) + \exp\left(\frac{-8\pi^2 \alpha t_{0.5}}{L^2}\right)\right]$$

Using the two term approximation for the exponential term

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right)\left[1 - \exp(3(-1.3863)) + \exp(8(-1.3863))\right]$$

$$-\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right) = 0.25416$$

$$\frac{-\pi^2 \alpha t_{0.5}}{L^2} = -1.3698$$

Solving for α for three terms gives

$$\alpha = \frac{0.13879 L^2}{t_{0.5}}$$

Evaluating the expression and solving for four terms

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right) + \exp\left(\frac{-4\pi^2 \alpha t_{0.5}}{L^2}\right) - \exp\left(\frac{-9\pi^2 \alpha t_{0.5}}{L^2}\right) + \exp\left(\frac{-16\pi^2 \alpha t_{0.5}}{L^2}\right)$$

Isolating a single exponential

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right)\left[1 - \exp\left(\frac{-3\pi^2 \alpha t_{0.5}}{L^2}\right) + \exp\left(\frac{-8\pi^2 \alpha t_{0.5}}{L^2}\right) - \exp\left(\frac{-15\pi^2 \alpha t_{0.5}}{L^2}\right)\right]$$

Using the three term approximation for the exponential term

$$-1/4 = -\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right)\left[1 - \exp(3(-1.3698)) + \exp(8(-1.3698)) - \exp(15(-1.3698))\right]$$

$$-\exp\left(\frac{-\pi^2 \alpha t_{0.5}}{L^2}\right) = 0.25417$$

$$\frac{-\pi^2 \alpha t_{0.5}}{L^2} = -1.3698$$

Solving for α for four terms gives

$$\alpha = \frac{0.13879 L^2}{t_{0.5}}$$

The solution for four terms is the same as the solution for three terms at the numerical precision used. Thus, we are able to say that the solution has converged and three terms are all that are needed for the infinite sum and the solution given can be used.

APPENDIX B. DSFA DEVELOPMENT

$$\frac{1}{r}\frac{\partial}{\partial r}\left(kr\frac{\partial T}{\partial r}\right)+\frac{1}{r^2}\frac{\partial}{\partial \phi}\left(k\frac{\partial T}{\partial \phi}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial T}{\partial z}\right)+\dot{q}=\rho_m c\frac{\partial T}{\partial t}$$

$$\frac{\partial T}{\partial \phi}=0\,,\ \frac{\partial^2 T}{\partial \phi^2}=0$$

Assuming constant *k, ρ_m,* and *c* allows

$$\alpha=\frac{k}{\rho_m c}$$

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right)+\frac{\partial^2 T}{\partial z^2}+\frac{\dot{q}}{k}=\frac{1}{\alpha}\frac{\partial T}{\partial t}$$

$$\dot{q}=\frac{P\kappa}{\pi r_o{}^2}\exp\left[\frac{-r^2}{r_o{}^2}-\kappa(L-z)\right]f(t)$$

From Figure 3-7 the pulse is modeled as:

$$f(t)=\begin{cases}\dfrac{t}{t_m} & 0\le t\le t_m\\[2ex] \dfrac{t_p-t}{t_p-t_m} & t_m<t\le t_p\\[2ex] 0 & t_p<t\end{cases}$$

Boundary Conditions:

$$\left.\frac{\partial T}{\partial r}\right|_{r=0} = \left.\frac{\partial T}{\partial r}\right|_{r=R} = 0$$

$$\left.\frac{\partial T}{\partial z}\right|_{z=0} = 0$$

$$\left.-k\frac{\partial T}{\partial z}\right|_{z=L} = h\left(T\left(r,L,t\right)-T\infty\right)$$

Initial Condition:

$$\left.\frac{\partial T}{\partial z}\right|_{z=0} = 0$$

Nondimensionalize the equation with the following parameters:

$$\rho = \frac{r}{R}, \zeta = \frac{z}{L}, \tau = \frac{\alpha t}{R^2}, \theta = \frac{T-T_\infty}{T_{ref}}, a = \frac{R}{L}, S = \kappa L, Bi = \frac{hL}{k}$$

Rearranging the nondimensional values:

$$r = \rho R, z = \zeta L, L = \frac{z}{\zeta}, t = \frac{\tau R^2}{\alpha}$$

Substituting variables into base equation:

$$\frac{1}{\rho R}\frac{\partial}{R\partial\rho}\left(\rho R\frac{T_{ref}\partial\theta}{R\partial\rho}\right)+\frac{T_{ref}\partial^2\theta}{L^2\partial\zeta^2}+\frac{\dot{q}}{k}=\frac{T_{ref}}{R^2}\frac{\partial\theta}{\partial\tau}$$

$$\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial\theta}{\partial\rho}\right)+\frac{R^2}{L^2}\frac{\partial^2\theta}{\partial\zeta^2}+\frac{\dot{q}R^2}{kT_{ref}}=\frac{\partial\theta}{\partial\tau}$$

$$\frac{\dot{q}R^2}{kT_{ref}}=\frac{P\kappa R^2}{\pi r_o^{\,2}kT_{ref}}\exp\left[\frac{-R^2\rho^2}{R^2\rho_o^{\,2}}-S\left(1-\zeta\right)\right]\phi\left(\tau\right)$$

$$\phi(\tau) = \begin{cases} \dfrac{\tau}{\tau_m} & 0 \le \tau \le \tau_m \\ \dfrac{\tau_p - \tau}{\tau_p - \tau_m} & \tau_m < \tau \le \tau_p \\ 0 & \tau_p < \tau \end{cases}$$

$$\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial\theta}{\partial\rho}\right) + a^2\frac{\partial^2\theta}{\partial\zeta^2} + \frac{\dot{q}R^2}{kT_{ref}} = \frac{\partial\theta}{\partial\tau}$$

Thus requiring:

$$T_{ref} = \frac{P\kappa R^2}{\pi r_o^{\,2} k}$$

Substituting in to yield nondimensional equation:

$$\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial\theta}{\partial\rho}\right) + a^2\frac{\partial^2\theta}{\partial\zeta^2} + exp\left[\frac{-\rho^2}{\rho_o^{\,2}} - S(1-\zeta)\right]\phi(\tau) = \frac{\partial\theta}{\partial\tau}$$

Nondimensional boundary conditions:

$$\left.\frac{\partial\theta}{\partial\rho}\right|_{\rho=0} = \left.\frac{\partial\theta}{\partial\rho}\right|_{\rho=1} = 0$$

$$\left.\frac{\partial\theta}{\partial\zeta}\right|_{\zeta=0} = 0$$

$$\left.\frac{\partial T}{\partial\zeta}\right|_{\zeta=1} = -Bi\,\theta(\rho,1,\tau)$$

Using the method of Eigenfunction Expansion:

$$\theta(\rho,\zeta,\tau) = \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} b_{nm}(\tau)R_n(\rho)Z_m(\zeta)$$

Sturm-Louisville Problem in ρ:

$$\frac{1}{\rho}\frac{d}{d\rho}\left(\rho\frac{dR_n}{d\rho}\right)+\lambda_n{}^2R_n=0$$

Boundary condition:

$$\left.\frac{dR_n}{d\rho}\right|_{\rho=0}=\left.\frac{dR_n}{d\rho}\right|_{\rho=1}=0$$

For Neumann condition boundary conditions, use Bessel function for solution:

$$R_n(\rho)=-C_1J_0(\lambda_n\rho)+C_2Y_0(\lambda_n\rho)$$

$$\frac{dR_n}{d\rho}=-C_1\lambda_nJ_1(\lambda_n\rho)-C_2\lambda_nY_1(\lambda_n\rho)$$

Using boundary condition:

$$\left.\frac{dR_n}{d\rho}\right|_{\rho=0}=0=-C_1\lambda_nJ_1(\lambda_n0)-C_2\lambda_nY_1(\lambda_n0)$$

$$J_1(0)=0,\quad Y_1(0)=\infty$$

Thus to satisfy the boundary condition:

$$C_2=0$$

Which returns the Bessel function:

$$R_n(\rho)=-C_1J_0(\lambda_n\rho)$$

With the derivative:

$$\frac{dR_n}{d\rho}=-C_1\lambda_nJ_1(\lambda_n\rho)=0$$

Utilizing the other boundary condition:

$$\left.\frac{dR_n}{d\rho}\right|_{\rho=1}=-C_1\lambda_nJ_1(\lambda_n1)=0$$

Thus λ_n equals the roots of $J_1(\lambda_n) = 0$ starting at $n = 0$ as $\lambda_0 = 0$ and $J_1(0) = 0$. The value of C_1 cannot be zero as that would yield a trivial solution, as it can be any constant, there is no need for the negative sign, as it could be positive or negative nonzero constant.

When $\lambda_0 = 0$, $n = 0$:

$$\frac{1}{\rho}\frac{d}{d\rho}\left(\rho\frac{dR_0}{d\rho}\right)+0^2 R_0 = 0$$

$$\frac{d}{d\rho}\left(\rho\frac{dR_0}{d\rho}\right)=0$$

Integrate both sides in ρ:

$$\int d\left(\rho\frac{dR_0}{d\rho}\right)=\int 0dp$$

$$\rho\frac{dR_0}{d\rho}=C_3 \rightarrow \frac{dR_0}{d\rho}=\frac{C_3}{\rho}$$

Utilize boundary condition:

$$\left.\frac{dR_0}{d\rho}\right|_{\rho=1}=\frac{C_3}{\rho}=0 \rightarrow C_3 = 0 \rightarrow R_0 = C_4$$

Sturm-Louisville Problem in ζ:

$$\frac{d^2 Z_m}{d\zeta^2}+\beta_m{}^2 Z_m = 0$$

Boundary conditions:

$$\left.\frac{dZ_m}{d\zeta}\right|_{\zeta=0}=0, \quad \left.\frac{dZ_m}{d\zeta}\right|_{\zeta=1}=-BiZ_m$$

For Robin conditions, use *sin* and *cos*:

$$Z_m(\zeta)=C_5\, sin(\beta_m\zeta)+C_6\, cos(\beta_m\zeta)$$

$$\frac{dZ_m}{d\zeta} = \beta_m C_5 \, cos(\beta_m \zeta) - \beta_m C_6 \, sin(\beta_m \zeta)$$

Using boundary conditions:

$$\left. \frac{dZ_m}{d\zeta} \right|_{\zeta=0} = 0 = \beta_m C_5 \, cos(\beta_m 0) - \beta_m C_6 \, sin(\beta_m 0)$$

With *cos*(0) = 1, and *sin*(0) = 0:

$$\beta_m C_5 = 0 \rightarrow C_5 = 0 \rightarrow Z_m(\zeta) = C_6 \, cos(\beta_m \zeta)$$

Using the other boundary condition:

$$\left. \frac{dZ_m}{d\zeta} \right|_{\zeta=1} = -BiC_6 \, cos(\beta_m 1) = -\beta_m C_6 \, sin(\beta_m 1)$$

Thus:

$$Bi \, cos(\beta_m) = \beta_m \, sin(\beta_m) \rightarrow Bi = \beta_m \, tan(\beta_m)$$

Which makes the values for β_m the solutions to the equation $Bi = \beta_m \, tan(\beta_m)$. There is no zero eigenvalues in β, as there is no solution where there is a non-zero value for *Bi*. Thus in the infinite summation, the value of *m* is indexed from 1 through infinity rather than from 0 as is *n*.

Now recall that:

$$R_n(\rho) = J_0(\lambda_n \rho)$$

Where λ_n is the roots to the equation $J_1(\lambda_n) = 0$ from *n* = 0 through infinity and:

$$Z_m(\zeta) = cos(\beta_m \zeta)$$

Where β_m is the roots to the equation $Bi = \beta_m \, tan(\beta_m)$ from *m* = 1 through infinity. The value for θ can be evaluated:

$$\theta(\rho,\zeta,\tau)=\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)J_0(\lambda_n\rho)cos(\beta_m\zeta)$$

The value for θ can now be reinserted into the heat equation yielding:

$$\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial}{\partial\rho}\left(\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)J_0(\lambda_n\rho)cos(\beta_m\zeta)\right)\right)$$
$$+a^2\frac{\partial^2}{\partial\zeta^2}\left(\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)J_0(\lambda_n\rho)cos(\beta_m\zeta)\right)+exp\left[\frac{-\rho^2}{\rho_o{}^2}-S(1-\zeta)\right]\phi(\tau)$$
$$=\frac{\partial}{\partial\tau}\left(\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)J_0(\lambda_n\rho)cos(\beta_m\zeta)\right)$$

Which can be simplified:

$$\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)cos(\beta_m\zeta)\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial}{\partial\rho}(J_0(\lambda_n\rho))\right)$$
$$+a^2\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)J_0(\lambda_n\rho)\frac{\partial^2}{\partial\zeta^2}(cos(\beta_m\zeta))+e^{\left(\frac{-\rho^2}{\rho_o{}^2}\right)}e^{(-S(1-\zeta))}\phi(\tau)$$
$$=\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}\frac{\partial b_{nm}}{\partial\tau}J_0(\lambda_n\rho)cos(\beta_m\zeta)$$

Remembering that:

$$\frac{\partial J_0(\lambda_n\rho)}{\partial\rho}=-\lambda_nJ_1(\lambda_n\rho)\rightarrow\frac{1}{\rho}\frac{\partial}{\partial\rho}\left(\rho\frac{\partial}{\partial\rho}(J_0(\lambda_n\rho))\right)=-\lambda_n{}^2J_0(\lambda_n\rho)$$

$$\frac{\partial^2 cos(\beta_m\zeta)}{\partial\zeta^2}=-\beta_m{}^2cos(\beta_m\zeta)$$

Substituting into the previous equation:

$$-\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)cos(\beta_m\zeta)\lambda_n{}^2J_0(\lambda_n\rho)-a^2\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)J_0(\lambda_n\rho)\beta_m{}^2cos(\beta_m\zeta)$$
$$+e^{\left(\frac{-\rho^2}{\rho_o{}^2}\right)}e^{(-S(1-\zeta))}\phi(\tau)=\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}\frac{\partial b_{nm}}{\partial\tau}J_0(\lambda_n\rho)cos(\beta_m\zeta)$$

Multiply all by $\rho J_0(\lambda_q\rho)$ and integrating in ρ from 0 to 1:

$$-\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}b_{nm}(\tau)cos(\beta_m\zeta)\lambda_n^{\,2}\int_0^1\rho J_0(\lambda_q\rho)J_0(\lambda_n\rho)d\rho$$

$$-a^2\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}\beta_m^{\,2}b_{nm}(\tau)cos(\beta_m\zeta)\int_0^1\rho J_0(\lambda_q\rho)J_0(\lambda_n\rho)d\rho$$

$$+e^{(-S(1-\zeta))}\phi(\tau)\int_0^1\rho e^{\left(\frac{-\rho^2}{\rho_o^{\,2}}\right)}J_0(\lambda_q\rho)d\rho=\sum_{n=0}^{\infty}\sum_{m=1}^{\infty}\frac{\partial b_{nm}}{\partial\tau}cos(\beta_m\zeta)\int_0^1\rho J_0(\lambda_q\rho)J_0(\lambda_n\rho)d\rho$$

Using orthogonality:

$$\int_0^1\rho J_0(\lambda_q\rho)J_0(\lambda_n\rho)d\rho=\begin{cases}0 & if\quad n\neq q\\ \dfrac{J_0(\lambda_q)}{2} & if\quad n=q\end{cases}$$

This is proved by using the equation from the SLP in ρ:

$$\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_n\rho)}{d\rho}\right)+\rho\lambda_n^{\,2}J_0(\lambda_n\rho)=0$$

$$\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)+\rho\lambda_q^{\,2}J_0(\lambda_q\rho)=0$$

Subtract the second equation from the first to yield:

$$\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_n\rho)}{d\rho}\right)-\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)=-\left(\lambda_n^{\,2}-\lambda_q^{\,2}\right)\rho J_0(\lambda_n\rho)J_0(\lambda_q\rho)$$

Integrate by ρ from 0 to 1:

$$\int_0^1\rho J_0(\lambda_n\rho)J_0(\lambda_q\rho)d\rho=\frac{1}{\lambda_q^{\,2}-\lambda_n^{\,2}}\left(\int_0^1\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_n\rho)}{d\rho}\right)d\rho-\int_0^1\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)d\rho\right)$$

$$\int_0^1\rho J_0(\lambda_n\rho)J_0(\lambda_q\rho)d\rho=\frac{1}{\lambda_q^{\,2}-\lambda_n^{\,2}}\begin{pmatrix}\rho\dfrac{dJ_0(\lambda_n\rho)}{d\rho}\bigg|_{\rho=1}-\rho\dfrac{dJ_0(\lambda_n\rho)}{d\rho}\bigg|_{\rho=0}\\ -\rho\dfrac{dJ_0(\lambda_q\rho)}{d\rho}\bigg|_{\rho=1}+\rho\dfrac{dJ_0(\lambda_q\rho)}{d\rho}\bigg|_{\rho=0}\end{pmatrix}$$

Which simplifies to:

$$\int_0^1 \rho J_0(\lambda_n\rho)J_0(\lambda_q\rho)d\rho = \frac{1}{\lambda_q^2-\lambda_n^2}\left(\left.\frac{dJ_0(\lambda_n\rho)}{d\rho}\right|_{\rho=1} - \left.\frac{dJ_0(\lambda_q\rho)}{d\rho}\right|_{\rho=1}\right)$$

Again using the identity of $\frac{\partial J_0(\lambda_n\rho)}{\partial\rho} = -\lambda_n J_1(\lambda_n\rho)$ and $\frac{\partial J_0(\lambda_q\rho)}{\partial\rho} = -\lambda_q J_1(\lambda_q\rho)$

$$\int_0^1 \rho J_0(\lambda_n\rho)J_0(\lambda_q\rho)d\rho = \frac{1}{\lambda_q^2-\lambda_n^2}\left(-\lambda_n J_1(\lambda_n\rho)\big|_{\rho=1} + \lambda_q J_1(\lambda_q\rho)\big|_{\rho=1}\right)$$

$$\int_0^1 \rho J_0(\lambda_n\rho)J_0(\lambda_q\rho)d\rho = \frac{1}{\lambda_q^2-\lambda_n^2}\left(-\lambda_n J_1(\lambda_n) + \lambda_q J_1(\lambda_q)\right)$$

From the eigenvalues, $J_1(\lambda_n) = J_1(\lambda_q) = 0$ for all eigenvalues λ from 0 to infinity:

$$\int_0^1 \rho J_0(\lambda_n\rho)J_0(\lambda_q\rho)dp = 0 \quad iff \quad \lambda_n \neq \lambda_q$$

For $\lambda_n = \lambda_q$:

$$\int_0^1 \rho J_0(\lambda_n\rho)J_0(\lambda_q\rho)dp = \int_0^1 \rho J_0^2(\lambda_q\rho)dp$$

Using the previous identity:

$$\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right) + \rho\lambda_q^2 J_0(\lambda_q\rho) = 0$$

With a multiplication factor:

$$\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right) + \rho\lambda_q^2 J_0(\lambda_q\rho) = 0$$

$$\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\frac{d}{d\rho}\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right) + \rho^2\frac{dJ_0(\lambda_q\rho)}{d\rho}\lambda_q^2 J_0(\lambda_q\rho) = 0$$

$$\frac{d}{d\rho}\left[\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)^2\right] = -\lambda_q^2\rho^2\frac{dJ_0^2(\lambda_q\rho)}{d\rho}$$

Integrate in ρ from 0 to 1:

$$\int_0^1 \frac{d}{d\rho}\left[\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)^2\right]d\rho = -\lambda_q^{\ 2}\int_0^1 \rho^2\frac{dJ_0^{\ 2}(\lambda_q\rho)}{d\rho}d\rho$$

Integrate by parts:

$$\begin{aligned} u &= \rho^2 & dv &= \frac{d}{d\rho}\left(J_0^{\ 2}(\lambda_q\rho)\right) \\ du &= 2\rho d\rho & v &= J_0^{\ 2}(\lambda_q\rho) \end{aligned}$$

$$\left[\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)^2\right]_0^1 = -\lambda_q^{\ 2}\left(\left[\rho^2 J_0^{\ 2}(\lambda_q\rho)\right]_0^1 - \int_0^1 2\rho J_0^{\ 2}(\lambda_q\rho)dp\right)$$

$$\left[\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)\right]_0^1 = -\lambda_q^{\ 2}\left(\left[\rho^2 J_0^{\ 2}(\lambda_q\rho)\right]_0^1 - \int_0^1 2\rho J_0^{\ 2}(\lambda_q\rho)dp\right)$$

The left side of the equation is simplified:

$$\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)\Bigg|_0^1 = \left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)\Bigg|_{\rho=1} - \left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)\Bigg|_{\rho=0}$$

$$\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)\Bigg|_0^1 = \left(\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)\Bigg|_{\rho=1}$$

Again using the identity:

$$\frac{dJ_0(\lambda_q\rho)}{d\rho} = -\lambda_q J_1(\lambda_q\rho)$$

$$\left(\rho\frac{dJ_0(\lambda_q\rho)}{d\rho}\right)\Bigg|_0^1 = -\lambda_q J_1(\lambda_q)$$

From the eigenvalues we know $J_1(\lambda_q)$ for all λ_q thus the left hand side of the equation is zero, which of course forces the right hand side to also be zero:

$$0 = -\lambda_q^{\ 2}\left(\left[\rho^2 J_0^{\ 2}(\lambda_q\rho)\right]_0^1 - \int_0^1 2\rho J_0^{\ 2}(\lambda_q\rho)dp\right)$$

Evaluate the first portion:

$$\rho^2 J_0{}^2(\lambda_q\rho)\Big|_0^1 = \rho^2 J_0{}^2(\lambda_q\rho)\Big|_{\rho=1} - \rho^2 J_0{}^2(\lambda_q\rho)\Big|_{\rho=0}$$

$$\rho^2 J_0{}^2(\lambda_q\rho)\Big|_0^1 = J_0{}^2(\lambda_q)$$

Substituting it back into the equation:

$$0 = -\lambda_q{}^2\left(J_0{}^2(\lambda_q) - \int_0^1 2\rho J_0{}^2(\lambda_q\rho)dp\right)$$

Divide by $\lambda_q{}^2$ to simplify:

$$0 = -J_0{}^2(\lambda_q) + 2\int_0^1 \rho J_0{}^2(\lambda_q\rho)dp$$

Which yields for *n* = *q*:

$$\int_0^1 \rho J_0{}^2(\lambda_q\rho)dp = \frac{J_0{}^2(\lambda_q)}{2}$$

This is now input into the heat equation for *n* = *q*:

$$-\frac{\lambda_q{}^2 J_0{}^2(\lambda_q)}{2}\sum_{m=1}^{\infty} b_{qm}(\tau)cos(\beta_m\zeta) - \frac{a^2 J_0{}^2(\lambda_q)}{2}\sum_{m=1}^{\infty}\beta_m{}^2 b_{qm}(\tau)cos(\beta_m\zeta)$$
$$+ e^{(-S(1-\zeta))}\phi(\tau)\int_0^1 \rho e^{\left(\frac{-\rho^2}{\rho_o{}^2}\right)} J_0(\lambda_q\rho)d\rho = \frac{J_0{}^2(\lambda_q)}{2}\sum_{m=1}^{\infty}\frac{\partial b_{qm}}{\partial\tau}cos(\beta_m\zeta)$$

Simplify:

$$\sum_{m=1}^{\infty}\lambda_q{}^2 b_{qm}(\tau)cos(\beta_m\zeta) + \sum_{m=1}^{\infty} a^2\beta_m{}^2 b_{qm}(\tau)cos(\beta_m\zeta) + \sum_{m=1}^{\infty}\frac{\partial b_{qm}}{\partial\tau}cos(\beta_m\zeta)$$
$$= \frac{2e^{(-S(1-\zeta))}\phi(\tau)}{J_0{}^2(\lambda_q)}\int_0^1 \rho e^{\left(\frac{-\rho^2}{\rho_o{}^2}\right)} J_0(\lambda_q\rho)d\rho$$

Simplify to:

$$\sum_{m=1}^{\infty}\left(\frac{\partial b_{qm}}{\partial\tau} + \gamma_{qm} b_{qm}(\tau)\right)cos(\beta_m\zeta) = F_q e^{(-S(1-\zeta))}\phi(\tau)$$

Where:

$$\gamma_{qm} = \lambda_q{}^2 + a^2 \beta_m{}^2$$

$$F_q = \frac{2}{J_0{}^2(\lambda_q)} \int_0^1 \rho e^{\left(\frac{-\rho^2}{\rho_o{}^2}\right)} J_0(\lambda_q \rho) d\rho$$

Using a computationally efficient technique for finding the integral in F_q:

$$I_q = \int_0^1 \rho e^{\left(\frac{-\rho^2}{\rho_o{}^2}\right)} J_0(\lambda_q \rho) d\rho$$

Let:

$$\chi = \frac{\rho}{\rho_o} \rightarrow \rho = \chi \rho_0$$

$$d\chi = \frac{1}{\rho_0} d\rho$$

Making an integral substitution:

$$I_q = \int_0^{\frac{1}{\rho_0}} \rho_0 \chi e^{-\chi^2} J_0(\lambda_q \rho_0 \chi) \rho_0 d\chi$$

Let:

$$\tilde{\lambda}_q = \lambda_q \rho_0$$

Substitute into the integral:

$$I_q = \rho_0{}^2 \int_0^{\frac{1}{\rho_0}} \chi e^{-\chi^2} J_0(\tilde{\lambda}_q \chi) d\chi$$

From integral table, #6631.4

$$\int_0^\infty \chi^{v+1} e^{-\alpha\chi^2} J_v(\beta\chi) d\chi = \frac{\beta^v}{(2\alpha)^{v+1}} exp\left(\frac{-\beta^2}{4\alpha}\right)$$

Thus, if :

$$\rho_0 = {}^{r_0}\!/_{R} << 1 \rightarrow {}^{1}\!/_{\rho_0} \approx \infty$$

The substitution can be made to the integral rather than computing the integral numerically:

$$I_q \approx \rho_0^{\,2} \int_0^{\infty} \chi e^{-\chi^2} J_0\left(\tilde{\lambda}_q \chi\right) d\chi$$

$$I_q \approx \frac{\rho_0^{\,2}}{2} exp\left(\frac{\rho_0^{\,2} \lambda_q^{\,2}}{4}\right)$$

Substituting into F_q:

$$F_q \approx \frac{2}{J_0^{\,2}(\lambda_q)} \frac{\rho_0^{\,2}}{2} exp\left(\frac{\rho_0^{\,2} \lambda_q^{\,2}}{4}\right)$$

Simplify:

$$F_q \approx \frac{\rho_0^{\,2}}{J_0^{\,2}(\lambda_q)} exp\left(\frac{\rho_0^{\,2} \lambda_q^{\,2}}{4}\right)$$

This assumption is good for $\rho_0 < 0.5$. When the value is greater than 0.5, a numerical integration must be used to approximate the integral in F_q.

Evaluating F_0 for use in the equation when $q = 0$, from eigenvalues:

$$\lambda_o = 0$$

$$J_0^{\,2}(0) = 1 \rightarrow J_0(0) = 1$$

Substituted into the exact definition of F_q:

$$F_0 = 2\int_0^1 \rho e^{\frac{-\rho^2}{\rho_0^{\,2}}} d\rho$$

$$F_0 = \rho_0^{\,2} \int_0^1 e^{\frac{-\rho^2}{\rho_0^{\,2}}} \frac{2\rho}{\rho_0^{\,2}} d\rho$$

Using a u substitution:

$$u = \frac{\rho^2}{\rho_0^{\ 2}} \rightarrow du = \frac{2\rho}{\rho_0^{\ 2}} d\rho$$

$$F_0 = \rho_0^{\ 2} \int_0^{\frac{1}{\rho_0^{\ 2}}} e^{-u} du$$

$$F_0 = \rho_0^{\ 2} \left[-e^{-u} \right]_0^{\frac{1}{\rho_0^{\ 2}}}$$

$$F_0 = \rho_0^{\ 2} \left(-e^{\frac{-1}{\rho_0^{\ 2}}} + e^0 \right)$$

$$F_0 = \rho_0^{\ 2} \left(1 - e^{\frac{-1}{\rho_0^{\ 2}}} \right)$$

Returning to the heat equation, we have:

$$\sum_{m=1}^{\infty} \left(\frac{\partial b_{qm}}{\partial \tau} + \gamma_{qm} b_{qm}(\tau) \right) cos(\beta_m \zeta) = F_q e^{(-S(1-\zeta))} \phi(\tau)$$

We use orthogonality to remove the summation in *m* by multiplying by *cos(β_pζ)* and integrating from 0 to 1 in *ζ*.

$$\int_0^1 \sum_{m=1}^{\infty} \left(\frac{\partial b_{qm}}{\partial \tau} + \gamma_{qm} b_{qm}(\tau) \right) cos(\beta_p \zeta) cos(\beta_m \zeta) d\zeta = \int_0^1 F_q \phi(\tau) e^{(-S(1-\zeta))} cos(\beta_p \zeta) d\zeta$$

This can be simplified by rearranging:

$$\sum_{m=1}^{\infty} \left(\frac{\partial b_{qm}}{\partial \tau} + \gamma_{qm} b_{qm}(\tau) \right) \int_0^1 cos(\beta_p \zeta) cos(\beta_m \zeta) d\zeta = F_q \phi(\tau) \int_0^1 e^{(-S(1-\zeta))} cos(\beta_p \zeta) d\zeta$$

Using orthogonality:

$$\int_0^1 cos(\beta_p \zeta) cos(\beta_m \zeta) d\zeta = \begin{cases} 0 & if \quad m \neq p \\ \frac{1}{2} \frac{(1 + Bi(1 + Bi))}{\beta_p^{\ 2}} & if \quad m = p \end{cases}$$

This is proved using identities of the trigonometric *cos*, recall:

$$2cos(\alpha)cos(\beta) = cos(\alpha+\beta) + cos(\alpha-\beta)$$

This applied yields:

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{1}{2}\int_0^1 cos(\beta_m\zeta+\beta_p\zeta)d\zeta + \frac{1}{2}\int_0^1 cos(\beta_m\zeta-\beta_p\zeta)d\zeta$$

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{1}{2}\int_0^1 cos[(\beta_m+\beta_p)\zeta]d\zeta + \frac{1}{2}\int_0^1 cos[(\beta_m-\beta_p)\zeta]d\zeta$$

The integrals are evaluated:

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{1}{2}\left[\frac{sin[(\beta_m+\beta_p)\zeta]}{\beta_m+\beta_p}\right]_0^1 + \frac{1}{2}\left[\frac{sin[(\beta_m-\beta_p)\zeta]}{\beta_m-\beta_p}\right]_0^1$$

$$\begin{aligned}\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = &\frac{1}{2}\left(\frac{sin(\beta_m+\beta_p)}{\beta_m+\beta_p} - \frac{sin[(\beta_m+\beta_p)0]}{\beta_m+\beta_p}\right) + \\ &\frac{1}{2}\left(\frac{sin(\beta_m-\beta_p)}{\beta_m-\beta_p} - \frac{sin[(\beta_m-\beta_p)0]}{\beta_m-\beta_p}\right)\end{aligned}$$

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{1}{2}\left(\frac{sin(\beta_m+\beta_p)}{\beta_m+\beta_p}\right) + \frac{1}{2}\left(\frac{sin(\beta_m-\beta_p)}{\beta_m-\beta_p}\right)$$

This is simplified:

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{1}{2(\beta_m{}^2-\beta_p{}^2)}[(\beta_m-\beta_p)sin(\beta_m+\beta_p) + (\beta_m+\beta_p)sin(\beta_m-\beta_p)]$$

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{1}{2(\beta_m{}^2-\beta_p{}^2)}[(\beta_m-\beta_p)sin(\beta_m+\beta_p) + (\beta_m+\beta_p)sin(\beta_m-\beta_p)]$$

Expanding with *tan*:

$$\int_0^1 cos(\beta_p \zeta)cos(\beta_m \zeta)d\zeta = \frac{1}{2(\beta_m{}^2 - \beta_p{}^2)}\begin{bmatrix} \left(\frac{\beta_m tan(\beta_m)}{tan(\beta_m)} - \frac{\beta_p tan(\beta_p)}{tan(\beta_p)}\right) sin(\beta_m + \beta_p) \\ + \left(\frac{\beta_m tan(\beta_m)}{tan(\beta_m)} + \frac{\beta_p tan(\beta_p)}{tan(\beta_p)}\right) sin(\beta_m - \beta_p) \end{bmatrix}$$

Recall that $\beta_m tan(\beta_m) = Bi$:

$$\int_0^1 cos(\beta_p \zeta)cos(\beta_m \zeta)d\zeta = \frac{1}{2(\beta_m{}^2 - \beta_p{}^2)}\begin{bmatrix} \left(\frac{Bi}{tan(\beta_m)} - \frac{Bi}{tan(\beta_p)}\right) sin(\beta_m + \beta_p) \\ + \left(\frac{Bi}{tan(\beta_m)} + \frac{Bi}{tan(\beta_p)}\right) sin(\beta_m - \beta_p) \end{bmatrix}$$

This is simplified:

$$\int_0^1 cos(\beta_p \zeta)cos(\beta_m \zeta)d\zeta = \frac{Bi}{2(\beta_m{}^2 - \beta_p{}^2)}\begin{bmatrix} \left(\frac{1}{tan(\beta_m)} - \frac{1}{tan(\beta_p)}\right) \begin{pmatrix} sin(\beta_m)cos(\beta_p) \\ + cos(\beta_m)sin(\beta_p) \end{pmatrix} \\ + \left(\frac{1}{tan(\beta_m)} + \frac{1}{tan(\beta_p)}\right) \begin{pmatrix} sin(\beta_m)cos(\beta_p) \\ - cos(\beta_m)sin(\beta_p) \end{pmatrix} \end{bmatrix}$$

Substituting:

$$a = \frac{1}{tan(\beta_m)}$$
$$b = \frac{1}{tan(\beta_p)}$$
$$c = sin(\beta_m)cos(\beta_p)$$
$$d = cos(\beta_m)sin(\beta_p)$$

Simplifies the equation:

$$\int_0^1 cos(\beta_p \zeta)cos(\beta_m \zeta)d\zeta = \frac{Bi}{2(\beta_m{}^2 - \beta_p{}^2)}[(a-b)(c+d)+(a+b)(c-d)]$$

This is expanded and simplified:

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{Bi}{2(\beta_m{}^2-\beta_p{}^2)}[2ac-2bd]$$

Simplifying the expression and reinserting the expressions for *a*, *b*, *c*, and *d*:

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{Bi}{\beta_m{}^2-\beta_p{}^2}\left[\frac{sin(\beta_m)cos(\beta_p)}{tan(\beta_m)}-\frac{cos(\beta_m)sin(\beta_p)}{tan(\beta_p)}\right]$$

Using simple trigonometric identities it is simplified again:

$$\int_0^1 cos(\beta_p\zeta)cos(\beta_m\zeta)d\zeta = \frac{Bi}{\beta_m{}^2-\beta_p{}^2}[cos(\beta_m)cos(\beta_p)-cos(\beta_m)cos(\beta_p)]$$

It is clearly seen that this expression is zero as the two expressions inside the brackets are the same. When they are subtracted, zero is returned and the expression evaluates to zero when $\beta_m \neq \beta_p$.

When $\beta_m = \beta_p$, the base equation is simpler:

$$\int_0^1 cos(\beta_m\zeta)cos(\beta_p\zeta)d\zeta = \int_0^1 cos^2(\beta_p\zeta)d\zeta$$

Recall the trigonometric identity:

$$cos^2(\alpha) = \frac{1}{2}[1+cos(2\alpha)]$$

This is input into the integral:

$$\int_0^1 cos^2(\beta_p\zeta)d\zeta = \int_0^1 \frac{1}{2}[1+cos(2\beta_p\zeta)]d\zeta$$

$$\int_0^1 cos^2(\beta_p\zeta)d\zeta = \frac{1}{2}\left[\zeta+\frac{sin(2\beta_p\zeta)}{2\beta_p}\right]_0^1$$

$$\int_0^1 cos^2(\beta_p\zeta)d\zeta = \frac{1}{2}\left[1+\frac{sin(2\beta_p)}{2\beta_p}-0+\frac{sin(0)}{2\beta_p}\right]$$

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{1}{2}\left(1 + \frac{sin(2\beta_p)}{2\beta_p}\right)$$

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{1}{2}\left(\frac{2\beta_p + sin(2\beta_p)}{2\beta_p}\right)$$

Recall from the half angle formula:

$$sin(2\beta_p) = 2\,cos(\beta_p)sin(\beta_p)$$

Substitute into the previous equation:

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{2\beta_p + 2\,cos(\beta_p)sin(\beta_p)}{4\beta_p}$$

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{\beta_p + cos(\beta_p)sin(\beta_p)}{2\beta_p}$$

Multiply by a $cos(\beta_p)$ / $cos(\beta_p)$:

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{\beta_p + cos(\beta_p)sin(\beta_p)\left(\frac{cos(\beta_p)}{cos(\beta_p)}\right)}{2\beta_p}$$

Simplify:

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{\beta_p + cos^2(\beta_p)tan(\beta_p)}{2\beta_p}$$

Recall:

$$\beta_p\, tan(\beta_p) = Bi$$

Multiply the last term also by β_p / β_p:

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{\beta_p + cos^2(\beta_p)\frac{\beta_p\, tan(\beta_p)}{\beta_p}}{2\beta_p}$$

This can simplify:

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{\beta_p + cos^2(\beta_p)\frac{Bi}{\beta_p}}{2\beta_p}$$

Recalling the trigonometric identity:

$$sin^2(\beta_p) + cos^2(\beta_p) = 1$$

It can be derived that:

$$cos^2(\beta_p) = \frac{1}{1 + tan^2(\beta_p)}$$

Substitute this into the prior equation:

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{\beta_p + \left(\frac{1}{1 + tan^2(\beta_p)}\right)\left(\frac{Bi}{\beta_p}\right)}{2\beta_p}$$

From the definition of *Bi*:

$$\beta_p\, tan(\beta_p) = Bi \to tan(\beta_p) = \frac{Bi}{\beta_p}$$

Substitute into the equation:

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{\beta_p + \frac{1}{1 + Bi^2 / \beta_p{}^2}\left(\frac{Bi}{\beta_p}\right)}{2\beta_p}$$

This simplifies:

$$\int_0^1 cos^2(\beta_p \zeta)d\zeta = \frac{\beta_p\left(1 + \frac{Bi^2}{\beta_p{}^2}\right) + \frac{Bi}{\beta_p}}{2\beta_p}$$

$$\int_0^1 cos^2\left(\beta_p\zeta\right)d\zeta = \frac{\beta_p + \frac{Bi^2}{\beta_p} + \frac{Bi}{\beta_p}}{2\beta_p}$$

$$\int_0^1 cos^2\left(\beta_p\zeta\right)d\zeta = \frac{\beta_p{}^2 + Bi(1+Bi)}{2\beta_p{}^2}$$

Resulting in the end of the value for which is proved:

$$\int_0^1 cos^2\left(\beta_p\zeta\right)d\zeta = \frac{1}{2}\left(1+\frac{Bi(1+Bi)}{\beta_p{}^2}\right) \quad when \quad m = p$$

Using orthogonality on the heat equation:

$$\left(\frac{db_{qp}}{d\tau} + \gamma_{qp}b_{qp}(\tau)\right)\frac{1}{2}\left(1+\frac{Bi(1+Bi)}{\beta_p{}^2}\right) = F_q\phi(\tau)\int_0^1 e^{(-S(1-\zeta))}cos\left(\beta_p\zeta\right)d\zeta$$

$$\left(\frac{db_{qp}}{d\tau} + \gamma_{qp}b_{qp}(\tau)\right)\frac{1}{2}\left(1+\frac{Bi(1+Bi)}{\beta_p{}^2}\right) = F_q\phi(\tau)e^{(-S)}\int_0^1 e^{(S\zeta)}cos\left(\beta_p\zeta\right)d\zeta$$

Evaluating the integral by parts:

$$u = e^{(S\zeta)} \qquad dv = cos\left(\beta_p\zeta\right)d\zeta$$
$$du = Se^{(S\zeta)}d\zeta \qquad v = \frac{sin\left(\beta_p\zeta\right)}{\beta_p}$$

$$\int_0^1 e^{(S\zeta)}cos\left(\beta_p\zeta\right)d\zeta = \left[\frac{e^{(S\zeta)}sin\left(\beta_p\zeta\right)}{\beta_p}\right]_0^1 - \int_0^1 \frac{Se^{(S\zeta)}sin\left(\beta_p\zeta\right)}{\beta_p}d\zeta$$

Yields the equation:

$$\int_0^1 e^{(S\zeta)}cos\left(\beta_p\zeta\right)d\zeta = \frac{e^S sin\left(\beta_p\right)}{\beta_p} - e^{(S\cdot 0)}\frac{sin\left(\beta_p\cdot 0\right)}{\beta_p} - \frac{S}{\beta_p}\int_0^1 \frac{e^{(S\zeta)}sin\left(\beta_p\zeta\right)}{\beta_p}d\zeta$$

Which simplifies to:

$$\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta = \frac{e^S\, sin(\beta_p)}{\beta_p} - \frac{S}{\beta_p}\int_0^1 \frac{e^{(S\zeta)} sin(\beta_p\zeta)}{\beta_p}d\zeta$$

Integration by parts can be done on the integral using:

$$u = e^{(S\zeta)} \qquad dv = sin(\beta_p\zeta)d\zeta$$
$$du = Se^{(S\zeta)}d\zeta \qquad v = \frac{-cos(\beta_p\zeta)}{\beta_p}$$

$$\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta = \frac{e^S\, sin(\beta_p)}{\beta_p} - \frac{S}{\beta_p}\left(\left[e^{S\zeta}\left(\frac{-cos(\beta_p\zeta)}{\beta_p}\right)\right]_0^1 + \frac{S}{\beta_p}\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta\right)$$

Evaluating and simplifying to:

$$\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta = \frac{e^S\, sin(\beta_p)}{\beta_p} + \frac{S}{\beta_p{}^2}\left(e^S cos(\beta_p) - e^0 cos(0)\right) + \frac{S^2}{\beta_p{}^2}\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta$$

$$\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta = \frac{e^S\, sin(\beta_p)}{\beta_p} + \frac{S}{\beta_p{}^2}\left(e^S cos(\beta_p) - 1\right) + \frac{S^2}{\beta_p{}^2}\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta$$

$$\left(\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta\right)\left(1 + \frac{S^2}{\beta_p{}^2}\right) = \frac{e^S\, sin(\beta_p)}{\beta_p} + \frac{S}{\beta_p{}^2}\left(e^S cos(\beta_p) - 1\right)$$

$$\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta = \left(1 \Big/ 1 + \frac{S^2}{\beta_p{}^2}\right)\left(\frac{e^S\, sin(\beta_p)}{\beta_p} + \frac{S}{\beta_p{}^2}\left(e^S cos(\beta_p) - 1\right)\right)$$

$$\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta = \left(1 \Big/ 1 + \frac{S^2}{\beta_p{}^2}\right)\left(\frac{\beta_p e^S\, sin(\beta_p) + Se^S cos(\beta_p) - S}{\beta_p{}^2}\right)$$

$$\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta = \left(\frac{1}{\beta_p{}^2 + S^2}\right)\left(\beta_p e^S\, sin(\beta_p) + Se^S cos(\beta_p) - S\right)$$

$$\int_0^1 e^{(S\zeta)} cos(\beta_p\zeta)d\zeta = \left(\frac{1}{\beta_p{}^2 + S^2}\right)\left(e^S\left(\beta_p\, sin(\beta_p) + S\, cos(\beta_p)\right) - S\right)$$

Recall:

$$\beta_p \, tan(\beta_p) = Bi \quad \Rightarrow \beta_p = \frac{Bi}{tan(\beta_p)}$$

Substitute into previous equation and simplify:

$$\int_0^1 e^{(S\zeta)} \, cos(\beta_p \zeta) d\zeta = \left(\frac{1}{\beta_p{}^2 + S^2} \right) \left(e^S \left(\frac{Bi}{tan(\beta_p)} sin(\beta_p) + S \, cos(\beta_p) \right) - S \right)$$

$$\int_0^1 e^{(S\zeta)} \, cos(\beta_p \zeta) d\zeta = \left(\frac{1}{\beta_p{}^2 + S^2} \right) \left(e^S \left(Bi \; cos(\beta_p) + S \, cos(\beta_p) \right) - S \right)$$

$$\int_0^1 e^{(S\zeta)} \, cos(\beta_p \zeta) d\zeta = \left(\frac{1}{\beta_p{}^2 + S^2} \right) \left(e^S \, cos(\beta_p)(Bi + S) - S \right)$$

$$\int_0^1 e^{(S\zeta)} \, cos(\beta_p \zeta) d\zeta = \frac{e^S \, cos(\beta_p)(Bi + S) - S}{\beta_p{}^2 + S^2}$$

Using the solution to the integral:

$$e^{-S} \left(\int_0^1 e^{(S\zeta)} \, cos(\beta_p \zeta) d\zeta \right) = e^{-S} \left(\frac{e^S \, cos(\beta_p)(Bi + S) - S}{\beta_p{}^2 + S^2} \right)$$

$$e^{-S} \left(\int_0^1 e^{(S\zeta)} \, cos(\beta_p \zeta) d\zeta \right) = \frac{cos(\beta_p)(Bi + S) - Se^{-S}}{\beta_p{}^2 + S^2}$$

Substitute into the base heat equation:

$$\left(\frac{\partial b_{qp}}{\partial \tau} + \gamma_{qp} b_{qp}(\tau) \right) \frac{1}{2} \left(1 + \frac{Bi(1 + Bi)}{\beta_p{}^2} \right) = F_q \phi(\tau) \frac{cos(\beta_p)(Bi + S) - Se^{-S}}{\beta_p{}^2 + S^2}$$

The second half of the left hand side of the equation can be simplified:

$$\frac{1}{2} \left(1 + \frac{Bi(1 + Bi)}{\beta_p{}^2} \right) = \frac{1}{2} \left(\frac{\beta_p{}^2}{\beta_p{}^2} + \frac{Bi(1 + Bi)}{\beta_p{}^2} \right)$$

$$\frac{1}{2}\left(1+\frac{Bi(1+Bi)}{\beta_p{}^2}\right)=\frac{1}{2}\left(\frac{\beta_p{}^2+Bi(1+Bi)}{\beta_p{}^2}\right)$$

$$\frac{1}{2}\left(1+\frac{Bi(1+Bi)}{\beta_p{}^2}\right)=\frac{\beta_p{}^2+Bi(1+Bi)}{2\beta_p{}^2}$$

Substituting back into the base heat equation and simplifying:

$$\left(\frac{\partial b_{qp}}{\partial\tau}+\gamma_{qp}b_{qp}(\tau)\right)\left(\frac{\beta_p{}^2+Bi(1+Bi)}{2\beta_p{}^2}\right)=F_q\phi(\tau)\frac{cos(\beta_p)(Bi+S)-Se^{-S}}{\beta_p{}^2+S^2}$$

$$\frac{db_{qp}}{d\tau}+\gamma_{qp}b_{qp}(\tau)=\frac{2\beta_p{}^2}{\beta_p{}^2+Bi(1+Bi)}F_q\phi(\tau)\frac{cos(\beta_p)(Bi+S)-Se^{-S}}{\beta_p{}^2+S^2}$$

$$\frac{db_{qp}}{d\tau}+\gamma_{qp}b_{qp}(\tau)=\frac{2}{1+\left(S/\beta_p\right)^2}\left(\frac{(Bi+S)cos(\beta_p)-Se^{-S}}{Bi(1+Bi)+\beta_p{}^2}\right)F_q\phi(\tau)$$

Let:

$$G_p=\frac{2}{1+\left(S/\beta_p\right)^2}\left(\frac{(Bi+S)cos(\beta_p)-Se^{-S}}{Bi(1+Bi)+\beta_p{}^2}\right)$$

Then:

$$\frac{db_{qp}}{d\tau}+\gamma_{qp}b_{qp}(\tau)=F_qG_p\phi(\tau)$$

Solve for $b_{qp}(\tau)$ using the initial conditions:

$$\theta(\rho,\zeta,0)=0 \quad if \quad b_{qp}(0)=0$$

Use the integration factor $e^{\gamma_{qp}\tau}$:

$$e^{\gamma_{qp}\tau}\left(\frac{\partial b_{qp}}{\partial\tau}+\gamma_{qp}b_{qp}(\tau)\right)=e^{\gamma_{qp}\tau}F_qG_p\phi(\tau)$$

$$\int_0^{\tau}\frac{d}{d\tau'}\left(e^{\gamma_{qp}\tau'}b_{qp}\right)d\tau' = F_q G_p \int_0^{\tau} e^{\gamma_{qp}\tau'}\phi(\tau')d\tau'$$

$$e^{\gamma_{qp}\tau}b_{qp}(\tau) - e^{\gamma_{qp}0}b_{qp}(0) = F_q G_p \int_0^{\tau} e^{\gamma_{qp}\tau'}\phi(\tau')d\tau'$$

$$e^{\gamma_{qp}\tau}b_{qp}(\tau) = F_q G_p \int_0^{\tau} e^{\gamma_{qp}\tau'}\phi(\tau')d\tau'$$

$$b_{qp}(\tau) = F_q G_p e^{-\gamma_{qp}\tau}\int_0^{\tau} e^{\gamma_{qp}\tau'}\phi(\tau')d\tau'$$

Checking the initial condition by substituting a small value, ε for τ and taking the limit as ε goes to zero:

$$b_{qp}(\varepsilon) = F_q G_p e^{-\gamma_{qp}\varepsilon}\int_0^{\varepsilon} e^{\gamma_{qp}\tau'}\phi(\tau')d\tau'$$

$$lim_{\varepsilon\to 0}\, b_{qp} = F_q G_p e^{-\gamma_{qp}\varepsilon}\int_0^{\varepsilon} e^{\gamma_{qp}\tau'}\phi(\tau')d\tau' = 0$$

The initial condition is satisfied as when ε goes to zero, the integral goes to zero and the expression before the integral goes to F_qG_p as the exponential goes to one.

Now, let:

$$\mathrm{T}_{qp}(\tau) = e^{-\gamma_{qp}\tau}\int_0^{\tau} e^{\gamma_{qp}\tau'}\phi(\tau')d\tau'$$

Remembering that $\phi(\tau)$ varies in time, $\mathrm{T}_{qp}(\tau)$ will be solved in sections, for the first time segment, $0 \le \tau \le \tau_m$:

$$\phi(\tau) = \frac{\tau}{\tau_m}$$

Substituting into the equation and simplifying:

$$\mathrm{T}_{qp}^{\mathrm{A}}(\tau) = e^{-\gamma_{qp}\tau}\int_0^{\tau} e^{\gamma_{qp}\tau'}\frac{\tau'}{\tau_m}d\tau'$$

Using Maple 12 to evaluate the integral returns:

$$\mathrm{T}_{qp}^{\mathrm{A}}(\tau) = \frac{e^{-\gamma_{qp}\tau}}{\tau_m \gamma_{qp}^{2}}\left(1 - e^{\gamma_{qp}\tau} + \tau\gamma_{qp} e^{\gamma_{qp}\tau}\right)$$

Continuing with the second segment of $\mathrm{T}_{qp}(\tau)$ where $\tau_m < \tau \le \tau_p$:

$$\phi(\tau) = \frac{\tau_p - \tau}{\tau_p - \tau_m}$$

$$\mathrm{T}_{qp}^{\mathrm{B}}(\tau) = e^{-\gamma_{qp}\tau}\left[\int_0^{\tau_m} e^{\gamma_{qp}\tau'} \frac{\tau'}{\tau_m} d\tau' + \int_{\tau_m}^{\tau} e^{\gamma_{qp}\tau'} \frac{\tau_p - \tau'}{\tau_p - \tau_m} d\tau'\right]$$

Using Maple 12 to evaluate the integral for the second segment returns:

$$\mathrm{T}_{qp}^{\mathrm{B}}(\tau) = \frac{e^{-\gamma_{qp}\tau}}{\tau_m \gamma_{qp}^{2}(\tau_m - \tau_p)}\left(\begin{array}{l}\tau_m - \tau_p + \tau_p e^{\gamma_{qp}\tau_m} - \tau_p \tau_m \gamma_{qp} e^{\gamma_{qp}\tau} \\ -\tau_m e^{\gamma_{qp}\tau} + \tau_m \tau \gamma_{qp} e^{\gamma_{qp}\tau}\end{array}\right)$$

Finishing with the third segment of $\mathrm{T}_{qp}(\tau)$ where $\tau > \tau_p$:

$$\phi(\tau) = 0$$

$$\mathrm{T}_{qp}^{\mathrm{C}}(\tau) = e^{-\gamma_{qp}\tau}\left[\int_0^{\tau_m} \frac{\tau'}{\tau_m} e^{\gamma_{qp}\tau'} d\tau' + \int_{\tau_m}^{\tau_p} \frac{\tau_p - \tau'}{\tau_p - \tau_m} e^{\gamma_{qp}\tau'} d\tau'\right]$$

Using Maple 12 to evaluate the integral for the third segment returns:

$$\mathrm{T}_{qp}^{\mathrm{C}}(\tau) = \frac{e^{-\gamma_{qp}\tau}}{\tau_m \gamma_{qp}^{2}(\tau_p - \tau_m)}\left(\tau_p - \tau_m - \tau_p e^{\gamma_{qp}\tau_m} + \tau_m e^{\gamma_{qp}\tau_p}\right)$$

The value of $\mathrm{T}_{qp}(\tau)$ is solved for all values of τ:

$$
\mathrm{T}_{qp}(\tau)=\begin{cases}
\dfrac{e^{-\gamma_{qp}\tau}}{\tau_m{\gamma_{qp}}^2}\left(1-e^{\gamma_{qp}\tau}+\tau\gamma_{qp}e^{\gamma_{qp}\tau}\right) & 0\le\tau\le\tau_m \\
\dfrac{e^{-\gamma_{qp}\tau}}{\tau_m{\gamma_{qp}}^2\left(\tau_m-\tau_p\right)}\begin{pmatrix}\tau_m-\tau_p+\tau_p e^{\gamma_{qp}\tau_m}-\tau_p\tau_m\gamma_{qp}e^{\gamma_{qp}\tau} \\ -\tau_m e^{\gamma_{qp}\tau}+\tau_m\tau\gamma_{qp}e^{\gamma_{qp}\tau}\end{pmatrix} & \tau_m<\tau\le\tau_p \\
\dfrac{e^{-\gamma_{qp}\tau}}{\tau_m{\gamma_{qp}}^2\left(\tau_p-\tau_m\right)}\left(\tau_p-\tau_m-\tau_p e^{\gamma_{qp}\tau_m}+\tau_m e^{\gamma_{qp}\tau_p}\right) & \tau>\tau_p
\end{cases}
$$

The equation is complete once all the segments of the function T are completed for all the segments. The complete solution of the nondimensional temperature in the sample is a combination of all the equations:

$$\theta(\rho,\xi,\tau)=\sum_{q=1}^{\infty}\sum_{p=0}^{\infty}b_{qp}(\tau)cos(\beta_q\xi)J_0(\lambda_p\rho)$$

$$\beta_q\, tan(\beta_q)=Bi$$

$$J_1(\lambda_p)=0$$

$$b_{qp}(\tau)=F_pG_q\mathrm{T}_{qp}(\tau)$$

$$F_p=\frac{2}{J_0{}^2(\lambda_p)}\int_0^1\rho e^{-\rho^2/\rho_o{}^2}J_0(\lambda_p\rho)d\rho$$

$$G_q=\frac{2}{1+\left(S/\beta_q\right)^2}\frac{(Bi+S)cos(\beta_q)-Se^{-S}}{Bi(Bi+1)+\beta_q{}^2}$$

$$\mathrm{T}_{qp}(\tau)=\begin{cases}\dfrac{e^{-\gamma_{qp}\tau}}{\tau_m\gamma_{qp}{}^2}\left(1-e^{\gamma_{qp}\tau}+\tau\gamma_{qp}e^{\gamma_{qp}\tau}\right) & 0\le\tau\le\tau_m\\ \dfrac{e^{-\gamma_{qp}\tau}}{\tau_m\gamma_{qp}{}^2(\tau_m-\tau_p)}\begin{pmatrix}\tau_m-\tau_p+\tau_pe^{\gamma_{qp}\tau_m}-\tau_p\tau_m\gamma_{qp}e^{\gamma_{qp}\tau}\\-\tau_me^{\gamma_{qp}\tau}+\tau_m\tau\gamma_{qp}e^{\gamma_{qp}\tau}\end{pmatrix} & \tau_m<\tau\le\tau_p\\ \dfrac{e^{-\gamma_{qp}\tau}}{\tau_m\gamma_{qp}{}^2(\tau_p-\tau_m)}\left(\tau_p-\tau_m-\tau_pe^{\gamma_{qp}\tau_m}+\tau_me^{\gamma_{qp}\tau_p}\right) & \tau>\tau_p\end{cases}$$

$$\gamma_{qp}=\lambda_p{}^2+a^2\beta_q{}^2$$

$$Bi=\frac{hL}{k}=\frac{hL}{\rho_mc\alpha}$$

$$S=\kappa L,\ \rho={}^{r}\!/_{L},\ \xi={}^{z}\!/_{L},\ \tau={}^{\alpha t}\!/_{R^2},\ \rho_o={}^{r_o}\!/_{R},\ a={}^{R}\!/_{L}$$

$$\theta=(T-T_\infty)\frac{\pi r_o{}^2k}{P\kappa R^2}$$

APPENDIX C. SOURCE CODE FOR CREATION OF THE GOVERNING PARAMETER SETS FOR THE DSFA THERMAL MODEL

Source code file:	DSFA_code_builder.m
Compile instructions:	Ensure all the required functions are located in the same working directory as the program. Once the file is loaded in the editor window, simply click 'run' or type the name of the file in the command window.
User inputs:	None
Program input files:	None
Required program functions:	Matlab programmed functions bessel.m, trigfunc.m, Fint.m need to be saved in a directory that the current path in Matlab has defined.
Program output files:	DSFA_630.xls. Values of the time dependent nondimensional temperature profiles for each of the given parameter sets along with their respective parameter values.

Program source code:

```
% DSFA_code_builder.m

% This code is used to solve the non-dimensional temperature profiles
% in time for a small cylindrical sample that is irradiated upon by a
% laser with a Gaussian power distribution.  The time varying pulse is
% a triangle wave starting at zero, increasing linearly to a time, t_m,
% and then decreasing linearly back to zero at t_p.

clear all; clc;

tic;

% Set the governing parameters for 630 parameter sets

% Change the values of alpha and rhoc to match metals
alp_cas = [1e-4, 3e-5, 1e-5, 3e-6, 1e-6];
rhoc_cas = [0.5e6, 1.5e6, 2.5e6, 3.5e6, 4.5e6, 5.5e6];
h_cas = [20,30,40];
Kap_cas = [400,1000,1600, 2200, 2800, 3400, 4000];

size_Kap=size(Kap_cas,2);
size_alp=size(alp_cas,2);
size_rhoc=size(rhoc_cas,2);
size_h=size(h_cas,2);

cases=size_Kap*size_alp*size_rhoc*size_h;

max_alp=max(alp_cas);
min_alp=min(alp_cas);
max_rhoc=max(rhoc_cas);
min_rhoc=min(rhoc_cas);
max_h=max(h_cas);
min_h=min(h_cas);
max_Kap=max(Kap_cas);
min_Kap=min(Kap_cas);

P = 1000.0;          % Watts, power of the laser
r_o = 0.001;         % m, 1 sigma of the Gaussian curve away from center
                     %    of laser
R = 0.01;            % m, radius of the sample (should be >= 3*r_o)
L = 0.002;           % m, thickness of the sample
Tinf = 300.0;        % K, initial temp and temp of surroundings
t_m = 0.001;         % sec, time for pulse to get to peak power
t_p = 0.003;         % sec, time for pulse to finish
rho=0.0;             % m, location of center of pulse from center of
                     %    sample

% Define non-dimensional parameters

a = R / L;           % Aspect ratio
rho_o = r_o / R;     % Ratio of laser power curve to radius of sample

count = 1;
```

```
for conv=1:size_h
    for OpDep=1:size_Kap
        for alph=1:size_alp
            for rhocp=1:size_rhoc

h = h_cas(conv);
Kap = Kap_cas(OpDep);
alpha = alp_cas(alph);
heatcap = rhoc_cas(rhocp);
k = alpha * heatcap;

tau_m = t_m * alpha / R^2;
tau_p = t_p * alpha / R^2;

S = Kap * L;
Bi = h * L / k;

dimpar(count,:) = [h, Kap, alpha, heatcap, k];

Par(count,:) = [S, Bi, tau_m];
Check(count,:) = S * Bi * tau_m;

N=10;  % number of terms

% Calculate the Eigenvalues
% Eigenvalues in Lambda
lam(1)=0;
for n=2:N
    format long
    con=lam(n-1)+3;
    lam(n) = fzero(@bessel, con);
end

% Eigenvalues in Beta
beta(1)= fzero(@(x) trigfunc(x,Bi), 3e-4);
if beta(1)<0
    beta(1)=-beta(1);
end
for m=2:N
    format long
    con=beta(m-1)+3;
    beta(m)= fzero(@(x) trigfunc(x,Bi), con);
end

% Calculate Non-Dimensional Temperature at any non-dimensional location
% in non-dimensional time
for i=1:200
    tau = ( i - 1 ) * 0.0001;        % non-dimensional time
    tau_ar(count,i) = tau;
    time_ar(count,i) = R ^ 2 * tau / alpha;
    zeta = 0;
    zet = 1;
    Theta(zet,i)=0;
    for n=1:10
        % Evaluate Bessel J at each eigenvalue
        Jo(n)=besselj(0,lam(n));
        Jop(n)=besselj(0,lam(n)*rho);
        for m=1:N
            gam=lam(n)^2+a^2*beta(m)^2;
            if tau <= tau_m
                T=1/(gam*tau_m)*(tau-(1-exp(-gam*tau))/gam);
            elseif tau <= tau_p
                T=1/gam^2*((gam*(tau_p-tau)+1)/(tau_p-tau_m)+(...
                    exp(-gam*tau)-exp(gam*(tau_m-tau)))/tau_m-...
                    (exp(gam*(tau_m-tau)))/(tau_p-tau_m));
            else
                T=1/gam^2*((exp(-gam*tau)-exp(gam*(tau_m-tau)))/...
                    tau_m+(exp(gam*(tau_p-tau))-exp(gam*(tau_m-...
                    tau)))/(tau_p-tau_m));
            end
            if rho_o < 0.5
                F=rho_o^2/Jo(n)^2*exp(-rho_o^2*lam(n)^2/4);
            else
                F=2*Fint(rho_o,lam(n))/Jo(n)^2;
            end
            G=2*(beta(m)^2+Bi^2)*(cos(beta(m))*(Bi+S)-S*exp(-S))/...
                (beta(m)^2+Bi^2+Bi)/(beta(m)^2+S^2);
            GM(n,m)=G;
            b=F*G*T;
            Fnm(n,m)=F;
            Theta(zet,i)=Theta(zet,i)+b*cos(beta(m)*zeta)*Jop(n);
            Time(i)=tau*R^2/alpha;
            TAU(i)=tau;
        end
    end
end
Tref = P * Kap * R^2 /( pi * r_o^2 * k);
Temp(count,:) = Theta * Tref + Tinf;
THETA(count,:) = Theta;
t_end(count,:) = toc;
count = count + 1;

            end
        end
    end
end

time_end = toc;

% Write the variables to file DFSA_630.xls for use in ROM and GA
xlswrite('DSFA_630',Par,'Sheet1','A4')
xlswrite('DSFA_630',THETA,'Sheet1','D4')
xlswrite('DSFA_630',t_end,'Sheet2','A1')
xlswrite('DSFA_630',time_end,'Sheet3','A1')
xlswrite('DSFA_630',tau_ar(1,:),'Sheet1','D3')
xlswrite('DSFA_630',Temp,'Sheet4','A1')
xlswrite('DSFA_630',time_ar,'Sheet5','A1')
xlswrite('DSFA_630',r_o,'Vars','A5')
xlswrite('DSFA_630',R,'Vars','A6')
xlswrite('DSFA_630',L,'Vars','A7')
xlswrite('DSFA_630',t_p,'Vars','A8')
xlswrite('DSFA_630',t_m,'Vars','A9')
xlswrite('DSFA_630',P,'Vars','A10')
xlswrite('DSFA_630',Tinf,'Vars','A11')
xlswrite('DSFA_630',tau_ar(1,2),'Vars','A12')
xlswrite('DSFA_630',max_alp,'Vars','A1')
xlswrite('DSFA_630',min_alp,'Vars','B1')
xlswrite('DSFA_630',max_rhoc,'Vars','A2')
xlswrite('DSFA_630',min_rhoc,'Vars','B2')
xlswrite('DSFA_630',max_h,'Vars','A3')
xlswrite('DSFA_630',min_h,'Vars','B3')
xlswrite('DSFA_630',max_Kap,'Vars','A4')
xlswrite('DSFA_630',min_Kap,'Vars','B4')
xlswrite('DSFA_630',dimpar,'dimpar','A1')
% end 630 parameter case
```

Required functions source code:

bessel.m

```
% bessel.m

function J1 = f(Z);
J1 = besselj(1,Z);
```

trigfunc.m

```
% trigfunc.m

function trig = f(Z,Bi);
trig = Z*tan(Z)-Bi;
```

Fint.m

```
% Fint.m

function F=f(po,lam);
val=0;
num=100;
H=0;
for rhoi = 1:num
    p=rhoi/num;
    H=p*exp(-p^2/po^2)*besselj(0,lam*p);
    val=val+H;
end
F=val;
```

APPENDIX D. SOURCE CODE FOR THE DSFA MODEL FOR THE MATRIX CREATION OF THE ROM IN MATLAB

Source code file:	ROM_code_matrix.m
Compile instructions:	Ensure the Excel file is saved in the same working directory that Matlab is in. Once the file is loaded in the editor window, simply click ‘run’ or type the name of the file in the command window.
User Inputs:	None
Program input files:	DSFA_630.xls. Raw solutions from the coefficient builder program for each parameter set.
Required program functions:	None
Program output files:	DSFA_630.xls. Values of the time dependent nondimensional temperature profiles for each of the given parameter sets along with their respective parameter values. Contains the new data added to the file from this program.

Program source code:

```
% ROM_code_matrix.m

% Reduced Order Modeling program for the calculation of time dependant
% temperature profiles in a Pulsed Laser Diffusion test.
% Opens Microsoft Excel file DSFA_630.xls which contains the basis set
% of parameters and solved values.
% Reads 'DSFA_630' matrix from with database of governing parameters
% and their respective nondimensional time dependant profiles from
% 0.0 to 0.0199 Fourier numbers stepping in 0.0001 Fourier numbers

% Writes to DSFA_630.xls other needed parameters and sets that are
% stored and read in by the program that accepts arbitrary parameters
% and returns the arbitrary profile.

clear all; clc;
tic;                 % clock start

% Input matrix A from DSFA - Ensure that the range is correct

a=xlsread('DSFA_630', 'Sheet1', 'D4:GU633');

A=a';

[M,N]=size(A);

% Read parameters used to create the A matrix

par=xlsread('DSFA_630', 'Sheet1', 'A4:C633');

% Ensure that the range is correct
k = par';
kplus = pinv(k);

% Set values from read in values for use in ROM
S=par(:,1);
Bi=par(:,2);
tau_m=par(:,3);
maxS=max(S);
minS=min(S);
maxBi=max(Bi);
minBi=min(Bi);
maxtau_m=max(tau_m);
mintau_m=min(tau_m);

% For singularity values
tol=1e-13;

% Build matrix of interpolation functions
for i=1:N
    for j=1:N
        Svar(i,j)=((par(i,1)-par(j,1))/(maxS))^2;
        Bivar(i,j)=((log10(par(i,2))-log10(par(j,2)))/log10(maxBi))^2;
        tau_mvar(i,j)=((log10(par(i,3))-log10(par(j,3)))/...
            log10(maxtau_m))^2;
        F(i,j)=1/(Svar(i,j)+Bivar(i,j)+tau_mvar(i,j)+1)^(1/2);
    end
end

% Number of significant eigenvalues
n_fe=25;

% Singular value decomposition of A
[U_A SIG_A V_A]=svd(A);

fe=U_A(:,1:n_fe);

% Calculate coefficient matrix
B=fe'*A;
Bp=B';
festar=B*kplus;

% Perform Singular Value Decomposition on F to get Moore-Penrose
% inverse
[U_F SIG_F V_F]=svd(F);
Sa=SIG_F;

% Zero out small singular values
for i=1:N
    if SIG_F(i,i) > tol
        S_F(i,i)=1/SIG_F(i,i);
    else
        S_F(i,i)=0;
    end
end

% Get C matrix using singular value decomposition
C=B*V_F*S_F*U_F';

Fplus=V_F*S_F*U_F';

s=diag(SIG_A);
f=diag(SIG_F);

V = min(M,N);

% Plot the Singular values of A for determination of truncation
figure;%(2)
semilogy (s(1:V,1),'o')
xlabel('Number')
ylabel('Singular Value of A')

% Plot the singular values of F for determination of truncation
figure;%(3)
semilogy (f(1:N,1),'-')
xlabel('Number')
ylabel('Singular Value of F')

Cp=C';

% Write parameters to Excel to be used in the GA program
warning off MATLAB:xlswrite:AddSheet
```

```
xlswrite('DSFA_630', Cp, 'C', 'A1')         % Ensure this matrix call
xlswrite('DSFA_630', fe, 'fe', 'A1')        % is the same as above that
xlswrite('DSFA_630', Bp, 'B', 'A1')         % the matrix is read from
xlswrite('DSFA_630', festar, 'festar', 'A1')
xlswrite('DSFA_630', maxS, 'Vars', 'C1')
xlswrite('DSFA_630', minS, 'Vars', 'D1')
xlswrite('DSFA_630', maxBi, 'Vars', 'C2')
xlswrite('DSFA_630', minBi, 'Vars', 'D2')
xlswrite('DSFA_630', maxtau_m, 'Vars', 'C3')
xlswrite('DSFA_630', mintau_m, 'Vars', 'D3')
```

APPENDIX E. SOURCE CODE FOR THE INPUT OF ARBITRARTY PARAMETERS FOR THE ROM IN MATLAB

Source code file:	ROM_code_AP.m
Compile instructions:	Ensure the Excel file is saved in the same working directory that Matlab is in. Once the file is loaded in the editor window, simply click ‘run’ or type the name of the file in the command window.
User inputs:	Input the arbitrary parameters for the ROM in the code. For comparison to the input values, input the correct file and rows to retrieve the values for ‘w’ and ‘tim’ so the profiles may be plotted together.
Program input files:	DSFA_630.xls. Raw solutions from the coefficient builder program for each parameter set and the additions from the ROM_code_matrix.m program.
Required program functions:	None
Program output files:	None
Program source code:	

```
% ROM_code_AP.m

% Reduced Order Modeling simulation for the calculation time dependant
% temperature profiles in a Pulsed Laser Diffusion experiment
% Input Arbitrary parameters, read the Excel file with matrix data
% stored
% Returns Temperature, As.
% With correlating data from the Excel file, will plot the ROM data
% versus the numerical data from PLD Code.

clear all; clc;

tic;            % Start time clock
t1=toc;

% Read in Excel files that contain the ROM data
% File A.xls
Cp=xlsread('DSFA_630', 'C', 'A1:Y630');        % Ensure that the
par=xlsread('DSFA_630', 'Sheet1', 'A4:C633');  % range of all these
maxS=xlsread('DSFA_630', 'Vars', 'C1');        % read in matrices
maxBi=xlsread('DSFA_630', 'Vars', 'C2');       % match the actual
maxtau_m=xlsread('DSFA_630', 'Vars', 'C3');    % ranges in the Excel
fe=xlsread('DSFA_630', 'fe', 'A1:Y200');       % files
festar=xlsread('DSFA_630', 'festar', 'A1:C25');
Bp=xlsread('DSFA_630', 'B', 'A1:Y630');
r_o=xlsread('DSFA_630', 'Vars', 'A5');
R=xlsread('DSFA_630', 'Vars', 'A6');
L=xlsread('DSFA_630', 'Vars', 'A7');
t_p=xlsread('DSFA_630', 'Vars', 'A8');
t_m=xlsread('DSFA_630', 'Vars', 'A9');
P=xlsread('DSFA_630', 'Vars', 'A10');
Tinf=xlsread('DSFA_630', 'Vars', 'A11');

B=Bp';
C=Cp';
[i,M]=size(C);
[N,i]=size(fe);

t2=toc;

%Arbitrary Profile

% Manually input arbitrary conditions
alphaa=3.959e-6;
rho_cpa=5.052e6;
ha=37;
Kapa=960.5;

t2 = toc;

con=alphaa*rho_cpa;

% Non-Dimensionalize
Sa=Kapa*L;
Bia=ha*L/(alphaa*rho_cpa);
tau_ma=alphaa*t_m/R^2;
Pa=[Sa Bia tau_ma];
```

```
%interpolation function
for i=1:M
    Svara(i)=((Pa(1)-par(i,1))/maxS)^2;
    Bivara(i)=((log10(Pa(2))-log10(par(i,2)))/log10(maxBi))^2;
    tau_mvara(i)=((log10(Pa(3))-log10(par(i,3)))/log10(maxtau_m))^2;
    Fa(i)=1/(Svara(i)+Bivara(i)+tau_mvara(i)+1)^(1/2);
end
t3=toc;

Ba=C*Fa';
t4=toc;

%arbitrary field
Asl=fe*Ba;
for i = 1:N
    As(i,1)=Asl(i);
end
t5=toc;

% festar=B*pinv(par');
Bn=pinv(fe)*As;
k=pinv(festar)*Bn;

time = toc;             %final clock time
time1=t2-t1;
time2=t3-t2;
time3=t4-t3;
time4=t5-t4;
report=t5-t2;

% Input for comparison to a known case, change the row to match
parameters
w=xlsread('tc1','Sheet1', 'D67:Y67');

W=w';

tim=xlsread('tc1','Sheet1', 'D66:Y66');

% Redimensionalize to return temperature in time
ts = tim * R^2 / alphaa;
Temp = P * Kapa * R^2 * As / (con * pi * r_o^2) + Tinf;
TRet = P * Kapa * R^2 * W / (con * pi * r_o^2) + Tinf;

% Plot comparison of Input Profile and ROM Profile
figure;
plot(tim,w,'-')
hold on
plot(tim,Temp,'o')
legend({'DSFA Model';'ROM'})
ylabel('Temperature (K)')
xlabel('Time (s)')
hold off

AS=As';
TEMP=Temp';
```

APPENDIX F. SOURCE CODE FOR SOLVING THE INVERSE PROBLEM OF DSFA WITH GA

Source code file:	GA.m
Compile instructions:	Ensure the Excel file is saved in the same working directory that Matlab is in. Once the file is loaded in the editor window, simply click 'run' or type the name of the file in the command window.
User inputs:	None
Program input files:	DSFA_par_360.xls. Raw solutions from the coefficient builder program for each parameter set and the additions from the ROM_code_matrix.m program.
Required program functions:	Matlab programmed function ROM_Func.m needs to be saved in a directory that the current path in Matlab has defined.
Program output files:	None

Program source code:

```
% GA.m

% Genetic Algorithm Program for determining governing parameters h,
% kappa,alpha that would yield an arbitrary input temperature profile.
% Reads in parameters from the ROM_code_matrix.m program saved in Excel
% at 'DSFA_630.xls' and uses them in the ROM subroutine ROM_Func.m

% Inputs the FLUENT case from 'Blind_Test_##.xls' after running DSFA
% model

clear all; clc;
tic

Cp=xlsread('DSFA_630', 'C', 'A1:Y630');         % Ensure that the range
param=xlsread('DSFA_630', 'Sheet1', 'A4:C633'); % of all these read in
Smax=xlsread('DSFA_630', 'Vars', 'C1');         % matrices match the
Smin=xlsread('DSFA_630', 'Vars', 'D1');         % actual ranges in the
bimax=xlsread('DSFA_630', 'Vars', 'C2');        % Excel files
bimin=xlsread('DSFA_630', 'Vars', 'D2');
Tau_mmax=xlsread('DSFA_630', 'Vars', 'C3');
Tau_mmin=xlsread('DSFA_630', 'Vars', 'D3');
almax=xlsread('DSFA_630', 'Vars', 'A1');
almin=xlsread('DSFA_630', 'Vars', 'B1');
hcapmax=xlsread('DSFA_630', 'Vars', 'A2');
hcapmin=xlsread('DSFA_630', 'Vars', 'B2');
hmax=xlsread('DSFA_630', 'Vars', 'A3');
hmin=xlsread('DSFA_630', 'Vars', 'B3');
kapmax=xlsread('DSFA_630', 'Vars', 'A4');
kapmin=xlsread('DSFA_630', 'Vars', 'B4');
ro=xlsread('DSFA_630', 'Vars', 'A5');
R=xlsread('DSFA_630', 'Vars', 'A6');
L=xlsread('DSFA_630', 'Vars', 'A7');
tp=xlsread('DSFA_630', 'Vars', 'A8');
tm=xlsread('DSFA_630', 'Vars', 'A9');
P=xlsread('DSFA_630', 'Vars', 'A10');
Tinf=xlsread('DSFA_630', 'Vars', 'A11');
timestep=xlsread('DSFA_630','Vars','A12');
fe=xlsread('DSFA_630', 'fe', 'A1:Y200');
Bp=xlsread('DSFA_630', 'B', 'A1:Y630');

% Input time and temperature profiles

% timereadin =xlsread('Blind_Test_1_new','Sheet1','D2:D40002');
% Treadin=xlsread('Blind_Test_1_new','Sheet1','E2:E40002');
% timereadin=xlsread('Blind_Test_2','Sheet1','D2:D35302');
% Treadin=xlsread('Blind_Test_2','Sheet1','E2:E35302');
% timereadin=xlsread('Blind_Test_3','Sheet1','D2:D65536');
% Treadin=xlsread('Blind_Test_3','Sheet1','E2:E65536');
% timereadin = xlsread('Blind_Test_5','Sheet1','A2:A65536');
% Treadin = xlsread('Blind_Test_5','Sheet1','B2:B65536');
% timereadin = xlsread('Blind_Test_7','Sheet1','A2:A40002');
% Treadin = xlsread('Blind_Test_7','Sheet1','B2:B40002');
% timereadin = xlsread('Blind_Test_8','Sheet1','A2:A40002');
% Treadin = xlsread('Blind_Test_8','Sheet1','B2:B40002');
% timereadin = xlsread('Blind_Test_9','Sheet1','A2:A40002');
% Treadin = xlsread('Blind_Test_9','Sheet1','B2:B40002');
% timereadin = xlsread('Blind_Test_11','Sheet1','D2:D40002');
% Treadin = xlsread('Blind_Test_11','Sheet1','F2:F40002');
timereadin = xlsread('Blind_Test_12','Sheet1','A2:A65536');
Treadin = xlsread('Blind_Test_12','Sheet1','B2:B65536');
```

```
Bimax=log10(bimax);              % Convert to logarithmic for faster
Bimin=log10(bimin);              % ROM times
tau_mmax=log10(Tau_mmax);
tau_mmin=log10(Tau_mmin);
SP=size(param,1);
for i=1:SP
    param(i,2)=log10(param(i,2));
    param(i,3)=log10(param(i,3));
end

clear SDev
clear value

t1=toc;

C=Cp';
B=Bp';

testcases = 5;     % Run the case 5 times to ensure consistent
convergence.
for aaa = 1:testcases
    Tins(:,aaa)=Treadin;
    timein(:,aaa)=timereadin;
end

for tests=1:testcases

clear SDev
clear value
tests
call = 0;
Tin=Tins(:,tests); % Input profile

[tin_max,locin]=max(Tin);
trisein=timein(locin);
Ti=Tin;

[len, i]=size(Ti);
ep=.05;                 % convergence criteria

N = 150;                % Number of parameter sets

BEST_EVER(5) = -1e9;    % Initialize fitness check
Gen=1;

% Create Initial Parameter sets
for i=1:N
    % Conv. coefficient
    h = hmin+(hmax-hmin)*rand();
    % Absorbtion coefficient
    kappa = kapmin+(kapmax-kapmin)*rand();
    % Thermal Diffusivity
    alpha = 10^(log10(almin)+(log10(almax)-log10(almin))*rand());
    % Heat Capacity
```

```
hcap = hcapmin+(hcapmax-hcapmin)*rand();

S=kappa*L;
Bi=log10(h*L/(alpha*hcap));
tau_m=log10(alpha*tm/R^2);

% Determine Temperature profile for parameter set
theta = ROM_Func(S,Bi,tau_m,C,B,param,Smax,Bimax,tau_mmax,fe);
Temp = theta*(P*kappa*R^2/(pi*ro^2*alpha*hcap))+Tinf;
call=call+1;

for RT = 1:length(Temp)
    ROMtime(RT) = R^2 * timestep*(RT-1) / alpha;
end

% Find time of maximum temperature rise
[t_max,loc]=max(Temp);
trise=timein(loc);
triseg=ROMtime(loc);

% Create interpolated time and temperature arrays for comparison
timeint(1) = timein(1);
Tempint(1) = Tins(1);
intp(1,:) = [1,0];
t_i = 2;
for itp = 2:length(Temp)
    while ((ROMtime(itp) > timein(t_i-1))&&(t_i-1 < numel(timein)))
        t_i = t_i + 1;
    end
    if t_i > numel(timein)
        t_i = numel(timein);
    end
    omega = (timein(t_i-1) - ROMtime(itp))/(timein(t_i-1)-...
        timein(t_i-2));
    intp(itp,:) = [t_i-1,omega];
    timeint(itp) = ROMtime(itp);
    Tempint(itp) = Tins(t_i-1)-omega*(Tins(t_i-1)-Tins(t_i-2));
    t_i = t_i + 1;
end

% Calculate Fitness Function for Randomly compiled parameter sets
val=0;
for jj=2:length(Temp)
    val=abs(Tempint(jj)-Temp(jj))/Tempint(jj)*100+val;
end
avdif=val/len*100;
maxdiff = abs(tin_max-t_max)/(tin_max-Tinf) * 10;

if trisein == triseg
    timediff = 0;
else
    timediff = (max(trisein/triseg,triseg/trisein)-1)*10;
end

% Calculate Fitness Function
f = -(avdif+maxdiff+timediff);
start_par(i,:)=[h, kappa, alpha, hcap, f];
```

```
    % Store the best fitness function team
    if f > BEST_EVER(5)
        BEST_EVER(1,:)=[h, kappa, alpha, hcap, f];
    end
    C1 = max(start_par(:,5));
    value(Gen,1) = C1;

end
t2=toc;

par=start_par;
C1 = max(par(:,5));
val = C1;
SDev(Gen)=std(par(:,5));
SDEV=std(par(:,5));

NGen = 500;       % Number of Generations



% Generation Loop
while SDEV>ep


    % Start Tournament
    for i = 2:N
        first = ceil(N*rand());
        second = ceil(N*rand());

        % Tournament selections
        if par(first,5) > par(second,5)
            tourney(i) = first;
        else
            tourney(i) = second;
        end
    end

    par(1,:)=BEST_EVER;

    % Store set that won the tourney as the set
    for i = 2:N
        par(i,:)=par(tourney(i),:);
    end

    % Crossover, Blend Crossover
    i = 1;
    while i < N

        parent1=par(i,:);      % Store set i as Parent 1
        parent2=par(i+1,:);    % Store set i+1 as Parent 2

        for m=1:2
            if rand() < .5      % 50% Probability of crossover
                r=rand();
                y1=r*parent1(m)+(1-r)*parent2(m);
                y2=(1-r)*parent1(m)+r*parent2(m);
                parent1(m)=y1;
                parent2(m)=y2;
            end
        end
        for m=3:3
            if rand() < .5      % 50% Probability of crossover
                r=rand();
                y1=10^(r*log10(parent1(m))+(1-r)*log10(parent2(m)));
                y2=10^((1-r)*log10(parent1(m))+r*log10(parent2(m)));
                parent1(m)=y1;
                parent2(m)=y2;
            end
        end
        for m=4:4
            if rand() < .5      % 50% Probability of crossover
                r=rand();
                y1=r*parent1(m)+(1-r)*parent2(m);
                y2=(1-r)*parent1(m)+r*parent2(m);
                parent1(m)=y1;
                parent2(m)=y2;
            end
        end

        par(i,:)=parent1;      % set team i as child 1 (parent1)
        par(i+1,:)=parent2;    % set team i+1 as child 2 (parent2)

        i=i+2;
    end

    % Mutation Algorithm

    for i = 1:N
        SPM = .75;                         % Starting prob. for mutation
        alfa(Gen) = SPM*(1-(Gen-1)/NGen)^4;  % Change power for rate
        % Check case with random number against alfa for mutation
        for j = 1:4
            % Convective Coefficient Mutation
            if j == 1
                if rand()<alfa(Gen)
                    hmut = hmin+(hmax-hmin)*rand();
                    par(i,j)=hmut;
                end

            % Absorbtion Coefficient Mutation
            elseif j == 2
                if rand()<alfa(Gen)
                    kapmut = kapmin+(kapmax-kapmin)*rand();
                    par(i,j)=kapmut;
                end

            % Thermal Diffusivity Mutation
            elseif j == 3
                if rand()<alfa(Gen)
                    almut = 10^(log10(almin)+(log10(almax)- ...
                        log10(almin))*rand());
                    par(i,j)=almut;
                end
            % Heat Capacity Mutation
            elseif j == 4
                if rand()<alfa(Gen)
                    hcapmut = hcapmin+(hcapmax-hcapmin)*rand();
                    par(i,j)=hcapmut;
                end
            end
        end
    end

    % Calculate Fitness function of new sets after tournament,
    % crossover, and mutation

    for i = 1:N

        h=par(i,1);
        kappa=par(i,2);
        alpha=par(i,3);
        hcap=par(i,4);

        S = par(i,2)*L;
        Bi = log10(par(i,1)*L/(par(i,3)*par(i,4)));
        tau_m = log10(par(i,3)*tm/R^2);

        % Calculate Temperature profile for the set
        theta = ROM_Func(S,Bi,tau_m,C,B,param,Smax,Bimax,tau_mmax,fe);
        Temp = theta*(P*kappa*R^2/(pi*ro^2*alpha*hcap))+Tinf;
        call=call+1;

        for RT = 1:length(Temp)
            ROMtime(RT) = R^2 * timestep*(RT-1) / alpha;
        end

        [t_max,loc]=max(Temp);
        trise=timein(loc);
        triseg=ROMtime(loc);

        % Create interpolated time and temperature arrays for
        % comparison
        timeint(1) = timein(1);
        Tempint(1) = Tins(1);
        intp(1,:) = [1,0];
        t_i = 2;
        for itp = 2:length(Temp)
            while ((ROMtime(itp) > timein(t_i-1)) && (t_i-1 < ...
                    numel(timein)))
                t_i = t_i + 1;
            end
            if t_i > numel(timein)
                t_i = numel(timein);
            end
            omega = (timein(t_i-1) - ROMtime(itp))/(timein(t_i-1)-...
                timein(t_i-2));
            intp(itp,:) = [t_i-1,omega];
            timeint(itp) = ROMtime(itp);
            Tempint(itp) = Tins(t_i-1)-omega*(Tins(t_i-1)-Tins(t_i-2));
```

```
            t_i = t_i + 1;
        end

        % Calculate Fitness Functions for the sets
        val=0;
        for jj=2:length(Temp)
            val=abs(Tempint(jj)-Temp(jj))/Tempint(jj)*100+val;
        end
        avdif=val/len*100;
        maxdiff = abs(tin_max - t_max)/(tin_max-Tinf)*10;
        if trisein == triseg
            timediff = 0;
        else
            timediff = (max(trisein/triseg,triseg/trisein)-1)*10;
        end

        % Calculate Fitness Function
        f = -(avdif+maxdiff+timediff);
        par(i,5)= f;

        if f > BEST_EVER(5)
            BEST_EVER(1,:)=[par(i,1), par(i,2), par(i,3), par(i,4), f];
        end

    end

    % Increment Generation number
    Gen=1+Gen;
    % Keep the best set on to the next generation (elitism)
    Geners(tests)=Gen;   % Keep amount of generations run for each test
    par(1,:)=BEST_EVER(1,:);
    C1 = max(par(:,5));
    value(Gen,1) = C1;
    SDev(Gen)=std(par(:,5));
    SDEV=SDev(Gen);
    if SDev(Gen)<ep
        break
    end
    if Gen>NGen
        break
    end

end
time = toc;

h = BEST_EVER(1,1);
kappa = BEST_EVER(1,2);
alpha = BEST_EVER(1,3);
hcap = BEST_EVER(1,4);

S = kappa*L;
Bi = log10(h*L/(alpha*hcap));
taum = log10(BEST_EVER(1,3)*tm/R^2);

theta = ROM_Func(S,Bi,tau_m,C,B,param,Smax,Bimax,tau_mmax,fe);
dimTemp=P*kappa*R^2*theta/(alpha*hcap*pi*ro^2)+Tinf;
dimTempin=P*kappa*R^2*Tin/(alpha*hcap*pi*ro^2)+Tinf;
```

```
dtime = timein*R^2/alpha;

done(tests,:)=[kappa; alpha; hcap; h];

figure;
plot(timein,Tins);
hold all
plot(ROMtime,Temp,'bo');
hold off
xlabel('Time (sec)')
ylabel('Temperature (K)')
legend('Input Temperature Profile','GA with ROM Profile')

end
```

<u>Required functions source code:</u>

```
% Rom_Func.m

% Reduced Order Modeling simulation subroutine for the calculation time
% dependant temperature profiles in a Pulsed Laser Diffusion
% experiment.

% Parameters are brought in from GA.m program and input into subroutine
% for the ROM to simulate temperature profile.

% Returns Temperature, As.

function ROM_Func=f(Sa,Bia,tauma,C,B,par,maxS,maxBi,maxtaum,fe);

[i,M]=size(C);
[N,i]=size(fe);

% Read input arbitrary conditions from GA program
Pa=[Sa Bia tauma];

%interpolation function
for i=1:M
    Svara(i)=((Pa(1)-par(i,1))/maxS)^2;
    Bivara(i)=(((Pa(2))-(par(i,2)))/(maxBi))^2;
    taumvara(i)=(((Pa(3))-(par(i,3)))/(maxtaum))^2;
    Fa(i)=1/(Svara(i)+Bivara(i)+taumvara(i)+1)^(1/2);
end

% ROM matrix manipulation to return As

Ba=C*Fa';

Asl=fe*Ba;
for i = 1:N
    As(i,1)=Asl(i);
```

```
end

ROM_Func=As;
```