Optimal Control of Thermal Damage to Biological Materials

by

F. Scott Gayzik, B.S.

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Chair
Elaine P. Scott, Ph.D.
Thomas E. Diller, Sc.D.
Tahar Loulou, Ph.D.

September 2004

Blacksburg, Virginia

Keywords : Conjugate Gradient, Finite Difference, Arrhenius Damage Model, Thermal Dose

Copyright by F. Scott Gayzik, 2004

Optimal Control of Thermal Damage to Biological Materials

F. Scott Gayzik, B.S.
Virginia Polytechnic Institute and State University, 2004
Supervisor: Elaine P. Scott

Abstract

Hyperthermia is a cancer treatment modality that raises cancerous tissue to cytotoxic temperature levels for roughly 30 to 45 minutes. Hyperthermia treatment planning refers to the use of computational models to optimize the heating protocol to be used in a hyperthermia treatment. This thesis presents a method to optimize a hyperthermia treatment heating protocol. An algorithm is developed which recovers a heating protocol that will cause a desired amount of thermal damage within a region of tissue. The optimization algorithm is validated experimentally on an albumen tissue phantom.

The transient temperature distribution within the region is simulated using a two-dimensional, finite-difference model of the Pennes bioheat equation. The relationship between temperature and time is integrated to produce a "damage field" according to two different models; Henriques' model and the thermal dose model (Moritz and Henriques (1947)), (Sapareto and Dewey (1984)). A minimization algorithm is developed which reduces the value of an objective function based on the squared difference between an optimal and calculated damage field. Either damage model can be used in the minimization algorithm. The adjoint problem in conjunction with the conjugate gradient method is used to minimize the objective function of the control problem.

The flexibility of the minimization algorithm is proven experimentally and through a variety of simulations. With regards to the validation experiment, the optimal and recovered regions of permanent thermal damage are in good agreement for each test performed. A sensitivity analysis of the finite difference and damage models shows that the experimentally-obtained extent of damage is consistently within a tolerable error range.

Excellent agreement between the optimal and recovered damage fields is also found in simulations of hyperthermia treatments on perfused tissue. A simplified and complex model of the human skin were created for use within the algorithm. Minimizations using both the Henriques' model and the thermal dose model in the objective function are performed.

The Henriques' damage model was found to be more desirable for use in the minimization algorithm than the thermal dose model because it is less computationally intensive and includes a mechanism to predict the threshold of permanent thermal damage. The performance of the minimization algorithm was not hindered by adding complexity to the skin model. The method presented here for optimizing hyperthermia treatments is shown to be robust and merits further investigation using more complicated patient models.

Acknowledgments

I'd like to thank first and foremost my advisor, Dr. Elaine Scott for her support (both financial, and mental) during my thesis work. Special thanks goes to Dr. Tahar Loulou who did a wonderful job helping me with the control problem. I would also like to thank Dr. Thomas Diller for his insight and words of encouragement on all things experimental.

Next up, I must certainly thank my parents, Al and Fran for their unwavering support of my job-dodging. I would like to extent similar thanks to *la familia*: Adam, Jim, Nick, Nonna, PopPop, Aunt Sandy, Uncle Russ, Uncle Bill, Matt, Katie, Aunt Carol, Moose, Tina and of course Taylor. You all make going home a wonderful change of pace.

As far as the nuts and bolts of gettin 'er done, I need to thank the following.

- Minco for their generous donation of the submersible heater.
- The machine shop guys for making the test stand and Lee and Dan for their side work
- Special thanks to Kalian for the LabView help.

Now on to the friends. You know who you are. Thanks for making Blacksburg the best little city on earth. Viva Spartacus.

Last but not least, thanks for listening to me complain Caroline.

F. Scott Gayzik

Virginia Polytechnic Institute and State University

September 2004

Contents

List of Tables

List of Figures

Nomenclature

Nomenclature Used in Models

k	Thermal Conductivity ($W/m-K$)	$\bar{T}, \theta$	Temperature Referenced to a Base (K, C)
ρ	Density (kg/m^3)	$\dot{E}$, $\dot{Q}$, S_c	Volumetric Energy Source (W/m^3)
C_p	Specific Heat ($J/kg-K$)	q"	Heat Flux (W/m^2)
ω	Perfusion (kg/m^3-s)	h	Convection Coefficient (W/m^2-K)
T	Temperature (K, C)	W, H_i, S_p	Internal Convection Coefficient (W/m^3-K)
t	Time (s)	H	Height of Test Stand (cm)
Ω	Thermal Damage Coefficient ($Dimm$)	L	Radius of Test Stand (cm)
D	Thermal Dose (sec. at 43°C)	x	Radial or Horizontal Direction (cm, mm)
A	Molecular Frequency Factor ($1/s$)	y	Axial or Vertical Direction (cm, mm)
E	Activation Energy (J/mol)	N	A Discrete Quantity
$\Re$	Universal Gas Constant ($J/mol-K$)	μ	Average
R	Thermal Dose Constant ($Dimm$)	σ	Standard deviation
E	Activation Energy (J/mol)	X_i^+	Dimensionless Sensitivity Coefficient
H_{ab}	Maximum Extent of Axial Damage (mm)	D_{ab}	Maximum Extent of Radial Damage (mm)

Nomenclature Used in Finite Difference

aE, aW, etc.	Directional Coefficient (W/K)	Ae, Aw, etc.	Area of Cell at East, West,... face (m^2)
V	Volume of a cell (m^3)	b, $\hat{\text{b}}$	Simplifying Coefficients (W)
c	Simplifying Coefficient (W/K)		

Nomenclature Used in Minimization Algorithm

Λ	Generalized Damage Model	Γ	Generalized Control Function
$\mathcal{J}$	Objective Function	$\mathcal{L}$	Lagrangian of the System
$\Psi_{1\rightarrow 7}$	Lagrange Multipliers	Ψ	Adjoint Problem Variable
Δ	Variation in a Quantity	β	Decent Parameter
ϵ, ε	Small Perturbation	γ	Conjugation Coefficient
$\Delta\Gamma$	Descent Direction	E_{RMS}	RMS Error in Λ

Subscripts

tis	Tissue	bl	Blood
c	Calculated	d	Desired
f	Final	∞	Ambient
i	Horizontal, Radial Marker	j	Vertical, Axial marker
hor	Horizontal	vert	Vertical
o	Initial	ab	Albumen

Superscripts

p	Time Step	s	Iteration No.
’	Gradient		

Abbreviations

HTP	Hyperthermia Treatment Planning	RF	Radio Frequency
FD	Finite Difference	HIFU	High Intensity Focused Ultrasound
MRI	Magnetic Resonance Imaging	BHTE	Bioheat Transfer Equation
TDMA	Tridiagonal Matrix Algorithm	TC	Thermocouple

Chapter 1

Introduction

The study of bioheat transfer, or heat transfer in physiological systems, has been of interest to researchers for many years. As medicine shifts from a qualitative to more quantitative nature, applications of bioheat transfer to specific engineering problems are becoming more prevalent. A topic of research at the core of bioheat transfer and the thermal behavior of tissue is hyperthermia and more specifically, hyperthermia treatment planning (HTP). Hyperthermia is a modality used in the treatment of cancer where a targeted area of tissue is exposed to elevated temperatures for a specific time period. The elevated temperatures are commonly achieved percutaneously via an energy source such as microwaves or high-intensity focused ultrasound.

Hyperthermia treatment planning is very complex, relying on expertise in fields ranging from heat transfer to physiology to computer science. The crux of any HTP is an accurate computational model to predict the transient temperature distribution in and around a tumor site during a heating protocol. The state of the art of this practice is to use imaging techniques such as MRI to reconstruct patient specific models of the tumor region, which in some cases can even include local vasculature (Lagendijk (2000)). In addition to a model which predicts the transient temperature distribution, a separate model which integrates the time-temperature distribution in order to predict the extent of thermal damage is required. Taking these models as a foundation, a treatment planing system can be devised.

1.1 Goals and Objectives

The major goal of this research is to incorporate a bioheat transfer model and two different means of evaluating thermal damage within a problem of optimal control to recover an optimized heating protocol for use in a hyperthermia treatment. Supporting objectives that must be met to reach this goal are outlined below.

1. A two-dimensional finite difference algorithm to solve the bioheat equation is to be programmed in MATLAB. The program must also be able to integrate the resulting damage field from the calculated transient temperature history.

2. The robustness of this method will be investigated by solving the control problem for a variety of heating sources (surface heat flux and volumetric heating) as well as for a variety of physiologically relevant situations. These effects include areas of elevated blood perfusion, inhomogeneous and temperature dependent properties.

3. Because there multiple means of quantifying thermal damage the an objective of this research is to formulate the control problem based on two accepted thermal damage evaluation criterion. These are an Arrhenius type thermal damage model and a more clinically used thermal dose model, both of which are explained in detail below and in chapter 3. A secondary objective is to compare the performance between the thermal damage model and the thermal dose model within the algorithm.

4. The final objective is to test the validity of the optimization algorithm by means of an experiment on an albumen (egg white) tissue phantom. Comparisons of specific damage metrics from simulated and experimental tests are to be made.

Because of the complexity of HTP, some key assumptions were made to narrow the scope of this project. First, the mechanism chosen to deliver the heat in all simulations is fixed. Only variations in the intensity of a chosen heat source during a user-specified treatment time are optimized. Two different heating sources are simulated as a means of delivering energy to the control volume; surface heat flux and volumetric heating. Secondly, while the models used in the hyperthermia simulations presented in this work contain some of the complexities encountered in normal human physiology, they have not been rendered from any actual patient data. However regardless of these simplifications, the methods presented here can be applied to much more complicated models.

This project is a testament to the fact that biomedical engineering is a truly collaborative field. Several major areas of focus were brought together in the completion of this work, each of which is outlined below.

Bioheat transfer model

The Pennes bioheat equation was used as the model to calculate the two-dimensional transient temperature field throughout the control volume during a simulated treatment. This model is the standard used in the bioheat transfer studies because it includes the convective effects of blood perfusion. The model was solved via a fully-implicit finite difference (FD) scheme. The FD scheme was programmed using an alternating direction, line by line approach in conjunction the tridiagonal matrix algorithm (TDMA) to reduce computation time (Patankar (1980)).

Thermal damage evaluation

The model used to quantify thermal damage in an HTP is as critical to the overall goal as the bioheat transfer model. In this work two separate means of quantifying thermal damage are employed in the optimal control problem. The first is an Arrhenius type damage model based on first-order kinetics, henceforth referred to as the thermal damage coefficient (TDC). This model requires utilizing a scaling factor based on the molecular collision rate of the substance of interest and an activation energy. These parameters are available in the literature for a wide variety of biological materials, including human organ tissue and albumen. Because of the means by which these constants are obtained experimentally a specific value of TDC can be correlated to permanent thermal damage (ablation).

The second model used in this work to quantify thermal damage is known as the thermal dose, which was initially introduced by Separato and Dewey (Sapareto and Dewey (1984)). This is a widely used means of quantifying thermal damage by correlating the amount of damage to a specific exposure time at 43^{o}C. The strength of the thermal dose lies in its use as a means of comparing two completely different hyperthermia treatment protocols.

Optimal control problem formulation

The problem of optimal control is the backbone of the research. The bioheat transfer model and thermal damage models integral are parts of its framework. The objective or cost function is based on a sum of squares difference between an optimal, user-defined damage field and a damage field calculated via the bioheat transfer model and thermal damage evaluation method. The gradient of the objective functional is found via the calculus of variations, using the adjoint method. The gradient is reduced by the conjugate gradient method in an computational algorithm implemented in MATLAB.

Experimental validation of the control problem

The body of work is capped off with an experiment designed to validate the findings of the optimal control problem. Starting with a desired damage field, the control algorithm was used to recover an optimized heating protocol which would cause this damage field. In the experiment the source is a disc-shaped Teflon coated resistance heater from Minco. High accuracy data acquisition and control equipment from National Instruments was used in the execution of the experiment. LabView was used as the DAQ software because of its ease of programming and flexibility. Further explanation of the experimental setup and outcome is discussed in Chapter 5.

Document overview

Bringing together each of the above components, a simple HTP was derived, coded and evaluated experimentally. The present thesis is a comprehensive source of the work done in completing this task. It begins with a literature review of the current state of knowledge of the core areas of this project, including hyperthermia treatment planning, bioheat transfer models, thermal damage quantification models, engineering optimization and tissue phantoms. Following this, Chapter 3 introduces the Pennes Bioheat transfer model, thermal damage coefficient and thermal dose models. The control problem is then derived in detail in Chapter 4. Chapter 5 is dedicated to reporting the experimental methods used in the control problem validation. Finally results from the validation experiment as well as from the computational algorithm are discussed in Chapter 6. Recommendations for future work are included.

Chapter 2

Literature Review

Previous work in the fields of hyperthermia, bioheat transfer, thermal damage modeling and quantification, optimal control and hyperthermia treatment planning is summarized in this section. The section gives an overview of relevant research done in the major points of interest of this thesis.

2.1 Hyperthermia and Thermal Ablation Therapies

Hyperthermia is generally regarded as elevating temperatures of a cancerous region of tissue to cytotoxic temperature in the range of 41 to 44 ^{o}C for a duration of 30 to 45 minutes. It is often used in conjunction with other treatment modalities such as radiotherapy and chemotherapy. The treatments aim to maintain surrounding healthy tissue at physiologically normal temperatures and target only regions where malignancies are found.

Thanks to recent technological gains in fields ranging from medical imaging to energy delivery mechanisms, thermal ablation strategies have become an attractive alternative to standard surgical therapies in the treatment of malignant cancerous tumors (Ahmed and Goldberg (2002)). These treatment modalities have the added benefit of being only minimally invasive or completely non-invasive. There are a wide variety of energy sources currently used, but some of the most common are high-intensity focused ultrasound (HIFU), radio frequency(RF), laser and microwave heating. The heating simulations presented in this work are based on HIFU treatments because they are non-invasive and can be modeled as volumetric or boundary heating in the simulation.

HIFU is a transcutaneous, minimally invasive technique that focuses ultrasound

energy with a high peak intensity (Ahmed and Goldberg (2002)). This technology is capable of generating a power intensity of 5,500 kW/m^2 and temperatures as high as 80^{o}C in rat liver with the use of a 4 MHz transducer. Ultrasound heating is limited by interference with regional bone and air and the fact that only small volumes of tissue that can be thermally ablated with this technology during one treatment (Ahmed and Goldberg (2002)). This information is given to provide a brief background of ultrasound heating. The focus of the paper is on the optimization of the protocol not on the particular modality itself.

2.2 The Bioheat Transfer and Thermal Damage Models

The foundation of this research are models which predict both the transient temperature distribution within a section of tissue or biological material, and the corresponding extent of thermal damage. Since the particular modality is not the focus of this work, models which describe how a particular modality such as HIFU heats tissue are not included. Instead, energy enters the control volume as volumetric or boundary heating. The Bioheat transfer and thermal damage models are now presented.

2.2.1 The Bioheat equation

The one component common to nearly any hyperthermia simulation is the Pennes Bioheat Transfer Equation, (BHTE), which is shown below in generalized form (Pennes (1948)). This equation was developed as a means of quantifying the effect of blood within the microvasculature of the tissue on the local temperatures. The presence of blood in the tissue is considered a convective term with the blood temperature as the heat sink temperature. Pennes asserts that accurate modelling of the temperatures within the body, particularly the human forearm, relies on accurate knowledge of the blood flow or perfusion, $\omega(kg_{bl}/m^3_{tis}\,s)$ in that area. Nearly 60 years later, it remains a difficult parameter to quantify. The BHTE is shown in detail in chapter 3.

$$\rho C_{p,tis}(T)\frac{\partial T}{\partial t} = k\,\nabla T + \omega_{bl}\,(\rho\,Cp)_{bl} \tag{2.1}$$

Alternate approaches do exists for modeling heat transfer in the body. For example if a large scale number of vessels are included as a velocity vector, $\vec{u}$, the equation can be written as shown in Eqn. 2.2. A solution to this model would require detailed knowledge

of the vasculature in the area being modeled and would be much more computationally intensive than the BHTE.

$$\rho C_{p,tis}(T)\left(\frac{\partial T}{\partial t}+\vec{u}\,\nabla T\right)=k\,\nabla T+\omega_{bl}\,(\rho\,Cp)_{bl} \tag{2.2}$$

The focus of this work is on the optimization itself, not on which variation of the BHTE is used. Loulou and Scott (2002) and Alifanov (1994) have shown that the conjugate gradient method with the adjoint problem can be applied to a wide variety of different governing equations. For this reason the BHTE shown in Eqn. 2.1 will be used in this thesis to model the transient temperature field.

2.2.2 Thermal damage models

Two different thermal damage models are used in this study which both quantify the irreversible rate process of thermal injury. The models require *a priori* knowledge of the temperature distribution in the area of interest. Since the full transient temperature of the area of interest is calculated with the BHTE, they are a natural fit. The reasoning behind using this type of model is that the complexities of thermal denaturation can be somewhat overlooked if the temperature history is taken as a summary of the more complicated story going on at the molecular level (Diller (1992)).

The Henriques damage model

The first model of thermal damage is commonly known of as the Henriques model (Henriques and Moritz (1947)). Henriques and Mortiz formulated this model by investigating the extent of thermal injury to porcine skin. The damage was caused by applying a constant temperature boundary condition to a portion of porcine skin for a specified portion of time. The skin was investigated to quantify the extent of thermal damage. A summation of instantaneous damages gives the complete damage equation below.

$$\Omega(x,y)=\int_0^{t_f} A\,\exp\left(\frac{-E}{\Re\,T(x,y,t)}\right)\,dt \tag{2.3}$$

The equation requires material-specific values of the activation energy, E (J/mol) which is an energy barrier molecules must overcome to denature, and the molecular frequency factor, A (1/s) (Pearce and Thomsen (1992)). Here, $\Re$ is the universal gas constant.

The activation energy and frequency factor are usually found via constant temperature experiments. First a threshold of permanent damage, $\Omega = 1$ is selected. Then samples of a material are placed in a constant temperature bath of a non-reactive fluid such as water which is held at a constant temperature. Then precise measurements are taken for the time taken to achieve permanent thermal damage. These ordered pairs of time and temperature are then used in a least-squares fit to estimate E and A (Yang et al. (1991)). The method works well except at the extremes of the constant temperature bath range. Over a relatively small temperature range, denaturation times can vary by many orders of magnitude. For instance it may take hours or even days to burn skin at 39^oC but only seconds at 80^oC.

Pearce and Thomsen (1992) expanded on the damage model in their study of tissue fusion processes. They state that the damage integral is also equal to the natural log of the ratio of the concentration of undamaged material at the beginning of a treatment to the concentration of undamaged material at the end of a treatment.

$$\Omega(x,y) = \ln\left(\frac{C(t=0)}{C(t=t_f)}\right) = \int_0^{t_f} A \exp\left(\frac{-E}{\Re T(x,y,t)}\right) dt \tag{2.4}$$

If a damage coefficient of $\Omega = 1$ is used and the assumption that the material is 100% undamaged at the beginning of a treatment, complete thermal damage occurs when 63.2% of the local material has been permanently damaged.

The thermal dose model

Nearly 40 years after the work of Moritz and Henriques, Sapreto and Dewey (1984) published a model which was an effort to draw comparisons between different hyperthermia treatments. The comparison works by using a reference temperature of 43^oC and relating any time-temperature history to the "iso-effect" at that reference temperature. A thermal dose of one represents one hour at 43^oC. The equation is shown below and explained further in chapter 3. The thermal dose has been used to asses the extent of treatments and to optimize treatments particularly in the field of HIFU (Dorr (1992), Loulou and Scott (2002)).

$$D(x,y) = \int_0^{t_f} R(T)^{\overline{T}(x,y,t)}\, dt \tag{2.5}$$

In effect this model differs from the Henriques model in one major way. In their

paper the authors comment that a one degree increase in temperature over 43^oC required a two-fold decrease in exposure time. Since R is actually a function of the activation energy, E the implication of this statement is that the energy barrier described in the previous model varies with temperature (Dewey (1994)).

To account for this variation, the term R is allowed to change with temperature, with R = 0.5 for temperatures greater than the 43 ^oC reference temperature and R = 0.25 for temperatures below 43^o. While research has shown that the activation energy does depend on temperature (Wright (2003)) the reported values for R which are widely used may not apply to all cell lines and biological materials. Loulou and Scott (2002) propose a linear variation of R between its high and low bounds, however successful operation of the algorithm does not depend on knowledge of the exact values of R, or even how R changes with temperature.

Both models have strong points. Given the uncertainty that already will be present in any simulation due to imprecise thermal property and perfusion values used, the gains by adding a temperature dependent activation energy may be outweighed by the computational simplicity of the Henriques model. The Arrhenius parameters included in both models (E, A and R) are available in the literature for a wide variety of materials cells and proteins (Dewey (1994), Diller (1992)). Additionally this model uses parameters which are calculated based on an observable change, making it much more practical to use in a laboratory setting. On the other hand, the unit of the thermal dose model is "exposure time at 43^oC, which can be valuable knowledge in a clinical setting.

It is important to note that these models are simplifications of a much more complex phenomenon. Wright and Humphrey (2002) assert that despite the current practice of using damage and dose models used in research a detailed understanding of the biomechanics of tissue denaturation is lacking. The review gives some key background of protein denaturation at the molecular level. One point which Wright makes, which is reiterated later in this research is that a very specific level of denaturation can achieved via a wide range of time-temperature curves, and mechanical stresses; the latter being a factor which is not treated in this research.

2.3 Experimental Phantom

The results of any optimized heating protocol should be validated experimentally. This validation requires a tissue phantom. Madden and Scot (2003) presented an agar based tissue phantom that could be used to simulate physiologically accurate blood perfusion. While this model is experimentally applicable, tissue will undergo denaturation whereas agar will not. Other researchers have however used albumen (egg white) as a tissue phantom because like tissue it is composed of proteins that denature permanently when heated. This heating follows a first order rate process reaction and can be modelled via the Henriques model (Pfefer et al. (2000)), (Pearce and Thomsen (1992)). The Arrhenius properties of albumen can be found in the literature (Yang et al. (1991)). Albumen is composed of roughly 80% water and 14% protein known as ovalbumin (Yamamoto et al. (1997)). Ablated albumen is opaque to the naked eye, facilitating measurements of the extent of permeant thermal damage.

2.4 The Minimization Algorithm

Loulou and Scott (2002) showed that the conjugate gradient method with the adjoint problem could be used to successfully recover an optimal thermal dose in a simulated hyperthermia treatment. The model presented is a one dimensional model of the BHTE and can be used with a variety of boundary conditions. The finite difference method was used to solve the BHTE and the algorithm showed reasonable computational efficiency. The method has the ability to be extended to multiple dimensions; however there was no experimental validation of this algorithm (Loulou and Scott (2000)).

2.5 Hyperthermia Treatment Planning

In recent years, there has been much work done in hyperthermia treatment planning (HTP). Lagendijk (2000) provides an excellent review of the state of the art of HTP. He asserts that the ability of local hyperthermia to supply lethal doses of heat to targeted areas is largely effected by the presence of perfused blood at the capillary level acting as a heat sink. These findings have been supported by many researchers (Xu et al. (1992)). Deep seating of the tumor as well as other patient specific factors such as the individual thick-

ness of subcutaneous fatty tissue and threshold of pain often cause difficulty in applying treatments (Gellermann et al. (2000)). Additionally noninvasive temperature monitoring in three dimensions is currently beyond the ability of imaging systems so it is often difficult to monitor the progress or accuracy of a planned treatment (Lagendijk (2000)). Lagendijk and others have recently put forth great effort in developing and testing a flexible discreet vasculature model, DIVA, which aims to include more accurately the effects of blood perfusion and discrete vasculature. The model was successfully tested on bovine tongues.

Researchers have been planning treatments for some time. James and Sullivan (1992) reported a method to incorporate three-dimensional models of patients constructed from CT scans into a transient finite difference model which would be used to simulate treatments. The heat was deposited via radiofrequency electromagnetic energy. The reconstructed model was used only to predict temperature distributions within the control volume and no thermal damage metric or dose was reported.

Gellermann et al. (2000) used the same heating equipment that James and Sullivan used, along with a much more advanced hyperthermia treatment planning system called HYPERPLAN. The main components of a treatment planing system such as HYPERPLAN are modules for three-dimensional surface reconstruction, grid generation, numerical simulation and visualization techniques to see the outcome of the treatment (Beck et al. (1997)).

An optimization module built into the system minimizes an objective function based on penalizing the estimated treatment plan if temperatures are below 43^oC within the tumor and above 42^oC outside the tumor. This planning system was still a long way from clinical applicability with the segmentation and building of the patient specific model from CT scans a very time consuming process; it took taking up to 8 hours. But regardless of the time per treatment plan, HYPERPLAN serves as an excellent indicator of where the science is going. This treatment planing system was actually tested on six patients with successful results.

Hynynen has worked extensively in the field of HIFU and has shown it may be feasible to use MRI in conjunction with phased arrays for live in vivo temperature control to prevent over or under dosage (Hynynen et al. (1996)). Lalonde and Hunt (1995) also have researched the optimization of a HIFU based hyperthermia treatment protocol. A rectangular focal

plane with constant thermal properties is used to simulate tissue. The solution, which gives the placement and intensity of ultrasound focal points is a linear programming problem based on equality constraints at the boundary and inequality constraints inside the control volume, keeping temperatures within the tumor inside an acceptable (43^oC to 42^oC) range. This method works well at confining the temperature distribution within the tissue to acceptable ranges. The method has a disadvantage of assuming high, uniform perfusion values that do not vary with temperature throughout the control volume. The BHTE is used to calculate the temperature distribution.

2.6 Closing Remarks

Much of the work in this field of treatment planing focuses on matching experimental temperature fields to simulated temperature fields within the area of interest. Often it is not certain if the temperature field in the region of the interest is behaving as predicted due to the lack of knowledge about regionally varying properties like blood perfusion. This uncertainty creates a major problem in accurately planning thermal therapies, therefore it is essential that any optimization algorithm made for treatment planning must be able to model regional changes in perfusion. In the future, advances in imaging techniques may work to resolve some of these problems by monitoring temperatures during treatments.

Problems of property values aside, one must ask what is the goal of HTP? With a chosen delivery system, the answer is to find the heating protocol that creates a desired dose or damage field within a targeted area. Since what is to be optimized is a dose or damage field, it is more logical to build an objective function around Sapareto and Dewey's thermal dose model or Henriques' thermal damage model. The work of Loulou and Scott (2000) focuses on the former while in this study, the Henriques model for thermal damage is incorporated into a problem of optimal control. Unlike the thermal dose, the Henriques model is based on parameters that are found experimentally, by tying thermal destruction to an observable physiological change in the material. Therefore, the Henriques model is more applicable to experimental validation in a laboratory because the extent of thermal damage can be observed directly.

Chapter 3

Theoretical Considerations

The bioheat transfer model and the models used to quantify thermal damage are integral parts of this research. Each model presented in this section is commonly used in research and the practice of treatment plans. First the Pennes bioheat equation is presented as the model to predict the transient temperature distribution in the control volume. The finite difference technique used to solve this differential equation as well as an analytical validation are also included. Finally the two models which quantify thermal damage are presented along with the numerical integration technique used to solve each equation.

3.1 The Pennes Bioheat Transfer Model

The Pennes bioheat transfer equation is a form of the heat diffusion equation including a term accounting for the effects of blood perfusion in the body. Blood perfusion is the non-directional flow of blood at the capillary level within tissue. The diffusion equation is derived starting with an energy balance on a differential control volume shown in Fig. 3.1. The following derivation is shown in three dimensions although it can easily be simplified to two or one. The energy balance requires that the amount of stored energy in a control volume, $\dot{E}_{st}$, equals energy generated volumetrically, $\dot{E}_{gen}$, plus the net difference in energy flow into and out of the control volume, $(\dot{E}_{in} - \dot{E}_{out})$.

$$\dot{E}_{st} = \dot{E}_{gen} + \sum(\dot{E}_{in} - \dot{E}_{out}) \tag{3.1}$$

It is useful to rewrite this energy balance with temperature as the independent variable. To begin this process, the storage term can be written in terms of its thermal

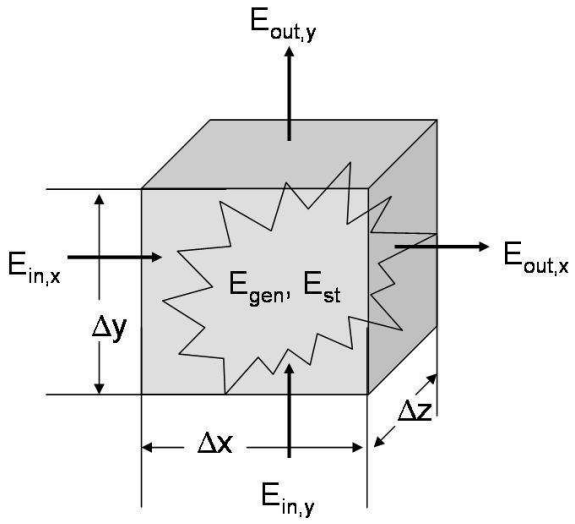

Fig. 3.1 Energy balance of generalized control volume.

capacitance, $\rho C_{p,t}$ and a first order derivative of temperature with time. The heat generation term can be substituted by a volumetric heat source $\dot{Q}$. Both of these terms are multiplied by a differential volume. Finally, taking the first order Taylor series expansion to the conduction rates through the control volume and accounting for convective cooling of blood perfusion, the energy balance equation becomes:

$$\begin{aligned} \rho C_{p,t}\frac{\partial T}{\partial t}\cdot dx\cdot dy\cdot dz &= \dot{Q}\cdot dx\cdot dy\cdot dz - \frac{\partial q_x}{\partial x}\cdot dx\cdot dy\cdot dz - \frac{\partial q_y}{\partial y}\cdot dy\cdot dx\cdot dz \\ &- \frac{\partial q_z}{\partial z}\cdot dz\cdot dx\cdot dy - \dot{Q}_b\cdot dx\cdot dy\cdot dz \end{aligned} \quad (3.2)$$

Where $\frac{\partial q_x}{\partial x}dx$ is the heat conduction rate in x, (follows similarly for the y and z directions) and $\dot{Q}_b$ is a representation of the heat loss due to the perfused tissue.

The Pennes bioheat equation, shown below in Eqn.(3.3), is obtained by using Fourier's law for conduction through the control volume and a linear variation of convective heat transfer outside the control volume due to blood perfusion. Both sides of the equation are divided by the differential volume term, $dx\cdot dy\cdot dz$ to simplify the form (Incropera and DeWitt (1996)).

$$\begin{aligned} \rho C_{p,tis}(T)\frac{\partial T}{\partial t} &= \dot{Q} + \frac{\partial}{\partial x}\left[k_{tis}(T)\frac{\partial T}{\partial x}\right] + \frac{\partial}{\partial y}\left[k_{tis}(T)\frac{\partial T}{\partial y}\right] \\ &+ \frac{\partial}{\partial z}\left[k_{tis}(T)\frac{\partial T}{\partial z}\right] - \omega_{bl}C_{p,bl}(T)(T-T_{bl}) \end{aligned} \quad (3.3)$$

Both the energy storage and volumetric heating terms in Eqn. (3.3) remain unchanged in the final form of the bioheat equation. The term $\dot{Q}$ solely represents the energy added to the system via an outside heating source. While it is neglected in the following derivation, the metabolic heat generation can be added into this term.

The remaining four terms on the right hand side represent the conductive and convective heat transfer either into or out of the control volume, with k_{tis} representing the thermal conductivity of the tissue and $w_{bl}C_{p,bl}$ being a proportionality constant relating the amount of perfusion-induced convective cooling to the magnitude of temperature difference between a local area and the core body temperature, T_b. It should also be noted that this is the nonlinear form of the bioheat equation since the thermo-physical properties are temperature dependent.

Boundary and initial conditions

As with all differential equations knowing the governing equation alone is not enough to solve the problem. The above derivation was done in three dimensions to be as general as possible, however the simulations presented in Chapter 5 are two dimensional. A two dimensional model requires the definition of four boundary conditions and an initial condition to solve the problem. The various types of boundary conditions used in the HTP simulations are shown below. Each is presented on the $x = 0$ face of the domain however they can be applied at all other faces just as easily. These equations can be derived in a manner similar to the derivation of Eqn. (3.3) starting from an energy balance on the boundary of the control volume. A boundary condition of the first kind is shown in Eqn. (3.4), it is for a prescribed surface temperature.

$$T(x = 0, y, z, t) = T_\infty \tag{3.4}$$

Boundary conditions of the second kind are based on the first derivative of temperature with respect to the direction of heat flow. This type of boundary condition is written in a general form shown in Eqn. (3.5)

$$-k_{tis}(T)\frac{\partial T(x = 0, y, z, t)}{\partial x} = q_{x=0}(t) + h_{x=0}\left(T_{\infty,x} - T\right) \tag{3.5}$$

Writing the boundary condition in this form allows for both convective heat transfer

and heat transfer due to an incident heat flux. If one of these modes of heat transfer is not present at a particular boundary, then either $q_{x=0}$ or h can be set to zero. Symmetry adiabats are commonly used to reduce the size of the modeled space when applicable. In this case, the boundary condition can be used by setting both modes to zero, which is the boundary condition in the case of a side which is perfectly insulated.

3.2 Finite Difference Formulation

No analytical solution was found which solved a multi-dimensional Pennes bioheat equation with the versatility of a finite difference scheme. Given that adequate computational power is readily available and the robustness of the finite difference method, a numerical approach was chosen to solve the bioheat model. A numerical solution allows for regional differences in thermo-physical properties and can also be programmed to allow for temperature dependent thermal properties. Additionally, with a numerical solution technique any combination of the above boundary conditions can be applied in a simulation with ease. In the above derivation, The Pennes bioheat equation, Eqn. (3.3) was developed in three dimensions, however in this research two dimensional models in either Cartesian or radial coordinates are used to simulate treatment plans. The rationale behind deriving the equation in three dimensions begins to emerge in the derivation of the finite difference equations. The following finite difference scheme is derived such that either radial or cartesian coordinates could be used.

3.2.1 Discretization strategy

In a transient FD application, the program author has many choices regarding how to work through the discretization process. Before a rigorous explanation of the equations used in the FD scheme can be presented, it is essential to briefly touch on the main points of the discretization strategy. These include, linearization of the source term, the method of time and space discretization, and finally accounting for interface conductivity.

Source term linearization

The derivation begins with the Pennes bioheat equation written in the form shown below. Conduction in the z direction is ignored. The Roman numerals below each part of equation

are expanded in the *Finite Difference Equations* section below (Patankar (1980)).

$$\underbrace{\rho C_{p,tis}(T)\frac{\partial T}{\partial t}}_{I} = \underbrace{\frac{\partial}{\partial x}\left[k_{tis}(T)\frac{\partial T}{\partial x}\right]}_{II} + \underbrace{\frac{\partial}{\partial y}\left[k_{tis}(T)\frac{\partial T}{dy}\right]}_{III} + \underbrace{S_c}_{IV} - \underbrace{S_p T}_{V} \tag{3.6}$$

The terms S_c and $S_p\,T$ are a linearized form of the source term, which in this case take the place of the volumetric heat source and temperature dependent cooling due to blood perfusion. The precise definition of these terms are shown in Eqn. (3.12), showing how the source term has been broken into a temperature-dependent component and a steady component.

$$S_c = \dot{Q} + \omega_{bl}\, Cp_{bl}\, T_{bl}\,, \qquad S_p = w_{bl}\, Cp_{bl} \tag{3.7}$$

Time discretization

Due to stability considerations, a fully implicit time discretization scheme was chosen in this research. The concept of implicit discretization assumes that the values of the present time step are unknown, and so a set of simultaneous equations must be solved to find the temperature field at each time step. While this scheme is more computationally intensive to implement, it is unconditionally stable. Computational time is reduced by using the tridiagonal matrix algorithm to solve the temperature field at each time step (Patankar (1980)). Implicit time discretization is shown mathematically in Eqn. (3.8) and schematically in Fig. 3.2.

$$\int_{p}^{p+1} T_{i,j}\,dt = T_{i,j}^{p+1} \tag{3.8}$$

Spacial discretization

As mentioned above, the bioheat models used for this research are two dimensional. A nodal network in two dimensions is used in the solution of the transient FD model. Evenly spaced nodes were chosen with control volumes (CVs) formed around each, such that side nodes are surrounded by half-volume CVs and corner nodes are surrounded by one-quarter volume CVs. Each CV is assigned its own value of thermal conductivity, thermal capacitance and blood perfusion, all of which can be made temperature dependent. Figure (3.3) shows a sample two-dimensional grid.

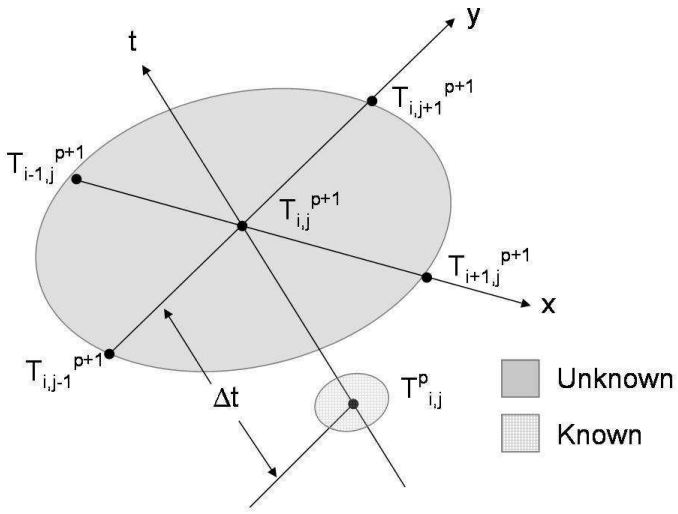

Fig. 3.2 Schematic of implicit time discretization.

Interface conductivity

During the transient heating process, the energy leaving one control volume must be accounted for as either entering another control volume or leaving the model at the boundary. Poor energy accounting will result in physically impossible results. At the interface between interior CVs, Patankar's method of interface conductivity is used (Patankar (1980)). The heat flux across the eastern boundary of an arbitrary CV (i,j) is shown in Fig. (3.4). Two different formulas for defining the amount of heat flux are defined below, see Eqns. (3.9) and (3.10). One is based on a thermal resistance network and the other assumes the interface conductivity, k_E is known. Since the discretization is done with equal spacing between control volumes the formulas is based on a single value, Δx.

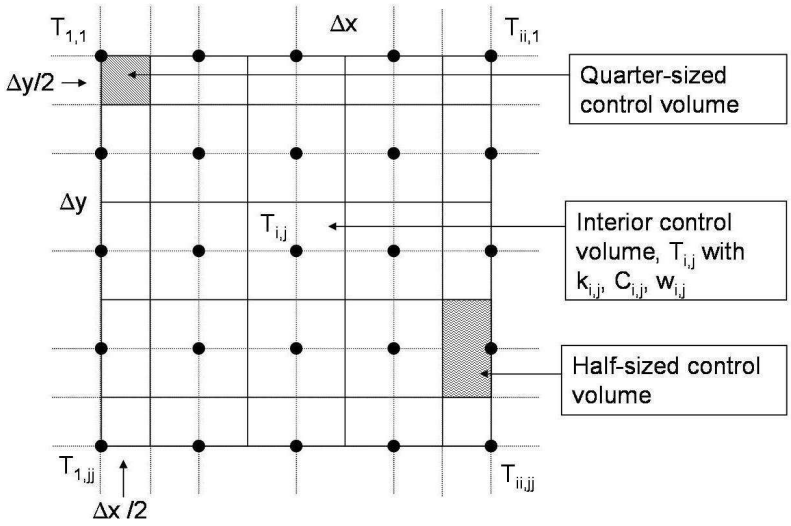

Fig. 3.3 Schematic of spacial discretization method.

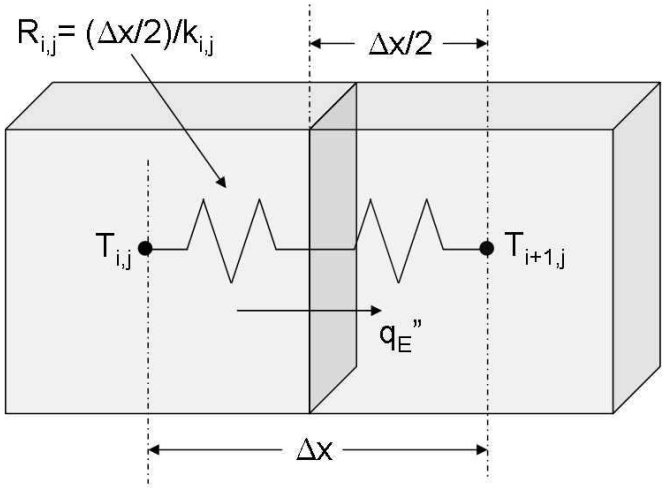

Fig. 3.4 Schematic of interface conductivity.

$$q''_E = \frac{T_{i,j} - T_{i+1,j}}{\frac{\Delta x/2}{k(T)_{i,j}} + \frac{\Delta x/2}{k(T)_{i+1,j}}} \tag{3.9}$$

$$q''_E = k_E \frac{T_{i,j} - T_{i+1,j}}{\Delta x} \tag{3.10}$$

Combining these equations and solving for k_E gives Eqn. (3.11). This is also known as the harmonic mean. In limiting cases, such as at the edge of the discretized domain k_E goes to zero. This is logical since heat cannot conduct in the absence of a medium. If both control volumes have the same conductivity, k, the harmonic mean reduces to that value. This derivation can be extended to the north, south and west directions readily and is accurate for radial coordinates with sufficiently small values of Δx. It is also important to remember that each individual control volume k can depend on the temperature of that control volume. Because of this dependence the interface conductivities are also temperature dependent but the notation is dropped for convenience.

$$k_E = \frac{2k(T)_i k(T)_{i+1}}{k(T)_{i+1} + k(T)_i} \tag{3.11}$$

3.2.2 The finite difference equations

With the above tools in place, the bioheat equation shown in Eqn. (3.6) can be fully discretized. This is done by integrating each term in the three coordinate directions as well as in time. The integrals are approximated with first order Taylor Series expansions. The following derivations are for an arbitrary interior node, $T_{i,j}$. In the integration, E, W, N, S refer to the east, west, north and south directions relative the arbitrary point (i,j). The symbols *in* and *out* refer to into and out of the page respectively. Equation (3.6) is discretized term by term, starting with term I, the storage term.

$$\int_{in}^{out} \int_{S}^{N} \int_{W}^{E} \int_{p}^{p+1} \rho C_{p,t}(T) \frac{\partial T}{\partial t} dt\, dx\, dy\, dz = (\rho C_{p,t}(T^p_{i,j}) \Delta V)_{i,j} \, (T^{p+1}_{i,j} - T^p_{i,j}) \tag{3.12}$$

Here and in subsequent equations, the subscript (i,j) denotes that this grouped quantity is particular to the individual cell, (i,j). This allows for regional differences in

thermal capacitance in the simulation. Also, $\Delta V = \Delta x\,\Delta y\,\Delta z$, the volume of the control volume.

In term II, conduction in the x direction is discretized. The same integration is applied however this time all temperatures are evaluated at the $(p+1)^{st}$ time step due to the implicit discretization scheme.

$$\int_{in}^{out}\int_{S}^{N}\int_{W}^{E}\int_{p}^{p+1}\frac{\partial}{\partial x}\left[k(T)\frac{\partial T}{\partial x}\right]\,dt\,dx\,dy\,dz = \left[\frac{k_E A_{E,(i,j)}\,(T_E^{p+1}-T_{i,j}^{p+1})}{\Delta x} - \frac{k_W A_{W,(i,j)}\,(T_{i,j}^{p+1}-T_W^{p+1})}{\Delta x}\right]\Delta t \tag{3.13}$$

The areas at the east and west edge of the control volume are shown as A_E and A_W. The precise definition of these terms are given later since they vary depending on the coordinate system used. The integration of term III is identical but the gradients are evaluated in the north-south directions. Consequently terms representing the north and south areas of the control volume, $\hat{A}_N$ and $\hat{A}_S$ respectively are present on the left hand side of the equation. The result of this integration is shown below which is nearly identical to the above equation. Again the definition of the area terms is given later.

$$\int_{in}^{out}\int_{S}^{N}\int_{W}^{E}\int_{p}^{p+1}\frac{\partial}{\partial y}\left[k(T)\frac{\partial T}{\partial y}\right]\,dt\,dx\,dy\,dz = \left[\frac{k_N A_{N,(i,j)}\,(T_N^{p+1}-T_{i,j}^{p+1})}{\Delta y} - \frac{k_S A_{S,(i,j)}\,(T_{i,j}^{p+1}-T_S^{p+1})}{\Delta y}\right]\Delta t \tag{3.14}$$

The integration of term IV is somewhat simpler since there are no gradients to evaluate. If the volumetric source changes in time this must be accounted for in the integration. The previous time step value of the source term is used. The discretized equation is shown below, with Eqn. (3.7) substituted for S_c.

$$\int_{in}^{out}\int_{S}^{N}\int_{W}^{E}\int_{p}^{p+1}\left[\dot{Q}+w_b\,Cp_b\,T_b\right]\,dt\,dx\,dy\,dz = (\dot{Q}_{i,j}^{p}+w_b\,Cp_b\,T_b)\,\Delta V_{i,j} \tag{3.15}$$

Finally term V is integrated. This term is a temperature dependent source and so care must be given to integrate the time component properly. Since the implicit method is used, the temperature is again evaluated at the $(p+1)^{st}$ time step.

$$\int_{in}^{out}\int_{S}^{N}\int_{W}^{E}\int_{p}^{p+1} w_b\, Cp_b\, T dt\, dx\, dy\, dz = (w_b\, Cp_b\, T_{i,j}^{p+1})\, \Delta V_{i,j} \tag{3.16}$$

With all terms of the bioheat equation discretized, they can be combined and rearranged to form one governing equation, Eqn. (3.17). This equation gives the temperature dependence of node (i,j) on its immediate neighbors and volumetric sources from one time step to the next. Several new terms are introduced which simplify the equation's presentation. Each is defined subsequently.

$$\begin{aligned} a_{i,j}\, T_{i,j}^{p+1} &= aE_{(i,j)} T_{(i+1,j)}^{p+1} + aW_{(i,j)} T_{(i-1,j)}^{p+1} \\ &+ aN_{(i,j)} T_{(i,j+1)}^{p+1} + aS_{(i,j)} T_{(i,j-1)}^{p+1} + b \end{aligned} \tag{3.17}$$

The term, $a_{i,j}$ is a collection of terms which are constant over the course of one time step, Δt. The definition of these terms are shown below, with the exception of S_p which was previously defined in Eqn. (3.7).

$$a_{i,j} = aO_{i,j} + aE_{i,j} + aW_{i,j} + aS_{i,j} + aN_{i,j} + S_p\, \Delta V_{i,j} \tag{3.18}$$

$$aE_{i,j} = \frac{(A_E\, k_E)_{(i,j)}}{\Delta x} \tag{3.19}$$

$$aW_{i,j} = \frac{(A_W\, k_W)_{(i,j)}}{\Delta x} \tag{3.20}$$

$$aN_{i,j} = \frac{(A_N\, k_N)_{(i,j)}}{\Delta y} \tag{3.21}$$

$$aS_{i,j} = \frac{(A_S\, k_S)_{(i,j)}}{\Delta y} \tag{3.22}$$

$$aO_{i,j} = \frac{(\rho\, Cp\, \Delta V)_{(i,j)}}{\Delta t} \tag{3.23}$$

The only term left to define in Eqn. (3.17) is b. This term is a collection of terms that are fully known at the beginning of the $(p+1)^{st}$ time step.

$$b = \underbrace{aO_{i,j}}_{c}\, T_{(i,j)}^{p} + \underbrace{S_{c\,(i,j)}^{p}\, \Delta V_{i,j}}_{\hat{b}} \tag{3.24}$$

The b term is further broken into two terms c and $\hat{b}$. While separating these terms may seem unprofitable at the moment, they become useful during the alternating direction method which is used to solve the two-dimensional temperature field. In this method both

take on terms which are normal to the sweep direction. The term c takes all terms which are multiplied by the node for which the equation is formulated. The term $\hat{b}$ plays a similar role but takes terms which can be considered constant over the time step. In the case of end nodes, both will also contain boundary condition information. This is introduced now to facilitate the understanding of the solution process applied to solve model, but a much more in-depth explanation is found below in the *Finite Difference Solution Technique* section.

Coordinate system specification

Until now the area terms, A_E, A_W, etc. have not been defined. At this point in the derivation it is essential to explain the purpose for including these terms. After all, if the model is only two-dimensional what is the purpose of integrating the bioheat equation in a direction normal to the page, thereby defining the areas normal to the heat flux in the x and y directions? The reason is that by simply changing how the area terms are defined, the model can be changed from Cartesian to cylindrical coordinates. Figures 3.5 and 3.6 illustrate the difference between a 2D cartesian and a 2D cylindrical coordinate system.

First we consider the simpler case of the cartesian coordinate system, Fig. 3.5, where conduction has been assumed to only occur in the $x-y$ plane. In this case a unit thickness for each control volume in the z direction is assumed, thereby defining the areas as:

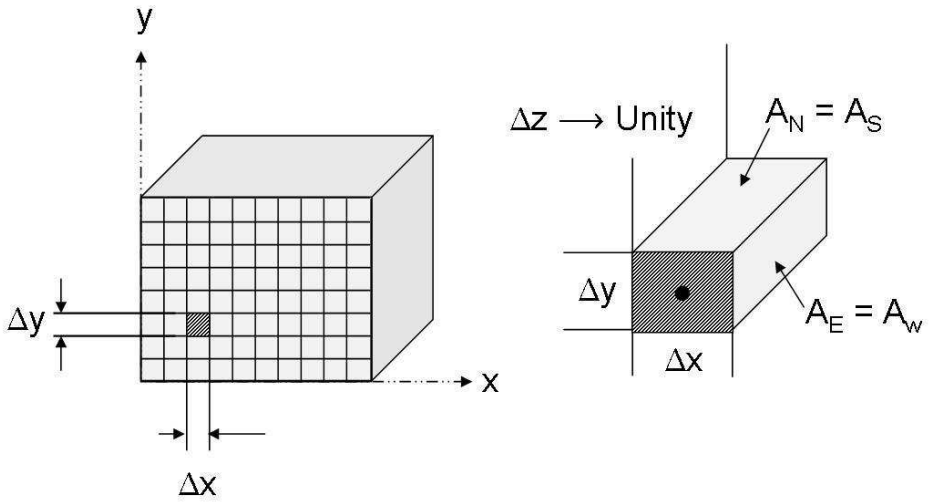

Fig. 3.5 Cartesian coordinate system with representative areas.

$$A_{E,(i,j)} = A_{W,(i,j)} = \Delta y \tag{3.25}$$

$$A_{N,(i,j)} = A_{S,(i,j)} = \Delta x \tag{3.26}$$

In cylindrical coordinates each control volume is ring-shaped, with the nodes along lying on a single arbitrary plane defined by the x and y axis. The location of the plane is arbitrary because there is no variation in temperature angularly around the center axis. An important number used in the calculation of the areas is the radius at the eastern edge of each control volume. This radius, along with each area can be see in Fig. 3.6.

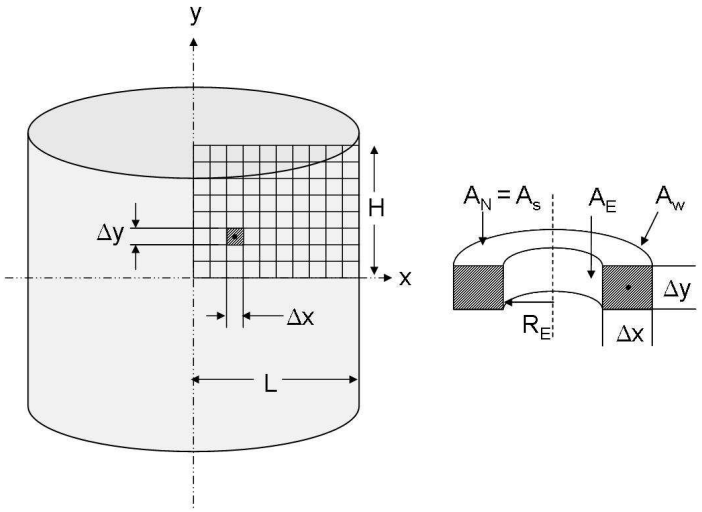

Fig. 3.6 Cylindrical coordinate system with representative areas.

$$r_{E,(i,j)} = \begin{cases} 0 & , \quad \text{for i} = 1 \\ \Delta x(\frac{1}{2} + i) & , \quad \text{for i} > 1 \quad \text{and} \quad 1 < j < N_y \end{cases} \tag{3.27}$$

With the eastern edge radius defined the corresponding areas are calculated according to the following equations. Note that the half width control volumes play an important role in determining the areas. Also at $r_E = 0$ the eastern area of the control is zero. Unlike the east and west control volume areas, the north and south counterparts are equal.

$$A_{E,(i,j)} = 2\,\pi \Delta y\, r_{E,(i,j)} \tag{3.28}$$

$$A_{W,(i,j)} = \begin{cases} 2\,\pi\Delta y\,\frac{\Delta x}{2} & , \quad \text{i} = 1 \\ 2\,\pi\Delta y\,(r_{E,(i,j)} + \Delta x) & , \quad \text{i} > 1 \\ 2\,\pi\Delta y\,(r_{E,(i,j)} + \frac{\Delta x}{2}) & , \quad \text{i} = N_x \end{cases} \tag{3.29}$$

$$A_{N,(i,j)} = A_{S,(i,j)} = \begin{cases} \pi(\frac{\Delta x}{2})^2 & , \quad \text{i} = 1 \\ \pi((r_{E,(i,j)} + \Delta x)^2 - r_{E,(i,j)}^2) & , \quad \text{i} > 1 \\ \pi((r_{E,(i,j)} + \frac{\Delta x}{2})^2 - r_{E,(i,j)}^2) & , \quad \text{i} = N_x \end{cases} \tag{3.30}$$

Applying boundary conditions

The boundary conditions described in the above section must also be included in the finite difference equations. To do so, we must revisit Eqn. (3.13) which shows how conduction in the east-west direction is discretized. Since the boundary conditions were developed for the $x = 0$ face of the domain we will use this as an example. In the case of a boundary condition of the second kind, heat transfer due to conduction on the western face is replaced by heat transfer via an external heat flux or convection. Substituting Eqn. (3.5) in Eqn. (3.13) gives:

$$\int_{in}^{out}\int_{S}^{N}\int_{W}^{E}\int_{p}^{p+1}\frac{\partial}{\partial x}\left[k(T)\frac{\partial T}{\partial x}\right]dt\,dx\,dy\,dz = \tag{3.31}$$
$$\frac{k_E A_{E,(i,j)}\,(T_E^{p+1} - T_{i,j}^{p+1})}{\Delta x} + \left(A_{W,(i,j)}\,q_{x=0} + A_{W,(i,j)}\,h_{x=0}(T_{\infty,\,x=0} - T_{i,j}^{p+1})\right)\Delta t$$

In the special case of a constant surface heat flux, Eqn. (3.17) is modified to "pin" the surface node to a specified temperature, in this case $T_{x=0}$. This is done by setting $a_{i,j} = 1$, setting all the directional coefficients, aE, aW, etc. to zero and setting $b = T_{x=0}$. This case is shown below.

$$T_{i,j}^{p+1} = T_{x=0} \tag{3.32}$$

Solution technique

The alternating direction method was used to solve for the transient temperature field. This method was employed by using orthogonal "sweeps" which solved for the temperature in the x and y directions each over one half time step. With this method, after both sweeps were finished, one full time step had passed.

The sweeps are performed so that the two dimensional problem can be broken down to look like a one dimensional problem. In order to do so, the temperatures at nodes orthogonal to the sweep direction are assumed to be equal to their value at the previous time step. For a horizontal sweep, this would mean that all nodes north and south of the line of interest would be assumed to have the same temperatures as at the previous (p^{th}) time step. This assumption allows the use of the Tridiagonal Matrix algorithm (TDMA) to be used to solve for the temperatures at the line of interest half a time step later. Marching line by line, the temperature field at the $(p+\frac{1}{2})^{th}$ time step is then known. Equations (3.33) through (3.36) shows in detail how the horizontal sweep is evaluated for an interior node. Note that the aO term is multiplied by 2, accounting for the half time step. Refer to Eqns. (3.17) through (3.23) to see the origin of these terms.

$$-aW_{(i,j)}T^{p+\frac{1}{2}}_{(i-1,j)} + ahor_{i,j}\,T^{p+\frac{1}{2}}_{i,j} - aE_{(i,j)}T^{p+\frac{1}{2}}_{(i+1,j)} = bhor_{(i,j)} \tag{3.33}$$

$$\begin{aligned} ahor_{i,j} &= 2\,aO_{i,j} + aE_{i,j} + aW_{i,j} + (S_p\,\Delta V)_{i,j} && (3.34)\\ bhor_{(i,j)} &= \underbrace{(2\,aO_{i,j} - aN_{(i,j)} - aS_{(i,j)})}_{chor}\,T^{p}_{(i,j)} && (3.35)\\ &+ \underbrace{S^{p}_{c\,(i,j)}\,\Delta V_{i,j} + aN_{(i,j)}T^{p}_{(i,j+1)} + aS_{(i,j)}T^{p}_{(i,j-1)}}_{\hat{b}hor} \end{aligned}$$

The vertical sweep equations are shown below. It is performed similarly to the horizontal sweep, but it assumes that the east and west nodal temperatures are the values calculated in the horizontal sweep. This time the TDMA is used to solve for temperatures along vertical lines. After another half time step the temperature field at the $p+1^{st}$ time step is known and the algorithm is ready for another horizontal sweep.

$$-aN_{(i,j)}T^{p+1}_{(i,j+1)} + avert_{i,j}\,T^{p+1}_{i,j} - aS_{(i,j)}T^{p+1}_{(i,j-1)} = bvert_{(i,j)} \tag{3.36}$$

$$\begin{aligned} avert_{i,j} &= 2\,aO_{i,j} + aN_{i,j} + aS_{i,j} + (S_p\,\Delta V)_{i,j} && (3.37)\\ bvert_{(i,j)} &= \underbrace{(2\,aO_{i,j} - aE_{(i,j)} - aW_{(i,j)})}_{cvert}\,T^{p+\frac{1}{2}}_{(i,j)} && (3.38)\\ &+ \underbrace{S^{p}_{c\,(i,j)}\,\Delta V_{i,j} + aE_{(i,j)}T^{p+\frac{1}{2}}_{(i+1,j)} + aW_{(i,j)}T^{p+\frac{1}{2}}_{(i-1,j)}}_{\hat{b}vert} \end{aligned}$$

It should be noted here that the equations for the boundary nodes will vary slightly because of the boundary conditions. Please refer to Appendix A to see the eight remaining boundary equations needed to fully program the FD method.

3.3 Finite Difference Validation

Once programmed the finite difference scheme was checked two different ways against both steady state and transient analytical solutions. The results from each test show that the agreement is nearly perfect in both cases.

Steady state validation

The code was checked in steady state by applying a uniform surface heat flux to the bottom ($y = 0$) of a cylindrical model. The sides of the cylinder were considered adiabatic so that all the incident heat would be forced out the top of the cylinder. A convective boundary condition was placed at the top and therefore the surface temperature ($y = H$) and the bottom temperatures could be predicted. The equations used for the surface temperature and the base temperature are shown below. Table 3.2 shows the values that were used in the steady state and transient tests. Figure 3.7 shows the results from analytical and FD solution. There is perfect agreement within computational accuracy.

$$T_s = T_\infty + \frac{q"}{h} \tag{3.39}$$

$$T_b = T_s + \frac{q" L}{k} \tag{3.40}$$

Table 3.1 Values used in steady state FD test

Parameter	Value	Units
q"	1000	W/m^2
k	0.56	$W/m\,K$
α	1	m^2/s
L	10	mm
h	10	$W/m^2\,K$
T_∞	300	K
T_s	400	K
T_b	417	K

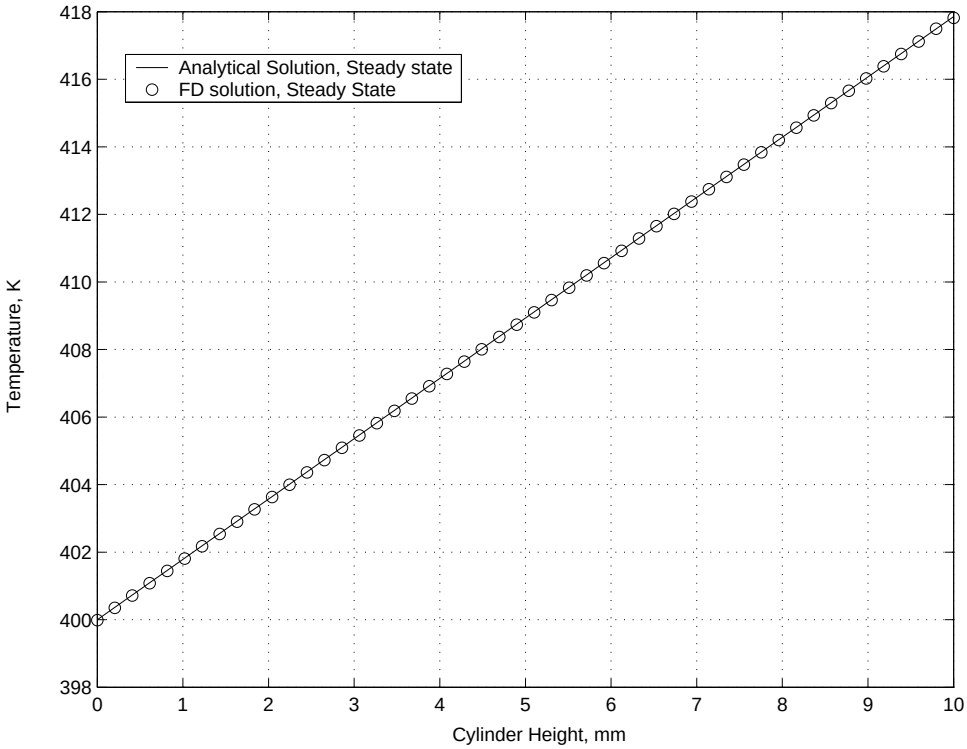

Fig. 3.7 Comparison of FD and analytical steady state temperatures.

Transient validation

The transient validation was done by comparing the results from the analytical solution of a constant surface heat flux case and the corresponding FD output. The solution for this case is readily available and is shown below (Incropera and DeWitt (1996)),(Carslaw and Jaeger (1959)).

Table 3.2 Values used in transient FD test

Parameter	Value	Units
q”	1000	W/m^2
k	0.56	$W/m\,K$
α	$1.34 \cdot 10^{-4}$	m^2/s
L	40	mm
T_0	300	K

$$T(x,t) - T_o = \frac{2\,q"\sqrt{\alpha\,t/\pi}}{k}\,\exp\left(\frac{-x^2}{4\alpha\,t}\right) - \frac{q"\,x}{k}\,\mathrm{Erfc}\left(\frac{x}{2\sqrt{\alpha\,t}}\right) \tag{3.41}$$

On the surface with incident heat flux and constant heat flux boundary was employed. Orthogonal boundaries were adiabatic. In Figs.3.8 and 3.9 results for both tran-

sient temperature profiles within the material at various times and through time at various locations show perfect agreement to machine accuracy. Since this is only a one-dimensional check, the surface heat flux was applied to both the $y = 0$ face and the $x = 0$ face. The results both show perfect agreement between the analytical and FD models.

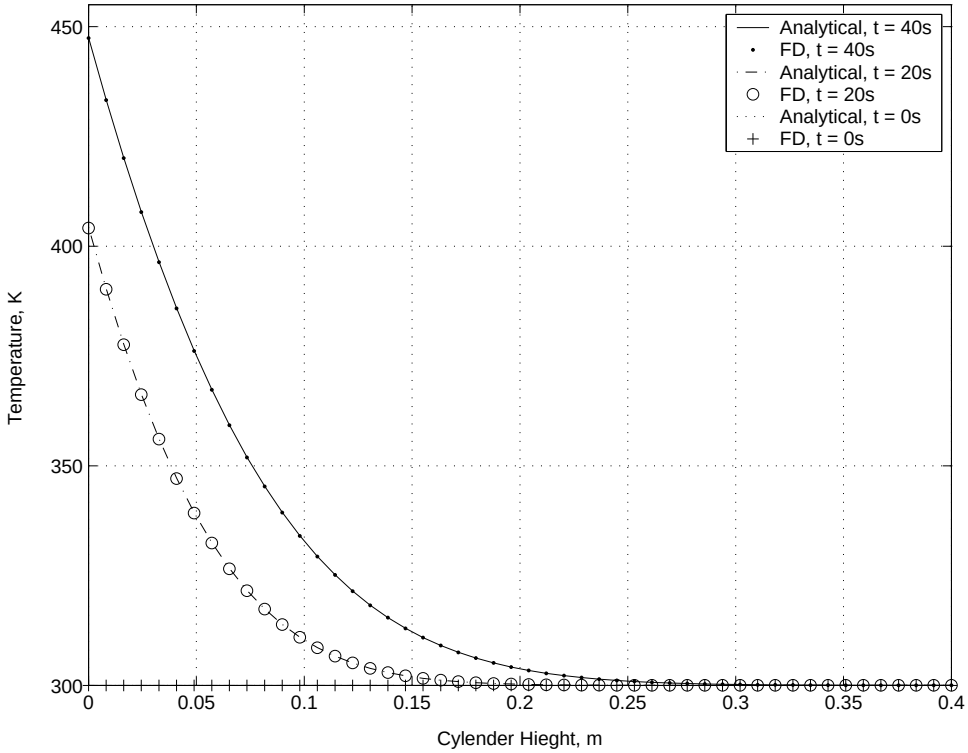

Fig. 3.8 Transient temperature comparison throughout material at various times.

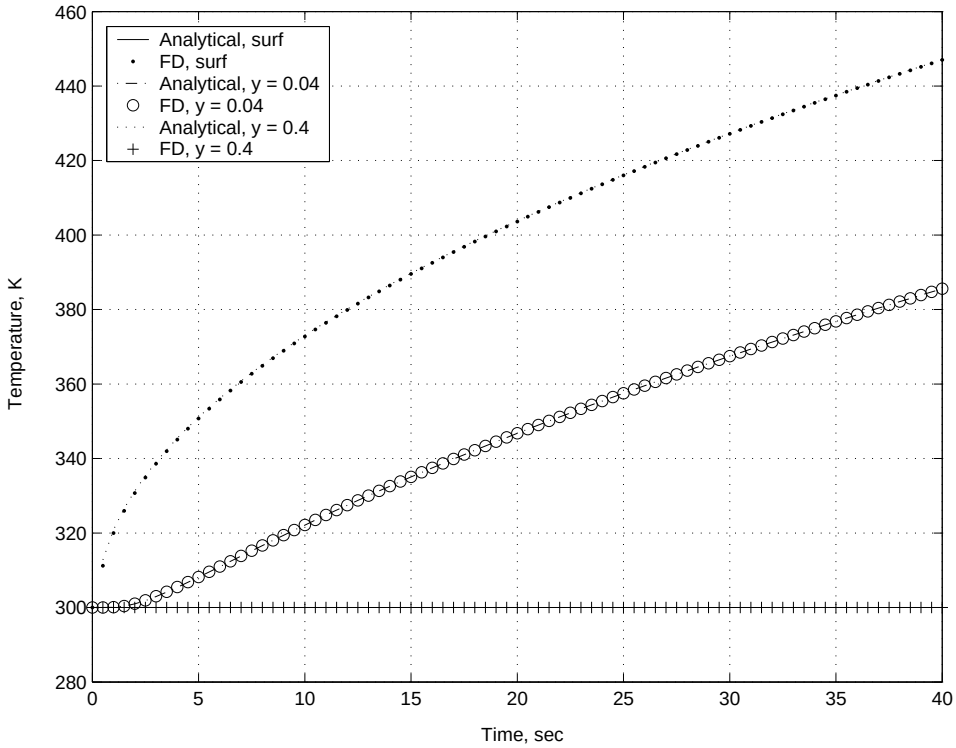

Fig. 3.9 Transient temperature comparison at various locations through time.

3.4 Thermal Damage Models

Two different means of quantifying the thermal exposure of the material in which a treatment is done were used in the optimization model. These will be introduced in this section. The first is a first-order rate process based using the Arrhenius equation (Pfefer et al. (2000)) ,(Henriques and Moritz (1947)). The equation, shown below, is also known as the Henriques damage model.

$$\Omega(x,y) = \int_0^{t_f} A \exp\left(\frac{-E}{\Re T(x,y,t)}\right) dt \tag{3.42}$$

This equation models the response of a biological material to prolonged temperature exposure. The variables in this equation, E and A are the activation energy and molecular frequency factor. They can be found in the literature for various materials but their values vary widely. In this equation, $\Re$ is the universal gas constant. The parameter Ω is used to quantify the thermal damage where the threshold of permanent damage is assigned a value of 1 (Pearce and Thomsen (1992)). Temperatures used in this formula must be in absolute units.

The Arrhenius parameters can be found via constant temperature tests where a small sample of the material is placed in a bath of water at a precise temperature. The time required for the material to visibly denature is recorded and at this point Ω is said to equal 1. After taking several time temperature pairs, the curve can be fit via least squares regression to the parameters E and A.

The second damage model used in this work is the thermal dose. The thermal dose was developed as a means of comparing different heat treatments by referencing all treatments to an exposure time at 43 ^{o}C. Similar to the damage model, it is also based on an integrated time-temperature response.

$$D(x,y) = \int_0^{t_f} R(T)^{\overline{T}(x,y,t)} dt \tag{3.43}$$

The parameter R in this case is not the universal gas constant but a parameter based on the activation energy. In accordance with the literature, the values of R in this work were chosen as R = 0.5 for temperatures greater than the 43 ^{o}C reference temperature and R = 0.25 for temperatures below 43^{o}C (Sapareto and Dewey (1984)). In this formula, the variable $\overline{T} = 43 - T(x,y,t)$. This is simply a shorthand notation. Around the "break" temperature

of 43^{o}C, the value of R is linearized so that sharp discontinuities in the optimization program can be avoided. The values of ϵ are small real values of temperature, taken to be in 0.5^{o}C in this paper (Loulou and Scott (2002)). This linearization is shown in Fig. 3.10 (Loulou and Scott (2002)).

Each time the amount of thermal damage from a simulated treatment is desired, these two damage models must be integrated. This integration is performed numerically in MATLAB using the trapezoidal method.

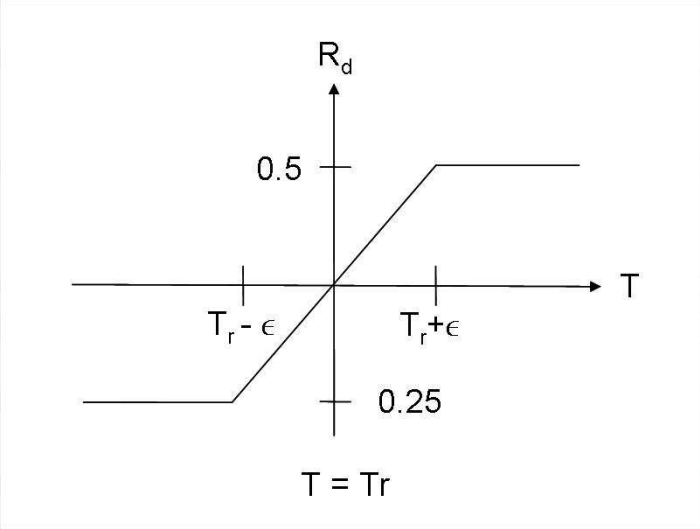

Fig. 3.10 Linearization of $R(T)$ value in thermal dose calculation.

Chapter 4

Control Problem Formulation

The backbone of the simulated HTP of this work begins here, in the formulation of the optimal control problem. The problem is solved via use of the conjugate gradient method (CGM), and the calculus of variations. CGM is used to find the gradient of the objective function while the calculus of variations is the name given to the theory of optimizing integrals. In this particular case, the integral being optimized (minimized) is the objective function, which is composed of one of the two damage models introduced in the last chapter. This chapter begins with two fundamental building blocks of the control problem. The first is the direct problem, which is the Pennes bioheat equation combined with the chosen damage model and the second is the objective function, both of which were introduced in the previous chapter. Following this the conjugate gradient method is introduced and the importance of the gradient of the objective function is underscored. The various intermediate equations needed to find the gradient of the objective function are then derived. Finally, the chapter ends with a step by step outline and flowchart of how to solve this minimization problem.

4.1 The Direct Problem Formulation

The direct problem is used each iteration to solve for the transient temperature field and damage model. The problem statement is below, along with a set of boundary conditions which completely define the problem. The boundary conditions used in this derivation are either adiabatic or constant temperature with the exception of the boundary which is the control function. Equations (4.2)-(4.7) along with either Eqn. (3.42) or (3.43) comprise the

complete mathematical definition of the problem. A schematic of what is outlined by this set of differential equations is shown in Fig. 4.1. It should be noted that the derivation could have been done for a convective or adiabatic boundary conditions on the outer edges as well.

$$C(T)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left[k(T)\frac{\partial T}{\partial x}\right] + \frac{\partial}{\partial y}\left[k(T)\frac{\partial T}{dy}\right]\ldots \quad (4.1)$$
$$-W(T)\,(T - T_\infty) + S(x,y,t)$$

$$-k(T)\frac{\partial T(x,t)}{\partial y} = q(t) \qquad 0 < x \le r_h\,,\; y = 0\,,\; t < t_o \quad (4.2)$$

$$\frac{\partial T(x,t)}{\partial y} = 0 \qquad r_h < x < L\,,\; 0\,,\; t < t_o \quad (4.3)$$

$$T(x,t) = T_\infty \qquad x\,,\; y = H\,,\; t < t_o \quad (4.4)$$

$$\frac{\partial T(y,t)}{\partial x} = 0 \qquad x = 0\,,\; y\,,\; t < t_o \quad (4.5)$$

$$T(y,t) = T_\infty \qquad x = L\,,\; y\,,\; t < t_o \quad (4.6)$$

$$T(x,y) = T_\infty \qquad x\,,\; y\,,\; t = t_o \quad (4.7)$$

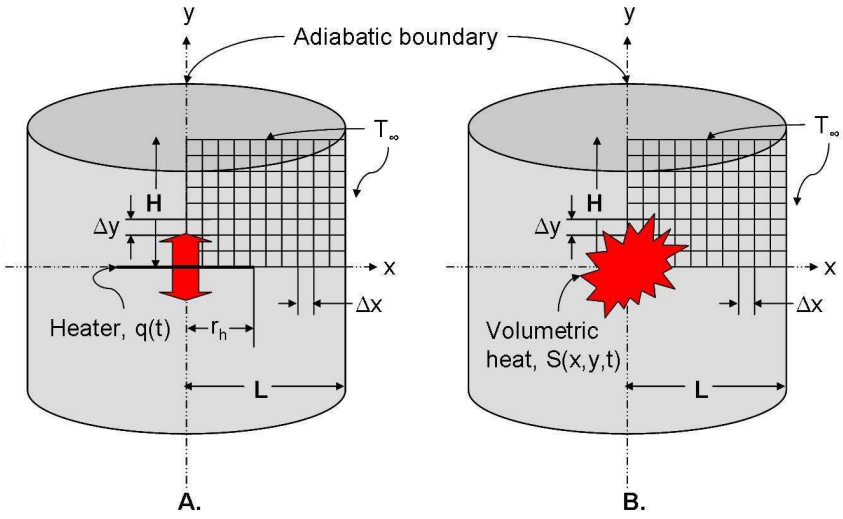

Fig. 4.1 Schematic of the direct problem and boundary conditions for (A) boundary heating and (B) volumetric heating. Boundary heating is used in the validation experiment and volumetric heating is used for simulations. A damage model (Eqn.3.42 or 3.43) must be included to calculate the extent of thermal damage

The direct problem is shown here in two dimensions; for the cylindrical case shown in the schematic the x direction is radial and the y direction is axial. The internal convection

and heat capacity terms have compacted notations, $W = w_b C_{p,b}$ and $C = \rho\, C_{p,t}$. Two options for the control function are shown in this set of equations. The first, $q(t)$ is an incident heat flux over a portion of the $x = 0$ boundary; the second is a controlled volumetric heating within the domain, $S(x, y, t)$. The the boundary heat flux which appears in Eqn. (4.2) is used in the experiment as the control parameter. Results simulations performed using $S(x, y, t)$ as the control parameter are shown in the next chapter. Please refer to Appendix B for additional development of the control problem using $S(x, y, t)$ as the control function. Henceforth in the derivation of the control problem the symbol $\Gamma(t)$ is used for a generalized control function.

4.2 Objective Function Formulation

The objective function is so called because it is what the optimization problem aims to minimize. It is composed of the either the damage integral, Eqn. (3.41) or the thermal dose, Eqn. (3.42). This function is the squared and summed difference between the damage field calculated from the current guess of the control function and the desired damage field. In its simplest sense, it is an indicator as to how close the control problem is to a final solution. As the objective function approaches zero, the control function is providing results which approach the desired optimum. The objective function is shown below, in Eqn. (4.8). Here a generalized field term, Λ is introduced which represents either the damage or dose integrals. This will be used in the derivation of the control problem when the field used is arbitrary. For the case of the thermal damage coefficient, $\Lambda_c = \Omega_c$ found in Eqn. (3.42). For the thermal dose, $\Lambda_c = D_c$, found in Eqn. (3.43). In all cases, the subscripts c and d stand for calculated and desired field respectively. Note the objective function is a function only of the control variable, $\Gamma(t)$.

$$\mathcal{J}(\Gamma(t)) = \int_0^H \int_0^L [\Lambda_c - \Lambda_d]^2 \, dx\, dy \quad , \quad \text{General Model} \tag{4.8}$$

4.3 Conjugate Gradient Method

The conjugate gradient method is used to minimize the objective function, $J(\Gamma(t))$. It is an iterative procedure, which updates a previous guess of the value of the control function

as shown in Eqn. (4.9).

$$\Gamma^{(s+1)}(t) = \Gamma^{(s)}(t) + \gamma^{(s)}\, D^{(s)}(t) \tag{4.9}$$

In this equation, the scalar term γ is known as the decent parameter and will be defined later, it is multiplied by $D(t)$, the decent direction, a vector whose equation is shown below. The superscript s indicates the iteration number.

$$D^{(s)}(t) = \mathcal{J}'^{(s)}(t) + \beta^{(s)}\, D^{(s-1)}(t) \tag{4.10}$$

The decent direction adds the value of the current objective function gradient to the product of the conjugate parameter, β and the previous decent direction. The conjugate parameter is shown below, it is a conjugation of the current objective function gradient and the objective function gradient of the previous iteration.

$$\beta^{(s)} = \frac{\langle \mathcal{J}'^{(s)}, \mathcal{J}'^{(s)} - \mathcal{J}'^{(s-1)} \rangle}{\langle \mathcal{J}'^{(s-1)}, \mathcal{J}'^{(s-1)} \rangle} \quad \text{with} \quad \beta^{(0)} = 0 \tag{4.11}$$

In order to perform the iterative procedure outlined above in Eqns. (4.9) through (4.11), two terms must be solved for; the decent parameter, $\gamma^{(s)}$ and the gradient of the objective function, $\mathcal{J}'^{(s)}$. The next part of this chapter is devoted to this task.

4.3.1 The gradient of the objective functional

To find the gradient of the functional an additional functional must be introduced, the Lagrangian of the system. The Lagrangian is comprised of the objective function as well additional equality constraint terms that come from the direct problem. Each constraint, from Eqns. (4.2) through (4.7) is multiplied by a Lagrange multiplier, $\Psi_{(1-7)}$.

$$\begin{aligned}
\mathcal{L} &= \int_0^H \int_0^L [\Lambda_c - \Lambda_d]^2 \, dx\, dy \\
&+ \int_0^H \int_0^L \int_0^{t_f} \left[\frac{\partial}{\partial x}\left[k(T) \frac{\partial T}{\partial x} \right] + \frac{\partial}{\partial y}\left[k(T) \frac{\partial T}{\partial y} \right] \dots \right. \\
&\qquad \left. -W(T)(T - T_b) + S(x,y,t) - C(T)\frac{\partial T}{\partial t} \right] \Psi_1(x,y,t)\, dt\, dx\, dy \\
&+ \int_0^{r_h} \int_0^{t_f} \left[k(T)\frac{\partial T}{\partial y} + q(t) \right] \Psi_2(x,t)\, dx\, dt \\
&+ \int_{r_h}^{L} \int_0^{t_f} \left[0 + \frac{\partial T}{\partial y} \right] \Psi_3(x,t)\, dx\, dt
\end{aligned}$$

$$
\begin{aligned}
&+ \int_0^L \int_0^{t_f} \left[T_\infty - T(x,t)\right]\, \Psi_4(x,t)\, dx\, dt \\
&+ \int_0^H \int_0^{t_f} \left[0 + \frac{\partial T(y,t)}{\partial x}\right]\, \Psi_5(y,t)\, dy\, dt \\
&+ \int_0^H \int_0^{t_f} \left[T_\infty - T(y,t)\right]\, \Psi_6(y,t)\, dy\, dt \\
&+ \int_0^H \int_0^{L} \left[T_0 - T(x,y,0)\right]\, \Psi_7(x,y)\, dx\, dy
\end{aligned} \tag{4.12}
$$

Next, the stationary points of equation (4.12) must be found, this is done by choosing the lagrange multipliers such that:

$$\Delta\mathcal{L} = 0 \tag{4.13}$$

The first variation of the Lagrangian, $\Delta\mathcal{L}$, is found by finding the variations of $\mathcal{J}(\Gamma(t))$ and the equality constraints. Once each variation is found, the proper choice of multipliers to make Eqn. (4.13) valid can be derived. First, we must introduce the variations of the objective function and each of the constraints.

The variations of the objective function and direct problem

A small, arbitrary perturbation in the control function noted $\varepsilon\Delta\Gamma(t)$ will cause the temperature within the system as well as all temperature dependent variables to change by a small amount. This idea is stated mathematically below.

$$
\begin{aligned}
\Gamma_\varepsilon(t) &= \Gamma(t) + \varepsilon\Delta\Gamma(t) && (4.14)\\
T_\varepsilon(x,y,t) &= T(x,y,t) + \varepsilon\Delta\, T(x,y,t) && (4.15)\\
k(T_\varepsilon) &= k(T) + \varepsilon\, \frac{\partial k}{\partial T}\Delta T && (4.16)\\
C(T_\varepsilon) &= C(T) + \varepsilon\, \frac{\partial C}{\partial T}\Delta T && (4.17)\\
W(T_\varepsilon) &= W(T) + \varepsilon\, \frac{\partial W}{\partial T}\Delta T && (4.18)\\
\Lambda_\varepsilon(x,y) &= \Lambda_c(x,y) + \varepsilon\, \Delta\Lambda_c(x,y) && (4.19)
\end{aligned}
$$

To find the variation of the parameters is it also necessary to define a limiting operator, $\mathcal{O}$ which is shown below. This limiting operator finds the variation of a parameter when $\varepsilon \to 0$.

$$\mathcal{O} = \lim_{\varepsilon \to 0} \frac{\mathcal{O}_\varepsilon - \mathcal{O}}{\varepsilon} \tag{4.20}$$

By perturbing the control function, and then applying the limiting operator to both the objective function and the direct problem, a formula for the variation of the Lagrangian begins to develop. Below, the variation of the objective function is shown for the general model.

$$\Delta \mathcal{J}(\Gamma(t)) \quad = \quad 2 \int_0^H \int_0^L \left[\Lambda_c - \Lambda_d\right] (\Delta \Lambda_c) \, dx \, dy \quad , \quad \text{General Model} \tag{4.21}$$

Recall that in this case Λ_c can be either Henriques damage model or the thermal dose. The variation term, $\Delta\Lambda_c$ is defined separately for damage and dose optimization below. Equations (3.42) and (3.43) are used to calculate the calculated field depending on whether the damage or dose integrals are used.

$$\Delta \Omega_c \quad = \quad \frac{AE}{\Re} \int_0^{t_f} \exp\left(\frac{-E}{\Re \, T(x,y,t)}\right) dt \frac{1}{T^2} \, \Delta T(x,y,t) \, dt \, dx \, dy \tag{4.22}$$

$$\Delta D_c \quad = \quad \int_0^{t_f} R^{\overline{T}(x,y,t)} \left[(\overline{T}(x,y,t)) \frac{1}{R} \frac{\partial R}{\partial T} - \ln R \right] \Delta T(x,y,t) \, dt \, dx \, dy \tag{4.23}$$

The same limiting process is then applied to each of the terms of the direct problem, Eqns. (4.2) through (4.7). The process is shown in detail for the left hand side of Eqn. (4.2), the thermal storage term. The definition of the perturbed thermal storage and temperatures can be found in Eqn. (4.17)

$$C(T_\varepsilon) \frac{\partial T_\varepsilon}{\partial t} \quad = \quad \left[C(T) + \varepsilon \frac{\partial C}{\partial T} \Delta T \right] \cdot \left[\frac{\partial T}{\partial t} + \varepsilon \frac{\partial \Delta \, T}{\partial t} \right] \tag{4.24}$$

$$= \quad C(T) \frac{\partial T}{\partial t} + \varepsilon C(T) \frac{\partial \Delta \, T}{\partial t} + \varepsilon \frac{\partial C}{\partial T} \frac{\partial T}{\partial t} \Delta \, T + \varepsilon^2 \frac{\partial C}{\partial T} \frac{\partial T}{\partial t} \Delta \, T \tag{4.25}$$

After applying the limiting operator the results are then written in compacted form.

$$lim_{\varepsilon \to 0} \frac{\mathcal{O}_\varepsilon - \mathcal{O}}{\varepsilon} \quad = \quad C(T) \frac{\partial \Delta \, T}{\partial t} + \frac{\partial C}{\partial T} \frac{\partial T}{\partial t} \Delta \, T \tag{4.26}$$

$$= \quad \frac{\partial \left[C(T) \Delta T \right]}{\partial t} \tag{4.27}$$

The above process is applied to all terms in the direct problem as well as each boundary condition. The end result is the variation problem, shown below in Eqns. (4.29) through (4.34).

$$\begin{aligned}
\frac{\partial[C(T)\,\Delta T]}{\partial t} &= \frac{\partial^2\,[k(T)\Delta\,T]}{\partial x^2} + \frac{\partial^2\,[k(T)\Delta\,T]}{\partial y^2}\cdots \\
&\quad -W(T)\,\Delta\,T + \Delta\,S(x,y,t) && (4.28)\\
-\frac{\partial\,[k(T)\Delta T(x,t)]}{\partial y} &= \Delta\,q(t) \qquad 0 < x \le r_h\,,\; y = 0\,,\; t > t_o && (4.29)\\
\frac{\partial\Delta\,T(x,t)}{\partial y} &= 0 \qquad r_h < x < L\,,\; 0\,,\; t > t_o && (4.30)\\
\Delta\,T(x,t) &= 0 \qquad x\,,\; y = H\,,\; t > t_o && (4.31)\\
\frac{\partial\Delta\,T(y,t)}{\partial x} &= 0 \qquad x = 0\,,\; y\,,\; t > t_o && (4.32)\\
\Delta\,T(y,t) &= 0 \qquad x = L\,,\; y\,,\; t > t_o && (4.33)\\
\Delta\,T(x,y) &= 0 \qquad x\,,\; y\,,\; t = t_o && (4.34)
\end{aligned}$$

With the variation problem and the variation of the objective function fully defined, it is possible to fully define the variation of the Lagrangian, $\Delta\mathcal{L}$. It follows precisely the same form as the Lagrangian of the system, Eqn. (4.12), however in all cases the equations are replaced with their counterpart in variations.

$$\begin{aligned}
\Delta\mathcal{L} &= \int_0^H\int_0^L \Delta\Lambda_c\,[\Lambda_c - \Lambda_d]\;dx\,dy \\
&+ \int_0^H\int_0^L\int_0^{t_f} \Bigg[\underbrace{\frac{\partial^2\,[k(T)\Delta\,T]}{\partial x^2}}_{I} + \underbrace{\frac{\partial^2\,[k(T)\Delta\,T]}{\partial y^2}}_{II}\cdots \\
&\qquad -W(T)\,\Delta\,T + \Delta\,S(x,y,t) - \underbrace{\frac{\partial[C(T)\,\Delta\,T]}{\partial t}}_{III}\Bigg]\;\Psi_1(x,y,t)\,dt\,dx\,dy \\
&+ \int_0^{r_h}\int_0^{t_f} \left[\Delta\,q(t) + \frac{\partial\,[k(T)\Delta T(x,t)]}{\partial y}\right]\;\Psi_2(x,t)\,dx\,dt \\
&+ \int_{r_h}^{L}\int_0^{t_f} \left[0 - \frac{\partial\Delta\,T(x,t)}{\partial y}\right]\;\Psi_3(x,t)\,dx\,dt \qquad (4.35)\\
&+ \int_0^L\int_0^{t_f} \left[0 - \Delta\,T(x,t)\right]\;\Psi_4(x,t)\,dx\,dt \\
&+ \int_0^H\int_0^{t_f} \left[0 - \frac{\partial\Delta\,T(y,t)}{\partial x}\right]\;\Psi_5(y,t)\,dy\,dt
\end{aligned}$$

$$+ \quad \int_0^H \int_0^{t_f} \left[0 - \Delta T(y,t)\right] \Psi_6(y,t)\, dy\, dt$$
$$+ \quad \int_0^H \int_0^{L} \left[0 - \Delta T(x,y)\right] \Psi_7(x,y)\, dx\, dy$$

Stationary conditions of $\Delta\mathcal{L}$

With the variation of the Lagrangian defined, it is now necessary to find the conditions where $\Delta\mathcal{L} = 0$ is satisfied. By inspection, it is apparent that the terms multiplied by Ψ_3 through Ψ_7 can be ignored because according to the boundary conditions of the variation problem they are satisfied by the solution of the variation problem. While this simplifies Eqn. (4.35) substantially there remains much more development in order to find the values of the remaining Lagrange multipliers, Ψ_1 and Ψ_2 which will lead to a stationary point.

To find this stationary point, the boundary conditions of the direct and variation problems will be used when possible. In order to use what is known about the boundaries of the system the terms in Eqn. (4.35) are integrated by parts to pass the integration from ΔT to $\Delta \Psi_1$. The breakdown of terms I, II and III shown below. Terms I and II are integrated by parts twice in the direction of the derivative, i.e., in x for term I and y for term II. Term three is only integrated by parts once and it is in time. For simplification, the unused integrals are ignored in these equations but will come back into the development later.

$$\underbrace{\int_0^L \frac{\partial^2 \left[k(T)\Delta T\right]}{\partial x^2}\Psi_1 dx}_{I} = \underbrace{\Psi_1 \frac{\partial \left[k(T)\Delta T\right]}{\partial x}\Big|_0^L}_{A} \cdots \tag{4.36}$$
$$\underbrace{- k(T)\frac{\partial \Psi_1}{\partial x}\Delta T\Big|_0^L}_{B} + \underbrace{\int_0^L k(T)\frac{\partial^2 \Psi_1}{\partial x^2}\Delta T(x,y,t)dx}_{C}$$

$$\underbrace{\int_0^H \frac{\partial^2 \left[k(T)\Delta T\right]}{\partial y^2}\Psi_1 dy}_{II} = \underbrace{\Psi_1 \frac{\partial \left[k(T)\Delta T\right]}{\partial x}\Big|_0^H}_{D} \cdots \tag{4.37}$$
$$\underbrace{- k(T)\frac{\partial \Psi_1}{\partial x}\Delta T\Big|_0^H}_{E} + \underbrace{\int_0^H k(T)\frac{\partial^2 \Psi_1}{\partial y^2}\Delta T(x,y,t)dx}_{F}$$

$$\underbrace{\int_{t_o}^{t_f} \frac{\partial \left[C(T)\Delta T\right]}{\partial t}\Psi_1 dt}_{III} = \underbrace{\Psi_1 \left[C(T)\Delta T\right]\Big|_{t_o}^{t_f}}_{G} - \underbrace{\int_0^{t_f} C(T)\frac{\partial \Psi_1)}{\partial t}\Delta T(x,y,t)dt}_{I} \tag{4.38}$$

Grouping like terms of ΔT

At this point it begins to become evident that a good way to ensure that a stationary point of $\Delta\mathcal{L}$ is found is to group like terms of ΔT and set them to zero. The first such term that can be grouped by is $\Delta T(x,y,t)$. After the integration by parts shown in Eqns. (4.36) through (4.38) terms C, F and I, along with the variation of the objective function, $\Delta\mathcal{J}$ and the perfusion term, $W(T)$ are all multiplied by $\Delta T(x,y,t)$. Grouping these terms and setting the result to zero yields the governing equation of what will eventually become the adjoint problem. Since the sum of these terms is forced to be zero, the integration over the time and space domains are dropped.

$$\begin{aligned} 0 \;=\; k(T)\frac{\partial^2\Psi_1(x,y,t)}{\partial x^2} \quad + \quad & k(T)\frac{\partial^2\Psi_1(x,y,t)}{\partial x^2} - W(T)\Psi_1(x,y,t)\ldots \\ & +Z(x,y,t) - C(T)\frac{\partial\Psi_1}{\partial t} \end{aligned} \tag{4.39}$$

The term, $Z(x,y,t)$ varies depending on whether the objective function is the thermal dose or thermal damage model. It is defined below.

$$Z(x,y,t) = \begin{cases} \frac{2\,AE}{\Re}\left[\Omega_c - \Omega_d\right]\exp\left(\frac{-E}{\Re\,T(x,y,t)}\right)\frac{1}{T(x,y,t)^2} & , \quad \text{Damage Model} \\ 2\,\left[D_c - D_d\right]R^{\overline{T}(x,y,t)}\left[(\overline{T}(x,y,t))\frac{1}{R}\frac{\partial R}{\partial T} - \ln R\right] & , \quad \text{Dose Model} \end{cases} \tag{4.40}$$

At this point in the development, it is beneficial to revisit Eqn. (4.35) and see what terms are left to be developed. Equations (4.36) through (4.38) remind the reader that terms A, B, D, E and G are still in the picture. Also terms containing the source function have not yet been dealt with. These terms are labelled M and N. Additionally, terms containing $\Delta S(x,y,t)$ and $\Psi_2(x,t)$ have yet to be developed. By setting the governing equation of the adjoint problem to zero, $\Delta\mathcal{L}$ is simplified. The following set of choices simplifies $\Delta\mathcal{L}$ further by setting the boundary conditions of the adjoint problem.

$$\begin{aligned} \Delta\mathcal{L} \;=\; & \int_{t_o}^{t_f}\int_0^H \underbrace{\Psi_1\frac{\partial\left[k(T)\Delta\,T\right]}{\partial x}\Big|_0^L}_{A} - \underbrace{k\frac{\partial\Psi_1}{\partial x}\Delta T\Big|_0^L}_{B}\,dy\,dt \\ + \; & \int_{t_o}^{t_f}\int_0^L \underbrace{\Psi_1\frac{\partial\left[k(T)\Delta\,T\right]}{\partial x}\Big|_0^H}_{D} - \underbrace{k\frac{\partial\Psi_1}{\partial x}\Delta T\Big|_0^H}_{E}\,dx\,dt \end{aligned}$$

$$+ \int_0^H \int_0^L \underbrace{\Psi_1 \left[C(T)\Delta T\right]\Big|_{t_o}^{t_f}}_{G} dx\, dy \tag{4.41}$$

$$+ \int_{t_o}^{t_f} \int_0^H \int_0^L \underbrace{\Delta S(x,y,t)\, \Psi_1}_{M} dx\, dy\, dt$$

$$+ \int_0^{r_h} \int_0^{t_f} \left[\Delta q(t) + \underbrace{\frac{\partial \left[k(T)\Delta T(x,t)\right]}{\partial y}}_{N} \right] \Psi_2(x,t)\, dx\, dt$$

To continue grouping by ΔT's, the additional terms need to be evaluated at the endpoints of the integration. Term A is expanded below.

$$\underbrace{\Psi_1 \frac{\partial \left[k(T)\Delta T\right]}{\partial x}\Big|_0^L}_{A} = \Psi_1 \frac{\partial k}{\partial T}\frac{\partial T}{\partial x}\Delta T\Big|_{x=L} + k(T)\Psi_1 \frac{\partial \Delta T}{\partial x}\Big|_{x=L} - \Psi_1 \frac{\partial k}{\partial T}\frac{\partial T}{\partial x}\Delta T\Big|_{x=0} - \Psi_1 \frac{\partial \Delta T}{\partial x}\Big|_{x=0} \tag{4.42}$$

Examining this expansion shows that the two rightmost terms in the above expression are zero from the definition of the variation and direct problems. This is not the case at $x = L$. This allows a choice for $\Psi_1(x,t)$ at $x = L$. Equation (4.43) will become the first known boundary condition of the adjoint problem it forces term A to be zero.

$$\Psi_1(L,y,t) = 0 \tag{4.43}$$

Expanding term B gives a similar result. Notice in Eqn. (4.44) that the expansion at $x = L$ is automatically zero by the definition of the variation problem, but at $x = 0$ the same cannot be said.

$$\underbrace{-\, k\frac{\partial \Psi_1}{\partial x}\Delta T\Big|_0^L}_{B} = -\, k(T)\frac{\partial \Psi_1}{\partial x}\Delta T\Big|_{x=L} + \, k(T)\frac{\partial \Psi_1}{\partial x}\Delta T\Big|_{x=0} \tag{4.44}$$

The variation problem statement only defines the behavior of $\frac{\partial \Delta T}{\partial x}$ at $x = 0$, not ΔT itself. Therefore $\frac{\partial \Delta \Psi_1}{\partial x}$ must be forced to zero at this point, thus creating the second boundary condition of the adjoint problem and forcing term B to be zero.

$$\frac{\partial \Psi_1(0,y,t)}{\partial x} = 0 \tag{4.45}$$

Because of their complicated nature terms D and E are skipped for the moment so that the more docile term G can be developed. Term G is evaluated at its integral end points below.

$$\underbrace{\Psi_1 \left[C(T)\Delta T\right]\Big|_{t_o}^{t_f}}_{G} = \Psi_1 \left[C(T)\Delta T\right]\Big|_{t=t_f} - \Psi_1 \left[C(T)\Delta T\right]\Big|_{t=t_o} \tag{4.46}$$

By the definition of initial condition of the variations problem, the far right hand side of Eqn. (4.46) is zero. The first term on the right hand side shows a curious result, term G will be zero if the "final" condition shown below is forced to occur. Therefore, the adjoint problem will have to be solved backwards in time.

$$\Psi_1(x, y, t_f) = 0 \tag{4.47}$$

At this point a decision must be made as to which of the two control functions, either boundary heat flux or volumetric heating will be used in the simulated treatment plan. Refer to Fig. 4.1 for the difference between the two heating sources. The source functions are developed separately with the boundary case presented in this chapter and the volumetric case left for development in Appendix B. Additionally, the boundary heating is used in the validation experiment so that a complete derivation of this part of control problem within the main text is important. The derivation then continues setting $\Delta S(x, y, t)$ to zero and concentrating on the remaining terms.

Expanding term D is much like expanding term A, however, the boundary at $y = 0$ requires extra attention since there is an incident heat flux over a portion of the boundary.

$$\underbrace{\Psi_1 \frac{\partial \left[k(T)\Delta T\right]}{\partial y}\Big|_0^H}_{D} = \Psi_1 \frac{\partial k}{\partial T}\frac{\partial T}{\partial y}\Delta T\Big|_{y=H} + k(T)\Psi_1 \frac{\partial \Delta T}{\partial y}\Big|_{y=H} - \Psi_1 \frac{\partial k}{\partial T}\frac{\partial T}{\partial y}\Delta T\Big|_{y=0} - \Psi_1 \frac{\partial \Delta T}{\partial x}\Big|_{y=0} \tag{4.48}$$

The terms evaluated at $y = H$ do provide another boundary condition for the adjoint problem in same way that term A does. By looking at the boundary conditions of the direct and variation problems we see that if Eqn. (4.49) is satisfied the first two equations on the right hand side of Eqn. (4.48) vanish.

$$\Psi_1(x, H, t) = 0 \tag{4.49}$$

The rightmost two terms in Eqn. (4.48) are put back into a compact form and revisited below in Eqn. (4.50). Recall that term D was only integrated by parts over the y domain, leaving an integration in both time and the x direction. The trouble here is that over the x domain this boundary is split. In Eqn. (4.50) this term is shown integrated in two steps over the x domain. The far right hand side of the equation (integration from $r_h \rightarrow L$) drops out because of the boundary conditions of the direct and variations problems over this portion of the boundary make it so, leaving only the first term on the righthand side.

$$\int_0^L -\Psi_1 \frac{\partial\,[k(T)\Delta\,T]}{\partial y}\bigg|_{y=0} dx = \int_0^{r_h} -\Psi_1 \frac{\partial\,[k(T)\Delta\,T]}{\partial y}\bigg|_{y=0} dx + \int_{r_h}^L -\Psi_1 \frac{\partial\,[k(T)\Delta\,T]}{\partial y}\bigg|_{y=0} dx \tag{4.50}$$

Expanding term E is much like the expansion of term B. The first term on the right hand side of the equation is forced to be zero by the definition of the variation problem.

$$\underbrace{-\,k(T)\frac{\partial\Psi_1}{\partial y}\Delta T\bigg|_0^H}_{E} = -\,k(T)\frac{\partial\Psi_1}{\partial y}\Delta T\bigg|_{y=H} + k(T)\frac{\partial\Psi_1}{\partial y}\Delta T\bigg|_{y=0} \tag{4.51}$$

The second term provides the final boundary condition for the adjoint problem. While nothing definitive can be said about ΔT at the $y = 0$ boundary because of the presence of $\Delta q(t)$, if Eqn. (4.52) is forced to be true then term E will vanish.

$$\frac{\partial\Psi_1(x,0,t)}{\partial y} = 0 \tag{4.52}$$

Now revisiting Eqn. (4.41) we can see that only three terms remain. The M term is gone because we have set $\Delta S(x,y,t) = 0$. Two of the terms are multiplied by $\partial[k(T)\Delta\,T]/\partial y|_{y=0}$. One of these two is the term N from Eqn. (4.41) which had yet to be discussed and the other is the remaining part of Eqn. (4.50). These terms are now grouped together, notice that they are conveniently integrated over the portion of the domain that receives the boundary heating.

$$\Delta\mathcal{L} \;=\; \int_0^{t_f}\int_0^{r_h} \Big[-\Psi_2(x,y) - \Psi_1(x,t)\Big] \cdot \frac{\partial\,[k(T)\Delta\,T]}{\partial y}\bigg|_{y=0} + \ldots \tag{4.53}$$
$$\Delta q(t)\,\Psi_2(x,t)\,dx\,dt$$

The following equality condition is then introduced which reduces the variation of the Lagrangian down to a single term. Remember since the goal is to reduce $\Delta\mathcal{L}$ this is permissible.

$$\Psi_2(x,t) = -\Psi_1(x,t) \qquad \text{for} \qquad [0 < x \leq r_h \, y = 0 \, t < t_o] \tag{4.54}$$

The remaining term is known as the first variation of the functional $\mathcal{J}(q)$ in the theory of calculus of variations. The expression shown below in Eqn. (4.57) is known as the scalar product in the working space, which is Hilbert space, or the space of squared integrable functions.

$$\begin{aligned} \Delta\mathcal{L} &= \int_{t_o}^{t_f} \int_0^{r_h} \Psi_2(x,t) \cdot \Delta q(t) \, dx \, dt & (4.55) \\ &= \int_{t_o}^{t_f} \mathcal{J}'(q(t)) \cdot \Delta q(t) \, dt & (4.56) \\ &= \langle \mathcal{J}'(q(t)), \Delta q(t) \rangle & (4.57) \end{aligned}$$

By definition of the scalar product and Eqn.(4.54) the gradient of the objective function can be written as shown below. This ends the first part of the development of the control problem. A complete description of this method for more general problems can be found in Alifanov (1994).

$$\mathcal{J}'(q(t)) = -\int_0^{r_h} \Psi_1(x,t) \, dx \tag{4.58}$$

Adjoint problem definition

During the development of the control problem, a governing differential equation, four boundary conditions and a final condition were found for $\Psi_1(x,y,t)$. Taken together as a complete set of differential equations it known as the adjoint problem. The solution of this set of differential equations when integrated as shown in Eqn. (4.58) gives the gradient of the objective function. It is shown as a complete set below, along with some notes on the equation. The subscript "1" has been dropped from the notation for convenience so that $\Psi = \Psi_1$.

$$C(T)\frac{\partial \Psi}{\partial \tau} = k(T)\frac{\partial^2 \Psi}{\partial x^2} + k(T)\frac{\partial^2 \Psi}{\partial y^2} - W(T)\,\Psi + Z(x,y,\tau) \tag{4.59}$$

$$\frac{\partial \Psi(x,\tau)}{\partial y} = 0 \qquad x\,,\, y=0\,,\, \tau>0 \tag{4.60}$$

$$\Psi(x,\tau) = 0 \qquad x\,,\, y=H\,,\, \tau>0 \tag{4.61}$$

$$\frac{\partial \Psi(y,\tau)}{\partial x} = 0 \qquad x=0\,,\, y\,,\, \tau>0 \tag{4.62}$$

$$\Psi(y,\tau) = 0 \qquad x=L\,,\, y\,,\, \tau>0 \tag{4.63}$$

$$\Psi(x,y) = 0 \qquad x\,,\, y\,,\, \tau=0 \tag{4.64}$$

Some notes on the above equations: the term Z(x,y,t) is defined above in Eqn. (4.40) and $\tau = t_f - t$. This variable change is used so that the final condition can be used as an initial condition.

4.3.2 The descent parameter

The above work is used solely to derive $\mathcal{J}'(\Gamma(t))$, however recalling the beginning of this chapter, there was another parameter which is essential to solving the optimal control problem. This parameter is known as the descent parameter and it is first introduced in Eqn. (4.9). Now the assumption is made that the decent direction is actually the variation of the heat flux. This can be done because up this point, $\Delta\Gamma(t)$ was an arbitrary perturbation of the control function.

$$\Delta\Gamma(t) = D^{(s)}(t) \tag{4.65}$$

With this assumption, a direct comparison between Eqn. (4.9) and Eqn. (4.14) can be made.

$$\begin{aligned} \Gamma^{(s+1)}(t) &= \Gamma^{(s)}(t) + \gamma^{(s)}\, D^{(s)}(t) \\ \Gamma_{\varepsilon}(t) &= \Gamma(t) + \varepsilon \Delta\Gamma(t) \end{aligned}$$

From this comparison, it is clear that $\gamma^{(s)} = \varepsilon$. The question is now how to find an expression for ε? Recall that the perturbed damage in general form is given by Eqn. (4.19). The development of the corresponding perturbed objective function is as follows.

$$\mathcal{J}_{\varepsilon}(\Gamma(t)) = \int_0^H \int_0^L \left[\Lambda_c(x,y) + \varepsilon\, \Delta\Lambda_c(x,y) - \Lambda_d(x,y)\right]^2 dx\, dy \tag{4.66}$$

$$= \int_0^H \int_0^L \left[\Lambda_c(x,y) - \Lambda_d(x,y)\right]^2 \ldots \tag{4.67}$$
$$2\varepsilon\Delta\Lambda_c(x,y)\left[\Lambda_c(x,y) - \Lambda_d(x,y)\right] + \varepsilon^2\left[\Delta\Lambda_c(x,y)\right]^2\, dx\, dy$$

The maximum descent can be found by setting the derivative of Eqn. (4.66) with respect to ε to zero and solving for ε. Remembering that $\gamma^{(s)} = \varepsilon$ an equation for the descent parameter can be written as follows.

$$\frac{d\left[\mathcal{J}_\varepsilon(\Gamma(t))\right]}{\varepsilon} = 0 \tag{4.68}$$

$$\gamma^{(s)} = -\frac{\displaystyle\int_0^H \int_0^L \Delta\Lambda_c(x,y)\Big[\Lambda_c(x,y) - \Lambda_d(x,y)\Big]\, dx\, dy}{\displaystyle\int_0^H \int_0^L \Big[\Delta\Lambda_c(x,y)\Big]^2\, dx\, dy} \tag{4.69}$$

Depending on which damage metric is used in the objective function, the evaluation of Eqn. (4.68) will change. $\Lambda_c(x,y)$ and $\Lambda_d(x,y)$ are the calculated and desired damage metrics, the definition of these can be found in the previous chapter, Eqns. (3.41) and (3.42). The only term that requires additional clarification is $\Delta\Lambda_c(x,y)$. This term is defined for both the thermal damage and thermal dose functions in Eqns. (4.22) and (4.23).

4.3.3 Parameterizing the gradient

It is often useful to parameterize the gradient of the functional. When implementing a computer program to solve this control problem, $\mathcal{J}'^s(t)$, $D^s(t)$ and $\Gamma^s(t)$ are vectors of length N unknown quantities. This comes as a result of discretizing the time domain. The more entries in time, the longer the vector, and consequently the more unknowns the program must find. If a parameterizing technique is used, the number of unknowns in these vectors can be reduced and a set of interpolating basis functions can be used to reconstruct the functions in time.

If M is a number of knots or parameters which are to be used in the spline. Then a continuous function can be composed of $M-1$ basis functions, each one covering a portion of the total time domain. If gradient is evaluated at M points in time, then M ordered pairs of $(t_i, \mathcal{J}'^s(t_i))$ can be recorded. Using this technique, the gradient of the functional and the descent direction will only be composed of M parameters. Two built-in functions were implemented in MATLAB to parameterize the gradient, "spline" and "ppval". The former

creates the parameters used in the cubic spline and the later reconstructs a parameterized function in time. Results showing the use of this technique are shown in the *Results and Discussion* section.

4.3.4 Convergence

Before presenting the solution algorithm below, it is important to make a decision on when the control problem has successfully converged. The iterative procedure will be considered finished when each component of the control function vector has satisfied the following stopping criteria (Loulou and Scott (2002)).

$$\left| \frac{\Gamma_i^{s+1} - \Gamma_i^s}{\Gamma_i^s} \right| \leq \epsilon \tag{4.70}$$

In the above equation, ϵ is a small user-chosen number which is not to be confused with ε used above the calculus of variations.

Convergence monitoring

If working properly the algorithm should minimize the objective function with each iteration until it can no longer reduce the value. While $\mathcal{J}$ was defined above, another convergence monitoring metric can be used, the RMS error of the damage or dose field (Beck et al. (1996)). The definition of this term is below in discrete form.

$$E_{RMS} = \frac{1}{N_x N_y} \left[\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} [\Lambda_c - \Lambda_d]^2 \right]^{\frac{1}{2}} \tag{4.71}$$

4.4 The Solution Algorithm

It is now possible to synthesize the theory presented in the previous two chapters into one minimization algorithm that can be used to find the desired thermal damage or thermal dose. The following steps assume that the model geometry, property values and desired damage or dose field are all known. The damage or dose field can be chosen or constructed from a known control function.

1. Give an initial guess for the control function, $\Gamma^0(t)$.

2. Solve the direct problem Eqns. (4.2) through (4.7) for the temperature distribution in the model in time, T(x,y,t).

3. Calculate the objective function, Eqn. (4.8).

4. Solve the adjoint problem Eqns. (4.59) through (4.64) and Eqn. (4.40).

5. Calculate the functional gradient, Eqn. (4.58). If desired, parameterize the functional gradient.

6. Calculate the conjugate parameter, β, Eqn. (4.11).

7. Calculate the descent direction, $\Delta\Gamma(t)$, Eqn. (4.10), recalling Eqn. (4.65).

8. Solve the variation problem, $\Delta T(x, y, t)$, Eqns. (4.29) through (4.34).

9. Calculate the descent parameter, γ, Eqn. (4.69).

10. Update the control function, Γ^{s+1}, Eqn. (4.9).

11. Check for convergence, Eqn. (4.70). If no convergence is reached restart algorithm with Γ^{s+1}.

The solution algorithm has been implemented in a MATLAB code called **Damage Dose Optimizer 3.0**. In Fig. 4.2 on the next page, a flow chart showing the main architecture of the minimization algorithm is shown. It is based on the above sequence of steps. Please refer to Appendix C for usage instructions. The code is available in Appendix D. A copy of this software is included with this thesis.

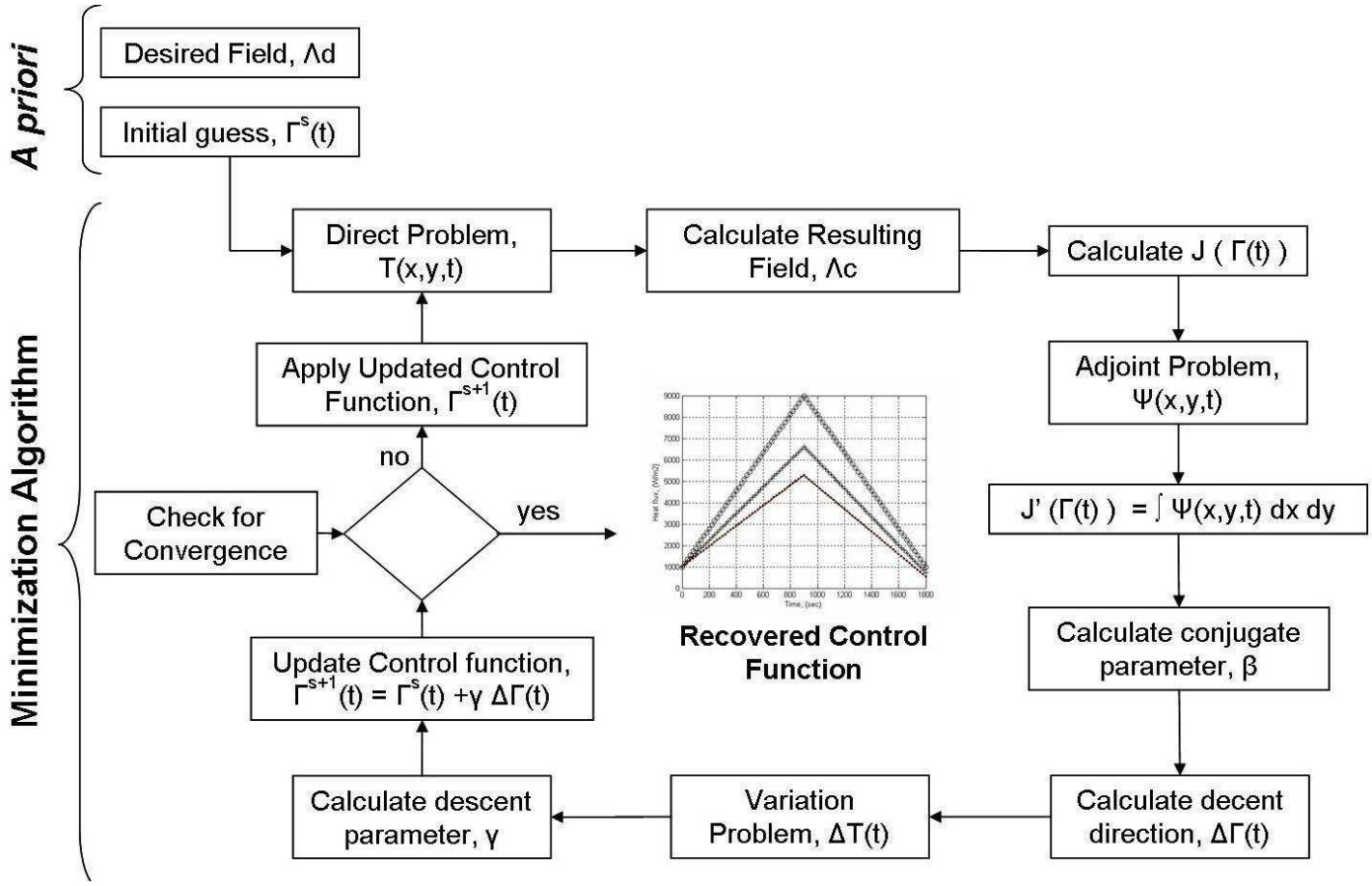

Fig. 4.2 Flow chart showing minimization program architecture.

Chapter 5

Experimental Methods

The methods used in performing the validation experiment are presented in this chapter. It begins with an overview of the goal of the experiment, which is followed by descriptions of the hardware, software and equipment used in the experiments. After this, the test stand and corresponding tissue phantom selection process is reviewed. The section ends with preparations made for matching the validation experiment to a corresponding finite difference simulation, including selections of time step and mesh density used in the minimization algorithm as well as calculations of convective boundary conditions and the temperature dependent internal convection coefficient.

5.1 Experiment Goal

The experiment is designed to validate the results from the minimization algorithm as well as the modelling accuracy of the finite difference routine. The overall experimental plan is shown below in Fig. 5.1. First a desired damage field is chosen. Using a finite difference model of the experimental setup in the direct problem, the heat flux that will cause this damage field is then calculated via the minimization algorithm. This ends the computational side of the validation experiment.

The validation experiment begins by taking the recovered heat flux from the minimization algorithm and applying it to a test stand which was closely modelled in the simulation. The chosen tissue phantom in the test stand is albumen, commonly known as egg white. Both naturally occurring albumen and powder based albumen purchased from a leading biological supply company were used. Albumen was chosen as the medium for a

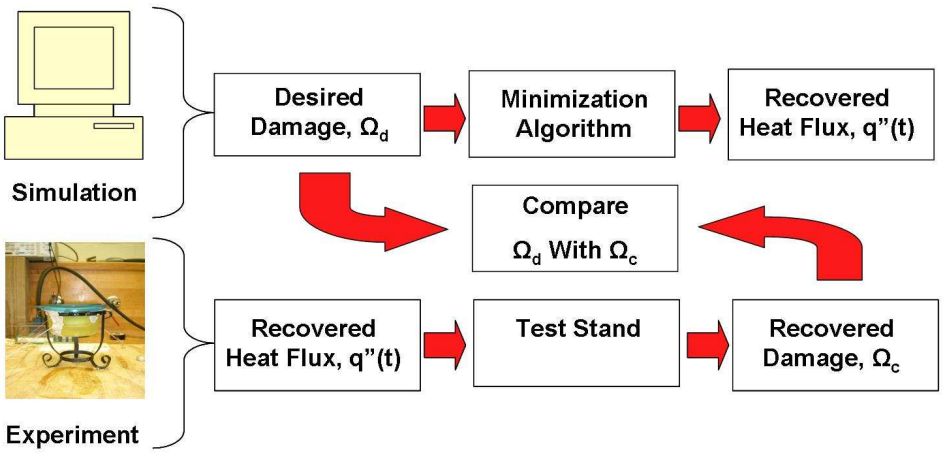

Fig. 5.1 Schematic of validation experiment

variety of reasons. First, the thermal ablation of albumen can be modelled by an Arrhenius model and the Arrhenius properties of albumen are available in the literature (Yang et al. (1991)). Second, the threshold of thermal damage, $\Omega = 1$ is easily identifiable by the naked eye since albumen becomes an opaque white when complete thermal damage is achieved. Finally, naturally occurring albumen is an inexpensive, accessible material that can be poured into the test stand to completely encase the heater and thermocouples. The fact that albumen is liquid at room temperature will cause some modelling problems but these can be overcome by using the internal convection coefficient which is built into the Pennes bioheat equation.

The recovered thermal damage is then compared with the desired thermal damage field. The damage field tends to thermally ablate a dome-like portion of the albumen above the heater. Therefore two metrics are used to quantify the extent of the thermal damage, H_{ab} and D_{ab}. These metrics are illustrated from an actual experimental result in Fig. 5.2. The ablated portion of the albumen is the white dome above the heater, which mostly obscured. The success of the minimization algorithm is measured by comparing the measured and simulated values of these metrics.

5.2 Test Stand Introduction

Two test stands were considered to be used in the validation experiment, the difference between the two was size. In both cases, holes drilled in the sides of the stand were used to thread the wires from the heater and thermocouples across the diameter. These holes

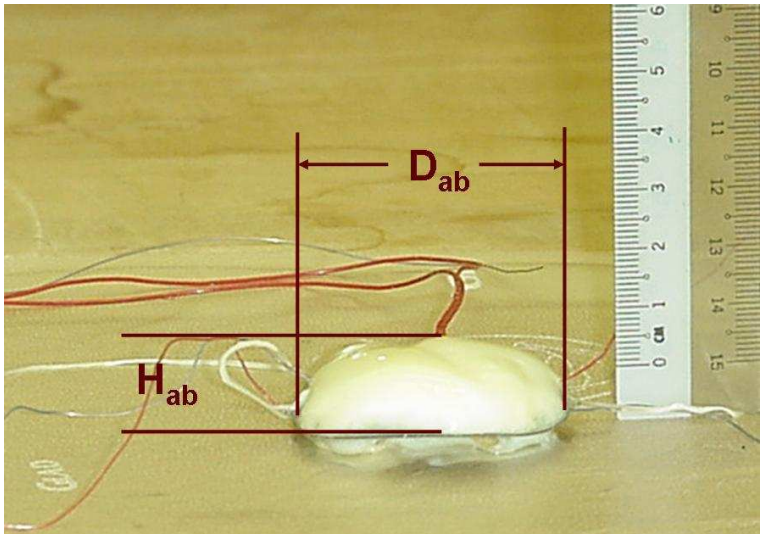

Fig. 5.2 Physical representation of metrics used to quantify thermal damage.

were patched with plumber's puddy so that no albumen would leak out during testing. Two threads of dental floss were used to sandwich the heater in place in the center of the cylinder.

5.2.1 Test stand No. 1

The first test stand was a 15 cm diameter cylinder constructed from Lexan. It was designed to be used to with an albumen mixture made from powdered albumen. The mixture consisted of 14% ovalbumin, which is the main protein composite of egg white and 86% water. The ovalbumin was purchased from a well known biochemical supply company. The mixture was made in the lab in the appropriate volume to completely fill the test stand. The test stand was fitted with six thermocouples, set in carefully chosen locations designed to experimentally verify the assumed symmetry above and below the heater as well as the radial independence of temperature. A schematic and photograph of the test stand are shown in Figs. 5.3 and 5.4.

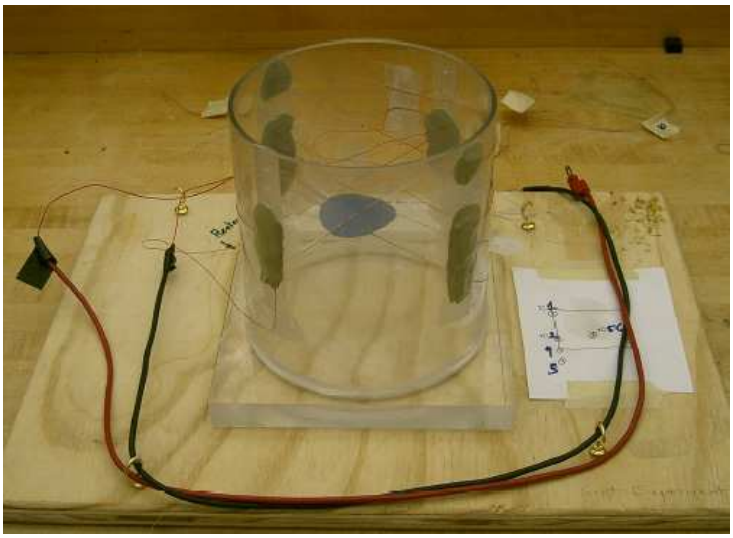

Fig. 5.3 Test stand one; 15 cm diameter lexan cylinder.

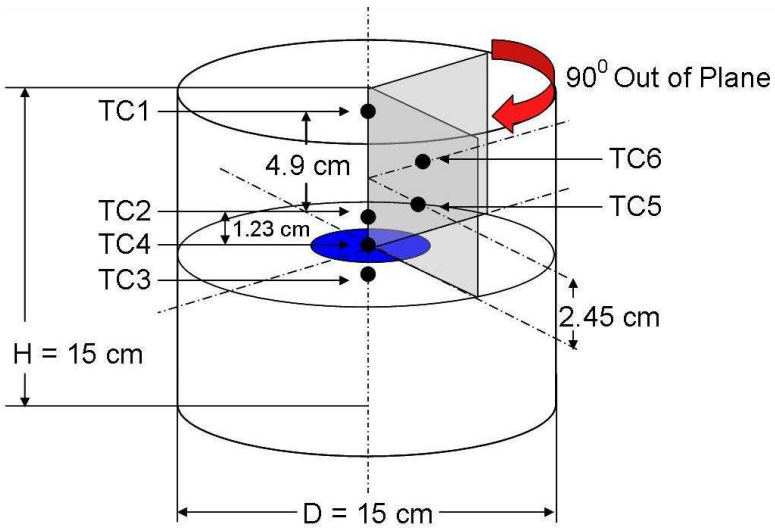

Fig. 5.4 Schematic of test stand one showing thermocouple placement.

Thermocouples two and three are each on the vertical axis of symmetry, 1.23 cm above and below the heater. Thermocouple four measures the top surface temperature history and thermocouple one is on the surface of the heater. Finally, thermocouples five and six are in place to ensure that there is angular symmetry.

5.2.2 Test stand No. 2

Test stand No. 2 was a smaller vessel used as an alternate if the relatively large size of test stand one became a problem. It was also cylindrical but slightly tapered such that the top diameter was slightly larger than the bottom. When modelling this smaller cylinder this tapering was omitted and an average diameter of 4.4 cm was assumed.

Several advantages of using the small stand will emerge later on in this section. This stand is actually a commercially available plastic product, driving down the cost. The stand is also suspended in air, allowing for a true convective boundary on the top and the bottom. A photograph of the test stand is shown below in Figs. 5.5 and 5.6.

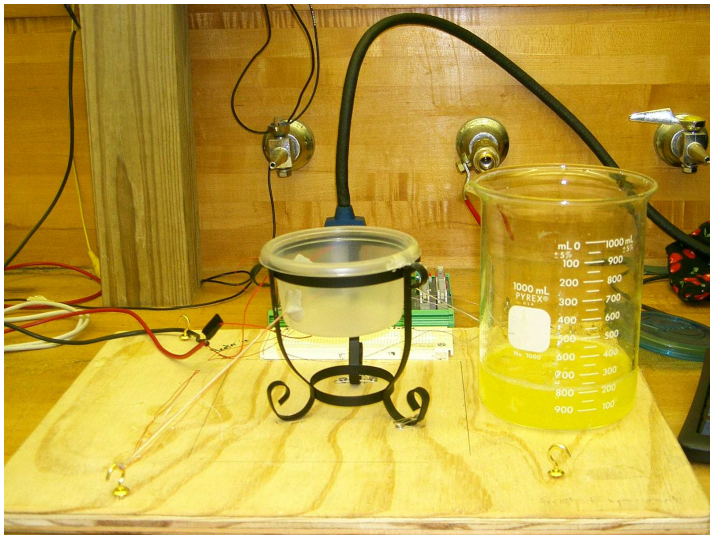

Fig. 5.5 Test stand two shown next to natural albumen.

Notice that fewer thermocouples are used in the second test stand. While this is elaborated on later in the chapter, fewer thermocouples were needed because the angular temperature symmetry was shown to be satisfactory in experiments using test stand No. 1.

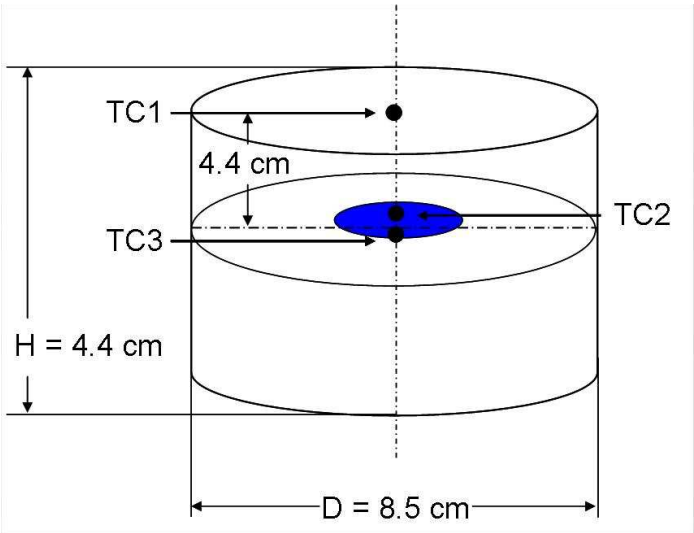

Fig. 5.6 Schematic of test stand two showing thermocouple placement.

5.3 Experiment Equipment

The experimental design requires the introduction of a variety of hardware, software and equipment. A schematic of the overall experimental setup is shown in Fig. 5.7.

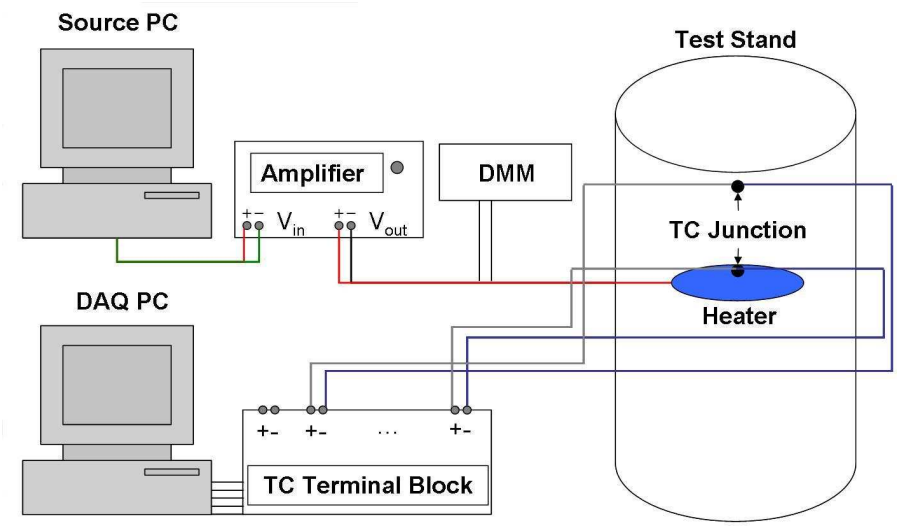

Fig. 5.7 Schematic of experimental setup.

5.3.1 Primary hardware

The hardware used in the experiment consists of two desktop computers, an amplifier and a digital multi-meter (DMM). The two computers are used to run the experiment; one

supplies the control function output and the other is a data acquisition (DAQ) machine. The control computer is a Dell ©Optiplex Gx110 with a Pentium III processor and the DAQ machine is a Dell, Dimension model with a Pentium IV. When running an experiment, the control and acquisition computers were started simultaneously. Originally the experimental design called for both control and acquisition software to be housed in the same computer. However because of compatibility problems between the two cards, two different computers had to be used.

An HP ©model 6842A DC power supply and amplifier maintained or amplified the voltage signal from the control computer to the heater. Table 5.1 shows the nonlinearity of the amplifier over a typically used voltage range. The gain in this sample was 1.45 times the input signal, which corresponds to one full rotation of the gain dial. The gain was set at the high end of the scale so that the discrepancy is largest at lower voltages. A Yokogowa © model 7551 DMM was used to verify the voltage to the heater was correct and to read the internal resistance of the heater.

Table 5.1 Nonlinearity of HP ©6824A amplifier, Gain = 1.45

V_{in}, V	$V_{measured}$, V	V_{exact}, V	$(V_{measured} - V_{exact})/V_{exact}\%$
0	0.041	0	-
1	1.48	1.45	2.01
2	2.92	2.90	0.68
3	4.37	4.35	0.45
4	5.82	5.80	0.35
5	7.26	7.25	0.14
6	8.70	8.70	0.0
7	10.15	10.15	0.0

5.3.2 Data acquisition and control system

The DAQ and control systems have two distinct but equally important functions. The control system is the most crucial because it's task is replicating the exact heating protocol output from the minimization algorithm. The DAQ system collects transient temperature data that are used to verify the accuracy of the model. National Instruments (NI) products were used for both systems, including LabView software.

The control system

The control system used was a NI model PCI 6722 control board. This is an eight channel analog output PCI card compatible with standard P.C's. In conjunction with this board, a $CB-68LP$ low-cost, unshielded I/O connector block was used to connect to the amplifier. The board has a maximum output of ± 10 V.

The board was controlled entirely through on the software side. National Instruments LabView software was used to create a control panel to operate the board. The front end of this "virtual instrument" (VI) is shown below in Fig. 5.8. To run an experiment, the user would choose the voltage update frequency and the gain, then click on the red arrow in the top-left of the screen. The panel would then ask the user for a data file containing output signal information. After this, the control side of the experiment needs no further interaction.

The acquisition system

The acquisition system used was a NI model PCI 4351, 14 channel, high precision temperature or voltage reader. This was used in conjunction with a $TBX-68T$ rack-mountable terminal block with built in cold junction compensation. Like the control system this DAQ system is entirely controlled on the software end. It can be used to log data from up to 14 thermocouples.

The front end of the LabView VI used to control the DAQ system is shown below in Fig. 5.9. Before starting the VI the user must specify the channels used, the types of thermocouples used and the file where the temperature readings are to be output.

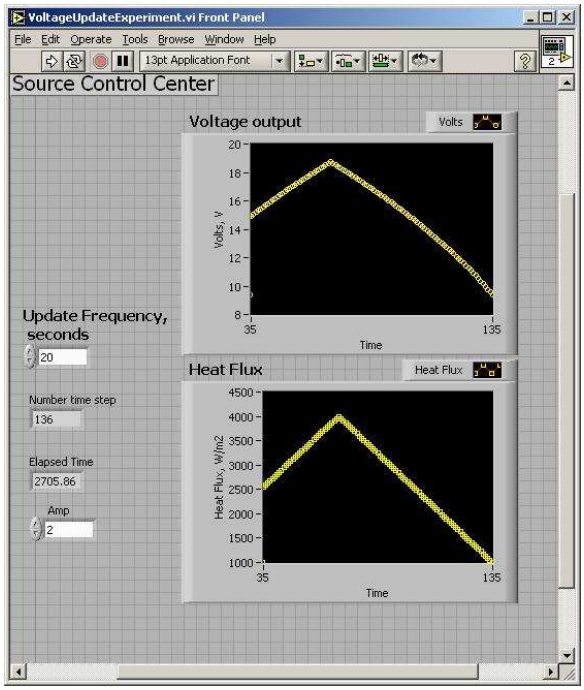

Fig. 5.8 LabView © user panel for control PC.

5.3.3 Electric resistance heater

The resistance heater is an extremely critical part of the experiment. Besides the test stand itself, it is the only piece of equipment that is included in the simulation, so that precise measurements of its size and resistance are required. A Minco © model HT 5545 R30 fully submersible, Teflon coated heater was used in the experiment. A picture of the heater is shown below in Fig. 5.10 and a table with important specifications is shown in Table 5.2. The heater was connected directly to the output of the amplifier. Tests were done to ensure that the resistance of the heater did not vary when heated.

Table 5.2 Heater specifications

Make	Minco©
Model	HT 5545 R30
Resistance	$32\,\Omega$
Maximum Current	3 A
Radius	2.3 cm
Effective Area	$1.37 \cdot 10^{-3}\mathrm{m}^2$

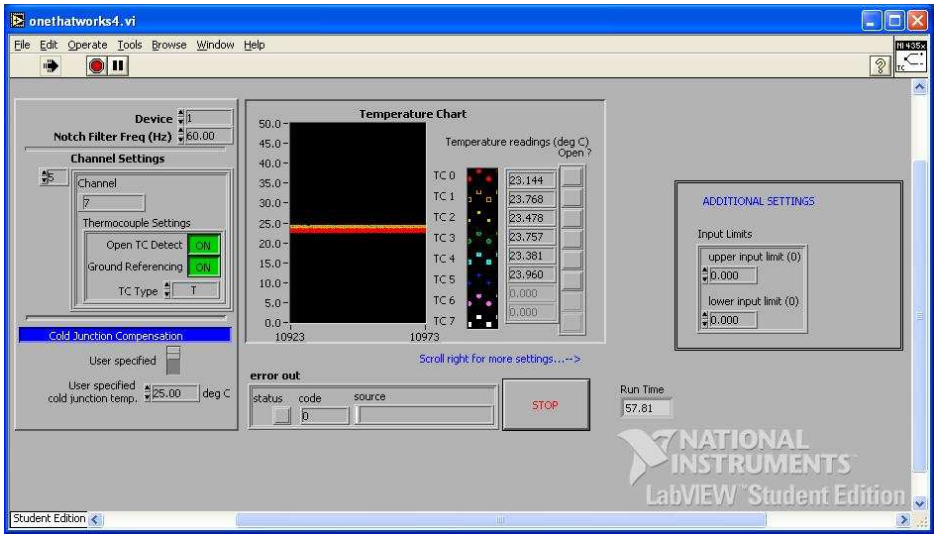

Fig. 5.9 LabView © user panel for DAQ system.

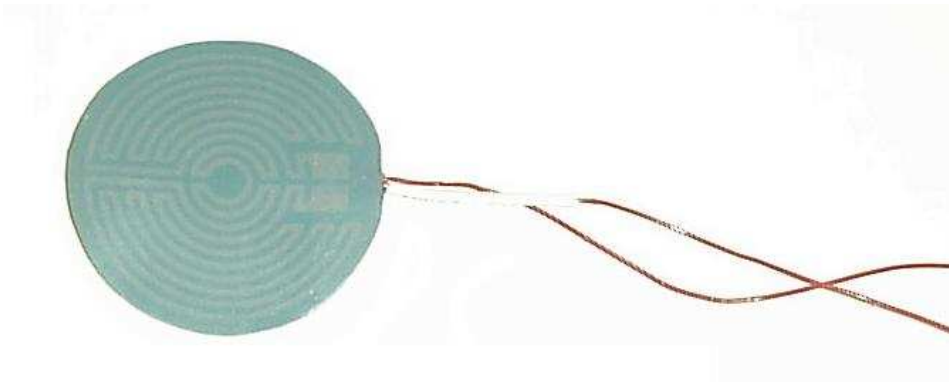

Fig. 5.10 Minco© heater used in experiments.

5.3.4 Thermocouple calibration

The thermocouples used in the experiment were type T, with wire diameter of $2.54{\cdot}10^{-2}$ cm. These are accurate to 1^oC or 0.75% of the temperature measurement. Because of the planned shape of the test stand, thermocouples used would have to be constructed from wire from two separate spools so that they could be threaded through opposite sides of the test stand. This threading put more tension on the thermocouple junction than commonly experienced and consequently several were broken over the course of the experiment.

The thermocouples were placed in an ice bath for calibration. The true temperature of the bath was measured with a thermometer and was verified to be 0^oC. The thermocouples were all placed in the bath simultaneously and data were recorded for 3 minutes. Table 5.3 shows the results from the calibration test for each thermocouple used in the experiment.

Table 5.3 T Type thermocouple calibration tests

TC No.	Mean Reading, μ, ^{o}C	Standard Deviation, σ, ^{o}C
1	-0.635	0.0488
2	+0.257	0.0584
3	+0.384	0.0684
4	-0.060	0.0475
5	-0.175	0.0533
6	-0.190	0.0541

The temperature data collected with the DAQ system serves to validate the temperature predictions within the tissue phantom during a heating protocol. The only parameter that is found via temperature data is the internal convection term, $H_i(T)$ which explained in more detail at the close of this chapter.

5.4 Modeling Considerations

Before running any experiments, an important set of modelling parameters would have to be set. Among these parameters were the mesh size of the finite difference grid, as well as the time step. Also, convective boundary conditions would have to be calculated to include the effects of the room temperature air surrounding the test stand. Table 5.4 summarizes all the parameters used in the finite difference program to simulate the experiment for both the large and small test stands (Pfefer et al. (2000)).

Table 5.4 Parameters used in simulations of experiments for both test stands

Parameter	Test Stand No. 1	Test Stand No. 2
L_x (cm)	7.5	4.22
L_y (cm)	7.5	2.2
N_x (nodes)	94	50
N_y (nodes)	32	25
$\Delta\, x$ (mm)	0.806	0.867
$\Delta\, y$ (mm)	2.4	0.925
$\Delta\, t$ (sec)	20	20
h_{top} $(W/m^2 - K)$	7.0	5
h_{side} $(W/m^2 - K)$	9.2	3.2
k $(W/m - K)$	0.56	0.56
C_p $(J/kg - K)$	4180	4180
$\rho(kg/m^3)$	997	997

5.4.1 Mesh size

A major consideration in choosing the mesh size was accurately including the radius of the heater. This is important because the number of cells over which the surface boundary condition is applied must be as accurate as possible. An error in this parameter, equates to too an incorrect amount of heat energy entering the system in the simulations. At the same time, the mesh size, Δx is based on the radius of the test vessel, not that of the heater. The result is that certain mesh sizes model the heater more accurately than others. An M-file was written in MATLAB plotted how accurately a range of mesh densities represented the true area of the heater. In Fig. 5.11 the discretized heater length error as calculated in Eqn. 5.1 is shown for a range of mesh sizes. In general the error is reduced as the number of control volumes increase since the size of an interior cell also decreased. Note however that at some radial mesh densities, such as 19, 32 and 45 control volumes, the radius of the heater is modelled accurately.

$$E_{heater} = \frac{N_{x,0 \to r_h} - r_h}{r_h} \cdot 100\% \tag{5.1}$$

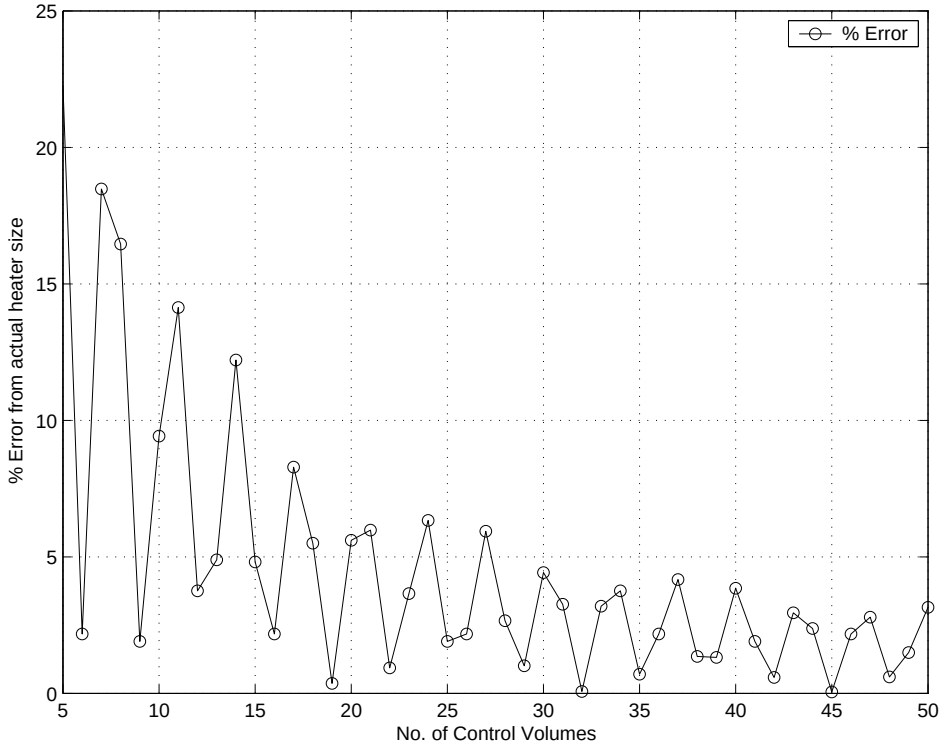

Fig. 5.11 Percent error in discretized heater length vs. mesh density.

Ideally, as sparse of a mesh density as possible would be used to represent the domain so that computational time can be reduced. But there is a trade-off between reduced computational time and reduced accuracy. After investigating the results from the heater length tests, five candidate control volume densities were chosen which all model the length of the heater accurately. For each of these densities, a simulation was performed wherein a constant heat flux of 500 W/m^2 was applied to the phantom holding the boundary conditions, initial conditions and material properties constant. The results are shown below in Fig. 5.12.

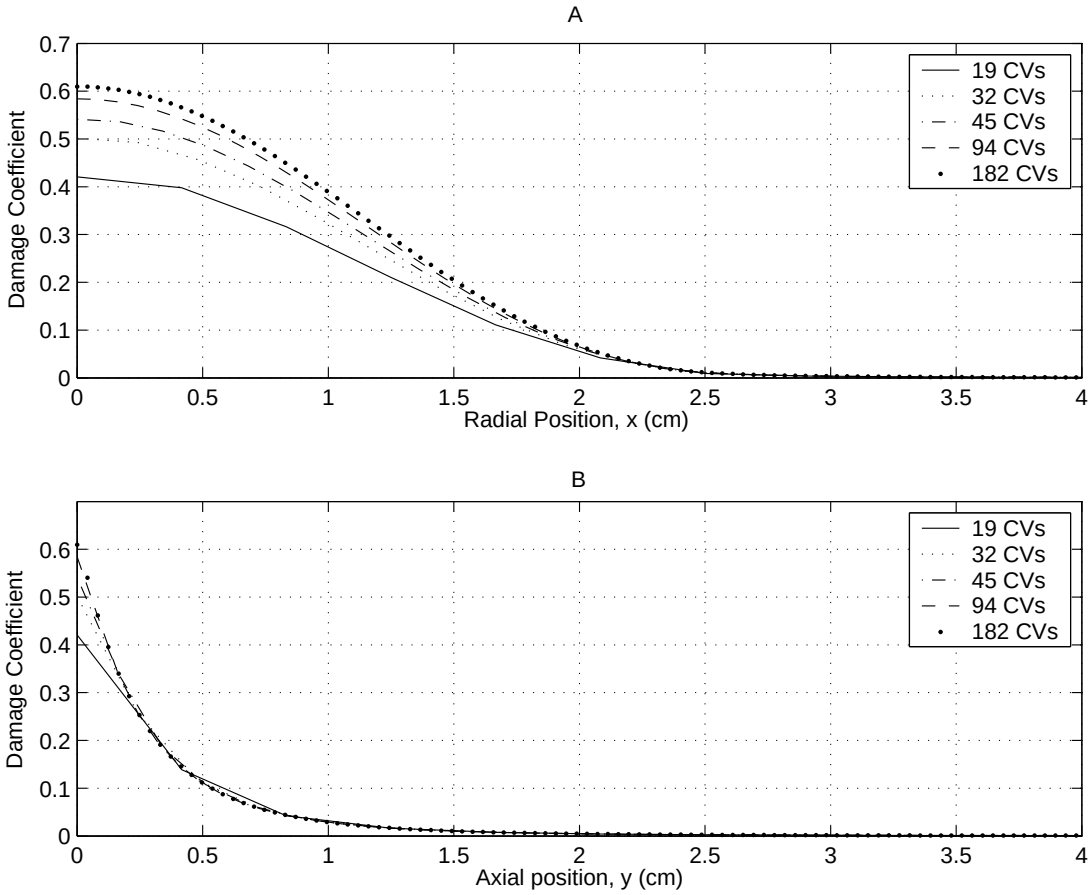

Fig. 5.12 Effect of spacial discretization on resulting damage coefficient, Ω_c profile. A, Effect on radial profile at horizontal midplane. B, Effect on axial profile at vertical midplane.

By inspecting the lower portion of the graph one can see that the mesh density affects the radial damage profile much more than the axial. This stands to reason because of the small but prevalent discrepancy between the total length over which the boundary heat flux is acting due to the different mesh densities. From the inspection of these graphs, the

decision to use a mesh density of 94 control volumes in the radial direction and 32 control volumes in the axial was made. While 94 control volumes wasn't the most accurate it was the most reasonable when considering computational cost. Also with 94 control volumes, the error in modelling the heater length, $E_{heater} = 0.07\%$.

5.4.2 Time step

The effect of changing the time step was also investigated with similar results. A nominal time step of 20 seconds was used along with variations of $\pm$ 50%. In all cases the total simulated treatment duration was 30 minutes. The results show in general that the damage coefficient has very little dependence on time step. The strongest dependence was found along the axial direction at $y = 0$. Below in Fig. 5.13 a 50% increase in time step predicts a small reduction in the extent of the damage, however a reduction of 50% provides only a minimal increase. A 50% reduction in time step correlates to a much larger computational cost so that the final time step used in the simulations remained $\Delta t = 20$ seconds.

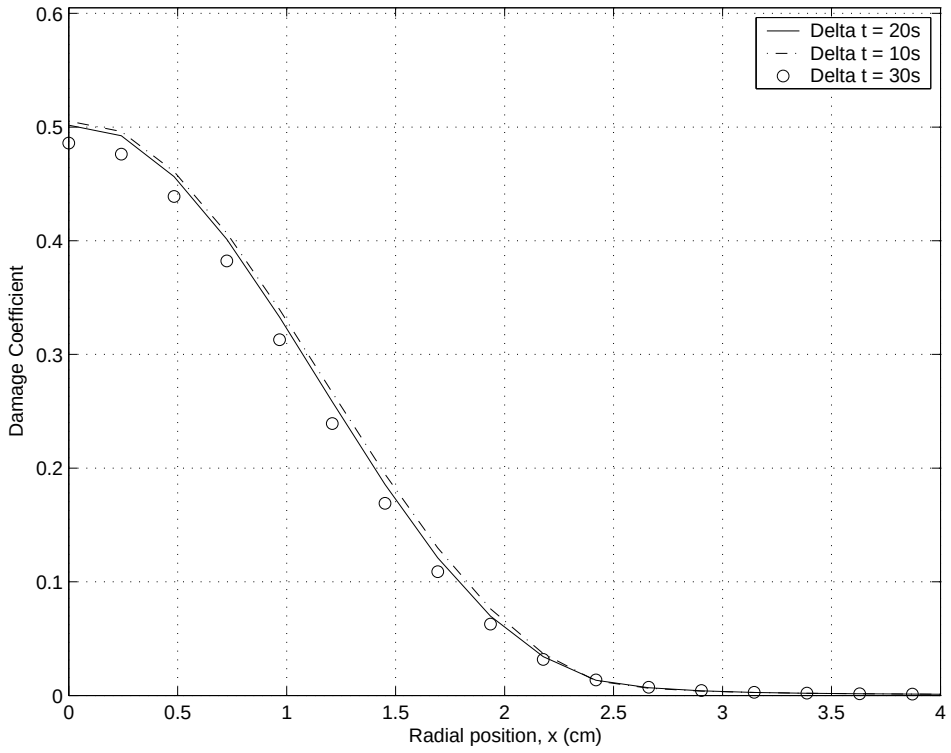

Fig. 5.13 Effect of time discretization on resulting damage coefficient, Ω_c profile.

5.4.3 Convective boundary conditions

Convective boundary conditions were used to simulate the behavior of the air in the lab surrounding the test stand. Commonly used formulas for finding the convective coefficients in situations where natural convection is present were used; the cylinder top was modelled as a plate heated from the bottom, Eqns. 5.2 through 5.4 and the side was modeled as a vertical plate, Eqns. 5.5 through 5.6.

$$R_a = \frac{g\beta_{air}\Delta T L_c^3}{(\nu\alpha)_{air}} \tag{5.2}$$

$$\overline{N_{u,top}} = 0.54\, R_a^{\frac{1}{4}} \tag{5.3}$$

$$h_{top} = \frac{k_{air}}{L_c}\,\overline{N_{u,top}} \tag{5.4}$$

$$\overline{N_{u,side}} = 0.68 + \frac{0.67\, R_a^{\frac{1}{4}}}{\left(1 + \frac{0.492}{Pr_{air}}\right)^{\frac{4}{9}}} \tag{5.5}$$

$$h_{side} = \frac{k_{air}}{L}\,\overline{N_{u,side}} \tag{5.6}$$

In the above equations, R_a is the Rayleigh number, $\overline{N_u}$ is known as the Nusselt number and h is the convection coefficient. The other parameters, β, ν, α, and k are the thermal expansion coefficient, kinematic viscosity, diffusivity and thermal conductivity of air evaluated at 300 K. Finally, g is the gravitational constant and L is a characteristic length.

5.4.4 Converting heat flux to volts

The output of a simulation in the MATLAB minimization algorithm is a heat flux protocol in time. This heat flux protocol must be converted to a voltage protocol to be used in the experiment. The conversion is done via Eqn. 5.7 which is based on Ohm's law and the sensor-specific parameters shown in Table 5.2. It is assumed that equal amounts of heat energy leave the top and bottom of the heater.

$$V(t) = \left(2\, q_h''(t)\, A\, R\right)^{\frac{1}{2}} \tag{5.7}$$

Where A is the effective area and R is the resistance.

5.5 Test Stand Selection

The experimental methods section ends with a description of measures taken to select which test stand would be used in the validation experiments. The two test stands differ in size as well as the phantom within them. Test stand No. 1 was used with albumen made from powdered ovalbumin whereas test stand No. 2 was filled with naturally occurring albumen from eggs which had been allowed to warm to room temperature.

5.5.1 Test stand No. 1

An experiment using test stand No. 1 was run according to the model presented in Fig. 5.1. A heat flux protocol was recovered from the optimization algorithm which predicted that a small portion of albumen around the heater would be thermally ablated. A summary of the algorithm's results are shown in Table 5.5. Figures 5.14 and 5.15 show the recovered heat flux and voltage protocols as well as the optimal and recovered damage coefficient profiles.

Table 5.5 Results summary: Simulation to be performed on test stand one

No. Iterations	21
CPU Time	789 sec
$\mathcal{J}(21)$	7180 [Dimm]
$E_{RMS}(21)$	28 [Dimm]
H_{al}	1 cm
D_{al}	2.4 cm

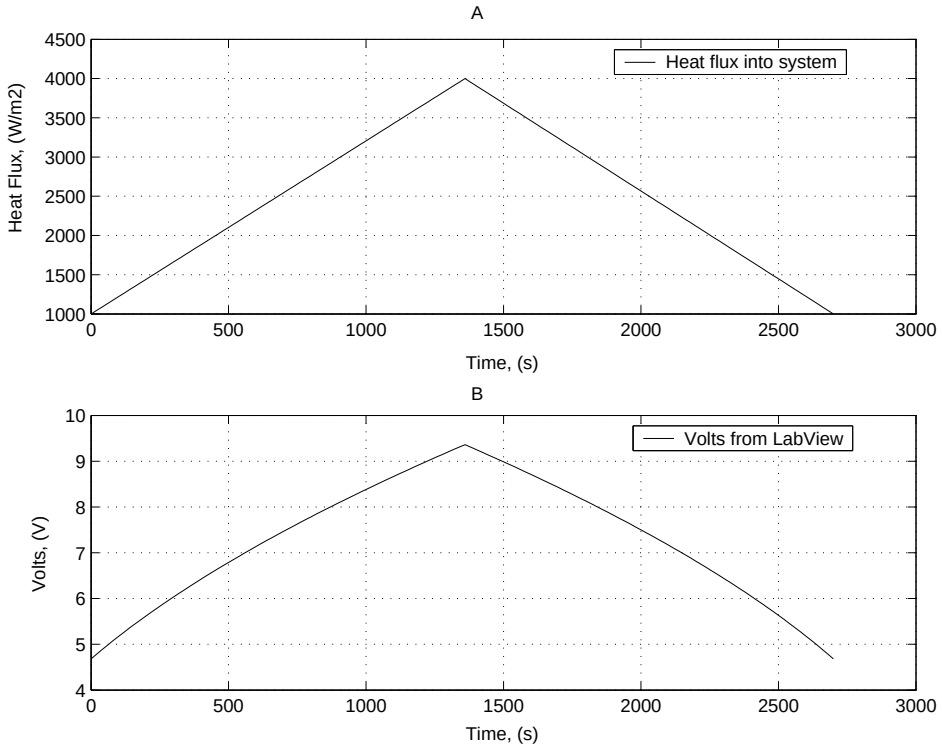

Fig. 5.14 A, Recovered heat flux used in preliminary experiment with test stand one, q"(t). B, Corresponding voltage protocol, V(t) before 2 x amplification.

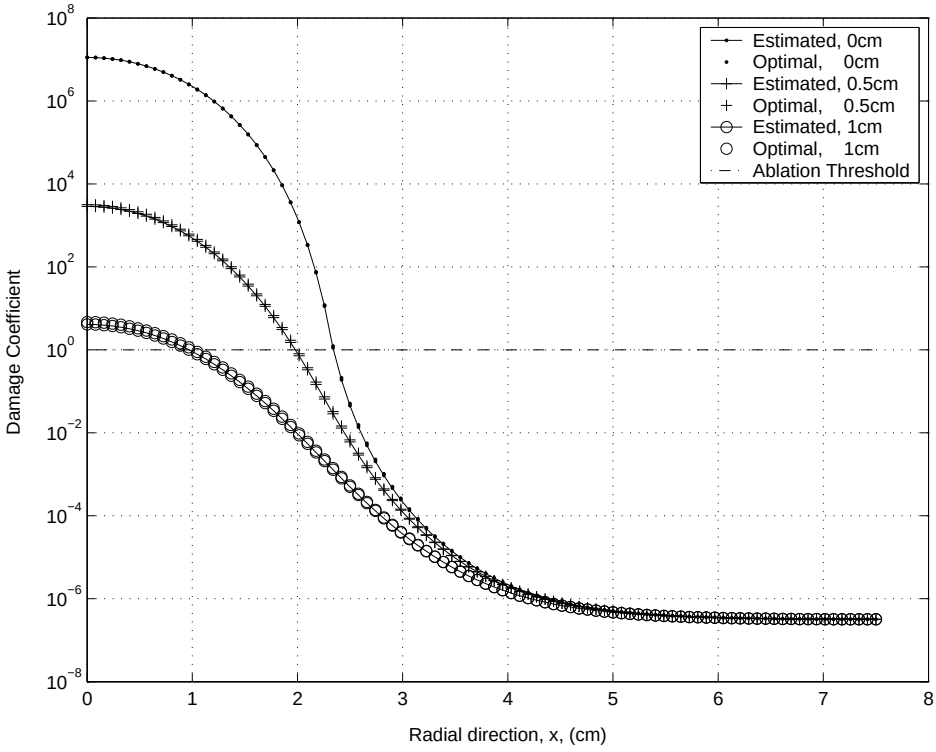

Fig. 5.15 Comparison of recovered thermal damage, Ω_c and desired thermal damage, Ω_d profiles from preliminary experiment using test stand one.

Albumen was made in accordance with the previously reported 14% ovalbumin by mass protocol and water to protein ratios found in naturally occurring albumen (Pfefer et al. (2000), Yamamoto et al. (1997). A picture of the test stand filled with the albumen is shown below in Fig. 5.16, note the opaque color. The major problem with this mixture was that it was much less viscous than naturally occurring egg white.

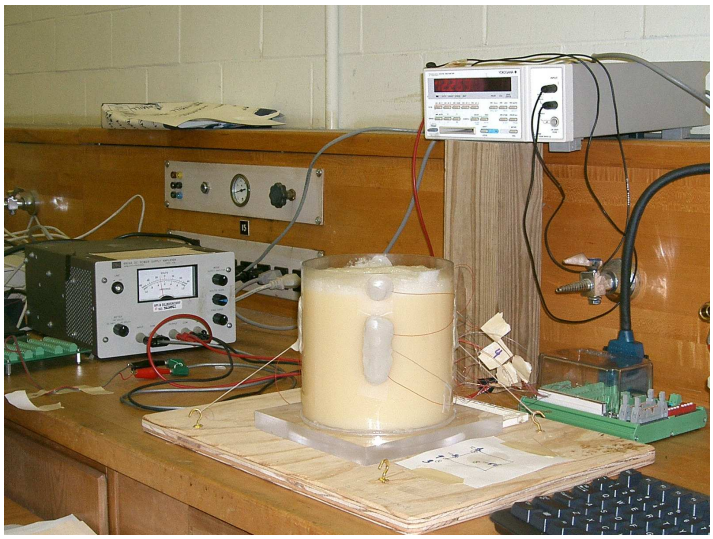

Fig. 5.16 Test stand one filled with powder-based albumen.

Results: Test stand no. 1

Figure 5.17 shows the predicted temperature histories at each thermocouple and the actual thermocouple output. Refer to Fig. 5.4 to see where the thermocouples were located. In all cases the discrepancy between predicted and experimental temperatures can be explained by the presence of natural convection. In the top left graph, the predicted temperature distribution is lower than the measured distribution. Since this thermocouple was directly above the heater the discrepancy was likely caused by a buoyancy driven flow delivering warmed albumen to the region of thermocouple 1. In graph B, the discrepancy between predicted and experimental temperatures is much larger. Notice that the thermocouple below the heater, TC #3, records cooler temperatures than its counterpart above the horizontal plane of symmetry, TC #2.

The error is greatest near the heater where the temperatures achieved in the experiment were up to 70 ^{o}C cooler than those predicted. This discrepancy is again believed to be caused by buoyancy driven flows mixing the albumen in the cylinder, thereby causing

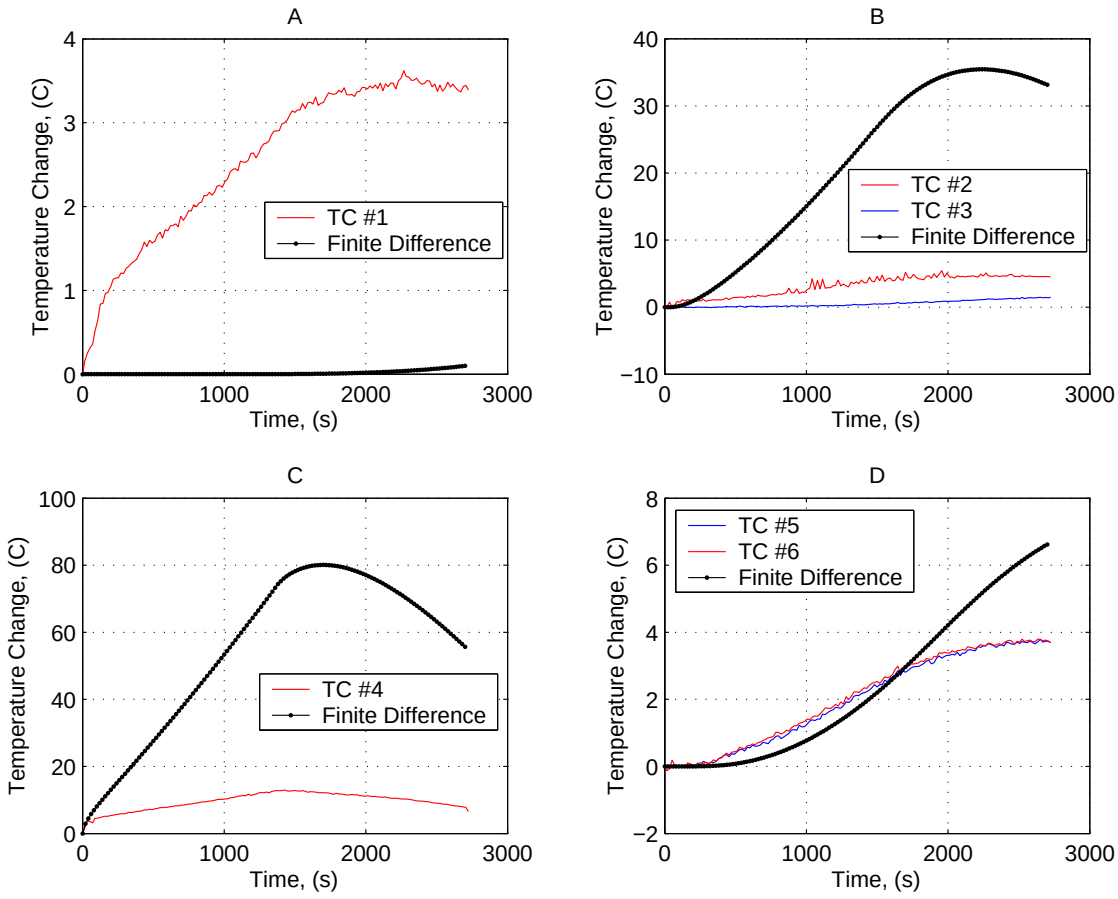

Fig. 5.17 Temperature histories, referenced to $T_o = 0$: FD prediction and experimental observation of test stand one. A, 1.4 cm below top of cylinder. B, 1.25 cm above $(TC\#2)$ and below $(TC\#3)$ the heater. C, Heater surface. D, 2.54 cm above and 2.3 cm radially outward, TC #6 is out of plane.

a cooling effect. Calculations showed that since the consistency of the albumen was much more like water than egg white, the Rayleigh number of this mixture predicted the presence buoyancy-driven flow. Injection dye tests could not be used to visually verify the presence of flow because of the opaque color of the powder-based albumen.

The experiment was not a total loss. Despite the presence of buoyancy-driven mixing, the assumption of angular temperature dependence was verified by TCs #5 and #6 in Fig. 5.17, graph D. Because the actual temperatures achieved near the heater were so much lower than those predicted, it should be no surprise that this experiment failed to validate the outcome of the minimization algorithm. It was not a failure on the side of algorithm. It was hypothesized that a more viscous medium, such as naturally occurring albumen would be needed to successfully carry out the experiment.

5.5.2 Test stand No. 2

In order to evaluate the hypothesis that higher temperatures could be achieved with a different material, test stand No. 2 was used with a natural albumen phantom. Naturally-occurring albumen from off-the-shelf eggs would be used, to take advantage of its increased viscosity. The stand is smaller, (8.5 cm diameter, 4.4 height) to reduce the number of eggs that would have to be used per experiment. The diameter of the test stand is slightly tapered from top to bottom, but in the simulation it is assumed to be perfectly cylindrical. Test stand No. 2, unlike No. 1 was not machined to any specifications. It is a disposable food storage container made of common polypropylene plastic. The walls of the test stand are not included in the model because of their low thickness ($\approx 1\,mm$) and distance from the heater. A picture of the test stand with natural albumen is shown in Fig. 5.18. Note the translucence compared to the powder based albumen.

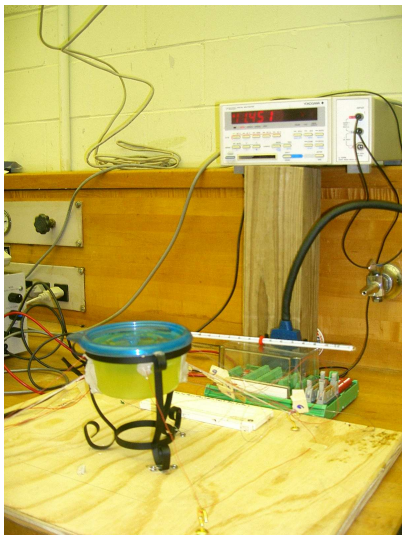

Fig. 5.18 Test stand two filled with natural albumen.

The minimization algorithm **was not** used to recover the heat flux protocol applied to test stand No. 2. By investigating the results of the direct problem on a variety of simulations, a linearly increasing protocol that predicted ample damage in the center of the phantom was selected for use as a baseline. The heat flux, voltage, and the resulting predicted damage profile are shown below in Figs.5.19 and 5.20. Since the temperature profile was shown to be independent of the angular position within the test stand, (see Fig. 5.17, panel C) three thermocouples aligned with the center axis were used in test stand

No. 2. A schematic is shown in Fig. 5.6.

The albumen used in the experiment came from store bought eggs and was extracted from the shell and yoke by hand. Before each treatment, the albumen was submerged in a room-temperature water bath for up to 5 hours.

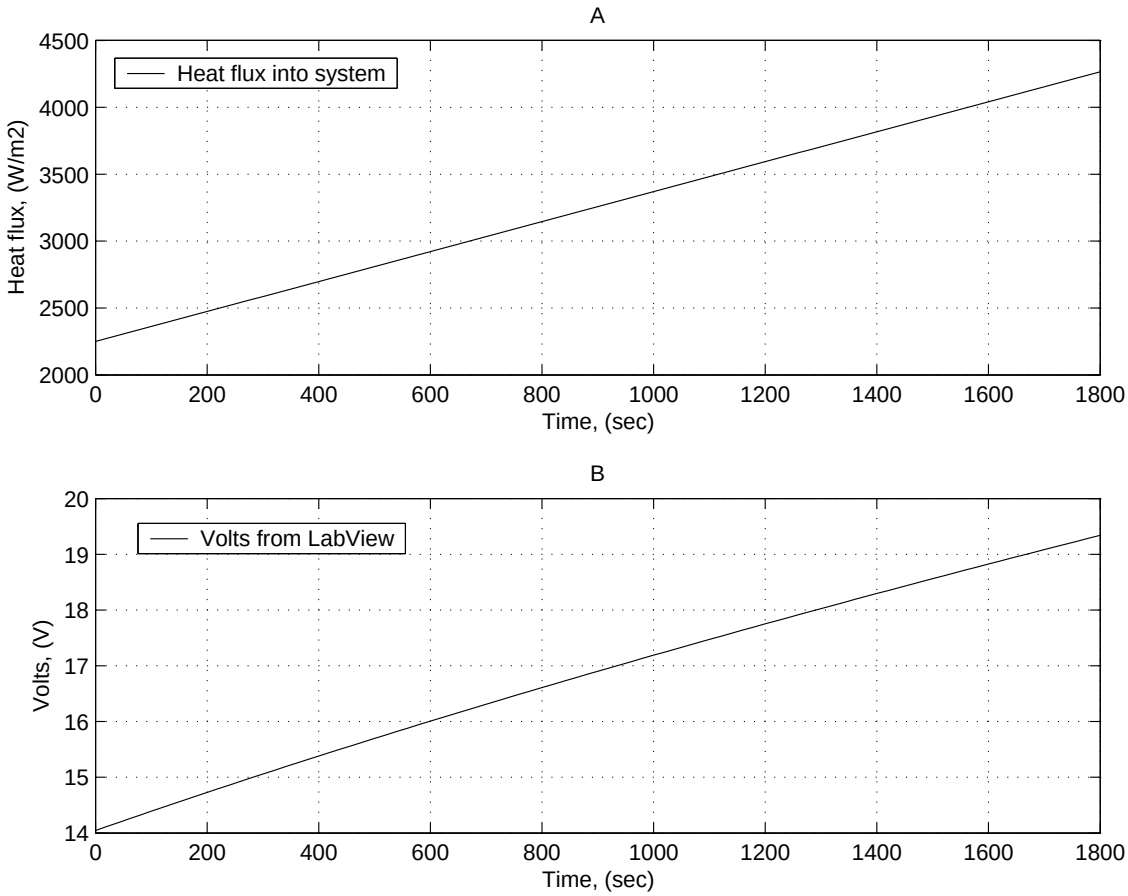

Fig. 5.19 Heat flux (A) and voltage (B) used to perform baseline experiment on test stand two. Amplifier gain set to 2x. Voltage shown prior to amplification.

Results: Test stand no. 2

A combination of the increased-viscosity albumen and the smaller test stand resulted in much higher temperatures within the natural albumen phantom than the powder-based albumen. Because of this success of this experiment, the results will be used as the baseline for further validation experiments. Figure 5.21 shows the resulting temperature histories within the test stand, temperature rise above initial temperature, $\theta = T - T_o$ is shown.

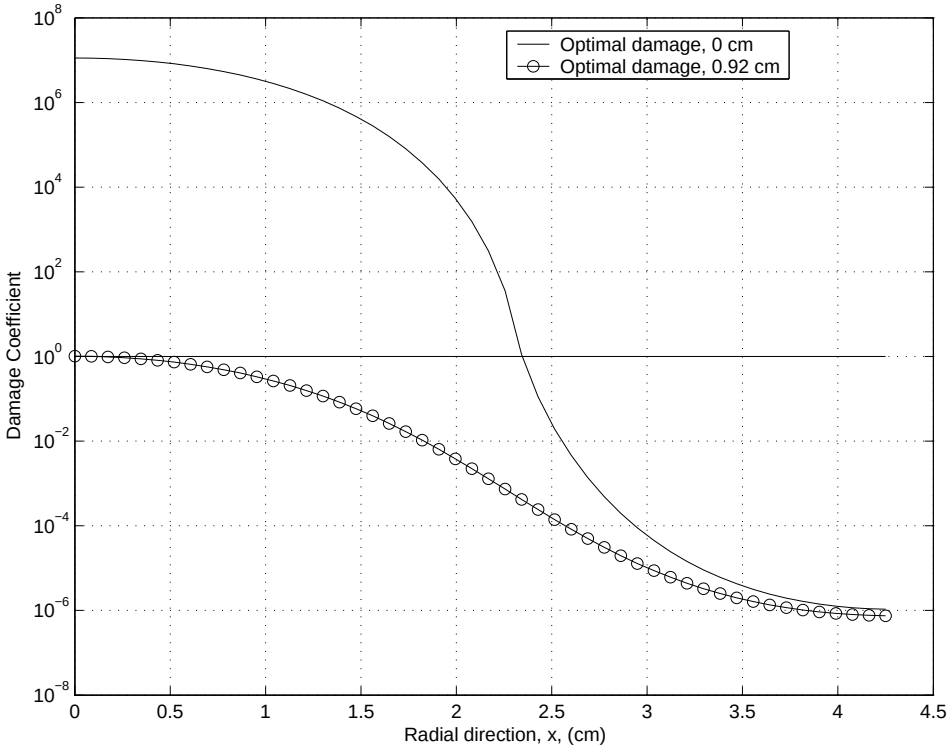

Fig. 5.20 Simulated damage profile resulting from applying the baseline heat flux shown in Fig. 5.19 to test stand No. 2. Curves shown correspond to two axial locations: Heater surface and H_{ab}

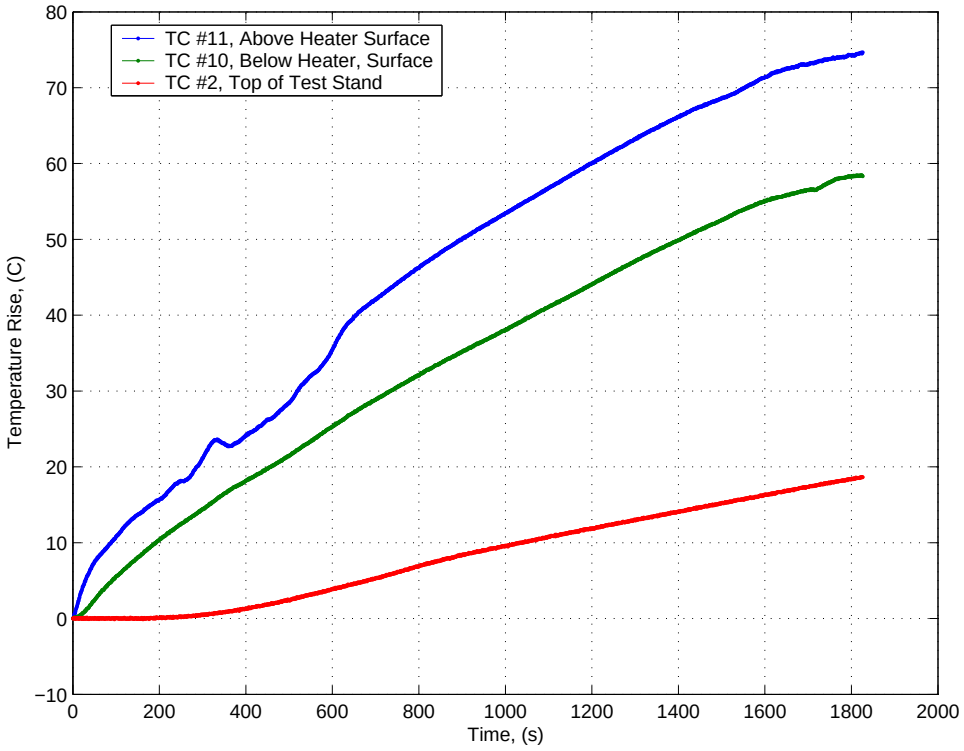

Fig. 5.21 Transient temperature profiles at thermocouple locations.

Adjusting for natural convection

From Fig. 5.21 it is hypothesized that natural convection, while not as prevalent as in powder-based phantom still has a cooling effect. The internal convection term built into the BHTE can be used to model this phenomenon. Even though the internal convection is a buoyancy driven phenomenon completely different than perfusion, it has the same cooling effect. Pennes defined the internal convection term as $W = \omega\, Cp_{bl}$, but the buoyancy driven analog can be described as $H_i = \omega_{ab}\, Cp_{ab}$ (Pennes (1948)). Previous work in has shown that ω is on the order of 1 $(kg/s)/m^3$ (Loulou and Scott (2002)). Working with the insight that the internal convection must be temperature dependent, the direct problem was used to fit the simulation results to the experimental data. A cubic spline function in MATLAB was used to create a function that described the internal convection behavior with temperature, it is shown below in Fig. 5.22. There is a sharp increase in the internal convection as the temperature rises, correlating to a simultaneous increase in buoyancy-driven mixing.

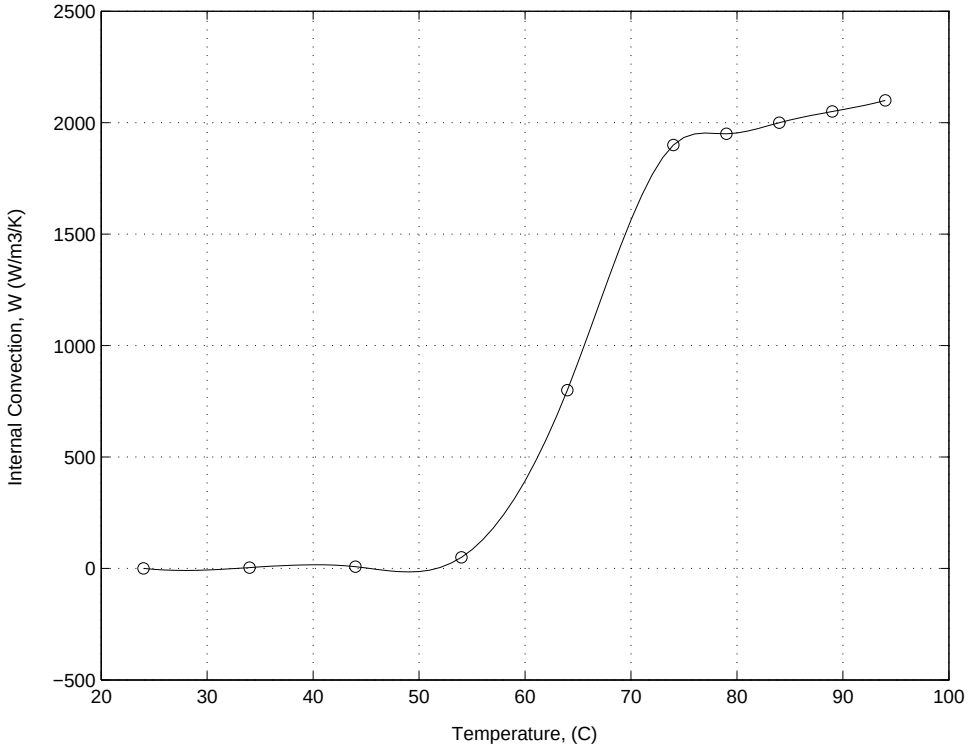

Fig. 5.22 Magnitude of internal convection coefficient, $H_i(T)$ vs. T, Empirically determined.

The resulting comparison of experimental temperature histories to simulated histories both with and without temperature dependent internal convection are shown in Fig.

5.23. The modelling accuracy is greatly improved at high temperatures when internal convection is included. The internal convection function shown above is only applicable for modelling transient temperatures above the heater. Just as in modelling natural convection, a separate function would be needed for fitting temperatures below the heater.

Near the heater, finite difference results are shown at 0 mm above the heater and $\approx$ 3 mm above the heater. This is done because it is difficult to isolate the exact axial location of the thermocouple bead. Also at this fine of a mesh grid, ($\Delta y = 0.86mm$) the bead itself is nearly the size of two control volumes. Plotting simulation results at these two locations provides a window where the experimental temperatures should be. By including the effects of the internal convection coefficient this requirement is met.

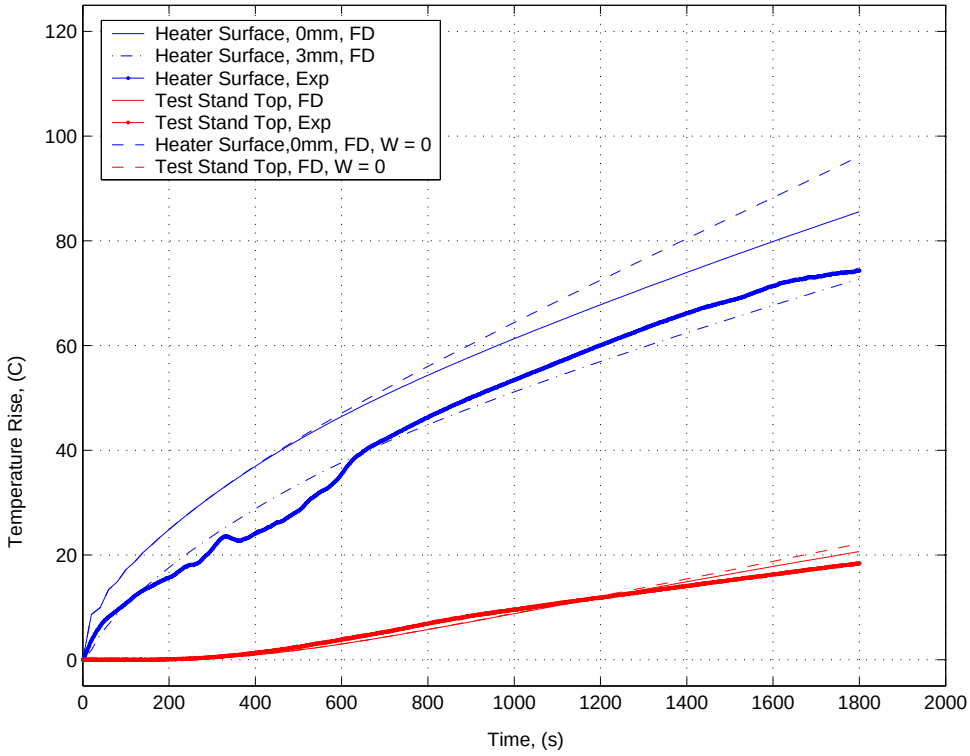

Fig. 5.23 Temperature histories vs. time for simulated and experimental results, test stand two. Simulated results without including internal convection effects are also shown.

Proof of internal convection

From Fig. 5.23 it is evident that some phenomenon is causing the temperature recorded from the test stand to be lower than the predicted values of the FD model. It is hypothesized

that this cooling is caused by natural convection plumes which have a mixing effect. The mixing in turn lowers the temperature achieved in the test stand, particularly at the surface of the heater. The model was made more accurate by including a temperature-dependent internal convection coefficient, H_i. But how do we know that the mixing is actually caused by natural convection? Additionally, can we prove that the amount of natural convection is temperature dependent? The following is a brief answer to the above questions.

Table 5.6 Parameters used in convection study

Property (Units)	Value
μ_{ab} (Poise at 273 K)	25
$\beta_{300\,K,H_2O} \cdot 10^6$ (1/K)	276
$\beta_{330\,K,H_2O} \cdot 10^6$ (1/K)	500
$\beta_{350\,K,H_2O} \cdot 10^6$ (1/K)	624
$\beta_{370\,K,H_2O} \cdot 10^6$ (1/K)	728

If the top of the heater is considered a flat heated plate in a surrounding medium of albumen, Eqns. 5.2 through 5.4 can be used to find the average convective coefficient over the plate. ΔT is taken to the be the temperature rise above the initial condition. The values of ΔT reflect values found in the experiment. The properties used in these calculations can be found in Table 5.6. Thermal properties can be found in Table 5.4. The viscosity of albumen was taken from a web site supported by Oregon State University, (Egg (2004)). Due to lack of information in the literature, the thermal expansion coefficient of water at various temperatures is used for albumen. Recall that albumen has a high (88%) water content Yamamoto et al. (1997).

With these data Eqn. 5.8 was used to calculate the convective heat transfer that would result. The calculated values are shown as a percent of the maximum heat flux applied to test stand no. 2,($4300\,W/m^2$). At high temperatures the convective effects are shown to dominate, proving natural convection is present.

$$q"_{conv} = h_{top}\,\Delta T \tag{5.8}$$

Order of magnitude study of $H_i(T)$

In a previous publication, Loulou and Scott (2002) describe the internal convection coefficient in terms of a mass flow rate per volume, ω, multiplied by the specific heat. If

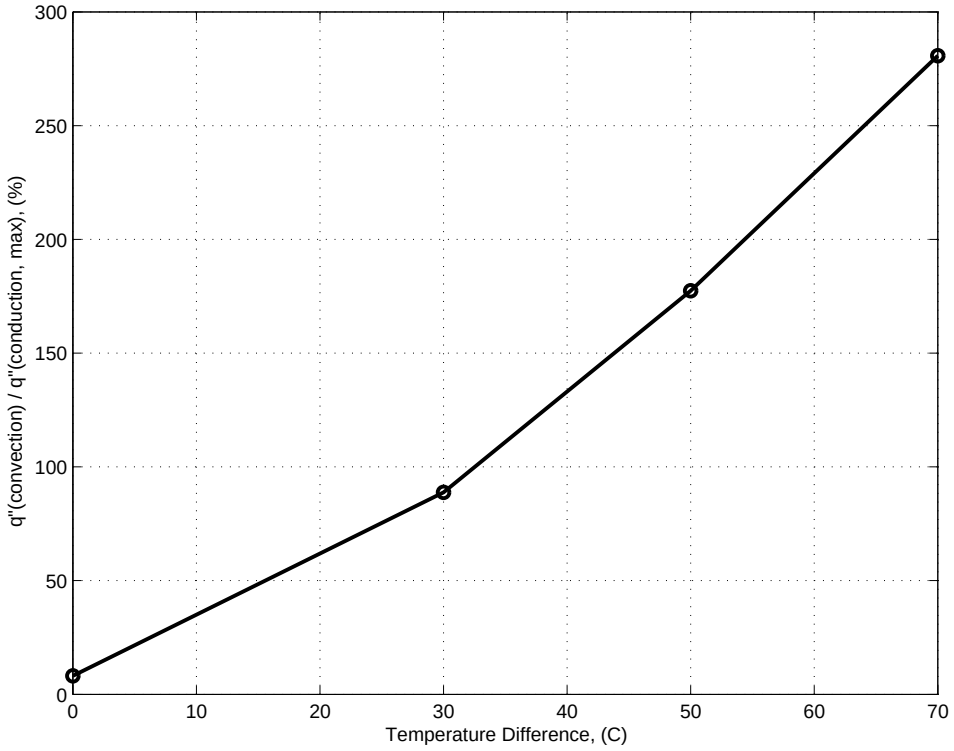

Fig. 5.24 Ratio of natural convection at heater surface to heat flux from heater ($q"_{conv}/q"_{cond} \cdot 100\%$) for various values of ΔT found during the baseline experiment.

$\omega_{ab} = 1(kg/s)/m^3$, then the magnitude of the convection would be on the same order as the magnitude of the specific heat. In this study, the maximum empirically used value of H_i is $2100\,W/m^3/K$; given the specific heat of albumen used, $4180\,J/kg/K$, this corresponds to a maximum mass flow per volume of $\omega_{ab,max} \approx 1/2(kg/s)/m^3$. In the preferred units of $ml_{ab}/ml_{ab}/s$, the value is $5 \cdot 10^{-4}$, which is in the range of commonly used values for perfusion in the literature (Scott et al. (1998)), (Chato (1985)). The following development asks the question, how much motion in terms of a mass flow or volume flow of albumen does this value of ω_{ab} correspond to?

Assume a volume, V_{flow}, is taken as a small cylinder of roughly the radius of the heater, 2cm and a height of 3 mm. This corresponds to the region above the heater where the thermocouple is located as well as the location of the greatest internal convection according to Fig. 5.23.

$$\omega_{ab,max} \quad = \quad \frac{\dot{m}}{V} \quad \approx \quad \frac{1}{2}\,\frac{kg_{ab}}{m_{ab}^3\,s} \tag{5.9}$$

$$
\begin{aligned}
\omega_{ab,max} &= \frac{\dot{V}}{V} \approx 5 \cdot 10^{-4} \, \frac{ml_{ab}}{ml_{ab}\, s} \\
\dot{V} &= \frac{\dot{V}}{V} V_{flow} \approx 19 \cdot 10^{-4} \, \frac{ml_{ab}}{s} \\
\frac{\dot{V}}{V}^{-1} &= 2000\,\mathrm{s}
\end{aligned}
$$

The empirically found convection coefficients calls for very little bulk motion within the albumen. The final equation above gives $\frac{\dot{V}}{V}^{-1}$ which is the amount of time required to completely replace the albumen in V_{flow} at the maximum flow rate. These numbers correlate to very little flow within the chamber. Injected die tests confirmed that the albumen showed little motion while heated, provided that the dye was injected into thick albumen. After this study, the empirically found H_i curve was used in the validation tests.

5.6 Setting the Baseline

On the experimental side, the heating protocol on test stand No. 2 was a success. The resulting extent of damage in the axial and radial directions is used as the baseline set of damage metrics. The damage profile which results from the heat flux protocol shown in Fig. 5.20 will be used as the optimal protocol (Ω_d) for all remaining validation experiments. All remaining validation experiments will also employ the empirically found, temperature dependent, internal convection coefficient, $H_i(T)$ shown in Fig. 5.22.

Chapter 6

Results and Discussion

This chapter discusses the results of the validation experiment and the minimization algorithm. To recap, the two dimensional heat conduction theory and finite difference code were presented in Chapter 3. In Chapter 4, the optimal control problem was formulated. The MATLAB minimization algorithm, **Damage Dose Optimizer 3.0** which is shown in full in Appendix D, is composed of the theory developed in these two chapters. In Chapter 5, the experimental setup for the validation experiment protocol was presented. The natural albumen phantom was found to perform better than the powder-based option because of its increased viscosity. A linearly-increasing protocol was applied to the phantom in test stand No. 2 and the resulting extent of thermal damage was used in the validation experiments as the optimal damage profile. In this chapter, the complete results of the validation experiments, a sensitivity analysis of the parameters in the simulation and results from two hyperthermia simulations of varying complexity are presented and discussed.

6.1 Validation Experiment

The heat flux protocol shown in Fig. 5.19 was used to create the damage profile shown below in Fig. 6.1. The threshold of permanent damage is $\Omega_d = 1$, which is illustrated by the grey plane. The desired extent of ablation is shown as well. These numbers correlate to the predicted values for H_{ab} and D_{ab}. Theoretically, these results should be reflected over both the x and y axes due to lines of adiabatic symmetry. However, due to the effects of natural convection and the curve fit for $H_i(T)$ the extent of damage will not be as large below the heater because the temperatures are lower.

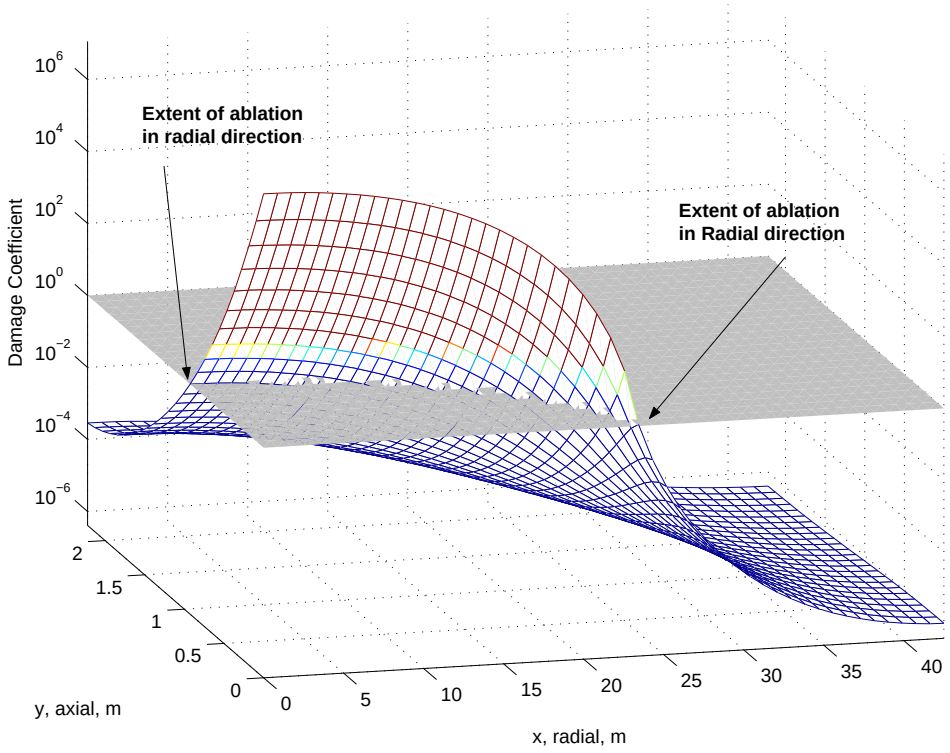

Fig. 6.1 Baseline damage profile used in validation experiments shown with predicted extent of complete thermal ablation.

This damage profile was considered the optimal damage profile, (Ω_d) for the remainder of the validation experiments. Besides the baseline case, two additional runs of the minimization algorithm were done using the profile from Fig. 6.1 as Ω_d. *Test 1*, used a triangular initial guess of the heat flux protocol, and *Test 2*, a linear guess was given. In both cases, the minimization algorithm was allowed parameterize the output protocol. Parametrization was used to avoid recovered heating protocols that used negative heat flux over part of the half hour treatment. A flowchart explaining how the validations experiments were performed is shown in Fig. 6.2. Red arrows indicate points of comparison and bold lines represent information from the baseline case applied to the two additional experiments.

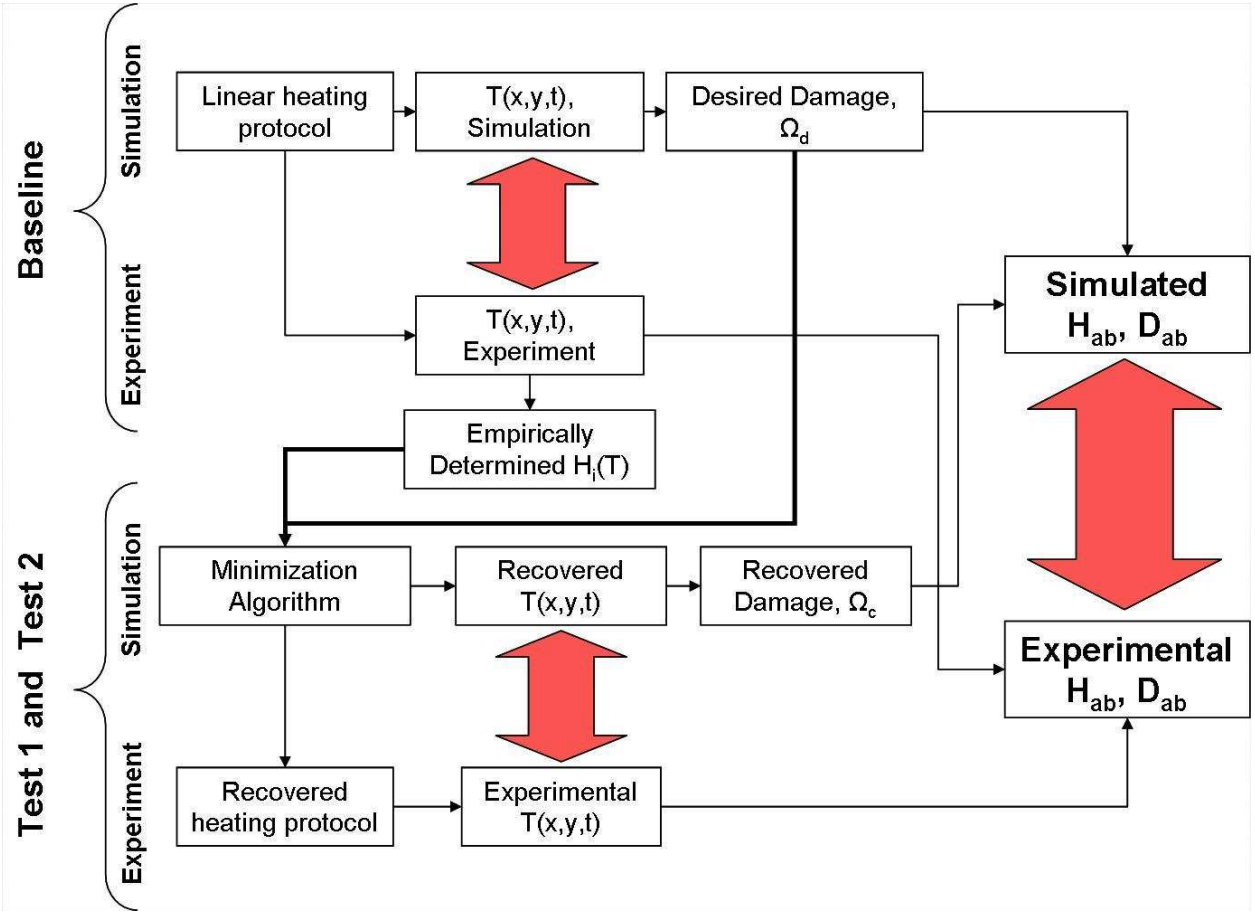

Fig. 6.2 Flowchart describing how the validation experiments were conducted. Red arrows indicate points of comparison and bold lines represent key information exchange.

6.1.1 Validation experiment: Simulation results

The simulations were run with the parameters shown in Table 5.4. To reiterate, the thermophysical properties used in the simulations were $k = 0.56\,W/m\,K$, $C_p = 4180\,J/kg\,K$ and $\rho = 997\,kg/m^3$ for thermal conductivity, specific heat and density of albumen respectively (Pfefer et al. (2000)). The algorithm also implemented the empirically-derived temperature dependent internal convection, $H_i(T)$ shown in Fig. 5.22.

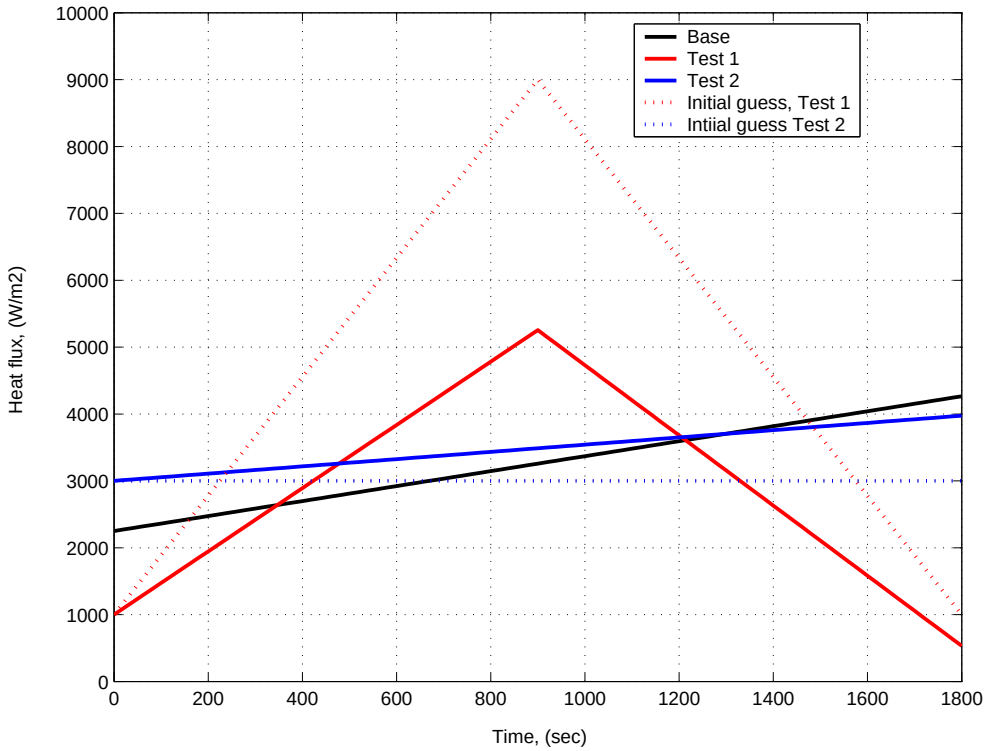

Fig. 6.3 Control functions resulting from minimization algorithm used in validation experiment. Initial estimates shown for *Test 1* and *Test 2*.

The resulting magnitude of thermal damage from the simulated experiments at two key axial (y direction) locations are also shown in Fig. 6.4. The locations shown are the surface of the heater, $y = 0$ and the point of ablation furthest away from the heater, $y = 9.25mm$. From these curves, the predicted values of H_{ab} and D_{ab} can be found, just as in 6.1. Table 6.1 summarizes the results of these two tests. Notice that after the two minimization tests, the objective function value is still on the order of 10^5. With regards to this the following observations should be made:

- The final value of the objective function must be thought of relative to its initial value.

  - Figure 6.5 shows that in the minimization algorithm reduced the objective function almost 20 orders of magnitude before it output the protocol for use in Test 2 and 10 orders of magnitude for test 1.
  - Figure 6.6 shows the evolution of the control function used in test 1 from initial guess to final the result. Note that the over the last several iterations there is very little change.
  - This "bottoming out" effect is also reflected in the objective function value shown in Fig. 6.5.

- It is also important to remember that the objective function is a function of the damage profiles squared. It is common for portions of the resulting damage field to be on the order of 10^5, because damage occurs rapidly at high temperatures. When the algorithm guesses higher than the optimal protocol the objective function value can be very large, See Fig. 6.5.

- Despite the above points, further research into the minimization technique may produce lower values of $\mathcal{J}(\Gamma(t))$. For example, penalty functions can be included to dissuade the algorithm from guessing updated control function which cause excessive damage.

Table 6.1 Minimization algorithm results summary: *Test 1* and *Test 2*

	Test 1	*Test 2*
No. Iterations	23	52
CPU Time (sec)	431	1230
$\mathcal{J}(end)$ (Dimm)	$1.41 \cdot 10^5$	$1.55 \cdot 10^5$
$E_{RMS}(end)$ (Dimm)	$4.68 \cdot 10^2$	$3.87 \cdot 10^2$

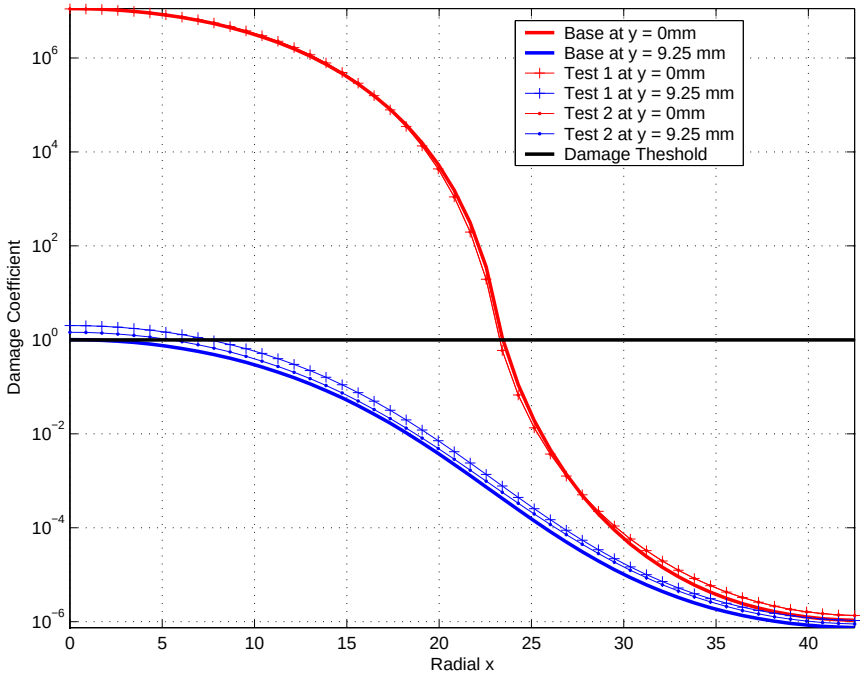

Fig. 6.4 Damage coefficient vs. radial distance from center for baseline, *Test 1* and *Test 2* simulations.

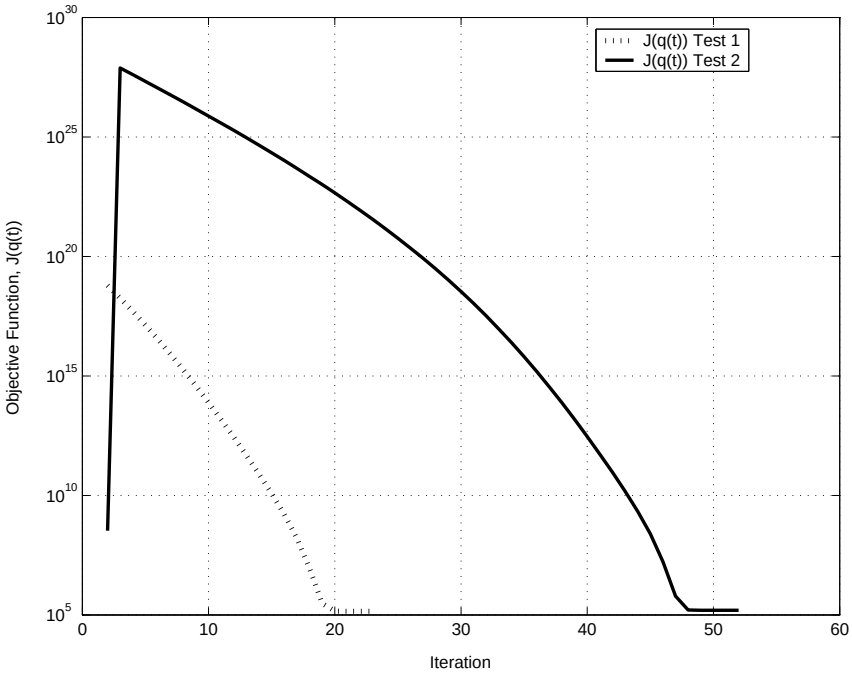

Fig. 6.5 Objective function vs. iteration for *Test 1* and *Test 2*.

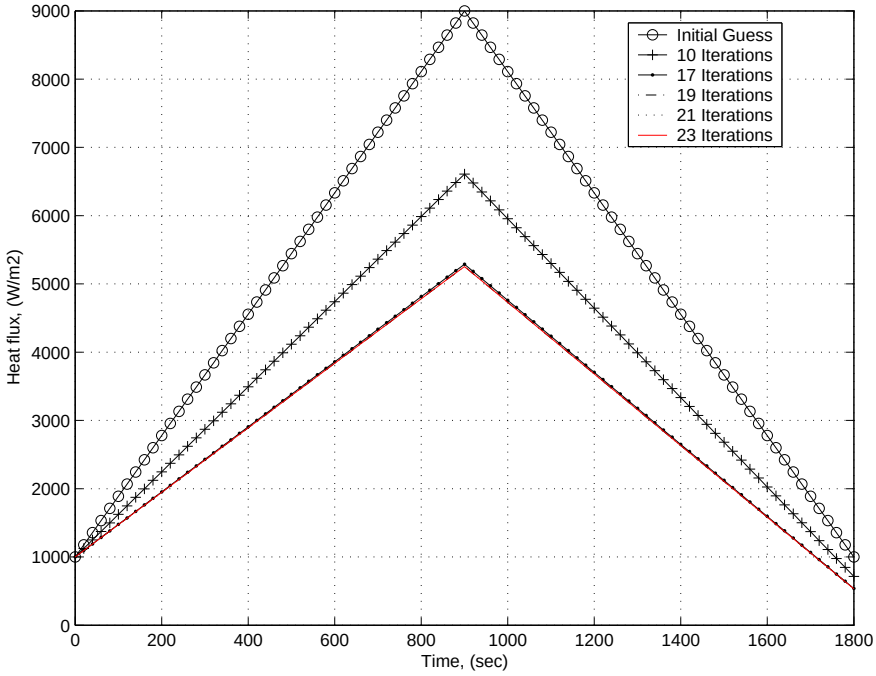

Fig. 6.6 Evolution of control function vs. iteration in simulation solving for optimal q"(t), *Test 1*.

Non-uniqueness of the damage integral

In Fig. 6.3 each of the heating protocols used in the validation experiments are shown. In the cases of tests 1 and 2 the corresponding initial guess given to the algorithm is also shown. On first glace it may seem unlikely that these functions all cause the same thermal damage field. This discrepancy brings up a key point: there is not one unique solution to this control problem. In other words, heat flux protocols which deliver precisely the same energy can cause different damage fields. Conversely, heat flux protocols which do not sum to the same amount of energy can cause nearly identical damage fields. This point is illustrated in the following example. Each heating protocol in Fig. 6.7 delivers the same amount of energy $\left(\int_{t_o}^{t_f} q_1" \, dt = \int_{t_o}^{t_f} q_2" \, dt = \int_{t_o}^{t_f} q_3" \, dt\right)$ but all cause different extents of damage, Fig. 6.8.

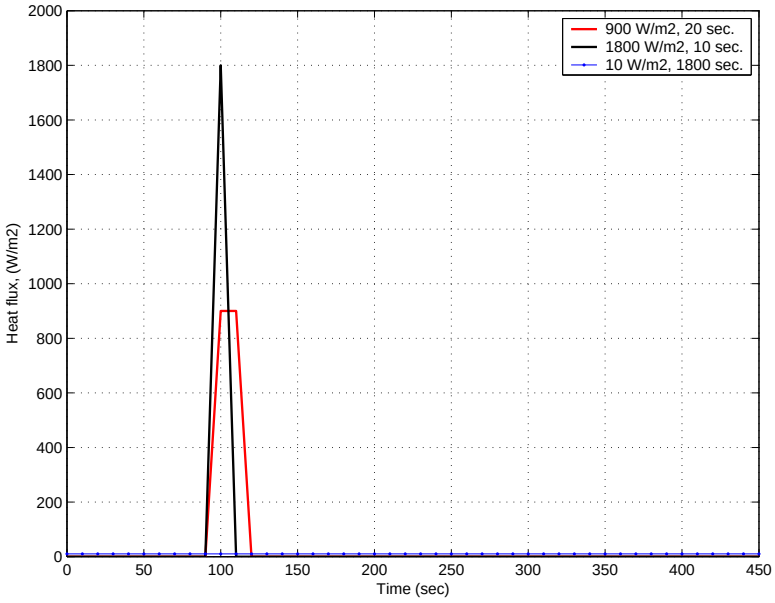

Fig. 6.7 Three control functions which integrate to give equal amounts of energy, but different amounts of thermal damage.

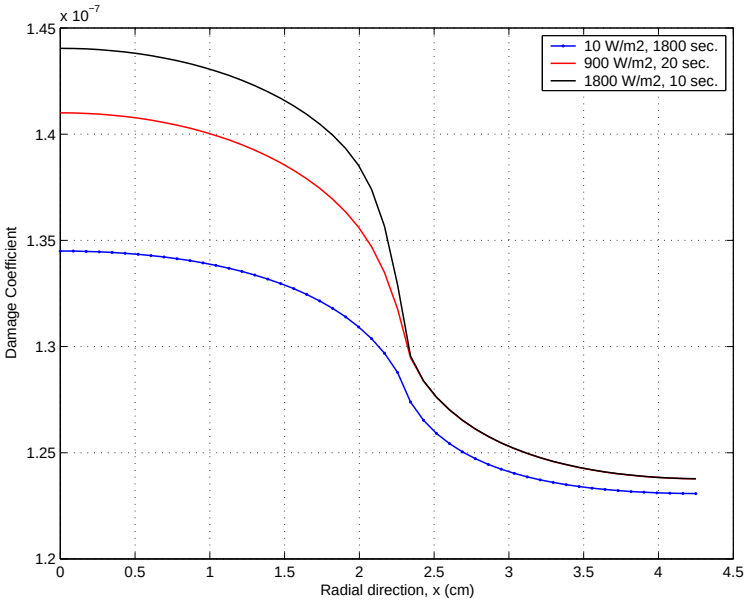

Fig. 6.8 Results damage profiles from control functions shown in Fig. 6.7.

6.1.2 Validation experiment: Laboratory results

The heat flux protocols from *Test 1* and *Test 2* were converted to voltages via Eqn. (5.7) and applied to the tissue phantom. A picture taken during on the validation experiments is shown below in Fig. 6.9, notice the ablated portions of the albumen are visible.

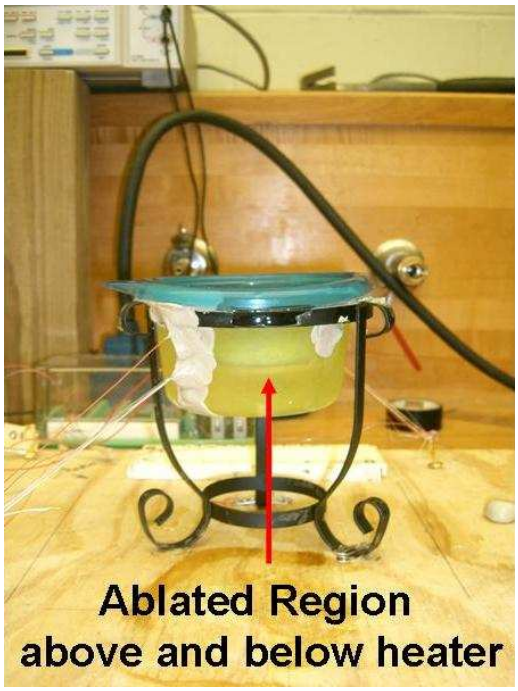

Fig. 6.9 Test stand No. 2 and natural albumen phantom during a heating protocol. Ablated portions of the albumen around the heater are visible.

The simulated and experimental temperature profiles at the thermocouple locations are shown in Figs. 6.10 and 6.11. The rise above the initial temperature is shown in both cases. The simulation results are shown at the surface of the heater and $\approx 3\,mm$ above the surface to account for the uncertainly of the axial location of the thermocouple bead.

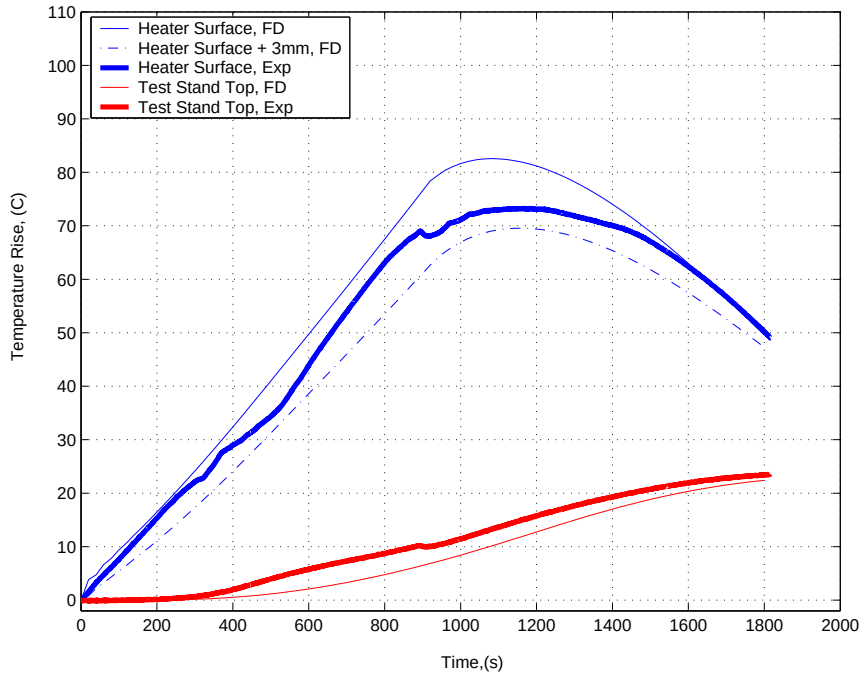

Fig. 6.10 Temperature rise vs. time at thermocouple location; *Test 1*.

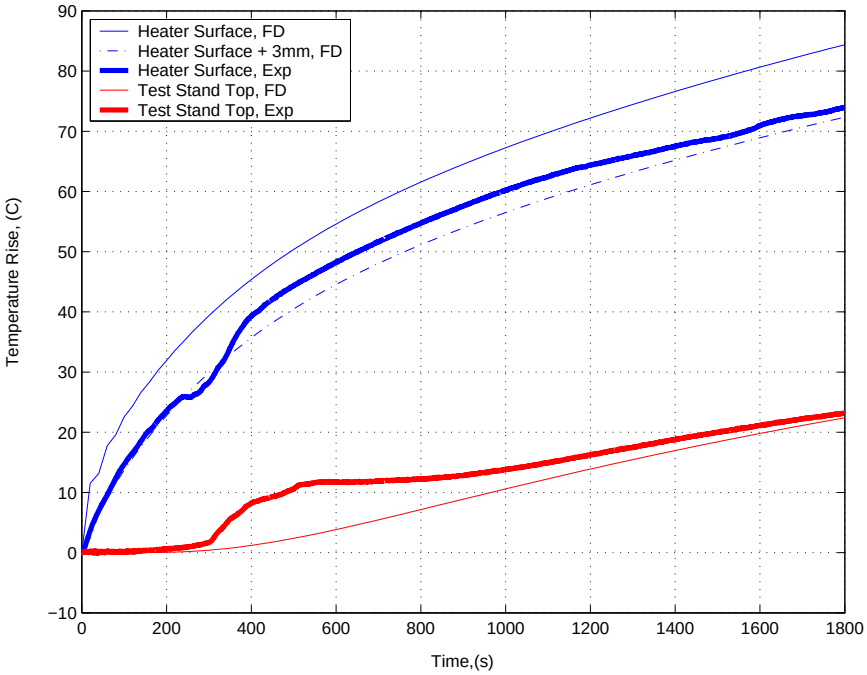

Fig. 6.11 Temperature rise vs. time at thermocouple location; *Test 2*.

Like in Fig. 5.23, Figs. 6.10 and 6.11 show that the finite difference model with the temperature dependent internal convection does a good job of predicting the transient temperatures distribution within the phantom. In both tests, the curves at the heater surface are completely within the 0 to 3mm window.

After each test, the liquid albumen was drained from the test stand thus isolating the dome-like portion of thermally ablated albumen. The heater and albumen were then carefully removed from the test stand. The corresponding damage above and below the heater was immediately measured with a pair of dial calipers accurate to $2.54 \cdot 10^{-3}$ cm. Table 6.2 compares the damage metrics measured after Tests 1 and 2 with the corresponding simulation value. The values reported in Table 6.2 are from the top side of the heater only, values reported from the bottom side of the heater are reported in Table 6.3. No simulated values below the heater are available because the internal convection was calibrated from the baseline case on the top side of the heater only . Figure 6.12 shows a side view of *Test 2*, it is obvious that more albumen is coagulated on top than on the bottom.

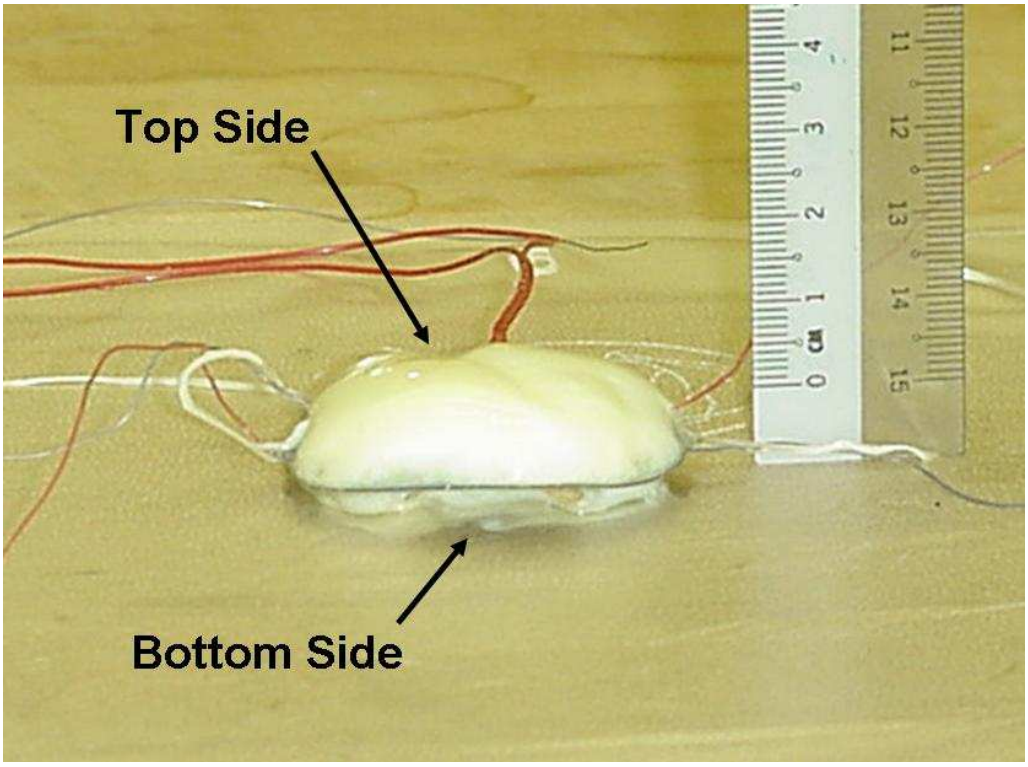

Fig. 6.12 Results from Test 2 shown next to cm scale. The region above the heater experienced much greater coagulation.

Table 6.2 Comparison of experimental and simulated values of D_{ab} and H_{ab}, top side of heater

Top Side	H_{ab} Exp. (mm)	H_{ab} Sim. (mm)	% Error
Baseline	6.10	9.25	35%
Test 1	9.78	9.25	5.7%
Test 2	10.9	9.25	17%
Top Side	D_{ab} Exp. (mm)	D_{ab} Sim. (mm)	% Error
Baseline	43.9	46.8	6.2%
Test 1	43.7	45.2	3.3%
Test 2	44.7	46.8	4.4%

Table 6.3 Experimental values of D_{ab} and H_{ab} on bottom side of heater for *Test 1* and *Test 2*

Bottom Side	H_{ab} Exp. (mm)	D_{ab} Exp. (mm)
Baseline	5.59	38.2
Test 1	6.35	38.6
Test 2	5.05	45.5

6.1.3 Analysis of validation experiment

Table 6.2 shows that the error associated with the radial extent of the damage is on average much lower than the axial error. As predicted, the baseline case shows the least agreement with the axial extent of the damage, H_{ab}. Recall that the simulated damage profile for the baseline case was calculated **without** including convective effects. Without internal convection included, the simulated temperature profiles are higher than they would actually be in the phantom. Consequently the the simulated extent of damage would be larger than the extent of damage found experimentally for the same heating protocol. This explains for the discrepancy and shows the importance of including internal convection.

Recall that Tests 1 and 2 used the same optimal damage profile as the baseline case, but the simulations included the internal convection term discussed in the previous chapter and found in Fig. 5.22. *Note: Tests 1 and 2 are not to be confused with Test stand No.1 and Test stand No.2.* Looking at the damage metrics for *Test 1* and *Test 2* in Table 6.2, one can see that the error is reduced. For *Test 1* the error between simulated and experimental damage extent is reduced six-fold and for *Test 2* the error is reduced by half.

The comparison between the experimental and predicted error for *Test 1* was quite good while for *Test 2*, the error was greater. There are a few mitigating factors that may

explain this inconsistency.

1. First while the desired damage field for all three of these test is the same, the recovered field, (Ω_c) is slightly different for *Test 1* and *Test 2*. This is known because the final value of the objective function was not reduced to zero for either *Test 1* or *Test 2*. The poor performance of *Test 2* is due in part to the fact that the value of the objective function at convergence is higher than that of *Test 1*.

2. While there is a difference in the objective function values, it is not more than one order of magnitude, which is small when one looks at the amount the objective function has changed over the minimization shown in Fig. 6.5.

3. In light of this, the large error associated with the last test may have to do with variability of the phantom itself. In all these cases natural albumen was used. Natural albumen is not a homogenous material and so variability from one test stand to the next may be partly responsible for the difference. In every egg used, both "thick" albumen and "thin" albumen was used in the test, adding the possibility for variations in the phantom (Yamamoto et al. (1997)).

Table 6.3 shows experimental measures of the extent of the damage below the heater. Notice that in Test 1 and Test 2 the axial damage, H_{ab} is much lower than the simulated values as well as the experimental values. Lower temperatures were recorded below the heater because buoyancy-driven motion carried warmer albumen towards the top of the test stand. Consequently, the extent of damage on this axis was less than predicted. The radial extent of damage, D_{ab} are closer to the simulated values but these are strongly influenced by the size of the heater. If a second calibration was done to fit the curve from the thermocouple readings on the bottom of the heater, one could expect agreement on par with the readings found in Table 6.2.

Parametric sensitivity of the finite difference and damage models

The results of the direct problem are only as accurate as the parameters it uses to calculate the transient temperatures within the phantom. In light of the potential variation of the phantom from test to test, it is important to see how sensitive the finite difference and damage models are to small changes in the parameters used in the simulation. If the error

on H_{ab} from *Test 1* and *Test 2* is within an acceptable uncertainly limit, then the validation experiment must be considered a success.

The sensitivity analysis begins by calculating sensitivity coefficients for each parameter which has the potential to affect the damage metrics. The sensitivity coefficient, X_i^+, is approximated as a linear change from a baseline damage field, Ω_{base}, see Eqn. (6.1). The baseline damage field was calculated using the same size and boundary conditions as the validation experiment and a constant heat flux of $1000\,W/m^2$. The "+" superscript is written because each term is non-dimensionalized.

$$X_i^+ = \frac{\partial \Omega_i^+}{\partial \Upsilon_i^+} \approx \frac{\Delta \Omega_i^+}{\Delta \Upsilon_i^+} \tag{6.1}$$

$$\Delta \Omega_i^+ = \frac{\Omega_{s,i} - \Omega_{base}}{\Omega_{base}} \tag{6.2}$$

The perturbed field, $\Omega_{s,i}$ is found by changing each parameter one at a time by a small amount, $\Delta\Upsilon$. For the present analysis, $\Delta\Upsilon = +1\%$ for all parameters. The simulation was run with the adjusted parameter and the resulting damage field was saved as $\Omega_{s,i}$. Equations 6.4 to 6.5 explain the rest of the method used in the sensitivity analysis. In this section, Υ_i is taken to be any thermophysical property or parameter involved in the sensitivity analysis.

$$\Delta \Upsilon_i^+ = \frac{\Upsilon_{s,i} - \Upsilon_{base,i}}{\Upsilon_{base,i}} \tag{6.3}$$

$$\Upsilon_{s,i} = \Upsilon_{base,i}\,(1 + \Delta\Upsilon) \tag{6.4}$$

$$\Delta \Upsilon_i^+ = \Delta\Upsilon \tag{6.5}$$

Figures 6.13 and 6.14 show the results from the sensitivity analysis in two different ways. The first is a bar chart of the value of the coefficients at the origin $(\mathrm{x}\,,\mathrm{y}) = (0\,,0)$. This corresponds to the center of the heater and the point of greatest thermal damage. The second is a graph of the sensitivity coefficients along the radial, $(\mathrm{x}\,,\mathrm{y}) = (\mathrm{x}\,,0)$. The maximum sensitivity coefficients at are shown below in Table 6.5.

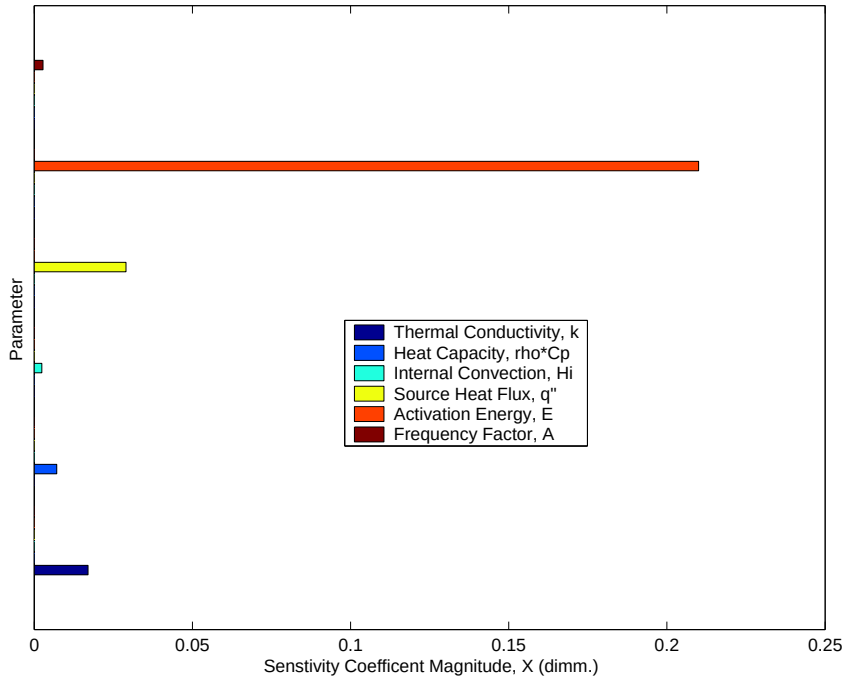

Fig. 6.13 Magnitude of sensitivity coefficients, X_i^+ for parameters of interest.

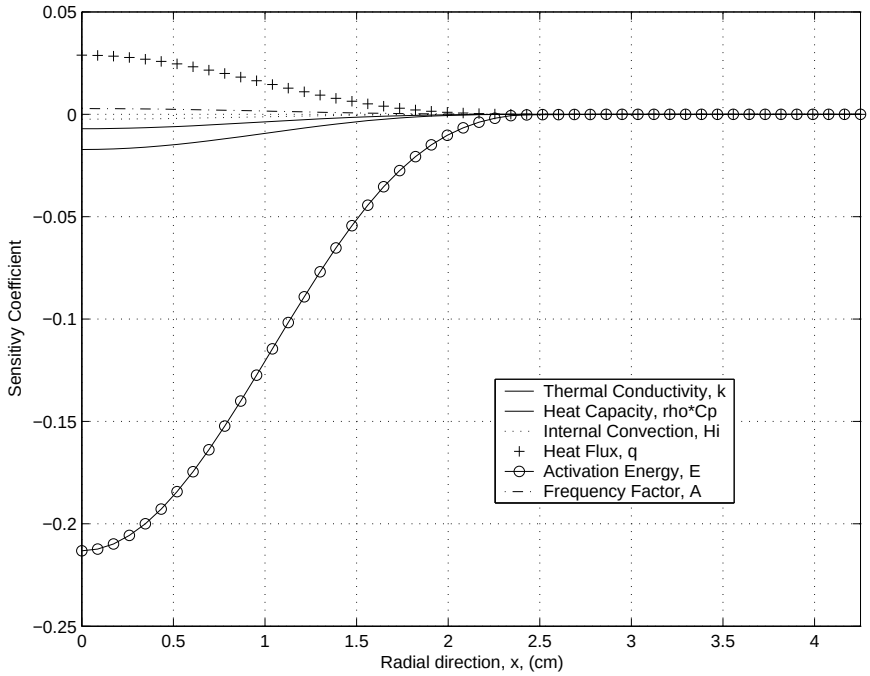

Fig. 6.14 Magnitude of sensitivity coefficients, X_i^+ along radial axis.

Figures 6.13 and 6.14 show that the model is very sensitive to changes in the activation energy. In fact, the model's sensitivity to the activation energy is nearly 10 times greater than the next most sensitive parameter, the heat flux. Another way of interpreting these results is that a small change in E will result in a 20 % change in the extent of damage. Referring back to Fig. 6.14, it is no surprise that the greatest sensitivity to parameters comes near the heater surface. This can be generalized to say that changes in the damage field will occur near the point or area of source energy entry.

Tolerable extent of damage in the validation experiments

There will be some amount of error between the experimentally found damage metrics from *Test 1* and *Test 2* and their simulated counterparts. But how much error is acceptable? From the sensitivity analysis the magnitude of change in model output relative to a 1% change in each of the key parameters was found. But in order to create a window in which the damage metric should be, it is essential to have some idea of the uncertainty of the parameters.

The uncertainty (σ_i) of the thermophysical properties of egg white is shown in Table 6.4. This table was compiled from several published sources in the literature. Liquid water is included because albumen has a high (88%) water content (Yamamoto et al. (1997)), (Pfefer et al. (2000)). The uncertainty, shown in percent of the mean value, is taken to be one standard deviation.

Table 6.4 Variation of thermophysical properties from the literature. Standard deviation shown in units of percent average value

Source	Value, k $W/m\,K$	Value, Cp $J/kg\,K$
Cebral et al. (2003)	0.5	3600
Pfefer et al. (2000)	0.56	4180
Heldman et al. (1977)	0.56	3642
Liquid Water (24^{o}C)	0.61	4180
Mean Value, μ	0.54	3807
Standard Deviation, σ	$\pm\,6.4\%$	$\pm\,8.5\%$

The uncertainty of the heat flux was taken as $\sigma_q = \pm 2\%$, which is the maximum error that the amplifier added to the voltage signal output, see table 5.1. For the remaining parameters, H_i, E and A, an uncertainty of $\sigma = \pm\,5\%$ was assumed due to lack of available published data.

Moffat (1988) provides a way of summing the uncertainty from each parameter in the model to find an overall model uncertainty, which is shown below in Eqn. 6.6. It is the basic equation for the analysis of uncertainty from a single sample. Each term in the summation represents the contribution made from the uncertainty in one parameter, Υ_i.

$$\begin{aligned} \frac{\delta\Omega}{\Omega_{base}} \cdot 100\% &= \left[\sum_{i=1}^{N_\Upsilon} \Big(X_i \cdot (\Upsilon_i\, \sigma_i)\Big)^2\right]^{\frac{1}{2}} \\ &= \left[\sum_{i=1}^{N_\Upsilon} \Big((X_i^+ \cdot \Omega_{base}) \cdot \sigma_i\Big)^2\right]^{\frac{1}{2}} \end{aligned} \tag{6.6}$$

In the above equation, X_i^+ is re-dimensionalized before it is used in the calculation. The value of the overall uncertainty is reported as a percentage of the maximum baseline damage coefficient. It is shown in Table 6.5.

It was known from the sensitivity analysis whether an increase in the parameter value would have an additive or diminishing effect on the extent of damage. From this information, two "worst case scenario" tests were performed using the direct problem. These scenarios used only parameter values at the extremes of the acceptable range. On the high side, each parameter was adjusted to produce the maximum extent of damage, on the low side, each was adjusted to produce the least damage. The same boundary conditions and initial conditions were used as in the validation experiments.

Table 6.5 summarizes the results of the sensitivity analysis. The overall uncertainty calculated via Eqn. 6.6 is included as well. The table shows the parameters, their baseline values, the variability of the parameters and the maximum value of each parameter's sensitivity coefficient. A + or - sign in the "Max" column refers to whether the value was increased or decreased from the nominal value to find the maximum extent of damage acceptable.

Figure 6.15 shows the results of the worst case scenario test on the baseline case. In the figure, the envelope which contains the expected range of D_{ab} is formed by the intersection of the high and low side curves with the threshold curve. Similar graphs were studied to find the envelope for the damage metrics of Tests 1 and 2. Figures 6.16 and 6.17 show the final results of the sensitivity analysis. In these graphs, the allowable envelope of the metrics are show by the error bars. The data are also shown in Table 6.1.3.

Table 6.5 Summary of sensitivity analysis. Nominal values, and uncertainties used in calculation of allowable range of damage metrics are shown. Sensitivity coefficients values at the origin are also shown. The High and Low columns refer to whether the parameter was increased or decreased to find the high-side and low-side limits of acceptable damage.

Parameter	Nominal Value, Υ_i	Uncertainty, σ_i	Sensitivity, X_i^+	Max	Min
k $(W/m\,K)$	0.56	$\pm 6.4\%$	$1.7 \cdot 10^{-2}$	-	+
$\rho\,Cp$ $(J/m^3\,K)$	$4.17 \cdot 10^6$	$\pm 8.5\%$	$7.1 \cdot 10^{-3}$	-	+
H_i $(W/m^3\,K)$	See Fig. 5.22	$\pm 5.0\%$	$2.4 \cdot 10^{-3}$	-	+
q" $(W/m^2\,K)$	See Fig. 6.3	$\pm 2.0\%$	$2.9 \cdot 10^{-2}$	+	-
E (J/mol)	$3.85 \cdot 10^5$	$\pm 5.0\%$	$2.1 \cdot 10^{-1}$	-	+
A $(1/s)$	$3.8 \cdot 10^{57}$	$\pm 5.0\%$	$2.8 \cdot 10^{-3}$	+	-
Overall Uncertainty	See Eqn. 6.6	$\pm 1.1\%$	NA	NA	NA

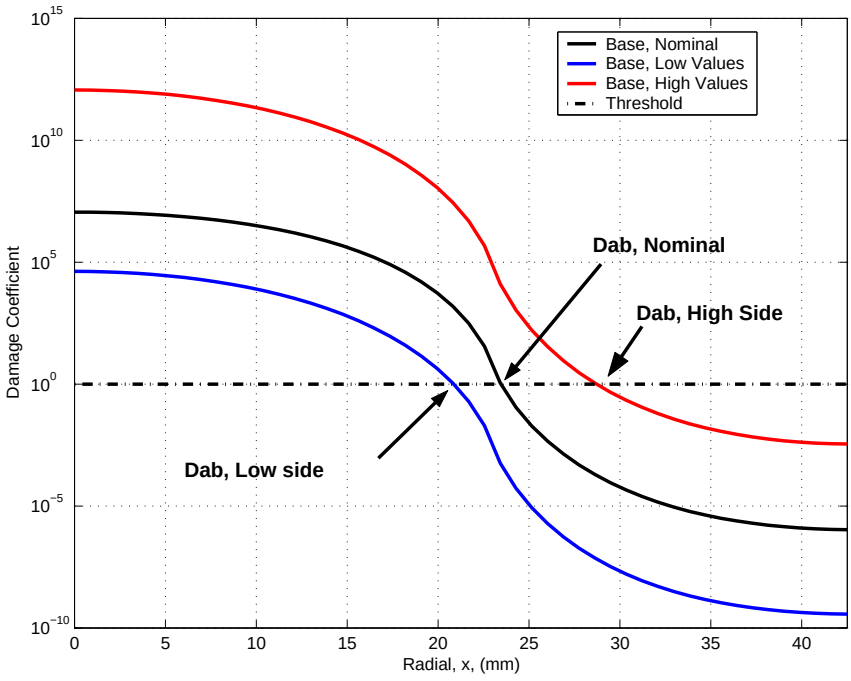

Fig. 6.15 Effect of worst case scenario variation on radial extent of damage, baseline heating protocol. Damage at the surface of the heater is shown, with D_{ab} being the extent of thermal ablation in the radial direction. Only half of D_{ab} is shown because the model is axially symmetric.

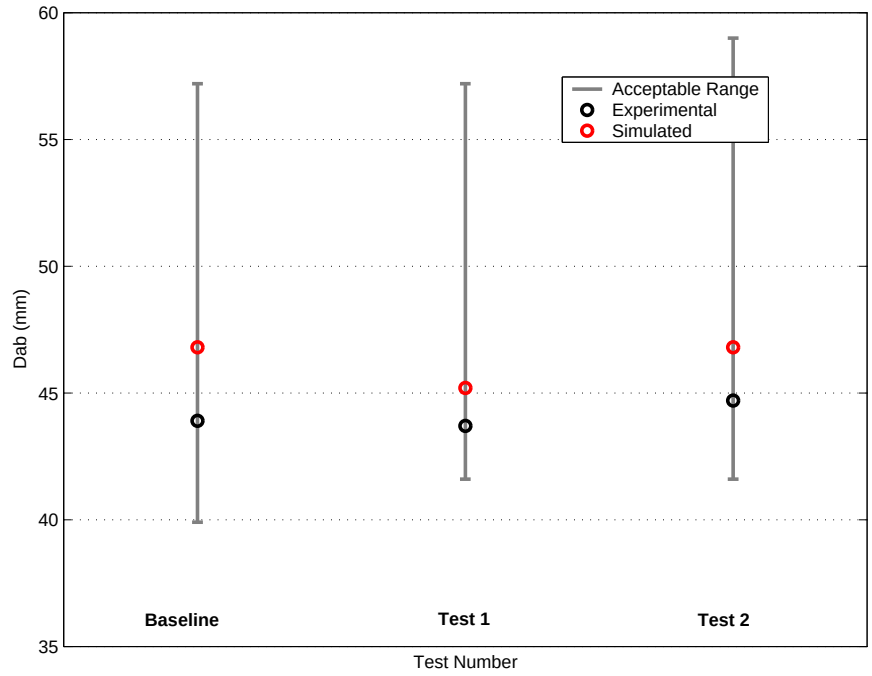

Fig. 6.16 **Extent of thermal damage above heater in the radial direction, D_{ab}. Each test is shown within a permissable damage envelope according to the sensitivity analysis.**

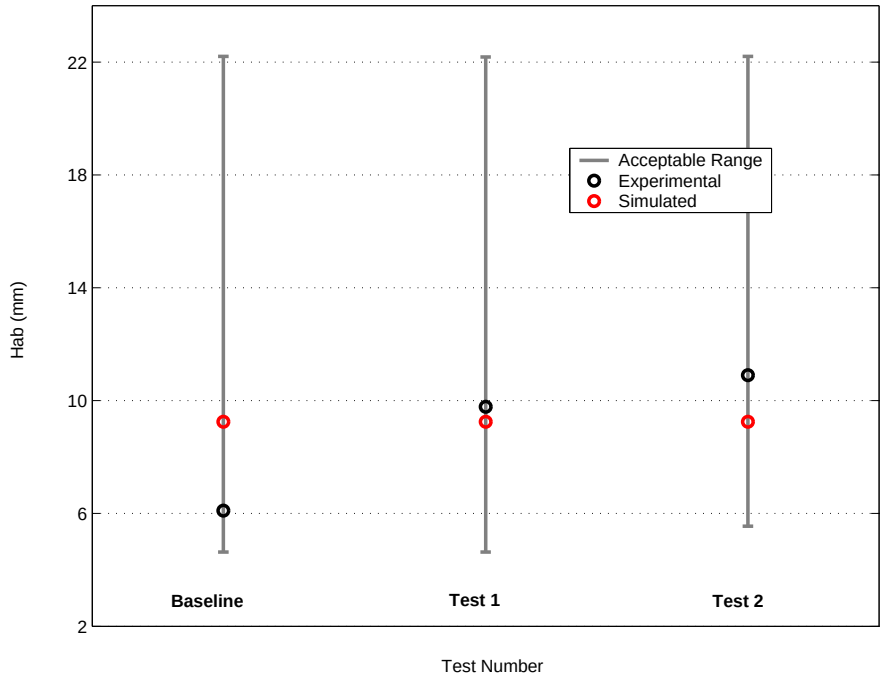

Fig. 6.17 **Extent of thermal damage in the axial direction above the heater, H_{ab}. Each test is shown within a damage envelope according to the sensitivity analysis.**

Table 6.6 Results from worst case scenario tests. Extent of damage envelope on high and low side.

Test No.	$D_{ab,High}$ (mm)	$D_{ab,Nom}$ (mm)	$D_{ab,Low}$ (mm)
Baseline	57.2	43.9	39.9
Test 1	57.2	43.7	41.6
Test 2	59.0	44.7	41.6
Test No.	$H_{ab,High}$ (mm)	$H_{ab,Nom}$ (mm)	$H_{ab,Low}$ (mm)
Baseline	22.2	6.10	4.63
Test 1	22.2	9.78	4.63
Test 2	22.2	10.9	5.55

Figures 6.16 and 6.17 show that the measured values the damage extent are well within the allowable brackets. However since the brackets are quite large this results is not surprising. For example, the baseline case which **did not** include internal convection is also within the allowable brackets . The allowable range of H_{ab} extends to the top of the test stand for all cases. From the sensitivity analysis it is obvious that the bracket size is dominated by the change in activation energy.

In summary, there are some important conclusions which should be drawn from the sensitivities analysis and validation experiment.

- It is essential to have a high-confidence in the value used for the **activation energy** if a thermal damage model is to be used to quantify thermal damage.

- Because of the sensitivity of the damage coefficient to points of entry of heat energy, discrete models used in planning treatments should have a **higher mesh density** near the point of entry of the source.

- **Albumen** proved worked well as a phantom in the validation experiments but mixing due to internal convection must be included.

  - Powder based albumen proved to be too inviscid for accurate modelling with the bioheat equation.

  - Even with natural albumen, the internal convection within the medium added additional uncertainty to the model.

  - It is quite possible that the material varied from one test to next since albumen is a naturally inhomogeneous liquid.

Table 6.7 Parameters used in simulated hyperthermia treatments to human skin. (Diller (1992))

Parameter (Unit)	Value	Parameter (Unit)	Value
Cp_{tis} $(J/kg\,K)$	4000	ρ_{tis} (kg/m^3)	1040
Cp_{bl} $(J/kg\,K)$	3300	ρ_{bl} (kg/m^3)	1100
$k_{epidermis}$ $(W/m\,K)$	0.21	ω_{bl} $(kg/m^3\,s)$	$2.4 \cdot 10^{-2}$
k_{dermis} $(W/m\,K)$	0.37	h $(W/m^2\,K)$	8.5
k_{fat} $(W/m\,K)$	0.16	T_o (^oC)	37
E (J/mol)	$6.28 \cdot 10^5$	A $(1/s)$	$3.1 \cdot 10^{98}$

6.2 Simulation Results

The program written in MATLAB, **Damage Dose Optimizer 3.0** was designed not only to simulate the validation experiment, but a wide variety of treatment situations. In this section, several simulation results are presented, showing the robustness and versatility of this program. All the following simulations are based on a volumetric heating protocol applied to human skin. The property values used are shown in Table 6.7 (Diller (1992)). Unlike the validation experiment the source function is volumetric acting over a quarter of the control volume, $Q(t)$. Cartesian coordinates are used in all cases. First simulations results from a simplified model are presented, followed by results from a more complex model.

6.2.1 Simplified skin simulation

The simplified model of the simulations is shown below in Fig. 6.18. A convective boundary on the top of the skin allows heat to escape, a symmetry plane is used to the right, and the other two boundaries are a constant temperature of 37^oC. The material properties are homogeneous, using the thermal conductivity of the dermis and the specific heat and density of tissue. A constant blood perfusion of $W = 3300W/m^3\,K$ is used.

Simulations with the objective function based on the thermal damage coefficient and thermal dose are shown. Both fields were created with a constant volumetric heating of 500 kW/m^3 acting over a quarter of the domain. The effect of parametrization is also shown, with the number of parameters used shown as N_β. If $N_\beta = \infty$ for a case, this means that the gradient of the objective function was not parameterized.

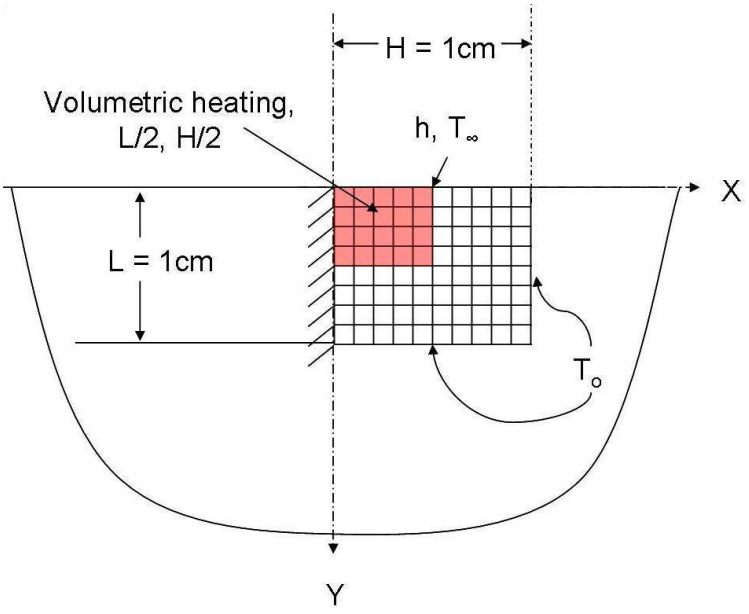

Fig. 6.18 Schematic of simplified skin model used in hyperthermia treatments.

Performance of minimization algorithm: Damage vs. Dose minimization

The minimization algorithm was used to perform simulations which found an optimal dose or damage field. In both cases, an optimal field was found via an arbitrary control function. This optimal field then was used as the input to the control algorithm. The minimization algorithm converged in most cases, but the initial guess has a large effect on whether or not the program will converge. If the user has a rough sense of the magnitude of energy needed in a treatment, it is suggested to guess higher. Lower guesses cause the algorithm to over compensate and take longer to reach a converged value. See Fig. 6.5.

Table 6.8 Minimization algorithm results summary: Simplified skin model

Model	Damage, Ω,	Damage, Ω,	Dose, D
Parametrization, N_β	∞	3	∞
No. Iterations	113	15	299
CPU Time (sec)	388	48	1071
$\mathcal{J}(end)$ (Dimm)	$2.48 \cdot 10^{-8}$	$5.31 \cdot 10^{-7}$	449
E_{RMS}	$1.09 \cdot 10^{-3}$	$1.48 \cdot 10^{-2}$	$5.1 \cdot 10^{-3}$

Table 6.8 gives a review of the performance of the minimization algorithm as it recovered heating protocols for the simplified model. Figure 6.19 shows the recovered damage and dose profiles and the difference between each and the optimal. From Table 6.8, we can see that the final objective function for the damage minimization is much less than that of

the dose minimization. This is also reflected in Fig. 6.19. The convergence problems may have to do with the relative magnitudes of the models themselves. The damage model's magnitude is about 5 orders smaller than the dose models. In the minimization algorithm these numbers are squared in the objective function calculation, so the dose model generates much larger numbers.

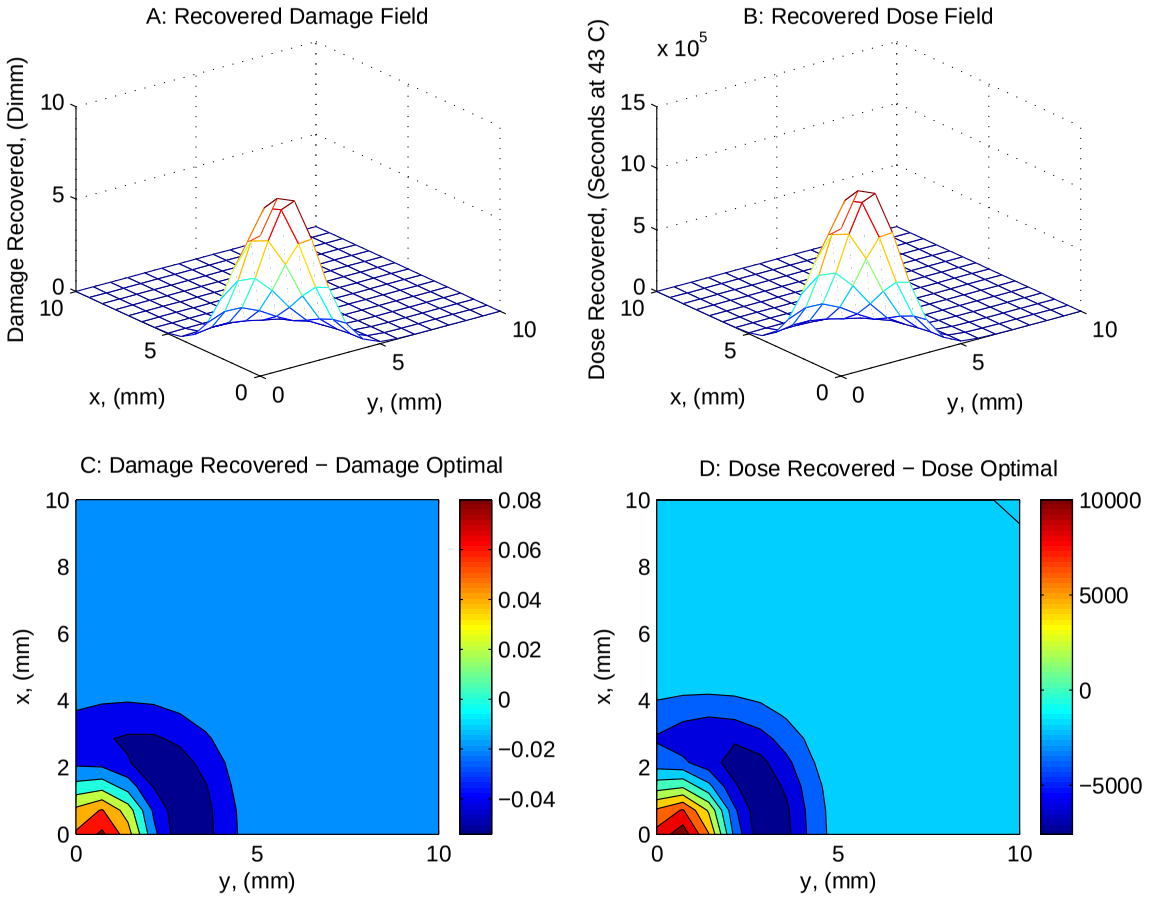

Fig. 6.19 Model results for non-parametrized minimization of thermal damage and thermal dose. A. Recovered thermal damage profile. B. Recovered thermal dose profile. C. Difference between recovered and optimal damage profile. D. Difference between recovered and optimal dose profile.

Figure 6.20 shows how the magnitude of the objective function changes each iteration for the simulations performed on the simple skin model. Notice that both the damage and dose models reduce their objective function values roughly 10 orders of magnitude. While the final objective function value are much lower for the damage integral minimization, the algorithm is equally effective on both models.

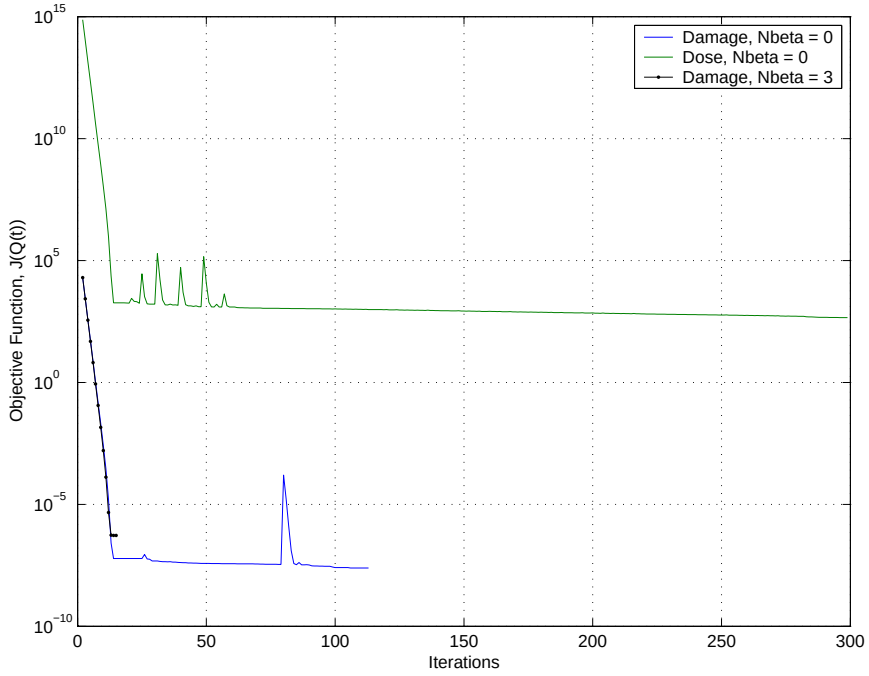

Fig. 6.20 Objective function for damage model, dose model and parameterized damage model.

The effect of parametrization

In chapter 4, a method parameterize the objective function was introduced. This function forces the algorithm to chose solutions that are constructed via a cubic spline interpolator. The effect of parametrization can reduce computational time of the model and create more clinically applicable control function, these benefits which are shown below in Fig. 6.21. Notice the parametrization keeps the shape of the initial guess. The parameterized damage minimization only required 15 iterations to reach the minimum relative to the non-parameterize model 113 iterations. The two non-parameterized functions are similar and oscillate around the heat flux protocol which was used to create the optimal field. Recall that these two minimization attempts were done with different objective functions.

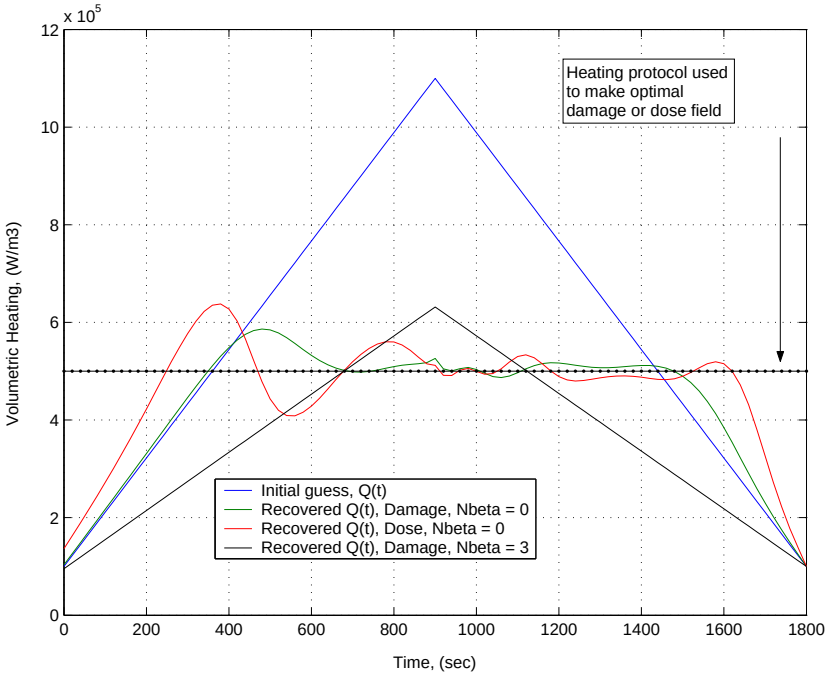

Fig. 6.21 Recovered control functions from simulations on simplified skin model. Heating protocol used to make the optimal field is also shown.

6.2.2 Complex skin simulation

In the complex model the simulation boundary conditions are conserved, but the properties are adjusted to match a more physiologically realistic situation. Figure 6.22 shows the added complexity. The epidermis, dermis and underlying fat are included in the model. The blood perfusion value in this simulation is larger and consistent with the values reported in Diller (1992). Additionally, only the dermis layer of the model is perfused, the fat and epidermis are not.

Again, simulations with the objective function based on the thermal damage coefficient and thermal dose are shown. Both fields were created with a constant volumetric heating of 1000 kW/m^3 acting over a quarter of the domain. The increased energy in is required for thermal ablation because of the increased blood perfusion. One addition simulation was performed with the complex model which included a temperature dependent blood perfusion. The perfusion was allowed to decrease linearly from its nominal value of $W = 8.7 \cdot 10^4 (W/m^3\,K)$ at 43^{o}C to a lower value of $W = 1.0 \cdot 10^4 (W/m^3\,K)$ at 100^{o}C. A summary of the minimization algorithm output is shown in Table 6.9.

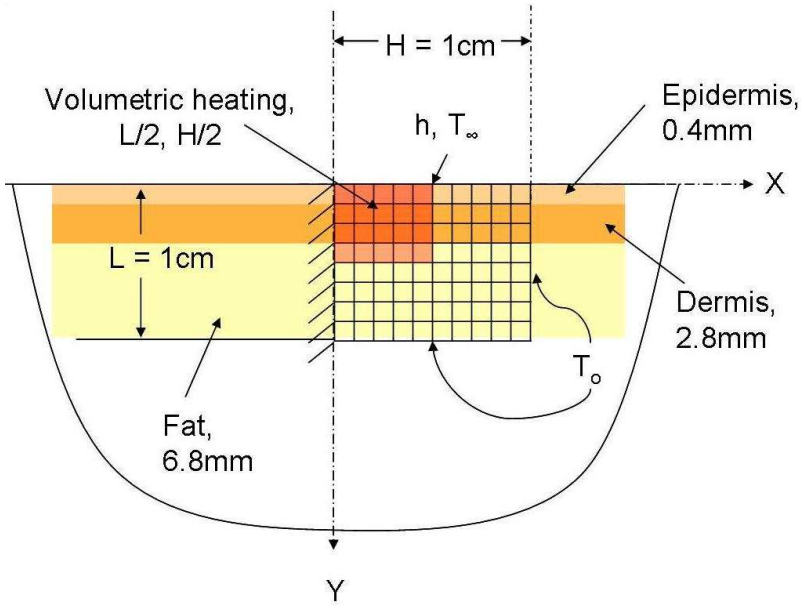

Fig. 6.22 Schematic of skin model used in hyperthermia treatments with added complexity.

Table 6.9 Minimization algorithm results summary: Complex skin model

Model	Damage, Ω	Dose, D	Dose, D	Damage Ω
Perfusion in Dermis	W $\neq f(T)$	W $\neq f(T)$	W $\neq f(T)$	W $= f(T)$
Parametrization, N_β	∞	∞	50	50
No. Iterations	91	86	91	149
CPU Time (sec)	407	389	400	716
$\mathcal{J}(end)$ (Dimm)	$7.11 \cdot 10^{-9}$	283	290	$4.99 \cdot 10^{-13}$
E_{RMS}	$5.6 \cdot 10^{-4}$	$1.12 \cdot 10^{2}$	$1.13 \cdot 10^{2}$	$4.69 \cdot 10^{-6}$

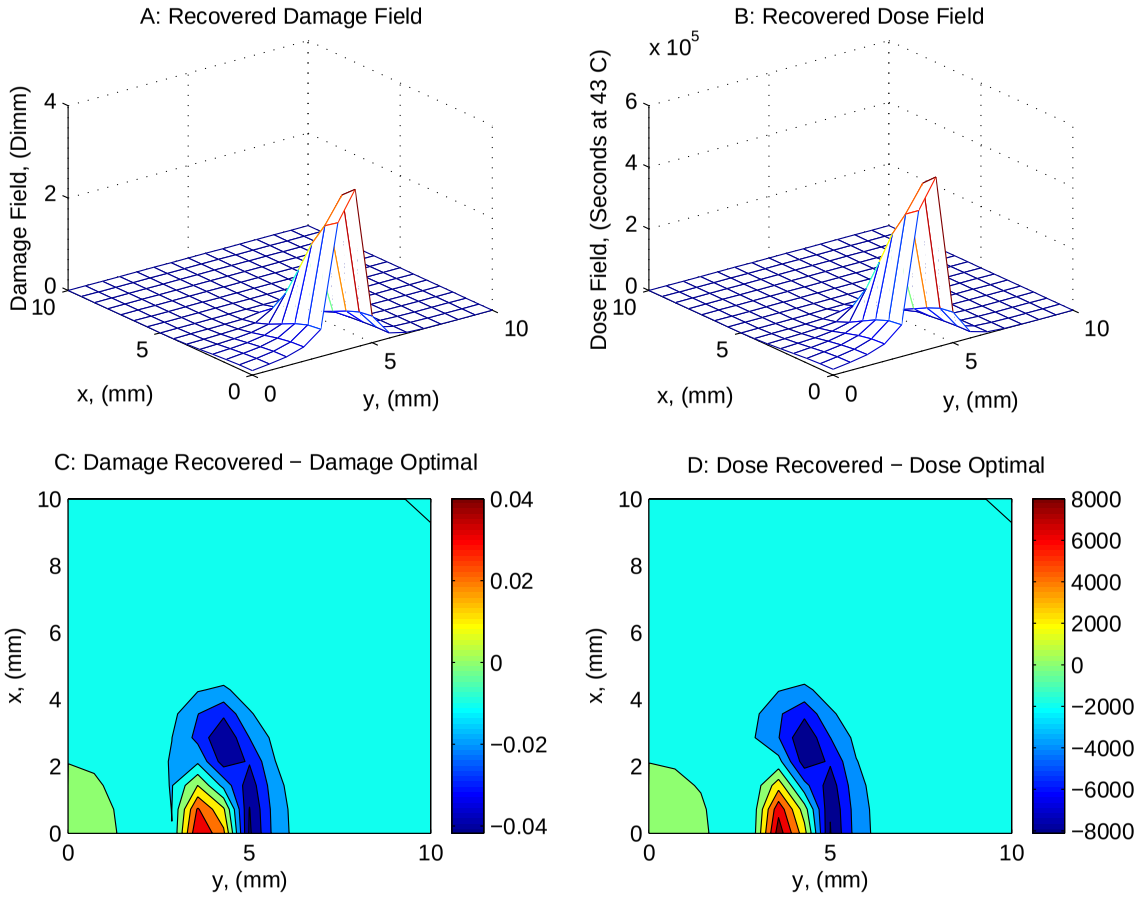

Fig. 6.23 Model results for non-parameterized minimization of thermal damage and thermal dose, complex skin model. A. Recovered thermal damage profile. B. Recovered thermal dose profile. C. Difference between recovered and optimal damage profile. D. Difference between recovered and optimal dose profile.

Performance of minimization algorithm: Complex skin model

Figure 6.23 shows results from damage model and dose model simulations on the complex skin model without parametrization. The elevated blood perfusion in the dermis causes less damage in this area, which is expected. The damage spikes in the fat area because no perfusion is present. For this simulation instead of a triangular heating a square pulse was used as the initial guess. The resulting control functions for all simulations on the complex model are shown below in Fig. 6.24.

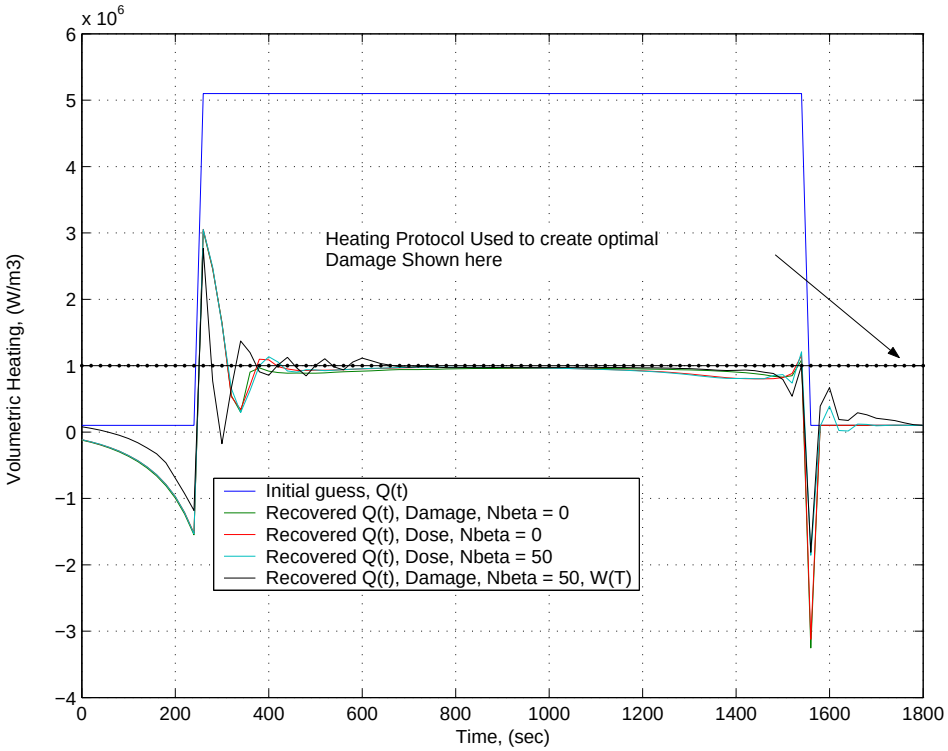

Fig. 6.24 Recovered control functions from simulations on complex skin model.

Looking at Fig. 6.24 it is evident that little is gained by parameterizing the control function when the initial guess is a pulse function. This is evidence that the spline scheme must be retooled for better performance on all guesses, not just triangular ones. Notice that these results are not clinically useful because the protocols call for wide swings of heating and cooling. These results show that the algorithm has trouble with sharp changes in the control function. One positive note is that the optimal fields used in these simulations were created with a constant heating of $Q(t) = 1000kw/m^3$. While the initial guess was much higher than this, it seems like the algorithm is reducing the control function to a magnitude

near the optimal Q(t).

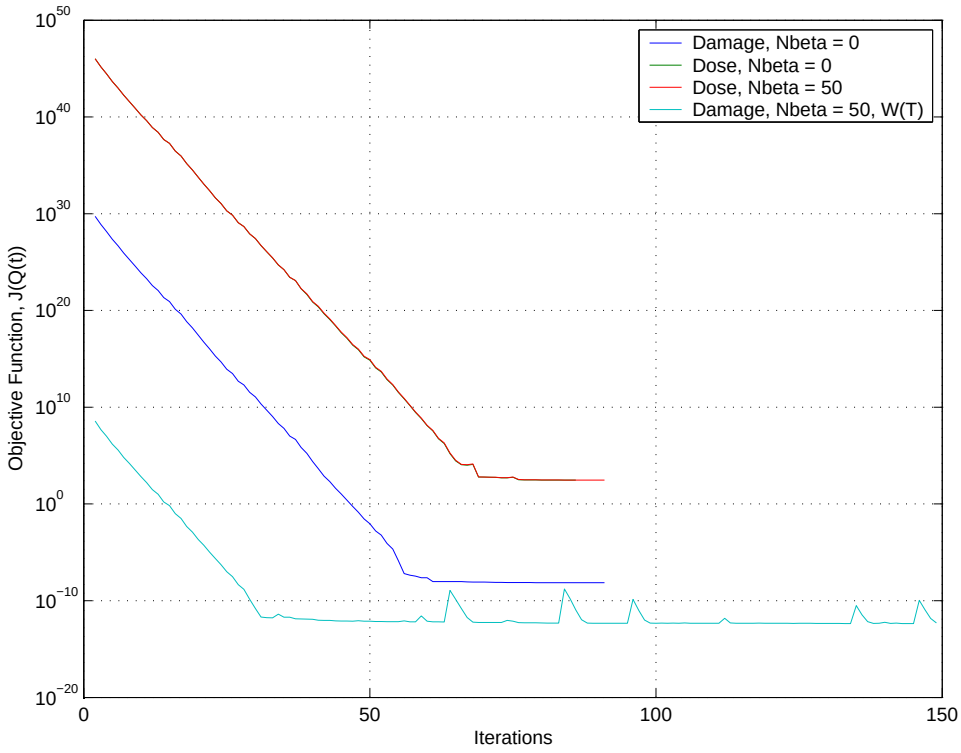

Fig. 6.25 Objective function vs. iteration for simulations on complex skin model.

Figure 6.25 shows the objective function vs. iteration throughout the minimization process. There is practically no difference between the parameterized and non-parameterized objective functions. Looking at Table 6.9 it appears that the damage optimizer with the perfusion as a function of temperature is actually the best performing minimization. This is based on the fact that the optimization algorithm achieved the lowest value. However, looking at Fig. 6.25 this simulation had the lowest initial value of the objective function and reduces the objective function the least orders of magnitude.

Some salient results from the simulation study are now presented.

- The minimization algorithm is robust enough to solve problems in cylindrical as well as cartesian coordinates, and for a variety of control functions, boundary conditions, and objective functions.

- The algorithm converges better if the initial guess much higher than an expected solution.

- In general the optimizations base on the damage coefficient reduce the value of the objective function to a lower value than dose optimizations. The algorithm reduces the magnitude of $\mathcal{J}$ of both the damage and dose models the same amount of orders of magnitude over the course of the minimization. See Figs. 6.25 and 6.20.

- Simulation times are generally longer for the thermal dose model. The thermal dose model is inherently more difficult to calculate because of the added feature of R (lumped parameter containing activation energy) depending on temperature, See Eqn. (3.43).

Chapter 7

Conclusions

As hyperthermia becomes a more commonly used modality in the treatment of cancer, the importance of treatment planning becomes more evident. Having knowledge of the what the effects of a heating protocol will be before and during a treatment is a powerful tool that doctors and researchers must take advantage of now that computational resources allow for it. This research attempted to show how a simplified hyperthermia treatment can be controlled and optimized. The specific objectives of this research and remarks on the degree success of meeting these objectives are now presented.

7.1 Objective 1: Solving the Optimal Control Problem

The major objective of this research was to incorporate the BHTE and two different means of evaluating thermal damage (thermal damage and thermal dose) within a problem of optimal control to recover a desired heating protocol for use in a hyperthermia treatment. This objective was met with a great deal of success. The resulting program, Damage Dose Optimizer 3.0, found in Appendix D shows the program in its entirety.

The base of the program is a two-dimensional finite difference model which was derived in Chapter 3. The robustness and versatility of the model was demonstrated in Chapters 5 and 6. The model allows for cylindrical and radial coordinate systems, boundary or volumetric source functions, regionally varying and / or temperature dependent thermal properties and well as any combination of boundary conditions of the first and second kind. A users manual is found in Appendix C. The program was successful at modeling both the validation experiment which was a cylindrical vessel filled with an albumen tissue phantom

and a cartesian coordinate model simulating a heat treatment on human skin.

Results from the optimal control problem were shown in Chapter 6. The control problem is the true backbone of the research because it alone is responsible for finding the optimized heating protocol. The final value of the objective function is the most accurate means of assessing how closely the algorithm matched the optimal and recovered damage or dose fields. The following conclusions can be made with regards to the algorithm:

- **The Henriques damage model** is preferred over the thermal dose model for use with the minimization algorithm.
  - First, the Henriques model is less computationally intensive than the thermal dose model. This can be seen in the CPU times shown in Tables 6.8 and 6.9.
  - Second, the Arrhenius rate parameters are actually calibrated to an observable physiological change in the protien structure of the solid under study. In the case of egg white, the activation energy, E, and molecular frequency factor, A, are chosen to coincide with visible coagulation. The thermal dose model is not tied to any physiological event.

  In light of this stated preference, it should be noted that the minimization algorithm performed equally well minimizing the thermal damage or thermal dose. In most cases, an arbitrary heating function was chosen to create the "optimal" damage or dose field. If the same arbitrary heating function and initial guess was used to create the optimal field, the algorithm minimized the objective function based on damage or dose by the same amount of orders of magnitude, see Figs. 6.20 and 6.25. In all cases however the final value of $\mathcal{J}$ is higher for the thermal dose because its magnitude is greater than the thermal damage for the same transient temperature field. For example in the simple skin model, the final values for the objective function were $\mathcal{J} = 5.31 \cdot 10^{-7}$ for the thermal damage coefficient optimization and $\mathcal{J} = 449$ for the thermal dose optimization.

- The algorithm is extremely sensitive to the initial guess of the control function and erring on the high side is recommended strongly. If $\Gamma(t)^{s=1}$ is too low, the algorithm overcompensates and the next guess, $\Gamma(t)^{s=s+1}$ will causes far more thermal damage than required. In some cases, this will cause an increase in the objective function

which can be more than 16 orders of magnitude. Since MATLAB works on a 16 digit format this will cause the algorithm to fail. See Figs. 6.5 and 6.25, the former shows the result of guess too low, notice the jump in the value of $\mathcal{J}$. It is not present in Fig. 6.25 because the initial guess was much higher than the energy need to cause the optimal damage.

- The algorithm was tested with two cartesian models of varying complexity of the human skin. The first model used uniform thermal properties and blood perfusion, see Fig. 6.18. The second is more complex with regionally changing properties and perfusion values, see Fig. 6.22. One additional simulation was run on the complex skin model where blood perfusion linearly decreased over the temperature range. Comments on how the algorithm performs with temperature dependent properties are reserved for later. Tables 6.8 and 6.9 show that the increased complexity had little effect on the ability of the algorithm to recover the optimal damage or dose field. For example, the final value of the objective function for the damage model was on the order of $10 \cdot -7$ and on the order of 10^2 for the thermal dose regardless of complexity level. This shows the robustness of the algorithm.

- As mentioned earlier, the complex skin model was run for one case which included a temperature-dependent perfusion parameter. The blood perfusion linearly decreased with increasing temperature, from a maximum value of $W = 8.7 \cdot 10^4 (W/m^3\,K)$ at 43^oC to $W = 1.0 \cdot 10^4 (W/m^3\,K)$ at 100^oC. For this particular simulation, the change in the magnitude of the objective function from the first to the last iteration spans roughly 20 orders of magnitude. Simulations on the same model with the same initial guess but no temperature dependent properties reduced the objective function by nearly 40 orders of magnitude. Despite this fact, this particular simulation achieved the lowest value of the objective function of all simulations run, $\mathcal{J}^{s=23} = 4.99 \cdot 10^{-13}$. See Fig. 6.25 for more details. The algorithm is therefore sensitive to temperature dependent properties.

- Parameterizing the control function is an important feature in creating more clinically feasible heating protocols. The parametrization of the control function shows great promise for certain initial estimates of the objective function. For example, with a triangular initial guess of $\Gamma(t)$, the the algorithm converged much faster, and the

shape of the function remained linear. See Figs. 6.21 and 6.6 and Table 6.8. The same cannot be said for when the initial guess of the treatment is a pulse. This failure depending on a shape is a point where the algorithm can be improved. Perhaps a linear interpolation instead of a cubic spline interpolation would create better results. B-splines instead of cubic splines have been used in previous work which might provide better results (Loulou and Scott (2002)).

Comparisons were presented between the thermal damage and thermal dose based on their performance in the minimization algorithm. But since the algorithm can compute both the thermal dose and thermal damage coefficient models for the same heating protocol, boundary conditions and initial conditions, it can also compare the two directly. Both have been shown to work equally well in the optimization algorithm in terms of how many magnitudes the objective function is reduced, although the damage model usually results in lower final values of the objective function (i.e., more accurate final solution) and does so with less computational cost. In Figs. 6.19 and 6.23 the resulting damage and dose profiles are shown from minimizations on the skin model. Whether the algorithm was set to minimize based on the dose or damage model, the optimal field was created with the same constant heating function, Q(t); $500W/m^3$ for the simplified model and $1000W/m^3$ for the complex model. In both cases, the Henriques model damage field is about 5 orders of magnitude smaller than the thermal dose field.

Figure 7.1 shows the results of D_d/Ω_d for the simple skin model. While the profiles have similar shapes, they do not differ by a constant. If that were the case, Fig. 7.1 would have the same shape as the two profiles. The effect of the temperature dependence of R(T) in the thermal dose equation can be seen by the non-smooth section of the curve. The break temperature takes into account the temperature dependence of the activation energy, making the dose model more complicated. Whether more complicated model is a more accurate model was not a focus of this study. But taking the validation experiment, as a basis, particularly the results from *Test 1*, the complication might not be worth the added computational cost.

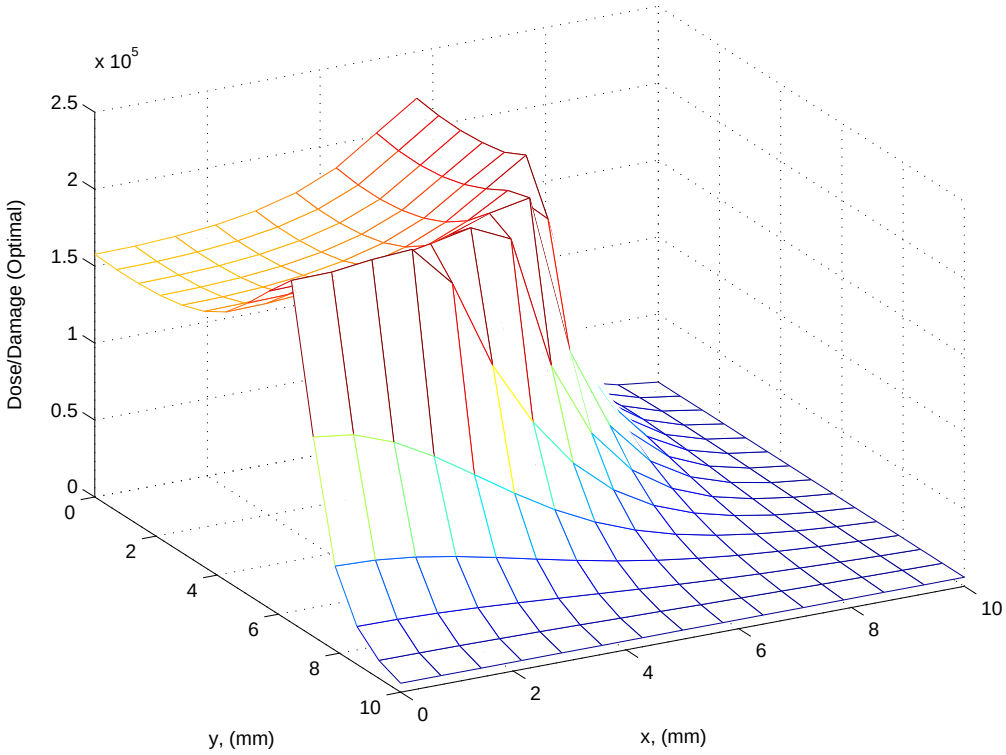

Fig. 7.1 Thermal dose field divided by thermal damage field. Both fields were created with a constant volumetric heating of 500 kW/m^3 acting over a quarter of the domain.

7.2 Objective 2: Perform an Experiment to Validate the Algorithm Output

The second major objective of this research was to validate the minimization algorithm findings via experiment. To perform this a variety of phantoms and test vessels were considered. In the end, the **natural albumen phantom performed better than powder based albumen**. Also, test stand No. 2, the 9 cm diameter, thin walled cylinder was found to be the best choice of test stand. Buoyancy-driven flow was found to have a significant cooling effect which had to be added to the model via an internal convection term, $H_i(T)$ to improve the accuracy of the extent of thermal damage.

The buoyancy driven cooling is a completely different phenomenon than perfusion in tissue, but its cooling effect is analogous allowing its use in the bioheat equation. The magnitude of H_i was found by fitting it to experimental temperature data taken at the surface of the heater via a guess and check method. Figure 5.22 shows how H_i varies with

temperature. Since the magnitude of internal convection was found empirically, an order of magnitude study justifying the values used is presented at the end of chapter 6.

A sensitivity analysis was performed on the model showing how small variations in the parameters used to predict thermal damage can effect the predicted extent of damage. The measured values of the extent of damage for three tests performed on the albumen phantom all fell within a tolerable error range. The following findings are important with regards to the experiment:

- Figures 6.16 and 6.17 show that the predicted damage metrics are well within the allowable ranges. In the case of *Test 1* the metrics are extremely close to the optimal metrics showing promise for further study of optimizing damage and dose fields on more realistic models. The slightly larger error values from *Test 2* are believed to be due to variation in the albumen phantom and a higher final objective function value than that of *Test 1*. Albumen is not a homogenous material, with each egg containing low and high viscosity albumen. It was difficult to know the exact proportions of thick and thin albumen within each test stand. Because of this complication, it is possible that equal heating protocols will cause varying results in the extent of thermal damage. But it is important to note that all three tests were within the permissable error bands.

- The baseline case shown in Figs 6.16 and 6.17 is the only case of the three that does not include the convective effects of H_i and consequently shows the least agreement with the optimal metrics, particularly for H_{ab}. The poor agreement underscores the importance of including convective or perfusion effects in the tissue models.

- The allowable ranges of H_{ab} and D_{ab} are quite large because the model is very sensitive to the activation energy, see Fig. 6.13. Knowing this term with a high degree of accuracy is extremely important. Since the activation energy has been shown to be temperature dependent, more work may be needed to incorporate this temperature dependence in the damage models (Wright (2003)). The thermal dose model takes into account that reactions occur faster at higher temperatures via the increase of R with temperature. Recall that in the thermal dose model, R = 0.25 for temperatures below 43^oC and R = 0.5 for temperatures above 43^oC. See Fig. 3.10 for more details.

Chapter 8

Recommendations

Hyperthermia treatment planning, and the broader field of thermal therapy planning is just now beginning to be explored. As with any young field, the research topics in this field are nearly boundless, but with regards to this specific body of work some recommendations for improvements are now presented.

First, improved models should be tested with the minimization algorithm. Since the algorithm shows promise with some added complexity, there is no reason not to explore just how realistic the models can be. Three-dimensional, patient-specific models made from imaging equipment should be considered as an alternative to simplified two dimensional models.

The optimization algorithm can be improved by adding penalty functions to "corral" potential solutions into a more clinically applicable subset of functions. Parametrization based on cubic spline functions showed limited ability to do perform this task and alternate methods should be implemented. These improvements also maybe as simple as choosing better so called "knots" where the basis functions are "tied" together. Also, limiting the tendency of the algorithm to over compensate when initial guesses of the source function provide too little energy input should be addressed. The user of the algorithm should not be limited too much in making the initial guess of what the optimal function will look like: if the user must know the final answer with too high of a degree of accuracy before starting, the algorithm will be of little practical use.

The albumen phantom had strong points: it was inexpensive, easily accessible and clear demarkation could be made between the damaged and undamaged regions. But be-

cause of the added uncertainty of the empirically determined internal convection it is recommended that more research is done to find better phantoms. Over the course of this research powder-based albumen and natural albumen were used. Both exhibited buoyancy driven flow although natural albumen was much more viscous. The ideal phantom would be homogenous, have a controllable, calibrated internal convection (perfusion) and exhibit a visible similar thermal denaturation based on a physiologically common Arrhenius relationship.

Bibliography

Ahmed, M., Goldberg, S.N., (2002), "Thermal ablation therapy for hepatocellular carcinoma," *Journal of Vasc. Interv. Radiology*, Vol. 13, pp. S231–S243.

Alifanov O.M., (1994), *Inverse Heat Transfer Problems*, Springer-Verlag, Berlin.

Beck, J.V., Blackwell, B., and Haji-Sheikh, A., (1996), "Comparison of some inverse heat conduction methods using experimental data," *Int. J. Heat Transfer*, Vol. 39, No. 17, pp. 3649–3657.

Beck, R., Deuflhard,P., Hege, H, Seebab, M., and Stalling, D., (1997), *Visualization and Mathematics*, Chap. Numerical algorithms and visualization in medical treatment planning, Springer, pp. 303–328.

Carslaw H.S., Jaeger J.C., (1959), *Conduction of Heat in Solids*, Oxford University Press.

Cebral, J.R., Soto, O.S., Lutz, R.J., and Wood, B.J., (2003), "Effects of blood flow on radiofrequency ablation of tumors: Finite elements and in vitro models," in Proceedings of the *ASME,IMECE*.

Chato, J.C., (1985), *Heat Transfer in Medecine and Biology; Analysis and Appliations*, Vol. 1, Chap. Estimation of Tissue Blood Flow, Plenum Press, pp. 167–192.

Dewey W.C., (1994), "Arrhenius relationships from the molecule and cell to the clinic," *Int. J. Hyperthermia*, Vol. 10, No. 4, pp. 457–483.

Diller, K.R., (1992), "Modeling of bioheat transfer processes at high and low temperatures," *Advances in Heat Transfer*, Vol. 22, pp. 157–357.

Dorr, L.N., Hynynen, K., (1992), "The Effects of Tissue Heterogeneities and Large Blood Vessels on the Thermal Exposure Induced by Short High-Power Ultrasound Pulses," *Int. J. Hyperthermia*, Vol. 8, No. 1, pp. 45–59.

Egg, (2004), *http://oregonstate.edu/instruct/nfm236/egg/index.cfm*, Oregon State University, June, 2004.

Gellermann, J. et al., (2000), "Clinical Evaluation and Verification of the Hyperthermia Treatment Planning System HYPERPLAN," *Int. J. Radiaiton Oncology Biol. Phys.*, Vol. 47, No. 4, pp. 1145–1156.

Heldman, D.R., (1977), *Food Process Engineering*, AVI Publishing Co., Inc.

Henriques, F.C., Moritz, A.R., (1947), "Studies of thermal injury I. The conduction of heat to and through skin and the temperatures attained therein. A theoretical and experimental investigation," *Am. J. Pathol.*, Vol. 23, pp. 531–549.

Hynynen, K., Chung, A., Fjield, T., Buchanan, M., Duam, D., Colucci, V., Lopath, P., and Jolesz, F., (1996), "Feasibility of Using Ultrasound Phased Arrays for MRI Monitored Noninvasive Surgery," *IEEE trans. on Ultrason. Ferroelect., and Freq. Contr.*, Vol. 43, No. 6, pp. 1043–1052.

Incropera, F.P., DeWitt, D.P., (1996), *Fundamentals of Heat and Mass Transfer, 4th Ed.*, J. Wiley and Sons.

James, B.J., Sullivan, D.M., (1992), "Creation of Three-Dimensional Patient Models for Hyperthermia Treatment Planning," *IEEE Transactions on Biomedical Engineering*, Vol. 39, No. 3.

Lagendijk, J.J.W., (2000), "Hyperthermia treamtment planning," *Phys. Med. Biol.*, Vol. 45, pp. R61–R76.

Lalonde, R.J., Hunt, J.W., (1995), "Optimizing Ultrasound Focus Distributions for Hyperthermia," *IEEE Transactions on Biomedical Engineering*, Vol. 42, No. 10.

Loulou T. , Scott E.P., (2000), "2-D Thermal Dose Optimization in High Intensity Focused Ultrasound treatments using the adjoint method," *ASME-IMECE'00*, Vol. HTD-386/BED-47, pp. 67–71.

Loulou T., Scott E.P., (2002), "Thermal dose optimization in hyperthermia treatments by using the Conjugate Gradient Method," *Numerical Heat Transfer, Part A*, Vol. 42, No. 7, pp. 661–683.

Madden, M.A., Scott, E.P., (2003), "An Agar and Saline Phantom for Perfused Tissue," in Proceedings of the *ASME,IMECE*, No. Paper No. IMECE2003-41415.

Moffat, R.J., (1988), "Describing the uncertainties in experimental results," *Experimental Thermal and Fluid Sciences*, Vol. 1, pp. 3–17.

Moritz, A.R., Henriques, F.C., (1947), "Studies of thermal injury II. The relative importance of time and surface temperature in the causation of cutaneous burns," *Am. J. Pathol.*, Vol. 23, pp. 695–720.

Patankar S.V., (1980), *Numerical Heat Transfer and Fluid Flow*, McGraw Hill, New York.

Pearce, J.A., Thomsen, S., (1992), "Kinetic models of tissue fusion processes," *Proc. of Laser Surgery (SPIE): Advanced Characterization, Therapeutics, and Systems III*, Vol. 1643, pp. 251–260.

Pennes, H.H., (1948), "Analysis of tissue and arterial blood temperatures in the resting human forearm," *J. Appl. Physiol.*, Vol. 1, pp. 93–122.

Pfefer, T.J., Chan, K.F., Hammer, D.X., and Welch, A.J., (2000), "Dynamics of pulsed holmium: YAG laser photocoagulation of albumen," *Phys. Med. Biol.*, Vol. 45, pp. 1099–1114.

Sapareto S.A., Dewey W.C., (1984), "Thermal dose determination in cancer therapy," *Int. J. Radiation Oncology Biol. Phys.*, Vol. 10, pp. 787–800.

Scott, E.P, Robinson, P., and Diller, T., (1998), "Development of Methodologies for the Estimation of Blood Perfusion using a Minimially Invasive Thermal Probe," *Measurement Science and Technology (Invited Paper)*, Vol. 9, Special Edition, pp. 888–897.

Wright, N.T., (2003), "On a relationship between the arrhenius parameters from thermal damage studies," *Journal of Biomech. Eng.*, Vol. 125, pp. 300–303.

Wright, N.T., Humphrey, J.D., (2002), "Denaturation of collagen via heating: An irreversible rate process," *Annual Reveiews in Biomedical Eng.*, Vol. 4, pp. 109–128.

Xu, X.L., Chen, M.M, et. al., (1992), "Theoretical Analysis of the Large Blood Vessel Influence on the Local Tissue Temperature Decay After Pulse Heating," *Journal of Biomechanical Engineering*, Vol. 115, pp. 165–178.

Yamamoto, T., Juneja, L.R., Hatta, H. and Kim, M., (1997), *Hen Eggs: Their Basic and Applied Science*, CRC Press, Boca Raton, FL.

Yang, Y., Welch, A.J. and Rylander, III, H.G., (1991), "Rate process parameters of albumen," *Lasers in Surgery and Medicine.*, Vol. 11, pp. 188–190.

Appendix A

Finite Difference Boundary Equations

In the text, the governing equations for an interior node of the two dimensional finite difference scheme was presented. In this appendix the remaining eight finite difference equations will be listed. For a full description of the development of these equations, see the chapter 3 in the text. Recall that the terms $aO_{i,j}$, and the directional terms are defined in chapter 3. The equations here are changes to those presented in chapter 3, so if something's not listed use the equations in the text.

Nodes go from $i = 1 \rightarrow ii$ in the x direction and $j = 1 \rightarrow jj$ in the y direction. If the formula is not at an end point the compact notation $i = [2 \leq i \leq i-1]$. The south-west corner is considered the origin, $(x = 0,\, y = 0) = (i = 1,\, j = 1)$. The horizontal sweep is first in the sequence, then the vertical.

A.1 Western edge: Horizontal sweep

$$ahor_{1,j}\, T_{1,j}^{p+\frac{1}{2}} - aE_{1,j} T_{(2,j)}^{p+\frac{1}{2}} = bhor_{1,j} \tag{A.1}$$

$$ahor_{1,j} = 2\, aO_{1,j} + aE_{1,j} + (S_p\, \Delta V)_{1,j} + A_{W,(1,j)}\, h_{x=0} \tag{A.2}$$

$$bhor = \underbrace{(2\, aO_{1,j} - aN_{1,j} - aS_{1,j})}_{chor}\, T_{1,j}^{p} \tag{A.3}$$

$$+ \quad \underbrace{S^p_{c\,(1,j)}\,\Delta V_{1,j} + aN_{(1,j)}T^p_{(1,j+1)} + aS_{1,j}T^p_{1,j-1} + A_{W,(1,j)}(q" + h\,T_\infty)_{(x=0)}}_{\hat{b}}$$

A.2 Western edge: Vertical sweep

$$-aN_{1,j}T^{p+1}_{1,j+1} + avert_{1,j}\,T^{p+1}_{1,j} - aS_{1,j}T^{p+1}_{1,j-1} = bvert_{1,j} \tag{A.4}$$

$$\begin{aligned} avert_{1,j} &= 2\,aO_{1,j} + aN_{1,j} + aS_{1,j} + (S_p\,\Delta V)_{1,j} && \text{(A.5)} \\ bvert_{1,j} &= \underbrace{(2\,aO_{1,j} - aE_{1,j} - A_{W,(1,j)}\,h_{x=0})}_{cvert}\,T^{p+\frac{1}{2}}_{1,j} && \text{(A.6)} \\ &+ \underbrace{S^p_{c\,(1,j)}\,\Delta V_{1,j} + aE_{1,j}T^{p+\frac{1}{2}}_{2,j} + A_{W,(1,j)}(q" + h\,T_\infty)_{(x=0)}}_{\hat{b}} \end{aligned}$$

A.3 Northern edge: Horizontal Sweep

$$-aE_{i,jj}T^{p+\frac{1}{2}}_{i+1,jj} + ahor_{i,jj}\,T^{p+\frac{1}{2}}_{i,jj} - aW_{i,jj}T^{p+\frac{1}{2}}_{i-1,jj} = bhor_{i,jj} \tag{A.7}$$

$$\begin{aligned} ahor_{i,jj} &= 2\,aO_{i,jj} + aW_{i,jj} + aE_{i,jj} + (S_p\,\Delta V)_{i,jj} && \text{(A.8)} \\ bhor_{i,jj} &= \underbrace{(2\,aO_{i,jj} - aS_{i,jj} - A_{N,(i,jj)}\,h_{y=H})}_{chor}\,T^p_{i,jj} && \text{(A.9)} \\ &+ \underbrace{S^p_{c\,(i,jj)}\,\Delta V_{i,jj} + aS_{i,jj}T^p_{i,jj} + A_{N,(i,jj)}(q" + h\,T_\infty)_{(y=H)}}_{\hat{b}} \end{aligned}$$

A.4 Northern edge: Vertical sweep

$$avert_{i,jj}\,T^{p+1}_{1,j} - aS_{i,jj}T^{p+1}_{i,j-1} = bvert_{i,jj} \tag{A.10}$$

$$\begin{aligned} avert_{i,jj} &= 2\,aO_{1,j} + aS_{i,jj} + (S_p\,\Delta V)_{i,jj} + A_{N,(i,jj)}\,h_{x=0} && \text{(A.11)} \\ bvert_{i,jj} &= \underbrace{(2\,aO_{i,jj} - aE_{(i,jj)} - aW_{i,jj})}_{cvert}\,T^{p+\frac{1}{2}}_{i,jj} && \text{(A.12)} \end{aligned}$$

$$+ \quad \underbrace{S^p_{c\,(i,jj)}\,\Delta V_{1,j} + aE_{i,jj}T^{p+\frac{1}{2}}_{i+1,jj} + aW_{i,jj}T^{p+\frac{1}{2}}_{(i-1,jj)} + A_{N,(i,jj)}(q" + h\,T_\infty)_{(y=H)}}_{\hat{b}}$$

A.5 Eastern edge: Horizontal sweep

$$ahor_{ii,j}\,T^{p+\frac{1}{2}}_{ii,j} - aW_{ii,j}T^{p+\frac{1}{2}}_{ii-1,j} = bhor - ii,j \tag{A.13}$$

$$\begin{aligned}
ahor_{ii,j} &= 2\,aO_{ii,j} + aW_{ii,j} + (S_p\,\Delta V)_{ii,j} + A_{E,(ii,j)}\,h_{x=L} && \text{(A.14)}\\
bhor &= \underbrace{(2\,aO_{ii,j} - aN_{(ii,j)} - aS_{ii,j})}_{chor}\,T^p_{ii,j} && \text{(A.15)}\\
&+ \underbrace{S^p_{c\,(ii,j)}\,\Delta V_{ii,j} + aN_{ii,j}T^p_{ii,j+1} + aS_{ii,j}T^p_{ii,j-1} + A_{E,(ii,j)}(q" + h\,T_\infty)_{(x=L)}}_{\hat{b}}
\end{aligned}$$

A.6 Eastern edge: Vertical sweep

$$-aN_{ii,j}T^{p+1}_{ii,j+1} + avert_{ii,j}\,T^{p+1}_{ii,j} - aS_{ii,j}T^{p+1}_{ii,j-1} = bvert_{ii,j} \tag{A.16}$$

$$\begin{aligned}
avert_{ii,j} &= 2\,aO_{ii,j} + aN_{ii,j} + aS_{ii,j} + (S_p\,\Delta V)_{ii,j} && \text{(A.17)}\\
bvert_{ii,j} &= \underbrace{(2\,aO_{ii,j} - aW_{ii,j} - A_{E,(ii,j)}\,h_{x=L})}_{cvert}\,T^{p+\frac{1}{2}}_{ii,j} && \text{(A.18)}\\
&+ \underbrace{S^p_{c\,(ii,j)}\,\Delta V_{ii,j} + aW_{ii,j}T^{p+\frac{1}{2}}_{ii-1,j} + A_{E,(ii,j)}(q" + h\,T_\infty)_{(x=L)}}_{\hat{b}}
\end{aligned}$$

A.7 Southern edge: Horizontal sweep

$$-aE_{i,1}T^{p+\frac{1}{2}}_{i+1,1} + ahor_{i,1}\,T^{p+\frac{1}{2}}_{i,1} - aW_{i,1}T^{p+\frac{1}{2}}_{i-1,1} = bhor_{i,1} \tag{A.19}$$

$$\begin{aligned}
ahor_{i,1} &= 2\,aO_{i,1} + aW_{i,1} + aE_{i,1} + (S_p\,\Delta V)_{i,1} && \text{(A.20)}\\
bhor_{i,1} &= \underbrace{(2\,aO_{i,1} - aN_{i,1} - A_{S,(i,1)}\,h_{y=0})}_{chor}\,T^p_{i,1} && \text{(A.21)}
\end{aligned}$$

$$+ \quad \underbrace{S^p_{c\,(i,1)}\,\Delta V_{i,1} + aN_{i,1}T^p_{(i,1)} + A_{S,(i,1)}(q"^p + h\,T_\infty)_{(y=0)}}_{\hat{b}}$$

A.8 Southern edge: Vertical sweep

$$avert_{i,1}\,T^{p+1}_{i,1} - aN_{i,1}T^{p+1}_{(i,j+1)} = bvert_{i,1} \tag{A.22}$$

$$avert_{i,1} = 2\,aO_{i,1} + aN_{i,1} + (S_p\,\Delta V)_{i,1} + A_{S,(i,1)}\,h_{x=0} \tag{A.23}$$

$$bvert_{i,1} = \underbrace{(2\,aO_{i,1} - aE_{i,1} - aW_{i,1})}_{cvert}\,T^{p+\frac{1}{2}}_{i,1} \tag{A.24}$$

$$+ \quad \underbrace{S^p_{c\,(i,1)}\,\Delta V_{i,1} + aE_{i,1}T^{p+\frac{1}{2}}_{i+1,1} + aW_{i,1}T^{p+\frac{1}{2}}_{i-1,1} + A_{S,(i,1)}(q"^p + h\,T_\infty)_{(y=0)}}_{\hat{b}}$$

A.9 Note on corner edges

Corner edges are not shown, but for corners where both conjoining edges were type 2 boundary conditions, the same pattern follows. See Appendix D containing the code, particularly DirectProblemFD.m for more details. For corners where one node was constant temperature, that node was set to the constant temperature. Remember, constant temperature boundary conditions set $ahor = avert = 1$ and $bhor = bvert = T_{boundary}$. All off-diagonal terms are set to zero.

Appendix B

Control Problem Formulation with Volumetric Heating

The optimal control problem can be formulated with a variety of heating sources. In the body of the report, results were reported for both boundary heating and volumetric heating. But in chapter 4 the derivation was only given for boundary heating. Here one will find the alternate equations to use in order to apply volumetric heating.

B.1 The direct problem

Here the direct problem is shown for volumetric heating only. Adiabatic symmetry is applied to the central and radial axes. Volumetric heating, $Q(t)$ is applied to any portion over the domain.

$$C(T)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left[k(T)\frac{\partial T}{\partial x}\right] + \frac{\partial}{\partial y}\left[k(T)\frac{\partial T}{dy}\right]\ldots \quad \text{(B.1)}$$
$$-W(T)\,(T - T_\infty) + Q(x, y, t)$$

$$\frac{\partial T(x,t)}{\partial y} = 0 \qquad x\,,\, y = 0\,,\, t < t_o \quad \text{(B.2)}$$

$$\Delta\, T(x,t) = T_\infty \qquad x\,,\, y = H\,,\, t < t_o \quad \text{(B.3)}$$

$$\frac{\partial T(y,t)}{\partial x} = 0 \qquad x = 0\,,\, y\,,\, t < t_o \quad \text{(B.4)}$$

$$\Delta\, T(y,t) = T_\infty \qquad x = L\,,\, y\,,\, t < t_o \quad \text{(B.5)}$$

$$\Delta\, T(x,y) = T_\infty \qquad x\,,\, y\,,\, t = t_o \quad \text{(B.6)}$$

B.2 The variation problem

The variation problem is formulated the way that it was formulated in chapter 4. The only difference that the perturbed heating function or descent direction is now given as $\Delta Q(t)$.

$$\begin{aligned}
\frac{\partial[C(T)\,\Delta\,T]}{\partial t} &= \frac{\partial^2\,[k(T)\Delta\,T]}{\partial x^2} + \frac{\partial^2\,[k(T)\Delta\,T]}{\partial y^2}\ldots \\
&\quad -W(T)\,\Delta\,T + \Delta\,Q(x,y,t) \qquad \text{(B.7)}\\
\frac{\partial\,[\Delta T(x,t)]}{\partial y} &= 0 \qquad x\,,\,y=0\,,\,t>t_o \qquad \text{(B.8)}\\
\Delta\,T(x,t) &= 0 \qquad x\,,\,y=H\,,\,t>t_o \qquad \text{(B.9)}\\
\frac{\partial\Delta\,T(y,t)}{\partial x} &= 0 \qquad x=0\,,\,y\,,\,t>t_o \qquad \text{(B.10)}\\
\Delta\,T(y,t) &= 0 \qquad x=L\,,\,y\,,\,t>t_o \qquad \text{(B.11)}\\
\Delta\,T(x,y) &= 0 \qquad x\,,\,y\,,\,t=t_o \qquad \text{(B.12)}
\end{aligned}$$

B.3 The Adjoint Problem and the Functional Gradient

The adjoint problem is solved the same way as with the boundary heating. The only difference between the boundary heating and volumetric heating is how the solution to the adjoint problem is integrated. First we consider that the volumetric heating is applied over the axial range $0 \le y \le a_h$ and the radial range $0 \le x \le r_h$.

The remaining term is known as the first variation of the functional $\mathcal{J}(q)$ in the theory of calculus of variations. The expression shown below in Eqn. (B.15) is known as the scalar product in the working space, which is Hilbert space, or the space of squared integrable functions.

$$\begin{aligned}
\Delta\mathcal{L} &= \int_{t_o}^{t_f}\int_0^{r_h}\int_0^{a_h} \Psi_1(x,y,t)\cdot\Delta Q(t)\,dx\,dy\,dt \qquad \text{(B.13)}\\
&= \int_{t_o}^{t_f} \mathcal{J}'(Q(t))\cdot\Delta Q(t)\,dx\,dt \qquad \text{(B.14)}\\
&= \langle\mathcal{J}'(Q(t)),\Delta Q(t)\rangle \qquad \text{(B.15)}
\end{aligned}$$

By definition of the scalar product the gradient of the objective function can be written as shown below. This ends the development of the control problem for volumetric heating.

$$\mathcal{J}'(Q(t)) \quad = \quad -\int_0^{r_h}\int_0^{a_h} \Psi_1(x,y,t)\,dx\,dy \tag{B.16}$$

Comparing the results of the boundary and volumetric cases, it becomes evident that the gradient of the objective function is the result of the adjoint problem integrated over the volume where volumetric heating is applied. This also applies to problems of one, two or three dimensions.

Appendix C

Users manual, Damage Dose Optimizer 3.0

The program Damage Dose Optimizer 3.0 is the main product of this research. It is written in MATLAB, and this section contains information how to use it, what it can do and how to get data out of it.

C.1 Getting Started

First thing to do is make sure all the files are in the same directory and point MATLAB to that directory. Open UserInput.m, this is the user interface for most cases. Units simply need to be consistent. The damage model requires Kelvin temperature units and the dose model requires Celsius. Things are self explanatory for the most part, with more complicated inputs explained here.

C.1.1 Source selection

```
% --- SELECT SOURCE FUNCTION
% *******************************************************
%     SourceType = boundary ->  Boundary heating on y = 0
%                               specify rheat
%     SourceType = volume   ->  Volumetric heating over a
%     portion of the boundary starting at y = 0, x = 0
%                               specify rheat - > radial
%                                       aheat - > axial
% *******************************************************
SourceType = 'vol'; rheat = Lx/2; aheat = Ly/2;
```

For the SourceType variable, choosing 'vol' will give volumetric heat and 'boundary' will give boundary heat. By default, the volumetric heating must be applied to a portion of the boundary starting in the southwest corner, with 'rheat' and 'aheat' defining radial and axial extent of the volumetric heating zone. Boundary heat must be applied to the southern edge, starting at x=0.

C.1.2 Nonlinear properties

```
\verb"% --- THERMAL PROPERTIES
% ****************************************************
%     Temperature dependent properties can be modeled.
%
%     This program is not developed to have regional
%     differences in thermal properties, but it can be
%     modified to do so.
%
%     Ttable is a vector of temperatures spanning the range
%     over which properties vary.  If Ttable is 0, then
%     properties are constant.  Ttable must be C, not K.
%
%     Variables_table = Values of thermal properties that
%     correspond to Ttable.  Only the first value listed is
%     used in the case of constant properties.
%
%     KtableX = thermal conductivity
%     Cptable = specific heat
%     Rhoval = value of density, constant
% ****************************************************
Ttable = [37  60 100];
 KtableX = [.21 .37 .16];
 Cptable = [4000];
Rhoval = [1040];
```

Thermal properties can be constant or vary with temperature, if they vary with temperature, Ttable is the temperature coordinates and KtableX and Cptable are the corresponding values. These vectors must be the same length. In the above example conductivity changes with temperature and specific heat does not. See nlconductivity.m and nlspecficheat.m to modify what kinds of functions are used to create the functions.

C.1.3 Choosing the source function

```
% --- 2. CHOOSE SOURCE FUNCTION
```

```
% ************************************************************
%     FluxString determines which function of the ones below will
%     be the heat flux in time
%
%     String Options for each face.
%     'off'    = this option is disabled
%     'constant' = constant heat flux with magnitude c1.
%     'squpls'   = square pulses heat source with magnitude c1.  Pulse
%                  persists for c2 seconds.  c3 is
%                  the time between pulses and c4 is the number of
%                  pulses
%     'square' = Single square pulse of magnitude c1, turns on at time
%                  c2 and goes off at c3.
%     'sinwav' = sinusiod heat flux with amplitude c1 and c2 cycles.
%     'triang' = single triangular heat flux pulse with hieght c1
%                  occuring at midpoint of time domain and with start
%                  time c2 and end time c3.  c4 is a base addtion added
%                  at every time step.
%     'tripls' = triangular heat flux pulses with hieght c1 occuring at
%                  midpoint of pulse length c2.  c3 is the time between
%                  pulses and c4 is the number of pulses.
%     'linear' = linearly increasing heat flux.  Starts at t = c1
%                   increases up to c2 over the time domain
%
%     For options with no c3 or c4 enter 0
%
%     For fully insulated surface use 'consta' and magnitude 0
% ************************************************************
% FluxString = ['tripls'];
% cf = [1100 6*dt 19*dt 4];
% FluxString = ['squpls'];
% cf = [3000 400 50 4];
% FluxString = ['triang'];
% cf = [200000 0 1800 0];
FluxString = ['consta'];
cf = [900000 0 2700 100000];"
```

The chose source function above is an important part. Read the instructions for understanding what the listed FluxString variable is doing. Fluxstring is a bad name for the variable since it can be flux or volumetric heating. The first gives 4 triangular spikes of magnitude 1100 W/m2 or W/m3 depending on what the user has selected. It occurs over the course of 6 time steps. There are 19 steps in between the end of the previous pulse and the beginning of the new one.

Table C.1 Summary of output files from Damage Dose Optimizer 3.0

File Name	Description
FieldStats.out	Convergence stats: E_{RMS} = Column 2, $\mathcal{J}$ = Column 5
Control.out	[t, $\Gamma(t)_{final}$, $\Gamma(t)_{optimal}$]
FieldEstimate.out	Final field estimate
FieldOptimal.out	Optimal field estimate
ControlEst.out	Evolution of the control function

Noise adding

Ignore this. It was never programmed into the optimizer since we are not estimating anything from acquired data.

C.2 Running the program

Once you have input the desired data in UserInput.m, open ThermaDamageMain.m and run the program.

C.2.1 Output data files

The code outputs several files which are valuable resource for plotting analyzing the performance of the run. See the table below for a summary.

C.2.2 Run overview

The very first entry in userinput.m lets you chose the name of an output file that the program prints a run overview to. You can open this file with a text editor. A sample copy is shown below.

```
 ***Thermal Field Optimization
-CONGUGATE GRADIENT W/ADJOINT***

TEST PERFORMED ON (year/month/day, hour:min:sec):
2004/08/19, 17:56:38

FD PARAMETERS
Number of Control Volumes,x:.................. 15.
Number of Control Volumes,y:.................. 15.
Number of Control Volumes of surface heater:.. 7.
Discretized heater length:.................... 7.
Error between discretized & actual heater:.... 0.00464286.
Number of Time Steps:......................... 91.
Length of interior FD Cell, x:................ 0.000714286.
Length of interior FD Cell, y:................ 0.000714286.
Length of Time steps in Seconds:.............. 20.
Fourier number:............................... 14504.

MODEL PARAMETERS
Length of domain in x (radial):............... 1.000000e-002
Length of domain in y (vertical):............. 1.000000e-002
Nominal Length / Radius of surface heater:.... 5.000000e-003
Heat flux initial guess:...................... triang
Constants for heat flux guess:................ 1e+006
Constants for heat flux guess:................ 0
Constants for heat flux guess:................ 1800
Constants for heat flux guess:................ 100000
Initial temperature, K:....................... 310.
Body reference temperature, K:................ 310.
Blood perfusion parameter:.................... 0.
Nonlinear k modeling on:...................... 0.
Nonlinear Cp tissue modeling on:.............. 0.
Nonlinear temperature range:.................. 0.
Thermal conductivity over temperature range:.. 0.37.
Specific heat over temperature range:......... 4000.
Nonlinear convergence criteria:............... 0.001.
Constant tissue density:...................... 1040.

Boundary condtion type, y=0:...................... 2.
Film coefficient y=0:......................... 0.
Convective ambient temp y=0:.................. 0.

Boundary condtion type, y=H:...................... 1.
Const temp y=H, K:............................ 310.
```

```
Boundary condtion type, x=0:..................... 2.
Film coefficient x=0:........................... 8.5.
Convective ambient temp x=0:.................... 297.
Additional Heat flux incident on x=0:......... 0.

Boundary condtion type, x=L:..................... 1.
Const temp x=L, K:............................. 310.

OPTIMAL CONTROL PROBLEM PARAMETERS
Type of field:................................ damage
Method used to create field:.................. generate
Control function type used to create Field:... consta
Optimal protocol constants:................... 500000
Optimal protocol constants:................... 0
Optimal protocol constants:................... 2700
Optimal protocol constants:................... 100000
Number of parameters to construct function:... 0
Stability parameter:.......................... 0
Convergence criteria for no error:............ 1e-007.
Max allowed iterations to minimize objective:. 300.

ARRHENIUS DAMAGE MODEL PARAMETERS
Activation Energy:............................ 628000.
Frequency Factor:............................. 3.1e+098.
Universal Gas Constant:....................... 8.31.

OPTIMIZATION RESULTS
ELAPSED TIME TO RESOLVE, sec:................. 388.097.
Number of completed iterations:............... 113.
Time per iteration:........................... 3.43449.
Objective function value at loop end:......... 2.43555e-008.

FINAL RMS / MAX REAL HEAT FLUX:............... 0.0117831.
```

Appendix D

Damage Dose Optimizer 3.0 Code

In this appendix the MATLAB code **Damage Dose Optimizer 3.0** is shown. The main file is, `thermaldamagemain.m`, if one wishes to understand what is going on, it is strongly recommended that he/she starts there. Each function m-file that is not a built in MATLAB function is found in this appendix. A flow chart with file names is shown in Fig. D.1 to guide the reader. The files shown normally have an explanation at the beginning of each which is commented.

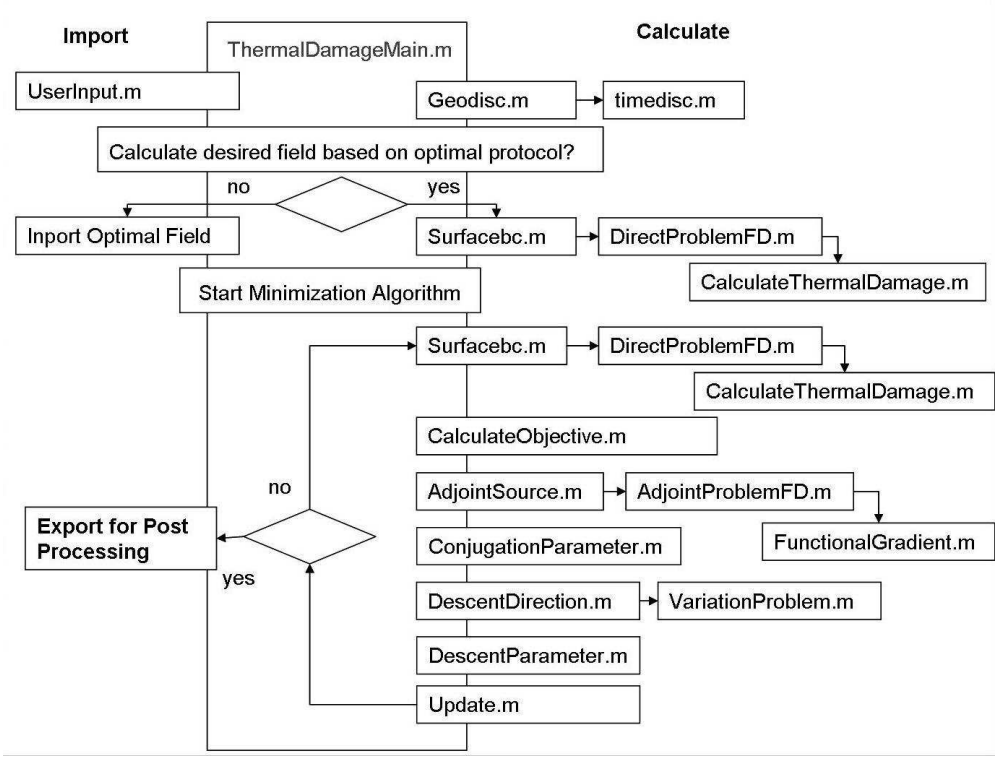

Fig. D.1 Flow chart showing call sequence of Damage Dose Optimizer 3.0.

```
 function [x,ControlEst,ControlOpt] = ThermalDamageMain;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   Objective:  Given a desired damage/dose field find a surface heat
%   flux function in time that causes the desired damage.
%
%   Model:  2 D Cylindrical finite difference, temperature
%   dependent thermal properties optional.
%
%   The conjugate gradient method with supporting problems:
%
%   Adjoint Problem: Calculated gradient of functional
%
%   Variation Problem: Calculates temperature sensitivity to
%   small changes in the control parameter
%
%   Control Parameter:  Heat flux incident on south face of domain
%
%
%   Sources: Ozisik, Orlande, "Inverse Heat Transfer", 2000
%          Loulou, Tahar, Ecole Des Mines d'Albi-Carmaux
%
%   Author:  Scott Gayzik, March '04
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;

% --- Global variables used in multiple functions
global D time SourceType
global x y dx dy
global NPARAM

% --- Input user-specified parameters
[L,BcParamDP,BcTypeDP,OptParam,ImportField,FieldMethod,...
        FieldVariable,FluxString,FluxGuess,C,outputfile] = UserInput;

% --- Unzip parameters
nCellsx = D(1); nCellsy = D(2);
dt = time(1); pmax = time(2);
epsilon = OptParam(1); timeout = OptParam(2);
noise = OptParam(3); amplitude = OptParam(4); arch = OptParam(5);
cf = C(1,:); cd = C(2,:); guess = C(3,:);

% --- discretize the domain
[heater] = geodisc(L,D);

% --- discretize the time domain
t = timedisc;

%*******************************************************
%        GENERATE DESIRED THERMAL DAMAGE FIELD
%   OR THERMAL DOSE VIA AN OPTIMAL CONTROL FUNCTION
%*******************************************************

if FieldMethod(1:6) == 'genera'

    % --- Surface heat flux for optimal damage
    ControlOpt = surfacebc(FluxString(1:length(FluxString)),cf);

    % --- Solve for temperature distribution
    [OptimalT,kxy,cpxy,Sp] = DirectProblemFD(BcTypeDP, BcParamDP, ControlOpt, heater);

    % --- Compute design variable
    if FieldVariable(1:4) == 'dose'
        FieldDesired = CalculateThermalDose(OptimalT);
    else
        FieldDesired = CalculateDamageField(OptimalT);
    end
else
%*******************************************************
%              IMPORT DESIRED DAMAGE FIELD
%                OR THERMAL DOSE DIRECTLY
%*******************************************************
%     load(ImportField);
    FieldDesired = load(ImportField);
    ControlOpt = surfacebc('consta',[0 0 0 0]);
end

% --- Save Desired Field
save Field.out FieldDesired -ascii -double;

% ************************************************
%          *** BEGIN MINIMIZATION ALGORITHM ***
%
%     Minimize objective function with conjugate
%     gradient method.
%
%     Direct Problem -> evaluates current guess for
%     control function
%     Adjoint Problem -> solution is the gradient of
%     the objective function
%     Variation Problem -> solution gives direction
%     descent
%
%     Design variable can be thermal dose or thermal
%     damage coef
%
%*************************************************

tic
% --- Set boundary conditons for Adjoint problem
BcParamAP =  [0 0 0 0 0 0 0 0 0;
              0 0 0 0 0 0 0 0 0];
```

```
% --- Set boundary parameters for Variation problem
BcParamVP =  [0 0 BcParamDP(1,3) BcParamDP(1,4) 0 0 0 0 0;
              0 0 BcParamDP(2,3) BcParamDP(2,4) 0 0 0 0 0];

% --- Initial guess of control function
ControlEst(:,1) = surfacebc(FluxGuess(1:length(FluxGuess)),guess);

% --- Initialize loop variables
FieldEstimate = zeros(nCellsx,nCellsy,timeout);
if NPARAM > 0
    gradient(:,1) = zeros(NPARAM,1);
    descent = zeros(NPARAM,1);
else
    gradient(:,1) = zeros(pmax,1);
    descent = zeros(pmax,1);
end
gamma(1) = 0;

finishloop = 0;
convcheck = 0;

k=2;
archive = 1;
while k < timeout & finishloop == 0

    % --- Solve direct problem
    [Tq,kxy,Cpxy,Sp,NlIter] = DirectProblemFD(BcTypeDP, BcParamDP, ControlEst(:,k-1),heater);

    % --- Compute Field Variable (Damage or Dose)
    if FieldVariable(1:4) == 'dose'
        [FieldEstimate,R] = CalculateThermalDose(Tq);
    else
        FieldEstimate = CalculateDamageField(Tq);
    end

    % --- Evaluate the objective function
    [J(k),FieldDifference,FieldRMSError(k)] = ...
        CalculateObjective(FieldEstimate,FieldDesired,ControlEst(:,k-1),t);

    FieldRMSError(k)
    J(k)

    % --- Solve adjoint problem (Damage or Dose)
    if FieldVariable(1:4) == 'dose'
        AdjointSc = AdjointSourceDose(Tq,FieldDifference,R);
    else
        AdjointSc = AdjointSourceDamage(Tq,FieldDifference);
    end
    [P] = AdjointProblemFD(BcTypeDP,BcParamAP,AdjointSc,kxy,Cpxy,Sp);

    % --- Compute gradient vector
    if SourceType(1:3) == 'bou'
        [gradient(:,2), tparam] = FunctionalGradient(P(nCellsy,1:heater(1,1),:),ControlEst(:,k-1));
    elseif SourceType(1:3) == 'vol'
        [gradient(:,2), tparam] = ...
            FunctionalGradient(P((nCellsy-heater(2,1)+1):nCellsy,1:heater(1,1),:),ControlEst(:,k-1));
    end

    % --- Solve for beta, the conjugation parameter
    [beta,gradient] = ConjugationParameter(gradient(:,1:2),k);

    % --- Compute the decent direction
    descent = DescentDirection(beta,gradient(:,1),descent);

    % --- Compute variation problem
    [delT,DelControl] = VariationProblemFD(BcTypeDP,BcParamVP,descent,heater,kxy,Cpxy,Sp,tparam);

    % --- Compute the search step size
    if FieldVariable(1:4) == 'dose'
        gamma = DescentParameterDose(Tq,delT,FieldDifference,R,ControlEst(:,k-1),DelControl);
    else
        gamma = DescentParameterDamage(Tq,delT,FieldDifference,ControlEst(:,k-1),DelControl);
    end

    % --- Update the estimated function
    ControlEst(:,k) = update(ControlEst(:,k-1),gamma,DelControl);

    % --- Check convergence
    if noise == 0
        ControlCheck(:,k) = abs((ControlEst(:,k) - ControlEst(:,k-1))./(ControlEst(:,k)+1E-20));
        stopcheck = max(ControlCheck(:,k));
        if stopcheck <= epsilon;
            display('convergence criterion for no error met')
            finishloop = 1;
        end
    end

    % --- Archive Data
    if k == 2
        FieldArchive(:,:,1) = FieldEstimate;
        archive = archive + 1;
    elseif mod(k,arch) == 0
        disp('archiving estimates')
        FieldArchive(:,:,archive) = FieldEstimate;
        archive = archive + 1;
    end
    k = k+1
end

% --- Save final damaged field
FieldArchive(:,:,archive) = FieldEstimate;
```

```
% ************************************************
%          *** END MINIMIZATION ALGORITHM ***
%          *** BEGIN POST PROCESSING STAGE ***
% ************************************************

timepassed = toc;
% --- Fourier No.
Fo = CalculateFourierNo(max(max(max(kxy))),min(min(min(Cpxy))));

% --- Statistics
[MaxErrors,FinalFieldError,EnergyError,ControlRmsError] =...
    FieldStats(FieldDifference,FieldDesired,ControlOpt,ControlEst);

RMSratio = FieldRMSError(k-1)/max(max(FieldDesired))*100;

% --- Plots
% FieldPlots(FieldEstimate,FieldDesired,ControlOpt,ControlEst(:,k-1),OptimalT,Tq,...
%     MaxErrors,FinalFieldError,FieldRMSError,EnergyError,ControlRmsError);

% --- Animations

% --- Archive Data
iter = 1:1:k-1;
stats = [iter; FieldRMSError; EnergyError; ControlRmsError; J]';
ControlFinal = [t' ControlEst(:,k-1) ControlOpt];

% --- Data Storage
save FieldStats.out stats -ascii -double;
save Control.out ControlFinal -ascii -double;
save FieldEstimate.out FieldEstimate -ascii -double;
save FieldOptimal.out FieldDesired -ascii -double;
save ControlEst.out ControlEst -ascii -double;

% --- Write output file
global Ttable KtableX Cptable Rhoval nlk nlCp NLEPSILON Ea A R ALPHA RDOSE
fid = fopen(outputfile,'w');
fprintf(fid,'***Thermal Field Optimization - CONGUGATE GRADIENT W/ADJOINT*** \n\n')
fprintf(fid,'TEST PERFORMED ON (year/month/day, hour:min:sec):\n')
fprintf(fid,'%4.0f/%02.0f/%02.0f, %02.0f:%02.0f:%02.0f\n',clock)

fprintf(fid,'\nFD PARAMETERS \n')
fprintf(fid,'Number of Control Volumes,x:..... %g.\n',nCellsx);
fprintf(fid,'Number of Control Volumes,y:..... %g.\n',nCellsy);
fprintf(fid,'Number of Control Volumes of surface heater:.. %g.\n',heater(1));
fprintf(fid,'Discretized heater length:..... %g.\n',heater(2));
fprintf(fid,'Error between discretized & actual heater:.... %g.\n',heater(3));
fprintf(fid,'Number of Time Steps:..... %g.\n',pmax);
fprintf(fid,'Length of interior FD Cell, x:..... %g.\n',dx(2));
fprintf(fid,'Length of interior FD Cell, y:..... %g.\n',dy(2));
fprintf(fid,'Length of Time steps in Seconds:..... %g.\n',dt);
fprintf(fid,'Fourier number:..... %g.\n\n',Fo);

fprintf(fid,'MODEL PARAMETERS \n');
fprintf(fid,'Length of domain in x (radial):..... %s\n',L(2));
fprintf(fid,'Length of domain in y (vertical):..... %s\n',L(1));
fprintf(fid,'Nominal Length / Radius of surface heater:.... %s\n',L(3));
fprintf(fid,'Heat flux initial guess:..... %s\n',FluxGuess);
fprintf(fid,'Constants for heat flux guess:..... %g\n',guess);
fprintf(fid,'Initial temperature, K:..... %g.\n',BcParamDP(1,9));
fprintf(fid,'Body reference temperature, K:..... %g.\n',BcParamDP(1,8));
fprintf(fid,'Blood perfusion parameter:..... %g.\n',BcParamDP(1,7));
fprintf(fid,'Nonlinear k modeling on:..... %g.\n',nlk);
fprintf(fid,'Nonlinear Cp tissue modeling on:..... %g.\n',nlCp);
fprintf(fid,'Nonlinear temperature range:..... %g.\n',Ttable);
fprintf(fid,'Thermal conductivity over temperature range:.. %g.\n',KtableX);
fprintf(fid,'Specific heat over temperature range:..... %g.\n',Cptable);
fprintf(fid,'Nonlinear convergence criteria:..... %g.\n',NLEPSILON);
fprintf(fid,'Constant tissue density:..... %g.\n',Rhoval);
fprintf(fid,'\nBoundary condtion type, y=0:..... %g.\n',BcTypeDP(1));
if BcTypeDP(1) == 1
    fprintf(fid,'Const temp y=0, K:..... %g.\n',BcParamDP(1,1));
else
    fprintf(fid,'Film coefficient y=0:..... %g.\n',BcParamDP(1,3));
    fprintf(fid,'Convective ambient temp y=0:..... %g.\n',BcParamDP(1,5));
end
fprintf(fid,'\nBoundary condtion type, y=H:..... %g.\n',BcTypeDP(2));
if BcTypeDP(2) == 1
    fprintf(fid,'Const temp y=H, K:..... %g.\n',BcParamDP(1,2));
else
    fprintf(fid,'Film coefficient y=H:..... %g.\n',BcParamDP(1,4));
    fprintf(fid,'Convective ambient temp y=H:..... %g.\n',BcParamDP(1,6));
    fprintf(fid,'Additional Heat flux incident on y=H:..... %g.\n',BcParamDP(2,9));
end
fprintf(fid,'\nBoundary condtion type, x=0:..... %g.\n',BcTypeDP(3));
if BcTypeDP(3) == 1
    fprintf(fid,'Const temp x=0, K:..... %g.\n',BcParamDP(2,1));
else
    fprintf(fid,'Film coefficient x=0:..... %g.\n',BcParamDP(2,3));
    fprintf(fid,'Convective ambient temp x=0:..... %g.\n',BcParamDP(2,5));
    fprintf(fid,'Additional Heat flux incident on x=0:..... %g.\n',BcParamDP(2,7));
end
fprintf(fid,'\nBoundary condtion type, x=L:..... %g.\n',BcTypeDP(4));
if BcTypeDP(4) == 1
    fprintf(fid,'Const temp x=L, K:..... %g.\n',BcParamDP(2,2));
else
    fprintf(fid,'Film coefficient x=L:..... %g.\n',BcParamDP(2,4));
    fprintf(fid,'Convective ambient temp x=L:..... %g.\n',BcParamDP(2,6));
    fprintf(fid,'Additional Heat flux incident on x=L:..... %g.\n',BcParamDP(2,8));
end

fprintf(fid,'\nOPTIMAL CONTROL PROBLEM PARAMETERS \n');
```

```
fprintf(fid,'Type of field:.............................. %s\n',FieldVariable);
fprintf(fid,'Method used to create field:................. %s\n',FieldMethod);
if FieldMethod(1:6) == 'genera';
    fprintf(fid,'Control function type used to create Field:... %s\n',FluxString);
    fprintf(fid,'Optimal protocol constants:................. %g\n',cf);
else
    fprintf(fid,'File containing imported field:............. %s\n',ImportField);
end
fprintf(fid,'Number of parameters to construct function:... %g\n',NPARAM);
fprintf(fid,'Stability parameter:........................ %g\n',ALPHA);
fprintf(fid,'Convergence criteria for no error:.......... %g.\n',epsilon);
fprintf(fid,'Max allowed iterations to minimize objective:. %g.\n',timeout);
if FieldVariable(1:4) == 'dose'
    fprintf(fid,'\nTHERMAL DOSE MODEL PARAMETERS \n');
    fprintf(fid,'Reference Temperature, C:................... %g.\n',RDOSE(1));
    fprintf(fid,'High R value:............................... %g.\n',RDOSE(2));
    fprintf(fid,'Low R value:................................ %g.\n',RDOSE(3));
    fprintf(fid,'Linearized range around Treference:......... %g.\n',RDOSE(4));
else
    fprintf(fid,'\nARRHENIUS DAMAGE MODEL PARAMETERS \n');
    fprintf(fid,'Activation Energy:.......................... %g.\n',Ea);
    fprintf(fid,'Frequency Factor:........................... %g.\n',A);
    fprintf(fid,'Universal Gas Constant:..................... %g.\n',R);
end

fprintf(fid,'\nOPTIMIZATION RESULTS\n');
fprintf(fid,'ELAPSED TIME TO RESOLVE, sec:............... %g.\n',timepassed);
fprintf(fid,'Number of completed iterations:............. %g.\n',k-1);
fprintf(fid,'Time per iteration:......................... %g.\n',timepassed/(k-1));
fprintf(fid,'Objective function value at loop end:....... %g.\n\n',J(k-1));
fprintf(fid,'FINAL RMS / MAX REAL HEAT FLUX:............. %g.\n',RMSratio(length(RMSratio)));
fclose(fid);
```

```
function[L,BcParam,BcType,Optparam,ImportField,FieldMethod,FieldVariable,...
        FluxString,FluxStringGuess,C,outputfile] = UserInput;
% ************************************************
%
%              *** Preprocessor ***
%
%     Begin here:  modify finite difference and
%     optimization parameters.  Once this file is
%     modified with the desired parameters the
%     optimization can be run with *main*
%
% ************************************************
global G RDOSE SourceType
global Ttable KtableX Cptable Rhoval nlk nlCp NLEPSILON exp
global Gvart Gtablet Sptable nlSp DEG
global Ea A R ALPHA NPARAM
global D time

% ************************************************
%
%  Name output file for current trial run
%
% ************************************************
outputfile = 'thesissim8.txt';

% ************************************************
%
%     **** Finite Difference Parameters ****
%
%     - fully implicit time discretization -
%
% ************************************************

% --- SELECT GEOMETRY & TYPE
% ************************************************
%     G = 0 -> Rectangular Coordinates
%     G = 1 -> Cylindrical Coordinates (x = r, y = z)
%     Lx = Length of domain in x direction
%     Ly = Length of domain in y direction
%     rheat = Length of 1/2 of heater at y = 0
%             or the radius of the heater if circular
% ************************************************
G = 0;
% Ly = 0.0222;
% Lx = 0.0425;
% rheat = 0.023;
Ly = 0.01;
Lx = 0.01;

% --- SELECT SOURCE FUNCTION
% ******************************************************
%     SourceType = boundary ->  Boundary heating on y = 0
%                               specify rheat
%     SourceType = volume   ->  Volumetric heating over a
%     portion of the boundary starting at y = 0, x = 0
%                               specify rheat - > radial
%                                       aheat - > axial
% ******************************************************
SourceType = 'vol';
rheat = Lx/2;
aheat = Ly/2;

L = [Ly Lx rheat aheat];

% --- DISCRETIZING PARAMETERS
% ************************************************
%     nCellsx = Number of cells in x direction (3 or greater)
%     nCellsy = Number oc cells in y direction (3 or greater)
%     nCellsq = Number of cells at y = 0 over which heat flux
%               occurs.  Must be < or = to nCellsx.
% ************************************************
nCellsx = 15;
nCellsy = 15;

D = [nCellsx nCellsy];

% --- TIME DISCRETIZING PARAMETERS
% ************************************************
%     dt = length of time step, seconds
%     pmax = number of time steps in solution
% ************************************************
dt = 20;
pmax = 91;

time = [dt pmax];

% --- LINEAR / NONLINEAR THERMAL PROPERTIES
% ******************************************************
%  nlk =  1 thermal conducivity can vary with temp
%  nlCp = 1 heat capacity can varry with temp
%  nlSp = 1 perfusion can vary with temp
%  DEG = degree of polynomial fit to tabular data
%
%     exp = 1 then its set up for the first experiment
%     exp = 0 then properties are uniform throughout
%  ******************************************************
exp = 0;
nlk = 1;
nlCp = 0;
nlSp = 1;
NLEPSILON = 0.001;
DEG = 4;
```

```
% --- INITIAL CONDITIONS
% ******************************************************
%   The program can only have uniform intitial condition
% ******************************************************
To = 310;
% To = 37;

% --- THERMAL PROPERTIES
% ******************************************************
%     Temperature dependent properties can be modeled.
%
%     This program is not developed to have regional
%     differences in thermal properties, but it can be
%     modified to do so.
%
%     Ttable is a vector of temperatures spanning the range
%     over which properties vary.  If Ttable is 0, then
%     properties are constant.  Ttable must be C, not K.
%
%     Variables_table = Values of thermal properties that
%     correspond to Ttable.  Only the first value listed is
%     used in the case of constant properties.
%
%     KtableX = thermal conductivity
%     Cptable = specific heat
%     Rhoval = value of density, constant
% ******************************************************
% Ttable = [24 28 32 64 94];
Ttable = [37 100];
% Ttable = [0];
KtableX = [.21 .37 .16];
Cptable = [4000];
% Cptable = [3990];
Rhoval = [1040];

% --- BOUNDARY CONDITIONS
% ******************************************************
%   Enter type of boundary condition here, y = 0 b.c. is
%   set later.  Type 1 is temperature dependent and type
%   2 is dependent on the first derivative of temperature
%   (heat flux).
% ******************************************************
BcTypeyo = 2;
BcTypeyH = 1;
BcTypexo = 2;
BcTypexL = 1;

% ************************************************************
%      BC's of the first kind - Constant Temperature Options
%      Corner cell constant temperatures are set to the
%      temperature at y = 0 and y = H.
%
%      *** SET BCTYPE = 1 ***
%
%      Tyo = Temperature at y = 0
%      TyH = Temperature at y = H
%      Txo = Temperature at x = 0
%      TxL = Temperature at x = L
% ************************************************************
Tyo = 0;
TyH = To;
Txo = 0;
TxL = To;

% ************************************************************
%      BC's of the second kind - Depend on first derivative of
%      independent variable
%
%      *** SET BCTYPE = 2 ***
%
%      hyo = convective cooling coef at y = 0
%      hyH = convective cooling coef at y = H
%      Tinfyo = Temperature of fluid surrounding material at y = 0
%      Tinfyl = Temperature of fluid surrounding material at y = H
%      qyH = Heat flux incident on y = H
%
%      hxo = convective cooling coef at x = 0
%      hxl = convective cooling coef at x = L
%      Tinfxo = Temperature of fluid surrounding material at x = 0
%      Tinfxl = Temperature of fluid surrounding material at x = L
%      qxo = Heat flux incident on x = 0
%      qxL = Heat flux incident on x = L
% ************************************************************
hyo = 0;
% hyH = 5;
hyH = 0;
Tinfyo = 0;
% TinfyH = To;
TinfyH = 0;
qyH = 0;

hxo = 8.5;
% hxL = 3.5;
hxL = 0;

Tinfxo = To - 13;
% TinfxL = To;
TinfxL = 0;

qxo = 0;
qxL = 0;
```

```
% --- VOLUMETRIC SOURCES
% ************************************************************
%       Fin type problem
%
%       *** Sptable length should be Ttable length ***
%
%       In this case you want to model convection throughout the
%       material.  Keep in mind, if this is specfied you can still
%       specify constant temperature boundary conditions.
%
%       Sfin is a lumped fin parameter, it is Perim*h/Area
%       Tinffin is the ambient fluid temperature around the fin
%
%       Leave these as zero if the problem is not fin-type
%
%        If exp = 0, then regional differences are used with 4
%        zones of varying convective effects.
% ************************************************************
% Sptable = [0 100 500 8000 10000];
% Sptable = [0 4 20 100 800 1900 1950 2000 2050 2100];
Sptable = [87000 10000]
% Sptable =  [0 0 0 0];
Tinffin = To;
% Tinffin = 0;

% ************************************************************
%     Space varying to be added later
%
%     Gvart = Value of volumetric heat while on (Watts/vol).
%     Gtablet = c4 = t on
%               c5 = t off
%
%     Leave g_table as zero if the problem has no volumetric
%     heat generation
% ************************************************************
Gvart = [0];
Gtablet = [0 0];

% ************************************************
%       **** Optimization Parameters ****
% ************************************************

% ************************************************
%     CHOSE YOUR FIELD FUNCTION,
%         'dose' = Optimized by thermal dose - Use C
%         'damage' = Optimized by thermal damage - Use K
% ************************************************
FieldVariable = 'damage';

% ************************************************
%     CHOSE YOUR METHOD OF CREATING THE FIELD VARIABLE,
%         'import' = Optimal field is imported
%         'generate' = Optimized field is generated
%         via a user-chosen heat flux function
% ************************************************
FieldMethod = 'generate';

% --- ARHENIUS PROPERTIES OF MATERIAL
%*******************************************************
%
%     FOR USE ONLY WITH THERMAL DAMAGE OPTIMIZER
%
%     The material used in this experiment should have
%     known arhenius properties.
%     Ea = Activation energy (J/mol)
%     A  = Molecular Frequency Factor (collison rate) 1/s
%     R  = Universal gas constant (J/mol/K)
%
%*******************************************************
% Ea = 3.85E5;
% A  = 3.8E57;
R  = 8.31;

Ea = 6.28E5;
A = 3.1E98;

% --- THERMAL DOSE PARAMETERS
%*******************************************************
%
%     FOR USE ONLY WITH THERMAL DOSE OPTIMIZER
%
%     ref = reference temperature, normally 43 C
%     rhi = High values of R in dose calculation
%     rlo = Low value of R in dose calculation
%     reps = Linearized range from RLo to Rhi
%
%     RDOSE = [ref rhi rlo reps]
%
%*******************************************************
RDOSE = [43 0.5 0.25 0.5];

% --- DAMAGE/DOSE FIELD TO BE ESTIMATED
% ************************************************************
%     The feild to be estimated can be created
%     of two ways.  NOT BOTH!
%
%     1.  A damage/dose field is imported from a premade
%     matrix.  How you make it is up to you. Make sure the
%     matrix is in MATLAB format and enter the file name below.
%     The file must be in the current directory.
%     2.  A heat flux function is chosen from a premade list
```

```
%     and the resulting damage/dose field is set equal to the
%     desired design function.
% *********************************************************

% --- 1. IMPORT DAMAGE/DOSE FIELD
ImportField = 'damage43.out';

% --- 2. CHOOSE SOURCE FUNCTION
% ************************************************************
%     FluxString determines which function of the ones below will
%     be the heat flux in time
%
%     String Options for each face.
%     'off'    = this option is disabled
%     'constant' = constant heat flux with magnitude c1.
%     'squpls'   = square pulses heat source with magnitude c1.  Pulse
%                  persists for c2 seconds.  c3 is
%                  the time between pulses and c4 is the number of
%                  pulses
%     'square' = Single square pulse of magnitude c1, turns on at time
%                  c2 and goes off at c3.
%     'sinwav' = sinusiod heat flux with amplitude c1 and c2 cycles.
%     'triang' = single triangular heat flux pulse with hieght c1
%                  occuring at midpoint of time domain and with start
%                  time c2 and end time c3.  c4 is a base addtion added
%                  at every time step.
%     'tripls' = triangular heat flux pulses with hieght c1 occuring at
%                  midpoint of pulse length c2.  c3 is the time between
%                  pulses and c4 is the number of pulses.
%     'linear' = linearly increasing heat flux.  Starts at t = c1
%                   increases up to c2 over the time domain
%
%     For options with no c3 or c4 enter 0
%
%     For fully insulated surface use 'consta' and magnitude 0
% ************************************************************
% FluxString = ['tripls'];
% cf = [1100 6*dt 19*dt 4];
% FluxString = ['squpls'];
% cf = [3000 400 50 4];
% FluxString = ['triang'];
% cf = [200000 0 1800 0];
FluxString = ['consta'];
cf = [900000 0 2700 100000];


% --- 2.1 PARAMETRIZE HEAT FLUX FUNCTION
% ************************************************************
%     The heat flux function can be parametrized if the user
%     chooses.  This guides the optimiation routine to recover a
%     heat flux which cases the thermal damage close to the heat
%     flux function which causes the thermal damage.
%
%     NPARAM = number of parameters used in spline formation of
%     heat flux.  If NPARAM = 0 parameter estimation is off
% ************************************************************
NPARAM = 50;

% --- INITIAL GUESS FOR ALGORITHM
% ************************************************************
%     The algorithm begins with an initial guess which is
%     entered below.  This applies to both dose and damage
%     optimization as well as imported and generated fields
% ************************************************************
% FluxStringGuess = ['consta'];
% guess = [520000 0 1800 0];
% FluxStringGuess = ['linear'];
% guess = [0 5000 0 3000];
FluxStringGuess = ['square'];
guess = [5000000 250 1550 100000];

% --- CONVERGENCE PARAMETERS
% ************************************************
% ALPHA = Stabilization parameter (zero recommended)
% epsilon = Convergence criteron of objective function
% timeout = Maximum attempts to converge
% arch = Number of iterations between data archiving
% ************************************************
ALPHA = 0;
epsilon = 10E-8;
timeout = 150;
arch = 10;

% --- NOISE ADDING
% ************************************************
%  In order to evaluate the robustness of this
%  routine, we can add noise to the control parameter,
%  which in this case is the surface heat flux.
%
%     Noisepack = 0 (noise off) or 1 (noise on)
%     Amplitude = percent of peak optimal heat flux
%     which will be randomly added to the heat flux
% ************************************************
noise = 0;
amplitude = 1;

% END INPUT FILE

%  DO NOT TOUCH!
C = [cf;[0 0 0 0];guess];

Optparam = [epsilon timeout noise amplitude arch];
```

```
BcParam = [Tyo TyH hyo hyH Tinfyo TinfyH 0 Tinffin To;
           Txo TxL hxo hxL Tinfxo TinfxL qxo  qxL     qyH];

BcType = [BcTypeyo BcTypeyH BcTypexo BcTypexL];

% PP = spline(Ttable,Sptable);
% Tvec = 24:94;
% WT = ppval(PP,Tvec);
% plot(Ttable,Sptable,'o',Tvec,WT,'k');
```

```
function [heater] = geodisc(L,D)
% ****************************************
%
%   [heater] = geodisc(L,D)
%   Function m file to solve for discretizing
%   geometry parameters.  Cells are uniformly
%   constructed with half cells at boundaries.
%
%     Input:  L -> vector of lengths
%             D -> vector of number of cells
%
%     Output: dx, dy ->  each element in these
%             vectors is the lenght of a cell
%             x, y ->  coordinate location of nodes
%             in x and y directions
%             heater ->  if there is a heater on the
%             bottom of the domain contains a
%             vector of the number of cells
%             over which the heat is entered.
%             the second element is the error
%             if the heater length doesn't
%             fit a integer number of cells.
%
%   F.S. Gayzik,  Jan 04
%
% ****************************************
global x y dx dy heater
% --- Geometry discretization
    dx = L(2)/(D(1)-1);
    midcells = dx*ones(1,D(1)-2);
    range = [0:(D(1)-1)];
%     range = [0:L(2)/dx];
    x = dx*range;
    dx = [dx/2 midcells dx/2];

    %Calculate discrete area x
    if L(3) > 0
        xCells = cumsum(dx);
        [val,nCellsqx] = min(abs(xCells-L(3)));
        rerrorx = abs(xCells(nCellsqx)-L(3))/L(3)*100;
        heatlengthx = cumsum(dx(1:nCellsqx));
        heater = [nCellsqx heatlengthx(nCellsqx) rerrorx];
    else
        heater = [0 0 0 ];
    end

    dy = L(1)/(D(2)-1);
    midcells = dy*ones(1,D(2)-2);
    range = [0:(D(2)-1)];
    y = dy*range;
    dy = [dy/2 midcells dy/2];

    %Calculate discrete area y
    if L(4) > 0
        yCells = cumsum(dy);
        [val,nCellsqy] = min(abs(yCells-L(4)));
        rerrory = abs(yCells(nCellsqy)-L(4))/L(4)*100;
        heatlengthy = cumsum(dy(1:nCellsqy));
        heater(2,:) = [nCellsqy heatlengthy(nCellsqy) rerrory];
    else
        heater(2,:) = [0 0 0 ];
    end


function t = timedisc;
% ****************************************
%
%   Function m file that returns
%   discretized time domain
%
% ****************************************
global time t

dt = time(1); pmax = time(2);
% --- Time discretization
t = dt*(0:pmax-1);
```

```
function [bcout] = surfacebc(string,c);
% ****************************************
%
%   Creates the various boundary condtions
%
%   It is not necisary to use the constant if
%   function is not constant for the 'source' case
%
% ****************************************
global time D t

dt = time(1); pmax = time(2);
ii = D(1);
bcout = zeros(pmax,1);

% ****************************************
% Constant
% ****************************************
if string == 'consta'
    bcout(:,1) = c(1);

% ****************************************
% Sinusoid
% ****************************************
elseif string == 'sinwav'
    bcout = (c(1)*sin(2*pi*c(2)/(t(length(t)))*t));

% ****************************************
% Single Square Pulse
% ****************************************
elseif string == 'square'
    bcout = c(1)*ones(length(t),1);

    for i = 1:length(t)
        if t(i) < c(2)
            bcout(i) = 0;
        elseif t(i) > c(3)
            bcout(i) = 0;
        end
    end
    bcout = bcout + c(4);

% ****************************************
% Square Pulses
% ****************************************
elseif string == 'squpls'
    bcout = c(1)*ones(length(t),1);
    pulsecount = 1;
    for i = 1:length(t)
        if c(4) >= pulsecount
            if t(i) > c(2)*pulsecount + (pulsecount-1)*c(3);
                bcout(i) = 0;
            end
            if t(i) >= pulsecount*(c(2)+c(3));
                pulsecount = pulsecount + 1;
            end
        else
            bcout(i) = 0;
        end
    end

% ****************************************
% Single Triangular Pulse
% ****************************************
elseif string == 'triang'
    bcout = zeros(pmax,1);

    mid = floor(pmax*.5)+1;
    offset = c(2) - t(1);
    a = c(3);

    bup = -c(1)*c(2)/(t(mid)-c(2));
    mup = c(1)/(t(mid)-c(2));

    mdown = -c(1)/(a-t(mid));
    bdown = c(1)*(1 + t(mid)/(a-t(mid)));

    for i = 1:length(t);
        if t(i) <= t(mid);
            bcout(i) = mup*t(i) + bup;
        end
        if t(i) >= t(mid);
            bcout(i) = mdown*t(i) + bdown;
        end
        if t(i) >= a;
            bcout(i) = 0;
        end
   end
   bcout = bcout + c(4);

% ****************************************
% Multiple Triangular Pulses
% ****************************************
elseif string == 'tripls'
    bcout = zeros(pmax,1);

    % slopes + intercepts
    mid = c(2)*.5;
    offset = 0;
    a = c(2);

    mup = c(1)/(mid-0);
```

```
    mdown = -c(1)/(a-mid);
    bdown = c(1)*(1 + mid/(a-mid));
    trilength = floor(c(2)/dt);
    waitlength = floor(c(3)/dt);

    % construct triangle
    for i = 1:trilength;
        if t(i) <= mid;
            tripulse(i) = mup*t(i);
        elseif t(i) >= mid;
            tripulse(i) = mdown*t(i) + bdown;
        end
    end

    % construct boundary condtion
    T = trilength+waitlength;
    for pc = 0:c(4)-1
        bcout(1+pc*T:(pc+1)*(T)) = [tripulse zeros(1,waitlength)];
    end
    bcout(length(t)+1:length(bcout)) = [];
%         if c(4) >= pulsecount
%             if t(i) > c(2)*pulsecount + (pulsecount-1)*c(3);
%                 bcout(i) = 0;
%             end
%             if t(i) >= pulsecount*(c(2)+c(3));
%                 pulsecount = pulsecount + 1;
%             end
%         else
%             bcout(i) = 0;
%         end
%     end


% ****************************************
% Linear Increase
% ****************************************
elseif string =='linear'
    bcout = zeros(pmax,1);

    m = c(2) / (t(length(t)) - c(1));
    b = -1*c(1)*c(2) / (t(length(t)) - c(1));

    for i = 1:length(t);
        if t(i) >= c(1)
            bcout(i) = m*t(i) + b;
        end
    end
    bcout = bcout+c(4);

% ****************************************
% Square Wave
% ****************************************
elseif string =='squwav'
    bcout = c(1)*ones(pmax,1);

    T = c(2)+c(3);
    Noff = floor(c(2)/dt);
    Npulse = floor(c(3)/dt);
    cycles = floor(pmax/(T/dt));

    for j = 1:cycles
        start = (j-1)*(Noff+Npulse)+1;
        bcout(start:(start+Noff)) = 0*bcout(start:(start+Noff));
        if j == cycles & (start+Npulse+Noff) < pmax
            bcout((start+Noff+Npulse+1):length(bcout)) = 0
        end
    end


% ****************************************
% Imported data file
% ****************************************
elseif string == 'import'
 %   load filename
else
    disp('unrecognizable boundary condition heat flux = zeros')
    bcout = zeros(pmax,1);
end
```

```
function [T,kxy,cpxy,Sp,NlIterations] =
DirectProblemFD(BcType,bcs, qt, R);
% ***********************************************
%    2D Finite Difference Code (Where the magic happens)
%
%    Description:  2D code for use with rectangular or
%          cylindrical coordinates.  Nearly explicit,
%          Crank-Nicolson and fully implicit time
%          discretization are possible.  Temperature
%          dependent thermal properties are included if
%          desired.
%
%     Global Inputs:
%             Ttable, KtableX, KtableY, Cptable, Rhoval
%             BCTYPExo, BCTYPExL, BCTYPEyH ->  are all b.c's
%             see the input file
%             FIN ->  Fin type term, or internal convection
%
%     Inputs:  dx, dy = space discretization
%              D      = vector with number of nodes in x and y
%              time   = vector with lenght of time step and number
%                       of time steps
%              bcs    = vector of boundary conditions (see input file
%                         for specifics)
%              qt     = time dependent boundary condtion
%              To     = initial temperature field
%              R      = vector containing ncells get heat
%
%     Outputs:  T ->  Temperature field in time
% ***************************************************
global Ttable KtableX KtableY Cptable Rhoval nlk nlCp nlSp NLEPSILON
global G
global Gvart SourceType
global dx dy D time

% --- "Unzip" discretization parameters
ii = D(1); jj = D(2);
dt = time(1);
pmax = time(2);

% --- "Unzip" boundary condition types
BCTYPExo = BcType(1);  BCTYPExL = BcType(2);
BCTYPEyo = BcType(3);  BCTYPEyH = BcType(4);

% --- "Unzip" boundary condtions parameters
Tyo = bcs(1,1); TyH = bcs(1,2); hyo = bcs(1,3); hyH = bcs(1,4); Tinfyo = bcs(1,5); TinfyH = bcs(1,6);
Txo = bcs(2,1); TxL = bcs(2,2); hxo = bcs(2,3); hxL = bcs(2,4); Tinfxo = bcs(2,5); TinfxL = bcs(2,6);
Sfin = bcs(1,7); Tinffin = bcs(1,8); To = bcs(1,9);
qxo = bcs(2,7); qxL = bcs(2,8); qyH = bcs(2,9);
nCellsqx = R(1,1); nCellsqy = R(2,1);

% --- create key matrices
T = zeros(jj,ii,pmax);
Sc = zeros(jj,ii);
a = zeros(jj,ii);
b = zeros(jj,ii);

% Sp = Sfin*ones(jj,ii);
% Sc = Gvart*ones(jj,ii) + Tinffin*Sp;

% *** SOLUTION ***

% --- Impose initial condition and initial properties
T(:,:,1) = To;
kxy(:,:,1) = NlConductivity(T(:,:,1));
cpxy(:,:,1) = NlHeatCap(T(:,:,1));
Sp(:,:,1) = Nlperfusion(T(:,:,1));

if SourceType(1:3) == 'bou'
    Gvart =  zeros(length(qt),1);
    Sc(:,:,1) = Tinffin*Sp(:,:,1);
elseif SourceType(1:3) == 'vol'
    Gvart = qt; qt = zeros(length(qt),1);
    Sc(:,:,1) = Tinffin*Sp(:,:,1);
    Sc((jj-nCellsqy+1):jj,1:nCellsqx,1) = Gvart(1)*ones(nCellsqy,nCellsqx)...;
        + Sc((jj-nCellsqy+1):jj,1:nCellsqx,1);
end


% --- Compute directional components used in area
% and volume calculations.
for i = 1:jj
    dmatx(i,:) = dx;
end

for i = 1:ii
    dmaty(:,i) = dy';
end

% --- Multiply coefs by normal areas and volumes
% ************************************************
% G = 0 -> Rectangular, G = 1 -> Cylindrical
% ************************************************
if G == 0
    areaE = dmaty;  areaW = areaE;
    areaN = dmatx;  areaS = areaN;
    V = dmaty.*dmatx;
else %Cylindrical case
    r = cumsum(dmatx,2);
    rw = RotateLeft(r); rw(:,1) = 0;
    areaE = 2*pi.*r.*dmaty;  areaW = 2*pi.*rw.*dmaty;
    rsq = r.^2;
```

```
    deltr = [r(:,1).^2 diff(rsq,1,2)];
    areaN = pi*deltr; areaS = areaN;
    V = dmaty.*areaN;
end

% Here is the part that I changed on June 22nd looking for
% the reason why I'm getting incorrect temperatures at the
% edges.  This should also save some computational time.
% The change has to do with edge nodes being only half
% thickness and the way the interface conductivity is
% calculated.
dmatx = dx(2); dmaty = dy(2);


for k = 2:pmax
    Tnl(:,:,1) = 1000000*ones(jj,ii);
    m = 1;
    while m < 100

        % --- prepare temperature dependent properties
        aE = EastSide(kxy(:,:,k-1),dmatx).*areaE;
        aW = WestSide(kxy(:,:,k-1),dmatx).*areaW;
        aS = SouthSide(kxy(:,:,k-1),dmaty).*areaS;
        aN = NorthSide(kxy(:,:,k-1),dmaty).*areaN;

        % --- Diagonal Coefs
        % Divide by delta t / 2 because of half step
        % solution technique
        a0 = Rhoval*cpxy(:,:,k-1).*V/(dt/2);

        % --- Compute initial Coefficients for diagonal and b vector
        %*********************************************************
        %
        %    hor / vert  -->  horizontal or vertical
        %    a for internal cells will include:
        %       - all constants that are multiplied by
        %         T(i,j)^(p+next) time step
        %    b for internal cells will include:
        %       - all constants added to RHS that don't
        %         get multiplied by a temperature
        %    c for internal cells will include:
        %       - all constants that are multipled by
        %         T(i,j)^(p) time step
        %
        %*********************************************************
        %Prepare x direction constants
        ahor = a0 + aE + aW + Sp(:,:,k-1).*V;
        chor = a0 - aS - aN;

        %Preparte y direction constants
        avert = a0 + aN + aS + Sp(:,:,k-1).*V;
        %Left out a0 b/c it gets multipled by T(p) not T(p+1/2)
        cvert = a0 -aE - aW;

        %Prepare shared components
        b = Sc(:,:,k-1).*V;

        % --- Create boundary conditions
        %*********************************************************
        %     Boundary conditons will created for each zone of
        %     cells that compose the boundary of the 2D area
        %     being modeled.  Start with central cells on each
        %     face and go clockwise around.
        %
        %     Next all the corners are done.
        %
        %     The first generation of this code will only allow
        %     for a time dependent boundary condtion at y = 0.
        %     This will only effect cells (1:nCellsx,1).
        %
        %     Note that the notation used in my notes is opposite
        %     that of MATLAB in y direction.  So that the boundary
        %     y = 0 is actually y = jj here.
        % *********************************************************
        midx = 2:jj-1;
        midy = 2:ii-1;

        %*****************
        %   SIDES
        %*****************

        % --- SET B.C's at x = 0,
        if BCTYPExo == 1
            % cells (1,2:jj-1)
            aE(midx,1) = 0; aN(midx,1) = 0; aS(midx,1) = 0; ahor(midx,1) = 1; avert(midx,1) = 1;
            b(midx,1) = Txo; chor(midx,1) = 0; cvert(midx,1) = 0;
        elseif BCTYPExo == 2
            % cells(1,2:jj-1)
            ahor(midx,1) = ahor(midx,1) + areaW(midx,1)*hxo;
            b(midx,1) = b(midx,1) + areaW(midx,1)*(hxo*Tinfxo + qxo);

            cvert(midx,1) = cvert(midx,1) - areaW(midx,1)*hxo;
        end

        % --- SET B.C's at y = H,
        if BCTYPEyH == 1
            % Cells(2:ii-1,jj)
            aW(1,midy) = 0; aE(1,midy) = 0; aS(1,midy) = 0; avert(1,midy) = 1; ahor(1,midy) = 1;
            cvert(1,midy) = 0; chor(1,midy) = 0;
            b(1,midy) = TyH;
        elseif BCTYPEyH == 2
            % Cells(2:ii-1,jj)
```

```
    avert(1,midy) = avert(1,midy) + areaN(1,midy)*hyH;
    b(1,midy) = b(1,midy) + areaN(1,midy)*(hyH*TinfyH + qyH);

    chor(1,midy) = chor(1,midy) - areaN(1,midy)*hyH;
end

% --- SET B.C's at x = L,
if BCTYPExL == 1
    % Cells(ii,2:jj-1)
    aW(midx,ii) = 0; aN(midx,ii) = 0; aS(midx,ii) = 0; ahor(midx,ii) = 1; avert(midx,ii) = 1;
    chor(midx,ii) = 0; cvert(midx,ii) = 0; b(midx,ii) = TxL;
elseif BCTYPExL == 2
    % Cells(ii,2:ii-1)
    ahor(midx,ii) = ahor(midx,ii) + areaE(midx,ii)*hxL;
    b(midx,ii) = b(midx,ii) + areaE(midx,ii)*(hxL*TinfxL + qxL);

    cvert(midx,ii) = cvert(midx,ii) - areaE(midx,ii)*hxL;
end

% --- SET B.C's at y = 0,
if BCTYPEyo == 1
    % Cells(2:jj-1,1)
    aW(jj,midy) = 0; aN(jj,midy) = 0; aE(jj,midy) = 0; avert(jj,midy) = 1; ahor(jj,midy) = 1;
    cvert(jj,midy) = 0; chor(jj,midy) = 0; b(jj,midy) = Tyo;
elseif BCTYPEyo == 2
    % Cells(2:ii-1,1)
    avert(jj,midy) = avert(jj,midy) + areaS(jj,midy)*hyo;
    b(jj,midy) = b(jj,midy) + areaS(jj,midy)*(hyo*Tinfyo);%<- q goes here;

    chor(jj,midy) = chor(jj,midy) - areaS(jj,midy)*hyo;
end

%*****************
%   CORNERS
%*****************

% --- SET B.C's at  Top left corner
if BCTYPExo == 1 & BCTYPEyH == 1
    aE(1,1) = 0; aS(1,1) = 0; ahor(1,1) = 1; avert(1,1) = 1;
    b(1,1) = (TyH+Txo)/2;
    chor(1,1) = 0; cvert(1,1) = 0;

elseif BCTYPExo == 2 & BCTYPEyH == 1
    % cell(1,jj) - Set to constant temperature at yH
    aE(1,1) = 0; aS(1,1) = 0; ahor(1,1) = 1; avert(1,1) = 1;
    b(1,1) = TyH;
    chor(1,1) = 0; cvert(1,1) = 0;

elseif BCTYPExo == 1 & BCTYPEyH == 2
    % cell(1,jj) - Set to constant temperature at yH
    aE(1,1) = 0; aS(1,1) = 0; ahor(1,1) = 1; avert(1,1) = 1;
    b(1,1) = Txo;
    chor(1,1) = 0; cvert(1,1) = 0;

elseif BCTYPExo == 2 & BCTYPEyH == 2
    ahor(1,1) = ahor(1,1) + areaW(1,1)*hxo;
    avert(1,1) = avert(1,1) + areaN(1,1)*hyH;

    b(1,1) = b(1,1) + areaW(1,1)*(hxo*Tinfxo + qxo)+ areaN(1,1)*(hyH*TinfyH + qyH);

    chor(1,1) = chor(1,1) - areaN(1,1)*hyH;
    cvert(1,1) = cvert(1,1) - areaW(1,1)*hxo;
end

% --- SET B.C's at Top right corner
if BCTYPExL == 1 & BCTYPEyH == 1
    aW(1,ii) = 0; aS(1,ii) = 0; avert(1,ii) = 1; ahor(1,ii) = 1;
    b(1,ii) = (TxL+TyH)/2;
    cvert(1,ii) = 0;    chor(1,ii) = 0;

elseif BCTYPExL == 1 & BCTYPEyH == 2
    aW(1,ii) = 0; aS(1,ii) = 0; avert(1,ii) = 1; ahor(1,ii) = 1;
    b(1,ii) = TxL;
    cvert(1,ii) = 0; chor(1,ii) = 0;

elseif BCTYPExL == 2 & BCTYPEyH == 1
    % cell(1,jj) - Artificially set biot number high on north face.
    aW(1,ii) = 0; aS(1,ii) = 0; avert(1,ii) = 1; ahor(1,ii) = 1;
    b(1,ii) = TyH;
    cvert(1,ii) = 0; chor(1,ii) = 0;

elseif BCTYPExo == 2 & BCTYPEyH == 2
    ahor(1,ii) = ahor(1,ii) + areaE(1,ii)*hxL;
    avert(1,ii) = avert(1,ii) + areaN(1,ii)*hyH;

    b(1,ii) = b(1,ii) + areaE(1,ii)*(hxL*TinfxL + qxL) + areaN(1,ii)*(hyH*TinfyH + qyH);

    chor(1,ii) = chor(1,ii) - areaN(1,ii)*hyH;
    cvert(1,ii) = cvert(1,ii) - areaE(1,ii)*hxL;
end

% --- SET B.C'S at Bottom right corner
if BCTYPExL == 1 & BCTYPEyo == 1
    aW(jj,ii) = 0; aN(jj,ii) = 0; ahor(jj,ii) = 1; avert(jj,ii) = 1;
    b(jj,ii) = (TxL+Tyo)/2;
    chor(jj,ii) = 0; cvert(jj,ii) = 0;

elseif BCTYPExL == 1 & BCTYPEyo == 2
    % cell(ii,jj) - Artificially set biot number high on north face.
    aW(jj,ii) = 0; aN(jj,ii) = 0; ahor(jj,ii) = 1; avert(jj,ii) = 1;
    b(jj,ii) = TxL;
    chor(jj,ii) = 0; cvert(jj,ii) = 0;
```

```
elseif BCTYPExL == 2 & BCTYPEyo == 1
    % cell(jj,ii) - Artificially set biot number high on south face.
    aW(jj,ii) = 0; aN(jj,ii) = 0; ahor(jj,ii) = 1; avert(jj,ii) = 1;
    b(jj,ii) = Tyo;
    chor(jj,ii) = 0; cvert(jj,ii) = 0;

elseif BCTYPExL == 2 & BCTYPEyo == 2
    ahor(jj,ii) = ahor(jj,ii) + areaE(jj,ii)*hxL;
    avert(jj,ii) = avert(jj,ii) + areaS(jj,ii)*hyo;

    b(jj,ii) = b(jj,ii) + areaE(jj,ii)*(hxL*TinfxL + qxL) + areaS(jj,ii)*(hyo*Tinfyo);% <- don't forget qyo

    chor(jj,ii) = chor(jj,ii) - areaS(jj,ii)*hyo;
    cvert(jj,ii) = cvert(jj,ii) - areaE(jj,ii)*hxL;
end

% --- SET B.C's at Bottom left corner
if BCTYPExo == 1 & BCTYPEyo == 1
    aE(jj,1) = 0; aN(jj,1) = 0; ahor(jj,1) = 1; avert(jj,1) = 1;
    b(jj,1) = (Txo+Tyo)/2;
    chor(jj,1) = 0; cvert(jj,1) = 0;

elseif BCTYPExo == 2 & BCTYPEyo == 1
    aE(jj,1) = 0; aN(jj,1) = 0; ahor(jj,1) = 1; avert(jj,1) = 1;
    b(jj,1) = Tyo;
    chor(jj,1) = 0; cvert(jj,1) = 0;

elseif BCTYPExo == 1 & BCTYPEyo == 2
    aE(jj,1) = 0; aN(jj,1) = 0; ahor(jj,1) = 1; avert(jj,1) = 1;
    b(jj,1) = Txo;
    chor(jj,1) = 0; cvert(jj,1) = 0;

elseif BCTYPExo == 2 & BCTYPEyo == 2
    ahor(jj,1) = ahor(jj,1) + areaW(jj,1)*hxo;
    avert(jj,1) = avert(jj,1) + areaS(jj,1)*hyo;

    b(jj,1) = b(jj,1) + areaW(jj,1)*(hxo*Tinfxo + qxo) + areaS(jj,1)*(hyo*Tinfyo); % <- don't forget qyo

    chor(jj,1) = chor(jj,1) - areaS(jj,1)*hyo;
    cvert(jj,1) = cvert(jj,1) - areaW(jj,1)*hxo;
end

%*******************************************
%
%   Enter Solver!
%
%*******************************************
Thalf = zeros(jj,ii);
bt = zeros(jj,ii);

% X direction sweep (horizontal)

for hor = 1:jj
    if hor == 1
        bhor(hor,:) = b(hor,:) + aS(hor,:).*T(hor+1,:,k-1);
    elseif (hor > 1) & (hor < jj)
        bhor(hor,:) = b(hor,:) + aS(hor,:).*T(hor+1,:,k-1) + aN(hor,:).*T(hor-1,:,k-1);
    elseif hor == jj
        bhor(hor,:) = b(hor,:) + aN(hor,:).*T(hor-1,:,k-1);
        %add in time dependent heating
        bhor(hor,1:nCellsqx) = bhor(hor,1:nCellsqx) + areaS(jj,1:nCellsqx)*qt(k-1);
    end

    % ***********************************************
    % Tridiagonal Solver - Solve horizontally T(p+1/2)
    % ***********************************************
    A = zeros(ii,3);
    % --- Boundary conditions
    A(1,[1:2]) = [ahor(hor,1) -aE(hor,1)];
    A(ii, [2:3]) = [-aW(hor,ii) ahor(hor,ii)];
    % --- Matrix body
    A(2:(ii-1),1) = -aW(hor,2:(ii-1))';
    A(2:(ii-1),2) = ahor(hor,2:(ii-1))';
    A(2:(ii-1),3) = -aE(hor,2:(ii-1))';
    % --- Compute solution
    Thalf(hor,:) = tridiag(T(hor,:,k-1).*chor(hor,:),A,bhor(hor,:));
    clear A;
end

clear bhor;
Tnl(:,:,2) = zeros(jj,ii);
% Y direction sweep (vertical)
for vert = ii:-1:1;
    if vert == 1
        bvert(:,vert) = b(:,vert) + aE(:,vert).*Thalf(:,vert+1);
    elseif (vert > 1) & (vert < ii)
        bvert(:,vert) = b(:,vert) + aE(:,vert).*Thalf(:,vert+1) + aW(:,vert).*Thalf(:,vert-1);
    elseif vert == ii
        bvert(:,vert) = b(:,vert) + aW(:,vert).*Thalf(:,vert-1);
    end

    %add time dependent heating at y = 0
    if vert <= nCellsqx;
        bvert(jj,vert) = bvert(jj,vert) + areaS(jj,vert)*qt(k-1);
    end

    % ***********************************************
    % Tridiagonal Solver - Solve vertically for T(p+1)
    % ***********************************************
    A = zeros(jj,3);
    % --- Boundary conditions
```

```
            A(1,[1:2]) = [avert(1,vert) -aS(1,vert)];
            A(jj, [2:3]) = [-aN(jj,vert) avert(jj,vert)];
            % --- Matrix body
            A(2:(jj-1),1) = -aN(2:(jj-1),vert);
            A(2:(jj-1),2) = avert(2:(jj-1),vert);
            A(2:(jj-1),3) = -aS(2:(jj-1),vert);
            % --- Compute solution
            Tvert = tridiag((Thalf(:,vert).*cvert(:,vert))',A,bvert(:,vert)');
            Tnl(:,vert,2) = Tvert';
            clear A;
        end

        clear bvert;bt;b;
        clear a0;aE;aW;aS;aN;ahor;avert;chor;cvert;
        % --- Skip convergence check if nonlinear properties are off
        if nlk == 0 & nlCp == 0 & nlSp == 0
            kxy(:,:,k) = kxy(:,:,k-1);
            cpxy(:,:,k) = cpxy(:,:,k-1);
            Sp(:,:,k) = Sp(:,:,k-1);
            Sc(:,:,k) = Tinffin*Sp(:,:,k-1);
            % Add in volumetric heating if that is the case
            Sc((jj-nCellsqy+1):jj,1:nCellsqx,k) = Sc((jj-nCellsqy+1):jj,1:nCellsqx,k)...
                + Gvart(k)*ones(nCellsqy,nCellsqx);
            % Set temperature at current time step to converged
            % nonlinear temperature
            T(:,:,k) = Tnl(:,:,2);
            clear Tnl;
            % Store number of iterations that occured to find
            % convergence
            NlIterations(k) = m;
            % Reset loop variable to get out of nonlinear loop
            m = 100;
        else
            % --- Check for convergence if properties are nl
            DelT = abs(Tnl(:,:,2)-Tnl(:,:,1));
            if m == 99
                % Force convergence
                kxy(:,:,k) = kxy(:,:,k-1);
                cpxy(:,:,k) = cpxy(:,:,k-1);
                Sp(:,:,k) = Sp(:,:,k-1);
                Sc(:,:,k) = Tinffin*Sp(:,:,k-1);
                % Add in volumetric heating if that is the case
                Sc((jj-nCellsqy+1):jj,1:nCellsqx,k) = Sc((jj-nCellsqy+1):jj,1:nCellsqx,k)...
                    + Gvart(k)*ones(nCellsqy,nCellsqx);
                T(:,:,k) = Tnl(:,:,2);
            elseif max(max(DelT)) >= NLEPSILON
                % No convergence, update thermal properties ONLY
                kxy(:,:,k-1) = NlConductivity(Tnl(:,:,2));
                Cpxy(:,:,k-1) = NlHeatCap(Tnl(:,:,2));
                Sp(:,:,k-1) = Nlperfusion(Tnl(:,:,2));
                Sc(:,:,k-1) = Tinffin*Sp(:,:,k-1);
                % Add in volumetric heating if that is the case
                Sc((jj-nCellsqy+1):jj,1:nCellsqx,k-1) = Sc((jj-nCellsqy+1):jj,1:nCellsqx,k-1)...
                    + Gvart(k-1)*ones(nCellsqy,nCellsqx);
                % Set the nonlinear temperature to check against
                Tnl(:,:,1) = Tnl(:,:,2);
                Tnl(:,:,2) = zeros(jj,ii);
                m = m+1;
            else
                % Convergence is reached, set new parameters
                kxy(:,:,k) = kxy(:,:,k-1);
                cpxy(:,:,k) = Cpxy(:,:,k-1);
                Sp(:,:,k) = Sp(:,:,k-1);
                Sc(:,:,k) = Tinffin*Sp(:,:,k-1);
                % Add in volumetric heating if that is the case
                Sc((jj-nCellsqy+1):jj,1:nCellsqx,k) = Sc((jj-nCellsqy+1):jj,1:nCellsqx,k)...
                    + Gvart(k)*ones(nCellsqy,nCellsqx);
                % Set temperature at current time step to converged
                % nonlinear temperature
                T(:,:,k) = Tnl(:,:,2);
                clear Tnl;
                % Store number of iterations that occured to find
                % convergence
                NlIterations(k) = m;
                % Reset loop variable to get out of nonlinear loop
                m = 100;
            end % end of inner (nl) if/else loop
        end % end of outer if else loop
    end % end of nl loop
end % end of kth time step loop


% *******************************************************
function a = EastSide(k,dmat)
[M,N] = size(k);
a = 2*k.*RotateRight(k)./( k*dmat+RotateRight(k)*dmat );
a(:,N) = 0;

% *******************************************************
function a = SouthSide(k,dmat)
[M,N] = size(k);
a = 2*k.*RotateUp(k)./( k*dmat+RotateUp(k)*dmat );
a(M,:) = 0;

% *******************************************************
function a = NorthSide(k,dmat)
[M,N] = size(k);
a = 2*k.*RotateDown(k)./( k*dmat+RotateDown(k)*dmat );
a(1,:) = 0;

% *******************************************************
function a = WestSide(k,dmat)
```

```
[M,N] = size(k);
a = 2*k.*RotateLeft(k)./( k*dmat+RotateLeft(k)*dmat );
a(:,1) = 0;

%*******************************************************
function u = RotateLeft(v)
[X,Y] = size(v);
u = [v(:,Y),v(:,1:Y-1)];

%*******************************************************
function u = RotateRight(v)
[X,Y] = size(v);
u = [v(:,2:Y),v(:,1)];

%*******************************************************
function u = RotateUp(v)
[X,Y] = size(v);
u = [v(2:X,:);v(1,:)];

%*******************************************************
function u = RotateDown(v)
[X,Y] = size(v);
u = [v(X,:);v(1:X-1,:)];

%*******************************************************
function T = tridiag(Told, A, b)
%************************************
%
%   Function m-file that solves tridiag matrix
%
%   Inputs: Told = vector of temperatures
%           going either vertically or
%           horizontally depending
%
%           A = [ncells x 3] - a matrix with
%           east west conductivities in the
%           case of a vertical solve or north
%           south conductivities in the case
%           of a horizontal solve
%
% ************************************

% --- Preprocessing
[M, N] = size(A);
A(:,N+1) = 0;
b = (Told + b)';

% --- Tridiagonal Routine from Patankar
P = zeros(1,N); Q = zeros(1,N);

P = -1*A(1,2)/A(1,1);
Q = b(1)/A(1,1);

for i = 2:M-1
    P(i) = -1*A(i,3) / (A(i,2) + A(i,1)*P(i-1));
    Q(i) = (b(i) - A(i,1)*Q(i-1)) / (A(i,2) + A(i,1)*P(i-1));
end

P(M) = 0;
Q(M) = (b(M) - A(M,2)*Q(M-1)) / (A(M,3) + A(M,2)*P(M-1));

T(M) = Q(M);

for i = 1:(M-1)
    T(M-i) = P(M-i)*T(M-i+1) + Q(M-i);
end
```

```
function [Dose,R] = CalcualteThermalDose(Tfield);
% ****************************************
%
%     The file that calculates the thermal dose
%
% ****************************************
global t RDOSE

% unzip rdose
Tref = RDOSE(1);

Tbar = Tref - Tfield;

R = Arr(Tfield,Tref);
% R = R.^Tbar;

Dose = trapz(t,R.^Tbar,3);

function R = Arr(T,Tref)
% ****************************************
%
%     Calculated the appropriate R value based
%     on a linear change over the critical value
%
%     Input: (Tfield,Tref)
%
%     T = an [M,N,P] Temperature matrix for which you want
%         to perform the R operation on.
%
%     Rhigh = high value R will take
%     Rlow = the low value R will take
%     eps = the temperature range over which the R function
%     is linear.  You should use hte same eps value for Arr
%     as DelArr.
%
% ****************************************
global RDOSE

Rhi = RDOSE(2); Rlo = RDOSE(3);
Reps = RDOSE(4);

[M,N,P] = size(T);
delR = Rhi - Rlo;

m = delR/2/Reps;

for k = 1:P
    for i = 1:M
        for j = 1:N
            if T(i,j,k) <= (Tref - Reps)
                R(i,j,k) = Rlo;
            elseif T(i,j,k) >= (Tref + Reps)
                R(i,j,k) = Rhi;
            else
                R(i,j,k) = +m*(T(i,j,k)-Tref+Reps) + Rlo;
            end
        end
    end
end

function D = CalculateDamageField(T)
% ****************************************
%
%     Calculates the damage field given a
%     Temperature field distribution in
%     x and y and time.
%
%     Input:  T(x,y,t) -> temp field [M,N,pmax]
%             t        -> time vector
%
%     Global Input:  Ea -> Activation engergy
%                    of experimental material
%                    A  -> Molecular Frequency
%                    factor of experimental
%                    material
%                    time -> time vector
%
%     Output:  Damage field          [M,N]
%
%****************************************
global Ea A R t

D = trapz(t,A*exp(-Ea/R./T),3);
%D = sum(A*exp(-Ea/R./(T+273.15)),3);
```

```
function [J,Ddiff,rms] = CalculateObjective(Dc,Dd,qest,t)
%*******************************************
%
%     Calculates the objective function.  This
%     function is minimized by the Conjugate
%     gradient method.  It is the sum of
%     squares between the calculated damage, Dc
%     and the desired damage, Dd.
%
%     Also calculates a statistical measure of
%     the computed error, rms error.
%
%     Input -> Dc, Dd, qest, t
%     Global Input -> ALPHA
%     Output -> J, Ddiff, rms
%
%*******************************************
global x y ALPHA

[M,N] = size(Dc);

Ddiff = Dc-Dd;
S = (Ddiff).^2;
IX = trapz(x,S,2);
J = trapz(y,IX,1);
J = J + ALPHA*trapz(t,qest.^2);

rms = 1/(N*M)*(sum(sum(Ddiff.^2,1)))^0.5;
```

```
function S = AdjointSourceDose(T,D,R)
% ************************************************
% Calculates the Source of the Adjoint problem for
% thermal dose optimization.
%
%    S = AdjointSourceDose(T,D)
%    T - T(x,y,t)
%    D - Difference between optimum and calculated
%        dose fields.
%
%    Global inputs:
%    RDOSE - contains parameters relative to the
%    thermal dose
%    t - vector of discretized time domain
% ************************************************
global RDOSE t

% unzip rdose
Tref = RDOSE(1);

[M,N,P] = size(T);
Tbar = Tref - T;

for i = 1:P
    DeltaDoseMatrix(:,:,i) = D;
end

% R = Arr(Tbar);
DeltaR = DelArr(R);

S = 2*(R.^Tbar).*(Tbar./R.*DeltaR - log(R)).*DeltaDoseMatrix;

S = flipdim(S,3);

% ***************************************
function deltaR = DelArr(R)
% ***************************************
%  Calculated the appropriate dR/dT value based
%  on a linear change over the critical value
%
%     Input: (T)
%
%     T = an [M,N,P] Temperature matrix for which you want
%         to perform the R operation on.  Must be Tbar.  (i.e.
%         Tbar = Tr - T).
%     Rhigh = high value R will take
%     Rlow = the low value R will take
%     eps = the temperature range over which the R function
%     is linear.  You should use hte same eps value for Arr
%     as DelArr.
% ***************************************
global RDOSE

Rhi = RDOSE(2); Rlo = RDOSE(3);
Tref = RDOSE(1); Reps = RDOSE(4);

[M,N,P] = size(R);
delR = Rhi - Rlo;

m = delR/2/Reps;

for k = 1:P
    for i = 1:M
        for j = 1:N
            if R(i,j,k) == Rhi
                deltaR(i,j,k) = 0;
            elseif R(i,j,k) == Rlo
                deltaR(i,j,k) = 0;
            else
                deltaR(i,j,k) = m;
            end
        end
    end
end

function Z = AdjointSourceDamage(T,D)
% ************************************************
% Calculates the "source" of the adjoint problem
% if the thermal damage is being optimized
%
%     Input:  Current time step temperature field, T,
%             (ncellsx,ncellsy,pmax)
%             Difference between desired and optimal
%             Damage field, D (ncellsx,ncellsy)
%
%     Global Input:  time, Ea, A (arrhenius properties)
%
%     Output:  Adjoint "volumetric" source term, Z,
%              returned in backwards time ready to be
%              solved by adjoint problem.
% ************************************************
global time Ea A R

K = Ea/R;

% --- Adjoint source term (x,y,time)
for i = 1:time(2)
    F1(:,:,i) = 2*A*K*exp(-K./T(:,:,i));
    F2(:,:,i) = T(:,:,i).^2;
    Z(:,:,i) = D.*F1(:,:,i)./F2(:,:,i);
end
```

```
Z = flipdim(Z,3);
```

```
function [T,Tout] = AdjointProblemFD(BcType, bcs, Z, kxy, cpxy,
Sp);
% ***********************************************
%    2D Finite Difference Code (Where the magic happens)
%
%    Description:  2D code for use with rectangular or
%          cylindrical coordinates.  Nearly explicit,
%          Crank-Nicolson and fully implicit time
%          discretization are possible.  Temperature
%          dependent thermal properties are included if
%          desired.
%
%     Global Inputs:
%             Ttable, KtableX, KtableY, Cptable, Rhoval
%
%             FIN ->  Fin type term, or internal convection
%
%     Inputs:  dx, dy = space discretization
%              D      = vector with number of nodes in x and y
%              time   = vector with lenght of time step and number
%                       of time steps
%              bcs    = vector of boundary conditions (see input file
%                        for specifics)
%              qt     = time dependent boundary condtion
%              To     = initial temperature field
%
%     Outputs:  T ->  Temperature field in time
% ***********************************************
global Ttable KtableX KtableY Cptable Rhoval
global FIN ORTH G B SENSOR
global Gvart
global dx dy D time

% --- "Unzip" discretization parameters
ii = D(1); jj = D(2);
dt = time(1);
pmax = time(2);

% --- "Unzip" boundary condition types
BCTYPExo = BcType(1);  BCTYPExL = BcType(2);
BCTYPEyo = BcType(3);  BCTYPEyH = BcType(4);

% --- "Unzip" boundary condtions parameters
Tyo = bcs(1,1); TyH = bcs(1,2); hyo = bcs(1,3); hyH = bcs(1,4); Tinfyo = bcs(1,5); TinfyH = bcs(1,6);
Txo = bcs(2,1); TxL = bcs(2,2); hxo = bcs(2,3); hxL = bcs(2,4); Tinfxo = bcs(2,5); TinfxL = bcs(2,6);
Sfin = bcs(1,7); Tinffin = bcs(1,8); To = bcs(1,9);
qxo = bcs(2,7); qxL = bcs(2,8); qyH = bcs(2,9);

% --- create key matrices
T = zeros(jj,ii,pmax);
Sc = zeros(jj,ii,pmax);
a = zeros(jj,ii);
%b = zeros(jj,ii,pmax);

% --- create volumetric heating for all other cases
Sc = Z;

% --- create vector of temperature dependent heating
% Sp = Sfin*ones(jj,ii);
% Sp(15:jj,1:15) = Sfin*2;

% *** SOLUTION ***

% --- flip properties since this problem is solved backwards
kxy = flipdim(kxy,3);
cpxy = flipdim(cpxy,3);
Sp = flipdim(Sp,3);

% --- Impose boundary condition
T(:,:,1) = To;

% --- Compute directional components
for i = 1:jj
    dmatx(i,:) = dx;
end

for i = 1:ii
    dmaty(:,i) = dy';
end

% --- Multiply coefs by normal areas and volumes
% ***************************************************
% G = 0 -> Rectangular, G = 1 -> Cylindrical
% ***************************************************
if G == 0
    areaE = dmaty;  areaW = areaE;
    areaN = dmatx;  areaS = areaN;
    V = dmaty.*dmatx;
else %Cylindrical case
    r = cumsum(dmatx,2);
    rw = RotateLeft(r); rw(:,1) = 0;
    areaE = 2*pi.*r.*dmaty;  areaW = 2*pi.*rw.*dmaty;
    rsq = r.^2;
    deltr = [r(:,1).^2 diff(rsq,1,2)];
    areaN = pi*deltr; areaS = areaN;
    V = dmaty.*areaN;
end

% Here is the part that I changed on June 22nd looking for
% the reason why I'm getting incorrect temperatures at the
% edges.  This should also save some computational time.
% The change has to do with edge nodes being only half
```

```
% thickness and the way the interface conductivity is
% calculated.
dmatx = dx(2); dmaty = dy(2);


for k = 2:pmax

        % --- prepare temperature dependent properties
        aE = EastSide(kxy(:,:,k-1),dmatx).*areaE;
        aW = WestSide(kxy(:,:,k-1),dmatx).*areaW;
        aS = SouthSide(kxy(:,:,k-1),dmaty).*areaS;
        aN = NorthSide(kxy(:,:,k-1),dmaty).*areaN;

        % --- Diagonal Coefs
        % Divide by delta t / 2 because of half step
        % solution technique
        a0 = Rhoval*cpxy(:,:,k-1).*V/(dt/2);

        % --- Compute initial Coefficients for diagonal and b vector
        %*********************************************************
        %
        %    hor / vert  -->  horizontal or vertical
        %    a for internal cells will include:
        %       - all constants that are multiplied by
        %         T(i,j)^(p+next) time step
        %    b for internal cells will include:
        %       - all constants added to RHS that don't
        %         get multiplied by a temperature
        %    c for internal cells will include:
        %       - all constants that are multipled by
        %         T(i,j)^(p) time step
        %
        %*****************************************************
        %Prepare x direction constants
        ahor = a0 + aE + aW + Sp(:,:,k-1).*V;
        chor = a0 - aS - aN;

        %Preparte y direction constants
        avert = a0 + aN + aS + Sp(:,:,k-1).*V;
        %Left out a0 b/c it gets multipled by T(p) not T(p+1/2)
        cvert = a0 -aE - aW;

        %Prepare shared components
        b = Sc(:,:,k).*V;

        % --- Create boundary conditions
        %*********************************************************
        %     Boundary conditons will created for each zone of
        %     cells that compose the boundary of the 2D area
        %     being modeled.  Start with central cells on each
        %     face and go clockwise around.
        %
        %     Next all the corners are done.
        %
        %     The first generation of this code will only allow
        %     for a time dependent boundary condtion at y = 0.
        %     This will only effect cells (1:nCellsx,1).
        %
        %     Note that the notation used in my notes is opposite
        %     that of MATLAB in y direction.  So that the boundary
        %     y = 0 is actually y = jj here.
        % *********************************************************
        midx = 2:jj-1;
        midy = 2:ii-1;

        %*****************
        %   SIDES
        %*****************

        % --- SET B.C's at x = 0,
        if BCTYPExo == 1
            % cells (1,2:jj-1)
            aE(midx,1) = 0; aN(midx,1) = 0; aS(midx,1) = 0; ahor(midx,1) = 1; avert(midx,1) = 1;
            b(midx,1) = Txo; chor(midx,1) = 0; cvert(midx,1) = 0;
        elseif BCTYPExo == 2
            % cells(1,2:jj-1)
            ahor(midx,1) = ahor(midx,1) + areaW(midx,1)*hxo;
            b(midx,1) = b(midx,1) + areaW(midx,1)*(hxo*Tinfxo + qxo);

            cvert(midx,1) = cvert(midx,1) - areaW(midx,1)*hxo;
        end

        % --- SET B.C's at y = H,
        if BCTYPEyH == 1
            % Cells(2:ii-1,jj)
            aW(1,midy) = 0; aE(1,midy) = 0; aS(1,midy) = 0; avert(1,midy) = 1; ahor(1,midy) = 1;
            cvert(1,midy) = 0; chor(1,midy) = 0;
            b(1,midy) = TyH;
        elseif BCTYPEyH == 2
            % Cells(2:ii-1,jj)
            avert(1,midy) = avert(1,midy) + areaN(1,midy)*hyH;
            b(1,midy) = b(1,midy) + areaN(1,midy)*(hyH*TinfyH + qyH);

            chor(1,midy) = chor(1,midy) - areaN(1,midy)*hyH;
        end

        % --- SET B.C's at x = L,
        if BCTYPExL == 1
            % Cells(ii,2:jj-1)
            aW(midx,ii) = 0; aN(midx,ii) = 0; aS(midx,ii) = 0; ahor(midx,ii) = 1; avert(midx,ii) = 1;
            chor(midx,ii) = 0; cvert(midx,ii) = 0; b(midx,ii) = TxL;
        elseif BCTYPExL == 2
```

```
    % Cells(ii,2:ii-1)
    ahor(midx,ii) = ahor(midx,ii) + areaE(midx,ii)*hxL;
    b(midx,ii) = b(midx,ii) + areaE(midx,ii)*(hxL*TinfxL + qxL);

    cvert(midx,ii) = cvert(midx,ii) - areaE(midx,ii)*hxL;
end

% --- SET B.C's at y = 0,
if BCTYPEyo == 1
    % Cells(2:jj-1,1)
    aW(jj,midy) = 0; aN(jj,midy) = 0; aE(jj,midy) = 0; avert(jj,midy) = 1; ahor(jj,midy) = 1;
    cvert(jj,midy) = 0; chor(jj,midy) = 0; b(jj,midy) = Tyo;
elseif BCTYPEyo == 2
    % Cells(2:ii-1,1)
    avert(jj,midy) = avert(jj,midy) + areaS(jj,midy)*hyo;
    b(jj,midy) = b(jj,midy) + areaS(jj,midy)*(hyo*Tinfyo);%<- q goes here;

    chor(jj,midy) = chor(jj,midy) - areaS(jj,midy)*hyo;
end

%*****************
%   CORNERS
%*****************

% --- SET B.C's at  Top left corner
if BCTYPExo == 1 & BCTYPEyH == 1
    aE(1,1) = 0; aS(1,1) = 0; ahor(1,1) = 1; avert(1,1) = 1;
    b(1,1) = (TyH+Txo)/2;
    chor(1,1) = 0; cvert(1,1) = 0;

elseif BCTYPExo == 2 & BCTYPEyH == 1
    % cell(1,jj) - Set to constant temperature at yH
    aE(1,1) = 0; aS(1,1) = 0; ahor(1,1) = 1; avert(1,1) = 1;
    b(1,1) = TyH;
    chor(1,1) = 0; cvert(1,1) = 0;

elseif BCTYPExo == 1 & BCTYPEyH == 2
    % cell(1,jj) - Set to constant temperature at yH
    aE(1,1) = 0; aS(1,1) = 0; ahor(1,1) = 1; avert(1,1) = 1;
    b(1,1) = Txo;
    chor(1,1) = 0; cvert(1,1) = 0;

elseif BCTYPExo == 2 & BCTYPEyH == 2
    ahor(1,1) = ahor(1,1) + areaW(1,1)*hxo;
    avert(1,1) = avert(1,1) + areaN(1,1)*hyH;

    b(1,1) = b(1,1) + areaW(1,1)*(hxo*Tinfxo + qxo)+ areaN(1,1)*(hyH*TinfyH + qyH);

    chor(1,1) = chor(1,1) - areaN(1,1)*hyH;
    cvert(1,1) = cvert(1,1) - areaW(1,1)*hxo;
end

% --- SET B.C's at Top right corner
if BCTYPExL == 1 & BCTYPEyH == 1
    aW(1,ii) = 0; aS(1,ii) = 0; avert(1,ii) = 1; ahor(1,ii) = 1;
    b(1,ii) = (TxL+TyH)/2;
    cvert(1,ii) = 0;    chor(1,ii) = 0;

elseif BCTYPExL == 1 & BCTYPEyH == 2
    aW(1,ii) = 0; aS(1,ii) = 0; avert(1,ii) = 1; ahor(1,ii) = 1;
    b(1,ii) = TxL;
    cvert(1,ii) = 0; chor(1,ii) = 0;

elseif BCTYPExL == 2 & BCTYPEyH == 1
    % cell(1,jj) - Artificially set biot number high on north face.
    aW(1,ii) = 0; aS(1,ii) = 0; avert(1,ii) = 1; ahor(1,ii) = 1;
    b(1,ii) = TyH;
    cvert(1,ii) = 0; chor(1,ii) = 0;

elseif BCTYPExo == 2 & BCTYPEyH == 2
    ahor(1,ii) = ahor(1,ii) + areaE(1,ii)*hxL;
    avert(1,ii) = avert(1,ii) + areaN(1,ii)*hyH;

    b(1,ii) = b(1,ii) + areaE(1,ii)*(hxL*TinfxL + qxL) + areaN(1,ii)*(hyH*TinfyH + qyH);

    chor(1,ii) = chor(1,ii) - areaN(1,ii)*hyH;
    cvert(1,ii) = cvert(1,ii) - areaE(1,ii)*hxL;
end

% --- SET B.C'S at Bottom right corner
if BCTYPExL == 1 & BCTYPEyo == 1
    aW(jj,ii) = 0; aN(jj,ii) = 0; ahor(jj,ii) = 1; avert(jj,ii) = 1;
    b(jj,ii) = (TxL+Tyo)/2;
    chor(jj,ii) = 0; cvert(jj,ii) = 0;

elseif BCTYPExL == 1 & BCTYPEyo == 2
    % cell(ii,jj) - Artificially set biot number high on north face.
    aW(jj,ii) = 0; aN(jj,ii) = 0; ahor(jj,ii) = 1; avert(jj,ii) = 1;
    b(jj,ii) = TxL;
    chor(jj,ii) = 0; cvert(jj,ii) = 0;

elseif BCTYPExL == 2 & BCTYPEyo == 1
    % cell(jj,ii) - Artificially set biot number high on south face.
    aW(jj,ii) = 0; aN(jj,ii) = 0; ahor(jj,ii) = 1; avert(jj,ii) = 1;
    b(jj,ii) = Tyo;
    chor(jj,ii) = 0; cvert(jj,ii) = 0;

elseif BCTYPExL == 2 & BCTYPEyo == 2
    ahor(jj,ii) = ahor(jj,ii) + areaE(jj,ii)*hxL;
    avert(jj,ii) = avert(jj,ii) + areaS(jj,ii)*hyo;

    b(jj,ii) = b(jj,ii) + areaE(jj,ii)*(hxL*TinfxL + qxL) + areaS(jj,ii)*(hyo*Tinfyo);% <- don't forget qyo
```

```
            chor(jj,ii) = chor(jj,ii) - areaS(jj,ii)*hyo;
            cvert(jj,ii) = cvert(jj,ii) - areaE(jj,ii)*hxL;
        end

        % --- SET B.C's at Bottom left corner
        if BCTYPExo == 1 & BCTYPEyo == 1
            aE(jj,1) = 0; aN(jj,1) = 0; ahor(jj,1) = 1; avert(jj,1) = 1;
            b(jj,1) = (Txo+Tyo)/2;
            chor(jj,1) = 0; cvert(jj,1) = 0;

        elseif BCTYPExo == 2 & BCTYPEyo == 1
            aE(jj,1) = 0; aN(jj,1) = 0; ahor(jj,1) = 1; avert(jj,1) = 1;
            b(jj,1) = Tyo;
            chor(jj,1) = 0; cvert(jj,1) = 0;

        elseif BCTYPExo == 1 & BCTYPEyo == 2
            aE(jj,1) = 0; aN(jj,1) = 0; ahor(jj,1) = 1; avert(jj,1) = 1;
            b(jj,1) = Txo;
            chor(jj,1) = 0; cvert(jj,1) = 0;

        elseif BCTYPExo == 2 & BCTYPEyo == 2
            ahor(jj,1) = ahor(jj,1) + areaW(jj,1)*hxo;
            avert(jj,1) = avert(jj,1) + areaS(jj,1)*hyo;

            b(jj,1) = b(jj,1) + areaW(jj,1)*(hxo*Tinfxo + qxo) + areaS(jj,1)*(hyo*Tinfyo); % <- don't forget qyo

            chor(jj,1) = chor(jj,1) - areaS(jj,1)*hyo;
            cvert(jj,1) = cvert(jj,1) - areaW(jj,1)*hxo;
        end

        %*******************************************
        %
        %   Enter Solver!
        %
        %*******************************************
        Thalf = zeros(jj,ii);
        bt = zeros(jj,ii);

        % X direction sweep

        for hor = 1:jj
            if hor == 1
                bhor(hor,:) = b(hor,:) + aS(hor,:).*T(hor+1,:,k-1);
            elseif (hor > 1) & (hor < jj)
                bhor(hor,:) = b(hor,:) + aS(hor,:).*T(hor+1,:,k-1) + aN(hor,:).*T(hor-1,:,k-1);
            elseif hor == jj
                bhor(hor,:) = b(hor,:) + aN(hor,:).*T(hor-1,:,k-1);
            end

            % ***********************************************
            % Tridiagonal Solver - Solve horizontally T(p+1/2)
            % ***********************************************
            A = zeros(ii,3);
            % --- Boundary conditions
            A(1,[1:2]) = [ahor(hor,1) -aE(hor,1)];
            A(ii, [2:3]) = [-aW(hor,ii) ahor(hor,ii)];
            % --- Matrix body
            A(2:(ii-1),1) = -aW(hor,2:(ii-1))';
            A(2:(ii-1),2) = ahor(hor,2:(ii-1))';
            A(2:(ii-1),3) = -aE(hor,2:(ii-1))';
            % --- Compute solution
            Thalf(hor,:) = tridiag(T(hor,:,k-1).*chor(hor,:),A,bhor(hor,:));
            clear A;
        end

        clear bhor;
        Tnl(:,:,2) = zeros(jj,ii);
        % Y direction sweep
        for vert = ii:-1:1;
            if vert == 1
                bvert(:,vert) = b(:,vert) + aE(:,vert).*Thalf(:,vert+1);
            elseif (vert > 1) & (vert < ii)
                bvert(:,vert) = b(:,vert) + aE(:,vert).*Thalf(:,vert+1) + aW(:,vert).*Thalf(:,vert-1);
            elseif vert == ii
                bvert(:,vert) = b(:,vert) + aW(:,vert).*Thalf(:,vert-1);
            end

            % ***********************************************
            % Tridiagonal Solver - Solve vertically for T(p+1)
            % ***********************************************
            A = zeros(jj,3);
            % --- Boundary conditions
            A(1,[1:2]) = [avert(1,vert) -aS(1,vert)];
            A(jj, [2:3]) = [-aN(jj,vert) avert(jj,vert)];
            % --- Matrix body
            A(2:(jj-1),1) = -aN(2:(jj-1),vert);
            A(2:(jj-1),2) = avert(2:(jj-1),vert);
            A(2:(jj-1),3) = -aS(2:(jj-1),vert);
            % --- Compute solution
            Tvert = tridiag((Thalf(:,vert).*cvert(:,vert))',A,bvert(:,vert)');
            Tnl(:,vert,2) = Tvert';
            clear A;
        end

        clear bvert;bt;b;
        clear a0;aE;aW;aS;aN;ahor;avert;chor;cvert;
        T(:,:,k) = Tnl(:,:,2);
        clear Tnl;
end % end of kth time step loop

T = flipdim(T,3);
```

```
% ******************************************************
function a = EastSide(k,dmat)
[M,N] = size(k);
a = 2*k.*RotateRight(k)./( k*dmat+RotateRight(k)*dmat );
a(:,N) = 0;

% ******************************************************
function a = SouthSide(k,dmat)
[M,N] = size(k);
a = 2*k.*RotateUp(k)./( k*dmat+RotateUp(k)*dmat );
a(M,:) = 0;

% ******************************************************
function a = NorthSide(k,dmat)
[M,N] = size(k);
a = 2*k.*RotateDown(k)./( k*dmat+RotateDown(k)*dmat );
a(1,:) = 0;

% ******************************************************
function a = WestSide(k,dmat)
[M,N] = size(k);
a = 2*k.*RotateLeft(k)./( k*dmat+RotateLeft(k)*dmat );
a(:,1) = 0;

%*******************************************************
function u = RotateLeft(v)
[X,Y] = size(v);
u = [v(:,Y),v(:,1:Y-1)];

%*******************************************************
function u = RotateRight(v)
[X,Y] = size(v);
u = [v(:,2:Y),v(:,1)];

%*******************************************************
function u = RotateUp(v)
[X,Y] = size(v);
u = [v(2:X,:);v(1,:)];

%*******************************************************
function u = RotateDown(v)
[X,Y] = size(v);
u = [v(X,:);v(1:X-1,:)];

%*******************************************************
function T = tridiag(Told, A, b)
%************************************
%
%   Function m-file that solves tridiag matrix
%
%   Inputs: Told = vector of temperatures
%           going either vertically or
%           horizontally depending
%
%           A = [ncells x 3] - a matrix with
%           east west conductivities in the
%           case of a vertical solve or north
%           south conductivities in the case
%           of a horizontal solve
%
% ************************************

% --- Preprocessing
[M, N] = size(A);
A(:,N+1) = 0;
b = (Told + b)';

% --- Tridiagonal Routine from Patankar
P = zeros(1,N); Q = zeros(1,N);

P = -1*A(1,2)/A(1,1);
Q = b(1)/A(1,1);

for i = 2:M-1
    P(i) = -1*A(i,3) / (A(i,2) + A(i,1)*P(i-1));
    Q(i) = (b(i) - A(i,1)*Q(i-1)) / (A(i,2) + A(i,1)*P(i-1));
end

P(M) = 0;
Q(M) = (b(M) - A(M,2)*Q(M-1)) / (A(M,3) + A(M,2)*P(M-1));

T(M) = Q(M);

for i = 1:(M-1)
    T(M-i) = P(M-i)*T(M-i+1) + Q(M-i);
end
```

```
function [gradout,tparam] = FunctionalGradient(P,q);
%**************************************
%
%     Calculates the functional gradient of
%     the control problem.
%
%     Input: Adjoint variable at location
%     where control parameter operates (0<x<rheater)
%
%     Global Inputs:  x
%
%     Output:  functional gradient at current
%     time step
%
%**************************************
global x y ALPHA NPARAM t

% --- Size the incoming vector
[L,M,N] = size(P);

% --- Prepare x varible for integration
xg = x(1:M);

if L > 1
    % --- Prepare y varible for integration
    yg = y((length(y)-L+1):length(y));
    grady = trapz(yg,P,1);

    % --- Integrate P over the x domain
    grad = trapz(xg,grady,2);
else
    % --- Integrate P over the x domain
    grad = trapz(xg,P,2);
end

% --- Flip grad to be [Nx1] vector
ind = 1:N;
gradout(ind,1) = grad(:,:,ind);
gradout = gradout + 2*ALPHA*q;
tparam = 0;

if NPARAM > 0
    % --- Paramiterize heat flux option
    tparam = t(1):t(length(t))/(NPARAM-1):t(length(t));
    PPgrad = spline(t,gradout);
    gradout = ppval(PPgrad,tparam)';
%     for i = 1:NPARAM
%         if gradout(i) < 0
%             gradout(i) = 0
%         end
%     end
%     clear gradout;
%     gradout = interp1(tparam,ParamGradout,t,'linear')';
end

function [beta,grad] = ConjugationParameter(grad,count);
%**************************************************
% Calculates conjugation parameter, Beta
%
% Input:  previous two functional gradient vectors
%
% Output:  Beta
%
% March, 2004
%**************************************************

[M,N] = size(grad);
%grad(M,:) = [];

if count == 2;
    % --- if its the first time through, beta = 0
    % part of the rules.
    beta = 0;
else
    up = sum((grad(:,2) - grad(:,1)).*grad(:,2));
    down = sum(grad(:,1).^2);
    beta = up/down;
end

grad(:,1) = grad(:,2); grad(:,2) = zeros(M,1);

function decent = DescentDirection(beta,grad,descent_prev);
%**************************************************
% Calculates the decent direction
%
% Input:
%
%     beta = conjugate coefficient from previous
%     iteration
%
%     grad_S = the gradient of the objective function
%     from the current time step
%
%     decent_prev = the decent direction from the
%     old time step
%
% Output:
%
%     New decent direction, vector in time
%
% March, 2004
%**************************************************
```

```
decent = -grad + beta*descent_prev;
```

```
function gamma = DescentParameterDose(T,delT,D,R,qest,delqest);
% ********************************************************
%     Call Format:
%     gamma = DescentParameter(T,delT,D,qest,delqest,t);
%
%     Calculates decent parameter
%
%     Inputs:  T, temperature field current time
%              step, [nCellsx,nCellsy,pmax]
%
%              delT, variation temperature field
%              [nCellsx,nCellsy,pmax]
%
%              D, difference between optimal and
%              current calculated dose fields
%              [nCellsx,nCellsy]
%
%              qest, current control parameter estimate
%
%              delqest, variation of qest
%
%              t, time vector
%
%      Global Inputs:  y, x, time,
%              ALPHA -> stabilization parameter
%              RDOSE -> vector containing R parameters
%
%      Output: gamma, decent parameter
%
% ********************************************************
global t RDOSE x y

Tref = RDOSE(1);

[M,N,P] = size(T);
Tbar = Tref - T;

DeltaR = DelArr(R);

Psi = trapz(t,((R.^Tbar).*delT.*(Tbar./R.*DeltaR - log(R))),3);

numerat = trapz(y,trapz(x,(D.*Psi),2));
denom = trapz(y,trapz(x,Psi.^2,2));

gamma = -1*numerat/denom;


function deltaR = DelArr(R)
% ***************************************
%
%  Calculated the appropriate dR/dT value based
%  on a linear change over the critical value
%
%     Input: (T)
%
%     T = an [M,N,P] Temperature matrix for which you want
%         to perform the R operation on.  Must be Tbar.  (i.e.
%         Tbar = Tr - T).
%     Rhigh = high value R will take
%     Rlow = the low value R will take
%     eps = the temperature range over which the R function
%     is linear.  You should use hte same eps value for Arr
%     as DelArr.
%
% ***************************************
global RDOSE

Rhi = RDOSE(2); Rlo = RDOSE(3);
Tref = RDOSE(1); Reps = RDOSE(4);

[M,N,P] = size(R);
delR = Rhi - Rlo;

m = delR/2/Reps;

for k = 1:P
    for i = 1:M
        for j = 1:N
            if R(i,j,k) == Rhi
                deltaR(i,j,k) = 0;
            elseif R(i,j,k) == Rlo
                deltaR(i,j,k) = 0;
            else
                deltaR(i,j,k) = m;
            end
        end
    end
end


function gamma = DescentParameterDamage(T,delT,D,qest,delqest);
% *****************************************
%     Call Format:
%     gamma = DescentParameter(T,delT,D,qest,delqest,t);
%
%     Calculates decent parameter
%
%     Inputs:  Tq, temperature field current time
%              step, [nCellsx,nCellsy,pmax]
%
%              delT, variation temperature field
%              [nCellsx,nCellsy,pmax]
```

```
%
%             D, difference between optimal and
%             current calculated damage fields
%             [nCellsx,nCellsy]
%
%             qest, current control parameter estimate
%
%             delqest, variation of qest
%
%             t, time vector
%
%      Global Inputs:  y, x, time,
%             ALPHA -> stabilization parameter
%
%      Output: gamma, decent parameter
%
% ****************************************
global x y t global Ea R A ALPHA

K = Ea/R;

% --- calculate the variation of damage coef
F0 = A*K*exp(-K./T); F01 = delT; F02 = T.^2; delOmega =
trapz(t,F0.*F01./F02,3);

% --- Break down the function into smaller parts
F1 = trapz(y,trapz(x,D.*delOmega,2)); F2 = trapz(t,qest.*delqest);
F3 = trapz(y,trapz(x,delOmega.^2,2)); F4 = trapz(t,delqest.^2);

gamma = -(F1+ALPHA*F2)/(F3+ALPHA*F4);
```

```
function Qnew = Update(q_old,gamma,decent)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This file serves to calculate the next iteration parameter vector
%
%
% Function calls P, beta and decent and returns step
%
%     q_old = current function estimate over tdomain
%
%     beta = search step size
%
%     decent = the decent direction
%
%     q_new = new functional form
%
%
% Scott Gayzik
% July, 2003
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global NPARAM

%extremely complex code
Qnew = q_old + gamma*decent;

% % --- Modify to avoid negative heat flux
% if NPARAM > 0
%     if Qnew

function Fo = CalculateFourierNo(k,cp)
% ************************************************************
%
%     Function Call:
%     Fo = CalculateFourierNo(max(max(kT)),min(min(Cp)))
%
%     Calculates the largest fourier number
%
%     Inputs:
%     max thermal conductivity
%     min heat capacity
%
%     Global Inputs:
%     dx, dt
%
%     Outputs:
%     Fourier no.
%
% ************************************************************
global dx time

alpha = k/cp; Fo = alpha*time(1)/dx(1)^2;
```

```
function [MaxErrors,FinalDamageError,EnergyError,qRMSError] =...
    DamageStats(DeltaDamage,DamageDesired,Optimalqt,qest);
% ************************************************
%
%     Function Call:
%       [DamageError,FinalDamageError,DamageRMSError,EnergyError,qRMSError] =...
%        DamageStats(DeltaDamage,DamageDesired,Optimalqt,qest,t);
%
%     Inputs:  DeltaDamage, DamageDesired, Optimalqt, qest, t
%
%     Outputs:  MaxErrors - Represents the maximum percent error
%               area at each spacial location after the optimization
%               routine.
%
%               FinalDamageError - max percent error for final iteration
%               only.
%
%               DamageRMSError - RMS error of thermal dose each iteration
%
%               EnergyError - Percent error of the integrated incident
%               heat flux (estimated and actual)
%
%               qRMSError - RMS error of heat flux function each iteration
%
% ************************************************
global t

[M,N] = size(DeltaDamage);
[Q,R] = size(qest);

% --- Percent error and the RMS error of dose
FinalDamageError = (abs(DeltaDamage./DamageDesired))*100;

[MaxDamageErrorY,I] = max(FinalDamageError,[],1);

MaxErrors = [MaxDamageErrorY;I];

% --- Stats about the heat flux estimation
prescribed = trapz(t,Optimalqt);
for i = 2:R+1;
    % --- RMS Error of the estimated heat flux function
    qRMSError(i-1) = 1/Q*(sum((Optimalqt - qest(:,i-1)).^2))^.5;
    % --- Integrated Error
    EnergyError(i-1) = ((trapz(t,qest(:,i-1)) - prescribed) / prescribed).^2 *100;
end

function
DamagePlots(DE,DD,Qgen,Qest,Tgen,TQest,MXERS,FDE,DRMS,EE,QRMS);
% ********************************************************
%
%     File that plots results of thermal dose optimization
%     routine.
%
%     Inputs:
%     DE - Estimated damage field
%     DD - Desired damage field
%     t - time vector
%     Qgen - heat flux used to get DD
%     Qest - Heat flux program estimated
%     x - vector of all nodes in x
%     Tgen - Temperature field resulting from Qgen
%     TQest - Temperautre field the program estimated
%     MXERS - Maximum % error at each x and y and where
%     FDE - Final percent error of thermal damage from DD
%     DRMS - Thermal damage rms error
%     EE - Engergy density error
%     QRMS - heat flux rms error
%
% ********************************************************
global x t

figure
% --- Compares the optimal and desired thermal damage field
subplot(2,2,1); mesh title('Optimal vs. Recovered Thermal Damage
Coefficient') legend('Optimal','Recovered') grid

figure plot(t,Qgen,t,Qest) title('Actual vs. Estimated Heat Flux')
legend('Actual','Estimated') grid

figure [M,N] = size(Tgen); mid = floor(M/2);
plot(x,Tgen(1,:),'b',x,TQest(1,:),'bo',...
    x,Tgen(mid,:),'r',x,TQest(mid,:),'ro',x,Tgen(M,:),'g',x,TQest(M,:),'go');
title('Temperature field resulting from Optimal heat flux and
Estimated Heat Flux') legend('Toptimal, t = 0','Trecovered, t =
0','Toptimal, t = tfinal/2','Trecovered, t = tfinal/2',...
    'Toptimal, t = tfinal','Trecovered, t = tfinal')
grid

figure subplot(2,2,1);plot(x,FDE) title('Final percent error of
thermal damage vs. Position')
xlabel('x'); ylabel('Thermal Dose Error (%)')
grid subplot(2,2,2);plot(2:length(DRMS),DRMS(2:length(DRMS)))
title('Damage RMS error vs. Iteration')
xlabel('Iteration'),ylabel('RMS error') grid
subplot(2,2,3);plot(2:length(EE),EE(2:length(EE)))
title('Normalized energy density error vs. Iteration')
xlabel('Iteration'),ylabel('Energy density error, (%)')
grid subplot(2,2,4);plot(2:length(QRMS),QRMS(2:length(QRMS)))
title('Heat flux RMS error vs. Iteration')
xlabel('Iteration'),ylabel('RMS error') grid
```

```
function k = NlConductivity(T)
% ------------------------------
%
%   Function that calculates nonlinear
%   thermal conductivity, k(T)
%
%   Input, current temperature vector, T
%   Out, thermal conductivity vector based on T
% ------------------------------
global Ttable KtableX DEG nlk exp dx

[P,Q] = size(T);

if nlk == 0
    if exp == 0
        k = KtableX*ones(size(T));
    else

%         k = KtableX*ones(size(T));
%         % find how many cells needed to model edge
%         % of cylinder
%         edge = 0.0035;
%         N = floor(edge/dx(2));
%         if (N*dx(2) + dx(1)) <= edge
%             N = N+1;
%         end
%         % Property values of lexan
%         klexan = 0.24; % W/m/K
%        % k(:,(Q-N+1):Q) = klexan*ones(P,(1:Q-N+1));
%         k(:,(Q-N+1):Q) = klexan;
    end
else
%--- Creates polynomial and evaluates polynomial at T
        k = 1000000*ones(size(T));
        k(:,1) = KtableX(1)*ones(15,1);
        k(:,2:5) = KtableX(2)*ones(15,4);
        k(:,6:15) = KtableX(3)*ones(15,10);

%     P = polyfit(Ttable,KtableX,DEG);
%     k = polyval(P,T);
% --- Can be modified to use a direct formula
%k = 1 +4*T.^2;
end

function Cp = NlHeatCap(T)
% ------------------------------
%
%   Function that calculates nonlinear
%   heat capacity, Cp(T)
%
%   Input, current temperature vector, T
%   Out, heat capacity vector based on T
% ------------------------------
global Ttable Cptable DEG nlCp exp dx

[P,Q] = size(T);

if nlCp == 0
    if exp == 0
        Cp = Cptable*ones(size(T));
    else
        Cp = Cptable*ones(size(T));
        % find how many cells needed to model edge
        % of cylinder
        edge = 0.0035;
        N = floor(edge/dx(2));
        if (N*dx(2) + dx(1)) <= edge
            N = N+1;
        end
        % Property values of lexan
        Cplexan = 1250; % W/m/K
       % k(:,(Q-N+1):Q) = klexan*ones(P,(1:Q-N+1));
        Cp(:,(Q-N+1):Q) = Cplexan;
    end
else
%--- Creates polynomial and evaluates polynomial at T
    P = polyfit(Ttable,Cptable,DEG);
    Cp = polyval(P,T);
% ---- Can be modified to be a direct formula
%     Cp = 1./(1+2.*T);
end

function Sp = NlPerfusion(T)
% ------------------------------
%
%   Function that calculates nonlinear
%   thermal conductivity, k(T)
%
%   Input, current temperature vector, T
%   Out, thermal conductivity vector based on T
% ------------------------------
global Ttable Sptable DEG nlSp exp dx heater dy

[R,Q] = size(T);

if nlSp == 0
    if exp == 0
        Sp = Sptable(1)*ones(size(T));
    else
        Sp = zeros(size(T));
        %zone 3, top left
        Sp(1:floor(R/2),1:heater(1)) = Sptable(1);
```

```
        %zone 2, mid left
        Sp(floor(R/2+1):floor(3*R/4),1:heater(1)) = Sptable(2);
        %zone 1, bot left
        Sp(floor(3*R/4+1):R,1:heater(1)) = Sptable(3);
        %zone 4, right side
        Sp(1:R,(heater(1)+1):Q) = Sptable(4);
    end
else
%--- Creates polynomial and evaluates polynomial at T
    Sp = zeros(size(T));


    PP = spline(Ttable,Sptable);
    Spfit = PPval(PP,T-273.15);
%     P = polyfit(Ttable,Sptable,DEG);
%     Sp = polyval(P,T-273.15);
    Sp(:,1:5) = Spfit*ones(15,5);
%     disp('I''m checking nl w')
% --- Can be modified to use a direct formula
%k = 1 +4*T.^2;
end
```

```
function [cells,length,error] = OptimumCv close all
%Tries to detect what the best cv numbers would be

Ly = 0.0268;
Lx = 0.075;
rheat = 0.023;

Locell = 5;
Hicell = 200;

L = [Ly Lx rheat];

for i = Locell:Hicell
    heater = geodisc(L,[i,i]);
    % gives resulting discretized length
    length(i) = heater(2);
    % gives percent error of this lenght
    error(i) = heater(3);
end

% for x axis
cells = 1:Hicell;

plot(cells,error,'ko-')
axis([5 100 0 25])
xlabel('No. of Control Volumes')
ylabel('% Error from actual heater size')
legend('% Error')
grid
% figure
% plot(cells,(length - rheat),'+-')
% grid
```

Vita

F. Scott Gayzik was born in Middlesex, New Jersey on August 29^{th} nineteen hundred and seventy nine to his loving parents, Albert and Frances. He also has an older brother Adam. He attended Middlesex High School and graduated in 1997. Following his wise brother he decided to attend college at Virginia Tech.

While at Virginia Tech Scott studied Mechanical Engineering and took advantage of any opportunity afforded to him to expand his horizons. In his 4^{th} year he was granted a year study in Zaragoza Spain where he honed his skill in Spanish, earned a minor in the language and became an authentic Maño along the way.

After a filthy year living in the now-infamous Center Street apartments, Scott graduated without incident and went on a soul-searching voyage that would take him half way across the world, from the mountains of central Italy to the beautiful and desolate desserts of Mexico. In the end he found two simple truths: He wasn't about to get a job, and surfing is gnarly. Scott enrolled in graduate school at Virginia Tech, went back to Europe once more, and plans on pursing a Ph.D. in the Virginia Tech/Wake Forest School of Biomedical Engineering.

He will travel for long spans of time again. It's only a matter of time.

Permanent Address: 710 Burruss Drive
Blacksburg, VA 24060

This thesis was typeset with LaTeX 2_ε [1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this thesis were written by Greg Walker, Department of Mechanical Engineering, Virginia Tech.