Summer 2016

Thermal analysis in a triple-layered skin structure with embedded vasculature, tumor, and gold nanoshells

Casey O. Orndorff

Follow this and additional works at: https://digitalcommons.latech.edu/dissertations

Part of the Other Applied Mathematics Commons
THERMAL ANALYSIS IN A TRIPLE-LAYERED SKIN STRUCTURE
WITH EMBEDDED VASCULATURE, TUMOR,
AND GOLD NANOSHELLS

by
Casey O. Orndorff, B.S. M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

August 2016
We hereby recommend that the dissertation prepared under our supervision by Casey O'Neal Orndorff entitled Thermal Analysis In A Triple-Layered Skin Structure With Embedded Vasculature, Tumor, And Gold Nanoshells be accepted in partial fulfillment of the requirements for the Degree of Doctorate of Philosophy: Computational Analysis and Modeling.

Supervisor of Dissertation Research

Head of Department
Computational Analysis and Modeling

Advisory Committee

Approved:
Director of Graduate Studies

Approved:
Dean of the Graduate School

Dean of the College
ABSTRACT

In hyperthermia skin cancer treatment, the objective is to control laser heating of the tumor (target temperatures of 42-46 °C) so that the temperatures of the normal tissue surrounding the tumor remains low enough not to damage the normal tissue. However, obtaining accurate temperature distributions in living tissue related to hyperthermia skin cancer treatment without using an intruding sensor is a challenge. The objective of this dissertation research is to develop a mathematical model that can accurately predict the temperature distribution in the tumor region and surrounding normal tissue induced by laser irradiation. The model is based on a modified Pennes’ equation for the bioheat transfer in a 3-D triple-layered skin structure embedded with a vascular countercurrent network and a tumor appearing in the subcutaneous region. The vascular network is designed based on the constructal theory of multiscale tree-shaped heat exchangers. The tumor is injected with gold nanoshells in order to be heated quickly. The proposed model is implemented numerically using a stable finite-difference scheme. To determine the laser intensity so that an optimal temperature distribution can be obtained, we pre-specify the temperature elevations to be obtained at the center of the tumor and on some locations on the perimeter of the skin’s surface. Using the least squares method, we obtain the optimal laser power and develop a computational procedure to obtain the temperature distribution.
The method was tested in a 3-D triple-layered skin structure embedded with a vascular countercurrent network and a tumor appearing in the subcutaneous region. Gold nanoshells are assumed to have been injected into the central region of the tumor. The tumor region that has the gold nanoshells has $7 \times 10^9$ particles/cm$^3$ for each voxel of $0.01 \text{ cm} \times 0.01 \text{ cm} \times 0.001 \text{ cm}$. The temperature is elevated by means of laser irradiation. The results show that the nanoshells have an effect on the tumor by heating the entire tumor to above 42 °C while not overheating the surrounding tissue. In comparison, results show that without nanoshells in the tumor region the tumor does heat up along its central axis; however, the perimeter of the tumor fails to reach 42 °C while the top of the skin reaches undesirable temperature levels due to the laser intensity required to heat the tumor. Such research may provide a useful tool for optimizing laser irradiation to kill the tumor while keeping the damage to the surrounding healthy tissue to a minimum ($\leq 42$ °C) during the hyperthermia cancer treatment.
APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that “proper request” consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author [Signature]

Date 07/19/2016
DEDICATION

I dedicate this work to my late mother, Marna Kay Orndorff, who lost her battle with lung cancer in the early years of my life. She fought for me when I was scrutinized for my disability, and now I fight for her by furthering research against cancer.
## TABLE OF CONTENTS

ABSTRACT ............................................................................................................................ iii

LIST OF TABLES .................................................................................................................. ix

LIST OF FIGURES ................................................................................................................ x

ACKNOWLEDGMENTS ....................................................................................................... xv

CHAPTER 1 INTRODUCTION ............................................................................................ 1

  1.1 General Overview .................................................................................................... 1

  1.2 Research Objectives ............................................................................................. 3

CHAPTER 2 LITERATURE REVIEW AND PREVIOUS WORK ...................................... 5

  2.1 Hierarchical Branching Network and Constructal Theory of Multi-scale
     Tree-shape Heat Exchangers ................................................................................ 5

  2.2 Nanoshell Heat Generation ................................................................................... 11

  2.3 Previous work related to the Bio-Heat Transfer Equation ............................. 21

CHAPTER 3 MATHEMATICAL MODEL ........................................................................ 29

  3.1 Network Design .................................................................................................... 29

  3.2 Bioheat Transfer Equation .................................................................................... 32

  3.3 Heat Source ........................................................................................................... 34

  3.4 Nanoshell Heat Generation ................................................................................... 35

  3.5 Boundary, Initial, and Interfacial Conditions ................................................... 38

CHAPTER 4 NUMERICAL METHOD ............................................................................ 41
4.1 Finite Difference Method ................................................................. 41
4.2 Least Squares Method ................................................................. 43
4.3 Computational Procedure .......................................................... 45

CHAPTER 5 NUMERICAL EXAMPLE .................................................. 49
  5.1 Example Description ................................................................. 49
  5.2 Results ....................................................................................... 50
  5.3 Discussion .................................................................................. 87

CHAPTER 6 CONCLUSIONS AND FUTURE WORK ............................ 94

APPENDIX A NOMENCLATURE ......................................................... 96
APPENDIX B SOURCE CODE .......................................................... 103
BIBLIOGRAPHY .............................................................................. 182
LIST OF TABLES

Table 3.1: Parameters for a 3-D skin structure............................................................ 33

Table 4.1: Parameters used in Computations............................................................... 44

Table 5.1: Parameters used in obtaining local heat generation spectrum being generated by the gold nanoshells ................................................................. 50
LIST OF FIGURES

Figure 2.1: A proposed geometric model used consists of seven levels of a regular, branching network of arteries (red) and veins (blue) within a cubic space ................................................................. 7

Figure 2.2: A figure demonstrating the location of a subset of the vessel network within the cubic space. A total of 64 such sets are used in the model, where the subsets are regularly and uniformly spaced throughout the cubic space [1] .................................................................................................................. 7

Figure 2.3: Models of the peripheral artery network embedding into the cellular tissue, (a) four-fold model and (b) a more realistic dichotomic artery tree uniformly embedded into a cubic tissue domain. When tested, both models had the same impact on the tissue domain [2] .......... 10

Figure 2.4: Illustration of light propagation within PDMS slab ......................... 16

Figure 2.5: Configuration of nanoshell-embedded tissue thermal analysis. The region of interest occupied the nanoshells is a pea-sized tumor ............. 19

Figure 2.6: Configuration of a 3D triple-layered rectangular skin structure with an embedded three-level countercurrent network in [5] ....................... 25

Figure 2.7: A 3D triple-layered skin structure embedded with countercurrent vasculature [6] .................................................................................................................. 26

Figure 2.8: A 3D triple-layered rectangular skin structure with embedded seven-level countercurrent network [7]. The red vessels denote the arterial network while the venous network is denoted as blue ....................... 28

Figure 3.1: A 3-D triple-layered skin structure embedded with seven levels of countercurrent vasculature network with an embedded tumor ........ 30

Figure 4.1: Flow chart for the computational procedure ...................................... 48

Figure 5.1: Number of iterations versus (a) laser intensity ($P_0$) and (b) least squares sum ($S(P_0)$) .................................................................................................................. 52
Figure 5.2: Contours of temperature distribution in the yz-plane at $x = 0.54$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$, while (e) represents the plane where the results were observed. Here, levels 2-4 of the arterial system are visible.

Figure 5.3: Contours of temperature distribution in the yz-plane at $x = 0.81$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 1, 5, 6, 7 of the arterial system and the center of the tumor are visible.

Figure 5.4: Contours of temperature distribution in the yz-plane at $x = 1.08$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, level 1 of the arterial system is visible.

Figure 5.5: Contours of the temperature distribution in the xz-plane at $y = 0.54$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.

Figure 5.6: Contours of the temperature distribution in the xz-plane at $y = 0.81$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 1 and 2 of the arterial system and the center of the tumor are visible.

Figure 5.7: Contours of the temperature distribution in the xz plane at $y = 1.08$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.

Figure 5.8: Number of iterations versus (a) laser intensity ($P_0$) and (b) least squares sum ($S(P_0)$).

Figure 5.9: Contours of temperature distribution in the yz-plane at $x = 0.54$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 2-4 of the arterial system are visible.

Figure 5.10: Contours of temperature distribution in the yz-plane at $x = 0.81$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 1, 5, 6, 7 of the arterial system and the center of the tumor are visible.
Figure 5.11: Contours of temperature distribution in the yz-plane at $x = 1.08$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, level 1 of the arterial system is visible............................................. 63

Figure 5.12: Contours of the temperature distribution in the xz-plane at $y = 0.54$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible...................... 64

Figure 5.13: Contours of the temperature distribution in the xz-plane at $y = 0.81$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 1 and 2 of the arterial system and the center of the tumor are visible.............................................................................................. 65

Figure 5.14: Contours of the temperature distribution in the xz plane at $y = 1.08$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.................... 66

Figure 5.15: Contours of temperature distribution in the yz-plane at $x = 0.54$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 2-4 of the arterial system are visible............................................. 68

Figure 5.16: Contours of temperature distribution in the yz-plane at $x = 0.54$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 2-4 of the arterial system are visible............................................. 69

Figure 5.17: Contours of temperature distribution in the yz-plane at $x = 0.81$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 1, 5, 6, 7 of the arterial system and the center of the tumor are visible................................................................. 71

Figure 5.18: Contours of temperature distribution in the yz-plane at $x = 0.81$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 1, 5, 6, 7 of the arterial system and the center of the tumor are visible.................................................................................... 72
Figure 5.19: Contours of temperature distribution in the yz-plane at \( x = 1.08 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s, while (e) represents the plane where the results were observed. Here, level 1 of the arterial system is visible ............................................. 74

Figure 5.20: Contours of temperature distribution in the yz-plane at \( x = 1.08 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s, while (e) represents the plane where the results were observed. Here, level 1 of the arterial system is visible ............................................. 75

Figure 5.21: Contours of the temperature distribution in the xz-plane at \( y = 0.54 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s. Here, levels 2, 6, and 7 of the arterial system are visible .......... 77

Figure 5.22: Contours of the temperature distribution in the xz-plane at \( y = 0.54 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible ....................... 78

Figure 5.23: Contours of the temperature distribution in the xz-plane at \( y = 0.81 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s, while (e) represents the plane where the results were observed. Here, levels 1 and 2 of the arterial system and the center of the tumor are visible .............................................................................................. 80

Figure 5.24: Contours of the temperature distribution in the xz-plane at \( y = 0.81 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s, while (e) represents the plane where the results were observed. Here, levels 1 and 2 of the arterial system and the center of the tumor are visible ............................................................................................... 81

Figure 5.25: Contours of the temperature distribution in the xz plane at \( y = 1.08 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible ....................... 83

Figure 5.26: Contours of the temperature distribution in the xz plane at \( y = 1.08 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible ....................... 84
Figure 5.27: Temperature profile at $t = 50$ s along lines (a) along the z direction from the top to bottom at the central point of the skin surface, (b) at the point (.81 cm, .81 cm, .365 cm) over time, (c) $y = .81$ cm, and (d) $x = .81$ cm on the surface of the skin, while (e)-(h) are the locations where the results were observed.
ACKNOWLEDGMENTS

Foremost, I thank my Creator. It is through Him that I am able to do anything. Furthermore, I want to thank my family. Greg, my father, thank you for your unmatched support you have given me through the years. Without your love and sacrifice, I could never have completed this work. Janna, my aunt, thank you for being a second mother to me and for being there when I needed advice on careers or life in general. To my cousins, grandparents, and siblings, thank you for your love and support as I pursued my hopes and dreams.

I would like to thank my friends for their understanding and assistance as I progressed during the past three years. In particular, I'd like to thank my friend, Stanislav Ponomarev, for the time he gave in assisting me in teaching several programming languages to further the goal of this dissertation. I'd also like to thank my friends who became family over time: Michael Manuel, Michael Wilson, and Jonathan Godby; thank you for your support and for the willingness to lend a listening ear when needed.

To my former mentors, Fred Cavalier and Joseph Urbanski, thank you for inspiring me to become a mathematician and to attend graduate school. Without your influences, none of this would have been possible.

I would also like to thank my advisor, Dr. Weizhong Dai, for his countless hours of support, input, direction, patience, and for the chance to learn under a brilliant
mathematician. Furthermore, Dr. Steve Jones has provided me with priceless help and theoretical insights. I would like to thank the rest of my committee members, Dr. Patrick O’Neal, Dr. Katie Evans, and Dr. Sumeet Dua, for their ideas and feedback.

I would like to thank all the faculty and staff in the Mathematics and Computer Science departments at Louisiana Tech for their support and help. Each person has influenced either my teaching or research in some way. I cannot thank them all enough.
CHAPTER 1

INTRODUCTION

1.1 General Overview

In hyperthermia skin cancer treatment, the objective is to control laser heating of the tumor (target temperatures of 42-46 °C) so that the temperatures of the normal tissue surrounding the tumor remains low enough not to damage the normal tissue. However, it is not easy to accurately determine of the temperature field over the entire treatment region during clinical hyperthermia treatments, because the number of invasive temperature probes that can be used is limited due to the pain tolerances of the patients. Hence, it is important to determine the laser intensity and pattern of laser exposure for optimizing the temperature distribution in the treated region. Temperature distributions during thermal therapy are determined by the power deposition pattern of the heating source, heat removal by conduction, and heat removal by blood flow forced convection.

Many numerical and experimental methods have been developed based on the Pennes’ bioheat transfer equation [1,5,7-24]. Among these, Liauh and Roemer [9] presented a semi-linear state and parameter estimation algorithm that decreases the total computational time between the temperature and the blood perfusion. The algorithm is based on the assumption that Pennes’ bioheat transfer equation (BHTE) is
generally nonlinear in the hyperthermia temperature estimation problem. Huang [11] considered the heat transfer within a perfused tissue in the presence of a vessel. Payne [12] designed a model to predict the irradiation dosage in a human body. The model was created from the combination of the convective fin equation and the Pennes' BHTE. Furthermore, an inverse technique was applied to experimental data to determine model parameters. Liu [14, 15] modeled wave-like behaviors of bioheat transfer in a 1-D triple-layered skin structure and used a finite-difference method to solve the problem. Sun et al. [16] developed a system that can treat tumor tissue and monitor the heating and cooling during the treatment by means of an invasive probe. The prediction of the temperature distribution in the tumor was modeled using the Pennes' equation. Dai and his colleagues [5, 17, 18] developed a domain decomposition method for solving the 3D Pennes' BHTE in a triple-layered skin structure. In particular, Zeng et al. [7] developed a model for a 3D triple-layered skin structure with a vascular countercurrent network that employed a modified Pennes' equation which accounts for thermal lag of the tissue where the vascular network is designed based on the constructal theory of multi-scale tree-shaped heat exchangers. Majchrzak and colleagues [19] developed a model based off the Pennes' equation to study the hyperthermia and hypothermia processes and to identify thermal parameters in a biological medium. Jamil and Ng [20] developed a model that uses electromagnetic radiation in a single-layered human tissue structure with an embedded tumor based off the Pennes' equation to predict the leading factor for hyperthermia treatment. Majchrzak and colleagues [22] modeled the dual-phase lag equation in a 3-D skin structure and obtained a solution numerically with an explicit finite difference method.
For hyperthermia cancer treatment, a recent advance is to inject golden nanoparticles into the tumor region in order to heat it up quickly [3,4,25–29]. Vera and Bayazitoglu [4] developed a model that used finite difference time domain (FDTD) to calculate the heat distribution of nanoshells on a single layer of human biological tissue at varying particle distribution densities in different host mediums. Pignol et al. [25] used the Monte Carlo method to predict gold nanoparticle radiosensitization needed to be lethal in a cell media on the nano-scale. Lin et al. [26] also used the Monte Carlo method to model the effect of gold nanoparticles in response to multiple beam intensities to enhance the effect of proton beam therapy. Zunino et al. [27] developed a model that predicts the effect of nanoparticles in a tumor microenvironment in the form of temperature, by using the finite element method while taking into consideration the tumor’s vascular system and heat transfer of the vascular system with the distribution of nanoparticles. Naghavi et al. [28] developed a model with gold nanoshells to predict the temperature distribution of a prostate with a tumor that is being irradiated by a laser and nanoparticles, which is solved numerically by the finite element method. Frieboes and Curtis et al. [29] evaluated the effects of drug-loaded gold nanoparticles in highly vascularized tumors, where the effect of the nanoparticles decreased tumor size in comparison to drug-free nanoparticles.

1.2 Research Objectives

To date, a model has not been developed for the laser heating in a 3-D triple-layered skin tissue where the tumor is injected by nanoparticles with a nearby countercurrent vascular network. For this purpose, the objective of the present study
is to develop a mathematical model that can accurately predict the temperature in the
tumor region and surrounding normal tissue. To this end, we employ a modified Pennes’
equation for the bioheat transfer in a 3-D triple-layered skin structure embedded
with a vascular countercurrent network and a tumor appearing in the subcutaneous
region. The vascular network design is based on the constructal theory of multi-scale
tree-shaped heat exchangers [30,31], for which the structure has been seen previously
in Zeng’s paper [7]. The tumor is injected with golden nanoshells accelerate heating.
A stable finite-difference scheme is then used to solve the proposed model numerically.
To determine the laser intensity so that an optimal temperature distribution can be
obtained, we pre-specify the temperature elevations to be obtained at the center of
the tumor and on some locations on the perimeter of the skin’s surface. We use the
least squares method to obtain the optimal laser power and develop a computational
procedure to obtain the temperature distribution.

The organization of the rest of the text is given as follows. In Chapter 2, the
literature that led to the current work is presented. In Chapter 3, the bioheat transfer
model for thermal analysis in a 3-D triple-layered skin structure with a countercurrent
vascular network with an embedded tumor and nanoshells in the subcutaneous region
is presented. In Chapter 4, we propose a finite difference method and computational
procedure to numerically approximate the solution to the model. In Chapter 5, we
test the method by an example and discuss the model’s potential use in a clinical
setting. In Chapter 6, a review of the current work and direction of future work is
discussed.
CHAPTER 2

LITERATURE REVIEW AND PREVIOUS WORK

Chapter 2 will review the main literature related to the hierarchical branching network and constructal theory of multi-scale tree-shaped heat exchangers, laser-induced nanoshell heat generation, and the previous work on the bio-heat transfer in skin tissue.

2.1 Hierarchical Branching Network and Constructal Theory of Multi-scale Tree-shape Heat Exchangers

This review section is based on Zeng’s dissertation [7], Heat transfer in the living tissue of organisms is a complex process that is required for life of any form to exist. In living tissue, heat transfer occurs through a series of vascular networks that have great levels of hierarchy. From a microscopic perspective, living tissue is comprised of two subsystems: a highly branching hierarchical vascular network complete with arterial and venous beds and the cellular tissue that surrounds the previously mentioned vascular network. The two subsystems coexist with the blood flow from the arteries supplying the tissue with essentials such as, but not limited to, oxygen and various nutritional products while controlling the heat balance of the biological system. Conversely, the venous blood flow withdraws products, such as carbon dioxide, resulting from life activity of cellular tissue. The depositing and
withdrawing of products by the arterial and venous systems occur seamlessly because the vascular network is embedded in the cellular tissue. Although the vascular system occupies a small percentage of the volume within the cellular tissue, the fast convection transport of the blood flow within the vessels causes it to be the main determinant of heat and mass propagation.

Blood vessels make up a complex network that varies significantly between individual organisms; however, the vessels exhibit similar behavior across all organisms. This behavior is a series of relations that is based on the size of a blood vessel: the larger the blood vessel, the faster the flow rate of the blood within the vessel. Consequently, the increased flow rate results in a stronger effect on heat transfer within the given blood vessel [7,11,32,33]. As a result, with capillaries being the smallest form of vasculature in the body, the blood flow does not have a strong effect on heat propagation; however, blood within large vessels moves quick enough that its heat interaction with the surrounding cellular tissue is negligible [7].

Several models were developed to simulate the vascular countercurrent network [1,2]. In particular, Huang and Roemer [33] presented a vascular countercurrent network as shown in Figure 2.1, which is based on the structure in Figure 2.2 in Roemer’s paper.
Figure 2.1: A proposed geometric model used consists of seven levels of a regular, branching network of arteries (red) and veins (blue) within a cubic space.

Figure 2.2: A figure demonstrating the location of a subset of the vessel network within the cubic space. A total of 64 such sets are used in the model, where the subsets are regularly and uniformly spaced throughout the cubic space [1]
In Figure 2.2, all the vessels are parallel to one of the three major axes. As mentioned previously, the network is comprised of seven levels of arteries and veins. At the base of the network (level one) are the largest vessels present; as the network progresses into different levels, the diameters of the vessels decrease by a constant ratio $\gamma$ between successive levels of branched vessels, according to Eq. (3.1),

$$\gamma = \frac{D_{i+1}}{D_i}, \quad (2.1)$$

where $D_{i+1}$ and $D_i$ are the diameters of two successive levels of branching vessels. When the vessels do not bifurcate, but change direction instead (levels 6 and levels 7), the vessel diameter does not change as shown in Figure 2.2.

From Zeng's dissertation, the blood vessel system can be described as follows:

"...from Figure 2.2 that the artery network consists of the large central vessel (level one) running lengthwise (x) along the control volume. This vessel has two pairs of symmetric, vertical (z) vessels (level two) branching from it, one pair just inside the beginning of the cube (x=0) and one pair at the central plane (x=L/2). The mass flow rate in the main arterial vessel decreases in a stepwise manner at both of those locations due to the blood removed by the second level vessels. In all cases studied in this paper, the Level 1 artery is assumed to terminate at the central plane where the second set of Level 2 vessels branch off of it. Each of the second level vessels branches into two level three vessels, which run crosswise (y). Each Level Three vessel feeds into a Level 4 vessel which runs lengthwise
with each Level 4 vessel extending (almost) one-half of the control volume length. Each of these Level 4 vessels has four pairs of vertical (z) Level 5 vessels periodically branching off of it, including the pairs at the corners where vessel levels three and four meet. Each Level 5 vessel branches into two Level 6 vessels which run crosswise (y). Each Level 6 vessel then changes direction and becomes a Level 7 vessel which runs lengthwise (x). All vessels parallel to the x direction (i.e. the level one, four, and seven vessels) have been symmetrically located within the y, z planes, i.e. the Level 1 artery is in the y, z center of the control volume, the level four vessels are located at centers of the four y, z quadrants of the control volume, and the Level 7 vessels are located at centers of the 16 squares that regularly divide up any y, z plane of the control volume. Similarly, the vessels parallel to the y, and z directions are uniformly spaced in the x direction.”

Lubashevsky and Gafyanchuk [2] considered the living tissue to be a heterogenous medium. They also assumed that blood flow in capillaries does not affect heat propagation due to the blood’s high velocity in larger vessels makes the interaction with surrounding cellular tissue negligible. In their research, two different blood vessel networks were proposed as shown in Figure 2.3, where the figure was obtained from Lubashevsky’s book published in 2002 [2].
Figure 2.3: Models of the peripheral artery network embedding into the cellular tissue, (a) four-fold model and (b) a more realistic dichotomic artery tree uniformly embedded into a cubic tissue domain. When tested, both models had the same impact on the tissue domain [2].

In Figure 2.3a, a simplified model is shown where the sequential hierarchical levels $n$ and $n+1$ are related as $l_n = 2l_{n+1}$ where $l_n$ represents the length of a vessel at Level $n$. Figure 2.3b illustrates a more simplistic structure for the arterial network which can be transformed into the previous case by combining three bifurcating nodes in sequence into an effective four-fold node at all levels of the network, due to the framework of the analysis performed in the study in [2]. Because the model can be transformed from 2.3b to 2.3a, the cubic domain of the volume falls per each artery of Level $n$, thus leading to a more realistic, yet simplistic, model than Figure 2.3a.

Later, Bejan [30,34] proposed “The Constructal Theory of Multi-scale Tree-shaped Heat Exchangers,” a relatively new geometric model for a vascular network. Based on the constructal theory, it was pointed out that an optical relationship exists among the lengths of the blood vessels such that

$$L_m = \sqrt{2}L_{m+1}, \quad m = 1, 2, ..., n.$$  \hspace{1cm} (2.2)
The number of blood vessels, $N_m$, in each level is given by

$$N_m = 2^{m-1}, \quad m = 1, 2, \ldots n. \quad (2.3)$$

The diameters of the blood vessels satisfy

$$\frac{D_{i+1}}{D_i} = \gamma = 2^{-\frac{i}{n}}, \quad i = 1, 2, \ldots, n. \quad (2.4)$$

Finally, the ratio of the flow rates, $\dot{m}_i$, satisfies

$$\dot{m}_i = 2^{-i} \dot{m}_1, \quad i = 1, 2, \ldots n. \quad (2.5)$$

### 2.2 Nanoshell Heat Generation

In recent years, there has been an interest in using nanoparticles with a laser to heat up tumor regions to advance minimally invasive procedures. Bayazitoglu and colleagues [3, 4, 35] performed studies and developed a model to describe the heat generation of gold nanoshells using the Finite-Difference Time Domain (FDTD) method to model the effects of the nanoshells at varying volumes within a slab of human tissue. Single laser and dual laser simulations were carried out. The temperature, $T$, was governed by the heat equation

$$\frac{1}{\alpha_{md}} \frac{dT}{dt} = \frac{d^2T}{dz^2} + \frac{u'''}{k_{md}}, \quad (2.6)$$

where $u'''$ is the local heat generation spectrum across the depth of the tissue slab, $\alpha_{md}$ is the thermal diffusivity of the slab, $k_{md}$ is the thermal conductivity of the slab. To
solve the equation, a conditionally-stable FDTD scheme is used and only works if the Fourier number ($F_o$) is below 0.5. The Fourier number is defined as $F_o = \alpha_{md} \Delta t / \Delta z^2$.

The heat generation term, $u''$, is defined as

$$u'' = -\frac{d q_{R\lambda}}{dz},$$

where $q_{R\lambda}(\tau)$ is the spectral radiative heat flux that is the summation of the heat generated by collimated radiation, $q_{c\lambda}(\tau)$, and diffuse radiation, $q_{d\lambda}(\tau)$ such that

$$q_{c\lambda}(\tau) = (1 - \rho) q_{in} e^{\frac{\tau}{\kappa_c}},$$

and

$$q_{d\lambda}(\tau) = C_1 e^{\xi_1\tau} + C_2 e^{-\xi_1\tau} + B_1 e^{-\tau}.$$  \hspace{1cm} (2.9)

Here, $q_{in}$ is the input laser beam power level and $\tau$ is the optical depth defined as $\tau = \beta_{tot\lambda} \times z$ where $\beta_{tot\lambda}$ is the total extinction of the system. $\mu_c$ is the angle of incidence defined as $\mu_c = \cos(\theta)$.

The analytical constants $C_1$ and $C_2$ are defined as

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} B_2 & B_3 \\ B_6 & B_7 \end{bmatrix}^{-1} \begin{bmatrix} B_4 B_1 + B_5 \\ B_5 B_1 + B_9 \end{bmatrix},$$

where the constants $B_1 - B_9$ are as
\[ B_1 = -\frac{1}{\mu_c} \frac{\omega_\lambda (1 - \rho_\lambda(0)) q_m^{\prime\prime}}{\frac{1}{\mu_c} - \xi_1^2}, \quad (2.11) \]

\[ B_2 = 2 - \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \xi_1, \quad (2.12) \]

\[ B_3 = 2 + \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \xi_1, \quad (2.13) \]

\[ B_4 = -\left(2 + \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \frac{1}{\mu_c}\right), \quad (2.14) \]

\[ B_5 = -\frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \frac{\omega_\lambda (1 - \rho_\lambda(0)) q_m^{\prime\prime}}{} + \frac{4(1 - \varepsilon_\lambda)}{2 - \varepsilon_\lambda} H_{c0,\lambda}, \quad (2.15) \]

\[ B_6 = -\left(2 e^{\xi_1 \tau_{L\lambda}} + \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \xi_1 e^{\xi_1 \tau_{L\lambda}} \right), \quad (2.16) \]

\[ B_7 = -2 e^{-\xi_1 \tau_{L\lambda}} + \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \xi_1 e^{-\xi_1 \tau_{L\lambda}}, \quad (2.17) \]

\[ B_8 = 2 e^{-\frac{\tau_{L\lambda}}{\mu_c}} - \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \frac{e^{-\frac{\tau_{L\lambda}}{\mu_c}}}{\mu_c}, \quad (2.18) \]

\[ B_9 = -\frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \frac{\omega_\lambda q_m^{\prime\prime}}{} e^{-\frac{\tau_{L\lambda}}{\mu_c}} + \frac{4(1 - \varepsilon_\lambda)}{2 - \varepsilon_\lambda} H_{cL,\lambda}, \quad (2.19) \]
and $\xi_1$ is

$$
\xi_1 = \sqrt{3(1 - \omega_\lambda)}.
$$

(2.20)

From Eqs. (2.11)-(2.20), we obtain $\varepsilon_\lambda$, $\tau_{L,\lambda}$, $\mu_c$, $q_{in,\lambda}$, $\rho_\lambda$, $\omega_\lambda$, $H_{cL,\lambda}$, and $H_{c0,\lambda}$.

Here, $\varepsilon_\lambda$ represents the proportion of incoming light that is transmitted through the boundary surface; it is equivalent to $1 - \rho_\lambda(0)$ and $1 - \rho_\lambda(\tau_{L,\lambda})$ at the entry and exit boundaries, respectively. $\omega_\lambda$ is, by definition, the scattering albedo for the system and is defined as follows

$$
\omega_\lambda = \frac{\sigma_{s-\lambda} + \sigma_{md-\lambda}}{\beta_{tot-\lambda}}.
$$

(2.21)

The optical length of tissue region, $\tau_{L,\lambda}$, is defined as

$$
\tau_{L,\lambda} = \beta_{tot-\lambda} L.
$$

(2.22)

The extinction coefficient is defined as

$$
\beta_{md} = k_{md-\lambda} + \sigma_{md-\lambda},
$$

(2.23)

where $k_{md-\lambda}$ and $\sigma_{md-\lambda}$ are the spectral absorption and scattering coefficients of the medium, respectively. However, for the tumor region, the extinction coefficient also has to take into account the nanoshells that are scattering and absorbing the laser. The extinction for such would be defined as

$$
\beta_{tot-\lambda} = k_{md-\lambda} + \sigma_{md-\lambda} + k_{s-\lambda} + \sigma_{s-\lambda}.
$$

(2.24)
where \( k_{s-\lambda} \) and \( \sigma_{s-\lambda} \) are the spectral absorption and scattering coefficients of the nanoshells, respectively. \( k_{s-\lambda} \) and \( \sigma_{s-\lambda} \) are defined as

\[
k_{s-\lambda} = \pi r_o^2 Q_{\sigma\lambda} N_T, \quad \sigma_{s-\lambda} = \pi r_o^2 Q_{k\lambda} N_T, (2.25)
\]

where \( r_o, Q_{\sigma\lambda}, Q_{k\lambda}, \) and \( N_T \) are the outer radius, spectral scattering efficiency, spectral absorption efficiency of the nanoshells, and the amount nanoshells per unit volume, respectively.

Finally, \( H_{c0,\lambda} \) and \( H_{cL,\lambda} \) represent the sum of incoming collimated rays at the entrance and exit boundaries; Considering the assumptions of \( \rho_\lambda \) previously stated, they are defined respectively as

\[
H_{c0,\lambda} = 0 \quad H_{cL,\lambda} = G_{c\lambda}, (2.26)
\]

where \( G_{c\lambda} \) is the collimated irradiance and is expressed as

\[
G_{c\lambda} = (1 - \rho_\lambda(0)) u_{in,\lambda} e^{-\mu r}. (2.27)
\]

The equations above were used in studies by Tjahjono [3] and Vera [4]. Tjahjono examined the effects of radiative heat transfer analysis of uniformly distributed gold nanoshells in a one-dimensional poly-dimethylsiloxane (PDMS) slab that is being exposed to a uniformly distributed light source at a wavelength of 820 nm as shown by Figure 2.4. To model the temperature distribution, Eq. 2.6 is used to model the temperature distribution with the following assumptions: (1) the physical and thermal properties of the host medium are constant, (2) the host medium is transparent.
to spectral irradiation, (3) the medium is homogeneous, (4) the medium is one-dimensional, (5) the convective terms are constant over time, (6) emission is negligible, and (7) the nanoshells contribute minimal additional effects to the physical and thermal properties of the medium.

![Diagram of light propagation within PDMS slab](image)

**Figure 2.4:** Illustration of light propagation within PDMS slab

In this study, two different types of gold nanoshells are used. One type has a scattering optical configuration and the other has an absorptive optical configuration. To study the extent of the uniform laser has on the PDMS slab, a laser intensity of 300 kW/m² was used to heat up the slab. The gold nanoshell concentrations used in computation were $7.0174 \times 10^{14}$, $7.0174 \times 10^{15}$, $7.0174 \times 10^{16}$ particles per cubic meter to produce optical lengths of 0.1, 1, and 10, respectively. An explicit finite difference method was used to solve the proposed model, where the starting temperature at all locations in the region was 27 °C and had a stopping criterion of 173 °C was used, for the laser, to keep the PDMS slab from melting; the simulations stopped when the temperature at the entry point for the laser reached 200 °C.
The results of this study showed that some of the thermal and physical properties correlate with each other. One such correlation is the relationship between the optical length and the extinction coefficient; as the optical length increases, the extinction coefficient increases by the same order of magnitude because the length of the medium does not change. This correlation causes the collimated radiative heat flux to experience a larger decay, where most of the radiation is absorbed near the entry location of the region with little of that radiation reaching the rear of the region. Because little heat reaches the region in this case, conduction on through the material becomes the primary method of heat transfer in this part of the region, driven by the temperature gradient between the entry and exit locations of the given medium. It was also observed under the absorptive configuration, that the collimated heat source had a major impact on the total radiative heat flux while the diffusive radiative heat flux did not have a significant contribution. Because of this, the radiative heat flux experiences little variability when the scattering is low. However, when the scattering configuration is used, the diffuse radiative heat flux has a significant impact on the total radiative heat flux which helps to disperse heat through the medium. In this case, the radiative heat source has more variability because of the collimated heat sources decay as a result of the scattering configuration.

Heat generation was also influenced by the scattering albedo of the medium. If the extinction of the medium is held constant, more radiation is absorbed in the absorptive configuration than the scattering configuration of the nanoshells. However, for a larger extinction the magnitude the absorption near the end of the region is much smaller because most of the radiation is being absorbed in the entry region.
With this known, and with the given nanoshell configuration, the following conclusion can be made: if the radiation available is minimal, the nanoshells have minimal energy to produce heat. As a conclusion, the overall temperature distribution is relatively higher in the scattering configuration than the absorptive configuration because the absorptive configuration absorbs more radiation per length than the scattering configuration. Since the scattering configuration absorbs less heat per length, more heat is distributing to other areas. Thus, if the goal is to heat an entire region over time, the scattering configuration should be used; however, if the goal is to heat up a small region quickly, the absorptive configuration should be used.

To use Eq. (2.6) to model the temperature distribution in Vera’s paper, several assumptions were taken into account in its development: (1) the physical and thermal properties of the host medium were are constant, (2) the nanoshells contribute negligible additional effects to the medium’s physical and thermal properties, (3) the medium in question is assumed homogeneous, (4) the medium is one-dimensional, and (5) Emission is negligible [4]. In this study, a wavelength of 633nm was used for the laser with various laser power levels of 5,000, 10,000, 15,000 and 20,000 W/m². Furthermore, the study considered different particle distribution densities of 0, 7 × 10¹⁴, 7 × 10¹⁵, 7 × 10¹⁶ particles per cubic meter. The configuration of this study is shown in Figure 2.5.
The 633 nm wavelength was chosen to allow for maximum transmittance through blood and water within tissue [3, 4]. Consequently, the nanoshells proposed are theoretical and are tuned to 633 nm [35, 36]. This assumption is based on Surface Plasmon Resonance (SPR), which allows the nanoshells to change their absorption and scattering at a particular wavelength by changing their core-to-shell ratio, thus the nanoshells, with an inner radius of 16 nm and outer radius of 22 nm, allow for maximum absorption of the laser in nanoshell-mediated tissue while being transparent to normal, healthy tissue [36–39]. However, 1035 nm shows the least absorbency for tissues [40]. In common practice, with nanoparticles that are available, most researchers use 800-830 nm [41–44] when attempting this type of therapy.

The study examined the scenarios of single laser heating applied to one side of the medium and a two-laser configuration applied to both sides of the one-dimensional
medium in question. The results were categorized into five major categories: (1) holding the power intensity constant while increasing the nanoshell density distribution, (2) holding the nanoshell density constant while increasing the power intensity, (3) using dual facing lasers instead of one laser, (4) holding the nanoshell dispersion density constant while increasing the power while using a dual facing lasers, and (5) holding the power intensity constant while increasing the nanoshell density distribution using a dual-laser configuration. The simulations were stopped when the entry location of the laser to the tissue reached 55 °C.

The results of their study showed that increasing the amount of nanoshells in the model significantly decreases the time required to heat tissue of different types (e.g. brain, dermis, liver, subcutaneous). In this case, it is beneficial to use nanoshells to heat a small area, such as a tumor, while leaving the surrounding regions unaffected. It should be noted that if the stopping criterion of 55 °C was not in place to stop the laser heating, the heating would continue to escalate to damaging levels. When the laser intensity was increased, it had a similar effect, especially on the entry region where most of the radiation was absorbed. The rapidly rising temperatures, it allows little time for the heat to diffuse evenly in the depths of the region. Furthermore, if the stopping criterion of 55 °C at the entry region was not applied, the temperature escalated to damaging temperatures before the depths of the tissue could heat up. The medium heated up more evenly with the dual laser approach than with the single laser approach, with or without gold nanoshells. Addition of more nanoshells and/or an increase in laser intensity caused beneficial rapid heating in the medium, where the heat in the medium had little time to diffuse into the surroundings. This situation
would create a plateau-effect on temperature elevation with the highest temperatures in the targeted region.

### 2.3 Previous work related to the Bio-Heat Transfer Equation

This review section is based on Tang’s dissertation [5] and Zeng’s dissertation [7]. From Tang’s dissertation, the Pennes’ BHTE is the most used model for hyperthermia treatment. When using the BHTE, heat transfer between the blood vessels and tissue occurs mainly across the capillaries where the velocity of the blood is low [45]. The contribution of the blood flow’s heat transfer was modeled as a heat sink where the blood in the capillary bed instantly balances with the temperature of the surrounding tissue. With this mechanism, the magnitude of this heat sink is proportional to the difference between the arterial blood supply’s temperature and the temperature of the local tissue.

The Pennes’ equation is

\[
\rho_l C_l \frac{dT_l}{dt} = k_l \left( \frac{\partial^2 T_l}{\partial x^2} + \frac{\partial^2 T_l}{\partial y^2} + \frac{\partial^2 T_l}{\partial z^2} \right) + W_b^l C_b^l (T_{\text{out}} - T_l) + Q_l, \quad l = 1, 2, 3
\]

where \( T_l \) is the temperature of the \( l \)th skin tissue layer; \( T_{\text{out}} \) is the blood temperature at the exit or entrance of the seventh level vessel for the artery or vein; \( \rho_l \), \( C_l \), and \( k_l \) denote the density, specific heat, and thermal conductivity of the \( l \)th skin tissue layer, respectively; \( C_b^l \) is the specific heat of blood; \( W_b^l \) is the blood perfusion rate; and \( Q_l \) is the volumetric heating due to spatial heating.

As pointed out by Tang, many experimental and numerical studies have been based on the Pennes’ BHTE as the governing equation for the behavior of heat transfer in biological mediums. Among those studies, Clegg and Roemer [46] accurately
predicted the complete three-dimensional temperature distribution in experimental situations on a normal canine thigh to test the ability of a state and parameter estimation method. The governing equation of the model was the Pennes’ equation along with an optimization algorithm that was based on a least squares error objective function. With the Pennes’ equation and optimization algorithm implemented, the model was used to predict unknown model parameters, such as blood perfusion and the power deposition. Martin and Bowman [47] implemented the Pennes’ equation in cylindrical coordinates. As a result, they presented the transient and exact state solutions for the temperature distribution in perfused and laser irradiated tissue. Consequently, their model was used to evaluate the significance of blood perfusion during continuous wave laser heating. While the relationship between temperature and blood perfusion is generally nonlinear in the hyperthermia temperature estimation problem, Liauh and Roemer [9] presented a semilinear state and parameter estimation algorithm based on the Pennes’ BHTE for the hyperthermia temperature estimation problem that decreases the total computational time required to accurately reconstruct hyperthermia temperature fields.

With Pennes’ equation and use of the mesh generation software ANSYS, Chatterjee and Adams [10] generated a 2D finite element thermal model of the human prostate. The results obtained demonstrate the effects of different blood flow rates and show how selective heating can be obtained in a tumor region with the prostate.

Huang [11] developed a model based on the Pennes’ BHTE equation that considered heat transfer within a perfused tissue in the presence of a blood vessel. In addition to the BHTE, a lumped capacitance analysis was used to model convection
of the vessel with a constant Nusselt number. As a result, analytical solutions of the Pennes' equation in the presence of a blood vessel were obtained. Majchrzak and Mochnacki [13] developed a model based on the BHTE that governs temperature within the tissue sub-domain while having ordinary differential equations resulting from the energy balance to govern the blood temperature's changes along the vessel. The model was developed to study the thermal processes of perfused tissue in the presence of a vessel. The solution to this equation is a combined numerical algorithm that results from the use of the boundary element method for the tissue's sub-domain and the finite difference method for the blood vessel's sub-domain.

Liu and his colleagues [14,15,48] introduced the thermal wave model of the Pennes' BHTE in its general form for all living tissues. Based upon a modified unsteady conduction equation, a general heat flux criterion was founded to determine the propagation of thermal waves that dominates the primary heat transfer process. Their model can be used to map temperature distributions in tissue. Lu and Liu [15,49] also used the dual reciprocity boundary element method to solve the integral inverse or direct bio-heat transfer problems.

Zhou and Liu [50] used the continuity, momentum, and energy equations from fluid dynamics to calculate the temperature distributions in three dimensions. Dai [17] developed a domain decomposition method for solving the Pennes' BHTE in three dimensions in a triple-layered skin structure. Dai and Zhang [51] developed a numerical method for obtaining an optimal temperature distribution in a cylindrical triple-layered skin structure. Comprised of the epidermis, dermis, and subcutaneous
layers of the skin, it was the first model to use the triple-layered structure for laser-induced hyperthermia and to solve it numerically. While the method proved to be useful in optimizing laser power for a given laser irradiation pattern, it did not consider the influence of blood vessels within the skin structure. The presence of thermally significant vessels, such as an arterial and venous network, can strongly influence the temperature distribution in hyperthermia applications [52].

Dai and Zhang [18] developed a numerical method for obtaining an optimal temperature distribution in a three-dimensional triple-layered cylindrical skin structure with a blood vessel embedded in the tissue while being influenced by laser heating. Although a blood vessel was present in the model, the influence of veins was ignored. Tang and Dai [5] developed a numerical method for an optimal temperature distribution in a three-dimensional rectangular triple-layered skin structure with an embedded three-level countercurrent vascular network with laser heating. The vascular network is based on Bejan's constructal theory of multi-scale tree-shaped heat exchangers [30, 34, 53] in terms of dimensionality and blood flow rates. In Figure 2.6 (copied from Tang's paper [5]), The basic arterial model consists of the large central vessel for level 1 running lengthwise in the z-direction of the tissue domain. The level 2 vessels bifurcate at the end of level 1 and run lengthwise in the x-direction. The second level vessels, however, do not bifurcate but change direction and run lengthwise in the z-direction. The vessels are modeled as slim cuboids for simplicity.
Figure 2.6: Configuration of a 3D triple-layered rectangular skin structure with an embedded three-level countercurrent network in [5]

Dai and his colleagues [6] developed a mathematical model for skin burn injury in a triple-layered skin structure embedded with a three-level countercurrent network with radiation heating (Figure 2.7). It has a similar vascular structure similar to that used in [5], but is based on the modified Pennes’ equation

\[
\rho_i C_i \left( \frac{\partial T_i}{\partial t} + \tau \frac{\partial^2 T_i}{\partial t^2} \right) + \tau W_b^l C_b^l \frac{\partial T_i}{\partial t} = k_l \left( \frac{\partial^2 T_i}{\partial x^2} + \frac{\partial^2 T_i}{\partial y^2} + \frac{\partial^2 T_i}{\partial z^2} \right) + W_b^l C_b^l (T_{out} - T_i) + Q_l, \quad l = 1, 2, 3, \tag{2.29}
\]

which governs bio-heat transport. Here, \(T_i\) is the temperature of the \(l\)th skin tissue layer; \(T_{out}\) is the blood temperature at the exit or entrance of the seventh level vessel.
for the artery or vein; $\rho_l$, $C_l$, and $k_l$ denote the density, specific heat, and thermal conductivity of the $l$th skin tissue layer, respectively; $C^l_b$ is the specific heat of blood; $W^l_b$ is the blood perfusion rate; and $Q_l$ is the volumetric heating due to spatial heating. $\tau$ is the relaxation time caused by the thermal lag of the heat flux. When $\tau$ is zero, the Eq. (2.27) reduces to the Pennes’ equation [45] Eq. (2.28).

Dai and Zeng [7] developed a model based on the modified Pennes’ equation in a rectangular triple-layered skin structure with an embedded seven-level countercurrent vascular network with the structure being exposed to radiation heating to model the burning of tissue. The vascular network reaches the maximum possible number of
levels in a countercurrent network, as defined by Roemer and his colleagues [1]. The dimensionality and blood flow rates are based on the constructal theory of multi-scale tree-shaped heat exchangers [30,31,34,53].

The vascular network is described, with reference to Figure 2.8 from Zeng's dissertation, as follows. We consider the skin tissue to be a rectangular structure embedded with a seven-level countercurrent vascular network, which is a highly branching and hierarchical network as described in [32], in the subcutaneous layer, as shown in Figure 2.8. It should be pointed out that only large blood vessels can be seen in the subcutaneous tissue because the dermis is very sparingly supplied with capillaries and the capillary beds of skin lying immediately below the epidermis, and thus, the contribution of these small vessels to the heat transfer can be ignored [50,54]. In Figure 2.8, the red color dendritic network represents arteries while the blue color dendritic network represents veins, where all are considered as slender cuboids for simplicity. Levels of arteries are designed such that the first-level artery runs from right to left along the x-coordinate; the second-level artery branches from the left end of the first-level artery and flows along the y-coordinate; two third-level arteries has two vessels branch from the two ends of the second-level artery, and flowing along the z-coordinate; four fourth-level arteries branch from the four ends of the third-level arteries, and flow along the x-coordinate; the eight fifth-level arteries branch from the eight ends of the fourth-level arteries, and so on. The vein network has the same number of blood vessels as its counterpart artery in corresponding levels. In total, the skin structure has 128 blood vessels.
Figure 2.8: A 3D triple-layered rectangular skin structure with embedded seven-level counter current network [7]. The red vessels denote the arterial network while the venous network is denoted as blue.

To date, laser heating in a 3-D triple-layered skin tissue where the tumor is injected by nanoparticles with a nearby countercurrent vascular network has not been modeled. For this purpose, the objective of the present study is to develop a mathematical model that can accurately predict the temperature in the tumor region and surrounding normal tissue. Our model is based on a modified Pennes’ equation for the bioheat transfer in a 3-D triple-layered skin structure embedded with a vascular countercurrent network and a tumor appearing in the subcutaneous region. The tumor is injected with gold nanoshells to accelerate the temperature increase.
CHAPTER 3

MATHEMATICAL MODEL

The model under consideration is comprised of a modified Pennes’ equation in 3-D triple-layered skin structure, a vascular network based on the constructal theory of multi-scale tree-shaped heat exchangers, and a laser heat that irradiates the area of consideration along with nanoshell heating in the tumor region.

3.1 Network Design

The vascular network considered in Figure 3.1 is the same as the one presented in Figure 2.8. The considered skin structure includes 128 blood vessels. Although the orthogonal structure of blood vessels (Figure 3.1) is a special design, the constructal-design literature has shown convincingly [31, 55] that when the complexity of the tree is as high as in the present model the volume averaged flow perfusion properties (such as the permeability) do not change with the details of the tree architecture. Because tree flow architectures are robust, the use of assumed structures in the numerical modeling of previously untractable ‘complex’ flow structure is real opportunity for advances.
To determine the diameters of the blood vessels on each level, we follow the constructal theory of multi-scale tree-shaped heat exchangers \cite{30,31,55,56} and assume that the diameters of arteries decrease by a constant ratio $\gamma$ between successive levels of branched vessels, given by \cite{56}

$$\gamma = \frac{NL_{b}^{m+1}}{NL_{b}^{m}} = \frac{NW_{b}^{m+1}}{NW_{b}^{m}} = 2^{-\frac{1}{3}}, \quad m = 1, ..., 6, \quad (3.1)$$

where $NL_{b}^{m}$ and $NW_{b}^{m}$ are the length and width, respectively, of the cross section of a blood vessel in level $m$. The length of a blood vessel is assumed to double after two consecutive construction steps, which can be expressed in the length-doubling rule as
follows

\[ L_b^m = 2^{1/2} L_b^{m+1}, \quad m = 1, ..., 6, \] (3.2)

where \( L_b^m \) is the length of the blood vessel in level \( m \). The mass flow of blood in the \( m \)-th level vessel, \( M_m = v_m F_m \), satisfies

\[ M_m = 2M_{m+1}, \quad m = 1, ..., 6, \] (3.3)

where \( v_m \) is the blood flow velocity and \( F_m (= NL_b^m \times NW_b^m) \) is the area of the cross-section in the \( m \)-th level vessel.

For the blood flow in the vascular countercurrent network, we assume that the blood temperature in the cross-section of a vessel is assumed to be uniform. We further assume that a steady-state energy balance in the blood vessel can be reached because the length of the considered blood vessel is relatively short and the blood velocity is relatively high. However, one may use the transient heat transfer equation for a more accurate solution. Hence, the convective energy balance equations, which are used to calculate the artery (levels 1-6) blood temperatures, can be expressed as [5,7,18,33,51,57]

\[ C_B M_1 \frac{\partial T_b^1}{\partial x} - \alpha P_1 (T_w^1 - T_b^1) = 0, \] (3.4)

\[ C_B M_2 \frac{\partial T_b^2}{\partial y} - \alpha P_2 (T_w^2 - T_b^2) = 0, \] (3.5)

\[ C_B M_3 \frac{\partial T_b^3}{\partial z} - \alpha P_3 (T_w^3 - T_b^3) = 0, \] (3.6)

\[ C_B M_4 \frac{\partial T_b^4}{\partial x} - \alpha P_4 (T_w^4 - T_b^4) = 0, \] (3.7)

\[ C_B M_5 \frac{\partial T_b^5}{\partial y} - \alpha P_5 (T_w^5 - T_b^5) = 0, \] (3.8)
where \( C_B \) is the heat capacity of blood, \( \alpha \) is the heat transfer coefficient between blood and tissue, and \( P_m \) is the vessel perimeter. In addition, \( T_{w}^m \) and \( T_{b}^m \) are the wall temperature and the blood temperature in the \( m \)th level vessel. For the smallest, terminal arterial vessels (level 7), a blood flow rate \((\dot{P})\) is included in the energy balance equation

\[
C_B M_6 \frac{\partial T_b^6}{\partial z} - \alpha P_b (T_b^6 - T_w^6) = 0,
\]

(3.9)

The venous network is assumed to be similar to the arterial network, except that the blood flow direction in each vein is opposite of that in the artery; i.e., countercurrent flow occurs in these two kinds of vessels (see Figure 3.1). Also, the diameter ratio, length ratio, and mass flow ratio of the blood between the successive levels of the branched veins take the same form, as shown in Eqs. (3.1)-(3.3) for the arteries. Moreover, the convective energy balance equations (3.4) - (3.10) used to calculate the blood temperature in the artery domain are applied to the vein domain at the corresponding levels.

### 3.2 Bioheat Transfer Equation

For the skin tissue, the modified Pennes’ equation that gives the temperature distribution in a 3-D triple-layered skin tissue and tumor region can be written as follows [5, 7, 14–16]
\[
\rho_l C_l \left( \frac{\partial T_l}{\partial t} + \tau \frac{\partial^2 T_l}{\partial t^2} \right) + \tau W^l_b C_b^l \frac{\partial T_l}{\partial t} \\
= k_l \left( \frac{\partial^2 T_l}{\partial x^2} + \frac{\partial^2 T_l}{\partial y^2} + \frac{\partial^2 T_l}{\partial z^2} \right) + W^l_b C_b^l (T_{out} - T_l) + Q_l, \ l = 1, 2, 3. \quad (3.11)
\]

Here, \( T_l \) is the temperature of the \( l \)th skin tissue layer; \( T_{out} \) is the blood temperature at the exit or entrance of the seventh level vessel for the artery or vein; \( \rho_l, C_l, \) and \( k_l \) denote the density, specific heat, and thermal conductivity of the \( l \)th skin tissue layer, respectively; \( C_b^l \) is the specific heat of blood; \( W^l_b \) is the blood perfusion rate; and \( Q_l \) is the volumetric heating due to spatial heating. \( \tau \) is the relaxation time caused by the thermal lag of the heat flux. When \( \tau \) is zero, the above equation reduces to the Pennes' equation. Physical parameters for the simulation are listed in Table 3.1.

**Table 3.1: Parameters for a 3-D skin structure**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 ) (J/g °C)</td>
<td>3.6</td>
<td>( \rho_l ) (g/cm³)</td>
<td>1</td>
</tr>
<tr>
<td>( C_2 ) (J/g °C)</td>
<td>3.4</td>
<td>( \rho_{gold} ) (g/cm³)</td>
<td>19.3</td>
</tr>
<tr>
<td>( C_3 ) (J/g °C)</td>
<td>3.06</td>
<td>( (\rho C)_{eff} ) (J/cm³ °C)</td>
<td>3.5027, 3.75</td>
</tr>
<tr>
<td>( C_4 ) (J/g °C)</td>
<td>3.75</td>
<td>( R_{eff_1} )</td>
<td>0.1</td>
</tr>
<tr>
<td>( C_{gold} ) (J/g °C)</td>
<td>0.129</td>
<td>( R_{eff_2} )</td>
<td>0.1</td>
</tr>
<tr>
<td>( C_b^1 ) (J/g °C)</td>
<td>0</td>
<td>( R_{eff_3} )</td>
<td>0.1</td>
</tr>
<tr>
<td>( C_b^2 ) (J/g °C)</td>
<td>4.2</td>
<td>( \alpha_1 ) (1/cm)</td>
<td>1.8</td>
</tr>
<tr>
<td>( C_b^3 ) (J/g °C)</td>
<td>4.2</td>
<td>( \alpha_2 ) (1/cm)</td>
<td>1.8</td>
</tr>
<tr>
<td>( C_b^4 ) (J/g °C)</td>
<td>4.2</td>
<td>( \alpha_3 ) (1/cm)</td>
<td>1.8</td>
</tr>
<tr>
<td>( k_1 ) (W/cm°C)</td>
<td>0.0026</td>
<td>( \alpha_1 ) (1/cm)</td>
<td>1.8</td>
</tr>
<tr>
<td>( k_2 ) (W/cm°C)</td>
<td>0.0052</td>
<td>( W_b^1 ) (g/cm²s)</td>
<td>0</td>
</tr>
<tr>
<td>( k_3 ) (W/cm°C)</td>
<td>0.0021</td>
<td>( W_b^2 ) (g/cm²s)</td>
<td>0.005</td>
</tr>
<tr>
<td>( k_4 ) (W/cm°C)</td>
<td>0.00642</td>
<td>( W_b^3 ) (g/cm²s)</td>
<td>0.005</td>
</tr>
<tr>
<td>( k_{gold} ) (W/cm°C)</td>
<td>3.17</td>
<td>( W_b^4 ) (g/cm²s)</td>
<td>0.005</td>
</tr>
<tr>
<td>( k_{sp} ) (W/cm°C)</td>
<td>1.7405</td>
<td>( v_1 ) (cm/s)</td>
<td>8.0</td>
</tr>
<tr>
<td>( k_{eff} ) (W/cm°C)</td>
<td>1.2745, 0.00642</td>
<td>( \alpha ) (W/cm² °C)</td>
<td>0.2</td>
</tr>
<tr>
<td>( \rho_1 ) (g/cm³)</td>
<td>1.2</td>
<td>( \dot{P} ) (1/s)</td>
<td>0.5 x 10⁻³</td>
</tr>
<tr>
<td>( \rho_2 ) (g/cm³)</td>
<td>1.2</td>
<td>( h ) (W/cm²)</td>
<td>0.001</td>
</tr>
<tr>
<td>( \rho_3 ) (g/cm³)</td>
<td>1</td>
<td>( C_B ) (J/cm³ °C)</td>
<td>4.134</td>
</tr>
</tbody>
</table>
Since the tumor has golden nanoshells embedded into the tissue, we must consider the effective thermal conductivities and heat capacities for the part of the tumor region that contains the golden nanoshells as follows [58]

\[(k_t)_{eff} = \varepsilon_t k_{gp} + (1 - \varepsilon_t)k_t, \quad (3.12)\]

\[(\rho C)_{eff} = \varepsilon_t \rho_{gold} C_{gold} + (1 - \varepsilon_t)\rho_tC_t, \quad (3.13)\]

where \(\varepsilon_t\) and \(k_{gp}\) are the porosity of the tumor and the thermal conductivity of the gold nanoshells, respectively. Similarly, \(C_{gold}\) and \(\rho_{gold}\) are the specific heat capacity and density of gold, respectively. The value for \(k_{gp}\) is obtained from

\[k_{gp} = \frac{k_{gold}}{1 + K_n}, \quad (3.14)\]

where \(K_n\) is the Knudsen number defined as \(K_n = \lambda_{ep}/D_p\), and \(k_{gold}\) is the thermal conductivity of gold. \(\lambda_{ep}\) and \(D_p\) are the effective mean free path of the electron and the diameter of the gold nanoshells, respectively.

### 3.3 Heat Source

We assume that the laser power is continuous with a normal distribution, and we express the heat source \(Q_t\) as [5,59]

\[Q_1 = \alpha_1 e^{-\alpha_1 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-((x-x_0(t))^2 + (y-y_0(t))^2)/2\sigma^2} P_0(1 - Ref f_1), \quad (3.15)\]

\[Q_2 = \alpha_2 e^{-\alpha_1 L_1 - \alpha_2 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-((x-x_0(t))^2 + (y-y_0(t))^2)/2\sigma^2} P_0(1 - Ref f_2), \quad (3.16)\]

\[Q_3 = \alpha_3 e^{-\alpha_1 L_1 - \alpha_2 L_2 - \alpha_3 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-((x-x_0(t))^2 + (y-y_0(t))^2)/2\sigma^2} P_0(1 - Ref f_3), \quad (3.17)\]
where $\alpha_1$, $\alpha_2$, $\alpha_3$ are laser absorptivities of the three layers, respectively; $Reff_1$, $Reff_2$, and $Reff_3$ are laser reflectivities of the three layers of the skin, respectively; $\sigma$ is the standard deviation of the width of a normally distributed laser beam; and $L_1$, $L_2$, and $L_3$ are the depths of the three layers, respectively. Here, $[x_0(t), y_0(t)]$ is the location where the laser is focused at time $t$. $P_0$ is the laser intensity, which will be determined later, optimize the temperature distribution.

However, in the nanoshell-embedded tumor region, the heat source is different. Here, we consider the nanoshells' heat spectrum in conjunction with the heat generation from the laser and give the heat source as

$$Q_t = \varepsilon t u''' + (1 - \varepsilon t)Q_3, \quad (3.18)$$

where $u'''$ represents local heat generation spectrum generated by the gold nanoshells in the tumor region in which the derivation is given in [3,4] and can be seen in the next section.

### 3.4 Nanoshell Heat Generation

The local heat generation spectrum generated by the gold nanoshells in the tumor region is related to the spectral radiative heat flux, $q_{R\lambda}(z)$, by [3,4]

$$u''' = -\frac{d q''_{R\lambda}}{dz}, \quad (3.19)$$

where $q_{R\lambda}(z)$ is the sum of the collimated radiation, $q_{\omega\lambda}(z)$, and the heat generation caused by diffuse radiation, $q_{d\lambda}(z)$. The collimated radiation is defined as

$$q_{\omega\lambda}(z) = (1 - \rho\lambda)q''_{\text{in}} e^{-\frac{\alpha_2 z}{\mu c}}, \quad (3.20)$$
where $\alpha_g$ is the absorptivity of the golden nanoshells. The diffuse radiation is defined as

$$q_{d\lambda}(z) = C_1 e^{\xi_1 \beta_{tot-\lambda} z} + C_2 e^{-\xi_1 \beta_{tot-\lambda} z} + B_1 e^{-\beta_{tot-\lambda} z}, \quad (3.21)$$

where $C_1$ and $C_2$ are calculated from the values that are defined by Eq. (3.22) as follows

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} B_2 & B_3 \\ B_6 & B_7 \end{bmatrix}^{-1} \begin{bmatrix} B_4 B_1 + B_5 \\ B_8 B_1 + B_9 \end{bmatrix}. \quad (3.22)$$

Based on the equations from [3,4], $B_1$ through $B_9$ are defined as

$$B_1 = -\frac{1}{\mu_c} \frac{\omega_\lambda q_{in,\lambda}''}{\mu_c - \xi_1^2}, \quad B_2 = 2 - \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \xi_1, \quad (3.23)$$

$$B_3 = 2 + \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \xi_1, \quad B_4 = -\left(2 + \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \frac{1}{\mu_c}\right), \quad (3.24)$$

$$B_5 = -\frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} q_{in,\lambda}'', \quad B_6 = -(2 e^{\xi_1 \tau_{L\lambda}} + \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \xi_1 e^{\xi_1 \tau_{L\lambda}}), \quad (3.25)$$

$$B_7 = -2 e^{-\xi_1 \tau_{L\lambda}} + \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \xi_1 e^{-\xi_1 \tau_{L\lambda}}, \quad B_8 = 2 e^{-\tau_{L\lambda} \mu_c} - \frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} \frac{e^{-\tau_{L\lambda} \mu_c}}{\mu_c}, \quad (3.26)$$

$$B_9 = -\frac{\varepsilon_\lambda}{(2 - \varepsilon_\lambda)(1 - \omega_\lambda)} q_{in,\lambda}'' e^{-\tau_{L\lambda} \mu_c} \xi_1 = \sqrt{3(1 - \omega_\lambda)}. \quad (3.27)$$

From Eqs. (3.19)-(3.27), we obtain $\varepsilon_\lambda, \tau_{L\lambda}, \mu_c, q_{in,\lambda}'', \rho_\lambda, \omega_{tot-\lambda}$. Here, $\varepsilon_\lambda$ represents the proportion of incoming light that is transmitted through the boundary surface; it is equivalent to $1 - \rho_\lambda(0)$ and $1 - \rho_\lambda(\tau_{L\lambda})$ at the entry and exit boundaries, respectively. For our study, as in the study in [3], reflections at the boundaries will
be minimal and insignificant resulting in negligible values $\rho_\lambda$; thus all the $\rho_\lambda$ will be considered zero. From this assumption, we conclude that $\varepsilon_\lambda = 1$.

Since $\rho_\lambda$ is negligible, Eq. (3.20) changes slightly to

$$q_{\epsilon\lambda}(z) = q_{\text{in,}\lambda} e^{-\mu z}. 	ag{3.28}$$

Here, $q_{\text{in,}\lambda}'''$ represents the heat flux entering the subcutaneous region and is defined as

$$q_{\text{in,}\lambda}' = \frac{P_0}{\sqrt{2\pi\sigma}} e^{-\alpha_1 L_1 - \alpha_2 L_2}. 	ag{3.29}$$

By definition, $\omega_\lambda$ is the scattering albedo for the system, so $\omega_\lambda$ and $\omega_{\text{tot,}\lambda}$ are equivalent variables; the previous is used for convenience in writing and is defined as follows

$$\omega_\lambda = \frac{\sigma_s - \lambda + \sigma_{md - \lambda}}{\beta_{\text{tot,}\lambda}}. 	ag{3.30}$$

The optical length of the tumor region, $\tau_{L,\lambda}$, is

$$\tau_{L,\lambda} = \beta_{\text{tot,}\lambda} L_\ell. 	ag{3.31}$$

The extinction coefficient is defined as

$$\beta_{md} = k_{md - \lambda} + \sigma_{md - \lambda}, 	ag{3.32}$$

where $k_{md - \lambda}$ and $\sigma_{md - \lambda}$ are the spectral absorption and scattering coefficients of the medium, respectively. However, for the tumor region, the extinction coefficient must also take into account the nanoshells that are scattering and absorbing the laser. The
extinction for the nanoshells would be defined as

$$\beta_{tot-\lambda} = k_{md-\lambda} + \sigma_{md-\lambda} + k_{s-\lambda} + \sigma_{s-\lambda},$$

(3.33)

where $k_{s-\lambda}$ and $\sigma_{s-\lambda}$ are the spectral absorption and scattering coefficients of the nanoshells, respectively. $k_{s-\lambda}$ and $\sigma_{s-\lambda}$ are defined as

$$k_{s-\lambda} = \pi r_0^2 Q_{\sigma\lambda} N_T, \quad \sigma_{s-\lambda} = \pi r_0^2 Q_{k\lambda} N_T,$$

(3.34)

where $r_0$, $Q_{\sigma\lambda}$, $Q_{k\lambda}$, and $N_T$ are the outer radius, spectral scattering efficiency, spectral absorption efficiency of the nanoshells, and the amount nanoshells per unit volume, respectively.

Finally, $\mu_c$ represents the angle of incidence, which is defined as

$$\mu_c = \cos(\theta).$$

(3.35)

3.5 Boundary, Initial, and Interfacial Conditions

On the skin surface, we assume that heat exchanges with the surrounding environment

$$k_1 \frac{\partial T_1}{\partial z} = h(T_1 - T_a),$$

(3.36)

where $h$ is the convective heat transfer coefficient and $T_a$ is the ambient temperature. For simplicity, other boundary conditions in the tissue are assumed to be \[5,7,15\]

$$\frac{\partial T_1}{\partial \bar{n}} = 0,$$

(3.37)

where $\bar{n}$ is the unit outward normal vector on the boundary. At the entrance to the first level vessel, we have

$$T_{b1} = T_{in},$$

(3.38)
where $T_{in}$ is the blood temperature at the entrance of the artery. At the exit of the artery, the blood temperature is equal to the surrounding tissue temperature

$$T_b^7 = T_{out}. \quad (3.39)$$

The continuity of heat transfer between the lateral blood vessel and the tissue requires [5, 7]

$$\frac{\partial T_{in}^m}{\partial n} = B_i(T_w^m - T_{in}^m), \ m = 1, \ldots, 7. \quad (3.40)$$

The interfacial condition between three skin tissue layers and the tumor is assumed to be perfectly thermal contact and is given by [5, 7]

$$T_1 = T_2, \ k_1 \frac{\partial T_1}{\partial z} = k_2 \frac{\partial T_2}{\partial z}, \ z = L_1; \quad (3.41)$$

$$T_2 = T_3, \ k_2 \frac{\partial T_2}{\partial z} = k_3 \frac{\partial T_3}{\partial z}, \ z = L_1 + L_2; \quad (3.42)$$

$$T_3^{in} = T_3^{out}, \ k_3 \frac{\partial T_3}{\partial z} = k_4 \frac{\partial T_3^{in}}{\partial z}, \ z = T_t, \frac{LX}{3} \leq x \leq \frac{2LX}{3}, \frac{LY}{3} \leq y \leq \frac{2LY}{3}; \quad (3.43)$$

$$T_3^{in} = T_3^{out}, \ k_i \frac{\partial T_3^{in}}{\partial z} = k_3 \frac{\partial T_3^{out}}{\partial z}, \ z = T_t + L_T, \frac{LX}{3} \leq x \leq \frac{2LX}{3}, \frac{LY}{3} \leq y \leq \frac{2LY}{3}; \quad (3.44)$$

$$T_3^{in} = T_3^{out}, \ k_i \frac{\partial T_3^{in}}{\partial y} = k_3 \frac{\partial T_3^{out}}{\partial y}, \ y = \frac{LY}{3}, \frac{2LY}{3}, T_t \leq z \leq T_t + L_T, \frac{LX}{3} \leq x \leq \frac{2LX}{3}; \quad (3.45)$$

$$T_3^{in} = T_3^{out}, \ k_3 \frac{\partial T_3^{in}}{\partial x} = k_3 \frac{\partial T_3^{out}}{\partial x}, \ x = \frac{LX}{3}, \frac{2LX}{3}, T_t \leq z \leq T_t + L_T, \frac{LY}{3} \leq y \leq \frac{2LY}{3}. \quad (3.46)$$
where $L_T$ is the length of the tumor in the $z$-direction. Because of the tumor's interior is saturated with gold nanoshells, we employ the following interfacial condition for the nanoshell-embedded region within the tumor as

$$T_3^{in} = T_3^{eff}, \quad k_i \frac{\partial T_3^{in}}{\partial n} = k_{eff} \frac{\partial T_3^{eff}}{\partial n}.$$  \hspace{1cm} (3.47)

Because the blood flow in the vein is oriented against the arterial flow, the entrance of the blood to the vein is located at the 7th level, and the blood temperature is equal to the surrounding tissue temperature. The initial conditions are assumed to be

$$T_l = T_0, \quad t = 0, \quad l = 1, 2, 3,$$ \hspace{1cm} (3.48)

where $T_1$, $T_2$, and $T_3$ are the tissue temperature in skin layer one, two and three, respectively, and $T_0$ is the initial temperature in the tissue.

In general cases, solving the above system analytically can be quite difficult, if not impossible, due to the complex blood vessel network, the heat source, and the multi-layered geometry. As a result, the above system must be solved numerically. In the next chapter, the numerical scheme used to solve our model will be described.
CHAPTER 4

NUMERICAL METHOD

The numerical procedure is outlined in Figure 4.1 and described in the following sections.

4.1 Finite Difference Method

The numerical method includes a stable finite difference scheme for solving the modified Pennes’ equation, the fourth-order Runge-Kutta method for blood flow equations, the least squares method for optimizing the laser power, and the computational procedure.

We denote \((u_i)^n_{ijk}\) and \(u_b\) as the numerical approximations of \(T_i(iAx, jAy, kAz, nAt)\) and \(T_b m\), respectively, where \(Ax, Ay, Az,\) and \(At\) are the spatial and temporal mesh sizes, and \(i, j, k\) are integers, \(0 < i < N_x, 0 < j < N_y, 0 < k < N_z\); so that \(N_x Ax = L_x, N_y Ay = L_y,\) and \(N_z Az = L_z, l = 1, 2, 3.\) In this mesh, we assume that \((u_3)^n_{ijk} = (u_b^m)^n_{ijk}\) when the grid point \((i, j, k)\) is in the \(m\)th level blood vessel. If \(T^m_b\) is known, then Eqs. (3.4)-(3.10) for blood temperature become first-order ordinary differential equations, which can be solved by using the fourth-order Runge-Kutta method [60]. Once the blood temperature \(u_b\) is determined, Eqs. (3.4)-(3.48) can be solved by using an unconditionally stable finite difference scheme that was developed in [6,7]. For convenience, we list those equations as follows.
\[
\frac{(s_l)^{n+1}_{ijk} + (s_l)^n_{ijk}}{2} = \left(1 + \frac{W_l^b C_l}{\rho_l C_l}\right) \frac{(u_l)^{n+1}_{ijk} + (u_l)^n_{ijk}}{2} + \frac{(u_l)^{n+1}_{ijk} - (u_l)^n_{ijk}}{\Delta t}, \quad (4.1)
\]

\[
\rho_l C_l \frac{(s_l)^{n+1}_{ijk} - (s_l)^n_{ijk}}{\Delta t} + W_l^b C_l \left[\frac{(u_l)^{n+1}_{ijk} + (u_l)^n_{ijk}}{2} - (u_b)_{out}\right]
= k_l (\delta_x^2 + \delta_y^2 + \delta_z^2) \frac{(u_l)^{n+1}_{ijk} + (u_l)^n_{ijk}}{2} + (Q_l)^{n+1/2}, \quad l = 1, 2, 3, \quad (4.2)
\]

where \((s_l)^n_{ijk}\) is the numerical approximation of \(S_l(i \Delta x, j \Delta y, k \Delta z, n \Delta t)\) defined as \(S_l = (1 + \frac{W_l^b C_l}{\rho_l C_l}) T_l + \tau \frac{\partial T_l}{\partial t}, \quad \delta_x^2 u_{ijk} = \frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{\Delta x^2}\) and so on, for the \(y\) and \(z\) directions. The discrete interfacial equations for (3.41) and (3.42) are assumed to be, for any time level,

\[
k_1 \frac{(u_1)^n_{ijN^*_z} - (u_1)^n_{ijN^*_z-1}}{\Delta z} = k_2 \frac{(u_2)^n_{ij1} - (u_2)^n_{ij0}}{\Delta z}, \quad (u_1)^n_{ijN^*_z} = (u_2)^n_{ij0}; \quad (4.3)
\]

\[
k_2 \frac{(u_2)^n_{ijN^*_z} - (u_2)^n_{ijN^*_z-1}}{\Delta z} = k_3 \frac{(u_3)^n_{ij1} - (u_3)^n_{ij0}}{\Delta z}, \quad (u_2)^n_{ijN^*_z} = (u_3)^n_{ij0}. \quad (4.4)
\]

Similar discrete interfacial equations can be obtained for Eqs. (3.43)-(3.46) and Eq. (3.47). The interfacial condition, Eq. (3.40), between the tissue and the lateral blood vessel is discretized as follows

\[
(u_3)^{n+1}_{ijk} = \frac{(u_3)^{n+1}_{i+1,j,k} + B_i \Delta x (u_3)^{n+1}_{i-1,j,k}}{(1 + B_i \Delta x)}, \quad (4.5)
\]

and similar discrete equations for the \(y\) and \(z\) directions, where the grid point \((i, j, k)\) is on the lateral walls of the blood vessel in the \(x, y, z\) directions, respectively. When the grid point \((i, j, k)\) is in the tissue, the initial and other boundary conditions are
discretized as follows

\[(u_i)^0_{ijk} = T_0; \]  
\[(u_i)^n_{0jk} = (u_i)^n_{ijk}; \]  
\[(u_i)^n_{N_xjk} = (u_i)^n_{N_xjk}; \]  
\[(u_i)^n_{i0k} = (u_i)^n_{i1k}; \]  
\[(u_i)^n_{iN_yk} = (u_i)^n_{iN_yk}; \]  
\[k_1 \frac{(u_1)_{ij0}^n - (u_1)_{ij1}^n}{\Delta z} = h[(u_1)_{ij0}^n - T_d]; \]  
\[(u_3)^n_{ijN_y} = (u_3)^n_{ijN_y-1}; \]

for any time level \( n \).

### 4.2 Least Squares Method

To determine the laser intensity \( P_0 \) for an optimal temperature distribution, we pre-specify the temperature elevations to be obtained at the center of the tumor and some locations in the perimeter on the skin surface. These locations are chosen because the highest temperature is assumed to be around the center of the tumor tissue, and it is necessary to have the temperature in the perimeter below a certain threshold to prevent damage the normal tissue. In addition, the temperature can be easily measured at these locations. By guessing an initial laser intensity \( P_0 \) and pre-specifying a laser exposure pattern, one can solve the above equations to obtain a temperature field in the entire 3D skin structure. The parameters for the 3D structure are given by Table 4.1. Once the temperatures, \( u_{cal} \), are calculated at the given locations \( (i = 0, 1, ..., M) \), a least squares approach can be employed to minimize the difference between the pre-specified temperature \( u_{pre} \) and the calculated temperature.
\( S(P_0) = \sum_{i=0}^{M} (u_{\text{pre}}^i - u_{\text{cal}}^i)^2, \quad i = 0, 1, ..., M. \) \( (4.11) \)

Table 4.1: Parameters used in Computations

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi = ( \alpha/k_3 )</td>
<td>95.23</td>
<td>NL_b^3, NW_b^3 (cm)</td>
<td>0.12</td>
</tr>
<tr>
<td>( L_1 ) (cm)</td>
<td>0.008</td>
<td>NL_b^4, NW_b^4 (cm)</td>
<td>0.1</td>
</tr>
<tr>
<td>( L_2 ) (cm)</td>
<td>0.2</td>
<td>NL_b^5, NW_b^5 (cm)</td>
<td>0.08</td>
</tr>
<tr>
<td>( L_T ) (cm)</td>
<td>0.25</td>
<td>NL_b^6, NW_b^6 (cm)</td>
<td>0.06</td>
</tr>
<tr>
<td>( L_3 ) (cm)</td>
<td>1.792</td>
<td>NL_b^7, NW_b^7 (cm)</td>
<td>0.04</td>
</tr>
<tr>
<td>( L_b^1 ) (cm)</td>
<td>1.0</td>
<td>( T_0 ) (°C)</td>
<td>34</td>
</tr>
<tr>
<td>( L_b^2 ) (cm)</td>
<td>0.72</td>
<td>( T_{\text{in}} ) (°C)</td>
<td>37</td>
</tr>
<tr>
<td>( L_b^3 ) (cm)</td>
<td>0.5</td>
<td>( \Delta x ) (cm)</td>
<td>0.01, 0.005</td>
</tr>
<tr>
<td>( L_b^4 ) (cm)</td>
<td>0.36</td>
<td>( \Delta y ) (cm)</td>
<td>0.01, 0.005</td>
</tr>
<tr>
<td>( L_b^5 ) (cm)</td>
<td>0.26</td>
<td>( \Delta z ) (cm)</td>
<td>0.001</td>
</tr>
<tr>
<td>( L_b^6 ) (cm)</td>
<td>0.18</td>
<td>( \sigma ) (cm)</td>
<td>0.1</td>
</tr>
<tr>
<td>( L_b^7 ) (cm)</td>
<td>0.14</td>
<td>( \tau ) (s)</td>
<td>20</td>
</tr>
<tr>
<td>( L, L_Y ) (cm)</td>
<td>1.62</td>
<td>( L, L_Z ) (cm)</td>
<td>2.6</td>
</tr>
<tr>
<td>NL_b^1, NW_b^1 (cm)</td>
<td>0.2</td>
<td>( \varepsilon_l )</td>
<td>0.4, 0</td>
</tr>
<tr>
<td>NL_b^2, NW_b^2 (cm)</td>
<td>0.16</td>
<td>( T_a ) (°C)</td>
<td>25</td>
</tr>
<tr>
<td>( \varepsilon_{\text{laser}} )</td>
<td>0.01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Minimizing \( S(P_0) \), one obtains

\[
\frac{d}{dP_0}S(P_0) = -2 \sum_{i=0}^{M} \left[ \frac{d(u_{\text{cal}}^i)}{dP_0} \right] (u_{\text{pre}}^i - u_{\text{cal}}^i) = 0, \quad i = 0, 1, ..., M. \] \( (4.12) \)

Thus, a new \( P_0 \) can be calculated iteratively as follows \([5, 61]\)

\[
P_0^{(j+1)} = P_0^{(j)} + (X^tX + \alpha^*I)^{-1}X^t(u_{\text{pre}} - u_{\text{cal}}), \] \( (4.13) \)

where \( \alpha^* \) is a relaxation parameter, \( I \) is an identity matrix, and \( X \) is the sensitivity coefficient matrix, which is an \((M + 1) \times 1\) vector

\[
X = \left[ \frac{\partial(u_{\text{cal}}^0)}{\partial P_0}, \frac{\partial(u_{\text{cal}}^1)}{\partial P_0}, ..., \frac{\partial(u_{\text{cal}}^M)}{\partial P_0} \right]^t, \] \( (4.14) \)
and
\[ u_{pre} = \left[u_{pre}^0, u_{pre}^1, \ldots, u_{pre}^M\right]^T, \quad u_{cal} = \left[u_{cal}^0, u_{cal}^1, \ldots, u_{cal}^M\right]^T. \] (4.15)

### 4.3 Computational Procedure

To reduce the computational time, we solve for \((s_t)^{n+1}_{ijk}\) from Eq. (4.1) and then substitute it into Eq. (4.2). This process gives

\[
\frac{\rho_t C_t}{\Delta t} \left[ (1 + \tau \frac{W_{b}^l C_{b}^l}{\rho_t C_t} + \frac{2\tau}{\Delta t})(u_t)^{n+1}_{ijk} + (1 + \tau \frac{W_{b}^l C_{b}^l}{\rho_t C_t} - \frac{2\tau}{\Delta t})(u_t)^n_{ijk} - 2(s_t)^n_{ijk} \right] \\
+ W_{b}^l C_{b}^l \left[ \frac{(u_t)^{n+1}_{ijk} - (u_t)^n_{ijk}}{2} - (u_b)_{out} \right] \\
= k_l(\delta_x^2 + \delta_y^2 + \delta_z^2) \frac{(Q_t)^{n+1/2}_{ijk}}{2} + (Q_t)^{n+1/2}_{ijk}, \quad l = 1, 2, 3. \tag{4.16}
\]

We further employ a preconditioned Richardson iteration for \((u_t)^{n+1}_{ijk}\) as follows [5, 7]

\[
L^1_{pre}[(u_t)^{n+1}_{ijk}]^{(l+1)} = L^1_{pre}[(u_t)^{n+1}_{ijk}]^{(l)} - \omega \left\{ \frac{\rho_t C_t}{\Delta t} \left[ (1 + \tau \frac{W_{b}^l C_{b}^l}{\rho_t C_t} + \frac{2\tau}{\Delta t})(u_t)^{n+1}_{ijk} \right] + (1 + \tau \frac{W_{b}^l C_{b}^l}{\rho_t C_t} - \frac{2\tau}{\Delta t})(u_t)^n_{ijk} + W_{b}^l C_{b}^l \left[ \frac{(u_t)^{n+1}_{ijk} + (u_t)^n_{ijk}}{2} - (u_b)_{out} \right] \\
- k_l(\delta_x^2 + \delta_y^2 + \delta_z^2) \frac{(Q_t)^{n+1/2}_{ijk} + (Q_t)^{n+1/2}_{ijk}}{2} \right\}, \quad l = 0, 1, 2, \ldots \tag{4.17}
\]

where \(\omega\) is the relaxation factor, \(l = 1, 2, 3\), and the preconditioner is chosen to be

\[
L^1_{pre} = 1 + \tau \frac{W_{b}^l C_{b}^l}{\rho_t C_t} + \frac{2\tau}{\Delta t} + \frac{W_{b}^l C_{b}^l \Delta t}{2\rho_t C_t} \left[ \frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right] - \frac{k_l \Delta t}{2\rho_t C_t} \delta_z^2. \tag{4.18}
\]

As such, the linear system, Eq. (4.16), can be transferred into a tridiagonal linear system, Eqs. (4.17) - (4.18). If we denote \((u_3)^n_{ijk} = (u_b^n)_{ijk}\) when the grid point \((i, j, k)\) is in the \(m\)th level blood vessel, then the Thomas algorithm [62] can be used
line by line along the z-direction. Thus, our computational procedure for predicting the skin's temperature distribution in a 3-D skin structure induced by laser-heating can be described as follows:

- **Step 1.** Pre-specify the temperature levels $u_{pre}^i$ at given $(M + 1)$ grid points $i = 0, 1, ..., M$; on the skin surface and the tumor's center, and pre-specify the laser exposure pattern for obtaining these pre-specified temperatures.

- **Step 2.** Guess an initial laser intensity $P_0$ and its small increment $P_0 + \Delta P_0$.

- **Step 3.** Guess the wall temperature of the blood vessel $u_{w}^m$ and obtain the blood temperature $u_{b}^m$ by solving Eqs. (3.4) - (3.10) using the fourth-order Runge-Kutta method. At each program loop, we first calculate the temperature of each level of blood vessel, where the coordinates of blood vessels are independent of the coordinates of consideration of skin domain.

- **Step 4.** Update the wall temperature of the blood vessel, $u_{w}^m$, by Eq. (4.5).

- **Step 5.** Obtain the temperature distribution $u^{n+1}$ in the entire 3-D skin structure by solving Eqs. (4.17)-(4.18) coupled with the interfacial equations, the initial and boundary conditions, Eqs. (4.3)-(4.10).

- **Step 6.** Repeat steps 3-5 until a convergent solution, $u$, at time level $n + 1$ is obtained.

- **Step 7.** Determine a new $P_0$ based on Eqs. (4.13)-(4.15).

- **Step 8.** Repeat the computation until the following criterion for convergence is satisfied:

$$\frac{|S((P_0)^{(J+1)}) - S((P_0)^{(J)})|}{S((P_0)^{(J+1)})} < \varepsilon_{laser}.$$  \hspace{1cm} (4.19)
• **Step 9.** With the optimized $P_0$, repeat steps 3-6 for the given time period.

In the computation, the temperature of blood vessels is first obtained, then the wall temperature of the blood vessel $u_w^n$ is updated, and the tissue temperature of the 3D domain is calculated. The coordinates of the blood vessels are independent of the coordinates of the tissue domain. Also, the vessels in the arterial system are chosen to coincide with the direction of the blood flow. As such for the arterial system, the first level artery coordinate coincides with the opposite of the x-direction of the tissue domain. The second level branches into two vessels. One is along the y-direction and the other opposite to the x-direction. The Level 3 artery consists of four parts, with two parts running lengthwise in the z-direction, and the other two run opposite of the z-direction. At the end of each of the arteries in Level 3, each of the vessels bifurcate into two more vessels, where four parts run lengthwise in the x-direction and the other four parts run in the opposite direction resulting in eight vessels in Level 4. Level 5 has sixteen vessels that are bifurcating at the ends of the Level 4 vessels; eight vessels run along the y-direction and eight run opposite to the y-direction. Level 6 has thirty-two individual vessels that branch off the ends of the Level 5 vessels. Sixteen of those vessels run in the z-direction while another sixteen run opposite of the z-direction. Finally, the seventh level of the arterial system has sixty-four vessels; thirty-two vessels run along the x-direction while the other thirty-two run opposite of the x-direction. The coordinates of the veins are similar to the corresponding level of arteries, except the direction of the blood flow is reversed. The algorithm is demonstrated by the following flow chart (Figure 4.1).
Guess the initial laser and its small increment, $P_0$ and $P_0 + \Delta P_0$, respectively. Prespecify temperatures in $u_{\mu}^\alpha$.

Calculate new laser intensity $P_1$ with Eqs. (4.13)-(4.15). $P_0 \leftarrow P_1$

If Eq. (4.19) is satisfied, go down, otherwise go right.

If the converged $u_{\mu}^{n+1}$ is obtained, and $t = T_{total}$, then go down to exit, otherwise go right to repeat.

Obtain $u_{\mu}^{n+1}$ in the entire 3D skin structure except for the area of blood vessels, by solving Eqs. (4.17) and (4.18) coupled with the interfacial and boundary equations, Eqs. (4.3)-(4.10).

Figure 4.1: Flow chart for the computational procedure
CHAPTER 5

NUMERICAL EXAMPLE

5.1 Example Description

To illustrate the present method, we chose a 3-D skin structure with the dimensions 1.62 cm \times 1.62 cm \times 2.6 cm and with embedded seven-level dendritic countercurrent vascular network, as shown in Figure 3.1. The values of the biological parameters used are listed in Table 3.1. We assumed heat convection occurring on the skin surface \((h = 0.001 \text{W/cm}^2)\) where the surface is exposed to an ambient temperature of 25 °C. The thermal relaxation time was taken to be \(\tau = 20\) s [14]. Other parameters used in these computations are listed in Tables 4.1 and 5.1. The tumor region is in the volume of \(0.54 \text{cm} \leq x \leq 1.08 \text{cm}, 0.54 \text{cm} \leq y \leq 1.08 \text{cm}, 0.240 \text{cm} \leq z \leq 0.490 \text{cm}\) and the nanoshell-embedded region within the tumor exists in the volume of \(0.65 \text{cm} \leq x \leq 0.97 \text{cm}, 0.65 \text{cm} \leq y \leq 0.97 \text{cm}, 0.315 \text{cm} \leq z \leq 0.415 \text{cm}\).

The pattern of the laser irradiation was designed as follows. The laser with beam width of 0.1 cm was focused on the center of the skin surface. The pre-specified temperatures, \(u_{\text{pre}}\), for the center of the tumor and four perimeters of the skin’s surface is defined as \(u_{\text{pre}} = [50.0, 34.0, 34.0, 34.0, 34.0]\). When the tumor's center temperature rose to 50 °C, the laser was turned off; the laser was then turned on when the temperature at the center of the tumor decreases by 4 °C. Our goal was to
raise the tumor’s central temperature from 34 °C to 50 °C within 10 seconds while maintaining the minimum temperature of the tumor region at 42 °C, which is the minimum temperature required for hyperthermia to occur [63,64]. Once the optimized laser intensity was obtained, we used it to obtain the temperature distribution for 600 seconds. We assume our laser to be exciting the nanoshells at 633 nm to exhibit peak absorption and minimal scattering [3,4] and that the number of nanoshells occupying a volume of 0.01 cm × 0.01 cm × 0.001 cm is $7.0 \times 10^9$ particles/cm$^3$. The parameters for the nanoshells are given by Table 5.1.

Table 5.1: Parameters used in obtaining local heat generation spectrum being generated by the gold nanoshells

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_T$ (Particles/cm$^3$)</td>
<td>$7.0 \times 10^9$</td>
</tr>
<tr>
<td>$Q_{k\lambda}$ (efficiency)</td>
<td>0.059492</td>
</tr>
<tr>
<td>$Q_{\sigma\lambda}$ (efficiency)</td>
<td>0.0092968</td>
</tr>
<tr>
<td>$R_i$ (cm)</td>
<td>$1.6 \times 10^{-6}$</td>
</tr>
<tr>
<td>$R_o$ (cm)</td>
<td>$2.2 \times 10^{-6}$</td>
</tr>
<tr>
<td>$D_p$ (cm)</td>
<td>$4.4 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\lambda_{ep}$ (cm)</td>
<td>$3.614 \times 10^{-6}$</td>
</tr>
<tr>
<td>$\sigma_{md-\lambda}$ (1/cm)</td>
<td>66.9</td>
</tr>
<tr>
<td>$k_{md-\lambda}$ (1/cm)</td>
<td>0.1</td>
</tr>
<tr>
<td>$\omega_{\lambda}$</td>
<td>0.9984</td>
</tr>
<tr>
<td>$\epsilon_{\lambda}$</td>
<td>1.0</td>
</tr>
<tr>
<td>$\beta_{tot-\lambda}$</td>
<td>67.0073</td>
</tr>
<tr>
<td>$\omega_{s-\lambda}$ (1/cm)</td>
<td>$9.8952 \times 10^{-4}$</td>
</tr>
<tr>
<td>$k_{s-\lambda}$ (1/cm)</td>
<td>$6.3 \times 10^{-3}$</td>
</tr>
<tr>
<td>$\alpha_{gold}$ (1/cm)</td>
<td>5.76969</td>
</tr>
</tbody>
</table>

5.2 Results

We used this model to simulate two conditions in the mesh of 162 × 162 × 2600: one where nanoshells exist in part of the tumor region ($\epsilon_t = 0.4$) and the other where
no nanoshells exist in the tumor region ($\epsilon_f = 0$). For the study where nanoshells are embedded in the tumor region, we optimized $P_0$ based on the algorithm described in the previous section with an initial value of $P_0$ as 1.29 W/cm and the value of $\Delta P_0$ to be 1 % of $P_0$. Figure 5.1 shows $P_0$ and the least-squares sum versus iteration, respectively. $P_0$ is convergent to 1.77 W/cm. We then used this convergent value of $P_0$ to compute the temperature distribution in the 3-D skin structure. Figures 5.2-5.7 show the contours of the temperature distribution of the skin structure with an embedded seven-level dendritic countercurrent vascular network along with a tumor with injected golden nanoshells appearing in the subcutaneous region at $t = 0$, 200, 400, and 600 seconds. Figures 5.2-5.7 show that we have achieved the objective, at 600 seconds, of raising the minimum temperature throughout the tumor to 42 °C.
Figure 5.1: Number of iterations versus (a) laser intensity ($P_0$) and (b) least squares sum ($S(P_0)$)
Figure 5.2: Contours of temperature distribution in the yz-plane at $x = 0.54$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$, while (e) represents the plane where the results were observed. Here, levels 2-4 of the arterial system are visible.
Figure 5.3: Contours of temperature distribution in the yz-plane at \( x = 0.81 \) cm at (a) \( t = 0 \) s, (b) \( t = 200 \) s, (c) \( t = 400 \) s, and (d) \( t = 600 \) s, while (e) represents the plane where the results were observed. Here, levels 1, 5, 6, 7 of the arterial system and the center of the tumor are visible.
Figure 5.4: Contours of temperature distribution in the yz-plane at $x = 1.08$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, level 1 of the arterial system is visible.
Figure 5.5: Contours of the temperature distribution in the xz-plane at \( y = 0.54 \text{ cm} \) at (a) \( t = 0 \text{ s} \), (b) \( t = 200 \text{ s} \), (c) \( t = 400 \text{ s} \), and (d) \( t = 600 \text{ s} \), while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.
Figure 5.6: Contours of the temperature distribution in the xz-plane at \( y = 0.81 \) cm at (a) \( t = 0 \) s, (b) \( t = 200 \) s, (c) \( t = 400 \) s, and (d) \( t = 600 \) s, while (e) represents the plane where the results were observed. Here, levels 1 and 2 of the arterial system and the center of the tumor are visible.
Figure 5.7: Contours of the temperature distribution in the xz plane at y = 1.08 cm at (a) t = 0 s, (b) t = 200 s, (c) t = 400 s, and (d) t = 600 s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.
For the study without nanoshells, again we optimized $P_0$ based on the algorithm described in the previous section with an initial value of $P_0$ as 1.29 W/cm and the value of $\Delta P_0$ to be 1 % of $P_0$. Figure 5.8 shows $P_0$ and the least-squares sum versus iteration, respectively. $P_0$ converges to 11.32 W/cm. Compared with Figure 5.1, we can see that without nanoshells, a larger laser power is needed in order to achieve the same criterion as the nanoshell case. Figures 5.9-5.14 show the contours of the temperature distribution of the skin structure with an embedded seven-level dendritic countercurrent vascular network along with a tumor with injected golden nanoshells appearing in the subcutaneous region at $t = 0$ s, 200 s, 400 s, and 600 s. Figures 5.2-5.7 and Figures 5.9-5.14 were plotted at the same faces and blood vessel locations, except that in Figures 5.9-5.14 contains no nanoshells in the tumor region. In contrast to the study with nanoshells, we can see through the center of the tumor does heat up to the desired range but the extremities of the tumor fail to reach the desired minimum temperature of 42 °C (see Figures 5.11 and 5.12.) Consequently, the substantially higher laser intensity may be damaging the healthy tissue around the tumor (see Figures 5.10 and 5.13.) This result indicates the advantage of using nanoshells to heat up a tumor region quickly while keeping the surrounding healthy tissue from damage.
Figure 5.8: Number of iterations versus (a) laser intensity ($P_0$) and (b) least squares sum ($S(P_0)$)
Figure 5.9: Contours of temperature distribution in the yz-plane at $x = 0.54$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, levels 2-4 of the arterial system are visible.
Figure 5.10: Contours of temperature distribution in the yz-plane at x = 0.81 cm at (a) t = 0 s, (b) t = 200 s, (c) t = 400 s, and (d) t = 600 s, while (e) represents the plane where the results were observed. Here, levels 1, 5, 6, 7 of the arterial system and the center of the tumor are visible.
Figure 5.11: Contours of temperature distribution in the yz-plane at $x = 1.08$ cm at (a) $t = 0$ s, (b) $t = 200$ s, (c) $t = 400$ s, and (d) $t = 600$ s, while (e) represents the plane where the results were observed. Here, level 1 of the arterial system is visible.
Figure 5.12: Contours of the temperature distribution in the xz-plane at \( y = 0.54 \) cm at (a) \( t = 0 \) s, (b) \( t = 200 \) s, (c) \( t = 400 \) s, and (d) \( t = 600 \) s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.
Figure 5.13: Contours of the temperature distribution in the xz-plane at \( y = 0.81 \) cm at (a) \( t = 0 \) s, (b) \( t = 200 \) s, (c) \( t = 400 \) s, and (d) \( t = 600 \) s, while (e) represents the plane where the results were observed. Here, levels 1 and 2 of the arterial system and the center of the tumor are visible.
Figure 5.14: Contours of the temperature distribution in the xz plane at y = 1.08 cm at (a) t = 0 s, (b) t = 200 s, (c) t = 400 s, and (d) t = 600 s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.
Examining the results more closely, Figures 5.2 and 5.9 represent the yz-plane at the point \( x = 0.54 \) cm (54 grid points away from the origin in the x-direction). Looking at the figures, levels 2 and 3 of the arterial system are visible and are heating up the surrounding tissue. Figures 5.2b - 5.2d appear to have a temperature distribution in the vicinity of the tumor that is smooth. This observation leads to the fact that there is no presence of the venous system near the tumor at this location. Furthermore, the yz-plane at \( x = 0.54 \) cm is one of the two faces of the tumor along the x-coordinate. To this end, the heat is distributing from the center of the tumor to the edges of the tumor, thus leading to a cooler temperature field in the tumor region than results in Figures 5.3, 5.6, 5.10, 5.13. Figures 5.15 and 5.16 are similar results, but are at different time steps. Figure 5.9 is at the same location as Figure 5.2, but with no nanoshells present.
Figure 5.15: Contours of temperature distribution in the yz-plane at $x = 0.54$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 2-4 of the arterial system are visible.
Figure 5.16: Contours of temperature distribution in the yz-plane at $x = 0.54\, \text{cm}$ at (a) $t = 50\, \text{s}$, (b) $t = 100\, \text{s}$, (c) $t = 300\, \text{s}$, and (d) $t = 500\, \text{s}$, while (e) represents the plane where the results were observed. Here, levels 2-4 of the arterial system are visible.
Figure 5.3 is along the yz-plane at x = 0.81 cm (81 grid points away from the origin in the x-direction); the temperature distribution at the center of the tumor is observable. Level 1 of the arterial system is visible as well; however, Levels 5, 6, and 7 are visible as time progresses, denoted by Figures 5.3b, 5.3c, and 5.3d at t = 200 s, 400 s, and 600 s, respectively. With the entirety of the nanoshell region shown along the y-coordinate, it is observed that the nanoshells are able to retain the heat generated from the laser and slowly diffuse to the extremities of the tumor, thus having the highest temperature levels than the other contours in the x-direction. In Figure 5.10, the same vascular structure is observed with no nanoshells present. Figure 5.17 and 5.18 are similar results, but at different time steps with and without nanoshells, respectively.
Figure 5.17: Contours of temperature distribution in the yz-plane at x = 0.81 cm at (a) t = 50 s, (b) t = 100 s, (c) t = 300 s, and (d) t = 500 s, while (e) represents the plane where the results were observed. Here, levels 1, 5, 6, 7 of the arterial system and the center of the tumor are visible.
Figure 5.18: Contours of temperature distribution in the yz-plane at $x = 0.81$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 1, 5, 6, 7 of the arterial system and the center of the tumor are visible.
Figure 5.4 is the contour of the yz-plane at the location \( x = 1.08 \text{ cm} \) (108 grid points away from the origin along the x-coordinate.) This particular plane shows the second face of the tumor in the x-direction. In this figure, Level 1 of the artery is clearly visible in Figure 5.4a. However, as more results are examined, Figure 5.4b has a rigid temperature distribution around the Level 1 artery. Below the artery, Level 1 of the venous system is visible. Level 7 of the venous system is also visible to the left and right of the Level 1 artery at \( t = 200 \text{ s}, 400 \text{ s}, \text{ and } 600 \text{ s} \). Due to Figure 5.4 being one of the faces of the tumor, the heat distribution has to diffuse to the extremities of the region. This results in a lower temperature distribution in comparison to Figure 5.3. Figure 5.19 represents the same configuration as Figure 5.4, but with results at different time steps. Figures 5.11 and 5.20 represent the same yz-plane at \( x = 1.08 \text{ cm} \) except with no nanoshells are present.
Figure 5.19: Contours of temperature distribution in the yz-plane at \( x = 1.08 \text{ cm} \) at (a) \( t = 50 \text{ s} \), (b) \( t = 100 \text{ s} \), (c) \( t = 300 \text{ s} \), and (d) \( t = 500 \text{ s} \), while (e) represents the plane where the results were observed. Here, level 1 of the arterial system is visible.
Figure 5.20: Contours of temperature distribution in the yz-plane at \( x = 1.08 \) cm at (a) \( t = 50 \) s, (b) \( t = 100 \) s, (c) \( t = 300 \) s, and (d) \( t = 500 \) s, while (e) represents the plane where the results were observed. Here, level 1 of the arterial system is visible.
Figure 5.5 represents the contour of the xz-plane at $y = 0.54$ cm (54 grid points away from the origin in the y-direction.) In this contour, the first face of the tumor in the y-direction is observable. Levels 2, 6, and 7 of the arterial system are visible at all time steps; however, as time progresses the heat convection from the Level 1 artery is visible along the depth of 1.5 cm in the z-direction. Also visible are parts of the venous system to the right of the visible arterial system. With $y = 0.54$ cm being the face of the tumor, it is observable that heat is diffusing to the edge of the tumor, which results in a lower temperature distribution than what would be observed in Figure 5.6. Figure 5.21 has similar results with nanoshells on the xz-plane at $y = 0.54$ cm, but are taken at different time steps. Figures 5.12 and 5.22 represents the same location with the same vascular network with no nanoshells present.
Figure 5.21: Contours of the temperature distribution in the xz-plane at $y = 0.54 \text{ cm}$ at (a) $t = 50 \text{ s}$, (b) $t = 100 \text{ s}$, (c) $t = 300 \text{ s}$, and (d) $t = 500 \text{ s}$. Here, levels 2, 6, and 7 of the arterial system are visible.
Figure 5.22: Contours of the temperature distribution in the xz-plane at $y = 0.54$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.
Figure 5.6 is the contour of the xz-plane at \( y = 0.81 \) cm (81 grid points away from the origin along the y-coordinate.) This contour shows the center of the tumor in the y-direction, where the nanoshell region is observable. Level 1 of the arterial system is visible in all cases. However, Level 1 of the venous system becomes visible beneath the Level 1 artery in Figures 5.6b - 5.6d. With the nanoshells present, the highest temperature levels will be on this plane, thus the other contours in the y-direction will have lower temperature distributions. Figure 5.23 represents the same configuration of vessels and nanoshells, but with results at different time steps at \( y = 0.81 \) cm. Figures 5.13 and 5.24 have the same vascular configuration and tumor placement except no nanoshells are present.
Figure 5.23: Contours of the temperature distribution in the xz-plane at $y = 0.81$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 1 and 2 of the arterial system and the center of the tumor are visible.
Figure 5.24: Contours of the temperature distribution in the xz-plane at $y = 0.81$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 1 and 2 of the arterial system and the center of the tumor are visible.
Figure 5.7 is the contour of the xz-plane at \( y = 1.08 \) cm (108 grid points from the origin in the y-direction.) Because of the symmetrical design of the vascular network, more branches of Levels 2, 6, and 7 are observable. Figures 5.7b - 5.7d also show convection from the Level 1 artery as time progresses in the simulation. Also noticeable in the figures, the temperature distribution becomes rigid due to the venous system conducting heat to transport out of the system. The contour also shows the second face of the tumor in the y-direction. With this contour showing the extremities of the tumor in this direction and with the heat diffusing from the center, the temperature distribution will be lower than that of Figure 5.7. Figure 5.25 represents the same vascular network configuration and will have similar results as Figure 5.7, except at different time steps. Figures 5.14 and 5.26 have the same vascular configuration except there are no nanoshells present.
Figure 5.25: Contours of the temperature distribution in the xz plane at \( y = 1.08 \text{ cm} \) at (a) \( t = 50 \text{ s} \), (b) \( t = 100 \text{ s} \), (c) \( t = 300 \text{ s} \), and (d) \( t = 500 \text{ s} \), while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.
Figure 5.26: Contours of the temperature distribution in the xz plane at $y = 1.08$ cm at (a) $t = 50$ s, (b) $t = 100$ s, (c) $t = 300$ s, and (d) $t = 500$ s, while (e) represents the plane where the results were observed. Here, levels 2, 6, and 7 of the arterial system are visible.
Finally, to show that our algorithm is independent of grid size, three different meshes of $162 \times 162 \times 2600$, $162 \times 324 \times 2600$, and $324 \times 162 \times 2600$ were employed in the computation. Figure 5.27 shows the four temperature profiles selected at $t = 50$ s. No significant differences were present between these solutions, implying that our scheme is grid-independent.
Figure 5.27: Temperature profile at $t = 50$ s along lines (a) along the z direction from the top to bottom at the central point of the skin surface, (b) at the point (.81 cm, .81 cm, .365 cm) over time, (c) $y = .81$ cm, and (d) $x = .81$ cm on the surface of the skin, while (e)-(h) are the locations where the results were observed.
5.3 Discussion

The computational complexity of the model presented reflects the model's structure (a 3-D triple-layered skin structure with embedded countercurrent vascular network with tumor and nanoshells). Since the model is three-dimensional, several three-dimensional arrays were implemented to store the information at each coordinate in the model's structure. The arrays are information specific and store information for each process (e.g., volumetric spatial heat, blood temperatures, blood vessel boundary temperatures, tissue temperatures, thermal lag approximations). The three-dimensional arrays lead to nested for-loops in each direction to store an instruction at each coordinate. Thus, as a result of three for-loops, the computational complexity is a product of the lengths of the three directions (x,y,z). The vascular system's temperature is calculated independently; it is later stored within the tissue temperature array at its appropriate locations. Since the lengths of the x-direction and the y-direction will be less than or equal to the length of the z-direction, the computational complexity will be $O(n^3)$.

Originally, it was proposed to raise the tumor's center temperature to 46 °C. However, when the calculations were performed the extremities of the tumor were not reaching the 42 °C threshold to achieve hyperthermia. The resulting decision was to raise the pre-specified temperature at the tumor's center high enough such that the extremities of the tumor would reach 42 °C. Through several simulations, it was observed that the difference between the center point of the tumor and the extremities along the center contours at $x = 0.81 \text{ cm}$ and $y = 0.81 \text{ cm}$ had a difference of
~ 8 °C on average at the end of each simulation. Because of this observation, 50 °C was selected as the pre-specified temperature at the center of the tumor.

The laser's irradiation has different effects on the heating of the center of the tumor when nanoshells are present. With the 1% increment of $P_0$ added for the second part of the optimization process, it was shown that the temperature would be affected by 0.2 °C on average in 10 seconds. Without nanoshells, the incrementation of the laser has an effect of the center of the tumor by 0.03 °C. This observation could be useful if a pattern can be obtained that relates the amount of nanoshells at a particular point of the tumor with the porosity and laser intensity to possibly speed up convergence to an optimal laser intensity.

For the time interval to optimize the laser intensity, it was initially proposed to allow the optimization period to be 600 seconds. Because of the implicit scheme used, the calculations for each optimization step for one iteration ($P_0$ and $P_0 + \Delta P_0$) would take 10 hours to complete. With eight iterations to convergence for the study with nanoshells and four iterations for the study without nanoshells, it is hypothesized that the simulation would take at least 90 hours and 50 hours, respectively, to achieve a solution. With the initial approach, it would also be difficult to observe the behavior of the nanoshells without the effects of the laser. With the current approach (optimization time = 10 seconds), we are able to observe the nanoshells' effect on the temperature distribution of the tumor while the laser is inactive.

Without the use of the compact finite-difference method, our model is reasonably accurate with second order accuracy in time and space. Despite having accuracy to this degree, there are some errors introduced due to underlying assumptions to
simplify calculations. Neglecting the reflection of the nanoshells would be one such error. Although the values for the reflections of the nanoshells are indeed small [3, 4], it could have an effect on the heating distribution of the tumor region. Another error that could affect the model is the denaturing of tissues [23]. As the tissue itself is affected by the heat, the properties of the tissue change which would effect the heating distribution of the tumor region. In our model, the properties of the tissue were kept at a constant which simplified calculations. Not accounting for the biological lag of the temperature gradient is also another source of error; however, this would fundamentally change the problem as the BHTE would be considered a dual-phase lag equation. To fully understand and determine the temperature distribution of this type of therapy, we would need to account for the biological lag, the changing properties of tissue as exposed to heat, and to account for calculations that could be negligible to be as accurate as possible to a real world scenario.

Hyperthermia therapy has been achieved in different ways such as using a laser coupled with the effects of nanoparticles [28, 42] or electromagnetic stimulation [65, 66]. However, most of the literature has focused on simulating treatment using a laser to treat tumors by irradiating the region embedded with nanoparticles. Sazgarnia et al. [28] examined the use of a laser to heat up nanoparticle-mediated prostate tissue using short-pulses. The model was developed in COMSOL and used the Finite Element Method while being based on the Pennes' BHTE; however, the model does not consider thermal lag from the heat flux nor does it consider blood vessels in the tissue region. Sanchez [23] developed a model for exposing healthy human tissue that accounted for the denaturing of tissue in the presence of a heat source. This model
was based on the Pennes’ BHTE; however, it did not account for thermal lag of the heat flux nor did it consider blood vessels in the tissue domain. In a clinical scenario, hyperthermia is usually achieved by the use of an interstitial laser or applicator to heat a small region, known as local hyperthermia. However, local hyperthermia is also combined with other treatment methods, such as radiation therapy, to treat a region in the chance that the tumor begins to metastasize to other local regions. In local hyperthermia therapy, tumor sizes have been shown to decrease by as much as 50% in volume when combined with radiation treatment; in comparison when treated with just radiation, treatment took nearly 12 weeks to achieve a similar result. There are some limits with local hyperthermia treatment as it is dedicated to deal with small tumors up to 6 cm in diameter [67].

The BHTE equation presented (Eq. (3.11)) was based on the Maxwell-Cattaneo flux law

\[ \vec{q}(x, y, z, t) = -k \nabla T(x, y, z, t), \]  

where \( \vec{q} \), \( k \), and \( T \) represents the thermal flux vector, thermal conductivity, and tissue temperature, respectively [7]. The modification to transform the Pennes’ equation into the modified Pennes’ equation that accounts for thermal lag is the result of using the Taylor series for time on the thermal flux

\[ \vec{q}(x, y, z, t + \tau) = -k \nabla T(x, y, z, t), \]
which leads to the following derivation for the single-phase lag modification of the Maxwell-Cattaneo flux law

\[
\left( 1 + \tau \frac{\partial}{\partial t} \right) \bar{q}(x, y, z, t) = -k \nabla T(x, y, z, t) \tag{5.3}
\]

This derivation leads to Eq. (3.11). To modify Eq. (3.11) to account for biological lag, we would employ another Taylor series with another lag term along the temperature gradient in Eq. (5.2)

\[
\bar{q}(x, y, z, t + \tau_q) = -k \nabla T(x, y, z, t + \tau_t) \tag{5.4}
\]

which leads to the following derivation

\[
\left( 1 + \tau_q \frac{\partial}{\partial t} \right) \bar{q}(x, y, z, t) = -k \left( 1 + \tau_t \frac{\partial}{\partial t} \right) \nabla T(x, y, z, t) \tag{5.5}
\]

where \( \tau_q \) and \( \tau_t \) represent the thermal lags of the heat flux and biological lag, respectively. This derivation leads to the dual-phase lag BHTE as follows

\[
\rho C_l \left( \frac{\partial T_l}{\partial t} + \tau_q \frac{\partial^2 T_l}{\partial t^2} \right) + \tau_q W^l B^l \frac{\partial T_l}{\partial t} = k_l \left( \frac{\partial^2 T_l}{\partial x^2} + \frac{\partial^2 T_l}{\partial y^2} + \frac{\partial^2 T_l}{\partial z^2} \right) + W^l B^l (T_{out} - T_l) + k_l \tau_l \frac{\partial}{\partial t} \left( \frac{\partial^2 T_l}{\partial x^2} + \frac{\partial^2 T_l}{\partial y^2} + \frac{\partial^2 T_l}{\partial z^2} \right) + Q_l, \quad l = 1, 2, 3. \tag{5.6}
\]

To get the results presented, measurements had to be taken within the in-house C++ code and stored within a text file at certain time increments. The text files would then be used as input into Matlab and with the graphs as the output. The results were taken when \( t = 0 \) s, 1 s, and 10 s to check the model’s operation in the
initial condition and after 10 sequential time steps. The measurement at 10 seconds was to observe the temperature distribution after the laser raised the center of the tumor to the desired level during the optimization process. After such measurements were taken, evenly distributed measurements occurred at $t = 50$ s and continued for every 50 s until the given time period has been reached.

With the project complete, an unexpected consequence of our model was during the testing phase without the nanoshells present in the model. Personally, I wasn't expecting for the laser to not have as much of an impact as the results show in the temperature distribution of the tumor; I was expecting the temperature levels to be slightly higher and more disperse similar to the study with nanoshells.

As a result of our study, new problems have been generated. With the laser warming up the tissue in the nanoshell case, it would be worth considering the use of electromagnetic stimulation to heat up the tumor region in consideration with the nanoshells presented. Another worthy problem would to be to determine an optimal amount of nanoshells with an optimal laser level to increase the efficiency of the treatment and include factors such as uneven distributions of nanoshells (e.g., nanoshells leaving the tumor).

Modeling the treatment of skin cancer and trying to treat skin cancer in a clinical setting are similar, yet different. In the model presented, we are considering a region of tissue with a tumor region, but instead of having a buffer zone on the boundaries (a continuation of tissue) we use an insulation boundary condition, Eq. (3.38), to focus on the heat transfer of the vascular system to transport heat out of the medium. Another difference between our model and the use in a clinical setting is the
use of an anesthetic to alleviate pain in the treatment region. In the model, the possible pain of the patient was not considered for the model as the nervous system was ignored. However, in a clinical setting anesthesia would be used to alleviate pain in the patient over the course of the therapy in the case that the burning could cause discomfort in the treatment region. Finally, the model presented used regular geometry (box-like structure with outright boundaries, interfaces, blood vessels, and other structures) whereas in a clinical setting, a tumor would have an irregular geometry and the blood vessels would not be as orderly as those used in the model.

Despite the limitations, the model does show the behavior of the nanoshells heating the tumor and the surrounding tissue along with its interaction with the vascular network and it is able to calculate the optimal laser intensity to achieve required temperature ranges. In a clinical setting, it is difficult to obtain an accurate temperature distribution without the use of invasive probes. However, with the use of Magnetic Resonance Thermal Imaging (MRTI), 3-D imaging of temperature distributions are possible while leaving the tissue in a non-invasive state. When interstitial probes and MRTIs were compared by measuring temperature in tissue, there was a difference of less than 1 °C between temperature distributions, thus the use of an MRTI has a promising future to determine the temperature distribution of hyperthermia treatments in real time [68,69]. With MRTI in mind, then a computer system could easily regulate the laser's functionality based off the temperature readings on the MRTI, thus making it autonomous once the procedure starts without the need of being invasive with interstitial probes.
CHAPTER 6

CONCLUSIONS AND FUTURE WORK

The objective of this dissertation research was to develop a mathematical model that can accurately determine the temperature distribution in the tumor region and surrounding normal tissue induced by laser irradiation. The model is based on a modified Pennes' equation for the bioheat transfer in a 3-D triple-layered skin structure embedded with a vascular countercurrent network and a tumor appearing in the subcutaneous region. The vascular network is based on the constructal theory of multi-scale tree-shaped heat exchangers. The tumor is injected with gold nanoshells accelerate heating. The proposed model is implemented numerically as a stable finite-difference scheme. To determine the laser intensity required for an optimal temperature distribution can be obtained, we pre-specify the temperature elevations to be obtained at the center of the tumor and on some locations on the perimeter of the skin’s surface. We then use the least squares method to obtain the optimal laser power and to develop a computational procedure that provides the temperature distribution where the tumor region is heated and maintained above 42 °C.

Two studies were performed to see the effects of the laser heating with and without gold nanoshells. When the gold nanoshells were present, the tumor was heated quickly and reached the tumor's minimum temperature of 42 °C with a low laser
intensity. In contrast, in the study without nanoshells, the tumor does heat up the tumor’s central axis reaches 42 °C, but not its extremities. The high laser intensity in the study without nanoshells causes the healthy tissue to be heated to undesirable levels. As a result, the nanoshells are able to achieve localized heating of the tumor while not overly heating the surrounding healthy tissue.

Further study will focus on the use of the dual-phase lagging equation as the modified bioheat transfer equation as considered by other researchers \[22,70-76\], as follows

\[
\rho_l C_l \left( \frac{\partial T_i}{\partial t} + \tau_q \frac{\partial^2 T_i}{\partial t^2} \right) + \tau_q W_l^i C_b \frac{\partial T_i}{\partial t} \\
= k_l \left( \frac{\partial^2 T_i}{\partial x^2} + \frac{\partial^2 T_i}{\partial y^2} + \frac{\partial^2 T_i}{\partial z^2} \right) + W_l^i C_b (T_{out} - T_i) \\
+ k_l \tau_l \frac{\partial}{\partial t} \left( \frac{\partial^2 T_i}{\partial x^2} + \frac{\partial^2 T_i}{\partial y^2} + \frac{\partial^2 T_i}{\partial z^2} \right) + Q_l, \quad l = 1, 2, 3. \tag{6.1}
\]

Other research areas that can be expanded within the dual-phase lagging equation may include the use of different heat sources, denaturating of the tissue, exposure to intense ambient heat, angiogenesis, and irregular geometries for a skin structure and tumor region where the finite element method may be included. Such research may provide an useful tool for optimizing irradiation to kill the tumor while keeping the damage to the surrounding healthy tissue to a minimum during the hyperthermia cancer treatment.
Nomenclature

Mathematical Model

\( Kn \quad \text{Knudsen number} \)

\( L_X, L_Y, L_Z \quad \text{length of model in x,y,z directions, respectively} \)

\( m \quad \text{level of blood vessel} \)

\( P \quad \text{vessel periphery} \)

\( t \quad \text{time} \)

\( x, y, z \quad \text{Cartesian coordinates} \)

\( (k_t)_{\text{eff}} \quad \text{effective thermal conductivity of tumor tissue} \)

\( \alpha \quad \text{Heat transfer coefficient between blood and tissue} \)

\( \alpha_l \quad \text{Laser absorptivity in lth layer} \)

\( \rho_{\text{gold}} \quad \text{density of gold} \)

\( \sigma \quad \text{Standard deviation of width of normally distributed laser beam} \)

\( \tau \quad \text{thermal lag time} \)

\( \tau_q, \tau_t \quad \text{thermal lag of heat flux and temperature gradient, respectively.} \)
\( \varepsilon_t \) porosity of the tumor

\( B_i \) Biot Number

\( C_B \) Heat capacity of blood

\( C_i^l \) Specific heat of blood at \( l \)th layer

\( C_{gold} \) specific heat capacity of gold

\( C_i \) specific heat of \( l \)th layer of skin tissue

\( D_p \) diameter of gold nanoshells

\( F_m \) Cross-sectional area in the \( m \)th level vessel

\( k_{gp} \) thermal conductivity of the gold nanoshells

\( k_i \) thermal conductivity of \( l \)th layer

\( L_{b}^m \) length of blood vessel in level \( m \)

\( L_i \) Depth of \( l \)th skin layers

\( M_m \) Mass flow of blood at level \( m \)

\( N L_{b}^m, N W_{b}^m \) Length and width of cross-section of a blood vessel at level \( m \), respectively

\( P_0 \) Laser Intensity

\( P_m \) Vessel perimeter at \( m \)th level blood vessel
\( Q_i \) \hspace{1cm} \text{Volumetric Heat of } i\text{th layer}

\( R_{eff_i} \) \hspace{1cm} \text{Laser reflectivity in } i\text{th layer}

\( T_0 \) \hspace{1cm} \text{Initial temperature of tissue}

\( T_a \) \hspace{1cm} \text{ambient Temperature}

\( T_{in}, T_{out} \) \hspace{1cm} \text{Blood temperature at entrance and exit of artery, respectively}

\( T_t, L_T \) \hspace{1cm} \text{location of the top of the tumor and length of the tumor in the}
\hspace{1cm} \text{} z\text{-direction, respectively}

\( T_w^m, T_b^m \) \hspace{1cm} \text{Wall and blood temperature in } m\text{th level vessel, respectively}

\( u'' \) \hspace{1cm} \text{local heat generation spectrum being generated by golden}
\hspace{1cm} \text{nanoshells}

\( v_m \) \hspace{1cm} \text{blood flow velocity at level } m

\( W_{b}^l \) \hspace{1cm} \text{Blood perfusion rate at } l\text{th layer}

\( x_0(t), y_0(t) \) \hspace{1cm} \text{Location of focused laser at time } t

\( T_{3}^{out, in, eff} \) \hspace{1cm} \text{temperature outside the tumor, inside the tumor, and within}
\hspace{1cm} \text{the nanoshell embedded region}

\( \dot{P} \) \hspace{1cm} \text{Decreased blood flow rate in level 7 vessel}

\textbf{Nanoshell Model}

\( B's, C's, \xi_1 \) \hspace{1cm} \text{analytical coefficients}
\( \alpha_g \) absorptivity of gold nanoshells

\( \beta_{\text{tot}-\lambda}, \beta_{\text{md}} \) total extinction of the tumor and extinction of the medium, respectively

\( \lambda_{ep} \) effective mean path of electron

\( \mu_c \) angle of incidence

\( \omega_\lambda \) scattering albedo

\( \rho_\lambda \) reflectivity for nanoshell model

\( \sigma_{\text{md}-\lambda}, \sigma_{s-\lambda} \) spectral scattering of medium and gold nanoshells, respectively

\( \tau_{\text{tum}} \) Optical length of tumor region

\( \varepsilon_\lambda \) propagation of light

\( k_{\text{md}-\lambda}, k_{s-\lambda} \) spectral absorption of medium and gold nanoshells, respectively

\( N_T \) amount of nanoshells per unit volume

\( q''_{R\lambda} \) total spectral radiative heat flux

\( q_{C\lambda} \) collimated radiative heat flux

\( q_{d\lambda} \) diffuse radiative heat flux

\( Q_{k\lambda}, Q_{s\lambda} \) spectral absorption efficiency and spectral scattering efficiency of nanoshells, respectively

\( r_o \) outer radius of gold nanoshells
Numerical Method

$\Delta t$  
- time increment used in calculating heat transfer

$\Delta x, \Delta y, \Delta z$  
- Mesh sizes of Finite Difference scheme for bioheat transfer model in x,y,z directions

$h$  
- Convective Heat Transfer Coefficient

$M$  
- number of prespecified temperatures

$S$  
- sum of squares

$\alpha^*$  
- Relaxation parameter

$\delta_z^2, \delta_y^2, \delta_z^2$  
- Second Order Finite Difference Operators

$\omega$  
- Relaxation Factor

$\varepsilon_{\text{laser}}$  
- laser convergence criterion

$L_{\text{pre}}$  
- Preconditioned Richardson Iteration at level 1

$N_x, N_y, N_z$  
- number of grid points in x,y,z directions, respectively

$u_{\text{cal}}^i$  
- Calculated temperature at location i

$u_{\text{pre}}^i$  
- Prespecified temperature at location i

$u_b^m$  
- Numerical solution of blood temperature in the mth blood vessel

$u_{ijk}^n$  
- Numerical solution of temperature in tissue
$\mathbf{X}$  Sensitivity Coefficient Matrix

$\mathbf{I}$  Identity Matrix
APPENDIX B

SOURCE CODE
#include "string.h"
#include <math.h>
#include <stdio.h>
#include "memory.h"
#include "Matrix.h"

#include <fstream>

********AUTHOR INFORMATION********
AUTHOR: CASEY O'NEAL ORNDORFF
ADVISOR: WEIZHONG DAI
COLLABORATOR: STANISLAV PONOMAREV

CODE USED FOR DISSERTATION: THERMAL ANALYSIS IN A TRIPLE-LAYERED SKIN STRUCTURE WITH EMBEDDED VASCULARATURE, TUMOR, AND GOLD NANOSHELLS

THE CODE PRESENTED HERE WAS USED TO PREDICT THE TEMPERATURE DISTRIBUTION OF A 3D TRIPLE LAYERED SKIN STRUCTURE EMBEDDED WITH A SEVEN LEVEL COUNTERCURRENT NETWORK AS IT IS IRRADIATED BY A LASER WITH NANOSHELLS IN AN EMBEDDED TUMOR REGION WITHIN THE SUBCUTANEOUS LAYER. THIS CODE WILL FIND THE OPTIMIZED LASER INTENSITY TO RAISE THE TUMOR'S CENTRAL TEMPERATURE TO 50 C WITHIN 10 SECONDS. THE GOAL TEMPERATURE AND TIME INTERVAL FOR OPTIMIZATION CAN BE MODIFIED AS NECESSARY. AFTER THE OPTIMIZATION PROCESS IS OVER, IT WILL FIND THE OPTIMAL TEMPERATURE DISTRIBUTION AT A GIVEN TIME INTERVAL (600 SECONDS BY DEFAULT, CAN BE MODIFIED AS NECESSARY).

#define SC R EE N O U T //flag for screen output
#define M 5 // Number of points used in Least Squares Method

*************** SET UP THE SCALE OF THE SYSTEM***************
#define SCALEX 1 //GRID POINT CHANGING SCALE (Original is 1)
#define SCALEY 1

#define NZ1 8 //layer 1 skin grid points in z direction
#define NZ2 208 //layer 1+2 skin grid points in z direction
#define NZ3 2600 //layer 1+2+3 skin grid points in z direction

#define LT 250 //Length of tumor, start from TT, ends at TT + LT (z-direction)
#define TT 240 //Top starting point of tumor

//#of grid points in x direction **MUST BE EVEN NUMBER**
#define NX 162 * SCALEX //Original is 162

//#of grid points in y direction **MUST BE EVEN NUMBER**
#define NY 162 * SCALEY //Original is 162

*************** END SET-UP OF SCALING THE SYSTEM***************

*************** VASCULAR SYSTEM DIMENSIONAL SET UP***************

*************** ARTERIAL SYSTEM DIMENSION SET UP***************

#define LX1A 100 * SCALEX
#define LY1A 20 * SCALEY
#define LZ1A 200
#define LX2A 16 * SCALEX
#define LY2A 72 * SCALEY //whole length
#define LZ2A 160
#define LX3A 12 * SCALEX //each branch
#define LY3A 12 * SCALEY //each branch
#define LZ3A 500 //whole length
#define LX4A 36 * SCALEX //whole length
//define LY4A 10 * SCALEY //each branch
#define LZ4A 100 //each branch
#define LX5A 8 * SCALEX //each branch
#define LY5A 26 * SCALEY //whole length
#define LZ5A 80 //each branch
#define LX6A 6 * SCALEX //each branch
#define LY6A 6 * SCALEY //each branch
#define LZ6A 180 //whole length
#define LX7A 14 * SCALEX //whole length
#define LY7A 4 * SCALEY //each branch
#define LZ7A 40 //50 //each branch

double deltaT = 0.1f;
double deltaX = 0.01f / SCALEX, deltaY = 0.01f / SCALEY;
double deltaZ = 0.001f; //tissue and blood use same grid size

#include <iostream>

using namespace std;

class BloodVessel { public:
    double length;
    //... other properties...

private:
    //... private members...

};

test() { //... test function...
    BloodVessel bv;
    bv.length = 100.0;
    //... other operations...
}

int main() { //... main function...
    test();
    return 0;
}
double CB = 0.004134;
double Cb1 = 0.0f, Cb2 = 4.2f, Cb3 = 4.2f;
//specific heat of blood in skin later 1 to 3
double v1 = 8.0f;  //v2, v3. velocity of blood flow in level 1 ??
//double v1 = 8.0f;  //v2, v3. velocity of blood flow in level 1 ??
double v1 = 8.0f;  //v2, v3. velocity of blood flow in level 1 ??
da double Pdot = 0.5e-3f;  //decreased blood flow rate (for level 7 only)
double p1 = 1.2f, p2 = 1.2f, p3 = 1.0f;  //density of tissue of layer 1,2,&3
const double psi = 3.14159265358979f;
double Hf = 0.001;
//convective heat transfer coefficient between the environment and the skin
double TF = 25.0;  //environment temperature
double tau = 20.0;  // delay of time
double sigma = 5.669e-12;  //Stefan - Boltzmann constant
double epsilon = 0.9;  //Reflectivity
double energy;
double frequency;
double gas = 8.314472;  //Unit: J/K mol
int centerX = (NX/2);  //center of x
int centerY = (NY/2);  //center of y
int centerTZ = (TT+(TT+LT))/2;  //center of the tumor z direction
//center of the layer power
//Absoptivity and Reflectivity of layers:
double Alpha1 = 1.8, Alpha2 = 1.8, Alpha3 = 1.8,
AlphaG = 5.76969;  //laser absorptivities of the three skin layers
don double Sigma2 = 0.01;  //standard deviation of the width of a normally distributed laser beam
don double Ref1 = 0.1, Ref2 = 0.1, Ref3 = 0.1;  //layer reflectivities of three layers of the skin
//Tumor parameters:
//Casey's input data
//Nanoparticle data
double NanoDense = 7.0e9;  //particles / cm3
double SpecScatEffNS = .92968;  //Spectral Scattering efficiency of Nanoshells
double SpecAbsorpEffNS = .059492;  //Spectral Absorption Efficiency of NS
don double SpecScatEffNS = .0592968;  //Spectral Scattering efficiency of Nanoshells
don double SpecAbsorpEffNS = .059492;  //Spectral Absorption Efficiency of NS
don double Ri = 1.6e-6;  //cm Inner radius of the nanoshells
don double Ro = 2.2e-6;  //cm Outer radius of the nanoshells
don double SpecScatCS = pi*(Ro*Ro)* SpecScatEffNS * NanoDense;  //cm, Spectral Scattering Cross-Section
ndon double SpecAbsbCS = pi*(Ro*Ro)* SpecAbsbEffNS * NanoDense;  //cm, Spectral Absorption Cross-Section
//Tumor Data
double TumScat = 66.9; // cm^-1, Scattering component of Tumor
double TumAbsb = .10; // cm^-1, Absorbing component of Tumor

double TumExt = TumScat + TumAbsb + SpecScatCS + SpecAbsbCS; // cm^-1, Tumor Extinction (SuperPosition)

double kTumor = .00642; // W/cm C Thermal Conductivity of Tumor //orig: .0055 // .00642

double pTumor = 1.0; // g/cm^3 Mass Density

double CTumor = 3.75; // J/g C Specific Heat of Tumor Tissue // 3.75

double WbTumor = .0005; // g/cm^3 s Perfusion Rate of Tumor //orig: .0005

double porous = .4; // porosity

double Albedo = (TumScat+SpecScatCS)/TumExt; // Scattering Albedo of the system

// Gold nanoparticle data

double CGold = .129; // Specific Heat capacity of Gold in J/g C

double KGold = 3.17; // Thermal conductivity of Gold in W/cm C

double pGold = 19.3; // Density of Gold in g/cm^3

double lambdaE = 3.614e-6; // Mean free path of electrons in the solid

double Kn = lambdaE/(2.0*Ro);

double Kpg = KGold*(1.0/(1.0+Kn));

double Keff = (porous)*Kpg + (1.0-porous)*kTumor; // Calculating Effective thermal conductivity for tumor region

double Ceff = (porous)*(CGold*pGold) + (1.0-porous)*(CTumor*pTumor);

double deltaPX = (NX/10);

double deltaPY = (NY/10);

double deltaPZ = (LT/5); // used for boundaries for nanoparticle region within tumor

const double omega = 1.0f;

double Bi = alpha/k3; // Biot number

double factor1[2], factor2[2], factor3[2], factor4[2];

double factor5[2], factor6[2], factor7[2]; // used for Runge-Kutta method

double theTA0 = 0.0f; // elevated blood temperature at entrance

double (*Tt)[NY+1][NZ3+1]; // tissue temperature at time level n

double (*Tt_n1)[NY+1][NZ3+1]; // tissue temperature at time level n+1 (loop l+1)

double (*Tt_n1_l)[NY+1][NZ3+1]; // tissue temperature at time level n+1 (loop l)

double (*Ut)[NY+1][NZ3+1]; // originated blood

double (*Ut_n1)[NY+1][NZ3+1];

double (*Damage_t)[NY+1][NZ3+1];

double (*Damage_n1)[NY+1][NZ3+1];

double (*Q1)[NY+1][NZ3+1]; // volumetric heat due to spatial heating

double (*Q2)[NY+1][NZ3+1];

double (*Q3)[NY+1][NZ3+1];

double (*Tbd1)[2][4]; // blood board temperature at time level 1

double (*Tbd2)[2][4]; // blood board temperature at time level 2

double (*Tbd3)[2][4]; // blood board temperature at time level 3

double (*Tbd4)[2][4]; // blood board temperature at time level 4

double (*Tbd5)[2][4]; // blood board temperature at time level 5

double (*Tbd6)[2][4]; // blood board temperature at time level 6

double (*Tbd7)[2][4]; // blood board temperature at time level 7

double **Tb1_n1; // first level blood temperature at time level n+1

double **Tb2_n1; // second level blood temperature at time level n+1
double **Tb3_nI; //third level blood temperature at time level n+1
double **Tb4_nI; //fourth level blood temperature at time level n+1
double **Tb5_nI; //fifth level blood temperature at time level n+1
double **Tb6_nI; //sixth level blood temperature at time level n+1
double **Tb7_nI; //seventh level blood temperature at time level n+1

double **Tv1_nI; //interpolated first level blood vessel temperature at time level n+1
double **Tv2_nI; //interpolated second level blood vessel temperature at time level n+1
double **Tv3_nI; //interpolated third level blood vessel temperature at time level n+1
double **Tv4_nI; //interpolated fourth level blood vessel temperature at time level n+1
double **Tv5_nI; //interpolated fifth level blood vessel temperature at time level n+1
double **Tv6_nI; //interpolated sixth level blood vessel temperature at time level n+1
double **Tv7_nI; //interpolated seventh level blood vessel temperature at time level n+1

double (*a)[NY+1][NZ3+1], (*b)[NY+1][NZ3+1],
(*c)[NY+1][NZ3+1];
double (*d)[NY+1][NZ3+1]; //tridiagonal system
double (*a0)[NY+1][NZ3+1], (*b0)[NY+1][NZ3+1],
(*c0)[NY+1][NZ3+1];

const int SPAN = 3; //grid points index on boarder of blood vessel

int X1[2], X2a[2], X2b[2], X3a[2], X3b[2], X4a[2], X4b[2],
X5a[2], X5b[2];

int X6a[2], X6b[2], X7a[2], X7b[2];

int Y1a[2], Y1b[2], Y2a[2], Y2b[2], Y3a[2], Y3b[2], Y4a[2], Y4b[2];

int Y5a[2], Y5b[2], Y6a[2], Y6b[2], Y7a[2], Y7b[2];

int Y8a[2], Y8b[2], Y9a[2], Y9b[2];

int Z1a[2], Z1b[2], Z2a[2], Z2b[2], Z3a[2], Z3b[2], Z4a[2], Z4b[2];

int Z5a[2], Z5b[2], Z6a[2], Z6b[2], Z7a[2], Z7b[2];

int t, I, loopP; //used to keep record of loop and output result

int k, p, q, r, h; //repeat variable used for blood vessels of same layer

//BLOOD BORDER VARIABLES

int x1a, x2a, x2b, x3a, x3b, x4a, x4b, x5a, x5b, x6a1, x6b1, x6a2,
x6b2, x7a1, x7b1, x7a2, x7b2;

int y1a, y1b, y2a, y2b, y3a, y3b, y4a1, y4b1, y4a2, y4b2, y5a1,
y5b1, y5a2, y5b2, y6a1, y6b1;

int y6a2, y6b2, y7a1, y7b1, y7a2, y7b2, y7a3, y7b3, y7a4, y7b4;
int z1a,z1b,z2a,z2b,z3a,z3b,z4a,z4b,z5a,z5b,z6a,z6b1;
int z6a2,z6b2,z7a1,z7b1,z7a2,z7b2;
int lx1,ly1,lz1,lx2,ly2,lz2,lx3,ly3,lz3,ly4,lz4,ly4,lz5,ly5,lz5;
int lx6,ly6,lz6,lx7,ly7,lz7;
int cenY,cenZ,cenX;
double Tm_pre[5] = {50, 34, 34, 34, 34}; // pre­specified surface temperature; (Tumor Center, edges of model x 4, Top of skin)
bool bPowerOn; //flag indicating whether the power is on or off
***************END PARAMETER SET­UP***************

***********************************************************************************
PARAMETER SETUP************************
***********************************************************************************

***********************************************************************************SUBROUTINE DECLARATION***********************

void initialize();
int CalcAll(double P0,bool dP);
int getTv_blood(int index);
int CalcTb();
int CalcTb2();
double CalcTt();
void CalcVessel(int index);
void Reset(void);
void setVesselBorder();
void AdjustMtx(int index);
void clearMem();
void setborderVariable(int index);//added by zeng
void reloadbloodeteperature(int index);//added by zeng
void InitQ(double P0);
double CalcNewP(double *Tm_pre, double *Tm0, double *Tm1, double P0, double deltaP);
int getTm(double *Tm);
void AdjustPower(double P0);
void writeSquareXZ(int x0, int z0, int x1, int z1, int y, int t, int I);
void writeSquareYZ(int y0, int z0, int y1, int z1, int x, int t, int I);
void writeSquareXY(int x0, int y0, int x1, int y1, int z, int t, int I);
void writeZCenter(int t);
void writeAll(int t, int I);
void writeLinearSys(int x, int y, int i, int I);
void writeblood(int i, double x);//test by zeng
void writeLog(char *line);
void writeQCenter(int t);
char tmp[512];
***********************************************************************************END OF SUBROUTINE DECLARATION**************

***********************************************************************************INITIALIZE LOOP PARAMETERS FOR PROGRAM************

int TOTAL T = 10; //This Total T is used for the optimization process
//double TOTAL_ T = 0.2f; //test value
double Err_I = 0.001f; //for I loop
double Err_P = 0.01f; //for P loop .01 orig
double temp;
double maxErr1, f;
double aa, bb;
int inde;
double fk1, fk2, fk3, fk4;
int n;
bool Optimized = false;
***********************************************************************************END OF INITIALIZE LOOP PARAMETERS************
BEGIN MAIN PROGRAM

int main(void)
{
    printf("screen out\n");
    printf("initializing...\n");
    initialize(); //memory allocation & variable initialization
    printf("running...\n");

    double Tm0[5]; //calculated surface temperatures - first run
    double Tm1[5]; //calculated surface temperatures - second run - with updated power level
    double P0, P1; //power level
    double deltaP; //power step
    double oldSP, newSP;
    int indexM;
    P0 = 1.2948; //Initial laser
    loopP = 0;
    sprintf(tmp, "Initial P:% 5.4lf, \n", P0);
    writeLog(tmp);

    //calculate tissue and blood temperature based on specified power level
    do{
        oldSP=newSP;
        //calculate tissue and blood temperature based on specified power level
        CalcAll(P0,0);
        //retrieve calculated surface temperature
        getTm(Tm0);
        //break;
        //calculate the S(P)
        newSP=0;
        for (indexM=0; indexM<M;
        indexM++)
            newSP=newSP+
            (Tm_pre[indexM]*Tm0[indexM])/(Tm_pre[indexM] - Tm0[indexM]);
    loopP++;
    //increment power level and calculate the new surface temperature
    CalcAll(P0+P0/100.0,1);
    getTm(Tm1);

    //calculate the next power level based on prespecified temperature and calculated surface temperature
    P1 = CalcNewP(Tm_pre, Tm0, Tm1, P0, P0/100.0);
    //\\\\ Record the temperatures of M points
    sprintf(tmp, "Tm0[0]=%6.4lf; Tm0[1]=%6.4lf; Tm0[2]=%6.4lf; Tm0[3]=%6.4lf; Tm0[4]=%6.4lf; \n",
            Tm0[0],Tm0[1],Tm0[2],Tm0[3],Tm0[4]);
    writeLog(tmp);
    sprintf(tmp, "Tm1[0]=%6.4lf; Tm1[1]=%6.4lf; Tm1[2]=%6.4lf; Tm1[3]=%6.4lf; Tm1[4]=%6.4lf; \n",
            Tm1[0],Tm1[1],Tm1[2],Tm1[3],Tm1[4]);
    writeLog(tmp);
    sprintf(tmp, "p%d:%6.4lf; newSP:%8.6lf; oldSP:%8.6lf; (newSP-oldSP)/newSP:%8.6lf \n", loopP, P1, newSP, oldSP,(newSP-oldSP)/newSP);
    writeLog(tmp);
    P0 = P1;
    loopP++;
    if (newSP==0) break;
} while(fabs((newSP-oldSP)/newSP) > Err_P);

Optimized = true; //laser optimized
printf("Optimized Laser Achieved! Starting therapy \
")
TOTAL_T = 600; //Total T used for primary
calculation of therapy
CalcAll(PO,0); //Run therapy
printf("CalcAll Finished!\n");
clearMem();
return 1;
}

******************************************************************************
BEGINNING OF
SUBROUTINES******************************************************************************

//memory allocation & variable initialization
******************************************************************************
INITIALIZATION
AND MEMORY ALLOCATION OF SYSTEM
(HEAT SOURCES, INITIAL TEMPERATURES,
TRIDIAGONAL SYSTEM, VASCULAR SYSTEM
PARAMETERS)******************************************************************************

void initialize()
{

//allocating memory for heat sources per layer
Q1 = new double [NX+1][NY+1][NZ3+1];
Q2 = new double [NX+1][NY+1][NZ3+1];
Q3 = new double [NX+1][NY+1][NZ3+1];

//allocating memory for Tissue temperatures
at n, n+1, and n+1 at iteration I time steps,
respectively.
Tt = new double [NX+1][NY+1][NZ3+1];
Tt n1 = new double
[NX+1][NY+1][NZ3+1];

//allocating memory for the Thermal lag term
at n and n+1 time steps
Ut= new double [NX+1][NY+1][NZ3+1];
Ut n1 = new double
[NX+1][NY+1][NZ3+1];

******************************************************************************
Initializing Tridiagonal system
******************************************************************************

memset(a0, 0,
sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
memset(b0, 0,
sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
memset(c0, 0,
sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));

******************************************************************************
_ALLOCATING memory for the length of the
blood vessels
******************************************************************************

*Lb1[0] = LX1A*deltaX; //arteries blood
length in one to seven layer
Lb2[0] = LY2A*deltaY;
Lb3[0] = LZ3A*deltaZ;
Lb4[0] = LX4A*deltaX;
Lb5[0] = LY5A*deltaY;
Lb6[0] = LZ6A*deltaZ;
Lb7[0] = LX7A*deltaX;
Lb1[1] = LX1B*deltaX; //vein blood length
in one to seven layer
Lb2[1] = LY2B*deltaY;
Lb3[1] = LZ3B*deltaZ;
Lb4[1] = LX4B*deltaX;
Lb5[1] = LY5B*deltaY;
Lb6[1] = LZ6B*deltaZ;
\[ Lb7[1] = LX7B*deltaX; \]

setVesselBorder();

// allocating memory for blood temperature at \( n+1 \)
Tb1_n1 = new double*[2];
Tb2_n1 = new double*[2];
Tb3_n1 = new double*[2];
Tb4_n1 = new double*[2];
Tb5_n1 = new double*[2];
Tb6_n1 = new double*[2];
Tb7_n1 = new double*[2];
Tv1_n1 = new double*[2];
Tv2_n1 = new double*[2];
Tv3_n1 = new double*[2];
Tv4_n1 = new double*[2];
Tv5_n1 = new double*[2];
Tv6_n1 = new double*[2];
Tv7_n1 = new double*[2];

// allocating memory for the blood vessel walls for arteries
Tbd1[0][0] = new double[LX1A+1];
Tbd2[0][0] = new double[LY2A+1];
Tbd3[0][0] = new double[LZ3A+1]*2;
Tbd4[0][0] = new double[LX4A+1]*4;
Tbd5[0][0] = new double[LY5A+1]*8;
Tbd6[0][0] = new double[LZ6A+1]*16;
Tbd7[0][0] = new double[LX7A+1]*32;

Tbd1[0][1] = new double[LX1A+1];
Tbd2[0][1] = new double[LY2A+1];
Tbd3[0][1] = new double[LZ3A+1]*2;
Tbd4[0][1] = new double[LX4A+1]*4;
Tbd5[0][1] = new double[LY5A+1]*8;
Tbd6[0][1] = new double[LZ6A+1]*16;

// allocating memory for Vessel temperature at \( n+1 \) time step
Tv1_n1[0] = new double[LX1A+1];
Tv2_n1[0] = new double[LY2A+1];
Tv3_n1[0] = new double[LZ3A+1]*2;
Tv4_n1[0] = new double[LX4A+1]*4;
Tv5_n1[0] = new double[LY5A+1]*8;
Tv6_n1[0] = new double[LZ6A+1]*16;
Tv7_n[0] = new double[(LX7A+l)*32];
//allocating memory for Blood vessel border
for Veins
Tbd1[1][0] = new double[LX1B+1];
Tbd2[1][0] = new double[LY2B+1];
Tbd3[1][0] = new double[(LZ3B+1)*2];
Tbd4[1][0] = new double[(LX4B+1)*4];
Tbd5[1][0] = new double[(LY5B+1)*8];
Tbd6[1][0] = new double[(LZ6B+1)*16];
Tbd7[1][0] = new double[(LX7B+1)*32];
Tbd1[1][1] = new double[LX1B+1];
Tbd2[1][1] = new double[LY2B+1];
Tbd3[1][1] = new double[(LZ3B+1)*2];
Tbd4[1][1] = new double[(LX4B+1)*4];
Tbd5[1][1] = new double[(LY5B+1)*8];
Tbd6[1][1] = new double[(LZ6B+1)*16];
Tbd7[1][1] = new double[(LX7B+1)*32];
Tbd1[1][2] = new double[LX1B+1];
Tbd2[1][2] = new double[LY2B+1];
Tbd3[1][2] = new double[(LZ3B+1)*2];
Tbd4[1][2] = new double[(LX4B+1)*4];
Tbd5[1][2] = new double[(LY5B+1)*8];
Tbd6[1][2] = new double[(LZ6B+1)*16];
Tbd7[1][2] = new double[(LX7B+1)*32];
Tbd1[1][3] = new double[LX1B+1];
Tbd2[1][3] = new double[LY2B+1];
Tbd3[1][3] = new double[(LZ3B+1)*2];
Tbd4[1][3] = new double[(LX4B+1)*4];
Tbd5[1][3] = new double[(LY5B+1)*8];
Tbd6[1][3] = new double[(LZ6B+1)*16];
Tbd7[1][3] = new double[(LX7B+1)*32];
// Calculating the perimeter of each blood vessel level for arteries
P1[0] = (LZ1A*deltaZ + LY1A*deltaY)*2;
P2[0] = (LX2A*deltaX + LZ2A*deltaZ)*2;
P3[0] = (LX3A*deltaX + LY3A*deltaY)*2;
P4[0] = (LZ4A*deltaZ + LY4A*deltaY)*2;
P5[0] = (LX5A*deltaX + LZ5A*deltaZ)*2;
P6[0] = (LX6A*deltaX + LY6A*deltaY)*2;
P7[0] = (LZ7A*deltaZ + LY7A*deltaY)*2;

// Calculating the perimeter of each blood vessel level for veins
P1[1] = (LZ1B*deltaZ + LY1B*deltaY)*2;
P2[1] = (LX2B*deltaX + LZ2B*deltaZ)*2;
P3[1] = (LX3B*deltaX + LY3B*deltaY)*2;
P4[1] = (LZ4B*deltaZ + LY4B*deltaY)*2;
P5[1] = (LX5B*deltaX + LZ5B*deltaZ)*2;
P6[1] = (LX6B*deltaX + LY6B*deltaY)*2;
P7[1] = (LZ7B*deltaZ + LY7B*deltaY)*2;
// Calculating the Cross-sectional area of the Arteries
\[ F_1[0] = (LZ1A*\Delta Z)*(LY1A*\Delta Y); \]
\[ F_2[0] = (LX2A*\Delta X)*(LZ2A*\Delta Z); \]
\[ F_3[0] = (LX3A*\Delta X)*(LY3A*\Delta Y); \]
\[ F_4[0] = (LZ4A*\Delta Z)*(LY4A*\Delta Y); \]
\[ F_5[0] = (LX5A*\Delta X)*(LZ5A*\Delta Z); \]
\[ F_6[0] = (LX6A*\Delta X)*(LY6A*\Delta Y); \]
\[ F_7[0] = (LZ7A*\Delta Z)*(LY7A*\Delta Y); \]

//Calculating the Cross-sectional area of the Veins
\[ F_1[1] = (LZ1B*\Delta Z)*(LY1B*\Delta Y); \]
\[ F_2[1] = (LX2B*\Delta X)*(LZ2B*\Delta Z); \]
\[ F_3[1] = (LX3B*\Delta X)*(LY3B*\Delta Y); \]
\[ F_4[1] = (LZ4B*\Delta Z)*(LY4B*\Delta Y); \]
\[ F_5[1] = (LX5B*\Delta X)*(LZ5B*\Delta Z); \]
\[ F_6[1] = (LX6B*\Delta X)*(LY6B*\Delta Y); \]
\[ F_7[1] = (LZ7B*\Delta Z)*(LY7B*\Delta Y); \]

//Calculating the Mass Flow of the Blood traveling through the arteries
\[ M_1[0] = v_1*F_1[0]; \]
\[ M_2[0] = 0.5f * M_1[0]; \]
\[ M_3[0] = 0.5f * M_2[0]; \]
\[ M_4[0] = 0.5f * M_3[0]; \]
\[ M_5[0] = 0.5f * M_4[0]; \]
\[ M_6[0] = 0.5f * M_5[0]; \]
\[ M_7[0] = 0.5f * M_6[0]; \]

//Calculating the Mass Flow of the Blood traveling through the veins
\[ M_1[1] = v_1*F_1[1]; \]
\[ M_2[1] = 0.5f * M_1[1]; \]
\[ M_3[1] = 0.5f * M_2[1]; \]
\[ M_4[1] = 0.5f * M_3[1]; \]
\[ M_5[1] = 0.5f * M_4[1]; \]
\[ M_6[1] = 0.5f * M_5[1]; \]
\[ M_7[1] = 0.5f * M_6[1]; \]

//factor[] is used for Runge-Kutta method:
\[ \text{factor1}[0] = \text{deltaX*alpha*P1[0]/(M1[0]*CB)}; \]
\[ \text{factor2}[0] = \text{deltaY*alpha*P2[0]/(M2[0]*CB)}; \]
\[ \text{factor3}[0] = \text{deltaZ*alpha*P3[0]/(M3[0]*CB)}; \]
\[ \text{factor4}[0] = \text{deltaX*alpha*P4[0]/(M4[0]*CB)}; \]
\[ \text{factor5}[0] = \text{deltaY*alpha*P5[0]/(M5[0]*CB)}; \]
\[ \text{factor6}[0] = \text{deltaZ*alpha*P6[0]/(M6[0]*CB)}; \]
\[ \text{factor7}[0] = \text{deltaX*alpha*P7[0]/(M7[0]*CB)}; \]
\[ \text{factor1}[1] = \text{deltaX*alpha*P1[1]/(M1[1]*CB)}; \]
\[ \text{factor2}[1] = \text{deltaY*alpha*P2[1]/(M2[1]*CB)}; \]
\[ \text{factor3}[1] = \text{deltaZ*alpha*P3[1]/(M3[1]*CB)}; \]
\[ \text{factor4}[1] = \text{deltaX*alpha*P4[1]/(M4[1]*CB)}; \]
\[ \text{factor5}[1] = \text{deltaY*alpha*P5[1]/(M5[1]*CB)}; \]
\[ \text{factor6}[1] = \text{deltaZ*alpha*P6[1]/(M6[1]*CB)}; \]
\[ \text{factor7}[1] = \text{deltaX*alpha*P7[1]/(M7[1]*CB)}; \]

bPowerOn = true;
writeLog("Initialization...");
//initialize tri-diagonal system, left side (fixed)
for(i=1;i<=NX-1;i++)
{
  for(j=1;j<=NY-1;j++)
  {
  ...
for(z=1;z<=NZ1-1;z++)
{
  // Laser with CCVN and Tumor: Casey's Method
  b0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);
  a0[i][j][z] =
      2.0*p1*C1*(1.0+tau*Wb1*Cb1/(p1*C1) + 2.0*tau/deltaT) + Wb1*Cb1*deltaT +
      (4.0*k1*deltaT)*(1.0/(deltaX*deltaX) + 1.0/(deltaY*deltaY)) +
      (2.0*k1*deltaT)/(deltaZ*deltaZ);
  c0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);
}

b0[i][j][NZ1] = -k1;

a0[i][j][NZ1] = k1+k2;

c0[i][j][NZ1] = -k2;

for(z=NZ1+1;z<=NZ2-1;z++)
{
  // Laser with CCVN and Tumor: Casey's Method
  b0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);
  a0[i][j][z] =
      2.0*p2*C2*(1.0+tau*Wb2*Cb2/(p2*C2) + 2.0*tau/deltaT) + Wb2*Cb2*deltaT +
      (4.0*k2*deltaT)*(1.0/(deltaX*deltaX) + 1.0/(deltaY*deltaY)) +
      (2.0*k2*deltaT)/(deltaZ*deltaZ);
  c0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);
}

b0[i][j][NZ2] = -k2;

a0[i][j][NZ2] = k2+k3;

c0[i][j][NZ2] = -k3;

for(z=1;z<=NZ3-1;z++)
{
  // Laser case with CVN and Tumor: Casey's Method
  iRz== TT && i>= LTX && i<= RTX && j>= LTY && j<= RTY)
  
  b0[i][j][z] = -k3;
  a0[i][j][z] = k3+kTumor;
  c0[i][j][z] = -kTumor;

  } } //top of tumor; Interface of top of tumor with subcutaneous region

else if(z>TT && z<TT+LT && i>LTX && i<RTX && j>LTY && j<RTY)

{ if( (i< centerX + deltaPX) && (i > centerX - deltaPX) && (j < centerY + deltaPY) && (j > centerY - deltaPY) && (z < centerTZ + deltaPZ) && (z > centerTZ - deltaPZ) )

  b0[i][j][z] = -(Keff*deltaT)/(deltaZ*deltaZ);
  a0[i][j][z] =
      2.0*Ceff*(1.0+tau*WbTumor*Cb3/(Ceff)+2.0*tau/deltaT) +
      WbTumor*Cb3*deltaT +
      (4.0*Keff*deltaT)*(1.0/(deltaX*deltaX) + 1.0/(deltaY*deltaY)) +
      (2.0*Keff*deltaT)/(deltaZ*deltaZ);
  c0[i][j][z] = -(Keff*deltaT)/(deltaZ*deltaZ);

  } } //Inside Tumor with nanoparticles

else if( (i==centerX - deltaPX) && (j < centerY + deltaPY) && (j > centerY - deltaPY) && (z < centerTZ + deltaPZ) && (z > centerTZ - deltaPZ) )

  b0[i][j][z] = -kTumor*deltaT/(deltaX*deltaX);
  a0[i][j][z] = (kTumor + Keff)*deltaT/(deltaX*deltaX);
  c0[i][j][z] = -Keff*deltaT/(deltaX*deltaX);

} } //Left Side of Nanoparticle-Tumor region in X direction

// third skin layer
else if (i == centerX + deltaPX) & (j < centerY + deltaPY) & (j < centerTZ + deltaPZ) & (j < centerTZ - deltaPZ)) {

    bO[i][j][z] = -Keff*deltaT/(deltaX*deltaX);
    aO[i][j][z] = (kTumor + Keff)*deltaT/(deltaX*deltaX);
    cO[i][j][z] = -kTumor*deltaT/(deltaX*deltaX);
}
} //Right side of Nanoparticle-Tumor region in X direction

else if (j == centerY - deltaPY) & (i < centerX + deltaPX) & (i > centerX - deltaPX) & (z < centerTZ + deltaPZ) & (z > centerTZ - deltaPZ)) {

    bO[i][j][z] = -kTumor*deltaT/(deltaY*deltaY);
    aO[i][j][z] = (kTumor + Keff)*deltaT/(deltaY*deltaY);
    cO[i][j][z] = -kTumor*deltaT/(deltaY*deltaY);
}
} //Left Side of Nanoparticle-Tumor region in Y direction

else if (j == centerY + deltaPY) & (i < centerX + deltaPX) & (i > centerX - deltaPX) & (z < centerTZ + deltaPZ) & (z > centerTZ - deltaPZ)) {

    bO[i][j][z] = -Keff*deltaT/(deltaY*deltaY);
    aO[i][j][z] = (kTumor + Keff)*deltaT/(deltaY*deltaY);
    cO[i][j][z] = -kTumor*deltaT/(deltaY*deltaY);
}
} //Right side of Nanoparticle-Tumor region in Y direction

else if (z == centerTZ - deltaPZ) & (i >= centerX + deltaPX) & (i <= centerX - deltaPX) & (j >= centerY + deltaPY) & (j <= centerY - deltaPY)) {

    bO[i][j][z] = -Keff;
    aO[i][j][z] = kTumor + Keff;
    cO[i][j][z] = -Keff;
}
} //Top of nanoparticle embedded region

else if (z == centerTZ + deltaPZ) & (i <= centerX + deltaPX) & (i >= centerX - deltaPX) & (j <= centerY + deltaPY) & (j >= centerY - deltaPY)) {

    bO[i][j][z] = -Keff;
    aO[i][j][z] = kTumor + Keff;
    cO[i][j][z] = -Keff;
}
} //Bottom of nanoparticle embedded region

else {

    bO[i][j][z] = -Keff;
    aO[i][j][z] = kTumor + Keff;
    cO[i][j][z] = -kTumor;
}
} //Inside Tumor with no nanoparticles

} //Inside Tumor

else if (z == TT + LT & i >= LTX & i <= RTX & j >= LTY & j <= RTY) {

    bO[i][j][z] = -Keff;
    aO[i][j][z] = kTumor + Keff;
    cO[i][j][z] = -Keff;
}
} //bottom boundary of tumor

else if (i == LTX & j >= LTY & j <= RTY & z >= TT & z <= TT + LT) {

    bO[i][j][z] = -kTumor;
    aO[i][j][z] = (kTumor + Keff)*deltaT/(deltaX*deltaX);
    cO[i][j][z] = -kTumor*deltaT/(deltaX*deltaX);
}
} //Tumor Boundary at NX/3
else if (i==RTX && j>=LTY && j<=RTY &&
  z>=TT && z<=TT+LT)
  bO[i][j][z] = -kTumor*deltaT/(deltaX*deltaX);
  aO[i][j][z] = (kTumor+k3)*deltaT/(deltaX*deltaX);
  cO[i][j][z] = -k3*deltaT/(deltaX*deltaX);
}  //Tumor Boundary at 2*NX/3

else if (j==LTY  && i>=LTX && i<=RTX &&
  z>=TT && z<=TT + LT)
  bO[i][j][z] = -k3*deltaT/(deltaY*deltaY);
  aO[i][j][z] = (kTumor+k3)*deltaT/(deltaY*deltaY);
  cO[i][j][z] = -kTumor*deltaT/(deltaY*deltaY);
}  //Tumor Boundary at NY/3

else if (j==RTY && i>=LTX && i<=RTX &&
  z>=TT && z<=TT + LT)
  bO[i][j][z] = -kTumor*deltaT/(deltaY*deltaY);
  aO[i][j][z] = (kTumor+k3)*deltaT/(deltaY*deltaY);
  cO[i][j][z] = -k3*deltaT/(deltaY*deltaY);
}  // Tumor Boundary at 2*NY/3

else{
  bO[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);
  aO[i][j][z] =
  2.0*p3*C3*(1.0+tau*Wb3*Cb3/(p3*C3)+2.0*tau/deltaT) + Wb3*Cb3*deltaT
  +
  (4.0*k3*deltaT)*(1.0/(deltaX*deltaX) + 1.0/(deltaY*deltaY))
  + (2.0*k3*deltaT)/(deltaZ*deltaZ);
  cO[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);
}  //Outside Tumor

}  //z

}  //i

return; //Exit Initialize

*******************************************************************************
END OF
INITIALIZATION AND MEMORY ALLOCATION
OF SYSTEM (HEAT SOURCES, INITIAL
TEMPERATURES, TRIDIAGONAL SYSTEM,
VASCULAR SYSTEM
PARAMETERS)*******************************************************************************

*******************************************************************************BEGIN CLEAR
MEMORY*******************************************************************************

void clearMem()
{
  if (Q1) delete [] Q1;
  if (Q2) delete [] Q2;
  if (Q3) delete [] Q3;
    if (T1) delete [] T1;
    if (T1_n1) delete [] T1_n1;
    if (T1_n1_1) delete [] T1_n1_1;
    if (T1_n) delete [] T1_n;
    if (U1) delete [] U1;
    if (U1_n1) delete [] U1_n1;
      if (a) delete [] a;
      if (b) delete [] b;
      if (c) delete [] c;
      if (d) delete [] d;
      if (Tb1_n1) {delete [] Tb1_n1[0]; delete []
                    Tb1_n1[1]; delete Tb1_n1;}  
      if (Tb2_n1) {delete [] Tb2_n1[0]; delete []
                    Tb2_n1[1]; delete Tb2_n1;}  
      if (Tb3_n1) {delete [] Tb3_n1[0]; delete []
                    Tb3_n1[1]; delete Tb3_n1;}  
      if (Tb4_n1) {delete [] Tb4_n1[0]; delete []
                    Tb4_n1[1]; delete Tb4_n1;}
}
if(Tb5_n1) {delete [] Tb5_n1[0]; delete [] Tb5_n1[1]; delete Tb5_n1;}
if(Tb6_n1) {delete [] Tb6_n1[0]; delete [] Tb6_n1[1];
if(Tb7_n1) {delete [] Tb7_n1[0]; delete [] Tb7_n1[1]; delete Tb7_n1;}
if(Tv1_n1) {delete [] Tv1_n1[0]; delete [] Tv1_n1[1]; delete Tv1_n1;}
if(Tv2_n1) {delete [] Tv2_n1[0]; delete [] Tv2_n1[1]; delete Tv2_n1;}
if(Tv3_n1) {delete [] Tv3_n1[0]; delete [] Tv3_n1[1]; delete Tv3_n1;}
if(Tv4_n1) {delete [] Tv4_n1[0]; delete [] Tv4_n1[1]; delete Tv4_n1;}
if(Tv5_n1) {delete [] Tv5_n1[0]; delete [] Tv5_n1[1];
if(Tv6_n1) {delete [] Tv6_n1[0]; delete [] Tv6_n1[1]; delete Tv6_n1;}
if(Tv7_n1) {delete [] Tv7_n1[0]; delete [] Tv7_n1[1]; delete Tv7_n1;}
writeLog("Memory released.");
return;
}
******************************************************************************
CLEAR MEMORY******************************************************************************
******************************************************************************BEGIN SET
VESSEL BORDER ROUTINE (SETS LOCATION OF ALL BLOOD VESSELS IN THEIR RESPECTIVE CARTESIAN COORDINATES)******************************************************************************
******************************************************************************
void setVesselBorder() {

//Set all vessel locations along the x-direction for the arterial system
X1[0] = NX;
X2b[0] = NX-LX1A;
X2a[0] = X2b[0]-LX2A;
X3a[0] = X2b[0]-LX2A/2-LX3A/2;
X3b[0] = X2b[0]-LX2A/2+LX3A/2;
X4a[0] = X2b[0]-LX2A/2-LX4A/2;
X4b[0] = X2b[0]-LX2A/2+LX4A/2;
X5a[0] = X4a[0]-LX5A;
X5b[0] = X4b[0]+LX5A;
X6a[0] = X4a[0]-LX5A/2-LX6A/2;
X6b[0] = X4b[0]+LX5A/2-LX6A/2;
X6a[0] = X4b[0]+LX5A/2+LX6A/2;
X6b[0] = X4b[0]+LX5A/2+LX6A/2;
X7a[0] = X4a[0]-LX5A/2-LX7A/2;
X7b[0] = X4a[0]-LX5A/2+LX7A/2;
X7a[0] = X4b[0]+LX5A/2-LX7A/2;
X7b[0] = X4b[0]+LX5A/2+LX7A/2;

//Set all vessel locations along the y-direction for the arterial system
Y1a[0] = centerY-LY1A/2;
Y1b[0] = centerY+LY1A/2;
Y2a[0] = centerY-LY2A/2;
Y2b[0] = centerY+LY2A/2;
Y3a[0] = Y2a[0]-LY3A;
Y3b[0] = Y2b[0]+LY3A;
Y4a[0] = Y2a[0]-LY3A/2-LY4A/2;
Y4b[0] = Y2a[0]-LY3A/2+LY4A/2;
Y4a[0] = Y2b[0]+LY3A/2-LY4A/2;
Y4b[0] = Y2b[0]+LY3A/2+LY4A/2;
Y5a[0] = Y2a[0]-LY3A/2-LY5A/2;
Y5b[0] = Y2a[0]-LY3A/2+LY5A/2;
Y5a[0] = Y2b[0]+LY3A/2-LY5A/2;
Y5b[0] = Y2b[0]+LY3A/2+LY5A/2;
Y6a1[0] = Y5a1[0]+LY6A;
Y6b1[0] = Y5b1[0]+LY6A;
Y6a2[0] = Y5a2[0]+LY6A;
Y6b2[0] = Y5b2[0]+LY6A;
Y7a1[0] = Y5a1[0]-LY6A-LY7A/2;
Y7b1[0] = Y5a1[0]-LY6A+LY7A/2;
Y7a2[0] = Y5b2[0]+LY6A/2-LY7A/2;
Y7b2[0] = Y5b2[0]+LY6A/2+LY7A/2;
Y7a3[0] = Y5a2[0]+LY6A/2-LY7A/2;
Y7b3[0] = Y5b2[0]+LY6A/2+LY7A/2;

//set all vessel locations along the z-direction
for the arterial system
Z1a[0] = (NZ3+NZ2)/2-LZ1A/2;
Z1b[0] = Z1a[0]-LZ1A;
Z2a[0] = (NZ3+NZ2)/2+LZ2A/2;
Z2b[0] = Z2a[0]+LZ2A;
Z3a[0] = (NZ3+NZ2)/2-LZ3A/2;
Z3b[0] = Z3a[0]+LZ3A;
Z4a[0] = Z3a[0]-LZ4A;
Z4b[0] = Z3b[0]+LZ4A;
Z5a1[0] = Z3a[0]-LZ4A/2-LZ5A/2;
Z5b1[0] = Z3a[0]-LZ4A/2+LZ5A/2;
Z5a2[0] = Z3b[0]+LZ4A/2-LZ5A/2;
Z5b2[0] = Z3b[0]+LZ4A/2+LZ5A/2;
Z6a1[0] = Z3a[0]-LZ6A/2-LZ6A/2;
Z6b1[0] = Z3a[0]-LZ6A/2+LZ6A/2;
Z6a2[0] = Z3b[0]+LZ6A/2-LZ6A/2;
Z6b2[0] = Z3b[0]+LZ6A/2+LZ6A/2;
Z7a1[0] = Z6a1[0]-LZ7A;
Z7b1[0] = Z6b1[0]+LZ7A;

Z7a2[0] = Z6a2[0]-LZ7A;
Z7b2[0] = Z6b2[0]+LZ7A;

//set all vessel locations along the x-direction
for the venous system
X1[1] = NX;
X2b[1] = NX-LX1A;
X2a[1] = X2b[1]-LX2A;
X3a[1] = X2b[1]-LX2A/2-LX3A/2;
X3b[1] = X2b[1]-LX2A/2+LX3A/2;
X4a[1] = X2b[1]-LX2A/2-LX4A/2;
X4b[1] = X2b[1]-LX2A/2+LX4A/2;
X5a[1] = X4a[1]-LX5A;
X5b[1] = X4b[1]+LX5A;
X6a1[1] = X4a[1]-LX5A/2-LX6A/2;
X6b1[1] = X4a[1]-LX5A/2+LX6A/2;
X6a2[1] = X4b[1]-LX5A/2-LX6A/2;
X6b2[1] = X4b[1]-LX5A/2+LX6A/2;
X7a1[1] = X4a[1]-LX5A/2-LX7A/2;
X7b1[1] = X4a[1]-LX5A/2+LX7A/2;
X7a2[1] = X4b[1]-LX5A/2-LX7A/2;
X7b2[1] = X4b[1]-LX5A/2+LX7A/2;

//set all vessel locations along the y-direction
for the venous system
Y1a[1] = centerY-LY1A/2;
Y1b[1] = centerY+LY1A/2;
Y2a[1] = centerY-LY2A/2;
Y2b[1] = centerY+LY2A/2;
Y3a[1] = Y2a[1]-LY3A;
Y4a[1] = Y2a[1]-LY3A/2-LY4A/2;
Y4b[1] = Y2a[1]-LY3A/2+LY4A/2;
Y4a2[1] = Y2b[1]-LY3A/2-LY4A/2;
Y4b2[1] = Y2b[1]-LY3A/2+LY4A/2;
Y5a1[1] = Y2a[1]-LY3A/2-LY5A/2;  
Y5b1[1] = Y2a[1]-LY3A/2+LY5A/2;  
Y6a1[1] = Y5a1[1]-LY6A;  
Y6a2[1] = Y5a2[1]-LY6A;  
Y7a3[1] = Y5a3[1]-LY7A;  
Z6b1[1] = Z3a[1]-LZ4A/2+LZ6A/2;  
Z6a2[1] = Z3b[1]-LZ4A/2-LZ6A/2;  
Z6b2[1] = Z3b[1]+LZ4A/2+LZ6A/2;  
Z7a1[1] = Z6a1[1]-LZ7A;  
Z7a2[1] = Z6a2[1]-LZ7A;  

//set all vessel locations in the z-direction for the venous system  
Z1a[1] = (NZ3+NZ2)/2-LZ1A/2;  
Z1b[1] = Z1a[1]+LZ1A;  
Z2a[1] = (NZ3+NZ2)/2-LZ2A/2;  
Z3a[1] = (NZ3+NZ2)/2-LZ3A/2;  
Z3b[1] = Z3a[1]+LZ3A;  
Z4a[1] = Z3a[1]-LZ4A;  
Z5a1[1] = Z3a[1]-LZ4A/2-LZ5A/2;  
Z5b1[1] = Z3a[1]-LZ4A/2+LZ5A/2;  
Z5a2[1] = Z3b[1]+LZ4A/2-LZ5A/2;  
Z5b2[1] = Z3b[1]+LZ4A/2+LZ5A/2;  
Z6a1[1] = Z3a[1]-LZ4A/2-LZ6A/2;  
Z6b2[1] = Z3b[1]+LZ4A/2+LZ6A/2;  
Z7a1[1] = Z6a1[1]-LZ7A;  
Z7a2[1] = Z6a2[1]-LZ7A;  

//separate the blood vessels  
int dx0 = 0; //offset of arteries  
int dx1 = 38*SCALEX; //offset of veins, made to scale when testing for grid independence  
int dz = 150;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] -= dx0;  
X[0] += 0; //if dx1 > 0, then X[1] will be out of range, so LX1B-dx1  
X[1] += dx1;
VESSEL BORDER ROUTINE (SETS LOCATION OF ALL BLOOD VESSELS IN THEIR RESPECTIVE CARTERSIAN COORDINATES)*********************************************************************

BEGIN RESET ROUTINE (TO REINITIALIZE SYSTEM TO STARTING VALUES)*********************************************************************

return;

******************************************************************************END OF SET VESSEL BORDER ROUTINE (SETS LOCATION OF ALL BLOOD VESSELS IN THEIR RESPECTIVE CARTERSIAN COORDINATES)******************************************************************************
void Reset()
{
    memset(a, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(b, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(c, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(d, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(Q1, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(Q2, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(Q3, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    for (int i=0; i<=NX; i++)
        for (int j=0; j<=NY; j++)
            for (int k=0; k<=NZ3; k++)
                Tt[i][j][k] = 34.0;
    for (int i=0; i<=NX; i++)
        for (int j=0; j<=NY; j++)
            for (int k=0; k<=NZ3; k++)
                Ttijkl[i][j][k] = 34.0;
    for (int i=0; i<=NX; i++)
        for (int j=0; j<=NY; j++)
            for (int k=0; k<=NZ3; k++)
                Ut[i][j][k] = 34.0;
    for (int i=0; i<=NX; i++)
        for (int j=0; j<=NY; j++)
            for (int k=0; k<=NZ3; k++)
                Utijkl[i][j][k] = 34.0;
    for (j=0; j<=LX1A; j++)
    {
        Tbd1[0][0][j] = 34.0;
        Tbd1[0][1][j] = 34.0;
        Tbd1[0][2][j] = 34.0;
        Tbd1[0][3][j] = 34.0;
    }
    for (j=0; j<=LX1B; j++)
    {
        Tbd1[1][0][j] = 34.0;
        Tbd1[1][1][j] = 34.0;
        Tbd1[1][2][j] = 34.0;
        Tbd1[1][3][j] = 34.0;
        Tbd1n1[0][j] = 37.0;
        Tbd1n1[1][j] = 37.0;
    }
    for (j=0; j<=LY2A; j++)
    {
        Tbd2[0][0][j] = 34.0;
        Tbd2[0][1][j] = 34.0;
        Tbd2[0][2][j] = 34.0;
        Tbd2[0][3][j] = 34.0;
        Tbd2n1[0][j] = 37.0;
        Tbd2n1[1][j] = 37.0;
    }
    for (j=0; j<=LY2B; j++)
    {
        Tbd2[1][0][j] = 34.0;
        Tbd2[1][1][j] = 34.0;
        Tbd2[1][2][j] = 34.0;
        Tbd2[1][3][j] = 34.0;
        Tbd2n2[0][j] = 37.0;
        Tbd2n2[1][j] = 37.0;
    }
    for (j=0; j<=LZ3A; j++)
    {
        Tbd3[0][0][j] = 34.0;
        Tbd3[0][1][j] = 34.0;
        Tbd3[0][2][j] = 34.0;
        Tbd3[0][3][j] = 34.0;
        Tbd3n1[0][j] = 37.0;
        Tbd3n1[1][j] = 37.0;
        Tbd3n2[0][j] = 37.0;
        Tbd3n2[1][j] = 37.0;
    }
    for (j=0; j<=LZ3B; j++)
    {
        Tbd3[1][0][j] = 34.0;
        Tbd3[1][1][j] = 34.0;
        Tbd3[1][2][j] = 34.0;
        Tbd3[1][3][j] = 34.0;
        Tbd3n3[0][j] = 37.0;
        Tbd3n3[1][j] = 37.0;
        Tbd3n4[0][j] = 37.0;
        Tbd3n4[1][j] = 37.0;
    }
    for (j=0; j<=LZ4A; j++)
    {
        Tbd4[0][0][j] = 34.0;
        Tbd4[0][1][j] = 34.0;
        Tbd4[0][2][j] = 34.0;
        Tbd4[0][3][j] = 34.0;
        Tbd4n1[0][j] = 37.0;
        Tbd4n1[1][j] = 37.0;
        Tbd4n2[0][j] = 37.0;
        Tbd4n2[1][j] = 37.0;
    }
    for (j=0; j<=LZ4B; j++)
    {
        Tbd4[1][0][j] = 34.0;
        Tbd4[1][1][j] = 34.0;
        Tbd4[1][2][j] = 34.0;
        Tbd4[1][3][j] = 34.0;
        Tbd4n3[0][j] = 37.0;
        Tbd4n3[1][j] = 37.0;
        Tbd4n4[0][j] = 37.0;
        Tbd4n4[1][j] = 37.0;
    }
}


\{
\begin{align*}
Tbd3[0][0][j] &= 34.0; \\
Tbd3[0][1][j] &= 34.0; \\
Tbd3[0][2][j] &= 34.0; \\
Tbd3[0][3][j] &= 34.0; \\
Tb3_n[0][j] &= 37.0; \\
Tv3_n[0][j] &= 37.0; \\
\end{align*}
\}

for (j=0; j<=2*LZ3B+1; j++)

\{
\begin{align*}
Tbd3[1][0][j] &= 34.0; \\
Tbd3[1][1][j] &= 34.0; \\
Tbd3[1][2][j] &= 34.0; \\
Tbd3[1][3][j] &= 34.0; \\
Tb3_n[1][j] &= 37.0; \\
Tv3_n[1][j] &= 37.0; \\
\end{align*}
\}

for (j=0; j<=4*LX4A+3; j++)

\{
\begin{align*}
Tbd4[0][0][j] &= 34.0; \\
Tbd4[0][1][j] &= 34.0; \\
Tbd4[0][2][j] &= 34.0; \\
Tbd4[0][3][j] &= 34.0; \\
Tb4_n[0][j] &= 37.0; \\
Tv4_n[0][j] &= 37.0; \\
\end{align*}
\}

for (j=0; j<=8*LY5A+7; j++)

\{
\begin{align*}
Tbd5[0][0][j] &= 34.0; \\
Tbd5[0][1][j] &= 34.0; \\
Tbd5[0][2][j] &= 34.0; \\
Tbd5[0][3][j] &= 34.0; \\
Tb5_n[0][j] &= 37.0; \\
Tv5_n[0][j] &= 37.0; \\
\end{align*}
\}

for (j=0; j<=8*LY5B+7; j++)

\{
\begin{align*}
Tbd5[1][0][j] &= 34.0; \\
Tbd5[1][1][j] &= 34.0; \\
Tbd5[1][2][j] &= 34.0; \\
Tbd5[1][3][j] &= 34.0; \\
Tb5_n[1][j] &= 37.0; \\
Tv5_n[1][j] &= 37.0; \\
\end{align*}
\}

for (j=0; j<=16*LZ6A+15; j++)

\{
\begin{align*}
Tbd6[0][0][j] &= 34.0; \\
Tbd6[0][1][j] &= 34.0; \\
Tbd6[0][2][j] &= 34.0; \\
Tbd6[0][3][j] &= 34.0; \\
Tb6_n[0][j] &= 37.0; \\
Tv6_n[0][j] &= 37.0; \\
\end{align*}
\}

for (j=0; j<=16*LZ6B+15; j++)

\{
\begin{align*}
Tbd6[1][0][j] &= 34.0; \\
Tbd6[1][1][j] &= 34.0; \\
Tbd6[1][2][j] &= 34.0; \\
Tbd6[1][3][j] &= 34.0; \\
Tb6_n[1][j] &= 37.0; \\
Tv6_n[1][j] &= 37.0; \\
\end{align*}
\}
SUBROUTINE (SUBROUTINE THAT CALCULATES ALL TEMPERATURE GENERATED IN MODEL)******************

int CalcAll(double P0, bool dp)
{
    double maxErr, oldE; // sum of square error of tissue temperatures
    FILE *fp1;
    FILE *fp2;
    fp1 = fopen("T_center_t_0.txt","w");
    fp2 = fopen("T_center_t_100.txt","w");
    Reset(); // Comment out to not run with laser;Stop laser and nanoparticles for test
    t = 0;
    //////////////////////////////////////////////////////////////////////////////////
    getTv_blood(0);
    getTv_blood(1);
    // calculate blood temperature based on given vessel temperature
    CalcTb();
    CalcTb2();
    reloadbloodtemperature(0);
    reloadbloodtemperature(1);
    CalcVessel(0);
   CalcVessel(1);
    //////////////////////////////////////////////////////////////////////////////////
    while((t*deltaT < TOTAL_T))
    {
        // while loop begin
        // extract results at different time steps
        if (t*deltaT == 0 && Optimized == true)
        {

        }
writeSquareXY(0, 0, NX, NY, 0, t, 1);
//Contour at z = 0cm
writeSquareXZ(0, 0, NX, NZ3, centerY, t, 1);
//Contour at y = .81cm
writeSquareXZ(0, 0, NX, NZ3, LTY, t, 1);
//Contour at y = .54cm
writeSquareXZ(0, 0, NX, NZ3, RTY, t, 1);
//Contour at y = 1.08cm
writeSquareYZ(0, 0, NY, NZ3, centerX, t, 1);
//Contour at x = .81cm
writeSquareYZ(0, 0, NY, NZ3, LTX, t, 1);
//Contour at x = .54cm
writeSquareYZ(0, 0, NY, NZ3, RTX, t, 1);
//Contour at x = 1.08cm
writeSquareXY(0, 0, NX, NY, centerTZ, t, 1);
//Tumor Cross Section XY in middle
writeSquareXY(0, 0, NX, NY, TT, t, 1);
//Tumor top and dermis/epiderm boundary
writeSquareXY(0, 0, NX, NY, TT+LT, t, 1);
//Tumor Bottom
writeZCenter(t);
writeQCenter(t);
}  //getting results

if( t == 1 && Optimized == true) {
writeSquareXY(0, 0, NX, NY, 0, t, 1);
//Contour at z = 0cm
writeSquareXZ(0, 0, NX, NZ3, centerY, t, 1);
//Contour at y = .81cm
writeSquareXZ(0, 0, NX, NZ3, LTY, t, 1);
//Contour at y = .54cm
writeSquareXZ(0, 0, NX, NZ3, RTY, t, 1);
//Contour at y = 1.08cm
writeSquareYZ(0, 0, NY, NZ3, centerX, t, 1);
//Contour at x = .81cm
writeSquareYZ(0, 0, NY, NZ3, LTX, t, 1);
//Contour at x = .54cm
writeSquareYZ(0, 0, NY, NZ3, RTX, t, 1);
//Contour at x = 1.08cm
writeSquareXY(0, 0, NX, NY, centerTZ, t, 1);
//Tumor Cross Section XY in middle
writeSquareXY(0, 0, NX, NY, TT, t, 1);
//Tumor top and dermis/epiderm boundary
writeSquareXY(0, 0, NX, NY, TT+LT, t, 1);
//Tumor Bottom
writeZCenter(t);
writeQCenter(t);
}  //getting results

if( t == 1 && Optimized == true) {
writeSquareXY(0, 0, NX, NY, 0, t, 1);
//Contour at z = 0cm
writeSquareXZ(0, 0, NX, NZ3, centerY, t, 1);
//Contour at y = .81cm
writeSquareXZ(0, 0, NX, NZ3, LTY, t, 1);
//Contour at y = .54cm
writeSquareXZ(0, 0, NX, NZ3, RTY, t, 1);
//Contour at y = 1.08cm
writeSquareYZ(0, 0, NY, NZ3, centerX, t, 1);
//Contour at x = .81cm
writeSquareYZ(0, 0, NY, NZ3, LTX, t, 1);
//Contour at x = .54cm
writeSquareYZ(0, 0, NY, NZ3, RTX, t, 1);
//Contour at x = 1.08cm
writeSquareXY(0, 0, NX, NY, centerTZ, t, 1);
//Tumor Cross Section XY in middle
writeSquareXY(0, 0, NX, NY, TT, t, 1);
//Tumor top and dermis/epiderm boundary
writeSquareXY(0, 0, NX, NY, TT+LT, t, 1);
//Tumor Bottom
writeZCenter(t);
writeQCenter(t);
}  //getting results

if( t == 1 && Optimized == true) {
writeSquareXY(0, 0, NX, NY, 0, t, 1);
//Contour at z = 0cm
writeSquareXZ(0, 0, NX, NZ3, centerY, t, 1);
//Contour at y = .81cm
writeSquareXZ(0, 0, NX, NZ3, LTY, t, 1);
//Contour at y = .54cm
writeSquareXZ(0, 0, NX, NZ3, RTY, t, 1);
//Contour at y = 1.08cm
writeSquareYZ(0, 0, NY, NZ3, centerX, t, 1);
//Contour at x = .81cm
writeSquareYZ(0, 0, NY, NZ3, LTX, t, 1);
//Contour at x = .54cm
writeSquareYZ(0, 0, NY, NZ3, RTX, t, 1);
//Contour at x = 1.08cm
writeSquareXY(0, 0, NX, NY, centerTZ, t, 1);
//Tumor Cross Section XY in middle
writeSquareXY(0, 0, NX, NY, TT, t, 1);
//Tumor top and dermis/epiderm boundary
writeSquareXY(0, 0, NX, NY, TT+LT, t, 1);
//Tumor Bottom
writeZCenter(t);
writeQCenter(t);
}  //getting results


writeSquareXY(0, 0, NX, NY, 0, t, 1);
// Contour at z = 0cm
writeSquareXZ(0, 0, NX, Nz3, centerY, t, 1);
// contour at y = .81cm
writeSquareXZ(0, 0, NX, Nz3, LTY, t, 1);
// contour at y = .54cm
writeSquareXZ(0, 0, NX, Nz3, RTY, t, 1);
// contour at y = 1.08cm
writeSquareYZ(0, 0, NY, Nz3, centerX, t, 1);
// contour at x = .81cm
writeSquareYZ(0, 0, NY, Nz3, LTX, t, 1);
// contour at x = .54cm
writeSquareYZ(0, 0, NY, Nz3, RTX, t, 1);
// contour at x = 1.08cm
writeSquareXY(0, 0, NX, NY, centerTZ, t, 1);
// Tumor Cross Section XY in middle
writeSquareXY(0, 0, NX, NY, TT, t, 1);
// Tumor top and dermis/epiderm boundary
writeSquareXY(0, 0, NX, NY, TT+LT, t, 1);
// Tumor Bottom
writeZCenter(t);
writeQCenter(t);
}
// getting results at t=10.
if( t % 500 == 0 && Optimized == true){
writeSquareXY(0, 0, NX, NY, 0, t, 1);
// Contour at z = 0cm
writeSquareXZ(0, 0, NX, Nz3, centerY, t, 1);
// contour at y = .81cm
writeSquareXZ(0, 0, NX, Nz3, LTY, t, 1);
// contour at y = .54cm
writeSquareXZ(0, 0, NX, Nz3, RTY, t, 1);
// contour at y = 1.08cm
writeSquareYZ(0, 0, NY, Nz3, centerX, t, 1);
// contour at x = .81cm
writeSquareYZ(0, 0, NY, Nz3, LTX, t, 1);
// contour at x = .54cm
writeSquareYZ(0, 0, NY, Nz3, RTX, t, 1);
// contour at x = 1.08cm
writeSquareXY(0, 0, NX, NY, centerTZ, t, 1);
// Tumor Cross Section XY in middle
writeSquareXY(0, 0, NX, NY, TT, t, 1);
// Tumor top and dermis/epiderm boundary
writeSquareXY(0, 0, NX, NY, TT+LT, t, 1);
// Tumor Bottom
writeZCenter(t);
writeQCenter(t);
}
// getting results at every 50 seconds (assuming deltaT = .1)
1 = 0;
maxErr = 0.0;
oldE = 99999999.0f;
do { // iteration, do while loop
    begin
        I++;
        // Calculate tissue temperature
        maxErr = CalcTt();
       printf("t: %d I: %d Err: %5.4lf T0: %10.6f T1: %10.6f T2: %10.6f\n", t, I, maxErr, Tt_n1[NX/2][NY/2][0], Tt_n1[0][NY/2][0], Tt_n1[NX/2][NY/2][centerTZ]);
        if(maxErr>=oldE)
            {
                writeLog("========== unstable ==========");
                #ifdef SCREEN_OUT
                printf("==========\n");
                #endif
                writeSquareXZ(0, 0, NX, NZ3, NY/2, t, 1);
                break;
            }
        oldE = maxErr;
    }
while(maxErr>Err_I); //do while
I loop end, Err_I=0.001
if(bPowerOn)
{
   if(dp)
   
   sprintf(tmp, "p%d+ l:9d Err:%5.4lf T0:%7.4lf T1:%7.4lf T2:%7.4lf P:%7.4lf", 
   loopP, t, 
   I, maxErr, Tt_nl[NX/2][NY/2][0], 
   Tt_nl[0][NY/2][0], 
   Tt_nl[NX/2][NY/2][centerTZ],0.0);
   else
   
   sprintf(tmp, "p%d+ l:9d Err:%5.4lf T0:%7.4lf T1:%7.4lf T2:%7.4lf P:%7.4lf", 
   loopP, t, 
   I, maxErr, Tt_nl[NX/2][NY/2][0], 
   Tt_nl[0][NY/2][0], 
   Tt_nl[NX/2][NY/2][centerTZ],0.0);
   
   if(dp)
   
   sprintf(tmp, "p%d t:9d I:9d Err:%5.4lf T0:%7.4lf T1 :%7.4lf T2:%7.4lf P:%7.4lf', 
   loopP, t, 
   I, maxErr, Tt_nl[NX/2][NY/2][0], 
   Tt_nl[0][NY/2][0], 
   Tt_nl[NX/2][NY/2][centerTZ],P0);
   else
   
   sprintf(tmp, "p%d+ t:9d I:9d Err:%5.4lf T0:%7.4lf T1 :%7.4lf T2:%7.4lf P:%7.4lf', 
   loopP, t, 
   I, maxErr, Tt_nl[NX/2][NY/2][0], 
   Tt_nl[0][NY/2][0], 
   Tt_nl[NX/2][NY/2][centerTZ],P0);
   
   writeLog(tmp);
   fprintf(fp2,"p%d %10.6f,",t, 
   Tt_nl[NX/2][NY/2][100]);

   // Calculating Sijk at the n+1 time step
   for (i=0; i<=NX; i++)
   
   for (j=0; j<=NY; j++)
   
   for (z=0; z<=NZ; z++)
   
   U_t_nl[i][j][z] = U[i][j][z] +
   (1.0+t^r*x*B_tumor*C_b3/(2.*C_f)*
   +2.0*t^r/(t^r*x)*T_t_nl[i][j][z]
   +(1.0+t^r*x*B_tumor*C_b3/(2.*C_f))*
   2.0*t^r/(t^r*x)*T_t[i][j][z];
   
   for (z=0; z<=NZ; z++)
   
   U_t_nl[i][j][z] = U[i][j][z] +
   (1.0+t^r*x*B_tumor*C_b3/(2.*C_f)*
   +2.0*t^r/(t^r*x)*T_t_nl[i][j][z]
   +(1.0+t^r*x*B_tumor*C_b3/(2.*C_f))*
   2.0*t^r/(t^r*x)*T_t[i][j][z];
   
   for (z=0; z<=NZ; z++)
   
   U_t_nl[i][j][z] = U[i][j][z] +
   (1.0+t^r*x*B_tumor*C_b3/(2.*C_f)*
   +2.0*t^r/(t^r*x)*T_t_nl[i][j][z]
   +(1.0+t^r*x*B_tumor*C_b3/(2.*C_f))*
   2.0*t^r/(t^r*x)*T_t[i][j][z];
   
   for (z=0; z<=NZ; z++)
   
   U_t_nl[i][j][z] = U[i][j][z] +
   (1.0+t^r*x*B_tumor*C_b3/(2.*C_f)*
   +2.0*t^r/(t^r*x)*T_t_nl[i][j][z]
   +(1.0+t^r*x*B_tumor*C_b3/(2.*C_f))*
   2.0*t^r/(t^r*x)*T_t[i][j][z];
+(1.0+tau*Wb*Tumor*Cb3/(pTumor*Ctumor)-
2.0*tau/deltaT)*Tt[i][j][z];
}  //Inside Tumor with no
nanoparticles in region
}
}  //Inside Tumor
else{
Ut_nl[i][j][z] = -Ut[i][j][z] +
(1.0+tau*Wb3*Cb3/(p3*C3))
+2.0*tau/deltaT)*Tt_n1[i][j][z]
+(1.0+tau*Wb3*Cb3/(p3*C3)-
2.0*tau/deltaT)*Tt[i][j][z];
}  //Outside Tumor
}  //z
}  //j
}  //i
memcpy(Ut, Ut_nl,
sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
memcpy(Tt, Tt_n1,
sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
memcpy(Damage_t, Damage_n1,
sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
AdjustPower(PO);
if(t+1 == TOTAL_T*10 && Optimized == true){
writeSquareXY(0, 0, NX, NY, 0, t+1, 1);
//Contour at z = 0cm
writeSquareXZ(0, 0, NX, NZ3, centerY, t+1, 1);
//contour at y = .81 cm
writeSquareXZ(0, 0, NX, NZ3, LTY, t+1, 1);
//contour at y = .54 cm
writeSquareXZ(0, 0, NX, NZ3, RTY, t+1, 1);
//contour at y = 1.08 cm
writeSquareXY(0, 0, NX, NY, centerX, t+1, 1);
//Contour at x = .81 cm
writeSquareXY(0, 0, NX, NY, LTX, t+1, 1);
//contour at x = .54 cm
writeSquareXY(0, 0, NX, NY, RTX, t+1, 1);
//contour at x = 1.08 cm
}
}  //Getting results at t=Total_T, thinking a bug
occurs outside of time loop with data extraction
}  //Time while loop block end, upper bound
TOTAL_T
fclose(fp2);
printf("Record the temperature of the last
second\n");
return(1);
}  //programm finish
*******************************************************************************/
CALL SUBROUTINE*******************
** * * * * * * Q p
** SETBOARDERVAR1ABLE SUBROUTINE (ONE
** SUBROUTINE TO HANDLE THE BORDER
** PLACEMENT OF THE VESSELS, INPUT 0 FOR
** ARTERIES, 1 FOR
** VEINS)*******************************************************************************/

void setboarderVariable(int index)
{
x1a=X1[index]; x2a=X2a[index]; x2b=X2b[index]; x3a=X3a[index]; x3b=X3b[index];
x4a=X4a[index]; x4b=X4b[index];
x5a=X5a[index]; x5b=X5b[index]; x6a=X6a1[index]; x6b=X6b1[index];
x6a2=X6a2[index]; x6b2=X6b2[index];
x7a1 = X7a1[index]; x7b1 = X7b1[index]; x7a2 = X7a2[index]; x7b2 = X7b2[index];
y1a = Y1a[index]; y1b = Y1b[index]; y2a = Y2a[index]; y2b = Y2b[index];
y3a = Y3a[index]; y3b = Y3b[index];
y4a1 = Y4a1[index]; y4b1 = Y4b1[index]; y4a2 = Y4a2[index]; y4b2 = Y4b2[index];
y5a1 = Y5a1[index]; y5b1 = Y5b1[index];
y5a2 = Y5a2[index]; y5b2 = Y5b2[index]; y6a1 = Y6a1[index]; y6b1 = Y6b1[index];
y6a2 = Y6a2[index]; y6b2 = Y6b2[index];
y7a1 = Y7a1[index]; y7b1 = Y7b1[index]; y7a2 = Y7a2[index]; y7b2 = Y7b2[index];
y7a3 = Y7a3[index]; y7b3 = Y7b3[index];
y7a4 = Y7a4[index]; y7b4 = Y7b4[index]; z1a = Z1a[index]; z1b = Z1b[index];
z2a = Z2a[index]; z2b = Z2b[index];
z3a = Z3a[index]; z3b = Z3b[index]; z4a = Z4a[index]; z4b = Z4b[index];
z5a1 = Z5a1[index]; z5b1 = Z5b1[index];
z5a2 = Z5a2[index]; z5b2 = Z5b2[index]; z6a1 = Z6a1[index]; z6b1 = Z6b1[index];
z6a2 = Z6a2[index];
z6b2 = Z6b2[index]; z7a1 = Z7a1[index]; z7b1 = Z7b1[index];
z7a2 = Z7a2[index]; z7b2 = Z7b2[index];

lx1 = index==0?LX1A:LX1B;
lx1 = index==0?LX1A:LX1B;
lx1 = index==0?LX1A:LX1B;
lx2 = index==0?LX2A:LX2B;

ly2 = index==0?LY2A:LY2B;
ly2 = index==0?LY2A:LY2B;
ly2 = index==0?LY2A:LY2B;
ly2 = index==0?LY2A:LY2B;

lz2 = index==0?LZ2A:LZ2B;
lz2 = index==0?LZ2A:LZ2B;
lz2 = index==0?LZ2A:LZ2B;
lz2 = index==0?LZ2A:LZ2B;

lx3 = index==0?LX3A:LX3B;
ly3 = index==0?LY3A:LY3B;
lz3 = index==0?LZ3A:LZ3B;
lz4 = index==0?LZ4A:LZ4B;
lx4 = index==0?LX4A:LX4B;
lx5 = index==0?LX5A:LX5B;
lx5 = index==0?LX5A:LX5B;
lx5 = index==0?LX5A:LX5B;
lx5 = index==0?LX5A:LX5B;

lz4 = index==0?LZ4A:LZ4B;
lx6 = index==0?LX6A:LX6B;
lz6 = index==0?LZ6A:LZ6B;
lz6 = index==0?LZ6A:LZ6B;
lx7 = index==0?LX7A:LX7B;
lz7 = index==0?LZ7A:LZ7B;
lz7 = index==0?LZ7A:LZ7B;
lz7 = index==0?LZ7A:LZ7B;

cenY = (y1a + y1b)/2;//actual center y after separate the arteries and veins

cenZ = (z1a + z1b)/2;

cenX = (x2a + x2b)/2; //symmetry center for x coordinate from level 2

return;

*****************************************************************************

************END OF SETVESSELBOARDER SUBROUTINE************

*****************************************************************************

************START OF GETTV_BLOOD SUBROUTINE************

(INTERPOLATES VESSEL TEMPERATURE FROM THE TISSUE TEMPERATURE NEAR THE BLOOD VESSEL)===============================================

*****************************************************************************

int getTv_blood(int index)

{
//interpolate vessel temperature from the tissue temperature near the vessel

//****LEFT and RIGHT side could be different if the blood vessel has an offset to the center

setborderVariable(index); //set the common blood borderer variables

//first level
for( i=0;i<lx1;i++)
    Tv1_n1[index][i] = ( Tbd1[index][0][i] + Tbd1[index][1][i] + Tbd1[index][2][i] + Tbd1[index][3][i] ) / 4.0f;

//second level
for( i=0;i<ly2;i++) //i=0 & LY2 are on the blood vessels
    Tv2_n1[index][i] = ( Tbd2[index][0][i] + Tbd2[index][1][i] + Tbd2[index][2][i] + Tbd2[index][3][i] ) / 4.0f;

//third level
for( k=0;k<2;k++)
    for( i=0;i<lx3;i++)
        Tv3_n1[index][i+k*(lx3+1)] = ( Tbd3[index][0][i+k*(lx3+1)] + Tbd3[index][1][i+k*(lx3+1)] + Tbd3[index][2][i+k*(lx3+1)] + Tbd3[index][3][i+k*(lx3+1)] ) / 4.0f;

//fourth level
for( r=0;r<2;r++)
    for( j=0;j<ly3;j++)
        for( i=0;i<lx4;i++)
            Tv4_n1[index][i+k*(lx4+1)] = ( Tbd4[index][0][i+k*(lx4+1)] + Tbd4[index][1][i+k*(lx4+1)] + Tbd4[index][2][i+k*(lx4+1)] + Tbd4[index][3][i+k*(lx4+1)] ) / 4.0f;

//fifth level
for( r=0;r<2;r++)
    for( j=0;j<ly4;j++)
        for( i=0;i<lx5;i++)
            Tv5_n1[index][i+k*(ly5+1)] = ( Tbd5[index][0][i+k*(ly5+1)] + Tbd5[index][1][i+k*(ly5+1)] + Tbd5[index][2][i+k*(ly5+1)] + Tbd5[index][3][i+k*(ly5+1)] ) / 4.0f;

//sixth level
for( r=0;r<2;r++)
    for( j=0;j<ly6;j++)
        for( i=0;i<lx6;i++)
for(p=0;p<2;p++)//repeat variable in x coordinate
{
    for(q=0;q<2;q++) //repeat variable in inner y coordinate
    {
        for(i=0;i<=lx6;i++)
        {
            Tv6_n1[index][i+k*(lx6+1)] = (Tbd6[index][0][i+k*(lx6+1)] + Tbd6[index][1][i+k*(lx6+1)] + Tbd6[index][2][i+k*(lx6+1)] + Tbd6[index][3][i+k*(lx6+1)] ) / 4.0f;
        }
        k++;
    }
}

//seventh level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
{
    for(j=0;j<2;j++) //repeat variable in y coordinate
    {
        for(p=0;p<2;p++)//repeat variable in x coordinate
        {
            for(q=0;q<2;q++) //repeat variable in inner y coordinate
            {
                for(h=0;h<2;h++)//repeat variable in inner z coordinate
                {
                    for(i=0;i<=lx7;i++)
                    {
                        Tv7_n1[index][i+k*(lx7+1)] = (Tbd7[index][0][i+k*(lx7+1)] + Tbd7[index][1][i+k*(lx7+1)] + Tbd7[index][2][i+k*(lx7+1)] + Tbd7[index][3][i+k*(lx7+1)] ) / 4.0f;
                    }
                    k++;
                }
            }
        }
    }
}
return(1);

****************** END OF GET_TV_BLOOD SUBROUTINE ***********

****************** START OF CALCULATE SUBROUTINE (CALCULATES THE TEMPERATURE IN THE ARTERIAL SYSTEM USING 4TH ORDER RUNGE-KUTTA)

int CalcTb() // calculate the value for artery
{
    //int index = 0;
    //double fkl, fk2, fk3, fk4;
    inde = 0;
    setborderVariable(inde);//set the common blood border variables
    Tb1_n1[index][0] = 37.0 + THETA0;
    //first level blood
    for(i=1;i<=lx1;i++)
    {
        fkl = factor1[inde]*Tv1_n1[inde][i-1]-Tb1_n1[inde][i-1]);
        fb2 = factor1[inde]*Tv1_n1[inde][i-1]- (Tb1_n1[inde][i-1]+fkl/2.0));
        fb3 = factor1[inde]*Tv1_n1[inde][i-1]- (Tb1_n1[inde][i-1]+fb2/2.0));
        fb4 = factor1[inde]*Tv1_n1[inde][i-1]- (Tb1_n1[inde][i-1]+fb3);
    }
\[ T_{b1_{n1}[inde][i]} = T_{b1_{n1}[inde][i-1]} + (f_{k1} + 2.0*f_{k2} + 2.0*f_{k3} + f_{k4})/6.0; \]

//second level blood
\[ T_{b2_{n1}[inde][ly/2]} = T_{b1_{n1}[inde][lx1]}; \]

//the interface grid point between level 1 and level 2
//left part
for (i=ly/2-1; i>=0; i--)
{
\[ f_{k1} = factor2[index]*(T_{v2_{n1}[inde][i+1]} - T_{b2_{n1}[inde][i+1]}); \]
\[ f_{k2} = factor2[index]*(T_{v2_{n1}[inde][i+1]} - (T_{b2_{n1}[inde][i+1]} + f_{k1}/2.0)); \]
\[ f_{k3} = factor2[index]*(T_{v2_{n1}[inde][i+1]} - (T_{b2_{n1}[inde][i+1]} + f_{k2}/2.0)); \]
\[ f_{k4} = factor2[index]*(T_{v2_{n1}[inde][i+1]} - (T_{b2_{n1}[inde][i+1]} + f_{k3})); \]
\[ T_{b2_{n1}[inde][i]} = T_{b2_{n1}[inde][i+1]} + (f_{k1} + 2.0*f_{k2} + 2.0*f_{k3} + f_{k4})/6.0; \]
}

//right part
for (i=ly/2+1; i<=ly2; i++)
{
\[ f_{k1} = factor2[index]*(T_{v2_{n1}[inde][i-1]} - T_{b2_{n1}[inde][i-1]}); \]
\[ f_{k2} = factor2[index]*(T_{v2_{n1}[inde][i-1]} - (T_{b2_{n1}[inde][i-1]} + f_{k1}/2.0)); \]
\[ f_{k3} = factor2[index]*(T_{v2_{n1}[inde][i-1]} - (T_{b2_{n1}[inde][i-1]} + f_{k2}/2.0)); \]
\[ f_{k4} = factor2[index]*(T_{v2_{n1}[inde][i-1]} - (T_{b2_{n1}[inde][i-1]} + f_{k3})); \]
\[ T_{b2_{n1}[inde][i]} = T_{b2_{n1}[inde][i-1]} + (f_{k1} + 2.0*f_{k2} + 2.0*f_{k3} + f_{k4})/6.0; \]
}

//third level blood
for (i=0; i<2*j++;)
{
\[ f_{k1} = factor3[index]*(T_{v3_{n1}[inde][i+1+j*(lz3+1)]} - T_{b3_{n1}[inde][i+1+j*(lz3+1)]}); \]
\[ f_{k2} = factor3[index]*(T_{v3_{n1}[inde][i+1+j*(lz3+1)]} + f_{k1}/2.0)); \]
\[ f_{k3} = factor3[index]*(T_{v3_{n1}[inde][i+1+j*(lz3+1)]} - (T_{b3_{n1}[inde][i+1+j*(lz3+1)]} + f_{k3})); \]
\[ T_{b3_{n1}[inde][i+j*(lz3+1)]} = T_{b3_{n1}[inde][i+j*(lz3+1)]} + (f_{k1} + 2.0*f_{k2} + 2.0*f_{k3} + f_{k4})/6.0; \]
}

//upper part
for (i=lz3/2-1; i>=0; i--)
{
\[ f_{k1} = factor3[index]*(T_{v3_{n1}[inde][i+1+j*(lz3+1)]} - T_{b3_{n1}[inde][i+1+j*(lz3+1)]}); \]
\[ T_{b3_{n1}[inde][i+j*(lz3+1)]} = (T_{b3_{n1}[inde][i+1+j*(lz3+1)]} + f_{k1}/2.0)); \]
\[ f_{k3} = factor3[index]*(T_{v3_{n1}[inde][i+1+j*(lz3+1)]} - (T_{b3_{n1}[inde][i+1+j*(lz3+1)]} + f_{k3})); \]
\[ T_{b3_{n1}[inde][i+j*(lz3+1)]} = (T_{b3_{n1}[inde][i+1+j*(lz3+1)]} + f_{k3})); \]
\[ T_{b3_{n1}[inde][i+j*(lz3+1)]} = T_{b3_{n1}[inde][i+j*(lz3+1)]} + (f_{k1} + 2.0*f_{k2} + 2.0*f_{k3} + f_{k4})/6.0; \]
}

//lower part
for (i=lz3/2+1; i<=lz3; i++)
{
\[ f_{k1} = factor3[index]*(T_{v3_{n1}[inde][i-1+j*(lz3+1)]} - T_{b3_{n1}[inde][i-1+j*(lz3+1)]}); \]
\[ f_{k2} = factor3[index]*(T_{v3_{n1}[inde][i-1+j*(lz3+1)]} + f_{k1}/2.0)); \]
\[ f_{k3} = factor3[index]*(T_{v3_{n1}[inde][i-1+j*(lz3+1)]} - (T_{b3_{n1}[inde][i-1+j*(lz3+1)]} + f_{k3})); \]
\[ T_{b3_{n1}[inde][i+j*(lz3+1)]} = T_{b3_{n1}[inde][i+j*(lz3+1)]} + (f_{k1} + 2.0*f_{k2} + 2.0*f_{k3} + f_{k4})/6.0; \]
\(-Tb3_{n1[\text{inde}]}[i-1+j*(lz3+1)]+fk4/2.0);\)

\[fk4 = \text{factor3[inde]}*(Tv3_{n1[\text{inde}]}[i-1+j*(lz3+1)] - (Tb3_{n1[\text{inde}]}[i-1+j*(lz3+1)]+fk3));\]

\[Tb3_{n1[\text{inde}]}[i+j*(lz3+1)] = Tb3_{n1[\text{inde}]}[i-1+j*(lz3+1)] + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;\]

}  

//fourth level blood

k=0;

for(r=0;r<2;r++)
{
  for(j=0;j<2;j++)
  {
    //the order of level 3 and level 4 are exception than others !!!!!!!!!!!
    Tb4_{n1[\text{inde}]}[lx4/2+k*(lx4+1)] = Tb3_{n1[\text{inde}]}[r*lz3+j*(lz3+1)];
    //left part
    for(i=lx4/2-1;i>=0;i--)
    {
      fk1 = \text{factor4[inde]}*(Tv4_{n1[\text{inde}]}[i+1+k*(lx4+1)] - Tb4_{n1[\text{inde}]}[i+k*(lx4+1)]);
      fk2 = \text{factor4[inde]}*(Tv4_{n1[\text{inde}]}[i+1+k*(lx4+1)] - (Tb4_{n1[\text{inde}]}[i+1+k*(lx4+1)]+fk1/2.0));
      fk3 = \text{factor4[inde]}*(Tv4_{n1[\text{inde}]}[i+1+k*(lx4+1)] - (Tb4_{n1[\text{inde}]}[i+1+k*(lx4+1)]+fk2/2.0));
      fk4 = \text{factor4[inde]}*(Tv4_{n1[\text{inde}]}[i+1+k*(lx4+1)] - (Tb4_{n1[\text{inde}]}[i+1+k*(lx4+1)]+fk3));
      Tb4_{n1[\text{inde}]}[i+k*(lx4+1)] = Tb4_{n1[\text{inde}]}[i+1+k*(lx4+1)] + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
    }
    k++;
  }

//right part

for(i=lx4/2+1;i<=lx4;i++)
{
  fk1 = \text{factor4[inde]}*(Tv4_{n1[\text{inde}]}[i-1+k*(lx4+1)] - Tb4_{n1[\text{inde}]}[i-1+k*(lx4+1)]);
  fk2 = \text{factor4[inde]}*(Tv4_{n1[\text{inde}]}[i-1+k*(lx4+1)] - (Tb4_{n1[\text{inde}]}[i-1+k*(lx4+1)]+fk1/2.0));
  fk3 = \text{factor4[inde]}*(Tv4_{n1[\text{inde}]}[i-1+k*(lx4+1)] - (Tb4_{n1[\text{inde}]}[i-1+k*(lx4+1)]+fk2/2.0));
  fk4 = \text{factor4[inde]}*(Tv4_{n1[\text{inde}]}[i-1+k*(lx4+1)] - (Tb4_{n1[\text{inde}]}[i-1+k*(lx4+1)]+fk3));
  Tb4_{n1[\text{inde}]}[i+k*(lx4+1)] = Tb4_{n1[\text{inde}]}[i-1+k*(lx4+1)] + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
}

//fifth level blood

k=0;

for(r=0;r<4;r++)
{
  for(j=0;j<2;j++)
  {
    //the interface grid point between level 4 and level 5
  }
}
\begin{aligned}
Tb_{5, n1}[inde][ly5/2+k*(ly5+1)] &= Tb_{4, n1}[inde][j*lx4++*(lx4+1)];
\end{aligned}

//left part

for(i=ly5/2-1;i>=0;i--)
{
\begin{aligned}
fk1 &= factor5[inde]*(Tv_{5, n1}[inde][i+1+k*(ly5+1)]
- Tb_{5, n1}[inde][i+1+k*(ly5+1)]);\\
fk2 &= factor5[inde]*(Tv_{5, n1}[inde][i+1+k*(ly5+1)]
- (Tb_{5, n1}[inde][i+1+k*(ly5+1)]+fk1/2.0));\\
fk3 &= factor5[inde]*(Tv_{5, n1}[inde][i+1+k*(ly5+1)]
- (Tb_{5, n1}[inde][i+1+k*(ly5+1)]+fk2/2.0));\\
fk4 &= factor5[inde]*(Tv_{5, n1}[inde][i+1+k*(ly5+1)]
- (Tb_{5, n1}[inde][i+1+k*(ly5+1)]+fk3));\\
\end{aligned}

\begin{aligned}
(Tb_{5, n1}[inde][i+1+k*(ly5+1)] &= Tb_{5, n1}[inde][i+1+k*(ly5+1)]
+ (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;\\
\end{aligned}

//right part

for(i=ly5/2+1;i<=ly5;i++)
{
\begin{aligned}
fk1 &= factor5[inde]*(Tv_{5, n1}[inde][i-
1+k*(ly5+1)]
- Tb_{5, n1}[inde][i-1+k*(ly5+1)]);\\
fk2 &= factor5[inde]*(Tv_{5, n1}[inde][i-
1+k*(ly5+1)]
- (Tb_{5, n1}[inde][i-1+k*(ly5+1)]+fk1/2.0));\\
fk3 &= factor5[inde]*(Tv_{5, n1}[inde][i-
1+k*(ly5+1)]+fk2/2.0));\\
\end{aligned}

\begin{aligned}
(Tb_{5, n1}[inde][i-
1+k*(ly5+1)]+fk2/2.0));\\
\end{aligned}

\begin{aligned}
fk4 &= factor5[inde]*(Tv_{5, n1}[inde][i-
1+k*(ly5+1)]
- (Tb_{5, n1}[inde][i-1+k*(ly5+1)]+fk3));\\
\end{aligned}

\begin{aligned}
(Tb_{5, n1}[inde][i+k*(ly5+1)] &= Tb_{5, n1}[inde][i-1+k*(ly5+1)]
+ (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;\\
\end{aligned}

k++;

} //sixth level blood

k=0;

for(r=0;r<8;r++)
{
\begin{aligned}
for(j=0;j<2j++;)
\end{aligned}

\begin{aligned}
//the interface grid point between level 5 and level 6
Tb_{6, n1}[inde][lx6/2+k*(lx6+1)] &= Tb_{5, n1}[inde][j*ly5++*(ly5+1)];\\
\end{aligned}

//left part

for(i=lx6/2-1;i>=0;i--)
{
\begin{aligned}
fk1 &= factor6[inde]*(Tv_{6, n1}[inde][i+1+k*(lx6+1)]
- Tb_{6, n1}[inde][i+1+k*(lx6+1)]);\\
fk2 &= factor6[inde]*(Tv_{6, n1}[inde][i+1+k*(lx6+1)]
- (Tb_{6, n1}[inde][i+1+k*(lx6+1)]+fk1/2.0));\\
fk3 &= factor6[inde]*(Tv_{6, n1}[inde][i+1+k*(lx6+1)]
- (Tb_{6, n1}[inde][i+1+k*(lx6+1)]+fk2/2.0));\\
\end{aligned}

\[ \text{fk4} = \text{factor6}[\text{inde}] \ast (T_{v6 \_n1}[\text{inde}][i+1+k* (lz6+1)]) - \]
\[ (T_{b6 \_n1}[\text{inde}][i+i+k* (lz6+1)] + \text{fk3}); \]
\[ T_{b6 \_n1}[\text{inde}][i+k* (lz6+1)] = T_{b6 \_n1}[\text{inde}][i+i+k* (lz6+1)] + \]
\[ (\text{fk1} + 2.0*\text{fk2} + 2.0*\text{fk3} + \text{fk4})/6.0; \]
\[ \}
\[ \] //right part
\[ \] for(i=lz6/2+1;i<=lz6;i++)
\[ \{ \]
\[ \text{fk1} = \text{factor6}[\text{inde}] \ast (T_{v6 \_n1}[\text{inde}][i-\]
\[ 1+k* (lz6+1)]) - T_{b6 \_n1}[\text{inde}][i-1+k* (lz6+1)]; \]
\[ \text{fk2} = \text{factor6}[\text{inde}] \ast (T_{v6 \_n1}[\text{inde}][i-\]
\[ 1+k* (lz6+1)]) - (T_{b6 \_n1}[\text{inde}][i-1+k* (lz6+1)]+\text{fk2}/2.0); \]
\[ \text{fk3} = \text{factor6}[\text{inde}] \ast (T_{v6 \_n1}[\text{inde}][i-\]
\[ 1+k* (lz6+1)]) - (T_{b6 \_n1}[\text{inde}][i-1+k* (lz6+1)]+\text{fk2}/2.0); \]
\[ \text{fk4} = \text{factor6}[\text{inde}] \ast (T_{v6 \_n1}[\text{inde}][i-\]
\[ 1+k* (lz6+1)]) - (T_{b6 \_n1}[\text{inde}][i-1+k* (lz6+1)]+\text{fk3}); \]
\[ T_{b6 \_n1}[\text{inde}][i+k* (lz6+1)] = T_{b6 \_n1}[\text{inde}][i-1+k* (lz6+1)] + \]
\[ (\text{fk1} + 2.0*\text{fk2} + 2.0*\text{fk3} + \text{fk4})/6.0; \]
\[ \}
\[ \] k++;
\[ ] //seventh level blood
k=0;
\[ \] for(r=0;r<16;r++)
\[ \]
\[ Tb7_n_l[index][i+k*(lx7+1)] = Tb7_n_l[index][i+1+k*(lx7+1)] + (fkl + 2.0*fk2 + 2.0*fk3 + fk4)/6.0; \]

//right part
for(i=lx7/2+1;i<=lx7;i++) {
    fkl = factor7[index]*(Tv7_n_l[index][i-1+k*(lx7+1)])
    + deltaZ*F7[index]
    *Pdot*(Tb7_n_l[index][i-l+k*(lx7+1)])/M7[index];
    fk2 = factor7[index]*(Tv7_n_l[index][i-1+k*(lx7+1)]
    -(Tb7_n_l[index][i-l+k*(lx7+1)]+fk1/2.0))
    + deltaZ*F7[index]*Pdot*(Tb7_n_l[index][i-1+1+k*(lx7+1)])
    +fk1/2.0)/M7[index];
    fk3 = factor7[index]*(Tv7_n_l[index][i-1+k*(lx7+1)]
    -(Tb7_n_l[index][i-l+k*(lx7+1)]+fk2/2.0))
    + deltaZ*F7[index]*Pdot*(Tb7_n_l[index][i-1+1+k*(lx7+1)])
    +fk2/2.0)/M7[index];
    fk4 = factor7[index]*(Tv7_n_l[index][i-1+k*(lx7+1)]
    -(Tb7_n_l[index][i-l+k*(lx7+1)]+fk3))
    + deltaZ*F7[index]*Pdot*(Tb7_n_l[index][i-1+k*(lx7+1)]+fk3)/M7[index];
    Tb7_n_l[index][i+k*(lx7+1)] = Tb7_n_l[index][i+1+k*(lx7+1)]
    + (fkl + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
    k++;
}
}

return(1);

******************************************************************************

SUBROUTINE

******************************************************************************

int CalcTb2() // Calculate the blood temperature of vein
{
    //int index = 1;
    //double fkl, fk2, fk3, fk4;
    index = 1;
    setborderVariable(index);//set the common blood border variables

    //seventh level blood
    n=0;
    k=0;
    for(r=0;r<2;r++)//repeat variable in z coordinate
    {
        z2=cenZ+(2*r-l)*lz3/2+(2*r-l)*lz4/2;
        for(j=0;j<2;j++)//repeat variable in y coordinate
        {
            for(p=0;p<2;p++)//repeat variable in x coordinate
            {
                x2=cenX+(2*p-l)*lx4/2+(2*p-l)*lx5/2lx7/2;
                for(q=0;q<2;q++)//repeat variable in inner y coordinate
                {

                }
            }
        }
    }
}
\[ y_2 = \text{cenY} + (2^j - 1) \cdot y_3/2 + (2^j - 1) \cdot y_5/2 + (2^j - 1) \cdot y_6/2; \]

\[
\text{for } h=0; h<2; h++ \} / \text{repeat variable in inner z coordinate} \\
\{ \\
// \text{left part} \\
\text{Tb}_7_{n1}[\text{inde}][0+k*(lx7+1)] = \\
\text{Tt}_{n1}[x2-2][y2][z2+(2*h-1)*lz6/2] \\
+ (2*h-1)*lz7/2]; //entry point \\
\text{for } (i=1; i<=lx7/2; i++) \\
\{ \\
fkl = \\
factor7[\text{inde}]*(\text{Tv7}_n1[\text{inde}][i-1+k*(lx7+1)] \\
- \text{Tb}_7_{n1}[\text{inde}][i-1+k*(lx7+1)]); \\
fk2 = \\
factor7[\text{inde}]*(\text{Tv7}_n1[\text{inde}][i-1+k*(lx7+1)] \\
- \text{Tb}_7_{n1}[\text{inde}][i-1+k*(lx7+1)]+fk1/2.0); \\
fk3 = \\
factor7[\text{inde}]*(\text{Tv7}_n1[\text{inde}][i-1+k*(lx7+1)] \\
- \text{Tb}_7_{n1}[\text{inde}][i-1+k*(lx7+1)]+fk2/2.0); \\
fk4 = \\
factor7[\text{inde}]*(\text{Tv7}_n1[\text{inde}][i-1+k*(lx7+1)] \\
- \text{Tb}_7_{n1}[\text{inde}][i-1+k*(lx7+1)]+fk3); \\
\text{Tb}_7_{n1}[\text{inde}][i+k*(lx7+1)] = \\
\text{Tb}_7_{n1}[\text{inde}][i-1+k*(lx7+1)] + \\
(fk1 + 2.0*fk2 + 2.0*fk3 + \\
fk4)/6.0; \\
\} \\
// \text{store the middle temperature} \\
(\text{intersection of blood level 6 \\& 7}) \\
\text{Tb}_6_{n1}[\text{inde}][h*lg6+n*(lg6+1)] = \\
\text{Tb}_7_{n1}[\text{inde}][lx7/2+k*(lx7+1)]; \\
// \text{right part} \\
\text{Tb}_7_{n1}[\text{inde}][lx7+k*(lx7+1)] = \\
\text{Tt}_{n1}[x3+2][y2][z2+(2*h-1)*lz6/2] \\
* lz6/2 + (2*h-1)*lz7/2]; //entry point \\
\text{for } (i=lx7-1; i>=lx7/2; i--) \\
\{ \\
fk1 = \\
factor7[\text{inde}]*(\text{Tv7}_n1[\text{inde}][i+1+k*(lx7+1)] \\
- \text{Tb}_7_{n1}[\text{inde}][i+1+k*(lx7+1)]+fk1/2.0)); \\
fk2 = \\
factor7[\text{inde}]*(\text{Tv7}_n1[\text{inde}][i+1+k*(lx7+1)] \\
- \text{Tb}_7_{n1}[\text{inde}][i+1+k*(lx7+1)]+fk2/2.0)); \\
fk3 = \\
factor7[\text{inde}]*(\text{Tv7}_n1[\text{inde}][i+1+k*(lx7+1)] \\
- \text{Tb}_7_{n1}[\text{inde}][i+1+k*(lx7+1)]+fk3)); \\
\text{Tb}_7_{n1}[\text{inde}][i+k*(lx7+1)] = \\
\text{Tb}_7_{n1}[\text{inde}][i+1+k*(lx7+1)] + \\
(fk1 + 2.0*fk2 + 2.0*fk3 + \\
fk4)/6.0; \\
\} \\
// \text{take the average of calculation of left} \\
\text{and right sides} \\
\text{Tb}_6_{n1}[\text{inde}][h*lg6+n*(lg6+1)] = \\
(\text{Tb}_6_{n1}[\text{inde}][h*lg6+n*(lg6+1)] + \\
\text{Tb}_7_{n1}[\text{inde}][lx7/2+k*(lx7+1)]+2.0)); \\
\text{Tb}_7_{n1}[\text{inde}][lx7+k*(lx7+1)] = \\
\text{Tb}_6_{n1}[\text{inde}][h*lg6+n*(lg6+1)]; \\
k++; \\
// h loop end \\
n++;
//q loop end
}  //p loop end
}  //j loop end
}  //r loop end

//sixth level blood
k=0;
for(r=0;r<8;r++)
{
    for(j=0;j<2;j++)
    {
//upper part
        for(i=l;i<=lz6/2;i++)
        {
            fkl = factor6[ind][i*(Tv6[nl][inde][i-1+k*(lz6+1)]
- Tb6[nl][inde][i-1+k*(lz6+1)])];
            fk2 = factor6[ind][i*(Tv6[nl][inde][i-1+k*(lz6+1)]
- (Tb6[nl][inde][i-1+k*(lz6+1)]+fk1/2.0))];
            fk3 = factor6[ind][i*(Tv6[nl][inde][i-1+k*(lz6+1)]
- (Tb6[nl][inde][i-1+k*(lz6+1)]+fk2/2.0))];
            fk4 = factor6[ind][i*(Tv6[nl][inde][i-1+k*(lz6+1)]
- (Tb6[nl][inde][i-1+k*(lz6+1)]+fk3)];
            Tb6[nl][inde][i+k*(lz6+1)] = Tb6[nl][inde][i-1+k*(lz6+1)]
+ (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
        }
//store the middle temperature (intersection of blood level 5 & 6)
    Tb5[nl][inde][*ly5+r*(ly5+1)] = Tb6[nl][inde][lz6/2+k*(lz6+1)];
//lower part
    Tb6[nl][inde][lz6+k*(lz6+1)] = Tb7[nl][inde][lx7/2+(2*k+1)*(lx7+1)];
//the interface grid point between level 6 and level 7
    for(i=lz6-1;i>=lz6/2;i--)
    {
        fk1 = factor6[ind][i*(Tv6[nl][inde][i+1+k*(lz6+1)]
- Tb6[nl][inde][i+1+k*(lz6+1)])];
        fk2 = factor6[ind][i*(Tv6[nl][inde][i+1+k*(lz6+1)]
- (Tb6[nl][inde][i+1+k*(lz6+1)]+fk1/2.0))];
        fk3 = factor6[ind][i*(Tv6[nl][inde][i+1+k*(lz6+1)]
- (Tb6[nl][inde][i+1+k*(lz6+1)]+fk2/2.0))];
        fk4 = factor6[ind][i*(Tv6[nl][inde][i+1+k*(lz6+1)]
- (Tb6[nl][inde][i+1+k*(lz6+1)]+fk3)];
        Tb6[nl][inde][i+k*(lz6+1)] = Tb6[nl][inde][i+1+k*(lz6+1)]
+ (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
    }
//take the average of calculation of left and right sides
    Tb5[nl][inde][*ly5+r*(ly5+1)] = (Tb5[nl][inde][*ly5+r*(ly5+1)]
+ Tb6[nl][inde][lz6/2+k*(lz6+1)])/2.0;
//fifth level blood
k=0;
for(r=0;r<4;r++)
{
  for(j=0;j<2;j++)
  {
    //left part
    for(i=l;i<=ly5/2;i++)
    {
      fkl = factor5[index]*(Tv5_nl[index][i-l+k*(ly5+l)] - Tb5_nl[index][i-l+k*(ly5+l)]);
      fk2 = factor5[index]*(Tv5_nl[index][i-l+k*(ly5+l)] - (Tb5_nl[index][i-l+k*(ly5+l)]+fkl/2.0));
      fk3 = factor5[index]*(Tv5_nl[index][i-l+k*(ly5+l)] - (Tb5_nl[index][i-l+k*(ly5+l)]+fk2/2.0));
      fk4 = factor5[index]*(Tv5_nl[index][i-l+k*(ly5+l)] - (Tb5_nl[index][i-l+k*(ly5+l)]+fk3));
      Tb5_nl[index][i+k*(ly5+l)] = (Tb5_nl[index][i+k*(ly5+l)]+(fkl + 2.0*fk2 + 2.0*fk3 + fk4))/6.0;
    }
    //store the middle temperature (intersection of blood level 4 & 5)
    Tb4_nl[index][j*lx4+r*(lx4+l)] = Tb5_nl[index][ly5/2+k*(ly5+l)];
  }
  //right part
  for(i=ly5-1;i>=ly5/2;i--)
  {
    fk1 = factor5[index]*(Tv5_nl[index][i+1+k*(ly5+l)] - Tb5_nl[index][i+1+k*(ly5+l)]);
    fk2 = factor5[index]*(Tv5_nl[index][i+1+k*(ly5+l)]+fk1/2.0));
    fk3 = factor5[index]*(Tv5_nl[index][i+1+k*(ly5+l)]+fk2/2.0));
    fk4 = factor5[index]*(Tv5_nl[index][i+1+k*(ly5+l)]+fk3));
    Tb5_nl[index][i+k*(ly5+l)] = (Tb5_nl[index][i+k*(ly5+l)]+(fk1 + 2.0*fk2 + 2.0*fk3 +
      fk4))/6.0;
  }
  //take the average of calculation of left and right sides
  Tb4_nl[index][j*lx4+r*(lx4+l)] = (Tb4_nl[index][j*lx4+r*(lx4+l)] +
    +Tb5_nl[index][ly5/2+k*(ly5+l)])/2.0;
  Tb5_nl[index][ly5/2+k*(ly5+l)] = Tb4_nl[index][j*lx4+r*(lx4+l)];
  k++;
}
//fourth level blood
\( k=0; \)

\[ \text{for} (r=0; r<2; r++) \]
\[
\text{for} (i=0; i<lx4/2; i++) \]
\[
\]

//left part
\[
\text{for} (i=1; i<l; i++) \]
\[
\]

\( fk1 = \text{factor4[index]} \times (Tv4_n1[index][i-1+k*(lx4+1)] - Tb4_n1[index][i-1+k*(lx4+1)]); \)

\( fk2 = \text{factor4[index]} \times (Tv4_n1[index][i-1+k*(lx4+1)] - (Tb4_n1[index][i-1+k*(lx4+1)] + fk1/2.0)); \)

\( fk3 = \text{factor4[index]} \times (Tv4_n1[index][i-1+k*(lx4+1)] - (Tb4_n1[index][i-1+k*(lx4+1)] + fk2/2.0)); \)

\( fk4 = \text{factor4[index]} \times (Tv4_n1[index][i-1+k*(lx4+1)] - (Tb4_n1[index][i-1+k*(lx4+1)] + fk3)); \)

\( Tb4_n1[index][i+k*(lx4+1)] = Tb4_n1[index][i+k*(lx4+1)] + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0; \)

\]

//store the middle temperature (intersection of blood level 4 & 3)

//the order between 3 and 4 are exception to others

\( Tb3_n1[index][r*lx3+j*(lx3+1)] = Tb4_n1[index][lx4/2+k*(lx4+1)]; \)

//right part
\[
\text{for} (i=lx4-1; i>=lx4/2; i--) \]
\[
\]
$f_k1 = \text{factor3[inde]} \cdot (Tv3_n1[inde][i-1+j*(l3z+1)]) - Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$

$Tb3_n1[inde][i-1+j*(l3z+1)];$
```c
{ 

    fk1 = factor2[index]*(Tv2_nl[index][i+1].
        Tb2_nl[index][i+1]);

    fk2 = factor2[index]*(Tv2_nl[index][i+1].
        (Tb2_nl[index][i]+fk1/2.0));

    fk3 = factor2[index]*(Tv2_nl[index][i+1].
        (Tb2_nl[index][i+1]+fk2/2.0));

    fk4 = factor2[index]*(Tv2_nl[index][i+1].
        (Tb2_nl[index][i+1]+fk3));

    Tb2_nl[index][i] = Tb2_nl[index][i+1] +
        (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
}

//take the average of calculation of left and right
sides
Tb1_nl[index][ix1] = (Tb1_nl[index][ix1] +
    Tb2_nl[index][iy2/2])/2.0;
Tb2_nl[index][iy2/2] = Tb1_nl[index][ix1];
//First level blood
for(i=ix1-1; i>=0; i--)
{
    fk1 = factor1[index]*(Tv1_nl[index][i+1].
        Tb1_nl[index][i+1]);

    fk2 = factor1[index]*(Tv1_nl[index][i+1].
        (Tb1_nl[index][i]+fk1/2.0));

    fk3 = factor1[index]*(Tv1_nl[index][i+1].
        (Tb1_nl[index][i]+fk2/2.0));

    fk4 = factor1[index]*(Tv1_nl[index][i+1].
        (Tb1_nl[index][i]+fk3));

    Tb1_nl[index][i] = Tb1_nl[index][i+1] +
        (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
}
return(1);
}

*************************************************************************
END OF CALCTB2 SUBROUTINE
*************************************************************************
*************************************************************************
BEGIN CALCTT SUBROUTINE (CALCULATES ONLY THE TISSUE TEMPERATURES USING PRE-

CONDITIONED RICHARDSON ITERATION WITH THOMAS ALGORITHM)*************************************************************************
*************************************************************************
double CalCTt() //Calculate the tissue temperature
{
    //initialize tridiagonal system
    memcpy(a, a0, sizeof(double)*(NX+1)*(NY+1)*(NZ+1));
    memcpy(b, b0, sizeof(double)*(NX+1)*(NY+1)*(NZ+1));
    memcpy(c, c0, sizeof(double)*(NX+1)*(NY+1)*(NZ+1));
    memset(d, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ+1));

    //initialize tri-diagonal system
    for(i=1; i<=NX-1; i++)
    {
        for(j=1; j<=NY-1; j++)
        {
            for(z=1; z<=NZ-1; z++)
            {
                //Laser case with CVN and Tumor: Casey's Method
                f =
                    (2.0*p1*C1*(1.0+tau*Wb1*Cbl/(p1*C1)+2.0*tau/de
                        ltaT) + Wb1*Cbl*deltaT)*Tt_nl_I[i][j][z] +
                    (2.0*p1*C1*(1.0+tau*Wb1*Cbl/(p1*C1))
                        -2.0*tau(deltaT)*Wb1*Cbl*deltaT)*Tt[i][j][z] -
                        4.0*p1*C1*U[i][j][z]
                - Wb1*Cbl*deltaT*(Tt[i][j][z])
                - 2.0*Wb1*Cbl*deltaT*(Tb7_n1[l][1][X7A])
                    - k1*deltaT*(Tt[i][j][z])
                - 2.0*Wb1*Cbl*deltaT*(Tb7_n1[l][1][1][1])
                    - k1*deltaT*(Tt[i][j][z])
                - 2.0*Wb1*Cbl*deltaT*(Tb7_n1[l][1][1][1])
                    - k1*deltaT*(Tt[i][j][z])
                - 2.0*Wb1*Cbl*deltaT*(Tb7_n1[l][1][1][1])
                    - k1*deltaT*(Tt[i][j][z])
            }
        }
    }
    return(1);
}
```
\[2.0 \cdot T_{t_{nl,l}}(i,j,z)/(\delta x \cdot \delta x) + T_{t_{nl,l}}(i,j+1,z) \]
\[2.0 \cdot T_{t_{nl,l}}(i,j,z)/(\delta y \cdot \delta y) + T_{t_{nl,l}}(i,j+1,z) \]
\[-2.0 \cdot T_{t_{nl,l}}(i,j,z)/(\delta z \cdot \delta z) \]
\[-k_{1} \cdot \delta T \cdot (T_{t_{nl,l}}(i,j,z) + T_{t_{nl,l}}(i,j+1,z) - 2.0 \cdot T_{t_{nl,l}}(i,j,z)/(\delta x \cdot \delta x) + (T_{t_{nl,l}}(i,j-1,z) + T_{t_{nl,l}}(i,j+1,z) - 2.0 \cdot T_{t_{nl,l}}(i,j,z)/(\delta y \cdot \delta y) + (T_{t_{nl,l}}(i,j,z-1) + T_{t_{nl,l}}(i,j,z+1) - 2.0 \cdot T_{t_{nl,l}}(i,j,z)/(\delta z \cdot \delta z)) - 2.0 \cdot \delta T \cdot Q_{1}(i,j,z); \]
\[d_{i,j}(z) = (2.0 \cdot p_{1} \cdot C_{1} \cdot (1.0 + \tau \cdot \omega_{b_{1}} \cdot C_{1} / (p_{1} \cdot C_{1}) + 2.0 \cdot \tau / \delta T) + \omega_{b_{1}} \cdot C_{1} \cdot \delta T) \cdot T_{t_{nl,l}}(i,j,z) - k_{1} \cdot \delta T \cdot (T_{t_{nl,l}}(i,j,z-1) + T_{t_{nl,l}}(i,j,z+1)) - \omega_{b_{1}} \cdot f; \]
\[d_{i,j}(z) = (2.0 \cdot p_{2} \cdot C_{2} \cdot (1.0 + \tau \cdot \omega_{b_{2}} \cdot C_{2} / (p_{2} \cdot C_{2}) + 2.0 \cdot \tau / \delta T) + \omega_{b_{2}} \cdot C_{2} \cdot \delta T) \cdot T_{t_{nl,l}}(i,j,z) - k_{2} \cdot \delta T \cdot (T_{t_{nl,l}}(i,j,z-1) + T_{t_{nl,l}}(i,j,z+1)) - \omega_{2} \cdot f; \]
\[d_{i,j}(z) = (2.0 \cdot p_{2} \cdot C_{2} \cdot (1.0 + \tau \cdot \omega_{b_{2}} \cdot C_{2} / (p_{2} \cdot C_{2}) + 2.0 \cdot \tau / \delta T) + \omega_{b_{2}} \cdot C_{2} \cdot \delta T) \cdot T_{t_{nl,l}}(i,j,z) - k_{2} \cdot \delta T \cdot (T_{t_{nl,l}}(i,j,z-1) + T_{t_{nl,l}}(i,j,z+1)) - \omega_{2} \cdot f; \]
\[d_{i,j}(z) = 0; \]
\[d_{i,j}(z) = 0; \]
\[d_{i,j}(z) = 0; \]
\[d_{i,j}(z) = 0; \]
\[d_{i,j}(z) = 0; \]
//On Bottom Tumor boundary with subcutaneous region
else if (z > TT && z < TT + LT && i > RTX && i < RTY) {
    if ((i < centerX + deltaPX) && (i > centerX - deltaPX) &&
        (j < centerY + deltaPY) && (j > centerY - deltaPY) &&
        (z < centerTZ + deltaPZ) && (z > centerTZ - deltaPZ)) {
        f = (2.0 * Ceff * (1.0 + tau * WbTumor * Cb3 / Ceff) + 2.0 * tau / deltaT)
            + WbTumor * Cb3 * deltaT * Tt_n1[i][j][z] +
            (2.0 * Ceff * (1.0 + tau / deltaT) + WbTumor * Cb3 * deltaT)
            * (Tb7_n1[0][0][LX7A]) - Keff * deltaT
            * ((T_n1[i-1][j][z] + T_n1[i+1][j][z])
                - 4.0 * Keff * deltaT)
            * (Tt_n1[i][j][z-1] + Tt_n1[i][j][z+1]) - omega * f;
        } //Inside Tumor region with nanoparticles
else if ((i == centerX - deltaPX) && (j > centerY + deltaPY) &&
        (j < centerY - deltaPY) && (z > centerTZ + deltaPZ) &&
        (z > centerTZ - deltaPZ)) {
        d[i][j][z] = 0.0;
    } //Left Side of Nanoparticle-Tumor region in X direction
else if ((i == centerX + deltaPX) && (j > centerY + deltaPY) &&
        (j < centerY - deltaPY) && (z > centerTZ + deltaPZ) &&
        (z > centerTZ - deltaPZ)) {
        d[i][j][z] = 0.0;
    } //Right side of Nanoparticle-Tumor region in X direction
else if ((j == centerY - deltaPY) && (i > centerX + deltaPX) &&
        (i < centerX - deltaPX) && (z > centerTZ + deltaPZ) &&
        (z > centerTZ - deltaPZ)) {
        d[i][j][z] = 0.0;
    } //Left Side of Nanoparticle-Tumor region in Y direction
else if ((j == centerY + deltaPY) && (i > centerX + deltaPX) &&
        (i < centerX - deltaPX) && (z > centerTZ + deltaPZ) &&
        (z > centerTZ - deltaPZ)) {
        d[i][j][z] = 0.0;
    } //Right side of Nanoparticle-Tumor region in Y direction
else if ((z == centerTZ - deltaPZ) && (i <= centerX + deltaPX) &&
        (i >= centerX - deltaPX) && (j <= centerY + deltaPY) &&
        (j >= centerY - deltaPY) ){
        d[i][j][z] = 0.0;
    } //Top of nanoparticle embedded region
else if ((z == centerTZ + deltaPZ) && (i <= centerX + deltaPX) &&
        (i >= centerX - deltaPX) && (j <= centerY + deltaPY) &&
        (j >= centerY - deltaPY) ){
        d[i][j][z] = 0.0;
//Bottom of nanoparticle embedded region

else {
    f = (
        (2.0*pTumor*CTumor*(1.0+tau*WbTumor*Cb3/(pTumor*CTumor))+2.0*tau/deltaT)
        +WbTumor*Cb3*deltaT)*Tt_n1[i][j][z] +
        (2.0*pTumor*CTumor*(1.0+tau*WbTumor*Cb3/(pTumor*CTumor))
        -2.0*WbTumor*Cb3*deltaT)*Tt[i][j][z]
    ) - kTumor*deltaT
    * (Tt_n1[i-1][j][z]+Tt_n1[i+1][j][z]
    - 2.0*Tt_n1[i][j][z])/(deltaX*deltaX)+(Tt_n1[i][j-1][z]+Tt_n1[i][j+1][z]
    - 2.0*Tt_n1[i][j][z])/(deltaY*deltaY)+(Tt_n1[i][j][z-1]+Tt_n1[i][j][z+1]
    - 2.0*Tt_n1[i][j][z])/(deltaZ*deltaZ)
} //Inside Tumor Region

else if(i == LTX && z>=TT && z<=TT+LT &&
    j<LTY && j<=RTY) {
    d[i][j][z] = 0.0;
} //Tumor Boundary at NX/3

else if(i == LTY && z>=TT && z<=TT+LT &&
    j>=LTX && j<=RTX) {
    d[i][j][z] = 0.0;
} //Tumor Boundary at NY/3

else if(i == RTX && z>=TT && z<=TT+LT &&
    j>=LTX && j<=RTX) {
    d[i][j][z] = 0.0;
} //Tumor Boundary at 2*NX/3

else if(i == RTY && z>=TT && z<=TT+LT &&
    j>=LTX && j<=RTX) {
    d[i][j][z] = 0.0;
} //Tumor Boundary at 2*NY/3

else {
    f = (2.0*p3*C3*(1.0+tau*Wb3*Cb3/(p3*C3))+2.0*tau/deltaT)
        +Wb3*Cb3*deltaT)*Tt_n1[i][j][z] +
        (2.0*p3*C3*(1.0+tau*Wb3*Cb3/(p3*C3))
        -2.0*tau/deltaT)*Tt[i][j][z]
        +Wb3*Cb3*deltaT)*Tb7_n1[i][LX7A])
    - k3*deltaT
    * (Tt_n1[i-1][j][z]+Tt_n1[i+1][j][z]
    - 2.0*Tt_n1[i][j][z])/(deltaX*deltaX)+(Tt_n1[i][j-1][z]+Tt_n1[i][j+1][z]
    - 2.0*Tt_n1[i][j][z])/(deltaY*deltaY)+(Tt_n1[i][j][z-1]+Tt_n1[i][j][z+1]
    - 2.0*Tt_n1[i][j][z])/(deltaZ*deltaZ)
} //Inside Tumor Region

else if(i == LTX && z>=TT && z<=TT+LT &&
    j<LTY && j<=RTY) {
    d[i][j][z] = 0.0;
} //Tumor Boundary at NX/3

else if(i == LTY && z>=TT && z<=TT+LT &&
    j>=LTX && j<=RTX) {
    d[i][j][z] = 0.0;
} //Tumor Boundary at NY/3

else if(i == RTX && z>=TT && z<=TT+LT &&
    j>=LTX && j<=RTX) {
    d[i][j][z] = 0.0;
} //Tumor Boundary at 2*NX/3

else if(i == RTY && z>=TT && z<=TT+LT &&
    j>=LTX && j<=RTX) {
    d[i][j][z] = 0.0;
} //Tumor Boundary at 2*NY/3
+ (Tt[i][j][z] - Tt[i][j][z-1]) - 2.0*Tt[i][j][z])/(deltaY*deltaY) + (Tt[i][j][z] - Tt[i][j][z-1]) - 2.0*Tt[i][j][z]);

d[i][j][z] = (2.0*p3*C3*(1.0+tau*Wb3*Cb3/(p3*C3)+2.0*tau/deltaT) + Wb3*Cb3*deltaT + (4.0*k3*deltaT) * (1.0/(deltaX*deltaX)+1.0/(deltaY*deltaY)) + (2.0*k3*deltaT)/(deltaZ*deltaZ) * Tt_nl[I][i][j][z]) - k3*deltaT/(deltaZ*deltaZ) * (Tt_nl[I][i][j][z-1]+Tt_nl[I][i][j][z+1]) - omega * f;

} // In Subcutaneous Region, surrounding Tumor

} //for NZ2 to NZ3

} //j

} //i

double tE[NZ3], tf[NZ3];

//solve the tria-diagonial system

for(i=1;i<=NX-1;++i)
for(j=1;j<=NY-1;++j)
{

} // With convection on the surface

tF[1] = (d[i][j][1] - b[i][j][1]*Hf*deltaZ*TF(k+deltaZ*Hf)) / (b[i][j][1]*k/(k+deltaZ*Hf) + a[i][j][1]);

tF[z] = (d[i][j][z] - b[i][j][z]*tf[z-1]) / (a[i][j][z] + b[i][j][z]*tf[z-1]);

tF[z] = (d[i][j][z] - a[i][j][z]*tf[z-1]) / (a[i][j][z] + b[i][j][z]*tf[z-1]);

Tt_nl[I][i][j][z] = tF[NZ3-1];

} // Update Faces of Tumor here with n+1 time step:

} //Vertical and Horizontal Faces

for(i=0;i<=NX-1;++i)
for(j=0;j<=NY-1;++j)
{
for(z=NZ2;z<=NZ3-1;++z)
{
if (i == LTX && j >= TT && j <= RTY)
{Tt_nl[I][i][j][z] = (kTumor*Tt_nl[I][i][j][z] + k3*Tt_nl[I][i][j][z])/(kTumor + k3);
}

} //Tumor Boundary at NX/3

} //else if j == LTY && j >= TT && j <= RTY

for(i=LTX;i<=RTX;i++)
for(j=LTY;j<=RTY;j++)
for(z=NZ2;z<=NZ3-1;++z)
{

} //On top tumor boundary with subcutaneous region

else if (i >= LTX && i <= RTX && j >= LTY && j <= RTY)
{Tt_nl[I][i][j][z] = (kTumor*Tt_nl[I][i][j][z] + k3*Tt_nl[I][i][j][z])/(kTumor + k3);
}

} //Tumor Boundary at 2*NY/3

} //else if j == RTY && j >= TT && j <= RTY

for(i=LTX;i<=RTX;i++)
for(j=LTY;j<=RTY;j++)
for(z=TT+LT;z<=NZ3-1;++z)
{

} //On top tumor boundary with subcutaneous region

else if (i >= LTX && i <= RTX && j >= LTY && j <= RTY)
{Tt_nl[I][i][j][z] = (kTumor*Tt_nl[I][i][j][z] + k3*Tt_nl[I][i][j][z])/(kTumor + k3);
}

} //Tumor Boundary at 2*NX/3

} //else if j == RTY && j >= TT && j <= RTY

for(i=LTX;i<=RTX;i++)
for(j=LTY;j<=RTY;j++)
for(z=TT+LT;z<=NZ3-1;++z)
{

}
\[
T_{t_{nl}}[i][j][z] = (k3 \times T_{t_{nl}}[i][j][z+1] + k_{\text{Tumor}} \times T_{t_{nl}}[i][j][z-1])/(k_{\text{Tumor}} + k3);
\]

} //On Tumor boundary with subcutaneous region

else if (i==centerX - deltaPX) && (j < centerY + deltaPY) && (j < centerY - deltaPY) && (z < centerTZ + deltaPZ) && (z > centerTZ - deltaPZ)) {

\[
T_{t_{nl}}[i][j][z] = (K_{\text{Eff}} \times T_{t_{nl}}[i+1][j][z] + k_{\text{Tumor}} \times T_{t_{nl}}[i-1][j][z])/(k_{\text{Tumor}} + K_{\text{Eff}});
\]

} //Left Side of Nanoparticle-Tumor region in X direction

else if (i==centerX + deltaPX) && (j < centerY + deltaPY) && (j > centerY - deltaPY) && (z < centerTZ + deltaPZ) && (z > centerTZ - deltaPZ)) {

\[
T_{t_{nl}}[i][j][z] = (K_{\text{Eff}} \times T_{t_{nl}}[i-1][j][z] + k_{\text{Tumor}} \times T_{t_{nl}}[i+1][j][z])/(k_{\text{Tumor}} + K_{\text{Eff}});
\]

} //Right side of Nanoparticle-Tumor region in X direction

else if (j==centerY - deltaPY) && (i < centerX + deltaPX) && (i > centerX - deltaPX) && (z < centerTZ + deltaPZ) && (z > centerTZ - deltaPZ)) {

\[
T_{t_{nl}}[i][j][z] = (K_{\text{Eff}} \times T_{t_{nl}}[i][j+1][z] + k_{\text{Tumor}} \times T_{t_{nl}}[i][j-1][z])/(k_{\text{Tumor}} + K_{\text{Eff}});
\]

} //Left Side of Nanoparticle-Tumor region in Y direction

else if (j==centerY + deltaPY) && (i < centerX + deltaPX) && (i > centerX - deltaPX) && (z < centerTZ + deltaPZ) && (z > centerTZ - deltaPZ)) {

\[
T_{t_{nl}}[i][j][z] = (K_{\text{Eff}} \times T_{t_{nl}}[i][j-1][z] + k_{\text{Tumor}} \times T_{t_{nl}}[i][j+1][z])/(k_{\text{Tumor}} + K_{\text{Eff}});
\]

} //Right side of Nanoparticle-Tumor region in Y direction

else if (z==centerTZ - deltaPZ) && (i <= centerX + deltaPX) && (i >= centerX - deltaPX) && (j <= centerY + deltaPY) && (j >= centerY - deltaPY)) {

\[
T_{t_{nl}}[i][j][z] = (K_{\text{Eff}} \times T_{t_{nl}}[i][j][z+1] + k_{\text{Tumor}} \times T_{t_{nl}}[i][j][z-1])/(k_{\text{Tumor}} + K_{\text{Eff}});
\]

} //Top of nanoparticle embedded region

else if (z==centerTZ + deltaPZ) && (i <= centerX + deltaPX) && (i >= centerX - deltaPX) && (j <= centerY + deltaPY) && (j >= centerY - deltaPY) {

\[
T_{t_{nl}}[i][j][z] = (K_{\text{Tumor}} \times T_{t_{nl}}[i][j][z+1] + K_{\text{Eff}} \times T_{t_{nl}}[i][j][z-1])/(k_{\text{Tumor}} + K_{\text{Eff}});
\]

} //Bottom of nanoparticle embedded region

else {

//No updates to do for vertical faces

} } } }

//assign tissue boundary grid points

for(i=0;i<=NX;i++)

for(j=0;j<=NY;j++)

{  // With heat convection on the surface (Both 2 cases can use the following codes by setting different Hf)

\[
T_{t_{nl}}[i][j][0] = T_{t_{nl}}[i][j][1] \times k1/(k1 + \text{deltaZ} \times \text{Hf}) + \\
\text{Hf} \times \text{deltaZ} / (k1 + \text{deltaZ} \times \text{Hf}) \times T_f;
\]

\[
T_{t_{nl}}[i][j][\text{NZ3}] = T_{t_{nl}}[i][j][\text{NZ3}-1]; /* */
\]

} //assign tissue boundary grid points

for(i=0;i<=NX;i++)

for(j=0;j<=NY;j++)

{  // With heat convection on the surface (Both 2 cases can use the following codes by setting different Hf)

\[
T_{t_{nl}}[0][j][z] = T_{t_{nl}}[1][j][z];
\]

\[
T_{t_{nl}}[\text{NX}][j][z] = T_{t_{nl}}[\text{NX}-1][j][z];
\]

} //assign tissue boundary grid points

for(z=0;z<=NZ3;z++)

for(i=0;i<=NX;i++)

{  // With heat convection on the surface (Both 2 cases can use the following codes by setting different Hf)

\[
T_{t_{nl}}[i][0][z] = T_{t_{nl}}[i][1][z];
\]

\[
T_{t_{nl}}[i][\text{NY}][z] = T_{t_{nl}}[i][\text{NY}-1][z];
\]

} //assign tissue boundary grid points

//adjust tissue start here

getTv_blood(0);

getTv_blood(1);
//calculate blood temperature based on given vessel
temperature
CalcTb();
CalcTb2();
reloadbloodtemperature(0);
reloadbloodtemperature(1);
CalcVessel(0);
CalcVessel(1);
//calculate sum of square error
maxErr1 = 0;
for(i=1;i<NX;i++)
    for(j=1;j<NY;j++)
        for(z=1;z<NZ; z++)
        {
            temp = fabs(Tt_nl[i][j][z] -
                        Tt_nI_I[i][j][z]);
            if(temp > maxErr1)
                maxErr1 = (double)temp;
        }
//store result to loop 1
memcpy(Tt_nI_L, Tt_nI,
sizeof(double)*(NX+1)*(NY+1)*(NZ+1));
return maxErr1;
}

***********************************************************************
RELOADBLOODTEMPERATURESUBLROUTINE
(assigns the blood temperature to locations in the tissue's
domain)***********************************************************************
void reloadbloodtemperature(int index)
{
    setborderVariable(index); //set the common
    blood boarder variables
    //reassign back the grid points in blood
    //level 7
    k=0;
    for(r=0;r<2;r++)//repeat variable in z coordinate
    {
        z3 = cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2;
        for(j=0;j<2;j++)//repeat variable in y coordinate
        {
            y3 = cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2;
            for(p=0;p<2;p++)//repeat variable in x coordinate
            {
                for(q=0;q<2;q++)//repeat variable in
                inner y coordinate
                {
                    for(h=0;h<2;h++)//repeat variable in
                    inner z coordinate
                    {
                        for(x=0;x<=lx7;x++)
                        {
                            x2 = cenX+(2*p-1)*lx4/2+(2*p-
1)*lx5/2+lx7/2+x;
                            x3 = x+k*(lx7+1);
                            for(y=0;y<ly7;y++)
                            {
                                y2 = y+q*(ly7+1)*ly5/2+(2*q-
1)*ly6/2+ly7/2+y;
                                for(z=0;z<lz7;z++)
                                {
                                    z2 = z+h*(lz7+1)*lz6/2+(2*h-
1)*lz7/2+lz7/2+z;
                                    Tt_nI[x2][y2][z2] =
Tb7_nI[index][x3];
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
k++; 
} 

//sixth level 
k=0; 
for(r=0;r<2;r++)//repeat variable in z coordinate 
{ 
z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz6/2; 
for(j=0;j<2;j++){//repeat variable in y coordinate 

y3=cenY+(2*j-l)*ly2/2+(2*j-l)*ly3/2; 
for(p=0;p<2;p++)//repeat variable in x coordinate 
{ 
x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx6/2; 
for(q=0;q<2;q++){//repeat variable in inner y coordinate 

y2=y3+(2*q-l)*ly5/2+(2*q-l)*ly6/2-ly6/2; 
for(z=0;z<lz6;z++) 
{ 
for(x=1;x<lx6;x++) 
{ 
Tt_n1[x2+x][y2+y][z2+z]= 
Tb6_n1[index][z+k*(lz6+1)]; 
} 
} 
} 
} 
} 
} 

//fifth level 
k=0; 
for(r=0;r<2;r++)//repeat variable in z coordinate 
{ 
z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz5/2; 
for(j=0;j<2;j++){//repeat variable in y coordinate 

y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2-ly5/2; 
for(p=0;p<2;p++){//repeat variable in x coordinate 

x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx5/2; 
for(y=0;y<ly5;y++) 
{ 
for(x=1;x<lx5;x++) 
{ 
Tt_n1[x2+x][y2+y][z2+z] = 
Tb5_n1[index][y+k*(ly5+1)]; 
} 
} 
} 
} 

//fourth level 
k=0; 
for(r=0;r<2;r++) 
{ 
z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz4/2; 
for(j=0;j<2;j++) 
{ 
y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2-ly4/2; 
for(x=0;x<lx4;x++) 
{ 
x2=x4e+x; 
for(y=1;y<ly4;y++) 
{ 
x2=x4e+x; 
for(z=1;z<lx4;z++) 
{ 
} 
} 
} 
} 
}
START OF CALC VESSEL SUBROUTINE (CALCULATES THE TEMPERATURE ON THE INTERFACE BETWEEN THE VESSEL AND TISSUE AT ALL APPLICABLE LOCATIONS, ENTER 0 FOR ARTERIES, 1 FOR VEINS)

void CalcVessel(int index)
{
    setborderVariable(index); //set the common blood border variables

    //calculate blood vessel temperature on sides
    //first level
    for(y=yla+l;y<ylb;y++)//x-y plane
    
    
    
    
    
    
    
    
    
    
    
    for(z=zla+l;z<zlb;z++)//left most sides
}
for(y=y1a+1;y<y1b;y++)
    Tt_n1[x2b][y][z] = ( Tt_n1[x2b-1][y][z] +
    Tb1_n1[index][l][x][y][z] ) / (1+deltaX*Bi);

    //side edges
    Tt_n1[x2b][ylb][z] = ( Tt_n1[x2b+1][ylb][z] +
    Tt_n1[x2b][ylb-1][z] ) / 2;
    for(z=z2b+1;z<z1b;z++)
        for(y=y1a+1;y<y1b;y++)
            Tt_n1[x2b][y][z] = ( Tt_n1[x2b-1][y][z] +
            Tt_n1[x2b][ylb-1][z] ) / 2;

        //side edges
    Tt_n1[x2b][y1a][z] = ( Tt_n1[x2b-1][y1a][z] +
            Tt_n1[x2b][y1a+1][z] ) / 2;
    for(x=x2b+1;x<=x1a;x++)
        Tt_n1[x][y1a][z] = ( Tt_n1[x][y1a+1][z] +
            Tt_n1[x][y1a-1][z] ) / 2;
        for(y=y1a+1;y<y1b;y++)
            Tt_n1[x][y][z] = ( Tt_n1[x][y+1][z] +
                Tt_n1[x][y-1][z] ) / 2;

        //save blood boarder temperature
    Tbd1[index][l][x1a-x] = Tt_n1[x][y1b][z1a];
    Tbd1[index][2][x1a-x] = Tt_n1[x][ylb][z1b];
    Tbd1[index][3][x1a-x] = Tt_n1[x][ylb][z1b];
    }

    //vertix
    Tt_n1[x2b][y1a][z1a] = ( Tt_n1[x2b-1][y1a][z1a] +
            Tt_n1[x2b][y1a+1][z1a] ) / 3;
    Tt_n1[x2b][y1b][z1a] = ( Tt_n1[x2b-1][y1b][z1a] +
            Tt_n1[x2b][y1b+1][z1a] ) / 3;
    Tt_n1[x2b][y1b][z1b] = ( Tt_n1[x2b-1][y1b][z1b] +
            Tt_n1[x2b][y1b+1][z1b] ) / 3;
    Tt_n1[x2b][ylb-1][z1a] = ( Tt_n1[x2b+1][ylb-1][z1a] +
            Tt_n1[x2b][ylb][z1a] ) / 3;
    Tt_n1[x2b][ylb-1][z1b] = ( Tt_n1[x2b+1][ylb-1][z1b] +
            Tt_n1[x2b][ylb][z1b] ) / 3;
    //save blood boarder temperature
    Tbd1[index][0][x1a-x2b] = Tt_n1[x2b][y1a][z1a];
    Tbd1[index][1][x1a-x2b] = Tt_n1[x2b][ylb][z1b];
    Tbd1[index][2][x1a-x2b] = Tt_n1[x2b][ylb][z1b];
    Tbd1[index][3][x1a-x2b] = Tt_n1[x2b][ylb][z1b];
    //level 2
    for(y=y2a+1;y<y2b;y++)//x-y plane
        for(x=x2a+1;x<x2b;x++)
            Tt_n1[x][y][z2a] = ( Tt_n1[x][y][z2a-1] +
                Tt_n1[x][y][z2a-1] ) / (1+deltaZ*Bi);
                Tbd1[index][y][z2a] = ( Tt_n1[x][y][z2a-1] +
                Tbd1[index][y][z2a-1] ) / (1+deltaZ*Bi);

    for(z=z2a-1;z<z2b;z++)
        for(x=x2a+1;x<x2b;x++)
            Tt_n1[x][y][z] = ( Tt_n1[x][y][z-1] +
                Tt_n1[x][y][z-1] ) / (1+deltaZ*Bi);
//y-z plane
for(y=y2a+1;y<y2b;y++)

Tt_n1[x2a][y][z] = ( Tt_n1[x2a-1][y][z] + Tt_n1[0][y][z] )
  * deltaX*Bi ) / (1+deltaX*Bi);
for(y=y2a+1;y<y2b;y++)

Tt_n1[x2b][y][z] = ( Tt_n1[x2b+1][y][z] + Tt_n1[0][y][z] )
  * deltaX*Bi ) / (1+deltaX*Bi);

//x-z plane
for(x=x2a+1;x<x3a;x++)

{Tt_n1[x2a][y2a][z] = ( Tt_n1[x2a-1][y2a][z] + Tt_n1[0][y2a][z] )
  * deltaY*Bi ) / (1+deltaY*Bi);
Tt_n1[x2a][y2b][z] = ( Tt_n1[x2a+1][y2b][z] + Tt_n1[0][y2b][z] )
  * deltaY*Bi ) / (1+deltaY*Bi);

for(x=x3b+1;x<x2b;x++)

{Tt_n1[x][y2a][z] = ( Tt_n1[x-1][y2a][z] + Tt_n1[0][y2a][z] )
  * deltaY*Bi ) / (1+deltaY*Bi);
Tt_n1[x][y2b][z] = ( Tt_n1[x+1][y2b][z] + Tt_n1[0][y2b][z] )
  * deltaY*Bi ) / (1+deltaY*Bi);

//vertical edges
Tt_n1[x2a][y2a][z] = ( Tt_n1[x2a-1][y2a][z] + Tt_n1[x2a][y2a-1][z] )
  / 2;
Tt_n1[x2b][y2a][z] = ( Tt_n1[x2b-1][y2a][z] + Tt_n1[x2b][y2a-1][z] )
  / 2;
Tt_n1[x2b][y2b][z] = ( Tt_n1[x2b-1][y2b][z] + Tt_n1[x2b][y2b-1][z] )
  / 2;

//vertical boarder edges of level 2 and 1
Tt_n1[x2b][y1a][z] = ( Tt_n1[x2b+1][y1a][z] + Tt_n1[x2b][y1a-1][z] )
  / 2;
Tt_n1[x2b][y1b][z] = ( Tt_n1[x2b+1][y1b][z] + Tt_n1[x2b][y1b-1][z] )
  / 2;

//horizontal y direction side edges
for(y=y2a+1;y<y2b;y++)

{Tt_n1[x2a][y][z2a] = ( Tt_n1[x2a-1][y][z2a] + Tt_n1[x2a][y][z2a+1] )
  / 2;
Tt_n1[x2a][y][z2b] = ( Tt_n1[x2a-1][y][z2b] + Tt_n1[x2a][y][z2b-1] )
  / 2;

for(y=y2a+1;y<y2b;y++)

{Tt_n1[x2b][y][z2a] = ( Tt_n1[x2b-1][y][z2a] + Tt_n1[x2b][y][z2a+1] )
  / 2;
Tt_n1[x2b][y][z2b] = ( Tt_n1[x2b-1][y][z2b] + Tt_n1[x2b][y][z2b-1] )
  / 2;

for(y=y2a+1;y<y2b;y++)

{Tt_n1[x2a][y2a][z] = ( Tt_n1[x2a-1][y2a][z] + Tt_n1[x2a][y2a+1][z] )
  / 2;
Tt_n1[x2a][y2b][z] = ( Tt_n1[x2a-1][y2b][z] + Tt_n1[x2a][y2b-1][z] )
  / 2;
\[ T_{t_{nl}}[x_{2b}][y][z_{2a}] = \frac{(T_{t_{nl}}[x_{2b-1}][y][z_{2a}] + T_{t_{nl}}[x_{2b}][y][z_{2a+1}])}{2}; \]

\[ T_{t_{nl}}[x_{2b}][y][z_{2b}] = \frac{(T_{t_{nl}}[x_{2b-1}][y][z_{2b}] + T_{t_{nl}}[x_{2b}][y][z_{2b-1}])}{2}; \]

//save blood border temperature y_{lb}+1 to y_{lb+1}

\[ T_{bd2}[index][1][y_{lb+1}] = T_{t_{nl}}[x_{2b}][y][z_{2a}]; \]
\[ T_{bd2}[index][2][y_{lb+1}] = T_{t_{nl}}[x_{2b}][y][z_{2b}]; \]

//horizontal x direction edges

for( x=x_{2a+1}; x<x_{3a}; x++ )
{
\[ T_{t_{nl}}[x][y_{lb}][z_{2a}] = \frac{(T_{t_{nl}}[x][y_{lb+1}][z_{2a}] + T_{t_{nl}}[x][y_{lb}][z_{2a+1}])}{2}; \]
\[ T_{t_{nl}}[x][y_{lb}][z_{2b}] = \frac{(T_{t_{nl}}[x][y_{lb+1}][z_{2b}] + T_{t_{nl}}[x][y_{lb}][z_{2b-1}])}{2}; \]
\[ T_{t_{nl}}[x][y_{lb+1}][z_{2a}] = \frac{(T_{t_{nl}}[x][y_{lb+1}][z_{2a}] + T_{t_{nl}}[x][y_{lb}][z_{2a+1}])}{2}; \]
\[ T_{t_{nl}}[x][y_{lb+1}][z_{2b}] = \frac{(T_{t_{nl}}[x][y_{lb+1}][z_{2b}] + T_{t_{nl}}[x][y_{lb}][z_{2b-1}])}{2}; \]
for( x=x_{3b+1}; x<x_{2b}; x++ )
{
\[ T_{t_{nl}}[x][y_{lb}][z_{2a}] = (T_{t_{nl}}[x][y_{lb+1}][z_{2a}] + T_{t_{nl}}[x][y_{lb}][z_{2a+1}]) / 3; \]
\[ T_{t_{nl}}[x][y_{lb}][z_{2b}] = (T_{t_{nl}}[x][y_{lb+1}][z_{2b}] + T_{t_{nl}}[x][y_{lb}][z_{2b+1}]) / 3; \]
\[ T_{t_{nl}}[x][y_{lb+1}][z_{2a}] = (T_{t_{nl}}[x][y_{lb+1}][z_{2a}] + T_{t_{nl}}[x][y_{lb}][z_{2a+1}]) / 3; \]
\[ T_{t_{nl}}[x][y_{lb+1}][z_{2b}] = (T_{t_{nl}}[x][y_{lb+1}][z_{2b}] + T_{t_{nl}}[x][y_{lb}][z_{2b+1}]) / 3; \]

//horizontal boarder edges of level 2 and 1

for( y=y_{lb+1}; y<y_{lb+1}; y++ )
{
\[ T_{bd2}[index][2][y_{lb+1}] = T_{t_{nl}}[x_{2a}][y][z_{2a}]; \]
\[ T_{bd2}[index][3][y_{lb+1}] = T_{t_{nl}}[x_{2b}][y][z_{2b}]; \]

//vertex

\[ T_{t_{nl}}[x_{2a}][y_{lb}][z_{2a}] = (T_{t_{nl}}[x_{2a}][y_{lb+1}][z_{2a}] + T_{t_{nl}}[x_{2a}][y_{lb}][z_{2a+1}]) / 3; \]
\[ T_{t_{nl}}[x_{2a}][y_{lb}][z_{2b}] = (T_{t_{nl}}[x_{2a}][y_{lb+1}][z_{2b}] + T_{t_{nl}}[x_{2a}][y_{lb}][z_{2b+1}]) / 3; \]
\[ T_{t_{nl}}[x_{2a}][y_{lb+1}][z_{2a}] = (T_{t_{nl}}[x_{2a}][y_{lb+1}][z_{2a}] + T_{t_{nl}}[x_{2a}][y_{lb}][z_{2a+1}]) / 3; \]
\[ T_{t_{nl}}[x_{2a}][y_{lb+1}][z_{2b}] = (T_{t_{nl}}[x_{2a}][y_{lb+1}][z_{2b}] + T_{t_{nl}}[x_{2a}][y_{lb}][z_{2b+1}]) / 3; \]
\[ T_{t_{nl}}[x_{2a}][y_{lb+2}][z_{2a}] = (T_{t_{nl}}[x_{2a}][y_{lb+1}][z_{2a}] + T_{t_{nl}}[x_{2a}][y_{lb}] [z_{2a+1}]) / 3; \]
\[ T_{t_{nl}}[x_{2a}][y_{lb+2}][z_{2b}] = (T_{t_{nl}}[x_{2a}][y_{lb+1}][z_{2b}] + T_{t_{nl}}[x_{2a}][y_{lb}][z_{2b+1}]) / 3; \]
\[ T_{t, n}[x, y][z] = \frac{1}{2} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]

\[ T_{t, n}[x, y][z] = \frac{1}{2} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]

//save blood border temperature y1a+1 to y1b-1

\[ Tbd2[index][1][y-a2a] = T_{t, n}[x, y][z2a]; \]
\[ Tbd2[index][3][y-a2a] = T_{t, n}[x, y][z2b]; \]

//vertex of border edges of level 2 and 1

\[ T_{t, n}[x, y][z] = \frac{1}{4} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]

\[ T_{t, n}[x, y][z] = \frac{1}{4} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]

//save blood border temperature y1a

\[ Tbd2[index][1][y-a2a] = T_{t, n}[x, y][z2a]; \]
\[ Tbd2[index][3][y-a2a] = T_{t, n}[x, y][z2b]; \]

//level 3

for (j=0;j<2;j++)

\[ y7 = y3+j*3 \]
\[ y2 = y2a+j*3 \]

\[ y3 = y2a+j*3 \]
\[ y4 = y3a+j*3 \]

for (z=z3a; z<z3b; z++) //x-z plane

\[ T_{t, n}[x, y][z] = \frac{1}{4} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]

\[ T_{t, n}[x, y][z] = \frac{1}{4} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]

\[ T_{t, n}[x, y][z] = \frac{1}{4} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]

//vertical edges

for (z=z3a; z<z3b; z++) //x-z plane

\[ T_{t, n}[x, y][z] = \frac{1}{4} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]

\[ T_{t, n}[x, y][z] = \frac{1}{4} \left( T_{t, n-1}[x, y][z] + T_{t, n-1}[x, y][z+1] \right); \]
if (j == 0)
{
  // save blood border temperature z3a+1 to z3b-1
  Tbd3[index][0][z-z3a+j*(lz3+1)] = Ttnl[x3a][y4][z];
  Tbd3[index][1][z-z3a+j*(lz3+1)] = Ttnl[x3b][y4][z];
}
else
{
  // save blood border temperature z3a+1 to z3b-1
  Tbd3[index][2][z-z3a+j*(lz3+1)] = Ttnl[x3a][y3][z];
  Tbd3[index][3][z-z3a+j*(lz3+1)] = Ttnl[x3b][y3][z];
}
}
for (z = z3a+1; z < z3b; z++)
{
  Ttnl[x3a][y3][z] = (Ttnl[x3a+1][y3][z] + Ttnl[x3a][y3+2*j-l][z])/2;
  Ttnl[x3b][y3][z] = (Ttnl[x3b-1][y3][z] + Ttnl[x3b][y3+2*j-1][z])/2;
  if (j == 0)
  {
    // save blood border temperature z3a+1 to z2a-1
    Tbd3[index][0][z-z3a+j*(lz3+1)] = Ttnl[x3b][y3][z];
    Tbd3[index][1][z-z3a+j*(lz3+1)] = Ttnl[x3a][y3][z];
  }
  else
  {
    // save blood border temperature z3a+1 to z2a-1
    Tbd3[index][2][z-z3a+j*(lz3+1)] = Ttnl[x3a][y3][z];
    Tbd3[index][3][z-z3a+j*(lz3+1)] = Ttnl[x3b][y3][z];
  }
}
// vertex
Ttnl[x3a][y7][z3a] = (Ttnl[x3a+1][y7][z3a] + Ttnl[x3a][y7+2][z3a+1])/2;
Ttnl[x3a][y7][z3b] = (Ttnl[x3a+1][y7][z3b] + Ttnl[x3a][y7+2][z3b-1])/2;
Ttnl[x3a][y2][z3a] = (Ttnl[x3a+1][y2][z3a] + Ttnl[x3a][y2+2][z3a+1])/2;
Ttnl[x3a][y2][z3b] = (Ttnl[x3a+1][y2][z3b] + Ttnl[x3a][y2][z3b-1])/2;
\[ T_{t_n} l(x_{3b}[y_{7}][z_{3a}] = ( T_{t_n} l(x_{3b-1}[y_{7}][z_{3a}] + T_{t_n} l(x_{3b}[y_{7}][z_{3a+1}])/2; \]
\[ T_{t_n} l(x_{3b}[y_{7}][z_{3b}] = ( T_{t_n} l(x_{3b-1}[y_{7}][z_{3b}] + T_{t_n} l(x_{3b}[y_{7}][z_{3b-1}])/2; \]
\[ T_{t_n} l(x_{3b}[y_{2}][z_{3a}] = ( T_{t_n} l(x_{3b-1}[y_{2}][z_{3a}] + T_{t_n} l(x_{3b}[y_{2}][z_{3a+1}])/2; \]
\[ T_{t_n} l(x_{3b}[y_{2}][z_{3b}] = ( T_{t_n} l(x_{3b-1}[y_{2}][z_{3b}] + T_{t_n} l(x_{3b}[y_{2}][z_{3b-1}])/2; \]

//save blood border temperature \( z_{3a} \)
\[ T_{b_d3}[index][0][0+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{7}][z_{3a}]; \]
\[ T_{b_d3}[index][1][0+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{7}][z_{3a}]; \]
\[ T_{b_d3}[index][2][0+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{2}][z_{3a}]; \]
\[ T_{b_d3}[index][3][0+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{2}][z_{3a}]; \]

//save blood border temperature \( z_{3b} \)
\[ T_{b_d3}[index][0][l_{z3}+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{7}][z_{3b}]; \]
\[ T_{b_d3}[index][1][l_{z3}+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{7}][z_{3b}]; \]
\[ T_{b_d3}[index][2][l_{z3}+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{2}][z_{3b}]; \]
\[ T_{b_d3}[index][3][l_{z3}+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{2}][z_{3b}]; \]

//vertical boarder edges of level 2 and 3

for \( z = z_{2a}+1 \) to \( z_{2b} \)
\[ T_{t_n} l(x_{3a}[y_{3}][z_{2a}] = ( T_{t_n} l(x_{3a-1}[y_{3}][z_{2a}] + T_{t_n} l(x_{3a}[y_{3}][z_{2a-1}])/2; \]
\[ T_{t_n} l(x_{3a}[y_{3}][z_{2b}] = ( T_{t_n} l(x_{3a-1}[y_{3}][z_{2b}] + T_{t_n} l(x_{3a}[y_{3}][z_{2b+1}])/2; \]

//vertex of boarder edges of level 2 and 3

for \( z = z_{2a+1} \) to \( z_{2b} \)
\[ T_{t_n} l(x_{3a}[y_{3}][z_{2a}] = ( T_{t_n} l(x_{3a-1}[y_{3}][z_{2a}] + T_{t_n} l(x_{3a}[y_{3}][z_{2a+1}])/2; \]
\[ T_{t_n} l(x_{3a}[y_{3}][z_{2b}] = ( T_{t_n} l(x_{3a-1}[y_{3}][z_{2b}] + T_{t_n} l(x_{3a}[y_{3}][z_{2b+1}])/2; \]

if \( j == 0 \)
\[ T_{b_d3}[index][2][z_{2a}+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{3}][z_{2a}]; \]
\[ T_{b_d3}[index][3][z_{2a}+j*(l_{z3}+1)] = T_{t_n} l(x_{3a}[y_{3}][z_{2a}] + T_{t_n} l(x_{3a}[y_{3}][z_{2a}]); \]
{  //save blood boarder temperature z2a
    Tbd3[index][2][z2a-z3a+j*(lz3+1)] = Tt_nl[x3a][y3][z2a];
    Tbd3[index][3][z2a-z3a+j*(lz3+1)] = Tt_nl[x3b][y3][z2a];

    //save blood boarder temperature z2b
    Tbd3[index][2][z2b-z3a+j*(lz3+1)] = Tt_nl[x3a][y3][z2b];
    Tbd3[index][3][z2b-z3a+j*(lz3+1)] = Tt_nl[x3b][y3][z2b];
}
else
{  //save blood boarder temperature z2a
    Tbd3[index][0][z2a-z3a+j*(lz3+1)] = Tt_nl[x3a][y3][z2a];
    Tbd3[index][1][z2a-z3a+j*(lz3+1)] = Tt_nl[x3b][y3][z2a];

    //save blood boarder temperature z2b
    Tbd3[index][0][z2b-z3a+j*(lz3+1)] = Tt_nl[x3a][y3][z2b];
    Tbd3[index][1][z2b-z3a+j*(lz3+1)] = Tt_nl[x3b][y3][z2b];
}

//fourth level
k=0;
for(r=0;r<2;r++)
    for(j=0;j<2;j++)
        
        z1 = cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2; 
        z2 = cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2; 
        z3 = cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-
            lz5/2; 
        z4 = cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+lz5/2; 
        z5 = cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4; 
        z6 = cenZ+(2*r-1)*lz3/2; 
        y7 = cenY+(2*r-1)*ly2/2+(2*r-1)*ly3/2-
            ly4/2; 
        y2 = cenY+(2*r-1)*ly2/2+(2*r-1)*ly3/2+ly4/2; 
        for(y=y7+1;y<y2;++y)//x-y plane
            for(x=x4a+1;x<x4b;++x)
                Tt_n1[x][y][z5] = ( Tt_n1[x][y][z5+2*r-1] 
                                       + Tbd_n1[index][x-x4a+k*(lx4+1)] 
                                       * deltaZ*Bi ) / (1+deltaZ*Bi); 
        for(y=y7+1;y<y2;++y)//x-y plane
            for(x=x3b+1;x<x4a;++x)
                Tt_n1[x][y][z6] = ( Tt_n1[x][y][z6+1-2*r] 
                                       + Tbd_n1[index][x-x4a+k*(lx4+1)] 
                                       * deltaZ*Bi ) / (1+deltaZ*Bi); 
        for(y=y7+1;y<y2;++y)//x-y plane
            for(x=x4a+1;x<x4b;++x)
                Tt_n1[x][y][z6] = ( Tt_n1[x][y][z6+2-2*r] 
                                       + Tbd_n1[index][x-x4a+k*(lx4+1)] 
                                       * deltaZ*Bi ) / (1+deltaZ*Bi); 
        for(r=0;r<2;r++)
            for(j=0;j<2;j++)
                
                for(z=zl+1;z<z2;++z)
                    for(x=x4a+1;x<x4b;++x)//x-z plane
                        
                        Tt_n1[x][y7][z] = ( Tt_n1[x][y7-1][z] 
                                           + Tbd_n1[index][x-x4a+k*(lx4+1)] 
                                           * deltaY*Bi ) / (1+deltaY*Bi); 
                    for(z=zl+1;z<z2;++z)
                        for(y=y7+1;y<y2;++y)
                            
                            Tt_n1[x4a][y][z] = ( Tt_n1[x4a-1][y][z] 
                                           + Tbd_n1[index][x4a-x4a+k*(lx4+1)] 
                                           * deltaY*Bi ) / (1+deltaY*Bi); 
                        for(z=zl+1;z<z2;++z)
                            for(y=y7+1;y<y2;++y)
                                
                                Tt_n1[x4a][y][z] = ( Tt_n1[x4a-1][y][z] 
                                               + Tbd_n1[index][x4a-x4a+k*(lx4+1)] 
                                               * deltaY*Bi ) / (1+deltaY*Bi); 

\[ T_{t, n}[x_4b][y][z] = \frac{(T_{t, n}[x_4b+l][y][z] + T_{b4, n}[index][ix_4-1+k*(ix_4+l)] \cdot \delta x \cdot B_i)}{(l + \delta x \cdot B_i)}; \]

//vertical edges
\[ T_{t, n}[x_4a][y7][z] = \frac{(T_{t, n}[x_4a+l][y7][z] + T_{t, n}[x_4a][y7+1][z])}{2}; \]
\[ T_{t, n}[x_4a][y2][z] = \frac{(T_{t, n}[x_4a+l][y2][z] + T_{t, n}[x_4a][y2-1][z])}{2}; \]
\[ T_{t, n}[x_4b][y7][z] = \frac{(T_{t, n}[x_4b-l][y7][z] + T_{t, n}[x_4b][y7+1][z])}{2}; \]
\[ T_{t, n}[x_4b][y2][z] = \frac{(T_{t, n}[x_4b-l][y2][z] + T_{t, n}[x_4b][y2-1][z])}{2}; \]

for(z = z4+1; z < z2; z++) //y-z plane

\{ for(y = y7+1; y < y2; y++)

\[ T_{t, n}[x_4a][y][z] = \frac{(T_{t, n}[x_4a-l][y][z] + T_{b4, n}[index][ix_4+1]) \cdot \delta x \cdot B_i)}{(l + \delta x \cdot B_i)}; \]

//vertical edges
\[ T_{t, n}[x_4a][y7][z] = \frac{(T_{t, n}[x_4a+l][y7][z] + T_{t, n}[x_4a][y7+1][z])}{2}; \]
\[ T_{t, n}[x_4a][y2][z] = \frac{(T_{t, n}[x_4a+l][y2][z] + T_{t, n}[x_4a][y2-1][z])}{2}; \]
\[ T_{t, n}[x_4b][y7][z] = \frac{(T_{t, n}[x_4b-l][y7][z] + T_{t, n}[x_4b][y7+1][z])}{2}; \]
\[ T_{t, n}[x_4b][y2][z] = \frac{(T_{t, n}[x_4b-l][y2][z] + T_{t, n}[x_4b][y2-1][z])}{2}; \]

//horizontal y direction edges
for(y = y7+1; y < y2; y++)

\{

\[ T_{t, n}[x_4a][y][z] = \frac{(T_{t, n}[x_4a-l][y][z] + T_{t, n}[x_4a][y][z+1])}{2}; \]
\[ T_{t, n}[x_4a][y][z] = \frac{(T_{t, n}[x_4a+l][y][z] + T_{t, n}[x_4a][y-1][z])}{2}; \]
\[ T_{t, n}[x_4b][y][z] = \frac{(T_{t, n}[x_4b-l][y][z] + T_{t, n}[x_4b][y+1][z])}{2}; \]
\[ T_{t, n}[x_4b][y][z] = \frac{(T_{t, n}[x_4b-l][y][z] + T_{t, n}[x_4b][y-1][z])}{2}; \]

\}

//horizontal x direction edges
for(x = x4a+1; x < x4b; x++)

\{ for(y = y7+1; y < y2; y++)

\[ T_{t, n}[x_4a][y][z] = \frac{(T_{t, n}[x_4a-l][y][z] + T_{t, n}[x_4a][y][z+1])}{2}; \]
\[ T_{t, n}[x_4a][y][z] = \frac{(T_{t, n}[x_4a+l][y][z] + T_{t, n}[x_4a][y-1][z])}{2}; \]
\[ T_{t, n}[x_4b][y][z] = \frac{(T_{t, n}[x_4b-l][y][z] + T_{t, n}[x_4b][y+1][z])}{2}; \]
\[ T_{t, n}[x_4b][y][z] = \frac{(T_{t, n}[x_4b-l][y][z] + T_{t, n}[x_4b][y-1][z])}{2}; \]

\}

if(r == 0) //save blood boarder temperature x4a+l to x4b-1

\[ T_{bd4}[index][0][x-x4a+k*(ix_4+1)] = T_{t, n}[x_4a][y][z]; \]
\[ T_{bd4}[index][1][x-x4a+k*(ix_4+1)] = T_{t, n}[x][y][z5]; \]

else

\{ //save blood boarder temperature x4a+l to x4b-1

\[ T_{bd4}[index][2][x-x4a+k*(ix_4+1)] = T_{t, n}[x][y][z5]; \]
\[ T_{bd4}[index][3][x-x4a+k*(ix_4+1)] = T_{t, n}[x][y2][z5]; \]

\}

for(x = x4a+1; x < x4a+1; x++)

\{
\[
T_{t, n_1}[x][y][z_6] = (T_{t, n_1}[x][y][z_6] + T_{t, n_1}[x][y][z_6 + 2*r - 1]) / 2;
\]
\[
T_{t, n_1}[x][y][z_6] = (T_{t, n_1}[x][y][z_6] + T_{t, n_1}[x][y][z_6 + 2*r - 1]) / 2;
\]
\[
\text{if}(r == 0)
\]
\{

// save blood border temperature x4a+1 to x3a-1

\[
T_{bid}[index][0][x-x4a+k*(lx4+1)] = T_{t, n_1}[x][y][z_6];
\]
\[
T_{bid}[index][1][x-x4a+k*(lx4+1)] = T_{t, n_1}[x][y][z_6];
\]
\}
\}
\}

for(x=x3b+1; x<x4b; x++)
{

\[
T_{t, n_1}[x][y][z_6] = (T_{t, n_1}[x][y][z_6] + T_{t, n_1}[x][y][z_6 + 2*r - 1]) / 2;
\]
\[
T_{t, n_1}[x][y][z_6] = (T_{t, n_1}[x][y][z_6] + T_{t, n_1}[x][y][z_6 + 2*r - 1]) / 2;
\]
\[
\text{if}(r == 0)
\]
\{

// save blood border temperature x3b+1 to x4b-1

\[
T_{bid}[index][2][x-x4a+k*(lx4+1)] = T_{t, n_1}[x][y][z_6];
\]
\[
T_{bid}[index][3][x-x4a+k*(lx4+1)] = T_{t, n_1}[x][y][z_6];
\]
\}
\}

else
{


// vertex

\[
T_{t, n_1}[x][y][z_7] = (T_{t, n_1}[x][y][z_7] + T_{t, n_1}[x][y][z_7 + 1]) / 2;
\]
\[
T_{t, n_1}[x][y][z_7] = (T_{t, n_1}[x][y][z_7] + T_{t, n_1}[x][y][z_7 + 1]) / 2;
\]
\[
T_{t, n_1}[x][y][z_7] = (T_{t, n_1}[x][y][z_7] + T_{t, n_1}[x][y][z_7 + 1]) / 2;
\]
\}
\}

// save blood border temperature x4a to x4b

\[
T_{bid}[index][0][x-x4a+k*(lx4+1)] = T_{t, n_1}[x][y][z_7];
\]
Tbd4[index][1][0+k*(lx4+1)] = Tt_n1[x4a][y2][z1];
Tbd4[index][2][0+k*(lx4+1)] = Tt_n1[x4a][y7][z2];
Tbd4[index][3][0+k*(lx4+1)] = Tt_n1[x4a][y2][z2];
Tbd4[index][0][lx4+k*(lx4+1)] = Tt_n1[x4a][y7][z1];
Tbd4[index][2][lx4+k*(lx4+1)] = Tt_n1[x4a][y2][z2];
Tbd4[index][3][lx4+k*(lx4+1)] = Tt_n1[x4a][y2][z2];

//save blood boarder temperature x4b
Tbd4[index][1][lx4+k*(lx4+1)] = Tt_n1[x4a][y2][z1];
Tbd4[index][2][lx4+k*(lx4+1)] = Tt_n1[x4a][y7][z2];
Tbd4[index][3][lx4+k*(lx4+1)] = Tt_n1[x4a][y7][z1];

//horizontal x direction boarder edges of level 4 and 3
for(x=x3a+1;x<x3b;x++)
{
    Tt_n1[x][y7][z6] = ( Tt_n1[x][y7-1][z6] + Tt_n1[x][y7+1][z6] + Tt_n1[x][y7+2][z6]+ Tt_n1[x][y7+3][z6]) / 4;
    Tt_n1[x][y2][z6] = ( Tt_n1[x][y2-1][z6] + Tt_n1[x][y2+1][z6] + Tt_n1[x][y2+2][z6]+ Tt_n1[x][y2+3][z6]) / 4;
    if(r==0)
    {
        //save blood boarder temperature x3a+1 to x3b-1
        Tbd4[index][2][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z6];
        Tbd4[index][3][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z6];
    }
    else
    {
        //save blood boarder temperature x3a+1 to x3b-1
        Tbd4[index][0][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z6];
        Tbd4[index][1][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z6];
    }
}

//vertical y direction boarder edges of level 4 and 3
for(y=y7+1;y<y2+1;y++)
{
    Tt_n1[x3a][y][z6] = ( Tt_n1[x3a-1][y][z6] + Tt_n1[x3a+1][y][z6] + Tt_n1[x3a+2][y][z6]+ Tt_n1[x3a+3][y][z6]) / 4;
    Tt_n1[x3b][y][z6] = ( Tt_n1[x3b-1][y][z6] + Tt_n1[x3b+1][y][z6] + Tt_n1[x3b+2][y][z6]+ Tt_n1[x3b+3][y][z6]) / 4;
    if(r==0)
    {
        //save blood boarder temperature x3a to x3b
        Tbd4[index][2][x3a-x4a+k*(lx4+1)] = Tt_n1[x3a][y7][z6];
        Tbd4[index][3][x3a-x4a+k*(lx4+1)] = Tt_n1[x3a][y2][z6];
    }
    else
    {
        //save blood boarder temperature x3a to x3b
        Tbd4[index][0][x3a-x4a+k*(lx4+1)] = Tt_n1[x3a][y7][z6];
        Tbd4[index][1][x3a-x4a+k*(lx4+1)] = Tt_n1[x3a][y2][z6];
    }
}
Tbd4[index][3][x3b-x4a+k*(lx4+l)] = Tt_n1[x3b][y2][z6];

else
{
  //save blood border temperature x3a
  Tbd4[index][0][x3a-x4a+k*(lx4+l)] = Tt_n1[x3a][y7][z6];
  Tbd4[index][1][x3a-x4a+k*(lx4+l)] = Tt_n1[x3a][y2][z6];

  //save blood border temperature x3b
  Tbd4[index][0][x3b-x4a+k*(lx4+l)] = Tt_n1[x3b][y7][z6];
  Tbd4[index][1][x3b-x4a+k*(lx4+l)] = Tt_n1[x3b][y2][z6];
}

k++;
\[ T_{t,nl}[x][y][z] = \frac{( T_{t,nl}[x+1-2p][y][z] + T_{t,nl}[x][y-1][z] )}{2}; \]
\[ T_{t,nl}[x][y+1][z] = \frac{( T_{t,nl}[x+1-2p][y][z] + T_{t,nl}[x][y+1][z] )}{2}; \]

// horizontal y direction edges
for \( y = 1, y < n, y++ \)

\{
    T_{t,nl}[x][y][z1] = \frac{( T_{t,nl}[x+1-2p][y][z1] + T_{t,nl}[x][y][z1+1] )}{2};
    T_{t,nl}[x][y][z2] = \frac{( T_{t,nl}[x+1-2p][y][z2] + T_{t,nl}[x][y][z2-1] )}{2};
\}

if \( p == 0 \)

\{ 
    Tbd5[index][0][y][y+7+k*(y+1)] = T_{t,nl}[x][y][z1];
    Tbd5[index][2][y][y+7+k*(y+1)] = T_{t,nl}[x][y][z2];
\}

else 

\{ 
    Tbd5[index][0][y][y+7+k*(y+1)] = T_{t,nl}[x][y][z1];
    Tbd5[index][2][y][y+7+k*(y+1)] = T_{t,nl}[x][y][z2];
\}

// vertex
for \( y = 1, y < n, y++ \)

\{
    T_{t,nl}[x][y][z1] = \frac{( T_{t,nl}[x+2p-1][y][z1] + T_{t,nl}[x][y][z1+1] )}{2};
    T_{t,nl}[x][y][z2] = \frac{( T_{t,nl}[x+2p-1][y][z2] + T_{t,nl}[x][y][z2-1] )}{2};
\}

if \( p == 0 \)

\{ 
    Tbd5[index][0][y][y+7+k*(y+1)] = T_{t,nl}[x][y][z1];
    Tbd5[index][2][y][y+7+k*(y+1)] = T_{t,nl}[x][y][z2];
\}

else 

\{ 
    Tbd5[index][0][y][y+7+k*(y+1)] = T_{t,nl}[x][y][z1];
    Tbd5[index][2][y][y+7+k*(y+1)] = T_{t,nl}[x][y][z2];
\}
\[ T_{n1}[x1][y7][z1] = \left( T_{n1}[x1][y7][z1+1] \right)/2; \]
\[ T_{n1}[x1][y7][z2] = \left( T_{n1}[x1][y7+1][z2] + T_{n1}[x1][y7][z2+1] \right)/2; \]
\[ T_{n1}[x1][y2][z1] = \left( T_{n1}[x1][y2-1][z1] + T_{n1}[x1][y2][z1+1] \right)/2; \]
\[ T_{n1}[x1][y2][z2] = \left( T_{n1}[x1][y2-1][z2] + T_{n1}[x1][y2][z2+1] \right)/2; \]
\[ T_{n1}[x2][y7][z1] = \left( T_{n1}[x2][y7+1][z1] + T_{n1}[x2][y7][z1+1] \right)/2; \]
\[ T_{n1}[x2][y7][z2] = \left( T_{n1}[x2][y7+1][z2] + T_{n1}[x2][y7][z2+1] \right)/2; \]
\[ T_{n1}[x2][y2][z1] = \left( T_{n1}[x2][y2-1][z1] + T_{n1}[x2][y2][z1+1] \right)/2; \]
\[ T_{n1}[x2][y2][z2] = \left( T_{n1}[x2][y2-1][z2] + T_{n1}[x2][y2][z2+1] \right)/2; \]
\[ Tbd5[index][0][0+k*(ly5+l)] = T_{n1}[x1][y7][z1]; \]
\[ Tbd5[index][1][0+k*(ly5+l)] = T_{n1}[x2][y7][z1]; \]
\[ Tbd5[index][2][0+k*(ly5+l)] = T_{n1}[x1][y7][z2]; \]
\[ Tbd5[index][3][0+k*(ly5+l)] = T_{n1}[x2][y7][z2]; \]
\[ Tbd5[index][0][ly5+k*(ly5+1)] = T_{n1}[x1][y2][z1]; \]
\[ Tbd5[index][1][ly5+k*(ly5+1)] = T_{n1}[x2][y2][z1]; \]
\[ Tbd5[index][2][ly5+k*(ly5+1)] = T_{n1}[x1][y2][z2]; \]
\[ Tbd5[index][3][ly5+k*(ly5+1)] = T_{n1}[x2][y2][z2]; \]

//horizontal y direction boarder edges of level 4 and 5
for(y=y3+1;y<y4;y++)
{
    T_{n1}[x4][y][z1] = \left( T_{n1}[x4+2*p-1][y][z1] + T_{n1}[x4][y][z1-1] \right)/2; \]
    T_{n1}[x4][y][z2] = \left( T_{n1}[x4+2*p-1][y][z2] + T_{n1}[x4][y][z2+1] \right)/2; \]
    if(p==0)
    {
        Tbd5[index][1][y-7+k*(ly5+1)] = T_{n1}[x4][y][z1]; \]
        Tbd5[index][3][y-7+k*(ly5+1)] = T_{n1}[x4][y][z2]; \]
    } \]
else
{
    Tbd5[index][0][y-7+k*(ly5+1)] = T_{n1}[x4][y][z1]; \]
    Tbd5[index][2][y-7+k*(ly5+1)] = T_{n1}[x4][y][z2]; \]
}

//vertix of boarder edges of level 5 and 4
T_{n1}[x4][y3][z1] = \left( T_{n1}[x4][y3][z1+1] + T_{n1}[x4][y3-1][z1] \right)/4; \]
T_{n1}[x4][y3][z2] = \left( T_{n1}[x4][y3+1][z2] + T_{n1}[x4][y3-1][z2] \right)/4; \]
T_{n1}[x4][y4][z1] = \left( T_{n1}[x4][y4][z1+1] + T_{n1}[x4][y4-1][z1] \right)/4; \]
T_{n1}[x4][y4][z2] = \left( T_{n1}[x4][y4+1][z2] + T_{n1}[x4][y4-1][z2] \right)/4; \]
if(p==0)
{
} \]
Tbd5[index][1][y3-y7+k*(ly5+l)]=
Tt_nl[x4][y3][z1];

Tbd5[index][3][y3-y7+k*(ly5+l)]=
Tt_nl[x4][y3][z2];

Tbd5[index][1][y4-y7+k*(ly5+l)]=
Tt_nl[x4][y4][z1];

Tbd5[index][3][y4-y7+k*(ly5+l)]=
Tt_nl[x4][y4][z2];

Tbd5[index][0][y3-y7+k*(ly5+l)]=
Tt_nl[x4][y3][z1];

Tbd5[index][2][y3-y7+k*(ly5+l)]=
Tt_nl[x4][y3][z2];

Tbd5[index][0][y4-y7+k*(ly5+l)]=
Tt_nl[x4][y4][z1];

Tbd5[index][2][y4-y7+k*(ly5+l)]=
Tt_nl[x4][y4][z2];

k++;

//sixth level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
  for(j=0;j<2;j++)//repeat variable in y coordinate
    for(p=0;p<2;p++)//repeat variable in x coordinate{
      for(q=0;q<2;q++)//repeat variable in inner y coordinate
      {
        x1=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-
lx6/2;

        x2=cenX+(2*p-1)*lx4/2+(2*p-
1)*lx5/2+lx6/2;

        y7=cenY+(2*j-1)*ly2/2+(2*j-
1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2-
ly6/2;

        y2=cenY+(2*j-1)*ly2/2+(2*j-
1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2+ly6/2;

        z1=cenZ+(2*r-1)*lz3/2+(2*r-
1)*lz4/2+2lz6/2;

        z2=cenZ+(2*r-1)*lz3/2+(2*r-
1)*lz4/2+lz6/2;

        z3=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-
lz5/2;

        z4=cenZ+(2*r-1)*lz3/2+(2*r-
1)*lz4/2+lz5/2;

        y3=cenY+(2*j-1)*ly2/2+(2*j-
1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2;

        y4=cenY+(2*j-1)*ly2/2+(2*j-
1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2;

        for(z=z1;z<z2;z++)
          for(y=y7;y<y2;y++)
          {
            Tt_nl[x1][y][z] = ( Tt_nl[x1-
1][y][z] + Tbd5[index][z-zl+k*(lz6+1)]
                          * deltaX*Bi ) / (1+deltaX*Bi);

            Tt_nl[x2][y][z] = ( Tt_nl[x2+
1][y][z] + Tbd5[index][z-zl+k*(lz6+1)]
                          * deltaX*Bi ) / (1+deltaX*Bi);

            for(z=z1;z<z2;z++)
              for(x=x1+1;x<x2;x++)//x-z plane
                Tt_nl[x][y4][z] = ( Tt_nl[x][y4+
2*q-1][z] + Tbd6[index][z-z1+k*(lz6+1)]
                                * deltaY*Bi ) / (1+deltaY*Bi);

              for(z=z1;z<z3;z++)
                for(x=x1+1;x<x2;x++)//x-z plane
                  for(y=y7;y<y2;y++)
                    for(q=0;q<2;q++)//repeat variable in inner y coordinate
                      {
                        x1=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-
lx6/2;

                        x2=cenX+(2*p-1)*lx4/2+(2*p-
1)*lx5/2+lx6/2;

                        y7=cenY+(2*j-1)*ly2/2+(2*j-
1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2-ly6/2;
\[
T_{t_n l}[x][y][z] = (T_{t_n l}[x][y+1][z] + T_{b_n l}[index][z-l] + k*(lz6+1] * \text{deltaY*B}i) / (1+\text{deltaY*B}i);
\]

//vertical edges
\[
T_{t_n l}[x][y][z] = (T_{t_n l}[x][y+1][z] + T_{t_n l}[x+1][y][z]) / 2;
\]
\[
T_{t_n l}[x][y][z] = (T_{t_n l}[x][y+1][z] + T_{t_n l}[x][y+2][z]) / 2;
\]

\[
i(q==0)
\]

\[
T_{b_0 l}[index][2][z-l] + k*(lz6+1] = T_{t_n l}[x][y][z];
\]
\[
T_{b_0 l}[index][3][z-l] + k*(lz6+1] = T_{t_n l}[x+1][y][z];
\]

else

\[
T_{b_0 l}[index][0][z-l] + k*(lz6+1] = T_{t_n l}[x][y][z];
\]
\[
T_{b_0 l}[index][1][z-l] + k*(lz6+1] = T_{t_n l}[x+1][y][z];
\]

for(z=z+1; z<z+1; z++)

//vertical edges
\[
T_{t_n l}[x][y][z] = (T_{t_n l}[x][y+1][z] + T_{t_n l}[x][y+2][z]) / 2;
\]
\[
T_{t_n l}[x][y][z] = (T_{t_n l}[x+1][y][z] + T_{t_n l}[x+1][y+1][z]) / 2;
\]

\[
i(q==0)
\]

\[
T_{b_0 l}[index][2][z-l] + k*(lz6+1] = T_{t_n l}[x][y][z];
\]
\[
T_{b_0 l}[index][3][z-l] + k*(lz6+1] = T_{t_n l}[x+1][y][z];
\]

else

\[
T_{b_0 l}[index][0][z-l] + k*(lz6+1] = T_{t_n l}[x][y][z];
\]
\[
T_{b_0 l}[index][1][z-l] + k*(lz6+1] = T_{t_n l}[x+1][y][z];
\]

for(z=z+1; z<z+1; z++)

//vertical edges
\[
T_{t_n l}[x][y][z] = (T_{t_n l}[x][y+1][z] + T_{t_n l}[x][y+2][z]) / 2;
\]
\[
T_{t_n l}[x][y][z] = (T_{t_n l}[x+1][y][z] + T_{t_n l}[x+1][y+1][z]) / 2;
\]

\[
i(q==0)
\]
// vertex

\[
T_{t_n}[x_1][y_3][z_1] = \left( T_{t_n}[x_1+1][y_3][z_1] + T_{t_n}[x_1][y_3][z_1+1] \right) / 2;
\]

\[
T_{t_n}[x_1][y_2][z_1] = \left( T_{t_n}[x_1+1][y_2][z_1] + T_{t_n}[x_1][y_2][z_1+1] \right) / 2;
\]

\[
T_{t_n}[x_1][y_2][z_2] = \left( T_{t_n}[x_1+1][y_2][z_2] + T_{t_n}[x_1][y_2][z_2+1] \right) / 2;
\]

\[
T_{t_n}[x_1][y_2][z_3] = \left( T_{t_n}[x_1+1][y_2][z_3] + T_{t_n}[x_1][y_2][z_3+1] \right) / 2;
\]

\[
T_{t_n}[x_2][y_7][z_1] = \left( T_{t_n}[x_2+1][y_7][z_1] + T_{t_n}[x_2][y_7][z_1+1] \right) / 2;
\]

\[
T_{t_n}[x_2][y_7][z_2] = \left( T_{t_n}[x_2+1][y_7][z_2] + T_{t_n}[x_2][y_7][z_2+1] \right) / 2;
\]

\[
T_{t_n}[x_2][y_7][z_3] = \left( T_{t_n}[x_2+1][y_7][z_3] + T_{t_n}[x_2][y_7][z_3+1] \right) / 2;
\]

\[
T_{t_n}[x_2][y_7][z_4] = \left( T_{t_n}[x_2+1][y_7][z_4] + T_{t_n}[x_2][y_7][z_4+1] \right) / 2;
\]

\[
T_{t_n}[x_1][y_3][z] = \left( T_{t_n}[x_1-1][y_3][z] + T_{t_n}[x_1][y_3+2^q][z] \right) / 2;
\]

\[
T_{t_n}[x_2][y_3][z] = \left( T_{t_n}[x_2+1][y_3][z] + T_{t_n}[x_2][y_3+2^q][z] \right) / 2;
\]

\[
t_q = 0
\]

\[
T_{bd6}[index][2][z_1+k*(lz_6+1)] = T_{t_n}[x_1][y_3][z];
\]

\[
T_{bd6}[index][3][z_1+k*(lz_6+1)] = T_{t_n}[x_2][y_3][z];
\]

\[
T_{bd6}[index][0][z_1+k*(lz_6+1)] = T_{t_n}[x_2][y_3][z];
\]

\[
T_{bd6}[index][1][z_1+k*(lz_6+1)] = T_{t_n}[x_2][y_3][z];
\]

\[
T_{bd6}[index][2][z_1+k*(lz_6+1)] = T_{t_n}[x_2][y_3][z];
\]

\[
T_{bd6}[index][3][z_1+k*(lz_6+1)] = T_{t_n}[x_2][y_3][z];
\]

)// vertical border edges of level 6 and 5

for (z = z_3 + 1; z < z_4; z++)
{

\[
T_{t_n}[x_1][y_3][z] = \left( T_{t_n}[x_1-1][y_3][z] + T_{t_n}[x_1][y_3+2^q-1][z] \right) / 2;
\]

\[
T_{t_n}[x_2][y_3][z] = \left( T_{t_n}[x_2+1][y_3][z] + T_{t_n}[x_2][y_3+2^q-1][z] \right) / 2;
\]

\[
T_{t_n}[x_1][y_3][z] = \left( T_{t_n}[x_1-1][y_3][z] + T_{t_n}[x_1][y_3+2^q-1][z] \right) / 2;
\]

\[
T_{t_n}[x_2][y_3][z] = \left( T_{t_n}[x_2+1][y_3][z] + T_{t_n}[x_2][y_3+2^q-1][z] \right) / 2;
\]
if(q==0)
{
    Tbd6[index][2][z3-z1+k*(lz6+1)] = Tt_n1[x1][y3][z3];
    Tbd6[index][3][z3-z1+k*(lz6+1)] = Tt_n1[x2][y3][z3];
    Tbd6[index][2][z4-z1+k*(lz6+1)] = Tt_n1[x1][y3][z4];
    Tbd6[index][3][z4-z1+k*(lz6+1)] = Tt_n1[x2][y3][z4];
} else
{
    Tbd6[index][0][z3-z1+k*(lz6+1)] = Tt_n1[x1][y3][z3];
    Tbd6[index][1][z3-z1+k*(lz6+1)] = Tt_n1[x2][y3][z3];
    Tbd6[index][0][z4-z1+k*(lz6+1)] = Tt_n1[x1][y3][z4];
    Tbd6[index][1][z4-z1+k*(lz6+1)] = Tt_n1[x2][y3][z4];
}
k++;

//seventh level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
    for(j=0;j<2;j++)//repeat variable in y coordinate
        for(p=0;p<2;p++)//repeat variable in x coordinate
            for(q=0;q<2;q++)//repeat variable in inner y coordinate
                for(h=0;h<2;h++)//repeat variable in inner z coordinate
                {
                    x1 = cenX + (2*p-1)*lx4/2+(2*p-1)*lx5/2-lx7/2;
                    x2 = cenX + (2*p-1)*lx4/2+(2*p-1)*lx5/2+lx7/2;
                    x3 = cenX + (2*p-1)*lx4/2+(2*p-1)*lx5/2+lx7/2;
                    x4 = cenX + (2*p-1)*lx4/2+(2*p-1)*lx5/2+lx7/2;

                    y7 = cenY + (2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*j-1)*ly5/2+(2*j-1)*ly6/2+ly7/2;
                    y2 = cenY + (2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*j-1)*ly5/2+(2*j-1)*ly6/2+ly7/2;
                    z1 = cenZ + (2*r-1)*lz3/2+(2*r-1)*lz4/2+(2*r-1)*lz6/2+(2*r-1)*lz7/2;
                    z2 = cenZ + (2*r-1)*lz3/2+(2*r-1)*lz4/2+(2*r-1)*lz6/2+(2*r-1)*lz7/2;
                    z3 = cenZ + (2*r-1)*lz3/2+(2*r-1)*lz4/2+(2*r-1)*lz6/2+(2*r-1)*lz7/2;
                    z4 = cenZ + (2*r-1)*lz3/2+(2*r-1)*lz4/2+(2*r-1)*lz6/2+(2*r-1)*lz7/2;

                    for(y=y7+1;y<y2;y++)//x-y plane
                        for(x=x1+1;x<x2;x++)
                            Tt_n1[x][y][z4] = (Tt_n1[x][y][z3+1-2*h] + Tb7_n1[index][x-1+k*(lx7+1)]) * deltaZ*Bi) / (1+deltaZ*Bi);
                        for(y=y7+1;y<y2;y++)//x-y plane
                            for(x=x4+1;x<x2;x++)
                                Tt_n1[x][y][z3] = (Tt_n1[x][y][z3+1-2*h] + Tb7_n1[index][x-1+k*(lx7+1)]) * deltaZ*Bi) / (1+deltaZ*Bi);
\[(1 + \delta Z \cdot B_i) \]

\[
\frac{T_{nl}(x_i, y_j, z_k) =}{(1 + \delta Y \cdot B_i)}
\]

\[
\text{for}(z = z_l + 1; z < z_{l+1}; z++)
\]

\[
\text{for}(x = x_l + 1; x < x_{l+1}; x++)//x-z plane
\]

\[
\frac{T_{nl}(x_i, y_j, z_k) =}{(1 + \delta Y \cdot B_i)}
\]

\[
\text{for}(x = x_l + 1; x < x_{l+1}; x++)//x-z plane
\]

\[
\frac{T_{nl}(x_i, y_j, z_k) =}{(1 + \delta Y \cdot B_i)}
\]

\[
\text{for}(x = x_l + 1; x < x_{l+1}; x++)//x-z plane
\]

\[
\frac{T_{nl}(x_i, y_j, z_k) =}{(1 + \delta Y \cdot B_i)}
\]

\[
\text{for}(x = x_l + 1; x < x_{l+1}; x++)//x-z plane
\]

\[
\frac{T_{nl}(x_i, y_j, z_k) =}{(1 + \delta Y \cdot B_i)}
\]

\[
\text{for}(x = x_l + 1; x < x_{l+1}; x++)//x-z plane
\]
{ 
  \( T_{t_{nl}}[x][y][z3] = (\frac{T_{t_{nl}}[x][y+1][z3] + T_{t_{nl}}[x][y][z3+2\cdot h-1]}{2}; \)
  \( T_{t_{nl}}[x][y][z3] = (\frac{T_{t_{nl}}[x][y-1][z3] + T_{t_{nl}}[x][y][z3+2\cdot h-1]}{2}; \)
  if(h==0) 
    \( T_{bd7}[index][2][x\cdot x1+k\cdot (lx7+1)] = T_{t_{nl}}[x][y][z3]; \)
  else 
    \( T_{bd7}[index][0][x\cdot x1+k\cdot (lx7+1)] = T_{t_{nl}}[x][y][z3]; \)
  
} //horizontal y direction edges
for(y=y7+1;y<y2;y++) 
{
  \( T_{t_{nl}}[x1][y][z1] = (\frac{T_{t_{nl}}[x1+1][y][z1] + T_{t_{nl}}[x1][y][z1+1]}{2}; \)
  \( T_{t_{nl}}[x1][y][z2] = (\frac{T_{t_{nl}}[x1+1][y][z2] + T_{t_{nl}}[x1][y][z2+1]}{2}; \)
  \( T_{t_{nl}}[x2][y][z1] = (\frac{T_{t_{nl}}[x2-1][y][z1] + T_{t_{nl}}[x2][y][z1+1]}{2}; \)
  \( T_{t_{nl}}[x2][y][z2] = (\frac{T_{t_{nl}}[x2-1][y][z2] + T_{t_{nl}}[x2][y][z2+1]}{2}; \)
}
//vertex
\( T_{t_{nl}}[x1][y][z1] = (\frac{T_{t_{nl}}[x1+1][y][z1] + T_{t_{nl}}[x1][y+1][z1] + T_{t_{nl}}[x1][y][z1+1]}{3}; \)
for(y=y7+1;y<y2;y++) 

\[ T_{t,n}(x_3)[y][z_3] = \left( \frac{T_{t,n}(x_3-1)[y][z_3] + T_{t,n}(x_3)[y][z_3+1-2*h]}{2} \right); \]

\[ T_{t,n}(x_4)[y][z_3] = \left( \frac{T_{t,n}(x_4+1)[y][z_3] + T_{t,n}(x_4)[y][z_2+1-2*h]}{2} \right); \]

//horizontal x direction boarder edges of level 6 and 7

for(x=x3+1; x<x4; x++)
{
    \[ T_{t,n}(x)[y][z][z_3] = \left( \frac{T_{t,n}(x)[y][z_3]+2*h-1]}{2} \right); \]
    \[ T_{t,n}(x)[y][z_2]+2*h-1]}{2} \right); \]
    //if(h==0)
    \{ 
    Tbd7[index][2][x-x1]*k*(lx7+1) = T_{t,n}(x)[y][z3];
    Tbd7[index][3][x-x1]*k*(lx7+1) = T_{t,n}(x)[y][z3];
    \} 

else 
{
    \{ 
    Tbd7[index][0][x-x1]*k*(lx7+1) = T_{t,n}(x)[y][z3];
    Tbd7[index][1][x-x1]*k*(lx7+1) = T_{t,n}(x)[y][z3];
    \} 
    \} 

//vertix of boarder edges of level 7 and 6

\[ T_{t,n}(x_3)[y][z_3] = \left( \frac{T_{t,n}(x_3+1)[y][z_3]+T_{t,n}(x_3-1)[y][z_3]}{2} \right); \]

\[ T_{t,n}(x_3)[y][z_3] = \left( \frac{T_{t,n}(x_3+1)[y][z_3]+T_{t,n}(x_3-1)[y][z_3]}{2} \right); \]

\[ T_{t,n}(x_4)[y][z_3] = \left( \frac{T_{t,n}(x_4+1)[y][z_3]+T_{t,n}(x_4-1)[y][z_3]}{2} \right); \]

if(h==0)
{
    \{ 
    Tbd7[index][2][x-x1]*k*(lx7+1) = T_{t,n}(x)[y][z3];
    Tbd7[index][3][x-x1]*k*(lx7+1) = T_{t,n}(x)[y][z3];
    \} 

else 
{
    \} 

k++;

return;
BEGIN WRITESQUAREXY SUBROUTINE (OBTAINS ANY XY-PLANE DATA AT A POINT)********************************************************

void writeSquareXY(int x0, int y0, int x1, int y1, int z, int t, int i)
{
    char fname[256];
    sprintf(fname, "Z%d %d %d_%d_%d_t%d_l%d.txt", z, x0, y0, x1, y1, t, i);
    std::ofstream output(fname);
    for(i=x0;i<=x1;i++) {
        for(j=y0;j<=y1;j++) {
            output << Tt_nl[i][j][z] << " 
";
        }
    }
    return;
}
*************END OF WRITESQUAREXY SUBROUTINE********************************************************

BEGIN WRITESQUAREXZ SUBROUTINE (OBTAIN THE DATA FROM ANY XZ-PLANE FOR A GIVEN POINT Y)*************

void writeSquareXZ(int x0, int z0, int x1, int z1, int y, int t, int i)
{
    FILE *file;
    char fname[256], line[8196];
    sprintf(fname, "Y%d %d %d %d %d_t%d_l%d.txt", y, x0, z0, x1, z1, t, i);
    file = fopen(fname, "w");
    for(int k=z0;k<=z1;k++)
    {
        strcpy(line, "\n");
        for(int i=x0;i<=x1;i++)
        {
            sprintf(tmp, "%.10f", Tt_nl[i][y][k]);
            strcat(line, tmp);
        }
        fwrite(line,strlen(line), 1, file);
    }
    fclose(file);
}
*************END OF WRITESQUAREXZ SUBROUTINE*************

BEGIN WRITESQUAREYZ SUBROUTINE (OBTAIN THE DATA FOR A YZ-PLANE FOR ANY GIVEN POINT X)*************

void writeSquareYZ(int y0, int z0, int y1, int z1, int x, int t, int i)
{
    FILE *file;
    char fname[256], line[8196];
    sprintf(fname, "X%d %d %d %d %d_t%d_l%d.txt", x, y0, z0, y1, z1, t, i);
    file = fopen(fname, "w");
    for(int k=z0;k<=z1;k++)
    {
        sprintf(tmp, "%.10f", Tt_nl[i][y][k]);
        strcat(line, tmp);
        for(int i=x0;i<=x1;i++)
        {
            sprintf(tmp, "%.10f", Tt_nl[i][y][k]);
            strcat(line, tmp);
        }
        fwrite(line,strlen(line), 1, file);
    }
    fclose(file);
}
{  
    strcpy(line, """);
    for(int j=y0;j<=y1;j++)  
    {  
        sprintf(tmp, "%10.6f", Tt_n1[x][j][k]);  
        strcat(line, tmp);
    }
    strcat(line, "n");
    fwrite(line,strlen(line), 1, file);
}
fclose(file);

**************************************************END OF
WRITESQUAREYZ SUBROUTINE*******************************
**************************************************START OF
WRITECENTER SUBROUTINE (FOR A GIVEN TIME, OBTAIN THE TEMPERATURES
THROUGH THE CENTER OF THE MODEL)****************

void writeZCenter(int t)  
{  
    char fname[256];
    sprintf(fname, "Z_Center_t%d.txt", t);
    std::ofstream output(fname);
    for(int j=0;j<=NZ3;j++)  
    {  
        output << j << " " << Tt_n1[centerX][centerY][j] << std::endl;
    }
}

**************************************************END OF
WRITEALL SUBROUTINE*******************************
**************************************************BEGIN
WRITELINEARSYS SUBROUTINE (OBTAIN THE TRIDIAGONAL SYSTEM FOR A GIVEN
COORDINATE OF X AND Y AT A GIVEN TIME AND ITERATION)

void writeLinearSys(int x, int y, int t, int I)  
{  
    FILE *file;
    char fname[256], line[256];
    //strcpy(fname, outPath);
    sprintf(fname, "X%d_Y%d_t%d.txt", x, y, t, I);
    file = fopen(fname, "w");
    for(int k=0;k<=NZ3;k++)  
    {  
        sprintf(line, "%d\t%10.6f\t%10.6f\t%10.6f\n", k, 
            b[x][y][k],a[x][y][k],c[x][y][k],d[x][y][k]);
        fwrite(line,strlen(line), 1, file);
    }
}
fclose(file);

}*

*****************************************************************************
WRITELINEARSYS SUBROUTINE**************
*****************************************************************************
WRITELOG SUBROUTINE (WRITES THE LOG OF ACTIVITY THAT IS GIVEN AS AN INPUT)****************************************
*****************************************************************************

void writeLog(char *in)
{
    FILE *file;
    char fname[256], line[512];
    //strcpy(fname, outPath);
    strcpy(fname, "log.txt");
    file = fopen(fname, "a");
    strcpy(line, in);
    strcat(line, ":
    fwrite(line, strlen(line), 1, file);
    fclose(file);

}*****************************************************************************
WRITELOG SUBROUTINE*********************
*****************************************************************************
WRITEBLOOD SUBROUTINE*******************
*****************************************************************************

void writeblood(int i, double x)
{
    FILE *file;
    char fname[256], line[256];
    //strcpy(fname, outPath);
    sprintf(fname, "bloodTep.txt");
    file = fopen(fname, "a");
    sprintf(line, "%d\t%6f\n", i, x);
    fwrite(line, strlen(line), 1, file);
    fclose(file);

}*****************************************************************************
WRITEBLOOD SUBROUTINE******************
*****************************************************************************

// Initialize the laser power
void InitQ(double P0)
{
    double qc; // collimated heat source
    double qd; // Diffusive heat source
    double qn; // Nanoparticle heat source
    double laser; // laser heat source
    int i,j,z;
    double shift;
    double Flux = (P0/( sqrt(2.0*pi)*Sigma2))*exp(-
    Alpha1*NZ1*deltaZ-Alpha2*( NZ2-NZ1)*deltaZ); // heat flux for nanoparticle spectrum
    double TotOptL = TumExt*LT*deltaZ; // Total Optical Length of Tumor
    double Casel = sqrt(3.0*( 1.0-Albedo));
    double B1 = -(Albedo*Flux)/( 1.0-(Casel*Casel));
    double B2 = 2.0 - ( 1.0/((2.0-1.0)*(1.0-
    Albedo))*Casel);
    double B3 = 2.0 + ( 1.0/((2.0-1.0)*( 1.0-Albedo))*Casel);
    double B4 = -(2.0 + ( 1.0/(2.0-1.0)*1.0-Albedo)*Casel);
    double B5 = -(Albedo/( 1.0-Albedo))*Flux;
    double B6 = -(2.0*exp(Casel*TotOptL)+(1.0/(1.0-
    Albedo))*Casel*exp(Casel*TotOptL));
    double B7 = -2.0*exp(-Casel*TotOptL)+(LO/(1.0-
    Albedo))*Casel*exp(-Casel*TotOptL);
\[ B_8 = 2.0 \exp(-\text{TotOptL}) \cdot (1.0 - \text{Albedo}) \exp(-\text{TotOptL}) \]

\[ B_9 = -(\text{Albedo} / (1.0 - \text{Albedo})) \cdot \text{Flux} \exp(-\text{TotOptL}); \]

// Matrix Creation

```
std::cout << "About to create matrix" << std::endl;
Research::Matrix Result(2, 1);
Research::Matrix Mat1(2, 2);
Research::Matrix Mat2(2, 1);
std::cout << "Created them" << std::endl;
Mat1(0, 0) = B2;
Mat1(0, 1) = B3;
Mat1(1, 0) = B6;
Mat1(1, 1) = B7;
Mat2(0, 0) = B4 \cdot B1 + B5;
Mat2(1, 0) = B8 \cdot B1 + B9;
Result = Mat1.inverse() \cdot Mat2;
double Con1 = Result(0, 0);
double Con2 = Result(1, 0);
for (i = 0; i <= NX; i++) {
  for (j = 0; j <= NY; j++) {
    shift = \exp(-((double)((i-centerX) * (i-centerX) * deltaX * deltaX + (double)((j-centerY) * (j-centerY) * deltaY * deltaY)) / (2.0 * Sigma2 * Sigma2)) / (sqrt(2.0 * pai) * Sigma2);
    for (z = 0; z <= NZ1; z++) {
      Q1[i][j][z] = Alpha1 * \exp(-Alpha1 * z * deltaZ) \cdot \text{shift} \cdot P0 * (1.0 - \text{Refl1});
    }
  }
}
```
INTENSITY BASED ON THE LEAST SQUARES METHOD)

double CalcNewP(double *Tm_pre, double *Tm0, double *Tm1, double P0, double deltaP)
{
    int i;
    double P1;
    double alphaStar = 0;//????????
    double X[M];
    for(i=0;i<M;i++)
        X[i] = (Tm1[i] - Tm0[i]) / deltaP;
    double factor = 0;
    for(i=0;i<M;i++)
        factor += X[i]*X[i];
    factor += alphaStar;
    double temp=0;
    for(i=0;i<M;i++)
        temp += X[i] * (Tm_pre[i]-Tm1[i]);
    P1 = P0 + 1.0/factor * temp;
    return P1;
}

**************************END CALCNEWP
SUBROUTINE**************************

SUBROUTINE (OBTAINS TEMPERATURES USED IN POINTS FOR THE LEAST SQUARES METHOD CALCULATION)

int getTm(double *Tm)
{
    //assign surface temperature to Tm array here
    //for M==6 right now
    Tm[0] = T[NX/2][NY/2][centerTZ];
    //Center of Tumour

    Tm[1] = T[NX/2][0][0]; //side of model in Y-direction
    Tm[2] = T[NX/2][NY][0]; //Other side of model in Y-direction
    Tm[3] = T[0][NY/2][0]; //Side of model in X-direction
    Tm[4] = T[NX][NY/2][0]; //Side of model in other X-direction
    //Tm[5] = T[NX/2][NY/2][0]; //Center of Epidermis top.
    return(1);
}

SUBROUTINE*****************GETTM
SUBROUTINE**************************

ADJUSTPOWER SUBROUTINE (DETERMINES IF THE LASER NEEDS TO BE TURNED ON OR OFF BASED ON THE TEMPERATURE AT THE CENTER OF THE TUMOR)

void AdjustPower(double P0)
{
    if(bPowerOn)
    {
        //power is on now
        if((T1_n1[NX/2][NY/2][centerTZ] >= Tm_pre[0]))
        {
            //set power off
            memset(Q1, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            memset(Q2, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            memset(Q3, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            bPowerOn = false;
        }
    }
}
//writeLog("======Power Off======");

else {
    //power is off now
    if(Tt_n1[NX/2][NY/2][centerTZ] <= Tm_pre[0]-4 )
    {
        //set power on
        InitQ(P0);
        bPowerOn = true;
        sprintf(tmp, "======Power On:%7.4lf", P0);
        //writeLog(tmp);
    }
    return;
}

END OF
ADJUSTPOWER SUBROUTINE***************

%%%%%%%%%%%%%%%%%%%%%%%%BEGIN
WRITEQCENTER SUBROUTINE (OBTAIN THE HEAT SOURCE CALCULATIONS ALONG THE CENTER OF THE MODEL AT A GIVEN TIME)

void writeQCenter(int t)
{
    char fname[256];
    sprintf(fname, "Q_Center_t%d.txt", t);
    std::ofstream output(fname);
    for(int j=0;j<=NZ3;j++) {
        if(j<=NZ1) {
            output << j << " * * *
            Q1[centerX][centerY][j] << std::endl;
            } else if(j<=NZ2) {
                output << j << " * * *
                Q2[centerX][centerY][j] << std::endl;
                } else {
                    output << j << " * * *
                    Q3[centerX][centerY][j] << std::endl;
                    } } } 

**************************END OF
ADJUSTPOWER SUBROUTINE***************
/* Author: Stanislav Ponomarev
 * Date changed: Jan 5, 2015
 * Filename: Matrix.cpp
 *
 * Matrix math implementation
 *
 */
#include "Matrix.h"
#include <sstream>
#include <algorithm>
#include <iostream>

namespace Research {
  Matrix::Matrix(unsigned int rows, unsigned int cols): cols(cols), rows(rows), isUsingOneOffset(false) {
    if (!(rows && cols)) { // if rows or cols are zero
      std::stringstream ss;
      ss << "Matrix Constructor has zero size (" << rows << ", " << cols << ");"
      throw std::length_error(ss.str());
    }
    data = new double[rows*cols];
  }
  Matrix::Matrix( Matrix const &m) {
    cols = m.cols;
    rows = m.rows;
    isUsingOneOffset = m.isUsingOneOffset;
    data = new double[rows*cols];
    std::copy(m.data, m.data+rows*cols, data);
  }
  inline Matrix::~Matrix() {
    delete[] data;
  }
  Matrix& Matrix::operator=( Matrix const &m) {
    if (data)
      delete[] data;
    cols = m.cols;
    rows = m.rows;
    isUsingOneOffset = m.isUsingOneOffset;
    data = new double[rows*cols];
    std::copy(m.data, m.data+rows*cols, data);
    return *this;
  }
  Matrix& Matrix::operator=(double const a[]) {
    std::copy(a, a+rows*cols, data);
    return *this;
  }
  double& Matrix::operator() (unsigned int row, unsigned int col) {
    if (isUsingOneOffset) {
      row--;
      col--;
    }
    if (row > rows || col > cols) {
      throw std::out_of_range("Matrix size is smaller than index");
    }
    return data[col+row*cols];
  }
  double Matrix::operator() (unsigned int row, unsigned int col) const {
    if (isUsingOneOffset) {
      row--;
      col--;
    }
    if (row > rows || col > cols) {
      throw std::out_of_range("Matrix size is smaller than index");
    }
    return data[col+row*cols];
  }
}
if (isUsingOneOffset) {
    row --;
    col --;
}

if (row > rows || col > cols) {
    throw std::out_of_range("Matrix size is smaller than index");
}

return data[col + row * cols];

Matrix Matrix::operator+ (Matrix const &m) const {
    if (rows != m.rows || cols != m.cols)
    {
        throw std::out_of_range("Matrix addition requires same dimensions");
    }
    unsigned int row, col;
    Matrix result(*this);
    for (row = 0; row < rows; row++)
        for (col = 0; col < cols; col++)
        {
            result(row, col) = data[col + row * cols] + m.data[col + row * cols];
        }
    return result;
}

Matrix Matrix::operator- (Matrix const &m) const {
    if (rows != m.rows || cols != m.cols)
    {
        throw std::out_of_range("Matrix addition requires same dimensions");
    }
    unsigned int row, col;
    Matrix result(*this);
    for (row = 0; row < rows; row++)
        for (col = 0; col < cols; col++)
        {
            result(row, col) = data[col + row * cols] - m.data[col + row * cols];
        }
    return result;
}

Matrix Matrix::operator* (int const &s) const {
    unsigned int row, col;
    Matrix result(*this);
    for (row = 0; row < rows; ++row)
        for (col = 0; col < cols; ++col)
        {
            result(row, col) = data[col + row * cols] * s;
        }
    return result;
}

Matrix Matrix::operator* (float const &s) const {
    unsigned int row, col;
    Matrix result(*this);
    for (row = 0; row < rows; ++row)
        for (col = 0; col < cols; ++col)
        {
            result(row, col) = data[col + row * cols] * s;
        }
    return result;
}
Matrix Matrix::operator* (double const &s) const {
    unsigned int row, col;
    Matrix result(*this);
    for (row=0; row<rows; ++row)
        for (col=0; col<cols; ++col) {
            result(row, col) =
                data[col+row*cols]*s;
        }
    return result;
}

Matrix Matrix::operator* (Matrix const &m) const {
    if (cols != m.rows) {
        throw std::out_of_range("Matrices have unmultipliable dimensions");
    }
    Matrix result(rows, m.cols);
    unsigned int row, col, k;
    for (row=0; row<rows; ++row)
        for (col=0; col<m.cols; ++col) {
            double value = 0.0;
            for (k=0; k<cols; ++k)
                value +=
                    data[k+row*cols]*m.data[col+k*m.cols];
            result(row, col) =
                value;
        }
    return result;
}

Matrix Matrix::inverse() {
    if (rows != cols) {
        throw std::domain_error("Determinant of non-square matrix does not exist");
    }
    double result;
    if (cols == 2) {
        result = data[0]*data[3] -
                data[1]*data[2];
        return result;
    }
    int col, srows, scols, srow, scol;
    srows = rows-1;
    scols = cols-1;
    for (col=0; col<cols; ++col) {
        double submatrix_init[srows*scols];
        for (srow=0; srow<srows; ++srow)
            for (scol=0; scol<scols; ++scol) {
                submatrix_init[scol+srow*scols] =
                    data[(scol < col ? scol : scol+1)+(srow+1)*scols];
            }
        Matrix submatrix(srows, scols);
        submatrix =
            submatrix_init;
        result += (col % 2 ? -1.0 : 1.0)*data[col]*submatrix.determinant();
    }
    return result;
}

double Matrix::determinant() {
    if (rows != cols) {
        throw std::domain_error("Determinant of non-square matrix does not exist");
    }
    double result;
    if (cols == 2) {
        result = data[0]*data[3] -
                data[1]*data[2];
        return result;
    }
    int col, srows, scols, srow, scol;
    srows = rows-1;
    scols = cols-1;
    for (col=0; col<cols; ++col) {
        double submatrix_init[srows*scols];
        for (srow=0; srow<srows; ++srow)
            for (scol=0; scol<scols; ++scol) {
                submatrix_init[scol+srow*scols] =
                    data[(scol < col ? scol : scol+1)+(srow+1)*scols];
            }
        Matrix submatrix(srows, scols);
        submatrix =
            submatrix_init;
        result += (col % 2 ? -1.0 : 1.0)*data[col]*submatrix.determinant();
    }
    return result;
}

Matrix Matrix::inverse() {
throw std::domain_error("Inverse of non-square matrix does not exist");

if (rows == 2) {
    Matrix tmp(2, 2);
    //col+row*cols
    double tmp_init[4] = {data[3], -data[1], -data[2], data[0]};
    tmp = tmp_init;
    return tmp*(1.0/determinant());
}

return adjoint()*(1.0/determinant());

Matrix Matrix::cofactor() {
    if (rows != cols) {
        throw std::domain_error("Cofactor of non-square matrix does not exist");
    }
    Matrix result(rows, cols);
    int row, col, srow, scol, srows, scols;
    srows = rows-1;
    scols = cols-1;
    std::cout << "srows = " << srows << " scols = " << scols << std::endl;
    for (row = 0; row < rows; ++row) {
        for (col = 0; col < cols; ++col) {
            for (srow=0; srow < srows; ++srow)
                submatrix_init[srows*scols];
            for (scol=0; scol<scols; ++scol)
                submatrix_init[scol+srow*scols] = data[(scol < col ? scol : scol+1)+(srow < row ? srow : srow+1)*cols];
            Matrix tmp(srows, scols);
            submatrix_init;
            result(row, col) = ((row+col)%2 ? -1.0 : 1.0)*tmp.determinant();
            return result;
        }
    }
    Matrix result(cols, rows);
    int col, row;
    for (col = 0; col<cols; col++)
        for (row = 0; row < rows; row++) {
            if (isUsingOneOffset)
                return cofactor().transpose();
        }
    inline unsigned int Matrix::getRows() const
    { return isUsingOneOffset ? rows+1 : rows; }
    inline unsigned int Matrix::getCols() const {
return (isUsingOneOffset ? cols+1 : cols);
            return os;
        }
    }
    void Matrix::setUsingOneOffset() {
        isUsingOneOffset = true;
    }
    void Matrix::setUsingZeroOffset() {
        isUsingOneOffset = false;
    }
    std::ostream& operator<<(std::ostream& os, const Matrix& obj) {
        unsigned int row, col;
        for (row=0; row<obj.getRows(); ++row) {
            if (row == 0)
                os << "/";
            else if (row == obj.getRows()-1)
                os << "\";
            else
                os << "|";
            for (col=0; col<obj.getCols(); ++col) {
                os << obj(row, col) << (col != obj.getCols()-1 ? "," : "");
            }
            if (row == 0)
                os << "/";
            else if (row == obj.getRows()-1)
                os << "\";
            else
                os << "|";
            os << std::endl;
        }
    }
BIBLIOGRAPHY


