A numerical method for obtaining an optimal temperature distribution in a three-dimensional triple-layered skin structure embedded with multi-level blood vessels

Xingui Tang
Louisiana Tech University

Follow this and additional works at: https://digitalcommons.latech.edu/dissertations
Part of the Applied Mathematics Commons, and the Mathematics Commons

Recommended Citation
https://digitalcommons.latech.edu/dissertations/567

This Dissertation is brought to you for free and open access by the Graduate School at Louisiana Tech Digital Commons. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of Louisiana Tech Digital Commons. For more information, please contact digitalcommons@latech.edu.
A NUMERICAL METHOD FOR OBTAINING AN OPTIMAL TEMPERATURE DISTRIBUTION IN A 3D TRIPLE-LAYERED SKIN STRUCTURE EMBEDDED WITH MULTI-LEVEL BLOOD VESSELS

by

Xingui Tang, 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

May 2006
INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.
We hereby recommend that the dissertation prepared under our supervision by XINGUI TANG
entitled A NUMERICAL METHOD FOR OBTAINING AN OPTIMAL TEMPERATURE DISTRIBUTION IN A 3D TRIPLE-LAYERED SKIN STRUCTURE EMBEDDED WITH MULTI-LEVEL BLOOD VESSELS
be accepted in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY IN COMPUTATIONAL ANALYSIS AND MODELING.

Richard Sheehy
Supervisor of Dissertation Research

CAM
Head of Department

Recommendation concurred in:

Adri Pater

Kaja Hassner

Advisory Committee

Approved:

Ratha Penumadhan
Director of Graduate Studies

Dean of the College

Approved:

Terry McLoatchy
Dean of the Graduate School

GS Form 13 (5/03)
ABSTRACT

The research related to hyperthermia has stimulated a lot of interest in recent years because of its application in cancer treatment. When heating the tumor tissue, the crucial problem is keeping the temperature of the surrounding normal tissue below a certain threshold in order to avoid the damage to the normal tissue. Hence, it is important to obtain the temperature field of the entire region during the treatment. The objective of this dissertation is to develop a numerical method for obtaining an optimal temperature distribution in a 3D triple-layered skin structure embedded with multi-level blood vessels where the surface of the skin is irradiated by laser. The skin structure is composed of epidermis, dermis and subcutaneous, while the dimension and blood flow of the multi-level blood vessels are determined based on the constructal theory of multi-scale tree-shaped heat exchangers. The method determines the optimal laser intensity to obtain pre-specified temperatures at the given locations of the skin after a pre-specified laser exposure time under a pre-specified laser irradiation pattern.

The modified Pennes bio-heat transfer model is employed to describe the thermal behavior for tissue coupled with the convective energy balance equations for blood. The finite difference schemes for solving these equations are developed and the least squares method is used to optimize the laser power. As such, we develop an algorithm which can be used to obtain an optimal temperature distribution. Furthermore, the preconditioned Richardson iteration and Thomas algorithm are employed to speed up and simplify the
computation. To demonstrate the applicability of the mathematical model and the numerical method, we test on three examples in each of which two cases are considered. The numerical examples show that the method is applicable and efficient.

This research is important since the results will provide the clinician with powerful tools to improve the ability to deliver safe and effective therapy and the means to assess treatment safety, efficacy, and clinical outcome for skin, head, and neck cancer treatments.
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  xiangui tan

Date  5/10/06
DEDICATION

I dedicate this work to my parents, my sister and brother, my husband Zongwen and my son Bill.
# TABLE OF CONTENTS

ABSTRACT ................................................................................................................................. iii
DEDICATION ............................................................................................................................. vi
TABLE OF CONTENTS ........................................................................................................... vii
LIST OF TABLES .................................................................................................................... x
LIST OF FIGURES .................................................................................................................. xi
ACKNOWLEDGEMENTS ........................................................................................................... xiv
CHAPTER ONE  INTRODUCTION ............................................................................................ 1
  1.1 General Overview ............................................................................................................ 1
  1.2 Research Objective ........................................................................................................ 3
  1.3 Organization of this Dissertation ................................................................................... 3
CHAPTER TWO  LITERATURE REVIEW ............................................................................... 5
  2.1 Inverse Heat Conduction Problem ................................................................................. 5
    2.1.1 Introduction .......................................................................................................... 5
    2.1.2 Difficulties and Solutions to the Inverse Problem .................................................. 6
    2.1.3 Least Squares Approach ....................................................................................... 7
  2.2 Preconditioned Richardson Iteration Method ................................................................. 12
  2.3 Thomas Algorithm ......................................................................................................... 14
  2.4 Constructal Theory of Multi-scale Tree-shaped Heat Exchangers ............................... 16
  2.5 Previous Work on Bio-heat Transfer ............................................................................ 18

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
5.3.1.2 Case 2 ................................................................. 53
5.3.2 Example 2 .............................................................. 59
5.3.2.1 Case 1 ................................................................. 59
5.3.2.2 Case 2 ................................................................. 71
5.3.3 Example 3 .............................................................. 77
5.3.3.1 Case 1 ................................................................. 77
5.3.3.2 Case 2 ................................................................. 89
5.4 Checking the Grid Independence .................................. 93

CHAPTER SIX CONCLUSION AND FUTURE STUDIES ................. 95
6.1 Conclusion .............................................................. 95
6.2 Future Studies .......................................................... 96

APPENDIX A SOURCE CODE FOR EXAMPLE 1 ......................... 98
APPENDIX B SOURCE CODE FOR EXAMPLE 2 AND 3 .......... 128
APPENDIX C NOMENCLATURE .......................................... 165
REFERENCES ............................................................... 168
LIST OF TABLES

Table 5.1 Thermal properties of the skin and the blood.................................43

Table 5.2 Geometry parameters for the skin structure and the embedded blood vessels..45
LIST OF FIGURES

Figure 2.1  The construction of the tree of convective heat currents, (a) the constrained optimization of the geometry of a T-shaped construct; (b) the stretched tree of optimized constructs; (c) the superposition of two identical trees oriented in counter flow; and (d) the convective heat flow along a pair of tubes in counter flow.................................................................16

Figure 3.1  Skin structure and its components.................................................................23

Figure 3.2  Configuration of a 3D skin structure embedded with a multi-level artery...25

Figure 3.3  Configuration of a 3D skin structure embedded with a multi-level artery and a multi-level vein.................................................................25

Figure 5.1  Geometry of a 3D skin structure embedded with a multi-level artery........42

Figure 5.2  Geometry of a 3D skin structure embedded with a multi-level artery and a multi-level vein.................................................................42

Figure 5.3  Laser intensity and sum of the least squares versus number of iterations for the case 1 of example 1.................................................................47

Figure 5.4  Profiles of temperature elevations at various times in the x-direction at y = 0.5 cm on the skin surface for the case 1 of example 1.................48

Figure 5.5  Profiles of temperature elevations at various times in the y-direction at x = 0.5 cm on the skin surface for the case 1 of example 1.................49

Figure 5.6  Contours of the temperature elevations at various times in the cross section of the xz-plane at y = 0.5 cm for the case 1 of example 1............50

Figure 5.7  Contours of the temperature elevations at various times in the cross section of the yz-plane at x = 0.5 cm for the case 1 of example 1............51

Figure 5.8  Profiles of temperature elevations at various times in the z-direction (depth) at the center of the skin surface for the case 1 of example 1............52
Figure 5.9 Profiles of temperature elevations and contours at \( t = 400 \) s and \( P_0 = 1.1947 \text{W/cm} \) for the case 2 of example 1..........................55

Figure 5.10 Laser intensity and sum of the least squares versus number of iterations for the case 2 of example 1..........................56

Figure 5.11 Profiles of temperature elevations and contours at \( t = 400 \) s and \( P_0 = 1.3166 \text{W/cm} \) for the case 2 of example 1..........................58

Figure 5.12 Profiles of temperature elevations and contours at \( t = 400 \) s and \( P_0 = 1.1947 \text{W/cm} \) for the case 1 of example 2..........................62

Figure 5.13 Laser intensity and sum of the least squares versus number of iterations for the case 1 of example 2..........................63

Figure 5.14 Profiles of temperature elevations at various times in the x-direction at \( y = 0.5 \text{cm} \) on the skin surface for the case 1 of example 2...............64

Figure 5.15 Profiles of temperature elevations at various times in the y-direction at \( x = 0.5 \text{cm} \) on the skin surface for the case 1 of example 2...............65

Figure 5.16 Contours of the temperature elevations at various times in the cross section of the xz-plane at \( y = 0.4 \text{cm} \) for the case 1 of example 2............66

Figure 5.17 Contours of the temperature elevations at various times in the cross section of the xz-plane at \( y = 0.56 \text{cm} \) for the case 1 of example 2............67

Figure 5.18 Contours of the temperature elevations at various times in the cross section of the xz-plane at \( y = 0.5 \text{cm} \) for the case 1 of example 2............68

Figure 5.19 Contours of the temperature elevations at various times in the cross section of the yz-plane at \( x = 0.5 \text{cm} \) for the case 1 of example 2............69

Figure 5.20 Profiles of temperature elevations at various times along the depth (the z-direction) at three locations for the case 1 of example 2............70

Figure 5.21 Profiles of temperature elevations and contours at \( t = 400 \) s and \( P_0 = 1.1993 \text{W/cm} \) for the case 2 of example 2..................73

Figure 5.22 Laser intensity and sum of the least squares versus number of iterations for the case 2 of example 2..................74

Figure 5.23 Profiles of temperature elevations and contours at \( t = 400 \) s and \( P_0 = 1.3450 \text{W/cm} \) for the case 2 of example 2..................76
Figure 5.24 Profiles of temperature elevations and contours at \( t = 400 \) s and \( P_0 = 1.1993 \) W/cm for the case 1 of example 3..........................80

Figure 5.25 Laser intensity and sum of the least squares versus number of iterations for the case 1 of example 3..............................81

Figure 5.26 Profiles of temperature elevations at various times in the x-direction at \( y = 0.5 \) cm on the skin surface for the case 1 of example 3......................82

Figure 5.27 Profiles of temperature elevations at various times in the y-direction at \( x = 0.5 \) cm on the skin surface for the case 1 of example 3....................83

Figure 5.28 Contours of the temperature elevations at various times in the cross section of the xz-plane at \( y = 0.4 \) cm for the case 1 of example 3.............84

Figure 5.29 Contours of the temperature elevations at various times in the cross section of the xz-plane at \( y = 0.56 \) cm for the case 1 of example 3..............85

Figure 5.30 Contours of the temperature elevations at various times in the cross section of the xz-plane at \( y = 0.5 \) cm for the case 1 of example 3..............86

Figure 5.31 Contours of the temperature elevations at various times in the cross section of the yz-plane at \( x = 0.5 \) cm for the case 1 of example 3........7

Figure 5.32 Profiles of temperature elevations at various times along the depth (the z-direction) at three locations for the case 1 of example 3.................................88

Figure 5.33 Laser intensity and sum of the least squares versus number of iterations for the case 2 of example 3.........................................................90

Figure 5.34 Profiles of temperature elevations and contours at \( t = 400 \) s and \( P_0 = 1.5552 \) W/cm for the case 2 of example 3.................................92

Figure 5.35 Four profiles of temperature elevations at \( t = 400 \) s for case 1 of example 1 (a) in the y-direction at \( x = 0.5 \) cm and (b) in the x-direction at \( y = 0.5 \) cm on the skin surface, and (c) in the z-direction (along the depth) at the center of the skin surface.................................94
ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere appreciation to my advisor, Dr. Weizhong Dai, for his invaluable guidance, encouragement, and generous support during the course of this work. I am also very grateful for Dr. Raja Nassar, Dr. Dick Greechie, Dr. Walter G. Besio and Dr. Andrei Paun, who graciously agreed to sacrifice their valuable time and energy to serve on my advisory committee and to offer me precious suggestion and help. I would like to thank all the faculty and staff of the College of Engineering and Science for being supportive during the various stages of my study and research at Louisiana Tech University. Thanks to all the friends who ever helped me.

Finally, I want to express my special and greatest appreciation to my husband, Zongwen Feng, for his support and help technically and spiritually all the time.
CHAPTER ONE

INTRODUCTION

1.1 General Overview

Hyperthermia, also called thermal therapy or thermotherapy, is one type of cancer treatment in which the temperature of the tissue is elevated artificially to 42-46 °C with the aim of receiving therapeutic benefits. Hyperthermia is almost always used with other forms of cancer therapy, such as radiation therapy and chemotherapy [Moroz 2002] [Muralidharan 2002] [Tsuda 1996] [Usatoff 2001] [Wust 2002] [Zee 2002]. It may make some cancer cells more sensitive to radiation or harm other cancer cells that radiation cannot damage. Hyperthermia can also enhance the effects of certain anticancer drugs. Lots of clinical trials have studied hyperthermia in combination with radiation therapy and/or chemotherapy. These studies have focused on the treatment of many types of cancer, such as cancers of the head and neck, brain, skin, breast, bladder, lung, liver and advanced kidney [Wust 2002] [Hall 1984] [Streffer 1981] [Falk 2001] [Zee 2002], and the results have shown that hyperthermia has an increased effectiveness in the treatment [Wust 2002], [Zee 2002].

Hyperthermia is usually classified as local, regional, and whole body hyperthermia [Wust 2002] [Falk 2001] [Feldman 2003] [Chang 2001]. In local hyperthermia, heat is applied to a small area, such as a tumor, using various techniques to deliver energy to heat the tumor. Different types of energy may be used to apply heat
including heating rods, microwaves, radiofrequencies, ultrasound, thermal blankets and lasers. Depending on the tumor location, there are different approaches to local hyperthermia. External approaches are used to treat tumors that are in or just below the skin. External approach is particularly suitable in treating small superficial tumors (within 7 cm under the surface), where applicators are positioned around or near the appropriate region, and energy is focused on the tumor to raise its temperature.

However, while heating the tumor region, the heat energy will spread around to raise the temperature of the surrounding normal tissue as well. Thus, it is a challenge to control the heat energy so that the temperature of the normal tissue surrounding the tumor remains low enough so as not to cause damage to the tissue. Hence, for process control, it is important to obtain a temperature field of the entire treatment region. With knowledge of the entire temperature field in the treatment region, clinical personnel can potentially control the heating source to deliver energy to the treatment target volume to raise its minimum temperature above 42 °C, while limiting the temperatures in the normal tissue to prevent damage. However, the desired and accurate determination of the temperature field over the entire treatment region is difficult to obtain during clinical hyperthermia treatments. Physically, the doctor inserts small needles or tubes with tiny thermometers into the treatment area to monitor the temperature. But the number of invasive temperature probes that can be used is limited due to the tolerance of pain by the patients. Hence, it is important to use the computer modeling and mathematical method to direct and control the heat source in order to optimize the temperature distribution in the treated region. This would involve numerical methods to solve the bio-heat transfer equation for the human body [Chatterjee 1994].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
1.2 Research Objective

In this dissertation, we will develop a numerical method for obtaining an optimal temperature distribution in a 3D triple-layered skin structure embedded with multi-level blood vessels. The work will be done first considering only a multi-level artery and then both a multi-level artery and a multi-level vein embedded in the skin structure. Moreover, we will also consider the cases that the blood is in the steady state and in the dynamic state. The method determines the required laser intensity to obtain pre-specified temperatures at the given locations of the skin after a pre-specified laser exposure time under a pre-specified laser irradiation pattern. To achieve this objective, the following development is pursued:

1. Present a mathematical model to describe the bio-heat transfer in the 3D triple-layered skin structure embedded with multi-level blood vessels.
2. Develop accurate finite difference schemes for solving the equations in the mathematical model.
3. Solve the finite difference scheme by using an iteration method.
4. Optimize the laser intensity by using the inverse heat conduction method.

This research is important since the results will provide the clinician with powerful tools to improve the ability to deliver safe and effective therapy and the means to assess treatment safety, efficacy, and clinical outcome for skin, head, and neck cancer treatments.

1.3 Organization of this Dissertation

This chapter gives a general introduction to hyperthermia and presents the research objective of this dissertation.
Chapter 2 will introduce the inverse heat conduction problem, preconditioned Richardson iteration method, Thomas algorithm, and the constructal theory of multi-scale tree-shaped heat exchangers followed by a review of the previous work on bio-heat transfer.

Chapter 3 will introduce the structure of the blood vessels based on the constructal theory, and present a mathematical model which is used to describe the bio-heat transfer in the 3D triple-layered skin structure embedded with multi-level blood vessels.

Chapter 4 will develop the numerical methods for solving the governing equations presented in Chapter 3. The inverse heat conduction method will be employed to optimize the laser intensity, and then a preconditioned Richardson iteration will be developed to speed up and simplify the computation.

Chapter 5 will work on three examples to test the mathematical model and the numerical methods. Numerical results will be shown and analyzed in this chapter.

Finally, the conclusion and future work will be discussed in Chapter 6.
CHAPTER TWO

LITERATURE REVIEW

2.1 Inverse Heat Conduction Problem

Inverse heat conduction problems (IHCPs) have a lot of important applications in various fields of engineering and sciences. Mechanical, aerospace and chemical engineers, mathematicians, astrophysicists, geophysicists, statisticians and specialists of many other disciplines are all interested in inverse problems, not only because of their difficulties, but also because they have unsubstitutable roles in numerous practical applications.

2.1.1 Introduction

Inverse heat conduction problems are different from the standard heat conduction problems in that the inverse problems are concerned with the determination of boundary condition, energy-generation rate, or thermophysical properties by utilizing the measured temperature history at one or more locations in the solid, whereas the standard problems are concerned with the determination of temperature distribution in the interior of the solid when the boundary and initial conditions, the energy generation rate and the thermophysical properties of the medium are specified [Ozisik 1993]. In the heat transfer field, the inverse analysis can be used to estimate surface conditions such as thermal conductivity and heat capacity of solids by utilizing the transient temperature measurements taken within the medium. For example, the direct measurement of heat flux at the surface of a wall subjected to fire, at the outer surface of a reentry vehicle, or
at the inside surface of a combustion chamber is extremely difficult. In such situations, the inverse method of analysis, using transient temperature measurements taken within the medium can be applied for the estimation of such quantities.

Inverse heat conduction problem can be used for function estimation and parameter estimation, which are frequently used in the study of inverse analysis [Ozisik 1993]. Function estimation involves the determination of an unknown function such as the timewise variation of the surface heat flux with no prior knowledge of the functional form of the unknown quantity, and requires the determination of a large number of surface heat flux components; hence, it is referred to as an infinite dimensional minimization problem. Parameter estimation applies when some prior knowledge is available on the functional form and only a limited number of parameters are to be estimated. Such problems are referred to as finite dimensional minimization problems.

2.1.2 Difficulties and Solutions to the Inverse Problem

Inverse heat conduction problems belong to the class of problems called ill-posed problems, because their solution does not satisfy the general requirement of existence, uniqueness, and stability under small changes to the input data [Ozisik 1993]. Since the solution to ill-posed problems is unstable and sensitive to the input data, i.e., measurement errors, they are difficult to solve.

A successful solution of an inverse problem generally involves the transformation of the inverse problem into a well posed approximate solution. In order to cast IHCP as a well-posed problem, the traditional heat conduction equation was replaced by a hyperbolic heat conduction equation and the well established techniques were used to solve the resulting IHCP [Weber 1981]. For the parameter estimation, let us suppose it
involves \( M \) unknown parameters. If there are no measurement errors, it would be sufficient to have the number of measurement equal to \( M \). But if the input data always contain measurement error, it would be necessary to have extra measurement besides \( M \); thus, the system of equations to be solved becomes over-determined. One popular approach to solve the system of over-determined equations is the use of the traditional least squares method, coupled with an appropriate minimization procedure.

2.1.3 Least Squares Approach

Using the least squares approach, we can guarantee the existence of the inverse solution since the method is to minimize the least squares norm rather than make it necessarily zero [Ozisik 1993].

To solve the inverse problem, it required that the estimated temperature \( T_j(\hat{p}_i), j = 1, 2, \ldots, M, \) computed from the solution of the direct problem by using the estimated parameters, \( \hat{p}_i, \ i = 1, 2, \ldots, M, \) should match the measured temperatures \( Y_j, \ j = 1, 2, \ldots, M, \) as closely as possible over a specified time domain. The least squares norm can be set up as [Ozisik 1993]

\[
S(\hat{p}) = \sum_{i=1}^{N} \left[ Y_i - \hat{T}_i(\hat{p}) \right]^2 + \alpha^* \sum_{j=1}^{M} \hat{p}_j^2 ,
\]

(2.1)

where \( Y_i \) is the measured temperature at position \( i, \hat{T}_i(\hat{p}) \) is the estimated temperature at position \( i \) obtained from the solution of the direct problem by using the estimated values of the unknown parameters \( \hat{p} = \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_M\} \), \( \hat{p}_j \) is the \( j \)th element of the estimated parameter vector \( \hat{p} \), and \( \alpha^* \) is the regularization parameter with \( \alpha^* \geq 0 \).
In Eq. (2.1), the first summation term \( \sum_{i=1}^{N} [Y_i - \hat{T}_i(\hat{p})]^2 \) is the traditional least squares. The second summation \( \alpha^* \sum_{j=1}^{M} \hat{p}_j^2 \) is the zero-order regularization term [Beck 1985] [Scott 1985], and it is added to reduce instability or oscillations that are inherent in the solution of ill-posed problems when a large number of parameters are to be estimated [Tikhonov 1977]. The coefficient \( \alpha^* \) is called the regularization parameter. When \( \alpha^* \to 0 \), the solution exhibits oscillatory behavior and becomes unstable if a large number of parameters are to be estimated. However, for large values of \( \alpha^* \), the solution is damped and deviates from the exact results. By proper selection of \( \alpha^* \), instability can be alleviated [Hensel 1991] [Scott 1985] [Tikhonov 1977]. Thus, selection of \( \alpha^* \) is crucial while the number of parameters is large.

In order to minimize the least squares norm, Eq. (2.1) is differentiated with respect to each of the unknown parameters \( \hat{p}_j \), \( j = 1, 2, \ldots, M \), and then the resulting expression is set equal to zero to get

\[
\frac{\partial S}{\partial \hat{p}_j} = 2 \sum_{i=1}^{N} \left( \frac{\partial \hat{T}_i(\hat{p})}{\partial \hat{p}_j} \right) \cdot [\hat{T}_i(\hat{p}) - Y_i] + 2 \alpha^* \sum_{k=1}^{M} \hat{p}_k \frac{\partial \hat{p}_k}{\partial \hat{p}_j} = 0,
\tag{2.2}
\]

where \( j, k = 1, 2, \ldots, M \). Because the components of unknown parameter vector \( \hat{p} \) are independent, we have

\[
\frac{\partial \hat{p}_k}{\partial \hat{p}_j} = \begin{cases} 
0, & k \neq j \\
1, & k = j 
\end{cases}
\tag{2.3a}
\]
Since measurement errors exist, it would be necessary to have extra measurement. In other words, the total number of grid points or measurements $N$ should be larger than the number of unknown parameters $M$.

Eq. (2.2) can be rearranged in the form

$$ \sum_{i=1}^{N} \left( \frac{\partial T_i(\hat{p})}{\partial \hat{p}_j} \right) [Y_i - \hat{T}_i(\hat{p})] = \alpha \sum_{k=1}^{M} \hat{p}_k \frac{\partial \hat{p}_k}{\partial \hat{p}_j}, \quad (2.3b) $$

where $i = 1, 2, \ldots, N$, $j, k = 1, 2, \ldots, M$, and

$$ \frac{\partial \hat{T}_i(\hat{p})}{\partial \hat{p}_j} = \frac{\partial \hat{T}_i(\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_M)}{\partial \hat{p}_j} \equiv X_{ji}. \quad (2.3c) $$

Here, $X_{ji}$ is the sensitivity coefficients with respect to $\hat{p}_j$.

Eq. (2.3b) can be written in the matrix form as:

$$ X^T(Y - T) = \alpha \cdot p, \quad (2.4a) $$

where

$$ T = \begin{bmatrix} \hat{T}_1 \\ \hat{T}_2 \\ \vdots \\ \hat{T}_N \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix}, \quad p = \begin{bmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \vdots \\ \hat{p}_M \end{bmatrix}, \quad (2.4b) $$

and

$$ X = \frac{\partial T}{\partial p'} = \begin{bmatrix} \frac{\partial \hat{T}_1}{\partial \hat{p}_1} & \frac{\partial \hat{T}_1}{\partial \hat{p}_2} & \ldots & \frac{\partial \hat{T}_1}{\partial \hat{p}_M} \\ \frac{\partial \hat{T}_2}{\partial \hat{p}_1} & \frac{\partial \hat{T}_2}{\partial \hat{p}_2} & \ldots & \frac{\partial \hat{T}_2}{\partial \hat{p}_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{T}_N}{\partial \hat{p}_1} & \frac{\partial \hat{T}_N}{\partial \hat{p}_2} & \ldots & \frac{\partial \hat{T}_N}{\partial \hat{p}_M} \end{bmatrix} \quad (2.4c) $$
Here, \( X \) is called the sensitivity coefficient matrix with respect to vector \( \hat{p} \), and the elements of this matrix are

\[
X_{ij} = \frac{\partial \hat{T}_j}{\partial p_i}, \quad i = 1, 2, \cdots, N \quad \text{and} \quad j = 1, 2, \cdots, M
\]  

(2.5)

The sensitivity coefficient \( X_{ji} \) defined by Eqs. (2.3c), (2.4c) and (2.5) is the first derivative of the dependent variable (i.e., temperature) with respect to the unknown parameter (i.e., laser power, beam width, etc.). It represents the changes in \( \hat{T}_i \) with respect to the changes in the unknown parameter \( \hat{p}_j \). A small value of \( X_{ji} \) indicates insensitivity of the dependent variable to changes in the value of the unknown parameter. For such cases, the inverse analysis becomes very sensitive to measurement errors and the estimation process becomes difficult. Therefore, it is preferable to have large, uncorrelated values of the sensitivity coefficients, \( X_{ji} \).

Through the above derivations, the IHCP is reduced to that of solving the system of least squares Eqs. (2.2) or (2.4).

It is desirable to express Eq. (2.2) in a more convenient form for the calculation of the parameter \( \hat{p}_j \). This form can be achieved by expanding \( \hat{T}_i(p) \) in a Taylor series with respect to an arbitrary value of a parameter as

\[
\hat{T}_i = \hat{T}_{0i} + \sum_{h=1}^{N} \frac{\partial \hat{T}_i}{\partial p_h} (\hat{p}_h - \hat{p}_0).
\]  

(2.6a)

This result is expressed in the matrix form as

\[
T = T_0 + \frac{\partial T}{\partial p} (p - p_0).
\]  

(2.6b)

If one chooses \( T_0 = 0 \) and \( p_0 = 0 \), Eqs. (2.6a) and (2.6b) reduce, respectively, to
\[
\dot{T}_j = \sum_{h=1}^{N} \frac{\partial \dot{T}_i}{\partial \hat{p}_h} \hat{p}_h \quad (2.7a)
\]

and

\[
T = \frac{\partial T}{\partial p^i} p \equiv Xp. \quad (2.7b)
\]

Substituting Eq. (2.7a) into Eq. (2.2) gives

\[
\sum_{i=1}^{N} \frac{\partial T_i}{\partial \hat{p}_j} (Y_i - \sum_{h=1}^{N} \frac{\partial \dot{T}_h}{\partial \hat{p}_h} \hat{p}_h) = \alpha \sum_{k=1}^{M} \frac{\partial \hat{p}_k}{\partial \hat{p}_j} \frac{\partial \hat{p}_k}{\partial \hat{p}_j}. \quad (2.8a)
\]

The matrix form of Eq. (2.8a) is obtained by introducing Eq. (2.7b) into Eq. (2.4a)

\[
X'(Y - Xp) = \alpha' p. \quad (2.8b)
\]

By solving for \( p \), we have the solution in the matrix form as

\[
p = (X'X + \alpha' I)^{-1} X'Y. \quad (2.9)
\]

The solution of Eq. (2.9) gives the estimated values of the parameters \( \hat{p}_i \) at each time \( t_i (i = 1, 2, ..., M) \).

Since the system of Eq. (2.9) is nonlinear, an iterative technique is necessary for its solution. The Levenberg-Marquardt algorithm is used to solve the nonlinear least squares equations by iteration. It is suitable to solve the system of Eq. (2.9) because it combines the Newton method which converges fast but requires a good initial guess, and the steepest descent method which converges slowly but does not require a good initial guess. The Levenberg-Marquardt algorithm [Levenberg 1944], when applied to the system of Eq. (2.9) is given by

\[
p^{k+1} = p^k + (X'X + \alpha' I)^{-1} X' (Y - T). \quad (2.10)
\]
When $\alpha^* \to 0$, Eq. (2.10) reduces to the Newton’s method, and when $\alpha^* \to \infty$, it becomes the steepest descent method. Calculations are started with large values of $\alpha^*$, and its value is gradually reduced as the solution approaches the converged result.

### 2.2 Preconditioned Richardson Iteration Method

A preconditioned Richardson iteration has been introduced in [Dai 1998]. Consider a three-dimensional Poisson equation as the follow:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = f(x, y, z). \tag{2.11}$$

Let $T_{jk}$ represent the approximation of $T(iAx, jAy, kAz)$, where $Ax$, $Ay$, and $Az$ are the grid sizes in the $x$, $y$, and $z$ directions, respectively, $i = 0, \ldots, N_x$, $j = 0, \ldots, N_y$, and $k = 0, \ldots, N_z$. Using the centered-difference equation,

$$\frac{1}{Ax^2} \delta^2_{x} T_{jk} = \frac{1}{Ax^2} (T_{i+1,jk} - 2T_{jk} + T_{i-1,jk}), \tag{2.12}$$

to approximate $\frac{\partial^2 T(x, y, z)}{\partial x^2}$, and so on. Then, the finite difference scheme for Eq. (2.11) can be expressed as

$$-(\frac{1}{Ax^2} \delta^2_{x} + \frac{1}{Ay^2} \delta^2_{y} + \frac{1}{Az^2} \delta^2_{z}) T_{jk} = f_{jk}. \tag{2.13}$$

Let $(A_x \overline{T})_{jk} = -\frac{1}{Ax^2} \delta^2_{x} T_{jk}$, $(A_y \overline{T})_{jk} = -\frac{1}{Ay^2} \delta^2_{y} T_{jk}$, and $(A_z \overline{T})_{jk} = -\frac{1}{Az^2} \delta^2_{z} T_{jk}$, where $A_x$, $A_y$, and $A_z$ are matrices and $\overline{T}$ is a vector consisting of $T_{jk}$, $i = 1 \cdots N_x - 1$, $j = 1 \cdots N_y - 1$, and $k = 1 \cdots N_z - 1$. Then the system Eq. (2.13) can be written in a vector form:
\[(A_x + A_y + A_z)T = f.\]  

(2.14)

It can be seen in [Li 1979] that the eigenvalues of \( A_z \) are 
\[
\frac{\lambda_{\text{max}}(A_z)}{\lambda_{\text{min}}(A_z)} = O\left(\frac{1}{\Delta z^2}\right)
\]
will be very large, where \( \lambda_{\text{max}}(A_z) \) and \( \lambda_{\text{min}}(A_z) \) are the maximum and minimum eigenvalues of \( A_z \), respectively. The results in the system Eq. (2.14) are ill-conditioned. Hence, if using the common iteration methods, such as the Gauss-Seidel method, to solve this problem, it will converge very slowly. To overcome this shortcoming, a preconditioning technique and the Richardson iteration on Eq. (2.14) will be applied. It gives

\[
L_{\text{pre}}^{-1}(T^{(n+1)}) = L_{\text{pre}}^{-1}(T^{(n)}) - \alpha[(A_x + A_y + A_z)\bar{T}^{(n)} - \bar{f}],
\]

(2.15)

where \( \alpha \) is a relaxation parameter, and the preconditioner \( L_{\text{pre}} \) is chosen to be:

\[
L_{\text{pre}} \equiv A_z + \left(\frac{4}{\Delta x^2} + \frac{4}{\Delta y^2}\right)I.
\]

(2.16)

It is well known from the numerical linear algebra that the iteration process converges if the iteration operator

\[
B = I - \alpha L_{\text{pre}}^{-1}(A_x + A_y + A_z)
\]

(2.17)

has a spectral radius \( \rho(B) < 1 \). Furthermore, the smaller \( \rho(B) \) is, the faster the iteration converges. One can see that the eigenvalues of \( L_{\text{pre}}^{-1}(A_x + A_y + A_z) \) has the form

\[
\lambda_{ijk} = \frac{4}{\Delta x^2} \sin^2 \frac{i\pi \Delta x}{2} + \frac{4}{\Delta y^2} \sin^2 \frac{j\pi \Delta y}{2} + \frac{4}{\Delta z^2} \sin^2 \frac{k\pi \Delta z}{2} - \frac{\frac{4}{\Delta x^2} + \frac{\frac{4}{\Delta y^2} + \frac{\frac{4}{\Delta z^2} \sin^2 \frac{k\pi \Delta z}{2}}{2}}}{4}
\]

(2.18)
When $\Delta z$ is very small compared with $\Delta x$ and $\Delta y$, $\lambda_{ijk}$ is dominated by $\frac{4}{\Delta z^2} \sin^2 \frac{k\pi\Delta z}{2}$.

Thus, $\lambda_{ijk}$ is close to 1. If one chooses a relaxation parameter $\alpha$ which is close to 1, then the spectral radius $\rho(B)$ will be much smaller than 1. Hence, the iteration method Eq. (2.15) converges very fast [Dai 1998].

### 2.3 Thomas Algorithm

Thomas algorithm [Ames 1977], also known as the tridiagonal matrix algorithm (TDMA), is a simplified version of Gaussian elimination method that can be used to solve tridiagonal systems of equations. A tridiagonal system may be written as

$$a_j x_{j+1} + b_j x_j + c_j x_{j-1} = d_j, \quad 1 \leq j \leq n, \quad (2.19)$$

where $x_1, x_2, \cdots, x_n$ are unknown and $x_0, x_{n+1}$ are known.

In matrix form, this system can be written as

$$
\begin{bmatrix}
  b_1 & c_1 & 0 & \cdots & 0 \\
  a_2 & b_2 & c_2 & \ddots & \vdots \\
  0 & a_3 & b_3 & \cdots & 0 \\
  \vdots & \ddots & a_{n-1} & b_{n-1} & c_{n-1} \\
  0 & \cdots & 0 & a_n & b_n
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \vdots \\
  x_n
\end{bmatrix}
= 
\begin{bmatrix}
  d_1 - a_1 x_0 \\
  d_2 \\
  \vdots \\
  d_{n-1} \\
  d_n - c_n x_{n+1}
\end{bmatrix}
= 
\begin{bmatrix}
  \tilde{d}_1 \\
  \tilde{d}_2 \\
  \vdots \\
  \tilde{d}_{n-1} \\
  \tilde{d}_n
\end{bmatrix}.
\quad (2.20)
$$

Thomas algorithm computes the inversion of such matrix systems in the following two steps:

Firstly, by eliminating the sub-diagonal coefficients $a_i$ ($2 \leq i \leq n$) from the matrix and normalizing the coefficients along the diagonal, one obtains
Thus, the new linear system likes:

\[ x_j + e_j x_{j-1} = f_j, \quad 1 \leq j \leq n, \tag{2.22} \]

The new coefficients \( e_j \) and \( f_j \) \((1 \leq j \leq n)\) can be computed in such a way as follows. Initially, we get

\[ e_1 = \frac{c_1}{b_1}, \quad f_1 = \frac{d_1}{b_1}. \tag{2.23} \]

Then, the coefficients are computed forward as the follow:

\[
\begin{align*}
\left\{ \ight.
\begin{array}{l}
\quad e_j = \frac{c_j}{b_j + a_j e_{j-1}}, \\
\quad f_j = \frac{\tilde{d}_j - a_j f_{j-1}}{b_j + a_j e_{j-1}}, \quad 2 \leq j \leq n.
\end{array}
\tag{2.24}
\end{align*}
\]

Secondly, by computing \( x_j \) \((n \geq j \geq 1)\) backwards, we obtain

\[ x_j = f_j - e_j x_{j+1}, \quad n \geq j \geq 1. \tag{2.25} \]

By using Thomas algorithm, the solution for such systems is obtained in \( O(n) \) operations instead of \( O(n^3) \) required by Gaussian Elimination. Moreover, a considerable reduction in memory requirements and an increase in computational speed can be gained by storing the nonzero elements in three 1D vectors instead of saving the entire \( n \times n \) matrix.
2.4 Constructal Theory of Multi-scale
Tree-shaped Heat Exchangers

Based on the constructal theory in [Bejan 2000], assume that the heat exchanger has the flow structure shown in Figure 2.1. The flow structures that emerge along this design route are tree-shaped, with multiple scales that are arranged hierarchically [Silva 2004].

Figure 2.1 The construction of the tree of convective heat currents, (a) the constrained optimization of the geometry of a T-shaped construct; (b) the stretched tree of optimized constructs; (c) the superposition of two identical trees oriented in counterflow; and (d) the convective heat flow along a pair of tubes in counterflow.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The constructal tree of Figure 2.1 distributes a stream uniformly over a square shaped area. The balanced countercurrent heat exchanger has two identical trees, one for the hot fluid and the other for the cold fluid. These two trees conjugate exactly and one tube of the hot tree is parallel to the corresponding tube of the cold tree. Each tree is made of many tubes of \((n+1)\) sizes. The tube at level \(i\) has the length \(L_i\) and internal diameter \(D_i\), where \(i = 0, 1, \cdots, n\). The number of tubes of type \(i\) is \(n_i\). In constructal design, the entire flow starts from the smallest tube scale \((L_n, D_n)\), called element system, which holds the smallest square area element. Larger constructs are made by paring smaller constructs. For example, two elemental systems are joined to consist of the first constructs \((i = n-1)\), and the two elemental streams are joined into a first-construct stem of size \((L_{n-1}, D_{n-1})\). Tube lengths halve after two consecutive construction steps means the stem \((L_{n-1}, D_{n-1})\) is twice as long as \(L_n\). When there are many construction levels, the relationship of the tube lengths at different levels can be expressed as follows [Bejan 2000]:

\[
L_i = 2^{i-1} L_{i+1}, \quad i = 0, 1, \cdots, n. \tag{2.26}
\]

Paring at every construction level means that the tube numbers [Bejan 2000] are ordered as

\[
n_i = 2^i, \quad i = 0, 1, \cdots, n. \tag{2.27}
\]

The stream \(\dot{m}_i\) encounters the flow resistance of two \(L_{i+1}\) tubes in parallel, which assemble to one \(L_i\) tube. Thus, the ratio of the flow rates [Bejan 2000] satisfies

\[
\dot{m}_i = 2^{-i} \dot{m}_0, \quad i = 0, 1, \cdots, n. \tag{2.28}
\]

Moreover, according to Murray's Law [Murray 1926] [Thompson 1942], when the
optimal diameter ratio satisfies

\[ \frac{D_{i+1}}{D_i} = 2^{-\frac{1}{3}}, \quad i = 0, 1, \ldots, n, \]  

(2.29)

the resistance is minimized by fixing the total tube volume.

2.5 Previous Work on Bio-heat Transfer

As hyperthermia combined with radiation and cytotoxic drugs has demonstrated increased effectiveness in the treatment of certain types of cancers, more and more people are interested in the related research and building the mathematical models for the clinic process. Until now, most utilized models for hyperthermia treatment planning involve the Pennes bio-heat transfer model (BHTE), in which the heat transfer between the blood vessels and tissue is assumed to occur mainly across the capillaries when the blood velocity is low [Pennes 1948]. The blood in the capillary bed instantly thermally equilibrates with the temperature of the surrounding tissue and enters the venous circulation at the local tissue temperature. Therefore, the contribution of the blood flow was modeled as a heat sink whose magnitude is proportional to the difference between the arterial supply temperature and the local tissue temperature.

There are many numerical and experimental methods developed based on the Pennes bio-heat transfer model. Clegg and Roemer [Clegg 1989], [Roemer 1989] performed hyperthermia sessions on a normal canine thigh to test the ability of a state and parameter estimation method to accurately predict the complete three-dimensional (3D) temperature distribution in experimental situations. They employed the Pennes equation as the system model and an optimization algorithm, which is based on a least squares
error objective function, used for predicting certain unknown model parameters, such as the blood perfusion and the power deposition. Martin and Bowman [Martin 1989] presented the exact steady state and transient solutions for the temperature distribution in laser irradiated and perfused tissue using the Pennes equation in cylindrical coordinates. The obtained solutions were used to evaluate the significance of blood perfusion during continuous wave laser heating.

Liauh and Roemer [Liauh 1993] presented a semilinear state and parameter estimation algorithm that decreases the total computational time required to accurately reconstruct complete hyperthermia temperature fields, since the relationship between the temperature and the blood perfusion based on the Pennes bio-heat transfer equation is generally nonlinear in the hyperthermia temperature estimation problem.

Chatterjee and Adams [Chatterjee 1994] generated a 2D finite element thermal model of the prostate region of the human body based on the Pennes equation using the automatic mesh generation capabilities of the software package ANSYS. The results show how selective heating can be obtained in the tumor region and the effects of varying blood flow rates.

Huang [Huang 1994] considered the heat transfer within a perfused tissue in the presence of a vessel. The Pennes bio-heat transfer equation was used for the perfused tissue, and a lumped capacitance analysis was used for the convection in the vessel with a constant Nusselt number. Analytical solutions of the Pennes equation with a blood vessel were obtained. Payne [Payne 1999] derived a design of the phantom from a combination of the convective fin equation and the Pennes BHTE, and developed a phantom model using an inverse technique applied to experimental data from a thin layer phantom to
determine model parameters. Majchrzak and Mochacki [Majchrzak 1999] considered
the thermal processes proceeding within a perfused tissue in the presence of a vessel. The
Pennes bio-heat transfer equation governs the steady state temperature field in the tissue
sub-domain, while the ordinary differential equation resulting from the energy balance
describes the change of blood temperature along the vessel. The problem was solved
using the combined numerical algorithm, in particular the boundary element method (for
the tissue sub-domain) and the finite difference method (for the blood vessel sub-
domain).

Liu and his co-workers [Liu 1995], [Liu 2000a], [Liu 1999] introduced a general
form of the thermal wave model of Pennes bio-heat transfer in living tissues. The model
was obtained based on a modified unsteady conduction equation (the CV equation). A
general heat flux criterion was established to determine when the thermal wave
propagation dominates the principal heat transfer process. This model can be used for
tissue temperature prediction. Liu and Lu [Liu 2000b], [Lu 1998] also used the dual
reciprocity boundary element method to solve the integral inverse or direct bio-heat
transfer problems.

Zhou and Liu [Zhou 2004] calculated temperature distributions based on the
continuity, momentum and energy equations used in the fluid dynamics. Dai [Dai 2003a],
[Dai 2003b] developed a domain decomposition method for solving the 3D Pennes bio-
heat transfer equation in a triple-layered skin structure. Recently, Zhang [Zhang 2005a]
[Zhang 2005b] developed a numerical method for obtaining an optimal temperature
distribution in a triple-layered cylindrical skin structure. It is the first time that the triple-
layered skin structure composed of epidermis, dermis and subcutaneous was considered
in the numerical model for the laser-induced hyperthermia by his work. The method proved to be useful in optimizing laser power for a given laser irradiation pattern. However, the influence of blood vessels in the study was ignored. The presence of thermally significant vessels can have a dramatic impact on the temperature distribution in hyperthermia applications [Klemick 1997]. In this dissertation, we will consider a more complex and realistic 3D triple-layered skin structure embedded with multi-level blood vessels, and develop a numerical method to obtain the optimal temperature distribution among it.

In this chapter, the inverse heat conduction problem, preconditioned Richardson iteration method, Thomas algorithm, and constructal theory of multi-scale tree-shaped heat exchangers which will be applied in our research have been introduced. Also, the relevant research on bio-heat transfer has been briefly reviewed.
CHAPTER THREE

MATHEMATICAL MODEL

3.1 Problem Set Up

3.1.1 Problem Description

In this study, we will develop a numerical method for obtaining an optimal temperature distribution in a 3D triple-layered skin structure embedded with multi-level blood vessels where the surface of the skin is irradiated by a laser. The skin structure is composed of epidermis, dermis and subcutaneous, while the dimension and blood flow of the multi-level blood vessels are determined based on the constructal theory of multi-scale tree-shaped heat exchangers [Bejan 2001] [Silva 2004] as described in the previous chapter. This method determines the required laser intensity to obtain pre-specified temperatures at the given locations of the skin after a pre-specified laser exposure time.

3.1.2 Structure of the Skin and the Embedded Blood Vessels

Based on the histological knowledge [Ham 1965] [Gartner 2000], the skin is composed of three layers: epidermis, dermis and subcutaneous. The tissue in different layer has different properties such as the density, the heat conductivity and the specific heat, etc. The largest blood vessels of the skin are arranged in the form of a flat network in the subcutaneous tissue, immediately below the dermis. This vascular network is called the rete cutaneum. From the rete cutaneum, branches of the blood vessels pass both
inwardly and outwardly. Those that pass outwardly are called arteries and they supply the blood for the skin, while those that pass inwardly are called veins and they collect the blood returned from the skin. On the other hand, the dermis is very sparingly supplied with capillaries and the capillary beds of skin lie immediately under the epidermis. Figure 3.1 [Lorimer 1999] shows the realistic skin structure.

![Skin Diagram](image)

**Figure 3.1 Skin structure and its components**

As described in [Huang 1996], there are up to seven levels of blood vessels beginning with the main ones. Here, we consider only the last three levels of them since these blood vessels are embedded in the skin. We label them as level 1, level 2, and level 3 vessel, respectively. The level 1 vessel which runs along the opposite direction of \( z \) axis branches into two level 2 vessels which run lengthwise (\( x \)). Each level 2 vessel then changes the direction and becomes a level 3 vessel, which has the same running direction.
as level 1 vessel. According to the constructal theory of multi-scale tree-shaped heat exchangers [Bejan 2000] [Bejan 2001] [Silva 2004] [Wechsatol 2002], the diameters of the blood vessels are assumed to be decreasing by a constant ratio \( \gamma \) between successive levels of branched vessels, which is shown in the following:

\[
\gamma = \frac{NL_{b}^{m+1}}{NL_{b}^{m}} = \frac{NW_{b}^{m+1}}{NW_{b}^{m}} = 2^{\frac{1}{3}}, \quad m = 1, 2, \quad (3.1)
\]

where \( NL_{b}^{m} \) and \( NW_{b}^{m} \) are the length and width of the cross-section of a blood vessel at level \( m \), respectively. The length of the blood vessel is assumed to be double after two consecutive construction steps, which can be expressed in the length-doubling rule [Bejan 2000] [Wechsatol 2002] [Silva 2004] as follows:

\[
L_{b}^{m} = 2^\frac{1}{2} L_{b}^{m+1}, \quad m = 1, 2, \quad (3.2)
\]

where \( L_{b}^{m} \) is the length of the blood vessel at level \( m \). The mass flow of blood in the \( m \)th level vessel, \( M_{m} = v_{m} F_{m}, m = 1, 2, 3, \) is assumed to satisfy [Silva 2004],

\[
M_{1} = 2M_{2}, \quad (3.3a)
\]

where \( v_{m} \) is the velocity of blood flow and \( F_{m} (= NL_{b}^{m} \times NW_{b}^{m}) \) is the area of the cross-section in the \( m \)th level vessel. Since each vessel in level 2 does not branch but just changes the direction to form the 3rd level blood vessel, we assume that the mass flow of blood in the 3rd level vessel is the same as in the 2nd level vessel. Thus, we obtain

\[
M_{3} = M_{2}. \quad (3.3b)
\]

3.1.3 3D Schematic Configuration

We consider the target region to be a rectangular structure embedded with multi-level blood vessels that cross through the subcutaneous layer from the bottom to the top.
Figure 3.2 shows the configuration of a 3D skin structure embedded with a multi-level artery, while Figure 3.3 shows the configuration of a 3D skin structure embedded with both a multi-level artery and a multi-level vein.
3.2 Governing Equations

3.2.1 Governing Equations for Tissue

We employ the modified Pennes model to describe the thermal behavior of the triple-layered skin structure when irradiated by the laser. Thus, the governing equation for the tissue region can be expressed as follows [Pennes 1948]:

\[
\rho_l C_l \frac{\partial \theta_l}{\partial t} - k_l \left[ \frac{\partial^2 \theta_l}{\partial x^2} + \frac{\partial^2 \theta_l}{\partial y^2} + \frac{\partial^2 \theta_l}{\partial z^2} \right] + W_b^l C_b^l (\theta_l - \theta_b) = Q_l, \quad l = 1, 2, 3,
\]

where \(\theta_l\) is the tissue temperature elevation due to heating by a laser; \(\theta_b\) is the blood temperature elevation at the end of the third level vessel; \(\rho_l, C_l\) and \(k_l\) denote density, specific heat, and thermal conductivity of tissue, respectively; \(C_b^l\) is the specific heat of blood; \(W_b^l\) is the blood perfusion rate; and \(Q_l\) is the volumetric heat due to heating. In this study, we use laser as the heat source. The laser power is compliant to Gaussian distribution. Consequently, the heat source \(Q_l\) can be written as follows [Jaesung 1994]:

\[
Q_l = \alpha_l e^{-\alpha_l^2} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{1}{e^{\left(\frac{(x-x_0(t))^2+(y-y_0(t))^2}{2\sigma^2}\right)}} P_0(1 - \text{Reff}_l), \quad l = 1, 2, 3
\]

\[
Q_2 = \alpha_2 e^{-\alpha_2^2} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{1}{e^{\left(\frac{(x-x_0(t))^2+(y-y_0(t))^2}{2\sigma^2}\right)}} P_0(1 - \text{Reff}_2), \quad (3.5b)
\]

\[
Q_3 = \alpha_3 e^{-\alpha_3^2} \frac{1}{\sqrt{2\pi\sigma^2}} \frac{1}{e^{\left(\frac{(x-x_0(t))^2+(y-y_0(t))^2}{2\sigma^2}\right)}} P_0(1 - \text{Reff}_3), \quad (3.5c)
\]

where \(\alpha_1, \alpha_2, \alpha_3\) are laser absorptivities of the three layers of the skin, respectively; \(\text{Reff}_1, \text{Reff}_2, \text{and Reff}_3\) are laser reflectivities of the three layers of the skin,
respectively; 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 space $t$. $P_0$ is the laser intensity, and $\sigma$ is the standard deviation of the laser beam width.

### 3.2.2 Governing Equations for Blood

Since the blood temperature elevation, $\theta_b$, is in the modified Pennes bio-heat transfer equation, Eq. (3.4), one must solve $\theta_b$ first. Here, we use the convective energy balance equations to describe the thermal behavior of blood vessels. Additionally, the temperature elevation of blood in the cross section of a vessel is assumed to be uniform. Hence, the convective energy balance equations used to calculate the blood temperature elevations at level 1 and 2 can be expressed as

$$C_B M_1 \frac{d(\theta_b^1)}{dz} - \alpha P_1 (\theta_w^1 - \theta_b^1) = C_B F_1 \frac{\partial \theta_b^1}{\partial t}, \quad (3.6a)$$

and

$$C_B M_2 \frac{d(\theta_b^2)}{dx} - \alpha P_2 (\theta_w^2 - \theta_b^2) = C_B F_2 \frac{\partial \theta_b^2}{\partial t}, \quad (3.6b)$$

where $C_B$ is the heat capacity of blood, and $\alpha$ is the heat transfer coefficient between blood and tissue, $M_m$ is the mass flow of blood in the $m$th level vessel, and $P_m$ is the periphery of the cross section of the level $m$ vessel. Further, $\theta_w^m$ and $\theta_b^m$ are the wall temperature elevation and the blood temperature elevation in the $m$th level vessel. For the smallest, terminal vessels (level 3) as shown in Figure (3.2) and Figure (3.3), a decreased blood flow rate ($\dot{P}$) is included in the energy balance equation

$$C_B M_3 \frac{d(\theta_b^3)}{dz} - \alpha P_3 (\theta_w^3 - \theta_b^3) - \dot{P} C_B F_3 \theta_b^3 = C_B F_3 \frac{\partial \theta_b^3}{\partial t}. \quad (3.6c)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Particularly, when the lengths of the blood vessels embedded in the skin structure are considered to be short, the temperature of the blood can then reach the steady state in a short time. Thus, under the steady state, the above governing equations for blood (Eqs. 3.6(a)-3.6(c)) can be expressed as followed [Huang 1992] [Majchrzak 1999]:

\[
\begin{align*}
C_b M_1 \frac{d(\theta_b^1)}{dz} - \alpha P_1 (\theta_w^1 - \theta_b^1) &= 0, \\
C_b M_2 \frac{d(\theta_b^2)}{dx} - \alpha P_2 (\theta_w^2 - \theta_b^2) &= 0,
\end{align*}
\]

and

\[
C_b M_3 \frac{d(\theta_b^3)}{dz} - \alpha P_3 (\theta_w^3 - \theta_b^3) - \hat{P} C_b F_3 \theta_b^3 = 0.
\]

### 3.3 Interfacial Conditions

The interfacial conditions between layers are assumed to be perfect thermal contact:

\[
\begin{align*}
\theta_1 &= \theta_2, \quad k_1 \frac{\partial \theta_1}{\partial z} = k_2 \frac{\partial \theta_2}{\partial z}, \quad z = L_1, \\
\theta_2 &= \theta_3, \quad k_2 \frac{\partial \theta_2}{\partial z} = k_3 \frac{\partial \theta_3}{\partial z}, \quad z = L_1 + L_2.
\end{align*}
\]

The continuity of the heat transfer between the lateral blood vessel and tissue obeys the Newton’s law of cooling, it requires [Huang 1994]

\[
\frac{\partial \theta_b^m}{\partial n} = B_i (\theta_w^m - \theta_b^m),
\]

where \(B_i\) is the Biot number, which equals \(\frac{\alpha}{k_s}\) since the blood vessels are located in the subcutaneous layer as we described in Chapter 2.
3.4 Boundary Conditions

On the skin surface, the heat exchange with the environment is [Deng 2004]

\[ k_1 \frac{\partial \theta_i}{\partial z} = h(\theta_i - \theta_{air}), \quad z = 0, \quad (3.10) \]

where \( h \) is the heat convection coefficient between the skin surface and the surrounding air. For simplicity, we assume that the heat flux approaches zero as the tissue depth increases, which is realistic for a biological body [Liu 1999]. Hence, the boundary conditions for other sides of tissue are assumed to be

\[ \frac{\partial \theta_i}{\partial \hat{n}} = 0, \quad (3.11) \]

where \( \hat{n} \) is the unit outward normal vector on the boundary. At the entrance to the first level of the artery, we have

\[ \theta_{b_1} = \theta_{in}, \quad (3.12) \]

where \( \theta_{in} \) is the blood temperature elevation at the entrance of the artery. At the exit of the artery, we assume that the blood temperature elevation is equal to the surrounding tissue temperature elevation

\[ \theta_{b_3} = \theta_{out}, \quad (3.13) \]

As mentioned earlier in Chapter 2, the blood in the vein has an opposite flowing direction to that in the artery. Thus, the entrance of the blood to the vein is located at the third level and the blood temperature elevation is equal to the surrounding tissue temperature elevation.
3.5 Initial Conditions

Initially, both the tissue temperature and blood temperature are assumed to be the normal body temperature. Thus, the initial condition for the temperature elevation in tissue is assumed to be

$$\theta_i = 0, \quad t = 0, \quad l = 1, 2, 3.$$ (3.14)

Similarly, the initial condition for the temperature elevation in blood is

$$\theta_b^l = 0, \quad t = 0, \quad l = 1, 2, 3.$$ (3.15)

So far, we have described the problem, the structure of the multi-level blood vessels, and the schematic configuration. The mathematical model has also been presented with the interfacial conditions, boundary conditions and initial conditions. In the next chapter, we will develop the numerical method for solving the described problem.
CHAPTER FOUR

NUMERICAL METHOD

4.1 Notations

To obtain an optimal temperature distribution numerically, we denote \((u_n^i)_{ijk}^n\) and \(u_b\) as the numerical approximations of \((\theta_i)(iAx, jAy, kAz, nAt)\) and \(\theta_b\), respectively, where \(\Delta x\), \(\Delta y\), and \(\Delta z\) are the mesh sizes, \(\Delta t\) is the time increment, \(n\) is the time level, and \(i, j, k\) are integers with \(0 \leq i \leq N_x\), \(0 \leq j \leq N_y\), \(0 \leq k \leq N_z\), so that \(N_x\Delta x = NX, N_y\Delta y = NY\), and \(N_z\Delta z = L_z\), \(l = 1, 2, 3\). In the mesh, we assume that \((u_3)_{ijk}^n = (u_b)_{ijk}\) when the grid point \((i, j, k)\) is in the \(m\)th level blood vessel.

4.2 Finite Difference Schemes

We employ the second-order finite differences to approximate \(\frac{\partial^2 \theta_i}{\partial x^2}\), \(\frac{\partial^2 \theta_i}{\partial y^2}\), and \(\frac{\partial^2 \theta_i}{\partial z^2}\) at point \((iAx, jAy, kAz)\) as follows:

\[
\frac{\partial^2 \theta_i}{\partial x^2} \approx \frac{(\theta_i)_{i+1,jk} - 2(\theta_i)_{ijk} + (\theta_i)_{i-1,jk}}{\Delta x^2}, \quad (4.1)
\]

\[
\frac{\partial^2 \theta_i}{\partial y^2} \approx \frac{(\theta_i)_{ij+1,k} - 2(\theta_i)_{ijk} + (\theta_i)_{ij-1,k}}{\Delta y^2}, \quad (4.2)
\]

31

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[
\frac{\partial^2 \theta_{i}}{\partial z^2} \approx \frac{(\theta_{i})_{j,k+1} - 2(\theta_{i})_{j,k} + (\theta_{i})_{j,k-1}}{\Delta z^2}.
\] (4.3)

A 7-point Crank-Nicholson scheme for solving the Pennes equation, Eq. (3.4), can be expressed as follows:

\[
\rho C_i \frac{\left(u_i\right)_{j,k+1} - \left(u_i\right)_{j,k}}{\Delta t} + \frac{\left(W_i C_b \left(u_i\right)_{j,k} + \left(u_i\right)_{j,k} - \left(u_b\right)_{m}\right)}{2} = k_i \left(\delta_x^2 + \delta_y^2 + \delta_z^2\right) \frac{\left(u_i\right)_{j,k} + \left(u_i\right)_{j,k}}{2} + \left(Q_i\right)_{j,k+1}^0, \quad i = 1, 2, 3,
\] (4.4)

where

\[
\delta_x^2 u_{j,k} = \frac{u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}}{\Delta x^2}, \quad \delta_y^2 u_{j,k} = \frac{u_{j+1,k} - 2u_{j,k} + u_{j-1,k}}{\Delta y^2}
\]

Because Eqs. (3.6a)-(3.6c) are hyperbolic equations, we use an upwind scheme [Smith 1986] to solve them. As such, the explicit finite difference approximations of Eqs. (3.6a)-(3.6c) for both artery and vein at these three levels can be written as follows:

At level 1 of artery

\[
C_b M_1 \frac{\left(\theta^1_{b}\right)_{j+1,k} - \left(\theta^1_{b}\right)_{j,k}}{\Delta z} - \alpha P_1 \left(\theta^1_{w}\right)_{j,k} - \left(\theta^1_{b}\right)_{j,k} = C_b F_1 \frac{\left(\theta^1_{w}\right)_{j,k} - \left(\theta^1_{b}\right)_{j,k}}{\Delta t}, \quad (4.5a)
\]

On the left branch of level 2 in artery

\[
C_b M_2 \frac{\left(\theta^2_{b}\right)_{j+1,k} - \left(\theta^2_{b}\right)_{j,k}}{\Delta x} - \alpha P_2 \left(\theta^2_{w}\right)_{j,k} - \left(\theta^2_{b}\right)_{j,k} = C_b F_2 \frac{\left(\theta^2_{w}\right)_{j,k} - \left(\theta^2_{b}\right)_{j,k}}{\Delta t}, \quad (4.5b)
\]

On the right branch of level 2 in artery

\[
C_b M_2 \frac{\left(\theta^2_{b}\right)_{j+1,k} - \left(\theta^2_{b}\right)_{j,k}}{\Delta x} - \alpha P_2 \left(\theta^2_{w}\right)_{j,k} - \left(\theta^2_{b}\right)_{j,k} = -C_b F_2 \frac{\left(\theta^2_{w}\right)_{j,k} - \left(\theta^2_{b}\right)_{j,k}}{\Delta t}, \quad (4.5c)
\]
At level 3 of artery
\[
C_B M_3 \frac{(\theta_b^3)_{k+1}^{n+1} - (\theta_b^3)_k^{n+1}}{\Delta z} - \alpha P_3 ((\theta_w^3)_k^{n+1} - (\theta_b^3)_k^{n+1}) - F_3 \dot{\varphi} C_B (\theta_b^3)_k^{n+1} = C_B F_3 \frac{(\theta_b^3)_k^{n+1} - (\theta_b^3)_k^n}{\Delta t};
\]  
(4.5d)

At level 1 of vein
\[
C_B M_1 \frac{(\theta_b^1)_{k+1}^{n+1} - (\theta_b^1)_k^{n+1}}{\Delta z} - \alpha P_1 ((\theta_w^1)_k^{n+1} - (\theta_b^1)_k^{n+1}) = C_B F_1 \frac{(\theta_b^1)_k^{n+1} - (\theta_b^1)_k^n}{\Delta t};
\]  
(4.6a)

On the left branch of level 2 in vein
\[
C_B M_2 \frac{(\theta_b^2)_{i+1}^{n+1} - (\theta_b^2)_i^{n+1}}{\Delta x} - \alpha P_2 ((\theta_w^2)_{i+1}^{n+1} - (\theta_b^2)_{i+1}^{n+1}) = C_B F_2 \frac{(\theta_b^2)_i^{n+1} - (\theta_b^2)_i^n}{\Delta t};
\]  
(4.6b)

On the right branch of level 2 in vein
\[
C_B M_2 \frac{(\theta_b^2)_{i+1}^{n+1} - (\theta_b^2)_i^{n+1}}{\Delta x} - \alpha P_2 ((\theta_w^2)_i^{n+1} - (\theta_b^2)_i^{n+1}) = C_B F_2 \frac{(\theta_b^2)_i^{n+1} - (\theta_b^2)_i^n}{\Delta t};
\]  
(4.6c)

At level 3 in vein
\[
C_B M_3 \frac{(\theta_b^3)_{k+1}^{n+1} - (\theta_b^3)_k^{n+1}}{\Delta z} - \alpha P_3 ((\theta_w^3)_k^{n+1} - (\theta_b^3)_k^{n+1}) - F_3 \dot{\varphi} C_B (\theta_b^3)_k^{n+1} = C_B F_3 \frac{(\theta_b^3)_k^{n+1} - (\theta_b^3)_k^n}{\Delta t}.
\]  
(4.6d)

In the above equations, the label of the grid point increases along the x-axis and decreases along the z-axis for both artery and vein.

For the steady state case, since Eqs. (3.7a)-(3.7c) are first-order ordinary differential equations, they are solved by using the fourth-order Runge-Kutta method [Burden 2001]. Thus, for Eq. (3.7a) and Eq. (3.7b) at any time level, the finite difference scheme can be written as
\[ f(t, \theta^l_b) = \frac{\alpha P}{C_b M_l} (\theta^l_w - \theta^l_b), \quad l = 1, 2, \]
\[ k_1 = hf(t, \theta^l_b) = \frac{\alpha P h}{C_b M_l} (\theta^l_w - \theta^l_b), \]
\[ k_2 = hf(t + \frac{h}{2}, \theta^l_b + \frac{1}{2} k_1) = \frac{\alpha P h}{C_b M_l} [\theta^l_w - (\theta^l_b + \frac{1}{2} k_1)], \]
\[ k_3 = hf(t + \frac{h}{2}, \theta^l_b + \frac{1}{2} k_2) = \frac{\alpha P h}{C_b M_l} [\theta^l_w - (\theta^l_b + \frac{1}{2} k_2)], \]
\[ k_4 = hf(t + h, \theta^l_b + k_3) = \frac{\alpha P h}{C_b M_l} [\theta^l_w - (\theta^l_b + k_3)], \]
\[ (\theta^l_b)^{ve} = (\theta^l_b)^v + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \]

where \( h = \Delta z \) when \( l = 1 \) and \( h = \Delta x \) when \( l = 2 \), the label of the grid point increases along the blood flowing direction for both artery and vein. The finite difference scheme for Eq. (3.7c) is

\[ f(t, \theta^3_b) = \frac{\alpha P_3}{C_b M_3} (\theta^3_w - \theta^3_b) + \frac{\dot{P}F_3}{M_3} \theta^3_b, \]
\[ k_1 = hf(t, \theta^3_b) = \frac{\alpha P_3 \Delta z}{C_b M_3} (\theta^3_w - \theta^3_b) + \frac{\dot{P}F_3 \Delta z}{M_3} \theta^3_b, \]
\[ k_2 = hf(t + \frac{h}{2}, \theta^3_b + \frac{1}{2} k_1) = \frac{\alpha P_3 \Delta z}{C_b M_3} [\theta^3_w - (\theta^3_b + \frac{1}{2} k_1)] + \frac{\dot{P}F_3 \Delta z}{M_3} (\theta^3_b + \frac{1}{2} k_1), \]
\[ k_3 = hf(t + \frac{h}{2}, \theta^3_b + \frac{1}{2} k_2) = \frac{\alpha P_3 \Delta z}{C_b M_3} [\theta^3_w - (\theta^3_b + \frac{1}{2} k_2)] + \frac{\dot{P}F_3 \Delta z}{M_3} (\theta^3_b + \frac{1}{2} k_2), \]
\[ k_4 = hf(t + h, \theta^3_b + k_3) = \frac{\alpha P_3 h}{C_b M_3} [\theta^3_w - (\theta^3_b + k_3)] + \frac{\dot{P}F_3 \Delta z}{M_3} (\theta^3_b + k_3), \]
\[ (\theta^3_b)^{ve} = (\theta^3_b)^v + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \]

The discrete interfacial equations for Eqs. (3.8a)-(3.8b) are assumed to be, for any time level,

\[ k_1 \frac{(u_1^n)_{y_0}}{\Delta z} - (u_1^n)_{y_0} = k_2 \frac{(u_2^n)_{y_1} - (u_2^n)_{y_0}}{\Delta z}, \quad (u_1^n)_{y_0} = (u_2^n)_{y_0}, \]
and when the grid point \((i, j)\) is in the tissue

\[
\frac{k_2}{\Delta z} \frac{(u_2)_j^0 - (u_2)_{j+1}^0}{\Delta z} = k_3 \frac{(u_3)_j^0 - (u_3)_{j-1}^0}{\Delta z}, \quad (u_2)_{j-1}^0 = (u_3)_{j+1}^0. \tag{4.8b}
\]

The interfacial condition, Eq. (3.9), between the tissue and the lateral blood vessel is discretized as follows:

\[
(u_3)_{jk}^{n+1} = \left( (u_3)_{jk}^{n+1} + Bi \cdot \Delta x \cdot (u_3)_{j-1, k}^{n+1} \right)/(1 + Bi \cdot \Delta x), \tag{4.9a}
\]

\[
(u_3)_{yk}^{n+1} = \left( (u_3)_{yk}^{n+1} + Bi \cdot \Delta y \cdot (u_3)_{y-1, k}^{n+1} \right)/(1 + Bi \cdot \Delta y), \tag{4.9b}
\]

\[
(u_3)_{hk}^{n+1} = \left( (u_3)_{hk}^{n+1} + Bi \cdot \Delta z \cdot (u_3)_{h+1, k}^{n+1} \right)/(1 + Bi \cdot \Delta z), \tag{4.9c}
\]

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 boundary conditions are discretized as follows:

\[
(u_3)_{b, jk}^n = (u_3)_{j+1, k}^n, \quad (u_3)_{n, jk}^n = (u_3)_{j-1, k}^n, \tag{4.10a}
\]

\[
(u_3)_{l, 0k}^n = (u_3)_{i, k}^n, \quad (u_3)_{l, 0k}^n = (u_3)_{l, 0, k-1}^n, \tag{4.10b}
\]

\[
k_1 \frac{(u_3)_j^0 - (u_3)_{j-1}^0}{\Delta z} = \hat{h}[(u_3)_j^0 - \theta_{air}], \tag{4.10c}
\]

\[
(u_3)_{j+1}^0 = (u_3)_{j}^0, \quad (u_3)_{j-1}^0 = (u_3)_{j}^0, \tag{4.10d}
\]

for any time level \(n\). The initial conditions are

\[
(u_i)_{b, jk}^0 = 0, \quad l = 1, 2, 3, \tag{4.11a}
\]

and

\[
(u_i)_{l, 0k}^0 = 0, \quad l = 1, 2, 3, \tag{4.11b}
\]

for the grid points in tissue and in blood, respectively.
4.3 Laser Intensity Optimization

To determine the laser intensity \( P_0 \) so that an optimal temperature distribution can be obtained, we pre-specify the temperature elevations to be obtained at the center and some locations in the perimeter on the skin surface. By guessing the initial laser intensity \( P_0 \) and pre-specifying a laser exposure pattern, we can solve the above equations to obtain a temperature field in the entire 3D skin structure. Once the temperatures, \( u_{\text{cal}}^i \), are calculated at the given locations, a least squares approach is employed to minimize the difference between the pre-specified temperature elevation \( \theta_{\text{pre}} \) and the calculated temperature \( u_{\text{cal}} \). To this end, we let

\[
S(P_0) = \sum_{i=0}^{M} (\theta_{\text{pre}}^i - u_{\text{cal}}^i)^2, \quad i = 0, 1, \cdots, M, \tag{4.12}
\]

where \( M \) is the number of the selected locations. Minimizing \( S(P_0) \) in Eq. (4.12), we obtain

\[
\frac{d}{dP_0} S(P_0) = -2 \sum_{i=0}^{M} \left( \frac{d(u_{\text{cal}}^i)}{dP_0} \right) \left( \theta_{\text{pre}}^i - u_{\text{cal}}^i \right) = 0, \quad i = 0, 1, \cdots, M. \tag{4.13}
\]

An iterative scheme for computing \( P_0 \) can then be expressed as follows [Ozisik 1993]:

\[
P_0^{(j+1)} = P_0^{(j)} + (X'X + \alpha^* I)^{-1} X'(\tilde{\theta}_{\text{pre}} - \tilde{u}_{\text{cal}}), \tag{4.14}
\]

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

\[
X = \left[ \frac{\partial(u_{\text{cal}})}{\partial P_0}, \frac{\partial(u_{\text{cal}})}{\partial P_0}, \cdots, \frac{\partial(u_{\text{cal}})}{\partial P_0} \right]^T, \tag{4.15}
\]

and
4.4 Preconditioned Richardson Iteration for Computation

It is noted that Eq. (4.4) is a three-dimensional implicit scheme with seven unknowns with a small $\Delta z$. Thus, the computation is very complex and very slow because the grid size $\Delta z$ is very small compared with $\Delta x$ and $\Delta y$ (a test problem can be seen in [Dai 2003a]). To simplify and speed up the computation, we employ a preconditioned Richardson iteration method based on the idea given in [Dai 2003a] [Dai 2003b] as described in Section 2.2:

\[
L^i_{\text{pre}}[(u_i)^n_{\text{jk}}]^{(l+1)} = L^i_{\text{pre}}[(u_i)^n_{\text{jk}}]^{(l)} - \omega \cdot \Re, 
\]  
(4.17a)

where

\[
\Re = [(u_i)^n_{\text{jk}}]^{(l)} - (u_i)^n_{\text{jk}} + \frac{W^i_k C^i_b \Delta t}{\rho_i C_i} \left( \frac{[u_i]^{n+1}_{\text{jk}}]^{(l)} + [u_i]^{n}_{\text{jk}} - (u_b)_{\text{out}} \right) 
- \frac{k_i \Delta t}{\rho_i C_i} \left( \frac{\delta^2_x + \delta^2_y + \delta^2_z}{2} \right) \left( [u_i]^{n+1}_{\text{jk}}]^{(l)} + [u_i]^{n}_{\text{jk}} \right) - \frac{\Delta t}{\rho_i C_i} (Q_i)^{n+1}_{\text{jk}}, \quad l = 1,2,3. 
\]  
(4.17b)

Here, $\omega$ is a relaxation parameter ($0 \leq \omega \leq 1$), $l$ is an iterative index, and $L^i_{\text{pre}}$ is the preconditioned operator which can be defined as follows:

\[
L^i_{\text{pre}} = 1 + \frac{W^i_k C^i_b \Delta t}{2 \rho_i C_i} + \frac{2k_i \Delta t}{\rho_i C_i} \left( \frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right) - \frac{k_i \Delta t}{2 \rho_i C_i} \delta^2_z, 
\]  
(4.18)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
where $\delta^*_z = \frac{(u_i)_{ijk} - 2(u_i)_{ijk} + (u_i)_{ijk-1}}{\Delta z^2}$. By substituting Eq. (4.18) into Eq. (4.17a), we obtain

$$
[2\rho_c C_i + W_b'C_b^i \Delta t + 4k_i \Delta t (\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}) - k_i \Delta t \delta^*_z ][(u_i)_{ijk}^{n+1}]^{(I+1)} = [2\rho_c C_i + W_b'C_b^i \Delta t + 4k_i \Delta t (\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}) - k_i \Delta t \delta^*_z ][(u_i)_{ijk}^{n+1}]^{(I)} - \omega \cdot \Re \cdot 2\rho_c C_i.
$$

That is,

$$
-\frac{k_i \Delta t}{\Delta z^2} [(u_i)_{ijk}^{n+1}]^{(I+1)} + [2\rho_c C_i + W_b'C_b^i \Delta t + 4k_i \Delta t (\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}) + \frac{2k_i \Delta t}{\Delta z^2}] [(u_i)_{ijk}^{n+1}]^{(I+1)}
-\frac{k_i \Delta t}{\Delta z^2} [(u_i)_{ijk}^{n+1}]^{(I+1)} - \omega \cdot \Re \cdot 2\rho_c C_i,
$$

where

$$
\Re \cdot 2\rho_c C_i = (2\rho_c C_i + W_b'C_b^i \Delta t)[(u_i)_{ijk}^{n+1}]^{(I)} + (-2\rho_c C_i + W_b'C_b^i \Delta t)(u_i)_{ijk}^n
- k_i \Delta t (\frac{[(u_i)_{ijk}^{n+1}]^{(I)} + [(u_i)_{ijk}^{n+1}]^{(I)} - 2[(u_i)_{ijk}^{n+1}]^{(I)} + [(u_i)_{ijk}^{n+1}]^{(I)} - 2[(u_i)_{ijk}^{n+1}]^{(I)}]}{\Delta x^2}
+ \frac{[(u_i)_{ijk}^{n+1}]^{(I)} + [(u_i)_{ijk}^{n+1}]^{(I)} - 2[(u_i)_{ijk}^{n+1}]^{(I)}}{\Delta y^2})

- k_i \Delta t (\frac{(u_i)_{ijk}^{n+1} + (u_i)_{ijk}^{n+1} - 2(u_i)_{ijk}^{n+1}}{\Delta x^2}
+ \frac{(u_i)_{ijk}^{n+1} + (u_i)_{ijk}^{n+1} - 2(u_i)_{ijk}^{n+1}}{\Delta y^2})
- 2\Delta t Q_{ijk}^{n+1/2}.
$$

(4.21)

It should be pointed out that there are only three unknowns, $[(u_i)_{ijk}^{n+1}]^{(I+1)}$, $[(u_i)_{ijk}^{n+1}]^{(I+1)}$, and $[(u_i)_{ijk}^{n+1}]^{(I+1)}$, in Eq. (4.20) at the $(I+1)$th loop. Combining Eq. (4.20) with
boundary conditions, Eq. (4.10c) and Eq. (4.10d), we obtain a tridiagonal linear system as follows:

\[
\begin{bmatrix}
-a_1 + b_1 & -c_1 & 0 & \cdots & 0 \\
-a_2 & b_2 & -c_2 & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & 0 \\
0 & \cdots & -a_{N_z-1} & b_{N_z-1} & -c_{N_z-1} \\
0 & \cdots & 0 & -a_{N_z} & b_{N_z} - c_{N_z} \\
\end{bmatrix}
\begin{bmatrix}
[u(t)_{0}^{n+1}]^T \\
[u(t)_{1}^{n+1}]^T \\
\vdots \\
[u(t)_{N_z-1}^{n+1}]^T \\
[u(t)_{0}^{n+1}]^T \\
\end{bmatrix}
= \begin{bmatrix}
d_1 \\
d_2 \\
\vdots \\
d_{N_z-1} \\
d_{N_z} \\
\end{bmatrix}
\]

(4.22a)

where

\[
a_k = \frac{k_i \Delta t}{\Delta z^2}, \quad k = 1, 2, \cdots, N_z, \quad (4.22b)
\]

\[
b_k = (2\rho_i C_i + \omega W_b C'_b) \Delta t + 4k_i \Delta t \left( \frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right) + \frac{2k_i \Delta t}{\Delta z^2}, \quad k = 1, 2, \cdots, N_z, \quad (4.22c)
\]

\[
c_k = \frac{k_i \Delta t}{\Delta z^2}, \quad k = 1, 2, \cdots, N_z, \quad (4.22d)
\]

\[
d_k = -\frac{2k_i \Delta t}{\Delta z^2} [(u(t)_{0}^{n+1})_{i+k-1}]^{(t)} + (2\rho_i C_i + \omega W_b C'_b) \Delta t + 4k_i \Delta t \left( \frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right) + \frac{2k_i \Delta t}{\Delta z^2} + \frac{2k_i \Delta t}{\Delta z^2} [(u(t)_{0}^{n+1})_{i+k}]^{(t)}
\]

\[-\frac{k_i \Delta t}{\Delta z^2} [(u(t)_{0}^{n+1})_{i+k+1}]^{(t)} - \omega \cdot \Re \cdot 2\rho_i C_i, \quad k = 1, 2, \cdots, N_z. \quad (4.22e)
\]

Hence, the linear system of Eq. (4.22a) can be solved by using the convenient Thomas algorithm introduced in Section 2.4 to obtain the temperatures along z-axis $(k)$ for a given $(i, j)$ pair. Then we can obtain the temperature distribution in the whole 3D skin structure by performing the above calculation for each $(i, j)$ pair, where $0 \leq i \leq N_x, \ 0 \leq j \leq N_y$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
4.5 Algorithm

The algorithm for calculating the required laser intensity $P_0$ in order to obtain the pre-specified temperatures at given locations on the skin surface after a pre-specified laser exposure time is described as follows:

Step 1. Pre-specify the temperature elevations $\theta_{pre}^i$ at given $(M+1)$ grid points $i = 0, 1, \ldots, M$, on the skin surface, and pre-specify the laser exposure pattern for obtaining these pre-specified temperatures.

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

Step 3. Guess the wall temperature of the blood vessel $\theta_w^m$, and obtain the blood temperature $\theta_b^m$, and then obtain the temperature distribution $\vec{u}_{cal}$ in the entire 3D skin structure with the interfacial equations and the initial and boundary conditions.

Step 4. Update the wall temperature of the blood vessel, $\theta_w^m$.

Step 5. Repeat steps 3 and 4 until a convergent solution, $\vec{u}_{cal}$, at time level $n+1$ is obtained.


Step 7. Repeat steps 3 to 6 until the following criterion for convergence is satisfied:

$$\frac{|S(P_0^{(j+1)}) - S(P_0^{(j)})|}{S(P_0^{(j+1)})} < \varepsilon.$$  (4.23)
CHAPTER FIVE

NUMERICAL RESULTS AND DISCUSSION

5.1 Laser Irradiation Pattern

To test the applicability of the mathematical model proposed in Chapter 3 and the
numerical method developed in Chapter 4, we consider three examples in this chapter.
The laser irradiation pattern for these three examples is designed as follows:

1. Laser is focused on the center of the skin surface;
2. Laser is shut down while the temperature of the skin center is greater than 8 °C;
3. Laser is turned on while the temperature of the skin center is lower than 4 °C;
4. The whole process lasts 400 seconds.

5.2 Description of the Examples

Three examples are considered as follows: In Example 1, we consider a 3D triple-
layered skin structure embedded with a multi-level artery, as shown in Figure 5.1. In
Example 2, we consider a 3D triple-layered skin structure embedded with two
countercurrent blood vessels: a multi-level artery and a multi-level vein, as shown in
Figure 5.2. In both Example 1 and Example 2, the blood is considered to be in the steady
state. In Example 3, we consider a 3D triple-layered skin structure embedded with a
multi-level artery and a multi-level vein, while the blood is considered to be in the
dynamic state (i.e., Eqs. (3.6a)-(3.6c)). For each of these three examples, we consider two
cases. In Case 1, we assume that there is no heat convection between the skin surface and the environment. In Case 2, we consider that there is heat convection between the skin surface and the environment.

Figure 5.1 Geometry of a 3D skin structure embedded with a multi-level artery

Figure 5.2 Geometry of a 3D skin structure embedded with a multi-level artery and a multi-level vein

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The parameter values of the tissue and the blood thermal properties are listed in Table 5.1. These data are obtained from the literature (see [Liu 1999] [Huang 1996] [Welch 1995]). The dimensions of the skin structure and the blood are given in Table 5.2. A mesh of $50 \times 50 \times 1208$ in $(x, y, z)$ is used in the computation. The blood temperature elevation at the entrance of the artery is assumed to be $1 ^\circ C$.

The elevated temperatures are pre-specified to be $8 ^\circ C$ at the center of the skin surface which the laser focuses on and $2 ^\circ C$ at the midpoint on each edge of the surface at $t = 400$ s. The reason that these locations are chosen is because the highest temperature is assumed to be around the center of the skin surface, and it is necessary to have the temperature in the perimeter below a certain threshold so as not to cause damage to the normal tissue. In addition, the temperature on these locations can be easily measured.

Table 5.1 Thermal properties of the skin and the blood.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>W/cm$^2$ $^\circ C$</td>
<td>0.2</td>
</tr>
<tr>
<td>$C_1$</td>
<td>J/g $^\circ C$</td>
<td>3.6</td>
</tr>
<tr>
<td>$C_2$</td>
<td>J/g $^\circ C$</td>
<td>3.4</td>
</tr>
<tr>
<td>$C_3$</td>
<td>J/g $^\circ C$</td>
<td>3.06</td>
</tr>
<tr>
<td>$C_{b1}$</td>
<td>J/g $^\circ C$</td>
<td>0</td>
</tr>
<tr>
<td>$C_{b2}$</td>
<td>J/g $^\circ C$</td>
<td>4.2</td>
</tr>
<tr>
<td>$C_{b3}$</td>
<td>J/g $^\circ C$</td>
<td>4.2</td>
</tr>
<tr>
<td>$C_B$</td>
<td>J/cm$^3$ $^\circ C$</td>
<td>4.134</td>
</tr>
<tr>
<td>Symbol</td>
<td>Unit</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$h$</td>
<td>J/cm$^2$ °C</td>
<td>0.001</td>
</tr>
<tr>
<td>$k_1$</td>
<td>W/cm °C</td>
<td>0.0026</td>
</tr>
<tr>
<td>$k_2$</td>
<td>W/cm °C</td>
<td>0.0052</td>
</tr>
<tr>
<td>$k_3$</td>
<td>W/cm °C</td>
<td>0.0021</td>
</tr>
<tr>
<td>$w_b^1$</td>
<td>g/cm$^3$s</td>
<td>0</td>
</tr>
<tr>
<td>$w_b^2$</td>
<td>g/cm$^3$s</td>
<td>0.0005</td>
</tr>
<tr>
<td>$w_b^3$</td>
<td>g/cm$^3$s</td>
<td>0.0005</td>
</tr>
<tr>
<td>$\rho_1$</td>
<td>g/cm$^3$</td>
<td>1.2</td>
</tr>
<tr>
<td>$\rho_2$</td>
<td>g/cm$^3$</td>
<td>1.2</td>
</tr>
<tr>
<td>$\rho_3$</td>
<td>g/cm$^3$</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>1/cm</td>
<td>1.8</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>1/cm</td>
<td>1.8</td>
</tr>
<tr>
<td>$\alpha_3$</td>
<td>1/cm</td>
<td>1.8</td>
</tr>
<tr>
<td>$Reff_1$</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>$Reff_2$</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>$Reff_3$</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>$v_1$</td>
<td>m/s</td>
<td>0.08</td>
</tr>
<tr>
<td>$\dot{P}$</td>
<td>1/s</td>
<td>0.5 x 10$^{-3}$</td>
</tr>
</tbody>
</table>
Table 5.2 Geometry parameters for the skin structure and the embedded blood vessels

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Unit</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Bi = \frac{\alpha}{k_3}$</td>
<td></td>
<td>95.23</td>
</tr>
<tr>
<td>$L_1$</td>
<td>cm</td>
<td>0.008</td>
</tr>
<tr>
<td>$L_2$</td>
<td>cm</td>
<td>0.2</td>
</tr>
<tr>
<td>$L_3$</td>
<td>cm</td>
<td>1</td>
</tr>
<tr>
<td>$L^1_b$</td>
<td>cm</td>
<td>0.4</td>
</tr>
<tr>
<td>$L^2_b$</td>
<td>cm</td>
<td>0.28</td>
</tr>
<tr>
<td>$L^3_b$</td>
<td>cm</td>
<td>0.2</td>
</tr>
<tr>
<td>$NX, NY$</td>
<td>cm</td>
<td>1</td>
</tr>
<tr>
<td>$NL^1_b, NW^1_b$</td>
<td>cm</td>
<td>0.1</td>
</tr>
<tr>
<td>$NL^2_b, NW^2_b$</td>
<td>cm</td>
<td>0.08</td>
</tr>
<tr>
<td>$NL^3_b, NW^3_b$</td>
<td>cm</td>
<td>0.06</td>
</tr>
<tr>
<td>$\Delta x$</td>
<td>cm</td>
<td>0.02</td>
</tr>
<tr>
<td>$\Delta y$</td>
<td>cm</td>
<td>0.02</td>
</tr>
<tr>
<td>$\Delta z$</td>
<td>cm</td>
<td>0.001</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>s</td>
<td>0.1</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>cm</td>
<td>0.01</td>
</tr>
<tr>
<td>$\omega$</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
5.3 Results and Discussion

5.3.1 Example 1

In this example, we consider a 3D triple-layered skin structure embedded with a multi-level artery, and the centerline of the artery at the first level is located at the x-z plane with $y = 0.5$ cm. The blood is in the steady state.

5.3.1.1 Case 1

In this case, we assume that there is no heat convection between the skin surface and the surroundings (i.e., $h = 0$). We start with an initial value $P_0$ of 1.2 W/cm and $\Delta P_0$ is initialized to be one percent of $P_0$, which is 0.012 W/cm. We optimize $P_0$ based on the algorithm described in Section 4.5, and $\Delta P_0$ is updated based on two consecutive values of $P_0$. Figure 5.3 shows $P_0$ and sum of the least squares versus number of iterations. It can be seen that $P_0$ converges to 1.1947 W/cm. Thus, we use the convergent value of $P_0$ to obtain the temperature distribution in the 3D skin structure.

Figures 5.4 and 5.5 show the temperature elevation profiles at various times along the lines $y = 0.5$ cm and $x = 0.5$ cm on the skin surface. The temperature elevation at the center rises to 8 °C while at the edge it rises gradually to 2 °C.

Figures 5.6 and 5.7 show the contours of the temperature elevation at various times in the $xz$-cross section at $y = 0.5$ cm and the $yz$-cross section at $x = 0.5$ cm. Figure 5.8 shows the profiles of temperature elevations at various times along the depth (the $z$-direction) at the center of the skin surface. The hot spot is located in the subcutaneous layer instead of the skin surface. This is because the epidermis layer is very thin and the diffusivity of the dermis layer is about twice as large as that of the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
subcutaneous layer. As a result, the heat is stored in the subcutaneous layer. Fortunately, the temperature elevation on the lateral sides is below 2°C, ensuring that the normal tissue is not damaged.

Figure 5.3 Laser intensity and sum of the least squares versus number of iterations for the case 1 of example 1
Figure 5.4 Profiles of temperature elevations at various times in the x-direction at y = 0.5 cm on the skin surface for the case 1 of example 1.
Figure 5.5 Profiles of temperature elevations at various times in the y-direction at \( x = 0.5 \) cm on the skin surface for the case 1 of example 1.
Figure 5.6 Contours of the temperature elevations at various times in the cross section of the xz-plane at $y = 0.5$ cm for the case 1 of example 1
Figure 5.7 Contours of the temperature elevations at various times in the cross section of the yz-plane at x = 0.5 cm for the case 1 of example 1.
Figure 5.8 Profiles of temperature elevations at various times in the z-direction (depth) at the center of the skin surface for the case 1 of example 1.
5.3.1.2 Case 2

In this case, we consider the heat convection on the skin surface \((h = 0.001 \text{ W/cm}^2 \text{ [Deng 2004]})\) and assume that the surrounding air temperature near the skin surface is 37 °C. We first try \(P_0 = 1.1947 \text{ W/cm}\), which is obtained in the previous case. Figure 5.9 shows the profiles of temperature elevations and contours after \(t = 400\) s. The temperature elevations at the center and the edge of the skin surface are 7.88 °C and 1.7°C, respectively. These are obviously lower than the required temperature elevations. We then optimize \(P_0\) again based on the algorithm described in Section 4.5, and obtain \(P_0 = 1.3166 \text{ W/cm}\). Figure 5.10 shows \(P_0\) and sum of the least squares versus number of iterations for this case. Figure 5.11 shows the profiles of temperature elevations and contours after \(t = 400\) s with \(P_0 = 1.3166 \text{ W/cm}\). It can be seen from Figure 5.11 that the temperature elevations at the center and at the edge of the skin surface reach 8 °C and 2 °C, respectively.
Figure 5.9 Profiles of temperature elevations and contours at $t = 400$ s and $P_0 = 1.1947\text{W/cm}$ for the case 2 of example 1.
Figure 5.10 Laser intensity and sum of the least squares versus number of iterations for the case 2 of example 1
Figure 5.11 Profiles of temperature elevations and contours at $t = 400$ s and $P_0 = 1.3166$ W/cm for the case 2 of example 1.
5.3.2 Example 2

In this example, we consider a 3D triple-layered skin structure embedded with a multi-level artery and a multi-level vein. The centerline of the artery at the first level is located at the xz-plane with $y = 0.4$ cm, and the centerline of the vein at the first level is located at the center of the xz-plane with $y = 0.56$ cm. The blood is in the steady state.

5.3.2.1 Case 1.

In this case, we first try the power intensity, $P_0 = 1.1947$ W/cm, which is the optimal laser intensity for Case 1 in Example 1 where only the artery is considered in the skin structure. Figure 5.12 shows the temperature elevation profiles and contours after $t = 400$ s. The temperature elevations at the center of the skin surface and the four midpoints on the edges of the skin surface at (0.5 cm, 0 cm), (0.5 cm, 1 cm), (0 cm, 0.5 cm), (1 cm, 0.5 cm) rise to 7.95 °C and 1.98 °C, 1.97 °C, 1.97 °C, respectively, which are little lower than the required temperature elevations. In order to obtain an optimal temperature distribution for this case, we optimize $P_0$ based on the algorithm described in Section 4.5 with an initial value $P_0$ of 1.2 W/cm and the initial $\Delta P_0$ to be one percent of $P_0$. Figure 5.13 shows $P_0$ and sum of the least squares versus iteration, respectively, for this case. It can be seen that $P_0$ is convergent to 1.1993 W/cm, which is slightly higher than 1.1947 W/cm. This can be interpreted because the vein is carrying some heat out of the region. We then use this convergent value of $P_0$ to compute the temperature distribution in the entire 3D skin structure.

Figures 5.14 and 5.15 show the temperature elevation profiles at various times along lines $y = 0.5$ cm and $x = 0.5$ cm on the skin surface. At $t = 352$ s, the temperature
elevation at the center of the skin surface rises to 8 °C for the first time. The laser is then turned off and on according to the laser irradiation pattern in order that the temperature elevation at the edge rises to 2 °C. Finally, at $t = 400$ s, the temperature elevations at the pre-specified locations satisfies the pre-specified temperature elevations.

Figures 5.16-5.19 show the contours of the temperature elevation distributions at various times in the xz-cross sections at $y = 0.4$ cm where the artery is located, at $y = 0.56$ cm where the vein is located, at $y = 0.5$ cm, and the yz-cross section at $x = 0.5$ cm, respectively. In Figure 5.19, we can see that at $t = 100$ s and $t = 200$ s the temperatures on the right-hand side where the vein is located are lower than those on the left-hand side where the artery is located; however, at $t = 300$ s and $t = 400$ s the temperatures on the right-hand side are higher than those on the left-hand side. Result coincides with the fact that the artery brings the heat in and the vein takes the heat out. It can also be seen from Figure 5.20, which shows the temperature elevation profiles with various times along the depth (the z-direction) at three locations, that the blood temperature along the depth (the z-direction) in the vein kept rising due to that the blood has been heated, and finally it is higher than that in the artery with the same level in the 3D skin structure.
Figure 5.12 Profiles of temperature elevations and contours at $t = 400$ s and $P_0 = 1.1947$ W/cm for the case 1 of example 2
Figure 5.13 Laser intensity and sum of the least squares versus number of iterations for the case 1 of example 2
Figure 5.14 Profiles of temperature elevations at various times in the x-direction at $y = 0.5$ cm on the skin surface for the case 1 of example 2.
Figure 5.15 Profiles of temperature elevations at various times in the y-direction at x = 0.5 cm on the skin surface for the case 1 of example 2
Figure 5.16 Contours of the temperature elevations at various times in the cross section of the xz-plane at y = 0.4 cm for the case 1 of example 2.
Figure 5.17 Contours of the temperature elevations at various times in the cross section of the xz-plane at y = 0.56 cm for the case 1 of example 2.
Figure 5.18 Contours of the temperature elevations at various times in the cross section of the xz-plane at y = 0.5 cm for the case 1 of example 2
Figure 5.19 Contours of the temperature elevations at various times in the cross section of the yz-plane at x = 0.5 cm for the case 1 of example 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.20 Profiles of temperature elevations at various times along the depth (the z-direction) at three locations for the case 1 of example 2.
5.3.2.2 Case 2.

In this case, we first try $P_0 = 1.1993$ W/cm, which is obtained in the previous case. Figure 5.21 shows the temperature elevation profiles and contours after $t = 400$ s. The temperature elevations at the center and the edge of the skin surface arrive at 7.91 °C and 1.75 °C, 1.73 °C, 1.74 °C, 1.73 °C, respectively, which are obviously lower than the required temperature elevations. Using our algorithm we obtain the optimal laser intensity $P_0 = 1.3450$ W/cm for this case. Figure 5.22 shows $P_0$ and sum of the least squares versus number of iterations for this case.

Figure 5.23 shows the temperature elevation profiles and contours after 400 seconds with $P_0 = 1.3450$ W/cm. It can be seen from Figure 5.23 that the temperature elevations at the center and the edges of the skin surface are 8 °C and 2 °C, respectively.
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.21 Profiles of temperature elevations and contours at $t = 400$ s and $P_0 = 1.1993$ W/cm for the case 2 of example 2
Figure 5.22 Laser intensity and sum of the least squares versus number of iterations for the case 2 of example 2.
Figure 5.23 Profiles of temperature elevations and contours at $t = 400$ s and $P_0 = 1.3450$ W/cm for the case 2 of example 2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
5.3.3 Example 3

In this example, the skin structure is the same as in Example 2. However, we consider that the blood is in the dynamic state.

5.3.3.1 Case 1.

In this case, we first try the power intensity, $P_0 = 1.1993$ W/cm, which is the optimal laser intensity for Case 1 in Example 2 where the blood is considered to be in the steady state. Figure 5.24 shows the temperature elevation profiles and contours after $t = 400$ s. The temperature elevations at the center of the skin surface and the four midpoints on the edges of the skin surface at $(0.5 \text{ cm}, 0 \text{ cm})$, $(0.5 \text{ cm}, 1 \text{ cm})$, $(0 \text{ cm}, 0.5 \text{ cm})$, $(1 \text{ cm}, 0.5 \text{ cm})$ rise to $7.97$ °C and $1.73$ °C, $1.73$ °C, $1.72$ °C, $1.72$ °C, respectively.

The temperature at the center of the skin surface almost arrives $8$ °C for the first time, but the temperature elevations at the midpoints on the edges of the skin surface are lower than the pre-specified temperatures. This means that the laser power needs to be optimized. In order to obtain an optimal temperature distribution for this case, we optimize $P_0$ based on the algorithm described in Section 4.5 with an initial value $P_0$ of $1.2$ W/cm and the initial $\Delta P_0$ to be one percent of $P_0$. Figure 5.25 shows $P_0$ and sum of the least squares versus iteration, respectively, for this case. It can be seen that $P_0$ is convergent to $1.3110$ W/cm. We then obtain the optimal temperature distribution in the entire 3D skin structure based on this convergent value of $P_0$.

At $t = 227$ s, the temperature elevation at the center of the skin surface rises to $8$ °C for the first time. The laser is then turned off and on according to the laser irradiation pattern in order that the temperature elevation at the edge rises to $2$ °C. Finally, at
$t = 400 \, s$, the temperature elevations at the pre-specified locations satisfy the pre-specified temperature elevations. Figures 5.26 and 5.27 show the temperature elevation profiles at various times along lines $y = 0.5 \, \text{cm}$ and $x = 0.5 \, \text{cm}$ on the skin surface.

Figures 5.28-5.31 show the contours of the temperature elevation distributions at various times in the $xz$-cross sections at $y = 0.4 \, \text{cm}$ where the artery is located, at $y = 0.56 \, \text{cm}$ where the vein is located, at $y = 0.5 \, \text{cm}$, and the $yz$-cross section at $x = 0.5 \, \text{cm}$, respectively. In Figure 5.31, we can see that the temperatures on the right-hand side where the vein is located are lower at the beginning but higher later than those on the left-hand side where the artery is located. This result is same as the conclusion of Example 2, which coincides with the fact that the artery brings the heat in and the vein takes the heat out. Figure 5.32 shows the temperature elevation profiles with various times along the depth (the $z$-direction) at three locations. Similarly as in Example 2, we can see that in the vein the blood temperature along the depth (the $z$-direction) keeps rising because of the heat taken out of the tissue. However, it should be pointed out that, in both artery and vein, the blood temperatures decrease significantly along the blood flowing direction, whereas there is no such phenomenon in Example 2 when the blood is considered to be in the steady state.
Figure 5.24 Profiles of temperature elevations and contours at $t = 400$ s and $P_0 = 1.1993$ W/cm for the case 1 of example 3
Figure 5.25 Laser intensity and sum of the least squares versus number of iterations for the case 1 of example 3
Figure 5.26 Profiles of temperature elevations at various times in the x-direction at $y = 0.5$ cm on the skin surface for the case 1 of example 3
Figure 5.27 Profiles of temperature elevations at various times in the y-direction at x = 0.5 cm on the skin surface for the case 1 of example 3.
Figure 5.28 Contours of the temperature elevations at various times in the cross section of the xz-plane at y = 0.4 cm for the case 1 of example 3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.29 Contours of the temperature elevations at various times in the cross section of the xz-plane at $y = 0.56$ cm for the case 1 of example 3
Figure 5.30 Contours of the temperature elevations at various times in the cross section of the xz-plane at $y = 0.5$ cm for the case 1 of example 3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.31 Contours of the temperature elevations at various times in the cross section of the yz-plane at $x = 0.5$ cm for the case 1 of example 3.
Figure 5.32 Profiles of temperature elevations at various times along the depth (the z-direction) at three locations for the case 1 of example 3
5.3.3.2 Case 2.

We first try to find the optimal laser intensity for this case based on the algorithm described in Section 4.5 with an initial value $P_0$ of 1.2 W/cm and the initial $\Delta P_0$ to be one percent of $P_0$. Because the calculated temperatures can not be close to the pre-specified temperatures under the laser irradiation pattern we design, the program does not converge. In order to overcome this problem, we adjust the laser beam width $\sigma$ from 0.01 cm to 0.015 cm and obtain the optimal laser intensity $P_0 = 1.5552$ W/cm for this case.

Figure 5.33 shows $P_0$ and sum of the least squares versus iteration for this case. Figure 5.34 shows the temperature elevation profiles and contours after 400 seconds with $P_0 = 1.5552$ W/cm. It can be seen from Figure 5.34 that the temperature elevations at the center and the edges of the skin surface are 8 °C and 2 °C, respectively.
Figure 5.33 Laser intensity and sum of the least squares versus number of iterations for the case 2 of example 3
Figure 5.34 Profiles of temperature elevations and contours at $t = 400 \text{ s}$ and $P_0 = 1.5552 \text{ W/cm}$ for the case 2 of example 3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
5.4 Checking the Grid Independence

In order to prove that our algorithm is independent of grid size, we employ four different meshes of \(50 \times 50 \times 1208\), \(50 \times 100 \times 1208\), \(100 \times 100 \times 1208\), and \(50 \times 50 \times 2416\) in the computation for Case 1 of Example 1. Figure 5.35(a) and Figure 5.35(b) show the four profiles of temperature elevations at \(t = 400\) s in the y-direction at \(x = 0.5\) cm and in the x-direction at \(y = 0.5\) cm on the skin surface, respectively. Figure 5.35(c) shows the four temperature elevation profiles at \(t = 400\) s in the z-direction (along the depth) at the center of the skin surface. It can be seen from Figures 5.35(a) to 5.35(c) that there are no significant differences among the solutions for the different grids. This indicates that our scheme is grid independent.
Figure 5.35 Four profiles of temperature elevations at $t = 400$ s for case 1 of example 1 (a) in the y-direction at $x = 0.5$ cm and (b) in the x-direction at $y = 0.5$ cm on the skin surface, and (c) in the z-direction (depth) at the center of the skin surface.
CHAPTER SIX

CONCLUSION AND FUTURE STUDIES

6.1 Conclusion

In this study, we have developed a numerical method for obtaining an optimal temperature distribution in a 3D triple-layered rectangular skin structure embedded with multi-level blood vessels, where the length and size of the blood vessels as well as the mass flow of blood were determined based on the constructal theory of multi-scale tree-shaped heat exchangers. The method consisted of pre-specifying the temperatures to be obtained at the center and at the midpoints of the edges on the skin surface, and optimizing the laser power by using the least squares method in conjunction with a numerical solution of the 3D Pennes bioheat equation for tissue coupled with the energy balance equations for blood vessels. The preconditioned Richardson iteration and Thomas algorithm were employed to speed up and simplify the computation.

We tested on three examples in each of which two cases were considered. In Example 1, we considered a 3D triple-layered skin structure embedded with a multi-level artery; in Example 2 and Example 3, we considered a 3D triple-layered skin structure embedded with a multi-level artery and a multi-level vein. In both Example 1 and Example 2, the blood was considered to be in the steady state, while in Example 3, the blood was considered to be in the dynamic state. In one of the two cases, we assumed there is no heat convection between the skin surface and the surrounding air; in the other
case, we considered there is a heat convection between the skin surface and the surrounding air. We conclude that (1) a stronger laser intensity is needed to satisfy the pre-specified temperature requirements if there is a heat convection between the skin surface and the surrounding air; (2) the optimal laser intensity is higher for the skin structure embedded with both a multi-level artery and a multi-level vein; (3) the temperature in the blood decreases significantly along the blood flowing direction in both artery and vein when the blood is considered to be in the dynamic state. The numerical examples showed that the method is applicable and efficient. Results could be useful for certain types of hyperthermia cancer treatment, such as skin cancer.

6.2 Future Studies

Future studies can focus on the following aspects:

(1) In our model, the blood vessels are considered to be rectangular, so that the grid points can be located on the vessels. In this way, it is convenient to calculate the temperatures on the blood vessel from the model. A more realistic approach could consider the cross section of the blood vessels to be round. Also, the skin region to be considered in the model could be either cylindrical or rectangular. This will, of course, require a new grid point system, and thus the problem will be more challengable to solve.

(2) The skin surface is considered to be flat in this dissertation. However, the tumor could be a mound shape. Thus, it is interesting to consider the skin surface to be a mound shape.

(3) Instead of using laser irritation, microwave is a good alternative for heat source. Generally, microwave sources have been expected to be less
expensive than lasers for a given power, but to be more limited in range and/or energy density.

(4) In this dissertation, a large number of grid points are taken in the $z$-direction because the epidermis layer is very thin. To reduce the number of grid points along the depth, a higher-order compact finite-difference method may be employed.
APPENDIX A

SOURCE CODE FOR EXAMPLE 1
Author: Xingui Tang
Date: 4/2/2006

Description: This program is used to calculate the optimal temperature distribution in a 3D triple-layered skin structure embedded with a multi-level artery. The models to describe the thermal behavior in tissue and blood are modified Penns model and energy balance equations. The blood is considered to be in steady state in this example. We worked on two cases in the example, one of them does not consider the heat convection between the skin surface and the environment and the other considers. Use the macro definition of "CONVECTION" to switch between Case 1 and Case 2 of Example 1.

**********************
#include "string.h"
#include <math.h>
#include <stdio.h>
#include "memory.h"

#define SCREEN_OUT
#define CONVECTION // This is a switch for Case 1 and Case 2.

#define M 5
#define NZ1 8
#define NZ2 208
#define NZ3 1208
#define NX 50
#define NY 50
#define LX1 5
#define LY1 5
#define LZ1 400
#define LX2 28
#define LY2 4
#define LZ2 80
#define LX3 3
#define LY3 3
#define LZ3 200

double deltaT = 0.1;
double deltaX = 0.02, deltaY = 0.02, deltaZ = 0.001;

double L1, L2, L3;
double Lb1, Lb2, Lb3;
double P1, P2, P3;
double F1, F2, F3;
double M1, M2, M3;
double CB = 4.134;
double Cb1 = 0.0, Cb2 = 4.2, Cb3 = 4.2;
double v1 = 8;
double alpha = 0.2;
double Pdot = 0.5e-3;
double p1 = 1.2, p2 = 1.2, p3 = 1.0;
double C1 = 3.6, C2 = 3.4, C3 = 3.06;
double k1 = 0.0026, k2 = 0.0052, k3 = 0.0021;
double Wb1 = 0.0, Wb2 = 0.0005, Wb3 = 0.0005;
double Alpha1 = 1.8, Alpha2 = 1.8, Alpha3 = 1.8;
double Sigma = 0.01;
double Ref1 = 0.1, Ref2 = 0.1, Ref3 = 0.1;

double Hf = 0.001;
double Tf = 0; //environment temperature

int centerX = NX/2;
int centerY = NY/2;

const double pai = 3.14159265358979;
const double omega = 1.0;
double Bi = alpha/k3;
double factor1, factor2, factor3;

double THETA0 = 1;

double (*Q1)[NY+1][NZ3+1];
double (*Q2)[NY+1][NZ3+1];
double (*Q3)[NY+1][NZ3+1];

double (*Tb)[NY+1][NZ3+1];
double (*Tt_n1)[NY+1][NZ3+1];
double (*Tt_n1_1)[NY+1][NZ3+1];

double *Tb1;
double *Tb2;
double *Tb3;
double *Tb1_n1;
double *Tb2_n1; double *Tb3_n1;
double *Tv1_n1;
double *Tv2_n1;
double *Tv3_n1;

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

int X1, X2, X3, X4, X5, X6, Y1, Y2, Y3, Y4, Y5, Y6, Z1, Z2, Z3; //grid points index on blood vessel

int i, j, z;
int t, I, loopP;

bool bPowerOn;

void initialize();
int CalcAll(double P0, bool dP);
int getTve_blood();
int CalcTb();
double CalcTt();
void Reset(void);
void InitQ(double P0);
double CalcNewP(double *Tm, double *Tm0, double *Tm1, double *Tm2, double P0, double deltaP);
int getTm(double *Tm);
void setVesselBorder();
void AdjustPower(double P0);
void clearMem();

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
101

```c
void writeSquareXZ(int x0, int z0, int x1, int t, int I);
void writeSquareYZ(int y0, int x0, int y1, int t, int I);
void writeSquareXY(int x0, int y0, int x1, int y1, int z, int I);
void writeZCenter(int t);
void writeQ(double P, double sigma);
void writeAll(int t, int I);
void writeLinearSys(int x, int y, int I, int I);
void writeLog(char *line);
bool PrintOut(int tm);

char outPath[255] = "/home/username/"

#define numT 37
int outT[numT] = {-1, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500,
                  550, 600, 650, 700, 750, 800, 850, 900, 950, 1000,
                  1050, 1100, 1150, 1200, 1250, 1300, 1350, 1400, 1450, 1500,
                  1550, 1600, 1650, 1700, 1750, 1800};

double Tm_pre[5] = {8, 2, 2, 2, 2}; // pre-specified surface temperature;
bool bReachTop;

int TOTAl_T = 400;
double Err_I = 0.001;  //for I loop
double Err_P = 0.01;  //for P loop

void testP()
{
    double oldSigma = Sigma;
    double p[10] = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0};
    double sigma[10] = {0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1};

    for(int j=0;j<10;j++)
        for(int i=0;i<10;i++)
            {
                Sigma = sigma[j];
                InitQ(p[i]);
                writeQ(p[i], Sigma);
            }
    return;
}

int main(void)
{
    double Tm0[5]; //calculated surface temperatures - first run
    double Tm1[5]; //calculated surface temperatures - second run - with updated power level
    double P0, P1_LI;  //power level
    double deltaP; //power step

    double oldSP, newSP;
    int indexM;
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
#ifndef CONVECTION
writeLog("Calculation without convection");
#else
writeLog("Calculation with convection");
#endif

calculate(); //memory allocation & variable initialization

P0 = 1.2; //set initial power level;
loopP = 0;

sprintf(tmp, "Initial P:.5.61f", P0);
writeLog(tmp);

newSP = 0.0;
CalcAll(P0, 0);
getTm(Tm0);

deltaP = P0/100.0;
do{
oldSP = newSP;
P0 += deltaP;

//calculate tissue and blood temperature based on specified power level
CalcAll(P0, 0);
//retrieve calculated surface temperature
getTm(Tm1);

//calculate the next power level based on pre-specified temperature and calculated surface
//temperature
P1_LI = CalcNewP(Tm_pre, Tm0, Tm1, P0, deltaP);
deltaP = P1_LI - P0;
//break;

//calculate the S(P)
oldSP = 0;
newSP = 0;
for (indexM = 0; indexM < M; indexM++)
{
    oldSP = oldSP + (Tm_pre[indexM] - Tm0[indexM]) * (Tm_pre[indexM] - Tm0[indexM]);
    newSP = newSP + (Tm_pre[indexM] - Tm1[indexM]) * (Tm_pre[indexM] - Tm1[indexM]);
}

//////// 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]);

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
writeLog(tmp);

sprintf(tmp, "P%d:%6.61f; newSP:%8.61f; oldSP:%8.61f; (newSP-oldSP)/newSP: %8.61f
", loopP, pl, newSP, oldSP, (newSP-oldSP)/newSP);
writeLog(tmp);

for (indexM=0; indexM<M; indexM++)
    Tmo[indexM] = Tml[indexM];

loopP++;

if (newSP==0) break;

} while(fabs((newSP-oldSP)/newSP) > Err_P);

clearMem();
return 1;

//memory allocation & variable initialization

void initialize()
{
    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];

    Tt = new double[NX+1][NY+1][NZ3+1];
    Tt_n1 = new double[NX+1][NY+1][NZ3+1];
    Tt_n1_I = new double[NX+1][NY+1][NZ3+1];

    a = new double[NX+1][NY+1][NZ3+1];
    b = new double[NX+1][NY+1][NZ3+1];
    c = new double[NX+1][NY+1][NZ3+1];
    d = new double[NX+1][NY+1][NZ3+1];
    a0 = new double[NX+1][NY+1][NZ3+1];
    b0 = new double[NX+1][NY+1][NZ3+1];
    c0 = new double[NX+1][NY+1][NZ3+1];

    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));

    Lb1 = LZ1*deltaZ;
    Lb3 = LZ3*deltaZ;
    Lb2 = LX2*deltaX;

    L1 = NZ1*deltaZ;
    L2 = (NZ2-NZ1)*deltaZ;
    L3 = (NZ3-NZ2)*deltaZ;

    setVesselBorder();

    Tb1 = new double[LZ1+1];
    Tb2 = new double[LX2+1];
    Tb3 = new double[(LZ3+1)*2];
Tb1_n1 = new double[(LZ1+1)];
Tb2_n1 = new double[(LX2+1)];
Tb3_n1 = new double[(LZ3+1)*2];
Tv1_n1 = new double[(LZ1+1)];
Tv2_n1 = new double[(LX2+1)];
Tv3_n1 = new double[(LZ3+1)*2];

P1 = (LX1*deltaX + LY1*deltaY)*2;
P2 = (LY2*deltaY + LZ2*deltaZ)*2;
P3 = (LX3*deltaX + LY3*deltaY)*2;

F1 = (LX1*deltaX) * (LY1*deltaY);
F2 = (LY2*deltaY) * (LZ2*deltaZ);
F3 = (LX3*deltaX) * (LY3*deltaY);

M1 = v1 * F1;
M2 = 0.5 * M1; //
M3 = 0.5 * M1;// M3 = 0.25 * M1; //

factor1 = deltaZ*alpha*P1/(M1*CB);
factor2 = deltaX*alpha*P2/(M2*CB);
factor3 = deltaZ*alpha*P3/(M3*CB);

bPowerOn = true;
bReachTop = false;

writeLog(" Initialization...");
sprintf(tmp, "Lbl: 5.41f Lb2:% 5.41f Lb3:% 5.41f", Lb1, Lb2, Lb3);
writeLog(tmp);
sprintf(tmp, "L1:% 5.41f L2:% 5.41f L3: %5.41f", L1, L2, L3);
writeLog(tmp);
sprintf(tmp, "P1:% 5.41f P2:% 5.41f P3:% 5.41f", P1, P2, P3);
writeLog(tmp);
sprintf(tmp, "F1:% 5.41f F2:% 5.41f F3:% 5.41f", F1, F2, F3);
writeLog(tmp);
sprintf(tmp, "M1:% 5.41f M2:% 5.41f M3:% 5.41f", M1, M2, M3);
writeLog(tmp);
sprintf(tmp, "X1:%d X2:%d X3:%d X4:%d X5:%d X6:%d", X1, X2, X3, X4, X5, X6);
writeLog(tmp);
sprintf(tmp, "Y1:%d Y2:%d Y3:%d Y4:%d Y5:%d Y6:%d", Y1, Y2, Y3, Y4, Y5, Y6);
writeLog(tmp);
sprintf(tmp, "Z1:%d Z2:%d Z3:%d", Z1, Z2, Z3);
writeLog(tmp);
sprintf(tmp, "Reff1:%5.3lf Reff2:%5.3lf Reff3:%5.3lf", Reff1, Reff2, Reff3);
writeLog(tmp);
sprintf(tmp, "Alpha1:%5.3lf Alpha2:%5.3lf Alpha3:%5.3lf", Alpha1, Alpha2, Alpha3);
writeLog(tmp);
sprintf(tmp, "Sigma:%5.3lf", Sigma);
writeLog(tmp);

// 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<=NZ-1;z++)
 {

 Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\{
\begin{align*}
b_0[i][j][z] &= -(k_1 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ}); \\
a_0[i][j][z] &= 2 \cdot p_1 \cdot C_1 + Wb_1 \cdot Cb_1 \cdot \text{deltaT} + \\
&\quad (4 \cdot k_1 \cdot \text{deltaT}) \cdot (1.0/\text{deltaX} \cdot \text{deltaX} + 1.0/\text{deltaY} \cdot \text{deltaY}) + \\
&\quad (2 \cdot k_1 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ}); \\
c_0[i][j][z] &= -(k_1 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ});
\end{align*}
\}

b_0[i][j][NZ1]=k_1; \\
a_0[i][j][NZ1]=k_1+k_2; \\
c_0[i][j][NZ1]=k_2;

\text{for}(z=NZ1+1; z<=NZ2-1; z++) 
{ 
\begin{align*}
b_0[i][j][z] &= -(k_2 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ}); \\
a_0[i][j][z] &= 2 \cdot p_2 \cdot C_2 + Wb_2 \cdot Cb_2 \cdot \text{deltaT} + \\
&\quad (4 \cdot k_2 \cdot \text{deltaT}) \cdot (1.0/\text{deltaX} \cdot \text{deltaX} + 1.0/\text{deltaY} \cdot \text{deltaY}) + \\
&\quad (2 \cdot k_2 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ}); \\
c_0[i][j][z] &= -(k_2 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ});
\end{align*}
\}

b_0[i][j][NZ2]=k_2; \\
a_0[i][j][NZ2]=k_2+k_3; \\
c_0[i][j][NZ2]=k_3;

// third skin layer 
\text{for}(z=NZ2+1; z<=NZ3-1; z++) 
{ 
\begin{align*}
b_0[i][j][z] &= -(k_3 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ}); \\
a_0[i][j][z] &= 2 \cdot p_3 \cdot C_3 + (4 \cdot k_3 \cdot \text{deltaT}) \cdot (1.0/\text{deltaX} \cdot \text{deltaX}) + \\
&\quad (2 \cdot k_3 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ}); \\
c_0[i][j][z] &= -(k_3 \cdot \text{deltaT})/(\text{deltaZ} \cdot \text{deltaZ});
\end{align*}
\}

\}
\}

\text{return;}

void clearMem()
{ 
if(Q1) delete [] Q1; \\
if(Q2) delete [] Q2; \\
if(Q3) delete [] Q3; \\
if(Tt) delete [] Tt; \\
if(Tt_n1) delete [] Tt_n1; \\
if(Tt_n1_1) delete [] Tt_n1_1; \\
if(a) delete [] a; \\
if(b) delete [] b; \\
if(c) delete [] c; \\
if(d) delete [] d;
if(T b l) delete [] T b l;
if(T b2) delete [] T b2;
if(T b3) delete [] T b3;

if(Tb1_n1) delete [] Tb1_n1;
if(Tb2_n1) delete [] Tb2_n1;
if(Tb3_n1) delete [] Tb3_n1;

if(Tv1_n1) delete [] Tv1_n1;
if(Tv2_n1) delete [] Tv2_n1;
if(Tv3_n1) delete [] Tv3_n1;

writeLog("Memory released.");
return;

}

void setVesselBorder()
{
   X 5 = (NX-LX1)/2;
   X 6 = X 5 + LX1;
   X 1 = (NX-LX2)/2;
   X 2 = X 1 + LX3;
   X 4 = X 1 + LX2;
   X 3 = X 4 - LX3;

   Y 5 = (NY-LY1)/2;
   Y 6 = Y 5 + LY1;
   Y 3 = (NY-LY2)/2;
   Y 4 = Y 3 + LY2;
   Y 1 = (NY-LY3)/2;
   Y 2 = Y 1 + LY3;

   Z3 = NZ3 - LZ1;
   Z2 = Z3 - LZ2;
   Z1 = Z2 - LZ3;

   return;
}

int CalcAll(double P0, bool dp)
{
   double maxErr, oldE; //sum of square error of tissue temperatures

   Reset();
   InitQ(P0);

   //writeQ(P0, Sigma);

   t = 0;
   while((t*deltaT < TOTAL_T))
   {
      t++;
}
I = 0;
maxErr = 0.0;
oldE = 99999999.0;

do { //I iteration to calculate temperature at time level n+1
    I++; //interpolate blood vessel temperature based on tissue temperature
    getTv_blood();
    //calculate blood temperature based on given vessel temperature
    CalcTb();
    //interpolate blood temperature at for use of blood vessel interface equation
    //getTv_tissue();
    //Calculate tissue temperature
    maxErr = CalcTt();
    if(maxErr>=oldE)
    {
        writeLog("= = = unstable = = = = ");
        #ifdef SCREEN_OUT
        printf("= = = = = = 
");
        #endif
        writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
        break;
    }
    oldE = maxErr;

} while(maxErr>Err_I);

if(bPowerOn)
{
    if (dp)
    sprintf(tmp, "p%d t:%2d I:%d Err:%5.4f T0:%7.4lf T1:%7.4lf
T2:%7.4lf P:%7.4lf", loopP, t, I, maxErr, Tt_n1[NX/2][NY/2][0],
    Tt_n1[0][NY/2][0], Tt_n1[NX/4][NY/2][NZ3], P0);
    else
    sprintf(tmp, "p%d t:%2d I:%d Err:%5.4f T0:%7.4lf T1:%7.4lf
T2:%7.4lf P:%7.4lf", loopP, t, I, maxErr, Tt_n1[NX/2][NY/2][0],
    Tt_n1[0][NY/2][0], Tt_n1[NX/4][NY/2][NZ3], P0);
}
else
{
    if (dp)
    sprintf(tmp, "p%d t:%2d I:%d Err:%5.4f T0:%7.4lf T1:%7.4lf
T2:%7.4lf P:%7.4lf", loopP, t, I, maxErr, Tt_n1[NX/2][NY/2][0],
    Tt_n1[0][NY/2][0], Tt_n1[NX/4][NY/2][NZ3], 0.0);
    else
    sprintf(tmp, "p%d t:%2d I:%d Err:%5.4f T0:%7.4lf T1:%7.4lf
T2:%7.4lf P:%7.4lf", loopP, t, I, maxErr, Tt_n1[NX/2][NY/2][0],
    Tt_n1[0][NY/2][0], Tt_n1[NX/4][NY/2][NZ3], 0.0);
}
writeLog(tmp);

#ifdef SCREEN_OUT
printf(tmp); printf("\n");
#endif

if (!dp) {
    if ((t % 1000) == 0) {
        writeSquareXZ(0, 0, NX3, NY/2, t, I);
        writeSquareYZ(0, 0, NY, NZ3, NX/2, t, I);
    }
    else if ((Tn_1[NX/2][NY/2][0]) >= 8) {
        writeSquareXZ(0, 0, NX, NZ3, NY/4, t, I);
        writeSquareXZ(0, 0, NX, NZ3, 3*NY/4, t, I);
        writeSquareYZ(0, 0, NY, NZ3, Nx/2, t, I);
        writeSquareXZ(0, 0, NX, NZ3, Ny/2, t, I);
        writeZCenter(t);
    }
}

memcpy(Tt, Tn_1, sizeof(double)*(NX+1)*(NY+1)*(NZ+1));
AdjustPower(P0);

int getTv_blood()
{
    //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
    int i;

    //first level
    for(i=0;i<LZ1;i++)
        Tv1[i] = (Tn1[X5][Y5][Z3+i] + Tn1[X6][Y5][Z3+i] + Tn1[X5][Y6][Z3+i] + Tn1[X6][Y6][Z3+i]) / 4.0;

    //second level
    for(i=0;i<=LX2;i++) //i=0 & LX2 are on the blood vessels
        Tv2[i] = (Tn1[i+X1][Y3][Z2] + Tn1[i+X1][Y4][Z2] + Tn1[i+X1][Y3][Z3] + Tn1[i+X1][Y4][Z3]) / 4.0;

    //Record the temperature of the last second
    //writeSquareXZ(0, 0, NX, NZ3, NY/4, t, I);
    //writeSquareXZ(0, 0, NX, NZ3, 3*NY/4, t, I);
    writeSquareYZ(0, 0, NY, NZ3, NX/2, t, I);
    writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
    return(1);
}
//third level
for(i=0;i<=LZ3;i++)
{
    TV3_n1[i] = ( Tn1[X1][Y1][Z2-i] + Tn1[X2][Y1][Z2-i] + Tn1[X1][Y2][Z2-i] +
                 Tn1[X2][Y2][Z2-i] ) / 4.0;
    TV3_n1[i+LZ3+1] = ( Tn1[X3][Y1][Z2-i] + Tn1[X4][Y1][Z2-i] +
                        Tn1[X3][Y2][Z2-i] + Tn1[X4][Y2][Z2-i] ) / 4.0;
}
return(1);

int CalcTb()
{
    int i;
    double fk1, fk2, fk3, fk4;
    Tbl_n1[0] = THETAO;

    //first level blood
    for(i=1;i<=LZ1;i++)
    {
        fk1 = factor1*(Tv1_n1[i-1]- Tbl_n1[i-1]);
        fk2 = factor1*(Tv1_n1[i-1]-(Tb1_n1[i-1]+fk1/2));
        fk3 = factor1*(Tv1_n1[i-1]-(Tb1_n1[i-1]+fk2/2));
        fk4 = factor1*(Tv1_n1[i-1]-(Tb1_n1[i-1]+fk3));
        Tbl_n1[i] = Tbl_n1[i-1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
    }

    //second level blood
    int centerX2 = NX/2;
    Tb2_n1[centerX2 - XI] = Tbl_n1[LZ1]; //the interface grid point between level 1 and level 2

    //left part
    for(i=centerX2-1;i>X1;i--)
    {
        fk1 = factor2*(Tv2_n1[i-X1+1]- Tb2_n1[i-X1+1]);
        fk2 = factor2*(Tv2_n1[i-X1+1]-(Tb2_n1[i-X1+1]+fk1/2));
        fk3 = factor2*(Tv2_n1[i-X1+1]-(Tb2_n1[i-X1+1]+fk2/2));
        fk4 = factor2*(Tv2_n1[i-X1+1]-(Tb2_n1[i-X1+1]+fk3));
        Tb2_n1[i-X1] = Tb2_n1[i-X1+1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
    }
    Tb2_n1[0] = Tn1[I][Y3+LY2/2][Z2+LZ2/2];

    //right part
    for(i=centerX2+1;i<X4;i++)
    {
        fk1 = factor2*(Tv2_n1[i-X1-1]- Tb2_n1[i-X1-1]);
        fk2 = factor2*(Tv2_n1[i-X1-1]-(Tb2_n1[i-X1-1]+fk1/2));
        fk3 = factor2*(Tv2_n1[i-X1-1]-(Tb2_n1[i-X1-1]+fk2/2));
        fk4 = factor2*(Tv2_n1[i-X1-1]-(Tb2_n1[i-X1-1]+fk3));
        Tb2_n1[i-X1] = Tb2_n1[i-X1-1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
    }
}
\[ T_{b2\_n1[LX2]} = T_{t\_n1[I[X4][Y3+LY2/2][Z2+1Z2/2]]; \]

// third level blood
\[ T_{b3\_n1[0]} = T_{b2\_n1[LX3/2]]; \] // the interface grid point between level 2 and level 3

for(i=1;i<=LZ3;i++)
{
    fk1 = factor3*(Tv3\_n[i-1]- Tb3\_n[i-1]);
    fk2 = factor3*(Tv3\_n[i-1]-Tb3\_n[i-1]+fk1/2);
    fk3 = factor3*(Tv3\_n[i-1]-Tb3\_n[i-1]+fk2/2);
    fk4 = factor3*(Tv3\_n[i-1]-Tb3\_n[i-1]+fk3);
    Tb3\_n[i] = Tb3\_n[i-1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
}

// right part
\[ T_{b3\_n1[LZ3+1]} = T_{b2\_n1[LX2-LX3/2]]; \] // the interface grid point between level 2 and level 3

for(i=1;i<=LZ3;i++)
{
    fk1 = factor3*(Tv3\_n[i+1+LZ3+1] - Tb3\_n[i+1+LZ3+1]);
    fk2 = factor3*(Tv3\_n[i+1+LZ3+1] - (Tb3\_n[i+1+LZ3+1] + fk1/2));
    fk3 = factor3*(Tv3\_n[i+1+LZ3+1] - (Tb3\_n[i+1+LZ3+1] + fk2/2));
    fk4 = factor3*(Tv3\_n[i+1+LZ3+1] - (Tb3\_n[i+1+LZ3+1] + fk3));
    Tb3\_n[i+1+LZ3+1] = Tb3\_n[i+1+LZ3+1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
}

return(1);

// calculate tissue temperature at time level n+1
// using preconditioned Richardson iteration
double CalcTt()
{
    double maxErr, f;

    // initialize tridiagonal system
    memcpy(a, a0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memcpy(b, b0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memcpy(c, c0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(d, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+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++)
            {
                f = ( (2*p1*Cl+Wb1*Cb1*deltaT)*Tt\_n1[I[i][j][z]
                      + (-2*p1*Cl+Wb1*Cb1*deltaT)*Tt[i][j][z]
                      + 2*Wb1*Cb1*deltaT*(Tb3\_n1[1LZ3]))
                      -k1*deltaT*( (Tn\_n1[I[i-1][j][z]+Tn\_n1[I[i+1][j][z]]
                      -2*Tt\_n1[I[i][j][z]]/(deltaX*deltaX)
                      +(Tt\_n1[I[i+1][j][z]+Tt\_n1[I[i][j+1][z]]

    return(f);}
-2*Tt_{n+1}[i][j][z]/(deltaY*deltaY) \\
+ (Tt_{n+1}[i][j][z+1]+Tt_{n+1}[i][j][z-1])/(deltaZ*deltaZ) \\
-2*Tt_{n+1}[i][j][z]/(deltaZ*deltaZ) \\
-k1*deltaT*( (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX) \\
+ (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaY*deltaY) \\
+(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaZ*deltaZ)) \\
-2*deltaT*Q1[i][j][z];

\begin{align*}
\text{d}[i][j][z] &= \left(2*p1*C1 + Wb1*Cb1*deltaT \right) \\
&+ (4*k1*deltaT)*(1.0/(deltaX*deltaX)+1.0/(deltaY*deltaY)) \\
&+ (2*k1*deltaT)/(deltaZ*deltaZ) * Tt_{n+1}[i][j][z] \\
&- k1*deltaT/(deltaZ*deltaZ) * (Tt_{n+1}[i][j][z+1]+Tt_{n+1}[i][j][z-1]) \\
&- \text{omega} * f;
\end{align*}

\text{d}[i][j][NZ1]=0;

\text{for}(z=\text{NZ1}+1; z<=\text{NZ2}-1; z++) {
\text{f} = \left(2*p2*C2 + Wb2*Cb2*deltaT \right) * Tt_{n+1}[i][j][z] \\
+ (2*p2*C2+2*Wb2*Cb2*deltaT)*Tt[i][j][z] \\
- 2*Wb2*Cb2*deltaT*(Tb3_{n+1}[LZ3]) \\
-k2*deltaT*( (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX) \\
+ (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaY*deltaY) \\
+(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaZ*deltaZ)) \\
-2*deltaT*Q2[i][j][z];

\begin{align*}
\text{d}[i][j][z] &= \left(2*p2*C2 + Wb2*Cb2*deltaT \right) \\
&+ (4*k2*deltaT)*(1.0/(deltaX*deltaX)+1.0/(deltaY*deltaY)) \\
&+ (2*k2*deltaT)/(deltaZ*deltaZ) * Tt_{n+1}[i][j][z] \\
&- k2*deltaT/(deltaZ*deltaZ) * (Tt_{n+1}[i][j][z+1]+Tt_{n+1}[i][j][z-1]) \\
&- \text{omega} * f;
\end{align*}

\text{d}[i][j][NZ2]=0;

// third skin layer 
\text{for}(z=\text{NZ2}+1; z<=\text{NZ3}-1; z++) {
\text{f} = \left(2*p3*C3 + Wb3*Cb3*deltaT \right) * Tt_{n+1}[i][j][z] \\
+ (2*p3*C3+2*Wb3*Cb3*deltaT)*Tt[i][j][z] \\
- 2*Wb3*Cb3*deltaT*(Tb3_{n+1}[LZ3]) \\
-k3*deltaT*( (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX) \\
+ (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaY*deltaY) \\
+(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaZ*deltaZ)) \\
-2*deltaT*Q3[i][j][z];

\begin{align*}
\text{d}[i][j][z] &= \left(2*p3*C3 + Wb3*Cb3*deltaT \right) \\
&+ (4*k3*deltaT)*(1.0/(deltaX*deltaX)+1.0/(deltaY*deltaY)) \\
&+ (2*k3*deltaT)/(deltaZ*deltaZ) * Tt_{n+1}[i][j][z] \\
&- k3*deltaT/(deltaZ*deltaZ) * (Tt_{n+1}[i][j][z+1]+Tt_{n+1}[i][j][z-1]) \\
&- \text{omega} * f;
\end{align*}

\text{d}[i][j][NZ3]=0;

// fourth skin layer 
\text{for}(z=\text{NZ3}+1; z<=\text{NZ4}-1; z++) {
\text{f} = \left(2*p4*C4 + Wb4*Cb4*deltaT \right) * Tt_{n+1}[i][j][z] \\
+ (2*p4*C4+2*Wb4*Cb4*deltaT)*Tt[i][j][z] \\
- 2*Wb4*Cb4*deltaT*(Tb3_{n+1}[LZ3]) \\
-k4*deltaT*( (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX) \\
+ (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaY*deltaY) \\
+(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaZ*deltaZ)) \\
-2*deltaT*Q4[i][j][z];

\begin{align*}
\text{d}[i][j][z] &= \left(2*p4*C4 + Wb4*Cb4*deltaT \right) \\
&+ (4*k4*deltaT)*(1.0/(deltaX*deltaX)+1.0/(deltaY*deltaY)) \\
&+ (2*k4*deltaT)/(deltaZ*deltaZ) * Tt_{n+1}[i][j][z] \\
&- k4*deltaT/(deltaZ*deltaZ) * (Tt_{n+1}[i][j][z+1]+Tt_{n+1}[i][j][z-1]) \\
&- \text{omega} * f;
\end{align*}

\text{d}[i][j][NZ4]=0;
\[ + (T_t[i][j-1][z]+T_t[i][j+1][z]-2*T_t[i][j][z])/(\delta Y \cdot \delta Y) \]
\[ + (T_t[i][j][z-1]+T_t[i][j][z+1]-2*T_t[i][j][z])/(\delta Z \cdot \delta Z) \]
\[-2*\Delta T \cdot Q_3[i][j][z] \];

\[ d[i][j][z] = (2*p3*C_3 + W_{b3} \cdot C_{b3} \cdot \Delta T \]
\[ + (4*k3*\Delta T)*((1.0/(\delta X \cdot \delta X)+1.0/(\delta Y \cdot \delta Y)) \]
\[ + (2*k3*\Delta T)/(\delta Z \cdot \delta Z)) \cdot T_{n1_I}[i][j][z] \]
\[ - k3*\Delta T/(\delta Z \cdot \delta Z) \cdot (T_{n1_I}[i][j][z-1]+T_{n1_I}[i][j][z+1]) \]
\[-\omega \cdot f; \]

//reassign back the grid points in blood
//level 3
for(z=Z1;z<=Z2;z++)
for(j = Y1+1;j<Y2;j++)
{
    //left branch
    for(i=X1+1;i<X2;i++)
        T_{n1_I}[i][j][z] = T_b3_n1[Z2-z];

    //right branch
    for(i=X3+1;i<X4;i++)
        T_{n1_I}[i][j][z] = T_b3_n1[Z2-z+LZ3+1];
}
//level 2
for(z=Z2+1;z<Z3;z++)
for(j = Y3+1;j<Y4;j++)
for(i=X1+1;i<X4;i++)
{
    T_{n1_I}[i][j][z] = T_b2_n1[i-X1];
}
//level 1
for(j=Y3+1;j<Y4;j++)
for(i=X5+1;i=X6;i++)
    T_{n1_I}[i][j][Z3] = T_b1_n1[LZ1]; //interface between level 2 & 3
for(z=Z3+1;z<=NZ3;z++)
for(j=Y5+1;j<Y6;j++)
for(i=X5+1;i<X6;i++)
    T_{n1_I}[i][j][z] = T_b1_n1[NZ3-z];
//adjust the matrix for blood vessel
//blood level 1
for(i=X5;i<=X6;i++)
for(j=Y5;j<=Y6;j++)
{
    //side walls
    d[i][j][Z3-1] = c[i][j][Z3-1]*T_{n1_I}[i][j][Z3];
    c[i][j][Z3-1] = 0;
}
for(z=Z3; z<NZ3; z++)
{
    b[i][j][z] = 0;
    a[i][j][z] = 1;
    c[i][j][z] = 0;
    d[i][j][z] = Tn1[i][j][z];
}
}
else
{
    //center (the interface part with level 2 will be fixed below on level 2)
    b[i][j][Z3] = -1;
    a[i][j][Z3] = 1 + Bi*deltaZ;
    c[i][j][Z3] = 0;
    d[i][j][Z3] = Bi*deltaZ*Tn1[i][j][Z3+1];
    for(z=Z3+1; z<NZ3; z++)
    {
        b[i][j][z] = 0;
        a[i][j][z] = 1;
        c[i][j][z] = 0;
        d[i][j][z] = Tn1[i][j][z];
    }
}

//blood level 2
for(i=I1; i<=I4; i++)
for(j=J3; j<=J4; j++)
{
    if(i==I1 || i==I4 || j==J3 || j==J4)
    {
        //side walls, use the previous step temperature (Loop I)
        //top part on the wall
        d[i][j][Z2-1] = c[i][j][Z2-1]*Tn1[i][j][Z2];
        c[i][j][Z2-1] = 0;
        for(z=Z2; z<=Z3; z++)
        {
            b[i][j][z] = 0;
            a[i][j][z] = 1;
            c[i][j][z] = 0;
            d[i][j][z] = Tn1[i][j][z];
        }
        //bottom part on the wall
        d[i][j][Z3+1] = b[i][j][Z3+1]*Tn1[i][j][Z3];
        b[i][j][Z3+1] = 0;
    }
else
    {
        //center, use temperature at loop I+1, (already updated by blood temperatures)
        //top wall (the interface part with level 3 will be fixed below on level 3)
        b[i][j][Z2] = -1;
        a[i][j][Z2] = 1 + Bi*deltaZ;
        c[i][j][Z2] = 0;
        d[i][j][Z2] = Bi*deltaZ*Tn1[i][j][Z2+1];
        //between
for(z = Z2 + 1; z < Z3; z++)
{
    b[i][j][z] = 0;
    a[i][j][z] = 1;
    c[i][j][z] = 0;
    d[i][j][z] = T_{t_n1[i][j][z]};
}
// bottom wall
b[i][j][Z3] = 0;
a[i][j][Z3] = 1 + Bi*deltaZ;
c[i][j][Z3] = -1;
d[i][j][Z3] = Bi*deltaZ*T_{t_n1[i][j][Z3-1]};

if(i >= X5 && i <= X6 && j >= Y3 && j <= Y4)
{
    // the joint between level 1 and level 2
    // bottom part fix
    b[i][j][Z3] = 0;
a[i][j][Z3] = 1;
c[i][j][Z3] = 0;
    if(i == X5 || i == X6 || j == Y3 || j == Y4)
    d[i][j][Z3] = T_{t_n1[i][j][Z3]};
    else
    d[i][j][Z3] = T_{t_n1[i][j][Z3]};
}

// blood level 3 left branch
for(i = X1; i <= X2; i++)
    for(j = Y1; j <= Y2; j++)
    {
        if(i == X1 || i == X2 || j == Y1 || j == Y2)
        {
            // side walls, use the previous step temperature (Loop I)
            // top part on the wall
            d[i][j][Z1-1] := c[i][j][Z1-1]*T_{t_n1[i][j][Z1]};
            c[i][j][Z1-1] = 0;
            for(z = Z1; z <= Z2; z++)
            {
                b[i][j][z] = 0;
                a[i][j][z] = 1;
                c[i][j][z] = 0;
                d[i][j][z] = T_{t_n1[i][j][z]};
            }
        }
        else
        {
            // center, use temperature at loop I+1
            for(z = Z1; z <= Z2; z++)
            {
                b[i][j][z] = 0;
                a[i][j][z] = 1;
                c[i][j][z] = 0;
                d[i][j][z] = T_{t_n1[i][j][z]};
            }
        }
    }
}
//blood level 3 right branch
for(i=X3;i<=X4;i++)
    for(j=Y1;j<=Y2;j++)
    {
        if(i==X3 || i==X4 || j==Y1 || j==Y2)
            //side walls, use the previous step temperature (Loop I)

                //top part on the wall
                d[i][j][Z1-1] := c[i][j][Z1-1]*T_t_n1[I][i][j][Z1];
                c[i][j][Z1-1] = 0;
                for(z=Z1;z<=Z2;z++)
                    
                        b[i][j][z] = 0;
                        a[i][j][z] = 1;
                        c[i][j][z] = 0;
                        d[i][j][z] = T_t_n1[I][i][j][z];

                }
            }
        else
            //center, use temperature at loop I+1,

                for(z=Z1;z<=Z2;z++)
                    
                        b[i][j][z] = 0;
                        a[i][j][z] = 1;
                        c[i][j][z] = 0;
                        d[i][j][z] = T_t_n1[I][i][j][z];
                }
        }

    //writeLinearSys(NX/2, NY/2, t, I);

double tE[NZ3], tF[NZ3];
//solve the tri-diagonal system
for(i=1;i<=NX-1;i++)
    for(j=1;j<=NY-1;j++)
    {
        #ifdef CONVECTION
            // Without heat convection on the surface
            tF[1] = d[i][j][1] / ( b[i][j][1] + a[i][j][1] );
            tE[1] = -c[i][j][1] / ( b[i][j][1] + a[i][j][1] );
        #else
            // With heat convection on the surface
            tF[1] = (d[i][j][1]*Hf*deltaZ*Tf/(k1+deltaZ*Hf)) / ( b[i][j][1]*k1/(k1+deltaZ*Hf) + a[i][j][1] );
            tE[1] = -c[i][j][1] / ( b[i][j][1]*k1/(k1+deltaZ*Hf) + a[i][j][1] );
        #endif

        for(z=2;z<=NZ3-1;z++)
        {
            tF[z] = (d[i][j][z] - b[i][j][z]*tF[z-1]) / ( a[i][j][z] + b[i][j][z]*tE[z-1] );
            tE[z] = -c[i][j][z] / ( a[i][j][z] + b[i][j][z]*tE[z-1] );
        }
    }

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ T_{t \_n1}[i][j][NZ3-1] = tF[NZ3-1] / (1-tE[NZ3-1]); \]

for\( z = NZ3-2; z >= 1; z -- \)
\[ T_{t \_n1}[i][j][z] = tF[z] + tE[z] \times T_{t \_n1}[i][j][z+1]; \]

//assign tissue boundary grid points
for\( i = 0; i <= NX; i ++ \)
for\( j = 0; j <= NY; j ++ \)
{
  \#ifdef CONVECTION
  // Without heat convection on the surface
  \[ T_{t \_n1}[i][j][0] = T_{t \_n1}[i][j][1]; \]
  \#else
  // With heat convection on the surface
  \[ T_{t \_n1}[i][j][0] = T_{t \_n1}[i][j][1]\times k1/(k1+\text{deltaZ}\times Hf) + Hf\times\text{deltaZ}/(k1+\text{deltaZ}\times Hf) \times T_{f}; \]
  \#endif

  \[ T_{t \_n1}[i][j][NZ3] = T_{t \_n1}[i][j][NZ3-1]; \]
}

for\( z = 0; z <= NZ3; z ++ \)
for\( j = 0; j <= NY; j ++ \)
{
  \[ T_{t \_n1}[0][j][z] = T_{t \_n1}[1][j][z]; \]
  \[ T_{t \_n1}[NX][j][z] = T_{t \_n1}[NX-1][j][z]; \]
}

for\( z = 0; z <= NZ3; z ++ \)
for\( i = 0; i <= NX; i ++ \)
{
  \[ T_{t \_n1}[i][0][z] = T_{t \_n1}[i][1][z]; \]
  \[ T_{t \_n1}[i][NY][z] = T_{t \_n1}[i][NY-1][z]; \]
}

//calculate blood vessel temperature on sides
//level 3
for\( z = Z1; z <= Z2; z ++ \)
{
  //left branch
  for\( j = Y1; j <= Y2; j ++ \)
  {
    \[ T_{t \_n1}[X1][j][z] = ( T_{t \_n1}[X1-1][j][z] + Tb3_\_n1[Z2-z] \times \text{deltaX} \times Bi ) / (1+\text{deltaX}\times Bi); \] //deltaX
    \[ T_{t \_n1}[X2][j][z] = ( T_{t \_n1}[X2+1][j][z] + Tb3_\_n1[Z2-z] \times \text{deltaX} \times Bi ) / (1+\text{deltaX}\times Bi); \]
  }

  for\( i = X1; i <= X2; i ++ \)
  {
    \[ T_{t \_n1}[i][Y1][z] = ( T_{t \_n1}[i][Y1-1][z] + Tb3_\_n1[Z2-z] \times \text{deltaY} \times Bi ) / (1+\text{deltaY}\times Bi); \] //deltaY
    \[ T_{t \_n1}[i][Y2][z] = ( T_{t \_n1}[i][Y2+1][z] + Tb3_\_n1[Z2-z] \times \text{deltaY} \times Bi ) / (1+\text{deltaY}\times Bi); \]
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
// right branch

for(j = Y1; j <= Y2; j++)
{
    Tt_n[1][X3][j][z] = ( Tt_n[1][X3-1][j][z] + Tb_3_n[1][Z2-z+LZ3+1] * deltaX*Bi ) / (1+deltaX*Bi);
    Tt_n[1][X4][j][z] = ( Tt_n[1][X4+1][j][z] + Tb_3_n[1][Z2-z+LZ3+1] * deltaX*Bi ) / (1+deltaX*Bi);
}

for(i = X3; i <= X4; i++)
{
    Tt_n[1][i][Y1][z] = ( Tt_n[1][i][Y1-1][z] + Tb_3_n[1][Z2-z+LZ3+1] * deltaY*Bi ) / (1+deltaY*Bi);
    Tt_n[1][i][Y2][z] = ( Tt_n[1][i][Y2+1][z] + Tb_3_n[1][Z2-z+LZ3+1] * deltaY*Bi ) / (1+deltaY*Bi);
}

// level 2

for(z = Z2; z <= Z3; z++)
{
    for(i = X1; i <= X4; i++)
    {
        Tt_n[1][i][Y3][z] = ( Tt_n[1][i][Y3-1][z] + Tb_2_n[1][i-X1] * deltaY*Bi ) / (1+deltaY*Bi);
        Tt_n[1][i][Y4][z] = ( Tt_n[1][i][Y4+1][z] + Tb_2_n[1][i-X1] * deltaY*Bi ) / (1+deltaY*Bi);
    }

    // left most and right most sides
    for(j = Y3; j <= Y4; j++)
    {
        Tt_n[1][X1][j][z] = ( Tt_n[1][X1-1][j][z] + Tb_2_n[1][i-X1] * deltaX*Bi ) / (1+deltaX*Bi);
        Tt_n[1][X4][j][z] = ( Tt_n[1][X4+1][j][z] + Tb_2_n[1][i-X1] * deltaX*Bi ) / (1+deltaX*Bi);
    }
}

// level 1

for(z = Z3; z <= NZ3; z++)
{
    for(j = Y5; j <= Y6; j++)
    {
        Tt_n[1][X5][j][z] = ( Tt_n[1][X5-1][j][z] + Tb_1_n[1][NZ3-z] * deltaX*Bi ) / (1+deltaX*Bi);
        Tt_n[1][X6][j][z] = ( Tt_n[1][X6+1][j][z] + Tb_1_n[1][NZ3-z] * deltaX*Bi ) / (1+deltaX*Bi);
    }
}

for(i = X5; i <= X6; i++)
{
    Tt_n[1][i][Y5][z] = ( Tt_n[1][i][Y5-1][z] + Tb_1_n[1][NZ3-z] * deltaY*Bi ) / (1+deltaY*Bi);
    Tt_n[1][i][Y6][z] = ( Tt_n[1][i][Y6+1][z] + Tb_1_n[1][NZ3-z] * deltaY*Bi ) / (1+deltaY*Bi);
}
//==handle the edges of blood vessels==
//use the average of the two neighbor points on the vessel
//==level 3

//joints between blood level 2 & 3
for(j=Y1; j<=Y2; j++)
{
    Tt_n1[X2][j][Z2] = ( Tt_n1[X2+1][j][Z2] + Tt_n1[X2][j][Z2-1] )/2;
    Tt_n1[X3][j][Z2] = ( Tt_n1[X3-1][j][Z2] + Tt_n1[X3][j][Z2-1] )/2;
}
for(i=X1; i<=X2; i++)
{
    Tt_n1[i][Y1][Z2] = ( Tt_n1[i][Y1-1][Z2] + Tt_n1[i][Y1][Z2-1] )/2;
    Tt_n1[i][Y2][Z2] = ( Tt_n1[i][Y2+1][Z2] + Tt_n1[i][Y2][Z2-1] )/2;
}
for(i=X3; i<=X4; i++)
{
    Tt_n1[i][Y1][Z2] = ( Tt_n1[i][Y1-1][Z2] + Tt_n1[i][Y1][Z2-1] )/2;
    Tt_n1[i][Y2][Z2] = ( Tt_n1[i][Y2+1][Z2] + Tt_n1[i][Y2][Z2-1] )/2;
}

//side edges
for(z=Z1; z<=Z2; z++)
{
    Tt_n1[X1][Y1][z] = ( Tt_n1[X1+1][Y1][z] + Tt_n1[X1][Y1+1][z] )/2;
    Tt_n1[X2][Y1][z] = ( Tt_n1[X2-1][Y1][z] + Tt_n1[X2][Y1+1][z] )/2;
    Tt_n1[X3][Y1][z] = ( Tt_n1[X3+1][Y1][z] + Tt_n1[X3][Y1+1][z] )/2;
    Tt_n1[X4][Y1][z] = ( Tt_n1[X4-1][Y1][z] + Tt_n1[X4][Y1+1][z] )/2;
    Tt_n1[X1][Y2][z] = ( Tt_n1[X1+1][Y2][z] + Tt_n1[X1][Y2+1][z] )/2;
    Tt_n1[X2][Y2][z] = ( Tt_n1[X2-1][Y2][z] + Tt_n1[X2][Y2+1][z] )/2;
    Tt_n1[X3][Y2][z] = ( Tt_n1[X3+1][Y2][z] + Tt_n1[X3][Y2+1][z] )/2;
    Tt_n1[X4][Y2][z] = ( Tt_n1[X4-1][Y2][z] + Tt_n1[X4][Y2+1][z] )/2;
}

//==level 2
//top surface
for(i=X1; i<=X4; i++)
{
    Tt_n1[i][Y3][Z2] = ( Tt_n1[i][Y3][Z2+1] + Tt_n1[i][Y3+1][Z2] )/2;
    Tt_n1[i][Y4][Z2] = ( Tt_n1[i][Y4][Z2+1] + Tt_n1[i][Y4+1][Z2] )/2;
}
for(j=Y3; j<=Y1; j++)
{
    Tt_n1[X1][j][Z2] = ( Tt_n1[X1][j][Z2+1] + Tt_n1[X1+1][j][Z2] )/2;
    Tt_n1[X4][j][Z2] = ( Tt_n1[X4][j][Z2+1] + Tt_n1[X4+1][j][Z2] )/2;
}
for(j=Y2+1; j<=Y4; j++)
{
    Tt_n1[X1][j][Z2] = ( Tt_n1[X1][j][Z2+1] + Tt_n1[X1+1][j][Z2] )/2;
    Tt_n1[X4][j][Z2] = ( Tt_n1[X4][j][Z2+1] + Tt_n1[X4+1][j][Z2] )/2;
}

//side vertical edges
for(z=Z2; z<=Z3; z++)
{
    Tt_n1[X1][Y3][z] = ( Tt_n1[X1+1][Y3][z] + Tt_n1[X1][Y3+1][z] )/2;
}
\[ T_{n1}^{(X4)[Y3][z]} = \frac{(T_{n1}^{(X4-1)[Y3][z]} + T_{n1}^{(X4+1)[Y3][z]})}{2}; \]

\[ T_{n1}^{(X4)[Y4][z]} = \frac{(T_{n1}^{(X4-1)[Y4][z]} + T_{n1}^{(X4+1)[Y4][z]})}{2}; \]

//bottom surface
for\( j = Y3; j \leq Y4; j++ \) \{
  \[ T_{n1}^{(X1)[j][Z3]} = \frac{(T_{n1}^{(X1+1)[j][Z3]} + T_{n1}^{(X1-1)[j][Z3-1]})}{2}; \]
  \[ T_{n1}^{(X4)[j][Z3]} = \frac{(T_{n1}^{(X4-1)[j][Z3]} + T_{n1}^{(X4+1)[j][Z3-1]})}{2}; \]
\}
for\( i = X1; i < X5; i++ \) \{
  \[ T_{n1}^{(i)[Y3][Z3]} = \frac{(T_{n1}^{(i+1)[Y3][Z3]} + T_{n1}^{(i-1)[Y3][Z3-1]})}{2}; \]
  \[ T_{n1}^{(i)[Y4][Z3]} = \frac{(T_{n1}^{(i+1)[Y4][Z3]} + T_{n1}^{(i-1)[Y4][Z3-1]})}{2}; \]
\}
for\( i = X5; i < X6; i++ \) \{ //joints of blood level 1 & 2
  \[ T_{n1}^{(i)[Y3][Z3]} = \frac{(T_{n1}^{(i-1)[Y3][Z3]} + T_{n1}^{(i+1)[Y3][Z3-1]})}{2}; \]
  \[ T_{n1}^{(i)[Y4][Z3]} = \frac{(T_{n1}^{(i+1)[Y4][Z3]} + T_{n1}^{(i-1)[Y4][Z3-1]})}{2}; \]
\}
for\( i = X6; i < X4; i++ \) \{ //vertex
  \[ T_{n1}^{(X1)[Y3][Z2]} = \frac{(T_{n1}^{(X1+1)[Y3][Z2]} + T_{n1}^{(X1-1)[Y3][Z2+1]} + T_{n1}^{(X1)[Y3][Z2+1]})}{3}; \]
  \[ T_{n1}^{(X1)[Y3][Z3]} = \frac{(T_{n1}^{(X1+1)[Y3][Z3]} + T_{n1}^{(X1-1)[Y3][Z3+1]} + T_{n1}^{(X1)[Y3][Z3+1]})}{3}; \]
  \[ T_{n1}^{(X1)[Y4][Z2]} = \frac{(T_{n1}^{(X1+1)[Y4][Z2]} + T_{n1}^{(X1-1)[Y4][Z2+1]} + T_{n1}^{(X1)[Y4][Z2+1]})}{3}; \]
  \[ T_{n1}^{(X1)[Y4][Z3]} = \frac{(T_{n1}^{(X1+1)[Y4][Z3]} + T_{n1}^{(X1-1)[Y4][Z3+1]} + T_{n1}^{(X1)[Y4][Z3+1]})}{3}; \]
  \[ T_{n1}^{(X4)[Y3][Z2]} = \frac{(T_{n1}^{(X4-1)[Y3][Z2]} + T_{n1}^{(X4+1)[Y3][Z2+1]} + T_{n1}^{(X4)[Y3][Z2+1]})}{3}; \]
  \[ T_{n1}^{(X4)[Y3][Z3]} = \frac{(T_{n1}^{(X4-1)[Y3][Z3]} + T_{n1}^{(X4+1)[Y3][Z3+1]} + T_{n1}^{(X4)[Y3][Z3+1]})}{3}; \]
  \[ T_{n1}^{(X4)[Y4][Z2]} = \frac{(T_{n1}^{(X4-1)[Y4][Z2]} + T_{n1}^{(X4+1)[Y4][Z2+1]} + T_{n1}^{(X4)[Y4][Z2+1]})}{3}; \]
  \[ T_{n1}^{(X4)[Y4][Z3]} = \frac{(T_{n1}^{(X4-1)[Y4][Z3]} + T_{n1}^{(X4+1)[Y4][Z3+1]} + T_{n1}^{(X4)[Y4][Z3+1]})}{3}; \]
\}//level 1
//top surface
for\( i = X5; i < X6; i++ \) \{
  \[ T_{n1}^{(i)[Y5][Z3]} = \frac{(T_{n1}^{(i+1)[Y5][Z3]} + T_{n1}^{(i-1)[Y5][Z3+1]} + T_{n1}^{(i)[Y5][Z3+1]})}{2}; \]
  \[ T_{n1}^{(i)[Y6][Z3]} = \frac{(T_{n1}^{(i+1)[Y6][Z3+1]} + T_{n1}^{(i-1)[Y6][Z3+1]} + T_{n1}^{(i)[Y6][Z3+1]})}{2}; \]
\}
for\( j = Y5; j < Y3; j++ \) \{
  \[ T_{n1}^{(X5)[j][Z3]} = \frac{(T_{n1}^{(X5+1)[j][Z3]} + T_{n1}^{(X5-1)[j][Z3+1]} + T_{n1}^{(X5)[j][Z3+1]})}{2}; \]
  \[ T_{n1}^{(X6)[j][Z3]} = \frac{(T_{n1}^{(X6+1)[j][Z3]} + T_{n1}^{(X6-1)[j][Z3+1]} + T_{n1}^{(X6)[j][Z3+1]})}{2}; \]
\}
for\( j = Y3; j < Y4; j++ \) \{ //corner between level 1 & 2
  \[ T_{n1}^{(X5)[j][Z3]} = \frac{(T_{n1}^{(X5-1)[j][Z3]} + T_{n1}^{(X5)[j][Z3+1]})}{2}; \]
  \[ T_{n1}^{(X6)[j][Z3]} = \frac{(T_{n1}^{(X6+1)[j][Z3]} + T_{n1}^{(X6-1)[j][Z3+1]})}{2}; \]
\}
for(z=Z3; z<NZ3+1; z++)
{
    T t_nl[X5][Y5][z] = (T t_nl[X5+1][Y5][z] + T t_nl[X5][Y5+1][z])/2;
    T t_nl[X6][Y5][z] = (T t_nl[X6-1][Y5][z] + T t_nl[X6][Y5+1][z])/2;
    T t_nl[X5][Y6][z] = (T t_nl[X5+1][Y6][z] + T t_nl[X5][Y6-1][z])/2;
    T t_nl[X6][Y6][z] = (T t_nl[X6-1][Y6][z] + T t_nl[X6][Y6-1][z])/2;
}
// vertix
T t_nl[X5][Y5][Z3] = (T t_nl[X5+1][Y5][Z3] + T t_nl[X5][Y5+1][Z3] + T t_nl[X5][Y5-1][Z3])/3;
T t_nl[X5][Y6][Z3] = (T t_nl[X5+1][Y6][Z3] + T t_nl[X5][Y6-1][Z3] + T t_nl[X5][Y6-1][Z3])/3;
T t_nl[X6][Y5][Z3] = (T t_nl[X6-1][Y5][Z3] + T t_nl[X6][Y5+1][Z3] + T t_nl[X6][Y5-1][Z3])/3;
T t_nl[X6][Y6][Z3] = (T t_nl[X6-1][Y6][Z3] + T t_nl[X6][Y6-1][Z3] + T t_nl[X6][Y6+1][Z3])/3;

// joints vertix interpolation
// level 1 & 2
T t_nl[X5][Y3][Z3] = (T t_nl[X5+1][Y3][Z3] + T t_nl[X5][Y3+1][Z3] + T t_nl[X5][Y3-1][Z3]
+ T t_nl[X5][Y3+1][Z3])/4;
T t_nl[X5][Y4][Z3] = (T t_nl[X5+1][Y4][Z3] + T t_nl[X5][Y4-1][Z3] + T t_nl[X5][Y4+1][Z3]
+ T t_nl[X5][Y4+1][Z3])/4;
T t_nl[X6][Y3][Z3] = (T t_nl[X6+1][Y3][Z3] + T t_nl[X6][Y3-1][Z3] + T t_nl[X6][Y3+1][Z3]
+ T t_nl[X6][Y3+1][Z3])/4;
T t_nl[X6][Y4][Z3] = (T t_nl[X6+1][Y4][Z3] + T t_nl[X6][Y4-1][Z3] + T t_nl[X6][Y4+1][Z3]
+ T t_nl[X6][Y4+1][Z3])/4;
// level 2 & 3
T t_nl[X1][Y1][Z2] = (T t_nl[X1+1][Y1][Z2] + T t_nl[X1][Y1-1][Z2] + T t_nl[X1][Y1+1][Z2]
+ T t_nl[X1][Y1][Z2-1])/4;
T t_nl[X1][Y2][Z2] = (T t_nl[X1+1][Y2][Z2] + T t_nl[X1][Y2-1][Z2] + T t_nl[X1][Y2+1][Z2]
+ T t_nl[X1][Y2][Z2-1])/4;
T t_nl[X2][Y1][Z2] = (T t_nl[X2+1][Y1][Z2] + T t_nl[X2][Y1+1][Z2] + T t_nl[X2][Y1][Z2-1])/3;
T t_nl[X2][Y2][Z2] = (T t_nl[X2+1][Y2][Z2] + T t_nl[X2][Y2+1][Z2] + T t_nl[X2][Y2][Z2-1])/3;
T t_nl[X3][Y1][Z2] = (T t_nl[X3+1][Y1][Z2] + T t_nl[X3][Y1-1][Z2] + T t_nl[X3][Y1][Z2-1])/3;
T t_nl[X3][Y2][Z2] = (T t_nl[X3+1][Y2][Z2] + T t_nl[X3][Y2-1][Z2] + T t_nl[X3][Y2][Z2-1])/3;
T t_nl[X4][Y1][Z2] = (T t_nl[X4+1][Y1][Z2] + T t_nl[X4][Y1-1][Z2] + T t_nl[X4][Y1][Z2-1])/4;
T t_nl[X4][Y2][Z2] = (T t_nl[X4+1][Y2][Z2] + T t_nl[X4][Y2-1][Z2] + T t_nl[X4][Y2][Z2-1])/4;

///////////////////////////////////////////
// calculate the sum of square error
maxErr = 0;

double temp;
for(i=1; i<NX; i++)
    for(j=1; j<NY; j++)
        for(z=1; z<NZ3; z++)
            {
                temp = fabs(T t_nl[i][j][z] - T t_nl[i][j][z]);
                if(temp > maxErr)
                    maxErr = temp;
            }

// store result of loop 1
memcpy(T t_n1_1, T t_n1, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
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));
    memset(Tt, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(Tt_n1, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(Tt_n1, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
    memset(Tb1, 0, sizeof(double)*(LZ1+1));
    memset(Tb2, 0, sizeof(double)*(LX2+1));
    memset(Tb3, 0, sizeof(double)*(LZ3+1)*2);
    memset(Tb1_n1, 0, sizeof(double)*(LZ1+1));
    memset(Tb2_n1, 0, sizeof(double)*(LX2+1));
    memset(Tb3_n1, 0, sizeof(double)*(LZ3+1)*2);
    memset(Tv1_n1, 0, sizeof(double)*(LZ1+1));
    memset(Tv2_n1, 0, sizeof(double)*(LX2+1));
    memset(Tv3_n1, 0, sizeof(double)*(LZ3+1)*2);

    Tb1[0] = THETAO;
    Tb1_n1[0] = THETAO;
    double deltaTheta = 0; // Tb1[0] / (LZ1+LX2+LZ3);
    return;
}

// Initialize the laser power
void InitQ(double P0)
{
    int i, j, z;
    double shift;

    for (i = 0; i <= NX; i++)
    {
        for (j = 0; j <= NY; j++)
        {
            shift = exp(-((i-centerX) * (i-centerX) * deltaX * deltaX
                             + (j-centerY) * (j-centerY) * deltaY * deltaY) / (2 * Sigma * Sigma))
                     / (sqrt(2 * pi) * Sigma);

            for (z = 0; z <= NZ1; z++)
                Q1[i][j][z] = Alpha1 * exp(-Alpha1 * z * deltaZ) * shift * P0 * (1 - Reff1);
            for (z = NZ1+1; z <= NZ2; z++)
                Q1[i][j][z] = Alpha1 * exp(-Alpha1 * z * deltaZ) * shift * P0 * (1 - Reff1);
        }
    }
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Q2[i][j][z] = Alpha2 * exp(-Alpha2*(z-NZ1)*deltaZ -Alpha1*deltaZ*NZ1) * shift * P0*(1-Refl2);

for(z=NZ2+1;z<=NZ3;z++)
    Q3[i][j][z] = Alpha3 * exp(-Alpha3*(z-NZ2)*deltaZ -Alpha1*deltaZ*NZ1 -Alpha2*deltaZ*(NZ2-NZ1)) * shift * P0*(1-Refl3);}

}
}

double CalcNewP(double *Tm_pre, double *Tm0, double *Tm1, double P0, double deltaP)
{
    int i;
    double P1_new;
    double alphaStar = 0;

double X[M];
    for(i=0;i<M;i++)
        X[i] = (Tm1[i] - Tm0[i]) / deltaP;

double double factor = 0;
    for(i=0;i<M;i++)
        factor += X[i]*X[i];
    factor += alphaStar;

double double temp=0;
    for(i=0;i<M;i++)
        temp += X[i] * (Tm_pre[i]-Tm1[i]);

    P1_new = P0 + 1/factor * temp;
    return P1_new;
}

int getTm(double *Tm)
{
    //assign surface temperature to Tm array here
    //for M==5 right now
    if(bReachTop)
        Tm[0] = Tm_pre[0];
    else
        Tm[0] = Tt[NX/2][NY/2][0];

    Tm[1] = Tt[NX/2][0][0];
    Tm[2] = Tt[NX/2][NY][0];
    Tm[3] = Tt[0][NY/2][0];
    Tm[4] = Tt[NX][NY/2][0];

    return(1);
}

void AdjustPower(double P0)
{

if(bPowerOn)
{
    // power is on now
    if(Tt_n1[NX/2][NY/2][0] >= 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;
        bReachTop = true;
        // writeLog("= = = = Power Off= = = =");
    }
}
else
{
    // power is off now
    if(Tt_n1[NX/2][NY/2][0] <= Tm_pre[0]-4)
    {
        // set power on
        InitQ(P0);
        bPowerOn = true;
        sprintf(tmp, "= = = = Power On:%7.4lf", P0);
        writeLog(tmp);
    }
}
return;

void writeSquareXY(int x0, int y0, int x1, int y1, int z, int t, int I)
{
    if(!PrintOut(t))
        return;

    FILE *file;
    char fname[256], line[8196];

    strcpy(fname, outPath);
    sprintf(tmp, "%d_Z%d_%d_%d_%d_%d_%d_t%d_I%d.txt", loopP, z, x0, y0, x1, y1, t, I);
    strcat(fname, tmp);
    file = fopen(fname, "w");

    for(int j=y0;j<=y1;j++)
    {
        strcpy(line, "");
        for(int i=x0;i<=x1;i++)
        {
            sprintf(tmp, "%7.4lf", Tt_n1[i][j][z]);
            strcat(line, tmp);
        }
        strcat(line, "n");
    }
}
```c
void writeSquareXZ(int x0, int z0, int x1, int z1, int y, int t, int I)
{
    if(!PrintOut(t))
        return;

    FILE *file;
    char fname[256], line[8196];
    strcpy(fname, outPath);
    sprintf(tmp, "%d_Y%d_%d_%d_%d_%d_t%d_t%d.txt", loopP, y, x0, z0, x1, z1, t, I);
    strcat(fname, tmp);
    file = fopen(fname, "w");
    for(int k=z0;k<=z1;k++)
    {
        strcpy(line, "
        for(int i=x0;i<=x1;i++)
        {
            sprintf(tmp, "%7.41f", Tt_n1[i][y][k]);
            strcat(line, tmp);
        }
        strcat(line, ":t
        fwrite(line,strlen(line), 1, file);
    }
    fclose(file);
}

void writeSquareYZ(int y0, int z0, int y1, int z1, int x, int t, int I)
{
    if(!PrintOut(t))
        return;

    FILE *file;
    char fname[256], line[8196];
    strcpy(fname, outPath);
    sprintf(tmp, "%d_X%d_%d_%d_%d_%d_t%d_t%d.txt", loopP, x, y0, z0, y1, z1, t, I);
    strcat(fname, tmp);
    file = fopen(fname, "w");
    for(int k=z0;k<=z1;k++)
    {
        strcpy(line, ");
        for(int j=y0;j<=y1;j++)
```
{ 
    sprintf(tmp, "%7.41f", T_t_n1[x][j][k]);
    strcat(line, tmp);
}
strcat(line, ";n");
fwrite(line,strlen(line), 1, file);
}
fclose(file);

void writeZCenter(int t)
{
    if(!PrintOut(t))
        return;

    FILE *file;
    char fname[256], line[8196];

    strcpy(fname, outPath);
    sprintf(tmp, "%d_Z_Center_t%d.txt", loopP, t);
    strcat(fname, tmp);
    file = fopen(fname, "w");

    for(int j=0;j<=NZ3;j++)
    {
        sprintf(tmp, "%7.41f", T_t_n1[centerX][centerY][j]);
        strcat(line, tmp);
        strcat(line, ";n");

        fwrite(line,strlen(line), 1, file);
    }
    fclose(file);
}

void writeAll(int t, int I)
{
    if(!PrintOut(t))
        return;

    FILE *file;
    char fname[256];

    strcpy(fname, outPath);
    sprintf(tmp, "%d_t%d_I%d.dat", loopP, t, I);
    strcat(fname, tmp);
    file = fopen(fname, "wb");

    for(int i=0;i<NX+1;i++)
        fwrite(T_t_n1[i],sizeof(double)*(NY+1)*(NZ3+1), 1, file);
    fclose(file);
return;
}

void writeLinearSys(int x, int y, int t, int I)
{
    FILE *file;
    char fname[256], line[256];
    strcpy(fname, outPath);
    sprintf(tmp, "%d_X%d_Y%d_t%d_I%d.txt", loopP, x, y, t, I);
    strcat(fname, tmp);
    file = fopen(fname, "w");
    for(int k=0;k<=NZ3;k++)
    {
        sprintf(line, "%d	%7.41f	%7.41f	%7.41f	%7.41f\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);
}

void writeQ(double P, double sigma)
{
    FILE *file;
    char fname[256], line[20000];
    strcpy(fname, outPath);
    sprintf(tmp, "sigma_%6.41f_p_%6.41f.txt", sigma, P);
    strcat(fname, tmp);
    file = fopen(fname, "w");
    int k;
    for(k=0;k<=NZ1;k++)
    {
        strcpy(line, "\n");
        for(int i=0;i<=NX;i++)
        {
            sprintf(tmp, "%7.41f", Q1[i][NY/2][k]);
            strcat(line, tmp);
        }
        strcat(line, "\n");
        fwrite(line,strlen(line), 1, file);
    }
    for(k=NZ1+1;k<=NZ2;k++)
    {
        strcpy(line, "\n");
        for(int i=0;i<=NX;i++)
        {
            sprintf(tmp, "%7.41f", Q2[i][NY/2][k]);
            strcat(line, tmp);
        }
        strcat(line, "\n");
        fwrite(line,strlen(line), 1, file);
    }
}
strcat(line, tmp);
}
strcat(line, "\n");
fwrite(line,strlen(line), 1, file);
}
for(k=NZ2+1;k<=NZ3;k++)
{
    strcpy(line, "\n");
    for(int i=0;i<=NX;i++)
    {
        sprintf(tmp, "%.7f", Q3[i][NY/2][k]);
        strcat(line, tmp);
    }
    strcat(line, "\n");
    fwrite(line,strlen(line), 1, file);
}
fclose(file);

void writeLog(char *ln)
{
    FILE *file;
    char fname[256], line[512];
    sprintf(fname, "%slog%d.txt", outPath, loopP);
    //strcpy(fname, outPath);
    //strcat(fname, "log.txt");
    file = fopen(fname, "a");
    strcpy(line, ln);
    strcat(line, "\n");
    fwrite(line,strlen(line), 1, file);
    fclose(file);
}

bool PrintOut(int tm)
{
    bool out = false;
    if(outT[0] == -1)
       return true;
    for(int i=0;i<numT;i++)
    {
        if(outT[i] == tm*deltaT)
        {
            out = true;
            break;
        }
    }
    return out;
}
APPENDIX B

SOURCE CODE FOR EXAMPLE 2 AND 3
Author: Xingui Tang
Date: 4/2/2006
Description: This program is used to calculate the optimal temperature distribution in a 3D triple-layered skin structure embedded with a multi-level artery and a multi-level vein. The models to describe the thermal behavior in tissue and blood are modified Penns model and energy balance equations. The blood is considered to be in steady state in Example 2 and to be in dynamic state in Example 3, so the numerical methods of solving the equations for them are different. We worked on two cases for each example, one of them does not consider the heat convection between the skin surface and the environment and the other considers. Use the macro definition of “DYN_MEM” to switch between Example 2 and Example 3 and “CONVECTION” to switch between Case 1 and Case 2 for both examples.

#include "string.h"
#include <math.h>
#include <stdio.h>
#include "memory.h"

//#define SCREEN_OUT
// #define CONVECTION // switch for Case 1 and Case 2
// #define DYN_MEM // switch for Example 2 and Example 3

#define SCALE_X 1
#define SCALE_Y 1
#define SCALE_Z 1

#define M 5
#define NZ1 8*SCALE_Z
#define NZ2 208*SCALE_Z
#define NZ3 1208*SCALE_Z
#define NX 50*SCALE_X
#define NY 50*SCALE_Y
#define LX1A 5*SCALE_X
#define LY1A 5*SCALE_Y
#define LZ1A 400*SCALE_Z
#define LX2A 28*SCALE_X
#define LY2A 4*SCALE_Y
#define LZ2A 80*SCALE_Z
#define LX3A 3*SCALE_X
#define LY3A 3*SCALE_Y
#define LZ3A 200*SCALE_Z

#define LX1B LX1A
#define LY1B LY1A
#define LZ1B LZ1A
#define LX2B LX2A
#define LY2B LY2A
#define LZ2B LZ2A
#define LX3B LX3A
#define LY3B LY3A
#define LZ3B LZ3A

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
float deltaT = 0.1f;
float deltaX = 0.02f / SCALE_X, deltaY = 0.02f / SCALE_Y, deltaZ = 0.001f / SCALE_Z;

float L1, L2, L3;
float Lb1[2], Lb2[2], Lb3[2];
float P1[2], P2[2], P3[2];
float F1[2], F2[2], F3[2];
float M1[2], M2[2], M3[2];

float CB = 4.134f;
float Cb1 = 0.0f, Cb2 = 4.2f, Cb3 = 4.2f;
float v1 = 8.0f;
float alpha = 0.2f;
float Pdot = 0.5e-3f;
float p1 = 1.2f, p2 = 1.2f, p3 = 1.0f;
float C1 = 3.6f, C2 = 3.4f, C3 = 3.06f;
float k1 = 0.0026f, k2 = 0.0052f, k3 = 0.0021f;
float Wb1 = 0.0f, Wb2 = 0.0005f, Wb3 = 0.0005f;
float Alpha1 = 1.8f, Alpha2 = 1.8f, Alpha3 = 1.8f;
float Sigma = 0.01f; //0.015f;
float Reff1 = 0.1f, Reff2 = 0.1f, Reff3 = 0.1f;

double Hf = 0.001;
double Tf = 0; // -17;

int centerX = NX / 2;
int centerY = NY / 2;

const float pai = 3.14159265358979f;
const float omega = 1.0f;
float Bi = alpha / k3;
float factor1[2], factor2[2], factor3[2];
float THETAO = 1.8f;

#define DYN_MEM
  float (*Q1)[NY+1][NZ3+1];
  float (*Q2)[NY+1][NZ3+1];
  float (*Q3)[NY+1][NZ3+1];
  float (*T)[NY+1][NZ3+1];
  float (*T_t_n1)[NY+1][NZ3+1];
  float (*T_t_n1_1)[NY+1][NZ3+1];
  float **Tb1;
  float **Tb2;
  float **Tb3;
  float **Tb1_n1;
  float **Tb2_n1;
  float **Tb3_n1;
  float **Tv1_n1;
  float **Tv2_n1;
  float **Tv3_n1;

  float (*a)[NY+1][NZ3+1], (*b)[NY+1][NZ3+1], (*c)[NY+1][NZ3+1], (*d)[NY+1][NZ3+1];
  float (*a0)[NY+1][NZ3+1], (*b0)[NY+1][NZ3+1], (*c0)[NY+1][NZ3+1];
#endif

#else

#endif
float Q1[NX+1][NY+1][NZ3+1];
float Q2[NX+1][NY+1][NZ3+1];
float Q3[NX+1][NY+1][NZ3+1];
float T1[NX+1][NY+1][NZ3+1];
float T1_n1[NX+1][NY+1][NZ3+1];
float T1_n1_1[NX+1][NY+1][NZ3+1];
double Tb1[2][LZ1A+1];
double Tb2[2][LX2A+1];
double Tb3[2][(LZ3A+1)*2];
double Tb1_n1[2][LZ1A+1];
double Tb2_n1[2][LX2A+1];
double Tb3_n1[2][(LZ3A+1)*2];
double Tv1_n1[2][LZ1A+1];
double Tv2_n1[2][LX2A+1];
double Tv3_n1[2][(LZ3A+1)*2];
double a[NX+1][NY+1][NZ3+1], b[NX+1][NY+1][NZ3+1], c[NX+1][NY+1][NZ3+1],
d[NX+1][NY+1][NZ3+1];
double a0[NX+1][NY+1][NZ3+1], b0[NX+1][NY+1][NZ3+1], c0[NX+1][NY+1][NZ3+1];
#endif

const int SPAN = 3;
int X1[2], X2[2], X3[2], X4[2], X5[2], X6[2], Y1[2], Y2[2], Y3[2], Y4[2], Y5[2], Y6[2], Z1[2], Z2[2],
Z3[2]; //grid points index on blood vessels

int i, j, z;
int t, I, loopP;

bool bPowerOn;

void initialize();
int CalcAll(float P0, bool dP);
int getTv_blood(int index);
int CalcTb();
int CalcTb2();
float CalcTt();
void CalcVessel(int index);
void Reset(void);
void InitQ(float P0);
float CalcNewP(float *Tm_pre, float *Tm0, float *Tm1, float P0, float deltaP);
int getTm(float *Tm);
void setVesselBorder();
void AdjustPower(float P0);
void AdjustMtx(int index);
void clearMtx();

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 writeQ(float P, float sigma);
void writeAll(int t, int I);
void writeLinearSys(int x, int y, int i, int I);
void writeLog(char *line);
bool PrintOut(int tm);

char outPath[255] = "/;";
```c
#define numT 37
int outT[numT] = {-1, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 
                  550, 600, 650, 700, 750, 800, 850, 900, 950, 1000, 
                  1050, 1100, 1150, 1200, 1250, 1300, 1350, 1400, 1450, 1500, 
                  1550, 1600, 1650, 1700, 1750, 1800};

float Tm_pre[5] = {8, 2, 2, 2, 2}; // pre-specified surface temperature;
bool bReachTop;
char tmp[256];

int TOTAL_T = 400;
float Err_I = 0.001f; //for I loop
float Err_P = 0.01f; //for P loop

void testP()
{

double p[10] = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0};
double sigma[10] = {0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1};

for(int j=0;j<10;j++)
    for(int i=0;i<10;i++)
        {
            Sigma = (float)sigma[i];
            InitQ((float)p[i]);
            writeQ((float)p[i], Sigma);
        }

return;
}

int main(void)
{
    float Tm0[5]; //calculated surface temperatures - first run
    float Tm1[5]; //calculated surface temperatures - second run - with updated power level
    float P0, P1_LL; //power level
    float deltaP; //power step
    float oldSP, newSP;
    int indexM;
    initialize(); //memory allocation & variable initialization

    P0 = 1.2f; //set initial power level;
    loopP = 0;
    sprintf(tmp, "Initial P:%5.4lf", P0);
    writeLog(tmp);
    newSP=0.0f;
```
CalcAll(P0,0);
getTm(Tm0);
deltaP = P0/100.f;

do{
    oldSP=newSP;
P0 += deltaP;

    //calculate tissue and blood temperature based on specified power level
    CalcAll(P0,0);
getTm(Tm1);

    //calculate the next power level based on prespecified temperature and calculated surface
    //temperature
    P1_LI = CalcNewP(Tm_pre, Tm0, Tm1, P0, deltaP);

deltaP = P1_LI - P0;
    //if(deltaP > P0/100.f)
    //    deltaP = P0/100.f;

oldSP=0;
newSP=0;
for (indexM =0; indexM<M; indexM++)
{
    oldSP=oldSP+ (Tm_pre[indexM]-Tm0[indexM])*(Tm_pre[indexM]
                 -Tm0[indexM]);
    newSP=newSP+ (Tm_pre[indexM]-Tm1[indexM])*(Tm_pre[indexM]
                 -Tm1[indexM]);
}

sprintf(tmp, "Tm0[0]=%6.4lf; Tm0[1]=%6.4lf; Tm0[2]=%6.4lf; Tm0[3]=%6.4lf;
Tm0[4]=%6.4lf;", 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;", 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_LI, newSP, oldSP,(newSP-oldSP)/newSP);
writeLog(tmp);

for (indexM =0; indexM<M; indexM++)
    Tm0[indexM] = Tm1[indexM];

loopP++;
    if (newSP==0) break;
}
while(fabs((newSP-oldSP)/newSP) > Err_P);

clearMem();
return 1;
/memory allocation & variable initialization

void initialize()
{
    #ifdef DYN_MEM
        Q1 = new float[NX+1][NY+1][NZ+1];
        Q2 = new float[NX+1][NY+1][NZ+1];
        Q3 = new float[NX+1][NY+1][NZ+1];
        Tt = new float[NX+1][NY+1][NZ+1];
        Tt_n1 = new float[NX+1][NY+1][NZ+1];
        Tt_n1_1 = new float[NX+1][NY+1][NZ+1];
        a = new float[NX+1][NY+1][NZ+1];
        b = new float[NX+1][NY+1][NZ+1];
        c = new float[NX+1][NY+1][NZ+1];
        d = new float[NX+1][NY+1][NZ+1];
        a0 = new float[NX+1][NY+1][NZ+1];
        b0 = new float[NX+1][NY+1][NZ+1];
        c0 = new float[NX+1][NY+1][NZ+1];
    #endif
    memset(a0, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    memset(b0, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    memset(c0, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    Lb1[0] = LZ1A*deltaZ;
    Lb2[0] = LX2A*deltaX;
    Lb3[0] = LZ3A*deltaZ;
    Lb4[0] = LX3A*deltaX;
    L1 = NZ1*deltaZ;
    L2 = (NZ2-NZ1)*deltaZ;
    L3 = (NZ3-NZ2)*deltaZ;
   -vesSELBorder();
    #ifdef DYN_MEM
        Tb1 = new float*[2];
        Tb2 = new float*[2];
        Tb3 = new float*[2];
        Tb1_n1 = new float*[2];
        Tb2_n1 = new float*[2];
        Tb3_n1 = new float*[2];
        Tb4_n1 = new float*[2];
        Tb5_n1 = new float*[2];
        Tb6_n1 = new float*[2];
        Tb7_n1 = new float*[2];
        Tb8_n1 = new float*[2];
    #endif
    Tb1[0] = new float[LZ1A+1];
    Tb2[0] = new float[LX2A+1];
    Tb3[0] = new float[LZ3A+1]*2;
    Tb1_n1[0] = new float[LZ1A+1];
    Tb2_n1[0] = new float[LX2A+1];
    Tb3_n1[0] = new float[LZ3A+1]*2;
    Tb4_n1[0] = new float[LZ4A+1];
    Tb5_n1[0] = new float[LZ5A+1];
    Tb6_n1[0] = new float[LX6A+1];
    Tb7_n1[0] = new float[LX7A+1];
    Tb8_n1[0] = new float[LX8A+1];
Tv2_nl[0] = new float[(LX2A+1)*2];
Tv3_nl[0] = new float[(LZ3A+1)*2];
Tb1[1] = new float[(LZ1B+1)*2];
Tb2[1] = new float[(LX2B+1)*2];
Tb3_nl[1] = new float[(LZ3B+1)*2];
Tv1_nl[1] = new float[(LZ1B+1)*2];
Tv2_nl[1] = new float[(LX2B+1)*2];
Tv3_nl[1] = new float[(LZ3B+1)*2];
#endif

P1[0] = (LX1A*deltaX + LY1A*deltaY)*2;
P2[0] = (LY2A*deltaY + LZ2A*deltaZ)*2;
P3[0] = (LX3A*deltaX + LY3A*deltaY)*2;
P1[1] = (LX1B*deltaX + LY1B*deltaY)*2;
P2[1] = (LY2B*deltaY + LZ2B*deltaZ)*2;
P3[1] = (LX3B*deltaX + LY3B*deltaY)*2;

F1[0] = (LX1A*deltaX) * (LY1A*deltaY);
F2[0] = (LY2A*deltaY) * (LZ2A*deltaZ);
F3[0] = (LX3A*deltaX) * (LY3A*deltaY);
F1[1] = (LX1B*deltaX) * (LY1B*deltaY);
F2[1] = (LY2B*deltaY) * (LZ2B*deltaZ);
F3[1] = (LX3B*deltaX) * (LY3B*deltaY);

M1[0] = v1 * F1[0];
M2[0] = 0.5f * M1[0]; //
M3[0] = 0.5f * M1[0]; //
M1[1] = v1 * F1[1];
M2[1] = 0.5f * M1[1]; //
M3[1] = 0.5f * M1[1]; //

factor1[0] = deltaZ*alpha*P1[0]/(M1[0]*CB);
factor2[0] = deltaX*alpha*P2[0]/(M2[0]*CB);
factor3[0] = deltaZ*alpha*P3[0]/(M3[0]*CB);
factor1[1] = -deltaZ*alpha*P1[1]/(M1[1]*CB);
factor2[1] = -deltaX*alpha*P2[1]/(M2[1]*CB);
factor3[1] = -deltaZ*alpha*P3[1]/(M3[1]*CB);

bPowerOn = true;
bReachTop = false;

writeLog("Initialization...");
sprintf(tmp, "Lb1:%.5.4lf Lb2:%.5.4lf Lb3:%.5.4lf | Lb1:%.5.4lf Lb2:%.5.4lf Lb3:%.5.4lf", Lb1[0], Lb2[0], Lb3[0], Lb1[1], Lb2[1], Lb3[1]);
writeLog(tmp);
sprintf(tmp, "L1:%.5.4lf L2:%.5.4lf L3:%.5.4lf", L1, L2, L3);
writeLog(tmp);
sprintf(tmp, "P1:%.5.4lf P2:%.5.4lf P3:%.5.4lf | P1:%.5.4lf P2:%.5.4lf P3:%.5.4lf", P1[0], P2[0], P3[0], P1[1], P2[1], P3[1]);
writeLog(tmp);
sprintf(tmp, "F1:%.5.4lf F2:%.5.4lf F3:%.5.4lf | F1:%.5.4lf F2:%.5.4lf F3:%.5.4lf", F1[0], F2[0], F3[0], F1[1], F2[1], F3[1]);
writeLog(tmp);
```c
sprintf(tmp, "M l:% 5.41f M 2:% 5.41f M 3:% 5.41f | M l:% 5.41f M 2:% 5.41f M 3:% 5.41f', M 1[l], M 2[l], M 3[l]);
writeLog(tmp);
sprintf(tmp, "X 1:% d X 2:% d X 3:% d X 4:% d X 5:% d X 6:% d | X 1:% d X 2:% d X 3:% d X 4:% d X 5:% d X 6:% d', X 1[0], X 2[0], X 3[0], X 4[0], X 5[0], X 6[0], X 1[l], X 2[l], X 3[l], X 4[l], X 5[l], X 6[l]);
writeLog(tmp);
sprintf(tmp, "Y 1:% d Y 2:% d Y 3:% d Y 4:% d Y 5:% d Y 6:% d | Y 1:% d Y 2:% d Y 3:% d Y 4:% d Y 5:% d Y 6:% d', Y 1[0], Y 2[0], Y 3[0], Y 4[0], Y 5[0], Y 6[0], Y 1[l], Y 2[l], Y 3[l], Y 4[l], Y 5[l], Y 6[l]);
writeLog(tmp);
sprintf(tmp, "Z 1:% d Z 2:% d Z 3:% d | Z 1:% d Z 2:% d Z 3:% d", Z 1[0], Z 2[0], Z 3[0], Z 1[l], Z 2[l], Z 3[l]);
writeLog(tmp);
sprintf(tmp, "Ref f 1:% 5.3lf Ref f 2:% 5.3lf Ref f 3:% 5.3lf'', Ref f 1, Ref f 2, Ref f 3);
writeLog(tmp);
sprintf(tmp, "Alpha 1:% 5.3lf Alpha 2:% 5.3lf Alpha 3:% 5.3lf'', Alpha 1, Alpha 2, Alpha 3);
writeLog(tmp);
sprintf(tmp, "Sigma:% 5.3lf'', Sigma);
writeLog(tmp);

//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++)
    {
      b0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);
      a0[i][j][z] = 2*p1*C1 + Wb1*Cb1*deltaT
        + (4*k1*deltaT)*(1.0f/(deltaX*deltaX) + 1.0f/(deltaY*deltaY))
        + (2*k1*deltaT)/(deltaZ*deltaZ);
      c0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);
    }
    
    b0[i][j][NZ1-1]=k1;
    a0[i][j][NZ1]=k1+k2;
    c0[i][j][NZ1]=-k2;
  }
  for(z=NZ1-1;z<=NZ2-1;z++)
  {
    b0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);
    a0[i][j][z] = 2*p2*C2 + Wb2*Cb2*deltaT
        + (4*k2*deltaT)*(1.0f/(deltaX*deltaX) + 1.0f/(deltaY*deltaY))
        + (2*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++)
  {
    b0[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);
  }

  for(z=1;z<=NZ4-1;z++)
  {
    b0[i][j][z] = -(k4*deltaT)/(deltaZ*deltaZ);
  }
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ a_{0[i][j][z]} = 2*p3*C3 + (4*k3*deltaT)*(1.0f/(deltaX*deltaX) \\
+ 1.0f/(deltaY*deltaY)) + Wb3*Cb3*deltaT \\
+ (2*k3*deltaT)/(deltaZ*deltaZ); \]
\[ c_{0[i][j][z]} = -(k3*deltaT)/(deltaZ*deltaZ); \]
Y₃[0] = (NY-LY2A)/2;
Y₄[0] = Y₃[0] + LY2A;
Y₁[0] = (NY-LY3A)/2;
Y₂[0] = Y₁[0] + LY3A;

Z₃[0] = NZ3 - LZ1A;
Z₂[0] = Z₃[0] - LZ2A;
Z₁[0] = Z₂[0] - LZ3A;

X₅[1] = (NX-LX1B)/2;
X₆[1] = X₅[1] + LX1B;
X₁[1] = (NX-LX2B)/2;

Y₅[1] = (NY-LY1B)/2;
Y₃[1] = (NY-LY2B)/2;
Y₁[1] = (NY-LY3B)/2;

Z₃[1] = NZ3 - LZ1B;

//seperate the blood vessels
int dY1 = Y₆[0] - (NY-SPAN)/2;
int dY2 = (NY-SPAN)/2 + SPAN - Y₅[1];
Y₁[0] -= dY1;
Y₂[0] -= dY1;
Y₃[0] -= dY1;
Y₄[0] -= dY1;
Y₅[0] -= dY1;
Y₆[0] -= dY1;

Y₁[1] += dY2;
Y₂[1] += dY2;
Y₃[1] += dY2;
Y₄[1] += dY2;
Y₅[1] += dY2;
Y₆[1] += dY2;
return;

int CalcAll(float P0, bool dp)
{
  float maxErr, oldE; //sum of square error of tissue temperatures

  Reset();
  InitQ(P0);

  //writeQ(P0, Sigma);
}
t = 0;
while((t*deltaT < TOTAL_T))
{
    t++;
    I = 0;
    maxErr = 0.0;
    oldE = 99999999.0f;
    do//I iteration to calculate temperature at time level n+1
        I++;
        //interpolate blood vessel temperature based on tissue temperature
        getTv_blood(0);
        getTv_blood(1);
        //calculate blood temperature based on given vessel temperature
        CalcTb();
        CalcTb2();
        //interpolate blood temperature at for use of blood vessel interface equation
        //getTv_tissue();
        //Calculate tissue temperature
        maxErr = CalcTt();
        if(maxErr>=oldE)
            { writeLog("========unstable========");
                #ifdef SCREEN_OUT
                printf("========
");
                #endif
                writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
                break;
            }
        oldE = maxErr;
    }while(maxErr>Err_I);
    if(bPowerOn)
    {
        if (dp)
            sprintf(tmp, "p%d t:2d I:2d Err:%5.4lf T0:%7.4lf T1:%7.4lf T2:%7.4lf P:%7.4lf", loopP, t, I, maxErr, T_t_n1[NX/2][NY/2][0],
            T_t_n1[NX/2][NY/2][0], T_t_n1[NX/4][NY/2][NZ3], P0);
        else
            sprintf(tmp, "p%d t:2d I:2d Err:%5.4lf T0:%7.4lf T1:%7.4lf T2:%7.4lf P:%7.4lf", loopP, t, I, maxErr, T_t_n1[NX/2][NY/2][0],
            T_t_n1[NX/2][NY/2][0], T_t_n1[NX/4][NY/2][NZ3], P0);
    }
    else
    {
        if (dp)
            { if (dp)


```
sprintf(tmp, "p%d t:%2d I:%d Err:%5.41f T0:% 7.41f T l:% 7.41f T2:% 7.41f P:%7.41f", loopP, t, I, maxErr, Tt_n1[NX/2][NY/2][0], Tt_n1[0][NY/2][0][N X /2][N Y /2][NZ3], 0.0);
else
    sprintf(tmp, "p%d t:%2d I:%d Err:%5.41f T0:% 7.41f T l:% 7.41f T2:% 7.41f P:%7.41f", loopP, t, I, maxErr, Tt_n1[NX/2][NY/2][0], Tt_n1[0][NY/2][0][N X /4][N Y /2][NZ3], 0.0);
}
writeLog(tmp);

#ifdef SCREEN_OUT
printf(tmp); printf("n");
#endif

if(t%1000==0)
{
    writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
    writeSquareXZ(0, 0, NX, NZ3, (Y5[0]+Y6[0])/2, t, I);
    writeSquareXZ(0, 0, NX, NZ3, (Y5[1]+Y6[1])/2, t, I);
    writeSquareYZ(0, 0, NY, NZ3, NX/2, t, I);
}
/*if(!dp){ */
    if((t % 1000)==0){
        writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
        //writeSquareYZ(0, 0, NY, NZ3, NX/2, t, I);
    }
    //if(t%200==0)
    //writeAll(t, I);
    else if ((Tt_n1[NX/2][NY/2][0])>=8)
    {
        writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
        writeSquareXZ(0, 0, NX, NZ3, (Y5[0]+Y6[0])/2, t, I);
        writeSquareXZ(0, 0, NX, NZ3, (Y5[1]+Y6[1])/2, t, I);
        writeSquareYZ(0, 0, NY, NZ3, NX/2, t, I);
        //writeZCenter(t);
    }

memcpy(Tt, Tt_n1, sizeof(float)*(NX+1)*(NY+1)*(NZ3+1));
AdjustPower(P0);
}
```

//Record the temperature of the last second
//writeSquareXZ(0, 0, NX, NZ3, NY/4, t, I);
//writeSquareXZ(0, 0, NX, NZ3, 3*NY/4, t, I);
writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
writeSquareXZ(0, 0, NX, NZ3, (Y5[0]+Y6[0])/2, t, I);
writeSquareXZ(0, 0, NX, NZ3, (Y5[1]+Y6[1])/2, t, I);
writeSquareYZ(0, 0, NY, NZ3, NX/2, t, I);
//writeZCenter(t-1);

return(1);
```
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
    int i;
    int x1 = XI[index], x2 = X2[index], x3 = X3[index], x4 = X4[index], x5 = X5[index],
        x6 = X6[index];
    int y1 = Y1[index], y2 = Y2[index], y3 = Y3[index], y4 = Y4[index], y5 = Y5[index],
        y6 = Y6[index];
    int z1 = Z1[index], z2 = Z2[index], z3 = Z3[index];
    int lzl = index==0?LZlA:LZlB;
    int lx2 = index==0?LX2A:LX2B;
    int lzl3 = index==0?LZ3A:LZ3B;

    //first level
    for(i=0;i<lzl;i++)
        Tvl_nl[index][i] = ( Tt_nl[x5][y5][NZ3-i] + Tt_nl[x6][y5][NZ3-i] +
            Tt_nl[x5][y6][NZ3-i] + Tt_nl[x6][y6][NZ3-i] ) / 4.0f;
    Tvl_nl[index][lzl] = ( Tt_nl[x5][y3][z3] + Tt_nl[x6][y3][z3] + Tt_nl[x5][y4][z3] +
            Tt_nl[x6][y4][z3] ) / 4.0f;

    //second level
    for(i=0;i<lx2;i++) //i=0 & LX2 are on the blood vessels
        Tv2_nl[index][i] = ( Tt_nl[i+x1][y3][z2] + Tt_nl[i+x1][y4][z2] + Tt_nl[i+x1][y3][z3] +
            Tt_nl[i+x1][y4][z3] ) / 4.0f;

    //third level
    for(i=0;i<lzl3;i++)
    {
        Tv3_nl[index][i] = ( Tt_nl[x1][y1][z2-i] + Tt_nl[x2][y1][z2-i] +
            Tt_nl[x1][y2][z2-i] + Tt_nl[x2][y2][z2-i] ) / 4.0f;
        Tv3_nl[index][i+lzl3+1] = ( Tt_nl[x3][y1][z2-i] + Tt_nl[x4][y1][z2-i] +
            Tt_nl[x3][y2][z2-i] + Tt_nl[x4][y2][z2-i] ) / 4.0f;
    }
    return(1);
}

int CalcTb()
{
    int lzl = LZ1A, lx2 = LX2A, ly2 = LY2A, lz2 = LZ2A, lx3 = LX3A, lz3 = LZ3A;
    int index = 0;
    int x1 = XI[index], x2 = X2[index], x3 = X3[index], x4 = X4[index], x5 = X5[index],
        x6 = X6[index];
    int y1 = Y1[index], y2 = Y2[index], y3 = Y3[index], y4 = Y4[index], y5 = Y5[index],
        y6 = Y6[index];
    int z1 = Z1[index], z2 = Z2[index], z3 = Z3[index];

    int i;
    float xx = -1;
    float fk1, fk2, fk3, fk4;
    Tb1_n1[index][0] = THETA0;
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
//first level blood
for(i=1;i<=lz1;i++)
{

#ifndef DYNAMIC
Tb1_n1[index][i] = (CB*F1[index]*xx/deltaT*TB1[index][i] - alpha*P1[index]*TV1_n1[index][i] - CB*M1[index]/deltaZ*TB1_n1[index][i-1]) / (CB*F1[index]*xx/deltaT - alpha*P1[index] - CB*M1[index]/deltaZ);
#else
fk1 = factor1[index]*(TV1_n1[index][i-1]- TB1_n1[index][i-1]);
fk2 = factor1[index]*(TV1_n1[index][i-1]-TB1_n1[index][i-1]+fk1/2);
fk3 = factor1[index]*(TV1_n1[index][i-1]-TB1_n1[index][i-1]+fk2/2);
fk4 = factor1[index]*(TV1_n1[index][i-1]-TB1_n1[index][i-1]+fk3);
Tb1_n1[index][i] = Tb1_n1[index][i-1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
#endif
}

//second level blood
int centerX2 = (xl+x4)/2;;
Tb2_n1[index][centerX2 - x1] = Tb1_n1[index][lz1];

//left part
for(i=centerX2-1;i>x1;i--)
{
#ifndef DYNAMIC
Tb2_n1[index][i-x1] = (CB*F2[index]*xx/deltaT*TB2[index][i-x1] - alpha*P2[index]*TV2_n1[index][i-x1] - CB*M2[index]/deltaX*TB2_n1[index][i-x1+1]) / (CB*F2[index]*xx/deltaT - alpha*P2[index] - CB*M2[index]/deltaX);
#else
fk1 = factor2[index]*(TV2_n1[index][i-x1+1]- TB2_n1[index][i-x1+1]);
fk2 = factor2[index]*(TV2_n1[index][i-x1+1]-TB2_n1[index][i-x1+1]+fk1/2);
fk3 = factor2[index]*(TV2_n1[index][i-x1+1]-TB2_n1[index][i-x1+1]+fk2/2);
fk4 = factor2[index]*(TV2_n1[index][i-x1+1]-TB2_n1[index][i-x1+1]+fk3);
Tb2_n1[index][i-x1] = Tb2_n1[index][i-x1+1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
#endif
}

Tb2_n1[index][0] = Tt_n1[(i1][y3+iy2/2][z2+iz2/2]; //the interface grid point between level 1 and level 2

//right part
for(i=centerX2+1;i<x4;i++)
{
#ifndef DYNAMIC
Tb2_n1[index][i-x1] = (CB*F2[index]*xx/deltaT*TB2[index][i-x1] - alpha*P2[index]*TV2_n1[index][i-x1] - CB*M2[index]/deltaX*TB2_n1[index][i-x1-1]) / (CB*F2[index]*xx/deltaT - alpha*P2[index] - CB*M2[index]/deltaX);
#else
fk1 = factor2[index]*(TV2_n1[index][i-x1-1]- TB2_n1[index][i-x1-1]);
fk2 = factor2[index]*(TV2_n1[index][i-x1-1]-TB2_n1[index][i-x1-1]+fk1/2);
fk3 = factor2[index]*(TV2_n1[index][i-x1-1]-TB2_n1[index][i-x1-1]+fk2/2);
fk4 = factor2[index]*(TV2_n1[index][i-x1-1]-TB2_n1[index][i-x1-1]+fk3);
#endif
Tb2_n1[index][i-x1] = Tb2_n1[index][i-x1-1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
#endif
}
Tb2_n1[index][lx2] = Tt_n1_[lx4][y3+ly2/2][z2+lz2/2];

//third level blood
Tb3_n1[index][0] = Tb2_n1[index][lx3/2]; //the interface grid point between level 2 and level 3
for(i=1;i<=lz3;i++)
{
    #ifdef DYNAMIC
        Tb3_n1[index][i] = (CB*F3[index]*xx/deltaT*Tb3[index][i] - alpha*P3[index]*Tv3_n1[index][i] - CB*M3[index]/deltaZ*Tb3_n1[index][i-1] ) / (CB*F3[index]*xx/deltaT - alpha*P3[index] - CB*M3[index]/deltaZ);
    #else
        fk1 = factor3[index]*(Tv3_n1[index][i-1]- Tb3_n1[index][i-1]) + deltaZ*F3[index]*Pdot*Tb3_n1[index][i-1]/M3[index];
        fk2 = factor3[index]*(Tv3_n1[index][i-1]-(Tb3_n1[index][i-1]+fk1/2)) + deltaZ*F3[index]*Pdot*(Tb3_n1[index][i-1]+fk1/2)/M3[index];
        fk3 = factor3[index]*(Tv3_n1[index][i-1]-(Tb3_n1[index][i-1]+fk2/2)) + deltaZ*F3[index]*Pdot*(Tb3_n1[index][i-1]+fk2/2)/M3[index];
        fk4 = factor3[index]*(Tv3_n1[index][i-1]-(Tb3_n1[index][i-1]+fk3)) + deltaZ*F3[index]*Pdot*(Tb3_n1[index][i-1]+fk3)/M3[index];
        Tb3_n1[index][i] = Tb3_n1[index][i-1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
    #endif

    //right part
    Tb3_n1[index][lz3+1] = Tb2_n1[index][lx2-lx3/2]; //the interface grid point between level 2 and level 3
    for(i=1;i<=lz3;i++)
    {
        #ifdef DYNAMIC
            Tb3_n1[index][i+lz3+1] = (CB*F3[index]*xx/deltaT*Tb3[index][i+lz3+1] - alpha*P3[index]*Tv3_n1[index][i+lz3+1] - CB*M3[index]/deltaZ*Tb3_n1[index][i+1+lz3+1] ) / (CB*F3[index]*xx/deltaT - alpha*P3[index] - CB*M3[index]/deltaZ);
        #else
            fk1 = factor3[index]*(Tv3_n1[index][i+1+lz3+1]- Tb3_n1[index][i+1+lz3+1]) + deltaZ*F3[index]*Pdot*Tb3_n1[index][i+1+lz3+1]/M3[index];
            fk2 = factor3[index]*(Tv3_n1[index][i+1+lz3+1]-(Tb3_n1[index][i+1+lz3+1] + fk1/2)) + deltaZ*F3[index]*Pdot*(Tb3_n1[index][i+1+lz3+1]+fk1/2)/M3[index];
            fk3 = factor3[index]*(Tv3_n1[index][i+1+lz3+1]-(Tb3_n1[index][i+1+lz3+1] + fk2/2)) + deltaZ*F3[index]*Pdot*(Tb3_n1[index][i+1+lz3+1]+fk2/2)/M3[index];
            fk4 = factor3[index]*(Tv3_n1[index][i+1+lz3+1]-(Tb3_n1[index][i+1+lz3+1] + fk3)) + deltaZ*F3[index]*Pdot*(Tb3_n1[index][i+1+lz3+1]+fk3)/M3[index];
            Tb3_n1[index][i+1+lz3+1] = Tb3_n1[index][i+1+lz3+1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
        #endif
    }
}
return(1);
int CalcTb2()
{
    int lx1 = LZ1B, lx2 = LX2B, lx3 = LX3B, lx4 = LX4B, lx5 = L
    int index = 1;
    int x1 = X1[index], x2 = X2[index], x3 = X3[index],
    int y1 = Y1[index], y2 = Y2[index], y3 = Y3[index],
    int z1 = Z1[index], z2 = Z2[index], z3 = Z3[index];

    int i;
    float xx = -1;
    float fk1, fk2, fk3, fk4;

    // third level blood
    Tb3_n1[index][lx3] = Tt_n1[(x1+x2)/2][(y1+y2)][z1]; // entry point
    for(i=lx3-1;i>=0;i--)
    {
        #ifdef DYNAMIC
            Tb3_n1[index][i] = (CB*F3[index]*xx/deltaT*Tb3[index][i] - alpha*P3[index]*Tv3_n1[index][i] - CB*M3[index]/deltaZ*Tb3_n1[index][i+1]) / (CB*F3[index]*xx/deltaT - alpha*P3[index] - CB*M3[index]/deltaZ);
        #else
            fk1 = factor3[index]*Tv3_n1[index][i+1] - Tb3_n1[index][i+1]);
            fk2 = factor3[index]*(Tv3_n1[index][i+1]-(Tb3_n1[index][i+1]+fk1/2));
            fk3 = factor3[index]*(Tv3_n1[index][i+1]-(Tb3_n1[index][i+1]+fk2/2));
            fk4 = factor3[index]*(Tv3_n1[index][i+1]-(Tb3_n1[index][i+1]+fk3));
            Tb3_n1[index][i] = Tb3_n1[index][i+1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
        #endif
    }
    // right part
    Tb3_n1[index][lx3*2+1] = Tt_n1[(x3+x4)/2][(y1+y2)][z1]; // entry point
    for(i=lx3-1;i>=0;i--)
    {
        #ifdef DYNAMIC
            Tb3_n1[index][i+lx3+1] = (CB*F3[index]*xx/deltaT*Tb3[index][i+lx3+1] - alpha*P3[index]*Tv3_n1[index][i+lx3+1] - CB*M3[index]/deltaZ*Tb3_n1[index][i+1+lx3+1]) / (CB*F3[index]*xx/deltaT - alpha*P3[index] - CB*M3[index]/deltaZ);
        #else
            fk1 = factor3[index]*(Tv3_n1[index][i+1+lx3+1] - Tb3_n1[index][i+1+lx3+1]);
            fk2 = factor3[index]*(Tv3_n1[index][i+1+lx3+1]-(Tb3_n1[index][i+1+lx3+1]+fk1/2));
            fk3 = factor3[index]*(Tv3_n1[index][i+1+lx3+1]-(Tb3_n1[index][i+1+lx3+1]+fk2/2));
            fk4 = factor3[index]*(Tv3_n1[index][i+1+lx3+1]-(Tb3_n1[index][i+1+lx3+1]+fk3));
            Tb3_n1[index][i+1+lx3+1] = Tb3_n1[index][i+1+lx3+1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
        #endif
    }
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
// second level blood
int centerX2 = (x1+x4)/2;

// left part
Tb2_n1[index][0] = Tb3_n1[index][0];
Tb2_n1[index][1] = Tb3_n1[index][0];
for(i=x1+2; i<=centerX2;i++)
{  
  #ifdef DYNAMIC
    Tb2_n1[index][i-x1] = (CB*F2[index]*xx/deltaT*Tb2[index][i-x1]  
    - alpha*P2[index]*Tv2_n1[index][i-x1] - CB*MB2[index]/deltaX*Tb2_n1[index][i-x1+1] )  
    / (CB*F2[index]*xx/deltaT - alpha*P2[index] - CB*MB2[index]/deltaX);
  #else
    fk1 = factor2[index]*(Tv2_n1[index][i-x1]-Tb2_n1[index][i-x1-1]);
    fk2 = factor2[index]*(Tv2_n1[index][i-x1]-Tb2_n1[index][i-x1+1]+fk1/2));
    fk3 = factor2[index]*(Tv2_n1[index][i-x1]-Tb2_n1[index][i-x1+1]+fk2/2));
    fk4 = factor2[index]*(Tv2_n1[index][i-x1]-Tb2_n1[index][i-x1+1]+fk3));
    Tb2_n1[index][i-x1] = Tb2_n1[index][i-x1-1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
  #endif
  
  Tb1_n1[index][lz1] = Tb2_n1[index][centerX2-x1];
}

// right part
Tb2_n1[index][lx2] = Tb3_n1[index][lx2];
Tb2_n1[index][lx2+1] = Tb3_n1[index][lx2+1];
for(i=x4-2; i>=centerX2;i--)
{  
  #ifdef DYNAMIC
    Tb2_n1[index][i-x1] = (CB*F2[index]*xx/deltaT*Tb2[index][i-x1]  
    - alpha*P2[index]*Tv2_n1[index][i-x1] - CB*MB2[index]/deltaX*Tb2_n1[index][i-x1+1] )  
    / (CB*F2[index]*xx/deltaT - alpha*P2[index] - CB*MB2[index]/deltaX);
  #else
    fk1 = factor2[index]*(Tv2_n1[index][i-x1]+Tb2_n1[index][i-x1+1]);
    fk2 = factor2[index]*(Tv2_n1[index][i-x1]+Tb2_n1[index][i-x1+1]+fk1/2));
    fk3 = factor2[index]*(Tv2_n1[index][i-x1]+Tb2_n1[index][i-x1+1]+fk2/2));
    fk4 = factor2[index]*(Tv2_n1[index][i-x1]+Tb2_n1[index][i-x1+1]+fk3));
    Tb2_n1[index][i-x1] = Tb2_n1[index][i-x1-1] + (fk1 + 2*fk2 + 2*fk3 + fk4)/6;
  #endif
  
  Tb1_n1[index][lz1] = (Tb1_n1[index][lz1] + Tb2_n1[index][centerX2-x1]) / 2.0f;
}

// first level blood
for(i=lz1-1;i>=0;i--)
{  
  #ifdef DYNAMIC
    Tb1_n1[index][i] = (CB*F1[index]*xx/deltaT*Tb1[index][i]  
    - alpha*P1[index]*Tv1_n1[index][i] - CB*MB1[index]/deltaZ*Tb1_n1[index][i+1] )  
    / (CB*F1[index]*xx/deltaT - alpha*P1[index] - CB*MB1[index]/deltaZ);
  #else
    fk1 = factor1[index]*(Tv1_n1[index][i]+Tb1_n1[index][i+1]);
    fk2 = factor1[index]*(Tv1_n1[index][i]+Tb1_n1[index][i+1]+fk1/2));
  #endif
}
float CalcTn1(float Tn1[NX][NY][NZ], float Tb3[NX][NY][NZ], float p1, float C1, float Wb1, float Cb1, float deltaTime, float Tb1[NX][NY][NZ], float Tt[NX][NY][NZ], float k1, float omega)
{
    float maxErr, f;

    //initialize tridiagonal system
    memcpy(a, a0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    memcpy(b, b0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    memcpy(c, c0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    memset(d, 0, sizeof(float)*(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++)
            {
                f = ( (2*p1*C1+Wb1*Cb1*deltaT)*Tt[i][j][z]
                    + (-2*p1*C1+Wb1*Cb1*deltaT)*Tt[i][j][z]
                    -2*Wb1*Cb1*deltaT*(Tb3[i][0][LZ3A])
                    -k1*deltaT*(Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX)
                    + (Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaY*deltaY)
                    + (Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ))
                    -k1*deltaT*(Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ))
                    -2*deltaT*Q1[i][j][z];

                d[i][j][z] = ( 2*p1*C1 + Wb1*Cb1*deltaT
                    + (4*k1*deltaT)*(1.0f/(deltaX*deltaX)+1.0f/(deltaY*deltaY))
                    + (2*k1*deltaT)/(deltaZ*deltaZ) ) * Tt[i][j][z]
                    - k1*deltaT/(deltaZ*deltaZ) * (Tt[i][j][z-1]+Tt[i][j][z+1])
                    - omega * f;
            }
        }
    }

    for(z=NZ1+1; z<=NZ2-1; z++)
    {
        f = ( (2*p2*C2+Wb2*Cb2*deltaT)*Tt[i][j][z]
            + (-2*p2*C2+Wb2*Cb2*deltaT)*Tt[i][j][z]
            -2*Wb2*Cb2*deltaT*(Tb3[n][0][LZ3A])
            -2*deltaT*Q2[i][j][z];

        d[i][j][z] = 0;
    }

    return(1);
}

float CalcTt()
{
    float maxErr, f;

    //initialize tridiagonal system
    memcpy(a, a0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    memcpy(b, b0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    memcpy(c, c0, sizeof(float)*(NX+1)*(NY+1)*(NZ+1));
    memset(d, 0, sizeof(float)*(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++)
            {
                f = ( (2*p1*C1+Wb1*Cb1*deltaT)*Tt[i][j][z]
                    + (-2*p1*C1+Wb1*Cb1*deltaT)*Tt[i][j][z]
                    -2*Wb1*Cb1*deltaT*(Tb3[i][0][LZ3A])
                    -k1*deltaT*(Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX)
                    + (Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaY*deltaY)
                    + (Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ))
                    -k1*deltaT*(Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ))
                    -2*deltaT*Q1[i][j][z];

                d[i][j][z] = ( 2*p1*C1 + Wb1*Cb1*deltaT
                    + (4*k1*deltaT)*(1.0f/(deltaX*deltaX)+1.0f/(deltaY*deltaY))
                    + (2*k1*deltaT)/(deltaZ*deltaZ) ) * Tt[i][j][z]
                    - k1*deltaT/(deltaZ*deltaZ) * (Tt[i][j][z-1]+Tt[i][j][z+1])
                    - omega * f;
            }
        }
    }

    for(z=NZ1+1; z<=NZ2-1; z++)
    {
        f = ( (2*p2*C2+Wb2*Cb2*deltaT)*Tt[i][j][z]
            + (-2*p2*C2+Wb2*Cb2*deltaT)*Tt[i][j][z]
            -2*Wb2*Cb2*deltaT*(Tb3[n][0][LZ3A])
            -2*deltaT*Q2[i][j][z];

        d[i][j][z] = 0;
    }

    return(1);
}
\[-k_2*\Delta t*(T_{i-1,j,k}+T_{i+1,j,k})/\Delta x^2\]
\[+(T_{i,j-1,k}+T_{i,j+1,k})/\Delta y^2\]
\[+(T_{i,j,k-1}+T_{i,j,k+1})/\Delta z^2\])\]
\[-k_2*\Delta t*(T_{i,j,k-1}+T_{i,j,k+1})/\Delta z^2\])\]
\[-2*\Delta t*Q_2[i,j,k];\]
\[d_{i,j,k} = (2*p_2*C_2 + W_b_2*C_2*\Delta t + 4*k_2*\Delta t)/((\Delta x^2 + \Delta y^2)/4) + (2*k_2*\Delta t)/((\Delta z^2)^2)*T_{i,j,k}\]
\[-k_2*\Delta t/(\Delta z^2)*T_{i,j,k-1} + T_{i,j,k+1}\]
\[-\omega * f;\]

\[d_{i,j}[\text{NZ2}]=0;\]

\[// \text{third skin layer}\]
\[\text{for(z=NZ2+1; z<=NZ3-1; z++)}\]
\[{\}
\[f = (2*p_3*C_3 + W_b_3*C_3*\Delta t)*T_{i,j,k}\]
\[-2*W_b_3*C_3*\Delta t)*T_{i,j,k}\]
\[(-2*k_3*\Delta t*(T_{i,j,k-1}+T_{i,j,k+1})/\Delta z^2\])\]
\[-k_3*\Delta t*(T_{i,j,k-1}+T_{i,j,k+1})/\Delta z^2\])\]
\[-2*\Delta t*Q_3[i,j,k];\]
\[d_{i,j}[\text{NZ}]= (2*p_3*C_3 + W_b_3*C_3*\Delta t + 4*k_3*\Delta t)/((\Delta x^2 + \Delta y^2)/4) + (2*k_3*\Delta t)/((\Delta z^2)^2)*T_{i,j,k}\]
\[-k_3*\Delta t/(\Delta z^2)*T_{i,j,k-1} + T_{i,j,k+1}\]
\[-\omega * f;\]

\[}\]

AdjustMtx(0);
AdjustMtx(1);

\[// \text{writeLinearSys(NX/2, NY/2, t, I)};\]

float tE[NZ3], tF[NZ3];

// solve the tria-diagonal system
\[\text{for(i=1; i<=NX-1; i++)}\]
\[\text{for(j=1; j<=NY-1; j++)}\]
{  
    #ifndef CONVECTION
    // Without convection on the surface
    tF[1] = d[i][j][1] / (b[i][j][1] + a[i][j][1]);
    tE[1] = -c[i][j][1] / (b[i][j][1] + a[i][j][1]);
    #else
    // With convection on the surface
    tF[1] = (d[i][j][1] - b[i][j][1] * Tf) / (b[i][j][1] / (k1 + deltaZ * Hf) + a[i][j][1]);
    tE[1] = -c[i][j][1] / (b[i][j][1] / (k1 + deltaZ * Hf) + a[i][j][1]);
    #endif
    
    for(z = 2; z <= NZ3-1; z++)
        {  
            tF[z] = (d[i][j][z] - b[i][j][z] * tF[z-1]) / (a[i][j][z] + b[i][j][z] * tE[z-1]);
            tE[z] = -c[i][j][z] / (a[i][j][z] + b[i][j][z] * tE[z-1]);
        }
    
    //assign tissue boundary grid points
    
    for(i = 0; i <= NX; i++)
        for(j = 0; j <= NY; j++)
            {  
                #ifndef CONVECTION
                // Without convection on the surface
                Tt_n1[i][j][0] = Tt_n1[i][j][1];
                #else
                // With convection on the surface
                Tt_n1[i][j][0] = Tt_n1[i][j][1] * k1 / (k1 + deltaZ * Hf) + Hf * deltaZ / (k1 + deltaZ * Hf) * Tf;
                #endif
                
                if((i >= X5[1] && i <= X6[1] && j >= Y5[1] && j <= Y6[1]) || (i >= X5[0] && i <= X6[0] && j >= Y5[0] && j <= Y6[0]))
                    Tt_n1[i][j][NZ3] = Tt_n1[i][j][NZ3-1];
            }
    
    for(z = 0; z <= NZ3; z++)
        for(j = 0; j <= NY; j++)
            {  
                Tt_n1[0][j][z] = Tt_n1[1][j][z];
                Tt_n1[NX][j][z] = Tt_n1[NX-1][j][z];
            }
    
    for(z = 0; z <= NZ3; z++)
        for(i = 0; i <= NX; i++)
            {  
                Tt_n1[i][0][z] = Tt_n1[i][1][z];
                Tt_n1[i][NY][z] = Tt_n1[i][NY-1][z];
            }
    
    CalcVessel(0);

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CalcVessel();

////////////////////////////////////////
//calculate sum of square error
maxErr = 0;

double temp;

for(i=1;i<NX;i++)
  for(j=1;j<NY;j++)
    for(z=1;z<NZ;z++)
    {
      temp = fabs(Tt_n1[i][j][z] - Tt_nl_I[i][j][z]);
      if(temp > maxErr)
        maxErr = (float)temp;
    }

//store result to loop I
memcpy(Tt_nl_I, Tt_n1, sizeof(float)*(NX+1)*(NY+1)*(NZ+3));

return maxErr;

void Reset()
{
  memset(a, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+3));
  memset(b, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+3));
  memset(c, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+3));
  memset(d, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+3));
  memset(Ql, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+3));
  memset(Q2, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+3));
  memset(Q3, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ+3));

  //ifdef DYN_MEM
  memset(Tb1[0], 0, sizeof(float)*(LZ1A+1));
  memset(Tb2[0], 0, sizeof(float)*(LX2A+1));
  memset(Tb3[0], 0, sizeof(float)*(LZ3A+1)*2);
  memset(Tb1[1], 0, sizeof(float)*(LZ1B+1));
  memset(Tb2[1], 0, sizeof(float)*(LX2B+1));
  memset(Tb3[1], 0, sizeof(float)*(LZ3B+1)*2);

  memset(Tb1n[0], 0, sizeof(float)*(LZ1A+1));
  memset(Tb2n[0], 0, sizeof(float)*(LX2A+1));
  memset(Tb3n[0], 0, sizeof(float)*(LZ3A+1)*2);
  memset(Tb1n[1], 0, sizeof(float)*(LZ1B+1));
  memset(Tb2n[1], 0, sizeof(float)*(LX2B+1));
  memset(Tb3n[1], 0, sizeof(float)*(LZ3B+1)*2);

  memset(Tv1n[0], 0, sizeof(float)*(LZ1A+1));
  memset(Tv2n[0], 0, sizeof(float)*(LX2A+1));

  #ifndef DYN_MEM
  memset(Tb1[0], 0, sizeof(float)*(LZ1A+1));
  memset(Tb2[0], 0, sizeof(float)*(LZ1A+1));
  memset(Tb3[0], 0, sizeof(float)*(LZ1A+1)*2);
  memset(Tb1[1], 0, sizeof(float)*(LZ1B+1));
  memset(Tb2[1], 0, sizeof(float)*(LZ1B+1));
  memset(Tb3[1], 0, sizeof(float)*(LZ1B+1)*2);

  memset(Tb1n[0], 0, sizeof(float)*(LZ1A+1));
  memset(Tb2n[0], 0, sizeof(float)*(LZ1A+1));
  memset(Tb3n[0], 0, sizeof(float)*(LZ1A+1)*2);
  memset(Tb1n[1], 0, sizeof(float)*(LZ1B+1));
  memset(Tb2n[1], 0, sizeof(float)*(LZ1B+1));
  memset(Tb3n[1], 0, sizeof(float)*(LZ1B+1)*2);

  memset(Tv1n[0], 0, sizeof(float)*(LZ1A+1));
  memset(Tv2n[0], 0, sizeof(float)*(LZ1A+1));

  #endif
  //endif DYN_MEM
}
memset(Tv3_n1[0], 0, sizeof(float)*(LZ3A+1)*2);
memset(Tv1_n1[1], 0, sizeof(float)*(LZ1B+1));
memset(Tv2_n1[1], 0, sizeof(float)*(LX2B+1));
memset(Tv3_n1[1], 0, sizeof(float)*(LZ3B+1)*2);
#else
memset(Tb1, 0, sizeof(float)*2*(LZ1A+1));
memset(Tb2, 0, sizeof(float)*2*(LX2A+1));
memset(Tb3, 0, sizeof(float)*2*(LZ3A+1)*2);
#endif
memset(Tv1_n1, 0, sizeof(float)*2*(LZ1A+1));
memset(Tv2_n1, 0, sizeof(float)*2*(LX2A+1));
memset(Tv3_n1, 0, sizeof(float)*2*(LZ3A+1)*2);
memset(Tb1_n1, 0, sizeof(float)*2*(LZ1A+1));
memset(Tb2_n1, 0, sizeof(float)*2*(LX2A+1));
memset(Tb3_n1, 0, sizeof(float)*2*(LZ3A+1)*2);
#endif
Tb1[0][0] = THETA0;
Tb1_n1[0][0] = THETA0;
//float deltaTheta = 0;//Tb1[0] / (LZ1+LX2+LZ3);
return;

// Initialize the laser power
void InitQ(float P0)
{
    int i, j, z;
    float shift;
    for(i=0; i<=NX; i++)
    {
        for(j=0; j<=NY; j++)
        {
            for(z=0; z<=NZ; z++)
            {
                shift = (float)exp( -((i-centerX)*(i-centerX)*deltaX*deltaX
                                     + (j-centerY)*(j-centerY)*deltaY*deltaY) / (2*Sigma*Sigma) )
                         / ( (float)sqrt(2*pi)*Sigma);
                Q1[i][j][z] = Alpha1 * (float)exp( -Alpha1*z*deltaZ ) * shift * P0
                                      *(1.0f-Ref1);
                Q2[i][j][z] = Alpha2 * (float)exp( -Alpha2*(z-NZ1)*deltaZ
                                      -Alpha1*deltaZ*NZ1) * shift * P0*(1-Ref2);
                Q3[i][j][z] = Alpha3 * (float)exp( -Alpha3*(z-NZ2)*deltaZ
                                      -Alpha1*deltaZ*NZ1 -Alpha2*deltaZ*(NZ2-NZ1)) * shift * P0
                                      *(1-Ref3);
            }
        }
    }
}
float CalcNewP(float *Tm_pre, float *Tm0, float *Tm1, float P0, float deltaP)
{
    int i;
    float P1_new;
    float alphaStar = 0;

    float X[M];
    for(i=0;i<M;i++)
        X[i] = (Tm1[i] - Tm0[i]) / deltaP;

    float factor = 0;
    for(i=0;i<M;i++)
        factor += X[i]*X[i];
    factor += alphaStar;

    float temp=0;
    for(i=0;i<M;i++)
        temp += X[i] * (Tm_pre[i]-Tm1[i]);

    P1_new = P0 + 1/factor * temp;
    return P1_new;
}

int getTm(float *Tm)
{
    //assign surface temperature to Tm array here
    //for M==5 right now
    if(bReachTop)
        Tm[0] = Tm_pre[0];
    else
        Tm[0] = Tt[NX/2][NY/2][0];
    Tm[1] = Tt[NX/2][0][0];
    Tm[2] = Tt[NX/2][NY][0];
    Tm[3] = Tt[0][NY/2][0];
    Tm[4] = Tt[NX][NY/2][0];
    return (1);
}

void AdjustPower(float P0)
{
    if(bPowerOn)
    {
        //power is on now
        if(Tt_n1[NX/2][NY/2][0] >= Tm_pre[0])
        {
            //set power off
            memset(Q1, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ3+1));
            memset(Q2, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ3+1));
            memset(Q3, 0, sizeof(float)*(NX+1)*(NY+1)*(NZ3+1));
            bPowerOn = false;
            bReachTop = true;
            //writeLog("========Power Off========");
        }
    }

    return;
}
} 
} 
else 
{

//power is off now
if(Tt_n1[NX/2][NY/2][0] <= Tm_pre[0]-4) 
{

//set power on
InitQ(P0);
bPowerOn = true;
sprintf(tmp, "======Power On:%7.4lf", P0);
//writeLog(tmp);
}
}

return;

}

void AdjustMtx(int index)
{

int x1 = X1[index], x2 = X2[index], x3 = X3[index], x4 = X4[index], x5 = X5[index],
x6 = X6[index];
int y1 = Y1[index], y2 = Y2[index], y3 = Y3[index], y4 = Y4[index], y5 = Y5[index],
y6 = Y6[index];
int z1 = Z1[index], z2 = Z2[index], z3 = Z3[index];
int lz1 = index==0?LZ1A:LZ1B;
int lz3 = index==0?LZ3A:LZ3B;

//reassign back the grid points in blood
//level 3
for(z=z1;z<=z2;z++)
{
    for(j=y1+1;j<y2;j++)
    {
        //left branch
        for(i=x1+1;i<x2;i++)
            Tt_n1[i][j][z] = Tb3_n1[index][z2-z];

        //right branch
        for(i=x3+1;i<x4;i++)
            Tt_n1[i][j][z] = Tb3_n1[index][z2-z+lz3+1];
    }
}

//level 2
for(z=z2+1;z<=z3;z++)
{
    for(j=y3+1;j<y4;j++)
        for(i=x1+1;i<x4;i++)
            Tt_n1[i][j][z] = Tb2_n1[index][i-x1];
}

//level 1
for(j=y3+1;j<y4;j++)
    for(i=x5+1;i<x6;i++)
        Tt_n1[i][j][z3] = Tb1_n1[index][lz1]; //interface between level 2 & 3
for(z=z3+1;z<=NZ3;z++)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
for(j=y5+1;j<y6;j++)
for(i=x5+1;i<x6;i++)
    Tt_n1[i][j][z] = Tb1_n1[index][NZ3-z];

//adjust the matrix for blood vessel
//blood level 1
for(i=x5;i<=x6;i++)
    for(j=y5;j<=y6;j++)
    {
        if(i==x5 || i==x6 || j==y5 || j==y6)
        {
            //side walls
            d[i][j][z3-1] -= c[i][j][z3-1]*Tt_n1_I[i][j][z3];
            c[i][j][z3-1] = 0;
            for(z=z3;z<NZ3;z++)
            {
                b[i][j][z] = 0;
                a[i][j][z] = 1;
                c[i][j][z] = 0;
                d[i][j][z] = Tt_n1_I[i][j][z];
            }
        }
        else
        {
            //center (the interface part with level 2 will be fixed below on level 2)
            b[i][j][z3] = -1;
            a[i][j][z3] = 1+B*i*deltaZ;
            c[i][j][z3] = 0;
            d[i][j][z3] = B*i*deltaZ*Tt_n1[i][j][z3+1];
            for(z=z3+1;z<NZ3;z++)
            {
                b[i][j][z] = 0;
                a[i][j][z] = 1;
                c[i][j][z] = 0;
                d[i][j][z] = Tt_n1_I[i][j][z];
            }
        }
    }

//blood level 2
for(i=x1;i<=x4;i++)
    for(j=y3;j<=y4;j++)
    {
        if(i==x1 || i==x4 || j==y3 || j==y4)
        {
            //side walls, use the previous step temperature (Loop 1)
        }
        //top part on the wall
        d[i][j][z2-1] -= c[i][j][z2-1]*Tt_n1_I[i][j][z2];
        c[i][j][z2-1] = 0;
        for(z=z2;z<NZ3;z++)
        {
            b[i][j][z] = 0;
            a[i][j][z] = 1;
        }
    }

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
c[i][j][z] = 0;
d[i][j][z] = T_{n1}[i][j][z];

//bottom part on the wall
d[i][j][z3+1] -= b[i][j][z3+1]*T_{n1}[i][j][z3];
b[i][j][z3+1] = 0;
}
else
{
    //center, use temperature at loop 1+1,(already updated by blood temperatures)

    //top wall (the interface part with level 3 will be fixed below on level 3)
    b[i][j][z2] = -1;
a[i][j][z2] = 1+B_{i}\delta Z;
c[i][j][z2] = 0;
d[i][j][z2] = B_{i}\delta Z*T_{n1}[i][j][z2+1];
    //between
    for(z=z2+1; z<z3; z++)
    {
        b[i][j][z] = 0;
a[i][j][z] = 1;
c[i][j][z] = 0;
d[i][j][z] = T_{n1}[i][j][z];
    }
    //bottom wall
    b[i][j][z3] = 0;
a[i][j][z3] = 1+B_{i}\delta Z;
c[i][j][z3] = -1;
d[i][j][z3] = B_{i}\delta Z*T_{n1}[i][j][z3-1];
}
if(i>=x5 && i<=x6 && j>=y3 && j<=y4)
{
    //the joint between level 1 and level 2
    //bottom part fix
    b[i][j][z3] = 0;
a[i][j][z3] = 1;
c[i][j][z3] = 0;
    if(i=x5 || i=x6 || j=y3 || j=y4)
        d[i][j][z3] = T_{n1}[i][j][z3]; //on the joint, use the previous temperature (Loop 1)
    else
        d[i][j][z3] = T_{n1}[i][j][z3]; //inside blood, use blood temperature
}
//blood level 3 left branch
for(i=x1; i<=x2; i++)
    for(j=y1; j<=y2; j++)
    {
        if(i==x1 || i==x2 || j==y1 || j==y2)
        {
            //side walls, use the previous step temperature (Loop 1)
            //top part on the wall
            d[i][j][z1-1] -= c[i][j][z1-1]*T_{n1}[i][j][z1];
c[i][j][z1-1] = 0;

        }

}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
for(z=z1; z<=z2; z++)
{
    b[i][j][z] = 0;
    a[i][j][z] = 1;
    c[i][j][z] = 0;
    d[i][j][z] = Tt_n1[i][j][z];
}
else
{
    //center, use temperature at loop 1+1, (already updated by blood temperatures)

    for(z=z1; z<=z2; z++)
    {
        b[i][j][z] = 0;
        a[i][j][z] = 1;
        c[i][j][z] = 0;
        d[i][j][z] = Tt_n1[i][j][z];
    }
}

//blood level 3 right branch
for(i=x3; i<=x4; i++)
    for(j=y1; j<=y2; j++)
    {
        if(i==x3 || i==x4 || j==y1 || j==y2)
        {
            //side walls, use the previous step temperature (Loop 1)

            //top part on the wall
            d[i][j][z1-1] -= c[i][j][z1-1]*Tt_n1[i][j][z1];
            c[i][j][z1-1] = 0;
            for(z=z1; z<=z2; z++)
            {
                b[i][j][z] = 0;
                a[i][j][z] = 1;
                c[i][j][z] = 0;
                d[i][j][z] = Tt_n1[i][j][z];
            }
        }
        else
        {
            //center, use temperature at loop 1+1, (already updated by blood temperatures)

            for(z=z1; z<=z2; z++)
            {
                b[i][j][z] = 0;
                a[i][j][z] = 1;
                c[i][j][z] = 0;
                d[i][j][z] = Tt_n1[i][j][z];
            }
        }
    }
return;
void CalcVessel(int index)
{
    int x1 = X1[index], x2 = X2[index], x3 = X3[index], x4 = X4[index], x5 = X5[index],
    x6 = X6[index];
    int y1 = Y1[index], y2 = Y2[index], y3 = Y3[index], y4 = Y4[index], y5 = Y5[index],
    y6 = Y6[index];
    int z1 = Z1[index], z2 = Z2[index], z3 = Z3[index];
    int lx2 = index==0?LX2A:LX2B;
    int lx3 = index==0?LX3A:LX3B;

    //////////////////////////////////////////
    //calculate blood vessel temperature on sides (grid points on XY plane already calculated)
    //level 3
    for(z=z1;z<=z2;z++)
    {
        //left branch
        for(j=y1;j<=y2;j++)
        {
            Tt_n1[x1][j][z] = ( Tt_n1[x1-1][j][z] + Tb3_n1[index][z2-z] * deltaX*Bi )
            / (1+deltaX*Bi);
            Tt_n1[x2][j][z] = ( Tt_n1[x2+1][j][z] + Tb3_n1[index][z2-z] * deltaX*Bi )
            / (1+deltaX*Bi);
        }
        for(i=x1;i<=x2;i++)
        {
            Tt_n1[i][y1][z] = ( Tt_n1[i][y1-1][z] + Tb3_n1[index][z2-z] * deltaY*Bi ) / (1+deltaY*Bi);
            Tt_n1[i][y2][z] = ( Tt_n1[i][y2+1][z] + Tb3_n1[index][z2-z] * deltaY*Bi ) / (1+deltaY*Bi);
        }

        //right branch
        for(j=y1;j<=y2;j++)
        {
            Tt_n1[x3][j][z] = ( Tt_n1[x3-1][j][z] + Tb3_n1[index][z2-z+lx3+1] * deltaX*Bi )
            / (1+deltaX*Bi);
            Tt_n1[x4][j][z] = ( Tt_n1[x4+1][j][z] + Tb3_n1[index][z2-z+lx3+1] * deltaX*Bi )
            / (1+deltaX*Bi);
        }
        for(i=x3;i<=x4;i++)
        {
            Tt_n1[i][y1][z] = ( Tt_n1[i][y1-1][z] + Tb3_n1[index][z2-z+lx3+1] * deltaY*Bi )
            / (1+deltaY*Bi);
            Tt_n1[i][y2][z] = ( Tt_n1[i][y2+1][z] + Tb3_n1[index][z2-z+lx3+1] * deltaY*Bi )
            / (1+deltaY*Bi);
        }
    }

    //level 2
    for(z=z2;z<=z3;z++)
    {
        for(i=x1;i<=x4;i++)
        {
            Tt_n1[i][y3][z] = ( Tt_n1[i][y3-1][z] + Tb2_n1[index][i-x1] * deltaY*Bi )
                / (1+deltaY*Bi);
        }
    }
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[
Tt_{n1}[i][y4][z] = \frac{( \ Tt_{n1}[i][y4+1][z] + Tt_{2.n1}[index][i-x1] * \delta Y*B_i )}{(1+\delta Y*B_i)};
\]

//left most and right most sides
for(j=y3;j<=y4;j++)
{
    \[
    Tt_{n1}[x1][j][z] = \frac{( \ Tt_{n1}[x1-1][j][z] + Tt_{2.n1}[index][1] * \delta X*B_i )}{(1+\delta X*B_i)};
    \]
    \[
    Tt_{n1}[x4][j][z] = \frac{( \ Tt_{n1}[x4+1][j][z] + Tt_{2.n1}[index][hx2-1] * \delta X*B_i )}{(1+\delta X*B_i)};
    \]
}

//level 1
for(z=z3; z<=NZ3; z++)
{
    for(j=y5; j<=y6; j++)
    { 
        Tt_{n1}[x5][j][z] = \frac{( \ Tt_{n1}[x5-1][j][z] + Tt_{1.n1}[index][NZ3-z] * \delta X*B_i )}{(1+\delta X*B_i)};
        Tt_{n1}[x6][j][z] = \frac{( \ Tt_{n1}[x6+1][j][z] + Tt_{1.n1}[index][NZ3-z] * \delta X*B_i )}{(1+\delta X*B_i)};
    }
    for(i=x5; i<=x6; i++)
    { 
        Tt_{n1}[i][y5][z] = \frac{( \ Tt_{n1}[i][y5-1][z] + Tt_{1.n1}[index][NZ3-z] * \delta Y*B_i )}{(1+\delta Y*B_i)};
        Tt_{n1}[i][y6][z] = \frac{( \ Tt_{n1}[i][y6+1][z] + Tt_{1.n1}[index][NZ3-z] * \delta Y*B_i )}{(1+\delta Y*B_i)};
    }
}

//===handle the edges of blood vessels===
//use the average of the two neighbor points on the vessel
//===level 3

//joints between blood level 2 & 3
for(j=y1; j<=y2; j++)
{
    Tt_{n1}[x2][j][z2] = \frac{1}{2} \left( Tt_{n1}[x2+1][j][z2] + Tt_{n1}[x2][j][z2-1] \right);
    Tt_{n1}[x3][j][z2] = \frac{1}{2} \left( Tt_{n1}[x3-1][j][z2] + Tt_{n1}[x3][j][z2+1] \right);
}

for(i=x1; i<=x2; i++)
{
    Tt_{n1}[i][y1][z2] = \frac{1}{2} \left( Tt_{n1}[i][y1-1][z2] + Tt_{n1}[i][y1][z2-1] \right);
    Tt_{n1}[i][y2][z2] = \frac{1}{2} \left( Tt_{n1}[i][y2+1][z2] + Tt_{n1}[i][y2][z2+1] \right);
}

for(i=x3; i<=x4; i++)
{
    Tt_{n1}[i][y1][z2] = \frac{1}{2} \left( Tt_{n1}[i][y1-1][z2] + Tt_{n1}[i][y1][z2-1] \right);
    Tt_{n1}[i][y2][z2] = \frac{1}{2} \left( Tt_{n1}[i][y2+1][z2] + Tt_{n1}[i][y2][z2+1] \right);
}

//side edges
for(z=z1; z<=z2; z++)
{

\[ T_{t_n l}[x_1][y_1][z] = \left( \frac{T_{t_n l}[x_1+1][y_1][z] + T_{t_n l}[x_1][y_1+1][z]}{2} \right); \]
\[ T_{t_n l}[x_2][y_1][z] = \left( \frac{T_{t_n l}[x_2-1][y_1][z] + T_{t_n l}[x_2][y_1+1][z]}{2} \right); \]
\[ T_{t_n l}[x_3][y_1][z] = \left( \frac{T_{t_n l}[x_3+1][y_1][z] + T_{t_n l}[x_3][y_1+1][z]}{2} \right); \]
\[ T_{t_n l}[x_4][y_1][z] = \left( \frac{T_{t_n l}[x_4+1][y_1][z] + T_{t_n l}[x_4][y_1+1][z]}{2} \right); \]
\[ T_{t_n l}[x_1][y_2][z] = \left( \frac{T_{t_n l}[x_1+1][y_2][z] + T_{t_n l}[x_1][y_2+1][z]}{2} \right); \]
\[ T_{t_n l}[x_2][y_2][z] = \left( \frac{T_{t_n l}[x_2-1][y_2][z] + T_{t_n l}[x_2][y_2+1][z]}{2} \right); \]
\[ T_{t_n l}[x_3][y_2][z] = \left( \frac{T_{t_n l}[x_3+1][y_2][z] + T_{t_n l}[x_3][y_2+1][z]}{2} \right); \]
\[ T_{t_n l}[x_4][y_2][z] = \left( \frac{T_{t_n l}[x_4+1][y_2][z] + T_{t_n l}[x_4][y_2+1][z]}{2} \right); \]

}  //==level 2

//top surface
for(i=x_1;i<=x_4;i++) { 
\[ T_{t_n l}[i][y_3][z2] = \left( \frac{T_{t_n l}[i][y_3][z2+1] + T_{t_n l}[i][y_3+1][z2]}{2} \right); \]
\[ T_{t_n l}[i][y_4][z2] = \left( \frac{T_{t_n l}[i][y_4][z2+1] + T_{t_n l}[i][y_4-1][z2]}{2} \right); \]
}

for(j=y_3;j<y_1;j++) { 
\[ T_{t_n l}[x_1][j][z2] = \left( \frac{T_{t_n l}[x_1][j][z2+1] + T_{t_n l}[x_1+1][j][z2]}{2} \right); \]
\[ T_{t_n l}[x_4][j][z2] = \left( \frac{T_{t_n l}[x_4][j][z2+1] + T_{t_n l}[x_4-1][j][z2]}{2} \right); \]
}

for(i=x_5;i<=x_6;i++) //joints of blood level 1 & 2
{ 
\[ T_{t_n l}[i][y_3][z3] = \left( \frac{T_{t_n l}[i][y_3][z3+1] + T_{t_n l}[i][y_3+1][z3]}{2} \right); \]
\[ T_{t_n l}[i][y_4][z3] = \left( \frac{T_{t_n l}[i][y_4][z3+1] + T_{t_n l}[i][y_4-1][z3]}{2} \right); \]
}

for(j=y_3;j<y_4;j++) { 
\[ T_{t_n l}[x_1][j][z3] = \left( \frac{T_{t_n l}[x_1][j][z3+1] + T_{t_n l}[x_1+1][j][z3]}{2} \right); \]
\[ T_{t_n l}[x_4][j][z3] = \left( \frac{T_{t_n l}[x_4][j][z3+1] + T_{t_n l}[x_4-1][j][z3]}{2} \right); \]
}

for(i=x_5;i<=x_6;i++) //joints
{ 
\[ T_{t_n l}[i][y_3][z3] = \left( \frac{T_{t_n l}[i][y_3-1][z3] + T_{t_n l}[i][y_3][z3-1]}{2} \right); \]
\[ T_{t_n l}[i][y_4][z3] = \left( \frac{T_{t_n l}[i][y_4-1][z3] + T_{t_n l}[i][y_4][z3-1]}{2} \right); \]
}

//bottom surface
for(j=y_3;j<y_4;j++) { 
\[ T_{t_n l}[x_1][j][z3] = \left( \frac{T_{t_n l}[x_1][j][z3+1] + T_{t_n l}[x_1+1][j][z3]}{2} \right); \]
\[ T_{t_n l}[x_4][j][z3] = \left( \frac{T_{t_n l}[x_4][j][z3+1] + T_{t_n l}[x_4-1][j][z3]}{2} \right); \]
}

//side vertical edges
for(z=z_2;z<z_3;z++) { 
\[ T_{t_n l}[x_1][y_3][z] = \left( \frac{T_{t_n l}[x_1+1][y_3][z] + T_{t_n l}[x_1][y_3+1][z]}{2} \right); \]
\[ T_{t_n l}[x_4][y_3][z] = \left( \frac{T_{t_n l}[x_4-1][y_3][z] + T_{t_n l}[x_4][y_3+1][z]}{2} \right); \]
\[ T_{t_n l}[x_1][y_4][z] = \left( \frac{T_{t_n l}[x_1+1][y_4][z] + T_{t_n l}[x_1][y_4+1][z]}{2} \right); \]
\[ T_{t_n l}[x_4][y_4][z] = \left( \frac{T_{t_n l}[x_4-1][y_4][z] + T_{t_n l}[x_4][y_4+1][z]}{2} \right); \]
}

//vertix

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[
T_{t_n[l][x1][y3][z2]} = \frac{(T_{t_n[l][x1+1][y3][z2]} + T_{t_n[l][x1][y3+1][z2]} + T_{t_n[l][x1][y3][z2+1]})}{3};
\]
\[
T_{t_n[l][x1][y3][z3]} = \frac{(T_{t_n[l][x1+1][y3][z3]} + T_{t_n[l][x1][y3+1][z3]} + T_{t_n[l][x1][y3][z3+1]})}{3};
\]
\[
T_{t_n[l][x1][y4][z2]} = \frac{(T_{t_n[l][x1+1][y4][z2]} + T_{t_n[l][x1][y4-1][z2]} + T_{t_n[l][x1][y4][z2+1]})}{3};
\]
\[
T_{t_n[l][x1][y4][z3]} = \frac{(T_{t_n[l][x1+1][y4][z3]} + T_{t_n[l][x1][y4-1][z3]} + T_{t_n[l][x1][y4][z3+1]})}{3};
\]
\[
T_{t_n[l][x4][y3][z2]} = \frac{(T_{t_n[l][x4-1][y3][z2]} + T_{t_n[l][x4][y3+1][z2]} + T_{t_n[l][x4][y3][z2+1]})}{3};
\]
\[
T_{t_n[l][x4][y3][z3]} = \frac{(T_{t_n[l][x4-1][y3][z3]} + T_{t_n[l][x4][y3-1][z3]} + T_{t_n[l][x4][y3][z3+1]})}{3};
\]
\[
T_{t_n[l][x4][y4][z2]} = \frac{(T_{t_n[l][x4-1][y4][z2]} + T_{t_n[l][x4][y4-1][z2]} + T_{t_n[l][x4][y4][z2+1]})}{3};
\]
\[
T_{t_n[l][x4][y4][z3]} = \frac{(T_{t_n[l][x4-1][y4][z3]} + T_{t_n[l][x4][y4-1][z3]} + T_{t_n[l][x4][y4][z3+1]})}{3};
\]

---

For \(i=x_5; i<=x_6; i++\)

\[
T_{t_n[l][i][y5][z3]} = \frac{(T_{t_n[l][i][y5+1][z3]} + T_{t_n[l][i][y5-1][z3]})}{2};
\]
\[
T_{t_n[l][i][y6][z3]} = \frac{(T_{t_n[l][i][y6+1][z3]} + T_{t_n[l][i][y6-1][z3]})}{2};
\]

For \(j=y_5; j<=y_3; j++\)

\[
T_{t_n[l][x5][j][z3]} = \frac{(T_{t_n[l][x5+1][j][z3]} + T_{t_n[l][x5-1][j][z3]})}{2};
\]
\[
T_{t_n[l][x6][j][z3]} = \frac{(T_{t_n[l][x6+1][j][z3]} + T_{t_n[l][x6-1][j][z3]})}{2};
\]

For \(j=y_4; j<=y_6; j++\)

\[
T_{t_n[l][x5][j][z3]} = \frac{(T_{t_n[l][x5+1][j][z3]} + T_{t_n[l][x5-1][j][z3]})}{2};
\]
\[
T_{t_n[l][x6][j][z3]} = \frac{(T_{t_n[l][x6+1][j][z3]} + T_{t_n[l][x6-1][j][z3]})}{2};
\]

---

For \(z=z_3; z<=NZ_3; z++\)

\[
T_{t_n[l][x5][y5][z]} = \frac{(T_{t_n[l][x5+1][y5][z]} + T_{t_n[l][x5-1][y5][z]})}{2};
\]
\[
T_{t_n[l][x6][y5][z]} = \frac{(T_{t_n[l][x6+1][y5][z]} + T_{t_n[l][x6-1][y5][z]})}{2};
\]
\[
T_{t_n[l][x5][y6][z]} = \frac{(T_{t_n[l][x5+1][y6][z]} + T_{t_n[l][x5-1][y6][z]})}{2};
\]
\[
T_{t_n[l][x6][y6][z]} = \frac{(T_{t_n[l][x6+1][y6][z]} + T_{t_n[l][x6-1][y6][z]})}{2};
\]

---

**//vertex**

\[
T_{t_n[l][x5][y3][z3]} = \frac{(T_{t_n[l][x5+1][y3][z3]} + T_{t_n[l][x5-1][y3][z3]} + T_{t_n[l][x5][y3-1][z3]} + T_{t_n[l][x5][y3+1][z3]})}{4};
\]
\[
T_{t_n[l][x5][y4][z3]} = \frac{(T_{t_n[l][x5+1][y4][z3]} + T_{t_n[l][x5-1][y4][z3]} + T_{t_n[l][x5][y4-1][z3]} + T_{t_n[l][x5][y4+1][z3]})}{4};
\]
\[
T_{t_n[l][x6][y3][z3]} = \frac{(T_{t_n[l][x6+1][y3][z3]} + T_{t_n[l][x6-1][y3][z3]} + T_{t_n[l][x6][y3-1][z3]} + T_{t_n[l][x6][y3+1][z3]})}{4};
\]
\[
T_{t_n[l][x6][y4][z3]} = \frac{(T_{t_n[l][x6+1][y4][z3]} + T_{t_n[l][x6-1][y4][z3]} + T_{t_n[l][x6][y4-1][z3]} + T_{t_n[l][x6][y4+1][z3]})}{4};
\]

---

For \(i=x_1; i<=x_2; i++\)

\[
T_{t_n[l][x1][y1][z2]} = \frac{(T_{t_n[l][x1+1][y1][z2]} + T_{t_n[l][x1][y1-1][z2]} + T_{t_n[l][x1][y1+1][z2]})}{3};
\]
\[
+ \frac{T_{t_{n1}[x1][y1][z2-1]}}{4};
T_{t_{n1}[x1][y2][z2]} = \left( \frac{T_{t_{n1}[x1+1][y2][z2]} + T_{t_{n1}[x1-1][y2][z2]} + T_{t_{n1}[x1][y2-1][z2]} + T_{t_{n1}[x1][y2+1][z2]}}{4} \right);
\]

\[
T_{t_{n1}[x2][y1][z2]} = \left( \frac{T_{t_{n1}[x2-1][y1][z2]} + T_{t_{n1}[x2][y1+1][z2]} + T_{t_{n1}[x2][y1-1][z2]} + T_{t_{n1}[x2][y1][z2-1]}}{3} \right);
\]

\[
T_{t_{n1}[x2][y2][z2]} = \left( \frac{T_{t_{n1}[x2-1][y2][z2]} + T_{t_{n1}[x2][y2+1][z2]} + T_{t_{n1}[x2][y2-1][z2]} + T_{t_{n1}[x2][y2][z2-1]}}{3} \right);
\]

\[
T_{t_{n1}[x3][y1][z2]} = \left( \frac{T_{t_{n1}[x3+1][y1][z2]} + T_{t_{n1}[x3][y1+1][z2]} + T_{t_{n1}[x3][y1-1][z2]} + T_{t_{n1}[x3][y1][z2-1]}}{3} \right);
\]

\[
T_{t_{n1}[x3][y2][z2]} = \left( \frac{T_{t_{n1}[x3+1][y2][z2]} + T_{t_{n1}[x3][y2+1][z2]} + T_{t_{n1}[x3][y2-1][z2]} + T_{t_{n1}[x3][y2][z2-1]}}{3} \right);
\]

\[
T_{t_{n1}[x4][y1][z2]} = \left( \frac{T_{t_{n1}[x4-1][y1][z2]} + T_{t_{n1}[x4][y1+1][z2]} + T_{t_{n1}[x4][y1-1][z2]} + T_{t_{n1}[x4][y1][z2-1]}}{3} \right);
\]

\[
T_{t_{n1}[x4][y2][z2]} = \left( \frac{T_{t_{n1}[x4-1][y2][z2]} + T_{t_{n1}[x4][y2+1][z2]} + T_{t_{n1}[x4][y2-1][z2]} + T_{t_{n1}[x4][y2][z2-1]}}{4} \right);
\]

return;

void writeSquareXY(int x0, int y0, int x1, int y1, int z, int t, int I)
{
    if(!PrintOut(t))
        return;

    FILE *file;
    char fname[256], line[8196];

    strcpy(fname, outPath);
    sprintf(tmp, "%d_Z%d_%d_%d_%d_%d_t%d_I%d.txt", loopP, z, x0, y0, x1, y1, t, I);
    strcat(fname, tmp);
    file = fopen(fname, "w");

    for(int j=y0; j<=y1; j++)
    {
        strcpy(line, "\n");
        for(int i=x0; i<=x1; i++)
        {
            sprintf(tmp, "%7.4lf", T_{t_{n1}[i][j][z]});
            strcat(line, tmp);
        }
        strcat(line, "\n");
        fwrite(line,strlen(line), 1, file);
    }
    fclose(file);
}

void writeSquareXZ(int x0, int z0, int x1, int z1, int y, int t, int I)
{
    if(!PrintOut(t))
        return;

    FILE *file;
    char fname[256], line[8196];

    strcpy(fname, outPath);
    sprintf(tmp, "%d_Y%d_%d_%d_%d_%d_t%d_I%d.txt", loopP, y, x0, z0, x1, z1, t, I);

    for(int i=x0; i<=x1; i++)
    {
        sprintf(tmp, "%7.4lf", T_{t_{n1}[i][j][z]});
        strcat(line, tmp);
    }
    strcat(line, "\n");
    fwrite(line,strlen(line), 1, file);
    fclose(file);
}
`void writeSquareYZ(int y0, int z0, int y1, int z1, int x, int t, int I) {
    if(!PrintOut(t))
        return;

    FILE *file;
    char fname[256], line[8196];

    strcpy(fname, outPath);
    sprintf(tmp, "%d_X%d_%d_%d_%d_%d_t%d_I%d.txt", loopP, x, y0, z0, y1, z1, t, I);
    strcat(fname, tmp);

    file = fopen(fname, "w");

    for(int k=z0;k<=z1;k++)
        {
            strcpy(line, "");
            for(int j=y0;j<=y1;j++)
                {
                    sprintf(tmp, "%7.41f ", Tt_n1[i][j][k]);
                    strcat(line, tmp);
                }
            strcat(line, "n");
            fwrite(line,strlen(line), 1, file);
        }
    fclose(file);
}

void writeZCenter(int t) {
    if(!PrintOut(t))
        return;
FILE *file;
char fname[256], line[8196];

strcpy(fname, outPath);
sprintf(tmp, "%d_Z_Center_t%d.txt", loopP, t);
strcat(fname, tmp);
file = fopen(fname, "w");

for(int j=0;j<=NZ3;j++)
{
    sprintf(tmp, "%7.41f ", Tt_n1[centerX][centerY][j]);
    strcat(line, tmp);
    strcat(line, "n");
    fwrite(line, strlen(line), 1, file);
}
fclose(file);
}

void writeAll(int t, int I)
{
    if(!PrintOut(t))
        return;

    FILE *file;
char fname[256];

strcpy(fname, outPath);
sprintf(tmp, "%d_t%d_I%d.dat", loopP, t, I);
strcat(fname, tmp);
file = fopen(fname, "wb");
for(int i=0;i<NX+1;i++)
    fwrite(Tt_n1[i],sizeof(float)*(NY+1)*(NZ3+1), 1, file);
fclose(file);
return;
}

void writeLinearSys(int x, int y, int t, int I)
{
    FILE *file;
char fname[256], line[256];

strcpy(fname, outPath);
sprintf(tmp, "%d_X%d_Y%d_t%d_I%d.txt", loopP, x, y, t, I);
strcat(fname, tmp);
file = fopen(fname, "w");
for(int k=0;k<=NZ3;k++)
{
void writeQ(float P, float sigma) {
    FILE *file;
    char fname[256], line[20000];
    strcpy(fname, outPath);
    sprintf(tmp, "sigma_%6.4lf_p_%6.4lf.txt", sigma, P);
    strcat(fname, tmp);
    file = fopen(fname, "w");
    int k;
    for(k=0;k<=NZ1;k++) {
        strcpy(line, "");
        for(int i=0;i<=NX;i++)
            { 
            sprintf(tmp, "%7.4lf", Q1[i][NY/2][k]);    
            strcat(line, tmp);
            }
        strcat(line, "n");
        fwrite(line,strlen(line), 1, file);
    }
    for(k=NZ1+1;k<=NZ2;k++) {
        strcpy(line, "");
        for(int i=0;i<=NX;i++)
            { 
            sprintf(tmp, "%7.4lf", Q2[i][NY/2][k]);    
            strcat(line, tmp);
            }
        strcat(line, "n");
        fwrite(line,strlen(line), 1, file);
    }
    for(k=NZ2+1;k<=NZ3;k++) {
        strcpy(line, "");
        for(int i=0;i<=NX;i++)
            { 
            sprintf(tmp, "%7.4lf", Q3[i][NY/2][k]);    
            strcat(line, tmp);
            }
        strcat(line, "n");
        fwrite(line,strlen(line), 1, file);
    }
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
fclose(file);
}

void writeLog(char *ln)
{
    FILE *file;
    char fname[256], line[512];
    sprintf(fname, "%slog%d.txt", outPath, loopP);
    //strcpy(fnam e, outPath);
    //strcat(fname, "log.txt");
    file = fopen(fname, "a");
    strcpy(line, ln);
    strcat(line, "n");
    strcat(line, "n");
    fwrite(line, strlen(line), 1, file);
    fclose(file);
}

bool PrintOut(int tm)
{
    bool out = false;
    if(outT[0] == -1)
        return true;
    for(int i=0;i<numT;i++)
        if(outT[i] == tm*deltaT)
            { 
                out = true;
                break;
            }
    return out;
}
APPENDIX C

NOMENCLATURE
$B_i$ Biot number
$C_i$ specific heat of tissue in layer 1
$C_{bi}$ specific heat of blood in layer 1
$C_b$ heat capacity of blood
$F_m$ area of cross-section in the $m$th level vessel
$M_m$ mass flow of blood in the $m$th level vessel
$I$ iterative index
$h$ heat convection coefficient between skin surface and air
$I$ identity matrix
$i,j,k$ indices of grid point
$k_i$ heat conductivity of tissue in layer 1
$I_{pre}$ pre-conditioned Richardson operator
$L_i$ thickness of layer 1
$L_{b_m}$ length of the blood vessel in level $m$
$M$ number of pre-specified elevated temperatures
$l$ layer of skin
$m$ level of blood vessel
$N_x, N_y, N_z$ numbers of grid points in the $x, y, z$ directions, respectively
$NX, NY$ lengths of the skin structure in the $x, y$ directions, respectively
$NL_{b_m}, NW_{b_m}$ length and width of the cross-section of the $m$th level vessel
$n$ time level
$P_0$ laser intensity
$P$ vessel periphery
$\dot{P}$ blood flow rate
$Q_i$ heat source in layer 1
Reff$_l$ laser reflectivity in layer 1
$S$ least squares sum
$t$ time
$\theta_{pre}$ pre-specified temperature elevations
$u_{cal}$ calculated temperature
$u_{ik}^a$ numerical solution of temperature elevation of tissue
$u_{bi}^a$ numerical solution of temperature elevation of blood in the $m$th level vessel
$\nu_m$ velocity of blood flow in the $m$th level vessel
$W_{b_i}$ blood perfusion rate in layer 1
$x, y, z$ cartesian coordinates
$\alpha$ heat transfer coefficient between vessel and tissue
$\alpha_i$ laser absorptivity of layer 1
$\delta_x^2, \delta_y^2, \delta_z^2$  
second order finite difference operators

$\rho_i$  
density of layer $i$

$\omega$  
relaxation parameter

$\sigma$  
standard deviation of laser beam width

$\theta_b^m, \theta_i^m, \theta_w^m$  
temperature elevations in blood, tissue, and vessel wall, respectively

$\Delta x, \Delta y, \Delta z$  
mesh sizes in the $x, y, z$ directions, respectively

$\Delta t$  
time increment

$\theta_{in}, \theta_{out}$  
temperature elevations of blood at entrance and exit, respectively
REFERENCES


