Spring 2005

A numerical method for obtaining an optimal temperature distribution in a 3D triple-layered cylindrical skin structure

Le Zhang

Follow this and additional works at: https://digitalcommons.latech.edu/dissertations
Part of the Biomedical Engineering and Bioengineering Commons, and the Mathematics Commons
NOTE TO USERS

This reproduction is the best copy available.

UMI®
A NUMERICAL METHOD FOR OBTAINING AN OPTIMAL
TEMPERATURE DISTRIBUTION IN A 3D TRIPLE-LAYERED
CYLINDRICAL SKIN STRUCTURE

by
Le Zhang, M.S.

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

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2005
We hereby recommend that the dissertation prepared under our supervision by Le Zhang entitled A NUMERICAL METHOD FOR OBTAINING AN OPTIMAL TEMPERATURE DISTRIBUTION IN A 3D TRIPLE-LAYERED CYLINDRICAL SKIN STRUCTURE be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computational Analysis and Modeling.

Supervisor of Dissertation Research

Head of Department

Department

Advisory Committee

Approved:

Director of Graduate Studies

Dean of the College

Approved:

Dean of the Graduate School

GS Form 13
(5/03)
ABSTRACT

In recent years, it has been interesting to research hyperthermia combined with radiation and cytotoxic drugs to enhance the killing of tumors. The crucial problem is that when heating the tumor tissues, one needs to keep the surrounding normal tissue below a temperature that will produce harm. Thus, it is important to obtain the temperature field of the entire treatment region. The objective of this dissertation is to develop a numerical model for obtaining an optimal temperature distribution in a 3D triple-layered cylindrical skin structure. To this end, we pre-specify the temperatures to be obtained at the center and perimeter on the surface of the cylinder. To deliver the energy to the perimeter of the skin structure during the certain exposure time, a laser irradiation pattern is configured, too. Further, the Pennes’ bioheat transfer model is employed in this study.

Finite difference scheme for solving the Pennes’ bioheat transfer equation in the 3D triple-layered cylindrical skin structure is then developed and is shown to be unconditionally stable with respect to the heat source. Since the laser power needs to be determined, the least squares sum between the pre-specified temperature and the calculated temperature is analyzed in order to optimize the laser power. As such, we have developed two algorithms which can be used for obtaining an optimal temperature distribution in a 3D triple-layered skin structure. To test these two algorithms, we have
applied them to calculate temperature distributions in a 3D triple-layered cylindrical skin structure without any blood vessels and with a blood vessel, respectively. Numerical results show that the method is efficient and it can be used for certain types of hyperthermia cancer treatments, such as skin cancer.
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

Date

4/18/05
# TABLE OF CONTENTS

| LIST OF TABLES ................................................................. | viii |
| LIST OF FIGURES ............................................................ | ix |
| NOMENCLATURE ............................................................... | xiv |
| ACKNOWLEDGMENTS .......................................................... | xv |

**CHAPTER I INTRODUCTION** .................................................. 1

1.1 Overview ................................................................. 1
1.2 Objective of the Research ............................................... 2
1.3 Organization of the Dissertation ....................................... 3

**CHAPTER II BACKGROUND AND PREVIOUS WORK** ................. 4

2.1 Inverse Heat Conduction Method ....................................... 4
  2.1.1 Solution of the Least-Squares Equations ....................... 5
2.2 Preconditioned Richardson Iteration .................................. 10
2.3 Previous Work on Bioheat Transfer ................................... 12

**CHAPTER III MATHEMATICAL MODEL AND SCHEME** ............... 16

3.1 Governing Equations .................................................... 16
  3.1.1 Problem Description .................................................. 16
  3.1.2 Three-Dimensional Schematic Configuration .................. 16
  3.1.3 Governing Equation Introduction ................................ 17
3.2 Finite Difference Scheme ............................................... 19
  3.2.1 Notations and Scheme ................................................ 19
  3.2.2 Least Square Method and Preconditioned Richardson Iteration 21
3.3 Stability ................................................................. 23
3.4 Laser Irradiation Pattern ................................................ 29
  3.4.1 Pattern Description .................................................... 29
  3.4.2 Algorithm for the Irradiation Pattern ............ 29
<table>
<thead>
<tr>
<th>CHAPTER IV NUMERICAL EXAMPLE</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Description of the Example</td>
<td>32</td>
</tr>
<tr>
<td>4.2 Calculation and Results</td>
<td>36</td>
</tr>
<tr>
<td>4.2.1 Calculation Case 1</td>
<td>36</td>
</tr>
<tr>
<td>4.2.2 Calculation Case 2</td>
<td>48</td>
</tr>
<tr>
<td>4.2.3 Calculation Case 3</td>
<td>54</td>
</tr>
<tr>
<td>4.3 Grid Independent</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER V SKIN MODEL EMBEDDED WITH A BLOOD VESSEL</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Introduction of the Skin Model Embedded with a Blood Vessel</td>
<td>64</td>
</tr>
<tr>
<td>5.2 Numerical Method of the Skin Model Embedded with a Blood Vessel</td>
<td>69</td>
</tr>
<tr>
<td>5.3 Result and Discussion of the Skin Model Embedded with a Blood Vessel</td>
<td>73</td>
</tr>
<tr>
<td>5.3.1 Calculation Case 1</td>
<td>74</td>
</tr>
<tr>
<td>5.3.2 Calculation Case 2</td>
<td>86</td>
</tr>
<tr>
<td>5.3.3 Calculation Case 3</td>
<td>92</td>
</tr>
<tr>
<td>5.4 Grid Independent Experiment</td>
<td>98</td>
</tr>
</tbody>
</table>

| CHAPTER VI FUTURE WORK AND CONCLUSION | 102 |

| APPENDIX A SOURCE CODE FOR SOLVING THE 3D SKIN STRUCTURE WITHOUT ANY BLOOD VESSELS | 104 |

| APPENDIX B SOURCE CODE FOR SOLVING THE 3D SKIN STRUCTURE EMBEDDED WITH A BLOOD VESSEL | 135 |

| REFERENCES | 170 |
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 4.1</td>
<td>Parameters for a 3D skin structure</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Pre-specified geometry parameters for the skin structure</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>SOURCE CODE FOR SOLVING THE 3D SKIN STRUCTURE WITHOUT ANY BLOOD VESSELS</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>SOURCE CODE FOR SOLVING THE 3D SKIN STRUCTURE EMBEDDED WITH A BLOOD VESSEL</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3.1</td>
<td>Schematic configuration of a 3D triple-layered skin structure irradiated by a laser and grid configuration on the surface</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>The flowchart of the laser irradiation pattern</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>A 3D skin structure</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>The elevated temperature profiles along the diameter on the skin surface with $\phi = 0$ and $\phi = \pi$, at $t = 410$ seconds without any blood vessels</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>The elevated temperature profiles along the diameter on the skin surface with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 410$ seconds without any blood vessels</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>The contours of the elevated temperature distributions in the cross section with $\phi = 0$ and $\phi = \pi$, at $t = 410$ seconds without any blood vessels</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>The contours of the elevated temperature distributions in the cross section with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 410$ seconds without any blood vessels</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 410$ seconds without any blood vessels</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>The elevated temperature profiles along the diameter on the skin surface with $\phi = 0$ and $\phi = \pi$, at $t = 674$ seconds without any blood vessels</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>The elevated temperature profiles along the diameter on the skin surface with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 674$ seconds without any blood vessels</td>
</tr>
</tbody>
</table>
The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 674$ seconds without any blood vessels.

The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 674$ seconds without any blood vessels.

The elevated temperature profile along the depth (the $z$-direction) at the center of the skin surface, at $t = 674$ seconds without any blood vessels.

The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 1062$ seconds without any blood vessels.

The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1062$ seconds without any blood vessel.

The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 1062$ seconds without any blood vessels.

The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1062$ seconds without any blood vessels.

The elevated temperature profile along the depth (the $z$-direction) at the center of the skin surface, at $t = 1062$ seconds without any blood vessels.

The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 1536$ seconds without any blood vessels.

The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1536$ seconds without any blood vessels.

The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 1536$ seconds without any blood vessels.

The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1536$ seconds without any blood vessels.
Figure 4.21 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at \( t = 1536 \) seconds without any blood vessels. ......................................................... 59

Figure 4.22 Elevated temperature profiles along the diameter on the skin surface with \( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 674 \) without any blood vessels. ................................................................. 61

Figure 4.23 Elevated temperature profiles along the diameter on the skin surface with \( \varphi = \frac{\pi}{2} \) and \( \varphi = \frac{3\pi}{2} \), at \( t = 674 \) without any blood vessels................................................................. 62

Figure 4.24 Elevated temperature profiles along the depth (the z-direction) at the center of the skin surface, at \( t = 674 \) without any blood vessels. ................................................................. 63

Figure 5.1 Skin structure. .................................................................................. 65

Figure 5.2 Configuration of a 3D skin structure. .................................................. 66

Figure 5.3 Streamline of the computation. ............................................................. 72

Figure 5.4 The elevated temperature profiles along the diameter on the skin surface with \( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 410 \) seconds with a blood vessel. ................................................................. 75

Figure 5.5 The elevated temperature profiles along the diameter on the skin surface with \( \varphi = \frac{\pi}{2} \) and \( \varphi = \frac{3\pi}{2} \), at \( t = 410 \) seconds with a blood vessel. ................................................................. 76

Figure 5.6 The contours of the elevated temperature distributions in the cross section with \( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 410 \) seconds with a blood vessel. ................................................................. 77

Figure 5.7 The contours of the elevated temperature distributions in the cross section with \( \varphi = \frac{\pi}{2} \) and \( \varphi = \frac{3\pi}{2} \), at \( t = 410 \) seconds with a blood vessel. ................................................................. 78

Figure 5.8 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at \( t = 410 \) seconds with a blood vessel. ................................................................. 79

Figure 5.9 The elevated temperature profiles along the diameter on the skin surface with \( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 672 \) seconds with a blood vessel. ................................................................. 81

Figure 5.10 The elevated temperature profiles along the diameter on the skin surface with \( \varphi = \frac{\pi}{2} \) and \( \varphi = \frac{3\pi}{2} \), at \( t = 672 \) seconds with a blood vessel. ................................................................. 82

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.11  The contours of the elevated temperature distributions in the cross section with $\phi = 0$ and $\phi = \pi$, at $t = 672$ seconds with a blood vessel. ......................................................... 83

Figure 5.12  The contours of the elevated temperature distributions in the cross section with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 672$ seconds with a blood vessel. ......................................................... 84

Figure 5.13  The elevated temperature profile along the depth (the $z$-direction) at the center of the skin surface, at $t = 672$ seconds with a blood vessel. ......................................................... 85

Figure 5.14  The elevated temperature profiles along the diameter on the skin surface with $\phi = 0$ and $\phi = \pi$, at $t = 1044$ seconds with a blood vessel. ......................................................... 87

Figure 5.15  The elevated temperature profiles along the diameter on the skin surface with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 1044$ seconds with a blood vessel. ......................................................... 88

Figure 5.16  The contours of the elevated temperature distributions in the cross section with $\phi = 0$ and $\phi = \pi$, at $t = 1044$ seconds with a blood vessel. ......................................................... 89

Figure 5.17  The contours of the elevated temperature distributions in the cross section with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 1044$ seconds with a blood vessel. ......................................................... 90

Figure 5.18  The elevated temperature profile along the depth (the $z$-direction) at the center of the skin surface, at $t = 1044$ seconds with a blood vessel. ......................................................... 91

Figure 5.19  The elevated temperature profiles along the diameter on the skin surface with $\phi = 0$ and $\phi = \pi$, at $t = 1509$ seconds with a blood vessel. ......................................................... 93

Figure 5.20  The elevated temperature profiles along the diameter on the skin surface with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 1509$ seconds with a blood vessel. ......................................................... 94

Figure 5.21  The contours of the elevated temperature distributions in the cross section with $\phi = 0$ and $\phi = \pi$, at $t = 1509$ seconds with a blood vessel. ......................................................... 95

Figure 5.22  The contours of the elevated temperature distributions in the cross section with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 1509$ seconds with a blood vessel. ......................................................... 96
Figure 5.23  The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1509$ seconds with a blood vessel .......................................................... 97

Figure 5.24  Elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ with a blood vessel .......................................................... 99

Figure 5.25  Elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 410$ with a blood vessel .......................................................... 100

Figure 5.26  Elevated temperature profiles along the depth (the z-direction) at the center of the skin surface, at $t = 410$ with a blood vessel .................................................. 101

Figure 6.1  Skin structure embedded with multi-level blood vessels ................. 103

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_i$</td>
<td>Biot number</td>
</tr>
<tr>
<td>$C_l$</td>
<td>specific heat of layer $l$</td>
</tr>
<tr>
<td>$C_b^l$</td>
<td>specific heat of blood in layer $l$</td>
</tr>
<tr>
<td>$I$</td>
<td>iterative index</td>
</tr>
<tr>
<td>$\mathbf{I}$</td>
<td>identity matrix</td>
</tr>
<tr>
<td>$i, j, k$</td>
<td>index of grid point</td>
</tr>
<tr>
<td>$k_l$</td>
<td>heat conductivity of layer $l$</td>
</tr>
<tr>
<td>$L_{pre}^l$</td>
<td>pre-conditioned Richardson operator</td>
</tr>
<tr>
<td>$L_l$</td>
<td>depth of layer $l$</td>
</tr>
<tr>
<td>$N_{r}, N_{\varphi}, N_{z}^l$</td>
<td>numbers of grid points in the $r, \varphi, z$ directions, respectively</td>
</tr>
<tr>
<td>$n$</td>
<td>time level</td>
</tr>
<tr>
<td>$P_0$</td>
<td>laser intensity</td>
</tr>
<tr>
<td>$Q_r^l$</td>
<td>heat source in layer $l$</td>
</tr>
<tr>
<td>$R$</td>
<td>number of grid point for the radius of the skin structure</td>
</tr>
<tr>
<td>$R_{bv}$</td>
<td>number of grid point for the radius of the blood vessel</td>
</tr>
<tr>
<td>$\text{Ref}_{l}$</td>
<td>laser reflectivity in layer $l$</td>
</tr>
<tr>
<td>$r, \varphi, z$</td>
<td>cylindrical coordinates</td>
</tr>
<tr>
<td>$S$</td>
<td>least squares sum</td>
</tr>
<tr>
<td>$t$</td>
<td>Time</td>
</tr>
<tr>
<td>$u_b^n, (u_b)_i$</td>
<td>numerical solutions of tissue and blood, respectively</td>
</tr>
<tr>
<td>$W_b^l$</td>
<td>blood perfusion rate of layer $l$</td>
</tr>
<tr>
<td>$\alpha_l$</td>
<td>laser absorptivity of layer $l$</td>
</tr>
<tr>
<td>$p_r^2, \delta_z^2, \delta_{\varphi}^2$</td>
<td>second order finite difference operators</td>
</tr>
<tr>
<td>$\rho_l$</td>
<td>density of layer $l$</td>
</tr>
<tr>
<td>$\omega$</td>
<td>relaxation parameter</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>standard deviation of laser beam width</td>
</tr>
<tr>
<td>$\theta_b, \theta, \theta_w$</td>
<td>elevated blood, tissue and vessel periphery temperatures, respectively</td>
</tr>
<tr>
<td>$\Delta r, \Delta \varphi, \Delta z$</td>
<td>mesh sizes in the $r, \varphi, z$ directions, respectively</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>time increment</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

I would like to take this opportunity to express my deep appreciation to my advisor, Dr. Weizhong Dai, for his invaluable guidance, encouragement, and generous support throughout my four years of study in the Ph.D. CAM program at Louisiana Tech University. Also, I would like to express special thanks to Dr. Raja Nassar, who gave generous advice and help for my research work and completion of my dissertation. I am also very grateful for Dr. Richard J. Greechie, Dr. Walter G. Besio and Dr. Andrei Paun, who graciously agreed to sacrifice their valuable time and energy to serve in my advisory committee and offer me precious suggestions 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 this research at Louisiana Tech University.

Finally, I wish to dedicate this dissertation and all my research work to my lovely family, for they have never lost their faith in me.
CHAPTER I

INTRODUCTION

1.1 Overview

In recent years, considerable research has been directed at hyperthermia combined with radiation and cytotoxic drugs to enhance the killing of tumors [Moroz 2002] [Muralidharan 2002] [Tsuda 1996] [Usatoff 2001] [Wust 2002]. Conventional hyperthermia (target temperatures of 42–46°C) in conjunction with radiation has demonstrated increased effectiveness in the treatment of certain types of cancer, such as those of liver metastases [Muralidharan 2002] [Hall 1984] [Streffer 1987]. The crucial problem is to heat the tumor tissue while keeping surrounding normal tissue below a temperature that will produce harm. Thus, it is important to obtain a temperature field of the entire treatment region. With the 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 pain and/or damage. However, it is not easy to obtain an accurate determination of the temperature field over the entire treatment region during clinical hyperthermia treatments because the number of invasive temperature probes that can be used is limited due to the pain tolerance of patients. The determinants of temperature distributions during thermal therapy are the power
deposition pattern of the heating source, heat removal by conduction, and heat removal by blood flow forced convection. They would involve numerical methods to solve the bioheat transfer equation for the human body [Chatterjee 1994].

1.2 Objective of the Research

The objective of this research is to develop a numerical model for optimizing laser power irradiating on a 3D triple-layered skin structure in cylindrical coordinates. 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. To achieve this objective, the following aims are pursued:

1. Develop a second-order accurate finite difference scheme for the 3D Pennes’ bioheat transfer equation.
2. Design a laser irradiation pattern.
3. Analyze the stability of the scheme by the discrete energy method.
4. Optimize the laser power by using inverse heat conduction method.
5. Solve the finite difference scheme by an iteration method.

The outcome of this study will provide an efficient and reliable numerical method for solving the 3D Pennes’ bioheat equation and give us better understanding of the nature of heat transport in such a skin structure. The research results will have a significant impact on hyperthermia combined with radiation and cytotoxic drugs to enhance the killing of tumors, such as skin cancer.
1.3 Organization of the Dissertation

The dissertation is organized as follows: in Chapter 2, we introduce the inverse heat conduction method and review previous relevant research. In Chapter 3, based on the Pennes' equation model, we state the Pennes' heat transport equation in 3D cylindrical coordinates with the initial and boundary conditions. The stability of the scheme is analyzed, and the inverse heat conduction method is applied. We then design a laser irradiation pattern to improve the efficiency in optimizing the laser power. To demonstrate the applicability of the scheme, the numerical examples are illustrated in Chapter 4. In Chapter 5, we apply the mathematical model to a skin structure embedded with a blood vessel. Numerical results of this model are shown in this chapter. Further, the conclusion and future work are discussed in Chapter 6.
CHAPTER II

BACKGROUND AND PREVIOUS WORK

2.1 Inverse Heat Conduction Method

Inverse problems are applied in the fields of mechanical, aerospace, and chemical engineers; mathematicians, astrophysicists, geophysicists, statisticians and specialists of many other disciplines. Many practical applications use the inverse analysis for the estimation of surface conditions, such as temperature and heat flux, or the determination of thermal properties like thermal conductivity and heat capacity of solids by using the transient temperature measurements taken within the medium. In the study of inverse analysis, the terminologies, function estimation, and parameter estimation are denoted. The problem is referred to be a problem of function estimation when it involves the determination of an unknown function, such as the timewise variation of surface heat flux without any prior knowledge of the functional form of the unknown quantity. On the other hand, if some prior knowledge is available on the functional form, it can be parameterized, and the inverse problem is called a problem of parameter estimation. Because we have pre-specified skin structure geometry, we deal with parameter estimation [Ozisik 1993] in this dissertation.
2.1.1 Solution of the Least-Squares Equations

The inverse problem is mathematically ill-posed. A successful solution of an inverse problem generally involves the transformation of the inverse problem into a well-posed approximate solution. Many techniques can be applied to transform an inverse problem into a well-posed approximate solution [Beck 1985]. In this research, we transform the inverse problem to a least squares problem. The inverse solution exists because the inverse solution minimizes the least squares norm. Solving the inverse problem required that the estimated temperature \( T_j(p), j = 1, 2, \ldots, M, \) computed from the solution of the direct problem by using the estimated values of the heat source \( \hat{p}_i, i = 1, 2, \ldots, M, \) should match the measured temperatures \( Y_j, j = 1, 2, \ldots, M, \) as closely as possible over a specified time domain \( 0 < t < t_f. \) Here, the superscript \( \hat{\cdot} \) over \( T \) or \( p \) denotes the estimated values. The least squares norm is modified by the addition of the zeroth-order regularization term [Hensel 1991]. The least squares norm is set up as

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

(2.1)

where

- \( i = \) the index number of grid points and \( N \) is the total number of grid points.
- \( j = \) the index number of unknown parameters, and \( M \) is the total number of unknown parameters to be predicted.
- \( Y_i = \) measured temperatures for each grid point.
- \( T_j(\hat{p}) = \) estimated temperature obtained from the solution of the direct problem by using the estimated values of the unknown parameters \( \hat{p} = \{\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_M\}. \)
\( \hat{p}_j \) = element of the estimated parameter vector \( \hat{p} = \{ \hat{p}_1, \hat{p}_2, ..., \hat{p}_M \} \).

\( \alpha^* \) = the regularization parameter, \( \alpha^* > 0 \).

In Eq. (2.1), the first summation term on the right-hand side is the traditional least squares. The second summation is the zero-order regularization term, added to reduce instability or oscillations 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 [Beck 1985]. Thus, selection of \( \alpha^* \) is crucial while the number of parameters is large. Because only optimized laser power interests us in this dissertation, \( \alpha^* \) is set to be zero.

Eq. (2.1) is minimized by differentiating it with respect to each of the unknown parameters \( p_j \) and then setting the resulting expression equal to zero.

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

where \( j, k = 1, 2, ..., M \), since components of unknown parameter vector \( p \) are independent,

\[
\frac{\partial \hat{p}_k}{\partial \hat{p}_j} = \begin{cases} 
0 & \text{for } k \neq j \\
1 & \text{for } k = j.
\end{cases} \tag{2.3a}
\]

Here, the total number of grid points \( N \) should be larger than the number of unknown parameters \( M \) [Beck 1977]. In addition, the number of grid points should also ensure uniqueness of the estimated thermal property parameters [Pzosol 1993].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Equation (2.2) can be rearranged in the form

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

(2.3b)

where \( i = 1, 2, ..., N \) and \( j, k = 1, 2, ..., M \) and

\[
\frac{\partial \hat{T}_i(\hat{p})}{\partial \hat{p}_j} = \frac{\partial \hat{T}_i(\hat{p}_1, \hat{p}_2, ..., \hat{p}_M)}{\partial \hat{p}_j} = X_{ji}.
\]

(2.3c)

\( X_{ji} \) is called the sensitivity coefficient with respect to \( \hat{q}_i \). Eq. (2.3b) can be written in matrix form as

\[
X^T (Y - T) = \alpha^* p
\]

(2.4a)

where

\[
T = \begin{bmatrix}
\hat{T}_1 \\
\hat{T}_2 \\
\vdots \\
\hat{T}_N
\end{bmatrix}, \quad Y = \begin{bmatrix}
Y_1 \\
Y_2 \\
\vdots \\
Y_N
\end{bmatrix}, \quad p = \begin{bmatrix}
\hat{p}_1 \\
\hat{p}_2 \\
\vdots \\
\hat{p}_M
\end{bmatrix}
\]

(2.4b)

\[
X = \frac{\partial T}{\partial p} = \begin{bmatrix}
\frac{\partial \hat{T}_1}{\partial \hat{p}_1} & \frac{\partial \hat{T}_1}{\partial \hat{p}_2} & \cdots & \frac{\partial \hat{T}_1}{\partial \hat{p}_M} \\
\frac{\partial \hat{T}_2}{\partial \hat{p}_1} & \frac{\partial \hat{T}_2}{\partial \hat{p}_2} & \cdots & \frac{\partial \hat{T}_2}{\partial \hat{p}_M} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial \hat{T}_N}{\partial \hat{p}_1} & \frac{\partial \hat{T}_N}{\partial \hat{p}_2} & \cdots & \frac{\partial \hat{T}_N}{\partial \hat{p}_M}
\end{bmatrix},
\]

(2.4c)

Here, \( X \) is called the sensitivity coefficient matrix with respect to vector \( p \), and the elements of this matrix are

\[
X_{ji} = \frac{\partial \hat{T}_i}{\partial \hat{p}_j}, \quad i = 1, 2, ..., N \text{ and } j = 1, 2, ..., M.
\]

(2.5)
The sensitivity coefficient $X_{ji}$ defined by Eqs (2.3b), (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 $X_{ji}$.

Thus, through the above derivations, the inverse heat conduction problem (IHCP) is reduced to solve the system of least squares sum by a suitable algorithm.

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

$$\hat{T}_i = \hat{T}_0 + \sum_{k=1}^{N} \frac{\partial \hat{T}_i}{\partial \hat{p}_k} (\hat{p}_k - \hat{p}_0).$$  \hspace{1cm} (2.6a)

This result is expressed in matrix form as

$$T = T_0 + \frac{\partial T}{\partial p'} (p - p_0).$$ \hspace{1cm} (2.6b)

If one chooses $T_0 = 0$ and $p_0 = 0$, Eqs. (2.6a) and (2.6b) reduce, respectively, to

$$\hat{T}_j = \sum_{k=1}^{N} \frac{\partial \hat{T}_i}{\partial \hat{p}_k} \hat{p}_k$$ \hspace{1cm} (2.7a)

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

\[ T = \frac{\partial T}{\partial p'} \cdot p = X_p. \]  

(2.7b)

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

\[ \sum_{i=1}^{N} \frac{\partial T_i}{\partial \hat{p}_j} \left( Y_i - \sum_{h=1}^{N} \frac{\partial T_i}{\partial \hat{p}_h} \hat{p}_h \right) = \alpha^* \sum_{k=1}^{M} \hat{p}_k \frac{\partial \hat{p}_k}{\partial \hat{p}_j}. \]  

(2.8a)

The matrix form of this equation is obtained by introducing Eq. (2.7b) into Eq. (2.4a) then we have

\[ X' (Y - \hat{X}p) = \alpha^* p. \]  

(2.8b)

The equivalence of Eqs. (2.8a) and (2.8b) can be verified by expanding Eq. (2.8b). The solution of Eq. (2.8a) or (2.8b) gives the estimated values of the heat flux components \( \hat{p}_i \) at each time \( t_i (i = 1, 2, ..., M) \). It is convenient to express the solution for the heat flux \( p \) in the matrix form as

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

(2.9)

Based on Eq. (2.9), the Levenberg-Marquardt's iterative algorithm [Beck 1977] is developed to calculate the unknown parameter vector \( p \) iteratively:

\[ p^{k+1} = p^k + (X'X + \alpha^* I)^{-1} X' (Y - T). \]  

(2.10)

This algorithm is a combination of 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. For \( \alpha^* \rightarrow 0 \), Eq. (2.10) reduces to the Newton's method and for \( \alpha^* \rightarrow \infty \), it becomes the steepest descent method.

The analysis and solution of this inverse problem are presented in the following basic steps:

Step 1. The formulation of direct and inverse problems
Step 2. The transformation of the inverse problem into a system of least squares sum equations

Step 3. Physical significance of sensitivity coefficients

Step 4. The solution of the least-squares equations

Step 5. The determination of the sensitivity coefficients

Step 6. Numerical results

2.2 Preconditioned Richardson Iteration

We now introduce a preconditioned Richardson iteration which is obtained in [Dai1998]. Consider the three-dimensional Poisson equation:

\[- \left( \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \right) = f(x, y, z), \tag{2.11} \]

let \( T_{ijk} \) denote the approximation to \( T(i\Delta x, j\Delta y, k\Delta z) \), where \( \Delta x, \Delta y \) and \( \Delta z \) are the grid sizes in the \( x, y \) and \( z \) directions, respectively, \( i = 0, ..., N_x, j = 0, ..., N_y \) and \( k = 0, ..., N_z \).

We use the center-difference equation:

\[ \frac{1}{\Delta x^2} \frac{\partial^2 T_{ijk}}{\partial x^2} = \frac{1}{\Delta x^2} \left( T_{i+1,jk} - 2T_{ijk} + T_{i-1,jk} \right) \tag{2.12} \]

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

\[- \left( \frac{1}{\Delta x^2} \frac{\partial^2}{\partial x^2} + \frac{1}{\Delta y^2} \frac{\partial^2}{\partial y^2} + \frac{1}{\Delta z^2} \frac{\partial^2}{\partial z^2} \right) T_{ijk} = f_{ijk}. \tag{2.13} \]
Let \((A_x \overline{T})_{ik} = -\frac{1}{\Delta x^2} \delta_x^2 T_{ijk}\), \((A_y \overline{T})_{ik} = -\frac{1}{\Delta y^2} \delta_y^2 T_{ijk}\), \((A_z \overline{T})_{ik} = -\frac{1}{\Delta z^2} \delta_z^2 T_{ijk}\), where \(A_x, A_y\) and \(A_z\) are matrices and \(\overline{T}\) is a vector consisting of \(T_{ijk}, i = 1, ..., N_x - 1, j = 1, ..., N_y - 1\) and \(k = 1, ..., N_z - 1\). Then the system Eq. (2.13) can be written in a vector form:
\[(A_x + A_y + A_z)\overline{T} = \overline{f}.\] (2.14)

It can be seen [Li 1979] that the eigenvalues of \(A_z\) are \(\frac{4}{\Delta z^2} \sin^2 \frac{k\pi\Delta z}{2}, k = 1, ..., N_z - 1\). Since \(\Delta z\) is very small compared with \(\Delta x\) and \(\Delta y\), the ratio \(\frac{\lambda_{\text{max}}(A_z)}{\lambda_{\text{min}}(A_z)} = O\left(\frac{1}{\Delta z^2}\right)\) is very large,
where \(\lambda_{\text{max}}(A_z)\) and \(\lambda_{\text{min}}(A_z)\) are maximum and minimum eigenvalues, respectively. The results in the system Eq. (2.14) are ill-conditioned. Hence, common iteration methods, such as the Gauss-Seidel method, will converge very slowly. To overcome this difficulty, we apply a preconditioning technique and the Richardson iteration on Eq. (2.14). It gives
\[L_{\text{pre}} \overline{T}^{(n+1)} = L_{\text{pre}} \overline{T}^{(n)} - \alpha \left[(A_x + A_y + A_z)\overline{T}^{(n)} - \overline{f}\right],\] (2.15)
where the preconditioner is chosen as follows:
\[L_{\text{pre}} \equiv A_z + \left(\frac{4}{\Delta x^2} + \frac{4}{\Delta y^2}\right)I,\] (2.16)
and \(\alpha\) is a relaxation parameter. 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 \(\rho(B) < 1\). Further, the smaller \(\rho(B)\) is, the faster the iteration converges. It can be shown that the eigenvalues of \(L_{\text{pre}}^{-1}(A_x + A_y + A_z)\) have the form
\[
\lambda_{ijk} = \frac{4}{\Delta x^2 \sin^2 \frac{i\pi \Delta x}{2} + 4 \sin^2 \frac{j\pi \Delta y}{2} + \frac{4 \sin^2 \frac{k\pi \Delta z}{2}}{2}}.
\]

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

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, we conclude that the iteration method Eq. (2.15) converges very fast [Dai 1998].

2.3 Previous Work on Bioheat Transfer

In the past few years, interest has been rekindled in the use of heat combined with radiation and cytotoxic drugs to enhance the killing of tumors [Moroz 2002] [Muralidharan 2002][Tsuda 1996][Usatoff 2001][Wust 2002]. Conventional hyperthermia (target temperatures of 42 – 46°C ) in conjunction with radiation has demonstrated increased effectiveness in the treatment of certain types of cancer, such as those of liver metastases [Muralidharan 2002][Streffer 1987][Hall 1984]. The challenge in hyperthermia lies in selectively heating the tumor tissue while maintaining the surrounding normal tissue below a temperature that will produce harm. Obtaining a temperature field of the entire treatment region is therefore critical. With the 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 pain and/or damage. However, during clinical hyperthermia treatments it is
difficult to obtain an accurate determination of the temperature field over the entire treatment region since the number of invasive temperature probes that can be used is limited because of the pain tolerance of patients. The determinants of temperature distributions during thermal therapy are the power deposition pattern of the heating source, heat removal by conduction, and heat removal by blood flow forced convection. They would involve numerical methods to solve the bioheat transfer equation for the human body [Chatterjee 1994]. Most utilized models for hyperthermia treatment planning involve the Pennes’ bioheat transfer equation (BHTE). In the BHTE model, heat transfer between the blood vessels and tissue is assumed to occur mainly across the capillaries where 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 blood flow could be 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 and based on these two models. [Clegg 1989 et al.] performed hyperthermia sessions on a normal canine thigh to test the ability of a state and parameter estimation method to accurately predict the complete 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 1989] presented the exact steady state and transient solutions for the temperature distribution in laser irradiated and perfused tissue using the Pennes’ equation under cylindrical coordinates. The solutions
obtained are used to evaluate the significance of blood perfusion during continuous-wave laser heating. [Liauh 1993] presented a semilinear state and parameter estimation algorithm that decreases the total computational time required to accurately reconstruct complete hyperthermia temperature fields because the relationship between the temperature and the blood perfusion based on the Pennes' bioheat transfer equation is generally nonlinear in the hyperthermia temperature estimation problem. [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 1994] considered the heat transfer within a perfused tissue in the presence of a vessel. The Pennes' bioheat 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' bioheat transfer equation with a blood vessel were obtained. [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 1999] considered the thermal processes proceeding within a perfused tissue in the presence of a vessel. The Pennes' bioheat transfer equation determines 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 is solved by 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 co-workers [Liu 1995] [Liu1999] [Liu 2000a] introduced a general form of the thermal wave model of Pennes’ bioheat transfer in living tissues. The model was obtained based on a modified unsteady at conduction equation (the CV equation). A general heat flux criterion has been established to determine when the thermal wave propagation dominates the principal heat transfer process and the model can be used for tissue temperature prediction. [Liu 1998], [Liu 2000b] also used the dual reciprocity boundary element method to solve the integral inverse or direct bioheat transfer problems. Although the laser-induced hyperthermia was studied [Roemer 1989] [Roemer 1991] [Usatoff 2001] [Waldow 1988] [Wang 1992], the numerical model for the laser-induced hyperthermia in a triple-layered skin structure composed of epidermis, dermis, and subcutaneous has not been studied. Recently, [Dai 2003a] [Dai 2003b] have developed a domain decomposition method for solving the 3D Pennes’ bioheat transfer equation in a rectangular triple-layered skin structure. This dissertation is to extend Dai and co-workers’ study to a 3D cylindrical triple-layered skin structure case.

In this chapter, we have introduced the IHCP and preconditioned Richardson iteration which will be applied for our research. We also have briefly reviewed the relevant research on bioheat transfer.
CHAPTER III

MATHEMATICAL MODEL AND SCHEME

3.1 Governing Equations

3.1.1 Problem Description

In this study, we will develop a numerical method for solving the 3D Pennes’ bioheat transfer equation in a triple-layered skin structure composed of epidermis, dermis and subcutaneous where the surface of the skin is irradiated by a laser. 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 Three-Dimensional Schematic Configuration

![Schematic configuration of a 3D triple-layered skin structure irradiated by a laser and grid configuration on the surface](image)

Figure 3.1 Schematic configuration of a 3D triple-layered skin structure irradiated by a laser and grid configuration on the surface
Figure 3.1 shows the 3D cylindrical coordinates, where $r$ is the radius of the target region, ranging from 0 to 0.5mm; $\varphi$ is the angle between the project of $r$ on the xy-plane with positive x-axis, ranging from 0 to $2\pi$; and $z$ is the depth of the region.

### 3.1.3 Governing Equation Introduction

The Pennes’ equation that describes the thermal behavior of triple-layered skin structure when irradiated by a laser can be expressed in cylindrical coordinate as follows:

$$\rho_i C_i \frac{\partial \theta_i}{\partial t} + W_i C_i \theta_i - k_i \left[ \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial \theta_i}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 \theta_i}{\partial \varphi^2} + \frac{\partial^2 \theta_i}{\partial z^2} \right] = Q'_i, \quad i = 1, 2, 3 \tag{3.1}$$

where $\theta_i$ is the elevated tissue temperature above the ambient temperature due to heating by a laser, $\rho_i$, $C_i$, and $k_i$ denote density, specific heat, and thermal conductivity of tissue, respectively, furthermore $C_i'$ is the specific heat of blood, $W_i'$ is the blood perfusion rate, and $Q'_i$ is volumetric heat due to spatial heating. Here, we assume that the laser power is continuous and spatial with a normal distribution. As such, the heat source $Q'_i$ can be described as follows [Jaesung 1994]:

$$Q_1 = \alpha_1 e^{-a_1 z} \frac{1}{\sqrt{2\pi}\sigma^2} e^{\frac{(r \cos \varphi - x_0)^2 + (r \sin \varphi - y_0)^2}{2\sigma^2}} P_0 (1 - R_{eff_1}) f(t),$$

$$Q_2 = \alpha_2 e^{-a_2 (L_1 - a_2 z)} \frac{1}{\sqrt{2\pi}\sigma^2} e^{\frac{(r \cos \varphi - x_0)^2 + (r \sin \varphi - y_0)^2}{2\sigma^2}} P_0 (1 - R_{eff_2}) f(t),$$

$$Q_3 = \alpha_3 e^{-a_3 (L_2 - a_3 z)} \frac{1}{\sqrt{2\pi}\sigma^2} e^{\frac{(r \cos \varphi - x_0)^2 + (r \sin \varphi - y_0)^2}{2\sigma^2}} P_0 (1 - R_{eff_3}) f(t), \tag{3.2}$$

where $\alpha_1, \alpha_2, \alpha_3$ are laser absorptivity of the three layers respectively; $R_{eff_1}, R_{eff_2}, R_{eff_3}$ are laser reflectivity of three layers of the skin, respectively; $\sigma$ is the standard deviation of the width of a normally distributed laser beam, and $L_1, L_2, L_3$ are the depths of the
three layers of the skin, respectively. Here, \((x_0, y_0)\) is the location where the laser is focused, and \(f(t)\) is a function of time \(t\). The interfacial conditions and boundary conditions are assumed to be as follows:

\[
\frac{\partial \theta_j}{\partial z} = 0, \quad z = 0, \tag{3.3}
\]

\[
\theta_1 = \theta_2, \quad k_j \frac{\partial \theta_1}{\partial z} = k_2 \frac{\partial \theta_2}{\partial z}, \quad z = L_j, \tag{3.4}
\]

\[
\theta_2 = \theta_3, \quad k_2 \frac{\partial \theta_2}{\partial z} = k_3 \frac{\partial \theta_3}{\partial z}, \quad z = L_1 + L_2, \tag{3.5}
\]

\[
\frac{\partial \theta_3}{\partial z} = 0, \quad z = L_1 + L_2 + L_3. \tag{3.6}
\]

On the lateral walls we assume that

\[
\frac{\partial \theta_j}{\partial r} = 0, \quad r = R, \tag{3.7a}
\]

and

\[
\theta_j(r, \varphi, z) = \theta_j(r, \varphi + 2m \pi, z). \tag{3.7b}
\]

The initial condition is assumed to be

\[
\theta_j = 0, \quad t = 0, \quad l = 1, 2, 3. \tag{3.8}
\]
3.2 Finite Difference Scheme

3.2.1 Notations and Scheme

To develop a numerical model for solving the above problem, we let \((u_t)_{ijk}^n\) be the numerical approximation of \((\theta_t)(i\Delta r, j\Delta \varphi, k\Delta z, n\Delta t)\), where \(\Delta r, \Delta \varphi, \Delta z\) and \(\Delta t\) are the spatial and temporal mesh sizes, respectively.

Here \(i, j, k\) are chosen to be \(1 \leq i \leq N_r, 1 \leq j \leq N_{\varphi}, 1 \leq k \leq N_z\), so that
\[N_r \cdot \Delta r = R, \quad N_{\varphi} \cdot \Delta \varphi = 2\pi, \quad N_z \cdot \Delta z = L_z, \quad l = 1, 2, 3.\]

We employ second-order finite differences to approximate
\[
\frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial \theta_t}{\partial r} \right), \quad \frac{\partial^2 \theta_t}{\partial z^2}
\]
and
\[
\frac{1}{r^2} \frac{\partial^2 \theta_t}{\partial \varphi^2}
\]
at point \(((i\Delta r, j\Delta \varphi, k\Delta z, n\Delta t)\) as follows:

\[
\frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial \theta_t}{\partial r} \right) \approx \frac{(i + \frac{1}{2})\theta_{i+1,jk} - 2i\theta_{jk} + (i - \frac{1}{2})\theta_{i-1,jk}}{r_i \Delta r},
\]
\[
\frac{\partial^2 \theta_t}{\partial z^2} \approx \frac{\theta_{ijk+1} - 2\theta_{ijk} + \theta_{ijk-1}}{\Delta z^2},
\]
\[
\frac{1}{r^2} \frac{\partial^2 \theta_t}{\partial \varphi^2} \approx \frac{\theta_{ij+k} - 2\theta_{ijk} + \theta_{ij-k}}{r_i^2 \cdot \Delta \varphi^2}.
\]

Using the Crank-Nicholson finite difference method, a scheme for solving the above initial and boundary triple-layered skin structure problem can be developed as follows:
\[
\rho C_i \frac{(u_i)_{i,j,k}^{n+1} - (u_i)_{i,j,k}^n}{\Delta t} + \frac{W_i C_i}{2} [(u_i)_{i,j,k}^{n+1} + (u_i)_{i,j,k}^n] \\
- \frac{r_i}{2} [(u_i)_{i+1,j,k}^{n+1} + (u_i)_{i+1,j,k}^n] - 2r_i [(u_i)_{i,j,k}^{n+1} + (u_i)_{i,j,k}^n] + r_i [(u_i)_{i-1,j,k}^{n+1} + (u_i)_{i-1,j,k}^n] \\
+ \frac{1}{2} [(u_i)_{i,j+1,k}^{n+1} + (u_i)_{i,j+1,k}^n] - 2[(u_i)_{i,j,k}^{n+1} + (u_i)_{i,j,k}^n] + [(u_i)_{i,j-1,k}^{n+1} + (u_i)_{i,j-1,k}^n] \\
r_i^2 \Delta r^2 \\
\delta_i^2[(u_i)_{i,j,k}^{n+1} + (u_i)_{i,j,k}^n]] = (Q_i)_{i,j,k}^{n+1}, \quad i = 1, 2, 3,
\]

where \(\delta_i^2 u_{i,j,k}^n = \frac{u_{i,j,k}^{n+1} - 2u_{i,j,k}^n + u_{i,j,k}^{n-1}}{\Delta z^2}\), and so on. The discrete interfacial equations are assumed to be, for any time level

\[
k_1 \frac{(u_i)_{i,j+1,k}^n - (u_i)_{i,j-1,k}^n}{\Delta z} = k_2 \frac{(u_i)_{i,j+1}^n - (u_i)_{i,j-1}^n}{\Delta z}, \quad (u_i)_{i,j+1}^n = (u_i)_{i,j}^n, \quad (u_i)_{i,j-1}^n = (u_i)_{i,j}^n,
\]

and

\[
k_2 \frac{(u_i)_{i,j+1,k}^n - (u_i)_{i,j+1,k}^n}{\Delta z} = k_3 \frac{(u_i)_{i,j+1}^n - (u_i)_{i,j-1}^n}{\Delta z} = k_3 \frac{(u_i)_{i,j+1}^n - (u_i)_{i,j-1}^n}{\Delta z}, \quad (u_i)_{i,j+1}^n = (u_i)_{i,j}^n.
\]

The initial and boundary condition are chosen to be

\[
(u_i)_{i,j,k}^0 = 0,
\]

\[
(u_i)_{i,j,k}^n = (u_i)_{i,j}^n, \quad (u_i)_{i,j+1,k}^n = (u_i)_{i,j+1,k}^n, \quad (u_i)_{i,j-1,k}^n = (u_i)_{i,j-1,k}^n,
\]

\[
(u_i)_{i,j,0}^n = (u_i)_{i,j}^n, \quad (u_i)_{i,j,k}^n = (u_i)_{i,j,k}^n, \quad (u_i)_{i,j+1,k}^n = (u_i)_{i,j+1,k}^n + \delta_i^2 u_{i,j,k}^n,
\]

for any time level \(n\). Since the heat source \(P_0\) is unknown, we need to show that the scheme is stable with respect to the heat source. A proof will be given in section 3.3.
3.2.2 Least Square Method and Preconditioned Richardson Iteration

To determine the laser power $P_0$, we first pre-specify the laser irradiation time to obtain the pre-specified temperatures at given locations in this 3D skin structure. By guessing an initial laser power, one may obtain a temperature distribution in the entire 3D skin structure from Eqs. (3.9)-(3.14). Once the elevated temperatures $(u_{cal})_i (i = 1,2,3\ldots M)$ at the given locations are obtained, a least squares approach is employed to minimize the difference between the pre-specified elevated temperature $\theta_{pre}$ and the temperature distribution $u_{cal}$ as follows:

$$S(P_0) = \sum_{i=0}^{M} [(\theta_{pre}^{i} - u_{cal}^{i})^2], \quad i = 0,1,\ldots, M.$$  

(3.15)

Minimizing $S(P_0)$ in the above Eq. (3.15), one can obtain

$$\frac{d}{dP_0} S(P_0) = 2 \sum_{i=0}^{M} \frac{d(u_{cal}^{i})}{dP_0} [(\theta_{pre}^{i} - (u_{cal})^{i})] = 0.$$  

(3.16)

Hence, a new $P_0$ can be calculated iteratively as follows:

$$(P_0^{(k+1)}) = (P_0^{(k)}) + (X'X + \alpha^* I)^{-1} X'[(\theta_{pre})^{i} - (u_{cal})^{i}] ,$$  

(3.17)

where $X$ is the sensitivity coefficient matrix, which is a $1 \times (M+1)$ vector:

$$X = \left[ \frac{\partial(u_{cal})^0}{\partial P_0} \frac{\partial(u_{cal})^1}{\partial P_0} \ldots \frac{\partial(u_{cal})^M}{\partial P_0} \right]^\top ,$$  

(3.18)

and

$$\theta_{pre} = \left[ \begin{array}{c} \theta_{pre}^0 \\ \theta_{pre}^1 \\ \vdots \\ \theta_{pre}^M \end{array} \right], \quad \rightarrow \quad u_{cal} = \left[ \begin{array}{c} u_{cal}^0 \\ u_{cal}^1 \\ \vdots \\ u_{cal}^M \end{array} \right] .$$  

(3.19)
Hence, an algorithm for calculating the required laser power $P_0$ to obtain the pre-specified temperatures at the given locations in a 3D skin structure after a pre-specified time can be described as follows:

Step 1. Pre-specify the elevated temperatures $\theta_{pre}^i (i = 1, 2, 3, \ldots, M)$ at given $M$ grid points in the skin structure, and pre-specify the laser irradiation time $t$ needed for obtaining these pre-specified temperatures.

Step 2. Guess an initial laser power and its small increment $P_0$ and $P_0 + \Delta P_0$, and obtain the temperature distribution in the entire 3D skin structure by solving Eqs. (3.9)-(3.14).

Step 3. Determine a new laser power by Eqs. (3.17)-(3.19) and repeat the computation until the following convergence criterion is satisfied:

$$\frac{|S(P_0^{k+1}) - S(P_0^k)|}{S(P_0^{k+1})} < \varepsilon.$$  \hspace{1cm} (3.20)

It should be pointed out that Eq. (3.9) is a three-dimensional implicit scheme and that the computation is very slow because the grid size is very small in the first layer. To speed up the computation, we employ a preconditioned Richardson iteration as described in section 2.2 and [Dai 2003a] [Dai 2003b] as follows:

$$L_{pre}^I ((u_{l_{ij}})^{n+1}) = L_{pre}^I ((u_{l_{ij}})^{n}) - \omega ((u_{l_{ij}})^{n+1}) - (u_{l_{ij}})^{n} + \frac{W_r C_r}{2 \rho C} \left( (u_{l_{ij}})^{n+1} + (u_{l_{ij}})^{n} \right)$$

$$\frac{k \Delta t}{2 \rho C} \left[ \frac{r_{i,j}}{r_{i+1,j}} [(u_{l_{ij}})^{n+1} + (u_{l_{ij}})^{n}] - 2r_i [(u_{l_{ij}})^{n+1}] + (u_{l_{ij}})^{n} \right] + r_i \frac{1}{r_{i-1,j}} \left[ (u_{l_{ij}})^{n+1} + (u_{l_{ij}})^{n} \right]$$

$$+ \delta^2 \left[ (u_{l_{ij}})^{n+1} + (u_{l_{ij}})^{n} \right] + \frac{\alpha \Delta t}{\rho C} \left( (Q_{l_{ij}})^{n+1} + (Q_{l_{ij}})^{n} \right)),$$  \hspace{1cm} (3.21)
where \( l = 1, 2, 3 \), \( I = 1, 2, 3, \ldots, n \) and
\[
\delta^2 = \frac{1}{r_i^2} \left( u_i^{n+1}_{jk} - 2(u_i^n_{jk} + u_i^n_{jl}) \right),
\]
\[
\delta^2 = \frac{\theta_{jk+1} - 2\theta_{jk} + \theta_{jk-1}}{\Delta r^2}
\]
and the preconditioned operator
\[
L_{prec}^l = 1 + \frac{W_C^l \Delta t}{2 \rho_i C_i} + \frac{k_i \Delta t}{2 \rho_i C_i} \left( \frac{2r_i + r_{i-\frac{1}{2}} + r_{i+\frac{1}{2}}}{r_i \Delta r^2} \right) I - \frac{k_i \Delta t}{2 \rho_i C_i} \Delta \varphi^2 \cdot r_i^2 \delta,
\]
and \( \omega \) is a relaxation parameter \((0 \leq \omega \leq 1)\). Combing Eqs. (3.21)-(3.22) with the interfacial equations, Eqs. (3.10)-(3.11), and the boundary conditions, Eqs. (3.13)-(3.14), one may obtain a tridiagonal linear system \( Au = d \).

### 3.3 Stability

In this section, we will show the scheme, Eqs. (3.9)-(3.14), to be unconditionally stable. For simplicity, we assume that \((u_i^n_{jk} = (u_i^n)_{1jk}\) and introduce the definitions of the inner products and norms between the mesh functions \(u^n_{jk}\) and \(v^n_{jk}\) as follows:

\[
(u^n, v^n)_I = \Delta r \Delta \varphi \Delta z \sum_{i=1}^{N_r-1} \sum_{j=1}^{N_q} \sum_{k=1}^{N_z-1} u^n_{ijk} v^n_{ijk}, \quad \|u^n\|^2_I = (u^n, u^n)_I,
\]

\[
\|\nabla_r u^n\|_I = (\nabla_r u^n, \nabla_r u^n)_I = \Delta r \Delta \varphi \Delta z \sum_{i=1}^{N_r-1} \sum_{j=1}^{N_q} \sum_{k=1}^{N_z-1} (\nabla_r u^n_{ijk})^2_i
\]

where \( l = 1,2,3 \), and \( \nabla_r \) is the first-order backward finite difference operator such that

\[
\nabla_r u^n_{ijk} = \frac{u^n_{ijk} - u^n_{i-1,jk}}{\Delta r},
\]

and so on for the \( \varphi \) and \( z \) directions.

**Lemma 1.** If \((u_i^n)_{1jk}, l = 1,2,3\) is the solution of Eqs. (3.9)-(3.14), then
\begin{align}
&k_1 \sum_{k=1}^{N_1-1} \delta_z^2 [(u_1)_{ijk}^{n+1} + (u_{1})_{ijk}^{n}] 
&+ \sum_{k=1}^{N_1-1} \delta_z^2 [(u_2)_{ijk}^{n+1} + (u_{2})_{ijk}^{n}] 
&+ \sum_{k=1}^{N_1-1} \delta_z^2 [(u_3)_{ijk}^{n+1} + (u_{3})_{ijk}^{n}] 
&\quad = -k_1 \sum_{k=1}^{N_1} \nabla_z [(u_1)_{ijk}^{n+1} + (u_{1})_{ijk}^{n}]^2 
&- k_2 \sum_{k=1}^{N_1} \nabla_z [(u_2)_{ijk}^{n+1} + (u_{2})_{ijk}^{n}]^2 
&- k_3 \sum_{k=1}^{N_1} \nabla_z [(u_3)_{ijk}^{n+1} + (u_{3})_{ijk}^{n}]^2.
\end{align}

\begin{align}
&\sum_{i=1}^{N_r-1} r_i p_i^2 [(u_1)_{ijk}^{n+1} + (u_{1})_{ijk}^{n}] 
&\quad = -\sum_{i=1}^{N_r-1} r_i \nabla_p [(u_1)_{ijk}^{n+1} + (u_{1})_{ijk}^{n}]^2.
\end{align}

**Proof.** Let $U_i^l = (u_i)_{ijk}^{n+1} + (u_{i})_{ijk}^{n}$, $l = 1, 2, 3$. As such, the left-hand-side (LHS) of Eq. (3.24) can be simplified as follows:

\begin{align}
LHS &= k_1 \sum_{k=1}^{N_1-1} \delta_z^2 U_k^{(1)} \cdot U_k^{(1)} 
&+ k_2 \sum_{k=1}^{N_2-1} \delta_z^2 U_k^{(2)} \cdot U_k^{(2)} 
&+ k_3 \sum_{k=1}^{N_3-1} \delta_z^2 U_k^{(3)} \cdot U_k^{(3)}
\end{align}
Based on Eq. (3.13), the LHS can be further written as follows:

\[ \text{LHS} = \frac{1}{\Delta z^2} k_1 \left[ \sum_{k=2}^{N_z-1} \left( U_k^{(1)} - U_{k-1}^{(1)} \right) \cdot U_{k-1}^{(1)} - \sum_{k=1}^{N_z-1} \left( U_k^{(1)} - U_{k-1}^{(1)} \right) \cdot U_k^{(1)} \right] 
\]
\[ + \frac{1}{\Delta z^2} k_2 \left[ \sum_{k=1}^{N_z-1} \left( U_k^{(2)} - U_{k-1}^{(2)} \right) \cdot U_{k-1}^{(2)} - \sum_{k=1}^{N_z-1} \left( U_k^{(2)} - U_{k-1}^{(2)} \right) \cdot U_k^{(2)} \right] 
\]
\[ + \frac{1}{\Delta z^2} k_3 \left[ \sum_{k=2}^{N_z-1} \left( U_k^{(3)} - U_{k-1}^{(3)} \right) \cdot U_{k-1}^{(3)} - \sum_{k=1}^{N_z-1} \left( U_k^{(3)} - U_{k-1}^{(3)} \right) \cdot U_k^{(3)} \right] 
\]
\[ + \frac{1}{\Delta z^2} k_1 \left( U_1^{(1)} - U_0^{(1)} \right) \cdot U_0^{(1)} - \frac{1}{\Delta z^2} k_2 \left( U_1^{(2)} - U_0^{(2)} \right) \cdot U_0^{(2)} 
\]
\[ + \frac{1}{\Delta z^2} k_3 \left( U_1^{(3)} - U_0^{(3)} \right) \cdot U_0^{(3)} . 
\]

Using Eqs. (3.10)-(3.11) and then Eq. (3.13), we simplify the above LHS as follows:

\[ \text{LHS} = -k_1 \sum_{k=1}^{N_z-1} \nabla_z U_k^{(1)} \cdot \nabla_z U_k^{(1)} - k_2 \sum_{k=1}^{N_z-1} \nabla_z U_k^{(2)} \cdot \nabla_z U_k^{(2)} - k_3 \sum_{k=1}^{N_z-1} \nabla_z U_k^{(3)} \cdot \nabla_z U_k^{(3)} 
\]
\[ + \frac{1}{\Delta z^2} k_1 \left( U_{N_z}^{(1)} - U_{N_z-1}^{(1)} \right) \cdot U_{N_z-1}^{(1)} - \frac{1}{\Delta z^2} k_2 \left( U_{N_z}^{(2)} - U_{N_z-1}^{(2)} \right) \cdot U_{N_z-1}^{(2)} 
\]
\[ + \frac{1}{\Delta z^2} k_3 \left( U_{N_z}^{(3)} - U_{N_z-1}^{(3)} \right) \cdot U_{N_z-1}^{(3)} 
\]
\[ = -k_1 \sum_{k=1}^{N_z-1} \nabla_z U_k^{(1)} \cdot \nabla_z U_k^{(1)} - k_2 \sum_{k=1}^{N_z-1} \nabla_z U_k^{(2)} \cdot \nabla_z U_k^{(2)} - k_3 \sum_{k=1}^{N_z-1} \nabla_z U_k^{(3)} \cdot \nabla_z U_k^{(3)} , 
\]

which is the right-hand-side of Eq. (3.24). Using a similar argument, one may obtain Eqs. (3.25) and (3.26).

To show the scheme to be unconditionally stable with respect to the heat source, we assume that solutions \( (u_i)^n \) and \( (v_i)^n \), \( i, l = 1, 2, 3 \), are obtained by the scheme, Eq.
(3.9), with the same initial, boundary and interfacial conditions, Eqs. (3.10)-(3.14), except different source terms, \((Q_1)'\) and \((Q_2)'\). We let \((\varepsilon_i)'_{\eta k} = (\mu_i)'_{\eta k} - (\nu_i)'_{\eta k}\) and \(\sigma_i = (Q_1)'_i - (Q_2)'_i\). One may see that \((\varepsilon_i)'_{\eta k}\) satisfies Eqs. (3.10)-(3.14) and the following equation:

\[
\frac{\rho_i C_i (\varepsilon_i)'_{\eta k} - (\varepsilon_i)'_{\eta k}}{\Delta t} + W'_b C'_b \frac{(\varepsilon_i)'_{\eta k} + (\varepsilon_i)'_{\eta k}}{2} - k_i (P_r^2 + \delta_{\phi}^2 + \delta_{\phi}^2) \frac{(\varepsilon_i)'_{\eta k} + (\varepsilon_i)'_{\eta k}}{2} = (\sigma_i)'_{\eta k}, \quad l = 1, 2, 3.
\]

Multiply Eq. (3.30) with \(l = 1\) by \(2r_i \Delta r \Delta \phi \Delta z \Delta t [(\varepsilon_i)'_{\eta k} + (\varepsilon_i)'_{\eta k}]\), Eq. (3.30) with \(l = 2\) by \(2r_i \Delta r \Delta \phi \Delta z \Delta t [(\varepsilon_2)'_{\eta k} + (\varepsilon_2)'_{\eta k}]\), and Eq. (3.46) with \(l = 3\) by \(2r_i \Delta r \Delta \phi \Delta z \Delta t [(\varepsilon_3)'_{\eta k} + (\varepsilon_3)'_{\eta k}]\).

Then, sum over \(i, j, k\) from \(1 \leq i \leq N_r, -1 \leq j \leq N_\phi, 1 \leq k \leq N_z - 1\), respectively, adding them together and then using lemma 1, one obtains

\[
\sum_{l=1}^{3} \rho_i C_i [\| \sqrt{r} (\varepsilon_i)'_{\eta k} \|^2 - \| \sqrt{r} (\varepsilon_i)'_{\eta k} \|^2] + \Delta t \sum_{l=1}^{3} W'_b C'_b \| \sqrt{r} [(\varepsilon_i)'_{\eta k} + (\varepsilon_i)'_{\eta k}] \|^2
+ \Delta t \sum_{l=1}^{3} k_i \| E_i \frac{1}{2} r [\nabla_r (\varepsilon_i)'_{\eta k} + \nabla_r (\varepsilon_i)'_{\eta k}] \|_l + \Delta t \sum_{l=1}^{3} k_i \| \sqrt{r} [\nabla_\phi (\varepsilon_i)'_{\eta k} + \nabla_\phi (\varepsilon_i)'_{\eta k}] \|_l
+ \Delta t \sum_{l=1}^{3} k_i \| \sqrt{r} [\nabla_z (\varepsilon_i)'_{\eta k} + \nabla_z (\varepsilon_i)'_{\eta k}] \|_l = 2 \Delta t \sum_{l=1}^{3} ((\sigma_i)'_{\eta k}^2, r[(\varepsilon_i)'_{\eta k} + (\varepsilon_i)'_{\eta k}]),
\]

where \(E_i \frac{1}{2}\) is a shift operator such that \(E_i \frac{1}{2} r_i = r_{i-1}\). By the generalized Cauchy-Schwarz's inequality, we have
where \( \varepsilon \) is a positive constant. Substituting Eq. (3.32) into Eq. (3.31), we obtain

\[
2((\sigma_i)^{n+1}, r[(\varepsilon_i)^{n+1} + (\varepsilon_i)^n]) \leq \varepsilon \left\| r[(\varepsilon_i)^{n+1} + (\varepsilon_i)^n] \right\|^2 + \varepsilon^{-1} \left\| r[(\sigma_i)^{n+1}] \right\|^2
\]

(3.32)

\[
\leq 2\varepsilon \left\| r[(\varepsilon_i)^{n+1}] \right\|^2 + 2\varepsilon \left\| r[(\varepsilon_i)^n] \right\|^2 + \varepsilon^{-1} \left\| r[(\sigma_i)^{n+1}] \right\|^2,
\]

We denote \( F(n) = \sum_{l=1}^{3} 2\rho_l C_l \left\| r[(\varepsilon_i)^n] \right\|^2 \). Choosing \( \varepsilon = \rho_l C_l \), taking out the second, third, fourth, and fifth terms on the left hand side of Eq. (3.33), we simplify Eq. (3.33) as follows:

\[
(1 - \Delta t)F(n + 1) \leq (1 + \Delta t)F(n) + \Delta t \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \left\| r[(\sigma_i)^{n+1}] \right\|^2.
\]

(3.34)

Thus, we obtain

\[
F(n + 1) \leq \frac{1 + \Delta t}{1 - \Delta t} F(n) + \frac{\Delta t}{1 - \Delta t} \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \left\| r[(\sigma_i)^{n+1}] \right\|^2
\]

\[
\leq \frac{1 + \Delta t}{1 - \Delta t} \left[ \frac{1 + \Delta t}{1 - \Delta t} F(n - 1) + \frac{\Delta t}{1 - \Delta t} \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \left\| r[(\sigma_i)^{n+1}] \right\|^2 \right]
\]

\[
+ \frac{\Delta t}{1 - \Delta t} \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \left\| r[(\sigma_i)^{n+1}] \right\|^2 \leq \ldots
\]
Using the inequalities $(1 + \varepsilon)^n \leq e^{\varepsilon n}$ for $\varepsilon > 0$, and $(1 - \varepsilon)^{-1} \leq e^{2\varepsilon}$ when $0 < \varepsilon \leq \frac{1}{2}$, we obtain

$$F(n + 1) \leq e^{3n+1} \left[ F(0) + \frac{1}{\rho_i C_i} \max_{0 \leq m \leq n} \left\| \sqrt{r} \sigma_i \right\|_{m+1/2}^2 \right],$$

when $\Delta t \leq \frac{1}{2}$. From Eq. (3.12), we obtain that $F(0) = 0$ and hence

$$F(n + 1) \leq e^{3n} \sum_{i=1}^{3} \frac{1}{\rho_i C_i} \max_{0 \leq m \leq n} \left\| \sqrt{r} \sigma_i \right\|_{m+1/2}^2,$$

for $0 \leq (n + 1) \Delta t \leq t_0$. The following theorem has been obtained:

**THEOREM.** Assume that solution $(u_i)_{ijk}^n$ and $(v_i)_{ijk}^n$, $l = 1, 2, 3$ are obtained by the scheme, Eq. (3.9), with the same initial, boundary and interfacial conditions, Eqs. (3.10)-(3.14), except different source terms, $(Q_1)_l$ and $(Q_2)_l'$. Let $(e_i)_{ijk}^n = (u_i)_{ijk}^n - (v_i)_{ijk}^n$ and $\sigma_i = (Q_1)_l' - (Q_2)_l'$. Then $(e_i)_{ijk}^n$ satisfies, for $0 \leq n \Delta t \leq t_0$,
which implies that the scheme is unconditionally stable with respect to the heat source.

3.4 Laser Irradiation Pattern

3.4.1 Pattern Description

We have found that if we apply the least squares method for the five points, where one is at the central skin and four at the perimeter, the temperatures at the perimeter are much smaller than the pre-specified temperatures. To overcome this problem, we developed a laser irradiation pattern and describe it in the next section.

3.4.2 Algorithm for the Irradiation Pattern

To design the laser irradiation pattern, we denote two new terms. One is $S_p(P_0) = \sum_{i=1}^{M} [(\theta_{pre})^i - (u_{cal})^i]^2$, which is the least squares sum for the given perimeter locations. The other is $S^{specified}_p$, the pre-specified value of the least squares sum of $S_p(P_0)$.

Since the laser power is compliant to the Gaussian distribution, when the laser is focused on the center of the surface, the energy is too weak to elevate the temperature to the pre-specified temperatures at the perimeter during the short exposure time. In particular, if the radius of the target region is large, the temperatures at the perimeter are almost unchangeable. Thus, we propose a laser irradiation pattern to overcome this difficulty. The solution of the problem should match two requirements: first, the temperature at the center or perimeter is not over the specified temperature; second, no patient can tolerate a long irradiation time.
This pattern is to elevate first the temperature at the center on the surface to the pre-specified temperature and then let heat conduct to the perimeter by turning off the laser. When the center temperature is lower than the pre-specified perimeter temperature, the laser is turned on and heats up the center on the skin surface again. The algorithm can be described as follows:

Step 1. The skin is irradiated by the laser at the center for 10 seconds, and then the laser is moved to the grid point with $\varphi = 0, r = \Delta r$. Next, the laser is circulating counterclockwise twice on the remaining 20 pixels with focusing on each pixel for 10 seconds. Here, laser power $P_0$ is obtained by the inverse heat conduction method, which is applied to the center point. The pixels configuration is shown in Figure 3.1.

Step 2. An integer flag is used to indicate 0 for turning off the laser and 1 for turning on the laser. Perform the following procedure:

Procedure one: if the flag = 0{

Laser is turned off and heat conducts to the perimeter

If $u_{\text{Center}} < \theta_{\text{Perimeter}}^\text{pre}$ or $u_{\text{Perimeter}} > \theta_{\text{Perimeter}}^\text{pre}$ {

if $S_p(P_0) < S_p^{\text{specified}}$, record time T1 and begin step 3;

else, set the flag equal to 1 and go to procedure two;

}

}

Procedure two: if the flag = 1{

Heat the center of the skin with power $P_0$;

if $S_p(P_0) < S_p^{\text{specified}}$, record time T1 and begin step 3;
if $u_{\text{cal}}^{\text{Center}} > \theta_{\text{pre}}^{\text{Center}}$, set flag to 0 and go to procedure one

Step 3. Heat the center with power $P_0$ till $u_{\text{cal}}^{\text{Center}} > \theta_{\text{pre}}^{\text{Center}}$ and record $\Delta T$ (predicted irradiation time). Apply the inverse heat conduction problem method to the five points to obtain a new $P_0$, and heat the center of the skin within $\Delta T$. The above algorithm is illustrated by a flowchart shown in Figure 3.2.

![Flowchart](image_url)

Figure 3.2 The flowchart of the laser irradiation pattern
CHAPTER IV

NUMERICAL EXAMPLE

4.1 Description of the Example

We tested our algorithm in a 3D skin structure as shown in Figure 4.1, where the parameter values were chosen from Table 4.1 and the size of skin structure is given in Table 4.2. A mesh of $30 \times 20 \times 1208$ in $(r, \varphi, z)$ was employed in the computation. In our calculation, we pre-specified the elevated temperatures at the center of the skin surface and four locations, 90 degree apart (that is, $\varphi = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$) at the perimeter.

A laser exposure pattern was designed as follows: 21 pixels on the skin surface were chosen for laser irradiation. They included the center pixel and those grid points (20 points) with a $\Delta r$ distance from the center. The laser was set to irradiate at the center for 10 seconds, after which it was moved to the grid point with $\varphi = 0$ and circulated twice counter-clockwise over the 20 pixels with laser focused for 10 seconds on each pixel.

There are three cases were tested in this chapter.
Figure 4.1 A 3D skin structure
Table 4.1 Parameters for a 3D skin structure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$ (J/g°C)</td>
<td>3.6</td>
</tr>
<tr>
<td>$C_2$ (J/g°C)</td>
<td>3.4</td>
</tr>
<tr>
<td>$C_3$ (J/g°C)</td>
<td>3.06</td>
</tr>
<tr>
<td>$Cb^1$ (J/g°C)</td>
<td>0</td>
</tr>
<tr>
<td>$Cb^2$ (J/g°C)</td>
<td>4.2</td>
</tr>
<tr>
<td>$Cb^3$ (J/g°C)</td>
<td>4.2</td>
</tr>
<tr>
<td>$C_b$ (J/m³K)</td>
<td>4.134</td>
</tr>
<tr>
<td>$K_1$ (W/cm°C)</td>
<td>0.0026</td>
</tr>
<tr>
<td>$K_2$ (W/cm°C)</td>
<td>0.0052</td>
</tr>
<tr>
<td>$K_3$ (W/cm°C)</td>
<td>0.0021</td>
</tr>
<tr>
<td>$Reff_1$</td>
<td>0.93</td>
</tr>
<tr>
<td>$Reff_2$</td>
<td>0.93</td>
</tr>
<tr>
<td>$Reff_3$</td>
<td>0.93</td>
</tr>
<tr>
<td>$Wb^1$ (g/cm³)</td>
<td>0</td>
</tr>
<tr>
<td>$Wb^2$ (g/cm³)</td>
<td>0.0005</td>
</tr>
<tr>
<td>$Wb^3$ (g/cm³)</td>
<td>0.0005</td>
</tr>
<tr>
<td>$\alpha$ (w/m²K)</td>
<td>2000</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>1</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>0.8</td>
</tr>
<tr>
<td>$\alpha_3$</td>
<td>0.4</td>
</tr>
<tr>
<td>$\rho_1$ (g/cm³)</td>
<td>1.2</td>
</tr>
<tr>
<td>$\rho_2$ (g/cm³)</td>
<td>1.2</td>
</tr>
<tr>
<td>$\rho_3$ (g/cm³)</td>
<td>1</td>
</tr>
<tr>
<td>$\sigma$ (cm)</td>
<td>0.1</td>
</tr>
<tr>
<td>$v$ (m/s)</td>
<td>0.08</td>
</tr>
</tbody>
</table>

Note: The data is coming from [Liu 1997].
Table 4.2 Pre-specified geometry parameters for the skin structure

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi</td>
<td>2</td>
</tr>
<tr>
<td>( R(cm) )</td>
<td>0.5</td>
</tr>
<tr>
<td>( L_\lambda(cm) )</td>
<td>0.008</td>
</tr>
<tr>
<td>( L_2(cm) )</td>
<td>0.02</td>
</tr>
<tr>
<td>( L_\lambda(cm) )</td>
<td>1</td>
</tr>
<tr>
<td>( \omega )</td>
<td>1</td>
</tr>
<tr>
<td>( S_p^{\text{Specified}} )</td>
<td>0.04</td>
</tr>
<tr>
<td>( \Delta r )</td>
<td>1/30</td>
</tr>
<tr>
<td>( \Delta t )</td>
<td>0.1</td>
</tr>
<tr>
<td>( \Delta z )</td>
<td>0.001</td>
</tr>
<tr>
<td>( \Delta \varphi )</td>
<td>( \frac{2\pi}{20} )</td>
</tr>
<tr>
<td>( N_1^x )</td>
<td>8</td>
</tr>
<tr>
<td>( N_2^x )</td>
<td>208</td>
</tr>
<tr>
<td>( N_2^z )</td>
<td>1208</td>
</tr>
<tr>
<td>( R_{\psi} )</td>
<td>2</td>
</tr>
<tr>
<td>( N_r )</td>
<td>30</td>
</tr>
<tr>
<td>( N_\varphi )</td>
<td>20</td>
</tr>
</tbody>
</table>
4.2 Calculation Results

4.2.1 Calculation Case 1

The elevated temperatures at the center and perimeter are pre-specified to be $8^\circ C$ and $2^\circ C$, respectively. In this case, the initial $P_0$ was determined to be $16.6159$ (W).

After $t = 410$ seconds, when the step one of the laser irradiation pattern (Figure 3.2) is completed, we recorded the data and plotted the elevated temperature profiles in Figures 4.2-4.6. Figure 4.2 and 4.3 show the elevated temperature profiles along the diameters on the skin surface with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 4.4 and 4.5 show the contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 4.6 shows the elevated temperature profile along the depth (the $z$-direction) at the center of the skin surface. It can be seen from these figures that the temperature at the center is close to $8^\circ C$. However, the temperature at the perimeter is much lower than the required temperature. A relative error $\sum \left[ \frac{\theta_{pre}^i - \theta_{cal}^i}{\theta_{pre}^i} \right]^2$, which shows the difference between the pre-specified temperatures and the calculated temperatures, is $0.513821$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4.2 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ seconds without any blood vessels.
Figure 4.3 The elevated temperature profiles along the diameter on the skin surface with 
\[ \phi = \frac{\pi}{2} \text{ and } \phi = \frac{3\pi}{2}, \] at \( t = 410 \) seconds without any blood vessels.
Figure 4.4 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ seconds without any blood vessels.
Figure 4.5 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 410$ seconds without any blood vessels.
Figure 4.6 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 410$ seconds without any blood vessels.
The record of computation shows that the laser was automatically shut off between $t = 410$ seconds and $t = 535$ seconds, and then on with a modified $P_0 = 16.47$ (W) until $t = 674$ seconds.

After $t = 674$ seconds, when the step three of the laser irradiation pattern (Figure 3.2) is completed, we recorded the data and plotted the elevated temperature profiles in Figures 4.7-4.11. Figures 4.7 and 4.8 show the elevated temperature profiles along the diameters on the skin surface with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 4.9 and 4.10 show the contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 4.11 displays the elevated temperature profile along the depth (the z-direction) at the center of the skin surface. It can be seen from these figures that both temperatures at the center and perimeter are close to the pre-specified temperatures. The relative error is reduced from 0.513821 to 0.0343988.
Figure 4.7 The elevated temperature profiles along the diameter on the skin surface with \( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 674 \) seconds without any blood vessels.
Figure 4.8 The elevated temperature profiles along the diameter on the skin surface with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 674$ seconds without any blood vessels.
Figure 4.9 The contours of the elevated temperature distributions in the cross section with \( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 674 \) seconds without any blood vessels.
Figure 4.10 The contours of the elevated temperature distributions in the cross section

with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 674$ seconds without any blood vessels
Figure 4.11 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 674$ seconds without any blood vessels.
4.2.2 Calculation Case 2

The elevated temperatures at the center and perimeter are 8°C and 3°C, respectively. The initial $P_0$ was 16.6159 (W). At $t = 410$ seconds, the temperature distribution is the same as those shown in Figure 4.2 to Figure 4.6. The relative error is 1.30985. The record of the computation shows that the laser was off between $t = 410$ seconds and $t = 675$ seconds, and on between $t = 675$ seconds and $t = 842$ seconds, and off again between $t = 842$ seconds and $t = 965$ seconds, and finally on with a modified $P_0 = 16.5661$ (W) between $t = 965$ seconds and $t = 1062$ seconds.

After $t = 1062$ seconds, when the step three of the laser irradiation pattern (Figure 3.2) is completed, we recorded the data and plotted the elevated temperature profiles in Figures 4.12-4.16. Figure 4.12 and 4.13 show the elevated temperature profiles along the diameters on the skin surface with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 4.14 and 4.15 show the contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 4.16 shows the elevated temperature profile along the depth (the z-direction) at the center of the skin surface. It can be seen from these figures that both temperatures at the center and perimeter are close to the pre-specified temperatures. The relative error is reduced from 1.30985 to 0.00471768.
Figure 4.12 The elevated temperature profiles along the diameter on the skin surface with 
\( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 1062 \) seconds without any blood vessels.
Figure 4.13 The elevated temperature profiles along the diameter on the skin surface with

\[ \varphi = \frac{\pi}{2} \text{ and } \varphi = \frac{3\pi}{2}, \text{ at } t = 1062 \text{ seconds without any blood vessels} \]
Figure 4.14 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 1062$ seconds without any blood vessels.
Figure 4.15 The contours of the elevated temperature distributions in the cross section with \( \varphi = \frac{\pi}{2} \) and \( \varphi = \frac{3\pi}{2} \), at \( t = 1062 \) seconds without any blood vessels.
Figure 4.16 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1062$ seconds without any blood vessels.
4.2.3 Calculation Case 3

The elevated temperatures at the center and perimeter are $8^\circ C$ and $4^\circ C$, respectively. The initial $P_0$ was 16.6159 (W). At $t = 410$ seconds, the temperature distribution is the same as those shown in Figure 4.2 to Figure 4.6. The relative error is 1.84527. In this case, the record shows that the laser was off in those periods between $t = 410$ seconds and $t = 542$ seconds, $t = 682$ seconds and $t = 865$ seconds, $t = 979$ seconds and $t = 1248$ seconds, and $t = 1350$ seconds and $t = 1468$ seconds, and on in the periods between $t = 542$ seconds and $t = 682$ seconds, $t = 865$ seconds and $t = 979$ seconds, and $t = 1248$ seconds and $t = 1350$ seconds. Finally, the laser was on with a modified $P_0 = 16.432$ (W) between $t = 1468$ seconds and $t = 1536$ seconds.

After $t = 1536$ seconds, when the step three of the laser irradiation pattern (Figure 3.2) is completed, we recorded the data and plotted the elevated temperature profiles in Figures 4.17-4.21. Figure 4.17 and 4.18 show the elevated temperature profiles along the diameters on the skin surface with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 4.19 and 4.20 show the contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 4.21 shows the elevated temperature profile along the depth (the $z$-direction) at the center of the skin surface. It can be seen from these figures that both temperatures at the center and perimeter are close to the pre-specified temperatures. The relative error is reduced from 1.84527 to 0.0231921.
Figure 4.17 The elevated temperature profiles along the diameter on the skin surface with \( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 1536 \) seconds without any blood vessels.
Figure 4.18 The elevated temperature profiles along the diameter on the skin surface with 
$\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1536$ seconds without any blood vessels
Figure 4.19 The contours of the elevated temperature distributions in the cross section with $\phi = 0$ and $\phi = \pi$, at $t = 1536$ seconds without any blood vessels.
Figure 4.20 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1536$ seconds without any blood vessels.
Figure 4.21 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1536$ seconds without any blood vessels.
4.3 Grid Independent

Finally, to test our algorithm to be independent of the grid size, there are three additional meshes \((r, \varphi, z)\) of \(20 \times 20 \times 1208\), \(50 \times 20 \times 1208\), \(30 \times 20 \times 2416\) are employed in the computation.

After \(t = 674\) seconds, when the step three of the laser irradiation pattern (Figure 3.2) for case one is completed, Figure 4.22 and 4.23 show the elevated temperature profiles along diameters on the skin surface with \(\varphi = 0\) and \(\varphi = \pi\) and with \(\varphi = \frac{\pi}{2}\) and \(\varphi = \frac{3\pi}{2}\), respectively. Figure 4.24 shows the elevated temperature profile along the depth (the \(z\)-direction) at the center of the skin surface.

It can be seen from Figures 4.22 to 4.24 that there are no significant differences among these solutions, implying that our scheme is grid independent.
Figure 4.22 Elevated temperature profiles along the diameter on the skin surface with \( \phi = 0 \) and \( \phi = \pi \), at \( t = 674 \) without any blood vessels.
Figure 4.23 Elevated temperature profiles along the diameter on the skin surface with 
\[ \phi = \frac{\pi}{2} \text{ and } \phi = \frac{3\pi}{2}, \] at \( t = 674 \) without any blood vessels.
Figure 4.24 Elevated temperature profiles along the depth (the $z$-direction) at the center of the skin surface, at $t = 674$ without any blood vessels
CHAPTER V

SKIN MODEL EMBEDDED WITH A BLOOD VESSEL

5.1 Introduction of the Skin Model Embedded With a Blood Vessel

In this chapter, we consider the skin structure embedded with a blood vessel, which is close to the realistic skin. Based on the histology knowledge [Ham 1965] [Gartner 2000], the largest arteries of the skin are arranged in the form of a flat network in the subcutaneous tissue, immediately below the dermis. This arterial network is called the rete cutaneum. It receives blood from branches of the larger arteries that run more deeply in the subcutaneous tissue. From the rete cutaneum, branches pass both inwardly and outwardly. Those branches that pass outwardly supply 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 [Ham 1965] [Gartner 2000]. Figure 5.1 shows the realistic skin structure configuration.
To simplify our computation, we consider the target region to be a cylindrical structure embedded with a blood vessel that crosses through the subcutaneous layer from the bottom to the top as shown in Figure 5.2. In this Figure, only the blood vessel in the subcutaneous is shown. Since there are only capillaries in the dermis layer, the contribution of these vessels to the heat transfer could be ignored [Zhou 2004].
Figure 5.2 Configuration of a 3D skin structure
Here, we employ the widely applied Pennes' equation for calculating the temperature distribution. However, it should be pointed out that our method is not limited to the Pennes' equation and could be replaced by other models. The temperature of blood in the cross section of the vessel is assumed to be uniform. The constraints of energy balance will lead to the following ordinary differential equation as follows [Majchrzak 1999]:

\[ C_b v F \frac{\partial(\theta_b)}{\partial z} - \alpha P(\theta_w - \theta_b) = 0, \]  

where \( C_b \) is the specific heat of blood, \( v \) the velocity of the blood, \( F \) the vessel lateral section, and \( \alpha, P \) the heat transfer coefficients between blood and tissue, and vessel perimeter respectively. Further, \( \theta_w \) is the vessel periphery elevated temperature and \( \theta_b \) is the elevated blood temperature above the ambient temperature. The Pennes' equation that describes the thermal behavior of the triple-layered skin structure when irradiated by the laser can be expressed in cylindrical coordinates as follows [Pennes 1948]:

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

where \( \theta_l \) is the elevated tissue due to heating by a laser; \( \rho_l, C_l \) 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 \( Q_r^l \) is volumetric heat due to spatial heating. Here, we assume that the laser power is continuous and spatial with a normal distribution. As such, the heat source \( Q_r^l \) can be written as follows, which is the same as that in the previous chapter.
\[ Q_1 = \alpha_1 e^{-\sigma z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r \cos \phi - x_0)^2 + (r \sin \phi - y_0)^2}{2\sigma^2}} P_0 (1 - \text{Reff}_1) f(t), \]
\[ Q_2 = \alpha_2 e^{-\sigma_1 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r \cos \phi - x_0)^2 + (r \sin \phi - y_0)^2}{2\sigma^2}} P_0 (1 - \text{Reff}_2) f(t), \]
\[ Q_3 = \alpha_3 e^{-\sigma_2 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r \cos \phi - x_0)^2 + (r \sin \phi - y_0)^2}{2\sigma^2}} P_0 (1 - \text{Reff}_3) f(t), \]

(5.3)

Similar to those in the previous chapter, the interfacial and boundary conditions in the tissue are assumed to be as follows:

\[ \frac{\partial \theta}{\partial z} = 0, \quad z = 0, \]  
(5.4)

\[ \theta_1 = \theta_2, \quad k_1 \frac{\partial \theta_1}{\partial z} = k_2 \frac{\partial \theta_2}{\partial z}, \quad z = L_1, \]  
(5.5)

\[ \theta_2 = \theta_3, \quad k_2 \frac{\partial \theta_2}{\partial z} = k_3 \frac{\partial \theta_3}{\partial z}, \quad z = L_1 + L_2, \]  
(5.6)

\[ \frac{\partial \theta}{\partial z} = 0, \quad z = L_1 + L_2 + L_3. \]  
(5.7)

On the other hand, the boundary conditions in the blood vessel are assume to be

\[ \theta_2 = \theta_3, \quad z = L_1 + L_2, \]  
(5.8)

\[ \theta_3 = \theta_3, \quad z = L_1 + L_2 + L_3, \]  
(5.9)

Furthermore, the boundary condition between the lateral blood vessel and the tissue is assumed to be [Huang 1994]

\[ \frac{\partial \theta}{\partial r} = Bi(\theta_w - \theta_b). \]  
(5.10)

The boundary conditions at the lateral wall of the tissue are
\[ \frac{\partial \theta_i}{\partial r} = 0, \quad r = R, \]  
(5.11)

and

\[ \theta_i(r, \varphi, z) = \theta_i(r, \varphi + 2m\pi, z), \]  
(5.12)

where \( m \) is an integer.

The initial condition is

\[ \theta_i = 0, \quad t = 0, \quad l = 1, 2, 3. \]  
(5.13)

### 5.2 Numerical Method of the Skin Model Embedded With a Blood Vessel

To obtain a temperature distribution numerically, we first let \( u_i )_{jk}^n \) and \( u_b \) be the numerical approximation of \( (\theta_i)(i\Delta r, j\Delta \varphi, k\Delta z, n\Delta t) \) and \( \theta_b \), and \( \Delta r, \Delta \varphi, \Delta z \), and \( \Delta t \) be the spatial and temporal mesh sizes, respectively. Here, \( i, j, k \) are chosen to be \( 1 \leq i \leq N_r \), \( 1 \leq j \leq N_\varphi \), \( 1 \leq k \leq N_z^i \), so that \( N_r \cdot \Delta r = R \), \( N_\varphi \cdot \Delta \varphi = 2\pi \) and \( N_z^i \cdot \Delta z = L_z, \ l = 1, 2, 3 \). We also let \( R_{hp} \) be the number of grid points along the radius direction in the blood vessel, which is embedded only in the subcutaneous layer. Eq. (5.1) can be solved by using the fourth-order Runge-Kutta method [Burden 1993]. Eqs. (5.2)-(5.13), can be discretized as follows:

\[
\begin{align*}
\rho_i C_i \frac{(u_i)_{jk}^{n+1} - (u_i)_{jk}^n}{\Delta t} + \frac{W_b^l C_b^l}{2} \left[ (u_i)_{jk}^{n+1} + (u_i)_{jk}^n - 2(u_b)_{jk} \right] \\
- k_i (P_r^2 + \delta_\varphi^2 + \delta_z^2) \frac{(u_i)_{jk}^{n+1} + (u_i)_{jk}^n}{2} = (Q_{i,jk}^l)^{n+0.5}, \quad l = 1, 2, 3.
\end{align*}
\]  
(5.14)
where \( P^2_{ij} u_{ijk} = \frac{r_{ij} (u_{i+1,j,k} - u_{ijk}) - r_{i-1,j,k} (u_{i-1,j,k} - u_{ijk})}{r_i^2 \Delta r^2} \), \( \delta^2 u_{ijk} = \frac{u_{ijk} - 2u_{ijk} + u_{i+1,jk}}{r_i^2 \Delta r^2} \), \( \delta^2 u_{ijk} = \frac{u_{ijk} - 2u_{ijk} + u_{ijk-1}}{\Delta z^2} \), and \( r_{i+1/2} = (i + \frac{1}{2}) \Delta r \). The discrete interfacial equations are assumed to be, for any time level,

\[
\left( \frac{u_t}{\Delta t} \right)_{i,j,N_z} = k_1 \frac{(u_t)_{i,j,N_z-1} - (u_t)_{i,j,N_z}}{\Delta z} = k_2 \frac{(u_t)_{i,j+N_z} - (u_t)_{i,j}}{\Delta z}, \quad (u_t)_{i,j}^n = (u_t)_{i,j}^{n-1}, \tag{5.15}
\]

and when the grid point \((i,j)\) is in the tissue

\[
k_2 \frac{(u_t)_{i,j,N_z}^n - (u_t)_{i,j,N_z-1}^n}{\Delta z} = k_3 \frac{(u_t)_{i,j+N_z} - (u_t)_{i,j}}{\Delta z}, \quad (u_t)_{i,j}^n = (u_t)_{i,j}^{n-1}, \tag{5.16}
\]

when the grid point \((i,j)\) is in the blood vessel,

\[(u_t)_{i,j}^n = (u_t)_{i,j}^{n-1}. \tag{5.17}\]

The boundary condition Eq. (10), between the tissue and the lateral blood vessel are descritized as follows:

\[(u_t)_{i,j}^{n+1} = ((u_t)_{i+1,j}^{n+1} + Bi \cdot \Delta r \cdot (u_t)_{i-1,j}^{n+1})/(1 + Bi \cdot \Delta r), \tag{5.18}\]

where \( i = R_{bv} \).

The initial and other boundary conditions are discretized as follows:

\[(u_t)_{i,j}^0 = 0, \tag{5.19}\]

\[(u_t)_{i,j}^n = (u_t)_{i,j}^{n-1}, \tag{5.20}\]

if \( i > R_{bv}, \quad (u_t)_{i,j}^n = (u_t)_{i,j}^{n-1}, \tag{5.21}\]

if \( i \leq R_{bv}, \quad (u_t)_{i,j}^n = u_b, \tag{5.22}\]

\[(u_t)_{i,j}^n = (u_t)_{i,j-1}^n. \tag{5.23}\]
\[(u_t)_{jk}^n = (u_t)_{jk}^{n,m} + \Delta u_{jk}^n, \tag{5.24}\]

for any time level \(n\).

Figure 5.3 gives the flow diagram or iteration scheme used in the numerical calculations in order to obtain an optimal temperature distribution in the irradiated region. This was necessary since it is difficult to determine the laser power by minimizing the sum of square between calculated (from the bioheat equation) and pre-specified temperatures at all \(M + 1\) points (center and perimeter) in one step. In step 1, Eq. (5.2) is calculated based on the result obtained by Eq. (5.1) which is solved first in each time loop. Also initial laser power \(P_0\) is determined at the center point only so that the pre-specified temperature \(\theta_{\text{center}}^{\text{pre}}\) is satisfied. It should be pointed out that in step 1, we calculate \(\theta_t\) from Eqs. (5.1)-(5.14) starting at a guessed \(\theta_w\). Least squares method is applied to determine the Laser Power \(P_0\). In step 2, the laser is focused on the center of the skin surface and is turned off in order to allow heat to diffuse from the center towards the perimeter of the region. This allows the perimeter temperature to increase and leads to a decrease in the temperature at the center. Based on certain criteria involving comparisons between calculated temperatures at the center \((u_{\text{cal}}^{\text{center}})\) and perimeter \((u_{\text{cal}}^{\text{primeter}})\) and pre-specified temperature \((\theta_{\text{center}}^{\text{cal}}, \theta_{\text{primeter}}^{\text{cal}})\), the laser may need to be turned on and off until \(S_p(P_0)\) is less than a pre-specified value, \(S_p^{\text{Specified}}\). At this point, the computation goes to step 3. The laser power \(P_0\) is optimized based on the least squares method, Eqs. (5.25)-(5.29), and the calculated temperature distribution, Eqs. (5.14)-(5.24), involving the \(M + 1\) points in Eq.(5.25).
Step 1
Obtain the temperature at the center point by Runge-Kutta method and Eqs. (14)-(24).

Step 2
Shut off the laser, let heat conduct to the boundary.

Step 3
Adjust $P_0$ by the least square method, Eqs. (25)-(29).

$S_p(P_0) < S_p^{\text{specified}}$

Figure 5.3 Streamline of the computation
5.3 Result and Discussion of the Skin Model
Embedded With a Blood Vessel

We tested our algorithm in a 3D skin structure as shown in Figure 5.2, where the parameter values were chosen from Table 4.1 and the size of skin structure is given in Table 4.2. A mesh of $30 \times 20 \times 1208$ in $(r, \varphi, z)$ was employed in the computation. In our calculation, we pre-specified the elevated temperatures at the center of the skin surface and at four locations, 90 degree apart (that is, $\varphi = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$) at the perimeter. The elevated blood temperature at bottom of the targeted region, $z = L_1 + L_2 + L_3$, is assumed to $1^\circ C$.

A laser exposure pattern was designed as follows: 21 pixels on the skin surface were chosen for laser irradiation. These pixels included the center pixel and those grid points (20 points) with a $\Delta r$ distance from the center. The laser was set to irradiate at the center pixel for 10 seconds, after which it was moved to the grid point at $\varphi = 0$ and circulated twice counter-clockwise over the 20 pixels with the laser focused for 10 seconds on each pixel. Three different cases were tested.
5.3.1 Calculation Case 1

The elevated temperatures at the center and perimeter of the skin surface were pre-specified to be 8°C and 2°C, respectively. In this case, the initial $P_0$ was determined to be 17.4119 (W).

After $t = 410$ seconds, when the step one of the streamline of the computation (Figure 5.3) is completed, we recorded the data and plotted the elevated temperature profiles in Figures 5.4-5.8. Figure 5.4 and 5.5 show the elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 5.6 and 5.7 show the contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 5.8 shows the elevated temperature profile along the depth (the z-direction) at the center of the skin surface. It can be seen from these figures that the temperature at the center is close to 8°C. However, the temperature at the perimeter is much lower than the required temperature. A relative error $\frac{1}{l=1} \sum [\frac{\theta^i_{pre} - u^i_{col}}{\theta^i_{pre}}]^2$, which shows the difference between the pre-specified temperature and the calculated temperature, is 0.505088.
Figure 5.4 The elevated temperature profiles along the diameter on the skin surface with \( \phi = 0 \) and \( \phi = \pi \), at \( t = 410 \) seconds with a blood vessel.
Figure 5.5 The elevated temperature profiles along the diameter on the skin surface with
\( \varphi = \frac{\pi}{2} \) and \( \varphi = \frac{3\pi}{2} \), at \( t = 410 \) seconds with a blood vessel.
Figure 5.6 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ seconds with a blood vessel.
Figure 5.7 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 410$ seconds with a blood vessel.
Figure 5.8 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 410$ seconds with a blood vessel
The computation shows that the laser was automatically shut off between $t = 410$ seconds and $t = 529$ seconds, and then on with a modified $P_b = 17.2661$ (W) until $t = 672$ seconds.

After $t = 672$ seconds, when the step three of the streamline of the computation (Figure 5.3) is completed, we recorded the data and plotted the elevated temperature profiles in Figures 5.9-5.13. Figures 5.9 and 5.10 show the elevated temperature profiles along the diameters on the skin surface with $\phi = 0$ and $\phi = \pi$ and with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, respectively. Figure 5.11 and 5.12 show the contours of the elevated temperature distributions in the cross section with $\phi = 0$ and $\phi = \pi$ and with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, respectively. Figure 5.13 displays the elevated temperature profile along the depth (the $z$-direction) at the center of the skin surface. It can be seen from these figures that both temperatures at the center and perimeter are close to the pre-specified temperatures. The relative error is reduced from $0.505088$ to $0.0360016$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.9 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 672$ seconds with a blood vessel.
Figure 5.10 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 672$ seconds with a blood vessel.
Figure 5.11 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 672$ seconds with a blood vessel.
Figure 5.12 The contours of the elevated temperature distributions in the cross section with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, at $t = 672$ seconds with a blood vessel.
Figure 5.13 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 672$ seconds with a blood vessel
5.3.2 Calculation Case 2

The elevated temperatures at the center and perimeter are 8°C and 3°C, respectively. The initial $P_0$ was 17.4119 (W). At $t = 410$ seconds, the temperature distribution is the same as those shown in Figure 5.4 to Figure 5.8. The relative error is 1.30051. The record of the computation shows that the laser was off between $t = 410$ seconds and $t = 632$ seconds, and on between $t = 633$ seconds and $t = 798$ seconds, and off again between $t = 799$ seconds and $t = 936$ seconds, and finally on with a modified $P_0 = 17.3435$ (W) between $t = 937$ seconds and $t = 1044$ seconds.

After $t = 1044$ seconds, when the step three of the streamline of the computation (Figure 5.3) is completed, we recorded the data and plotted the elevated temperature profiles in Figures 5.14-5.18. Figure 5.14 and 5.15 show the elevated temperature profiles along the diameters on the skin surface with $\phi = 0$ and $\phi = \pi$ and with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, respectively. Figure 5.16 and 5.17 show the contours of the elevated temperature distributions in the cross section with $\phi = 0$ and $\phi = \pi$ and with $\phi = \frac{\pi}{2}$ and $\phi = \frac{3\pi}{2}$, respectively. Figure 5.18 shows the elevated temperature profile along the depth (the z-direction) at the center of the skin surface. It can be seen from these figures that both temperatures at the center and perimeter are close to the pre-specified temperatures. The relative error is reduced from 1.30051 to 0.00588037.
Figure 5.14 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 1044$ seconds with a blood vessel.
Figure 5.15 The elevated temperature profiles along the diameter on the skin surface with

\[ \varphi = \frac{\pi}{2} \text{ and } \varphi = \frac{3\pi}{2} \], at \( t = 1044 \) seconds with a blood vessel

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.16 The contours of the elevated temperature distributions in the cross section with \( \varphi = 0 \) and \( \varphi = \pi \), at \( t = 1044 \) seconds with a blood vessel
Figure 5.17 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1044$ seconds with a blood vessel.
Figure 5.18 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1044$ seconds with a blood vessel.
5.3.3 Calculation Case 3

The elevated temperatures at the center and perimeter are 8°C and 4°C, respectively. The initial $P_0$ was 17.4119 (W). At $t = 410$ seconds, the temperature distribution is the same as those shown in Figure 4.2 to Figure 4.6. The relative error is 1.83683. In this case, the record shows that the laser was off in those periods between $t = 410$ seconds and $t = 525$ seconds, $t = 666$ seconds and $t = 815$ seconds, $t = 930$ seconds and $t = 1127$ seconds. It was on in the periods between $t = 526$ seconds and $t = 665$ seconds, $t = 816$ seconds and $t = 929$ seconds, and $t = 1128$ seconds and $t = 1228$ seconds. Finally, the laser was on with a modified $P_0 = 17.316$ (W) between $t = 1423$ seconds and $t = 1509$ seconds.

After $t = 1509$ seconds, when the step three of the streamline of the computation (Figure 5.3) is completed, we recorded the data and plotted the elevated temperature profiles in Figures 5.19-5.23. Figure 5.19 and 5.20 show the elevated temperature profiles along the diameters on the skin surface with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 5.21 and 5.22 show the contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 5.23 shows the elevated temperature profile along the depth (the z-direction) at the center of the skin surface. It can be seen from these figures that both temperatures at the center and perimeter are close to the pre-specified temperatures. The relative error is reduced from 1.83683 to 0.0298973.
Figure 5.19 The elevated temperature profiles along the diameter on the skin surface with $\phi = 0$ and $\phi = \pi$, at $t = 1509$ seconds with a blood vessel.
Figure 5.20 The elevated temperature profiles along the diameter on the skin surface with \( \rho = \frac{\pi}{2} \) and \( \varphi = \frac{3\pi}{2} \), at \( t = 1509 \) seconds with a blood vessel.
Figure 5.21 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 1509$ seconds with a blood vessel.
Figure 5.22 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1509$ seconds with a blood vessel.
Figure 5.23 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1509$ seconds with a blood vessel.
5.4 Grid Independent Experiment

Finally, to test our algorithm to be independent of the grid size, there are two additional meshes \((r, \varphi, z)\) of \(60 \times 20 \times 1208\), \(30 \times 20 \times 2416\) are employed in the computation.

After \(t = 410\) seconds, when the step one of the streamline of the computation (Figure 5.3) is completed Figure 5.24 and 5.25 show the elevated temperature profiles along diameters on the skin surface with \(\varphi = 0\) and \(\varphi = \pi\) and with \(\varphi = \frac{\pi}{2}\) and \(\varphi = \frac{3\pi}{2}\), respectively. Figure 5.26 shows the elevated temperature profile along the depth (the \(z\)-direction) at the center of the skin surface.

It can be seen from Figures 5.24 to 5.26 that there are no significant differences among these solutions, implying that our scheme is grid independent.
Figure 5.24 Elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ with a blood vessel.
Figure 5.25 Elevated temperature profiles along the diameter on the skin surface with 

\[ \varphi = \frac{\pi}{2} \text{ and } \varphi = \frac{3\pi}{2}, \text{ at } t = 410 \text{ with a blood vessel} \]
Figure 5.26 Elevated temperature profiles along the depth (the z-direction) at the center of the skin surface, at $t = 410$ with a blood vessel
CHAPTER VI

CONCLUSION AND FUTURE WORK

Numerical Scheme for solving a 3D Pennes’ bioheat transfer equation in the 3D triple-layered cylindrical skin structure is shown to be unconditionally stable with respect to the heat source. To calculate the optimal temperature distribution in the entire treatment region, two algorithms are obtained in this research. One is for the 3D triple-layered cylindrical skin structure without any blood vessels. The other is for the 3D triple-layered cylindrical skin structure with a blood vessel. We also have tested these two developed algorithms in the 3D triple-layered cylindrical skin structure without any blood vessels and the 3D triple-layered cylindrical skin structure with a blood vessel, respectively. Numerical results show that the method is efficient. It can be used for certain types of cancer treatments, such as skin cancer.

Further research should be focused on the development of numerical methods for obtaining an optimal treatment, while the skin structure embedded with multi-level blood vessels. It is closer to the realistic condition, based on the histology knowledge, as shown in Figure 6.1.
Figure 6.1 Skin structure embedded with multi-level blood vessels
APPENDIX A

SOURCE CODE FOR SOLVING THE 3D SKIN STRUCTURE WITHOUT ANY BLOOD VESSELS
Table A.1 Program 1: Source code of step 1 in Figure 3.2 is used for the skin model without any blood vessels.

Le Zhang
4/11/05

This program is about heat transfer in the skin of a human being. There are three layers in the skin. The first layer is epidermis, the second one is dermis and the last one is Subcutaneous.

/*
#include <fstream.h>
#include <string.h>
#include <iostream.h>
#include <math.h>
#include<stdio.h>
#define NZ1 8
#define NZ2 208
#define NZ3 1208
#define NPhi 20
class zlihcp
|
private:
double ***Q1,***Q2,***Q3;
double ***v,***vnew,***vold,***vn;
double ***beta,***f,***d;
double *b,***a,***c;
double MaxErr,h,e;
double DeltaZ,deltaT,deltaPhi,deltaR;
double Rchange,Phichange,Zchange,Rchangel,Phichangel,Zchangel;
double p1,p2,p3,q1,q2,q3,k1,k2,k3,wb1,wb2,wb3,cb1,cb2,cb3;
double Sigma,Alpha1,Alpha2,Alpha3,Refl1,Refl2,Refl3;
double P0,pi,judge,CenterX,CenterY;
int i,j,k,n,t,nz1,nz2,nz3,nz4,nz5,nz6,nz7,nz8,nz9,nz10,nz11,nz12,nz13,nz14,nz15,nz16,nz17,nz18,nz19,nz20;
int MaxLen,MaxWid,MaxHig;
public:
zlihcp(int i, int j, int k)
|
{ int i,j,k;
MaxLen = 1;
MaxWid = j;
MaxHig = k;
a = new double *[i];
for(j=0;j<i;j++)
a[j] = new double *[j];
b = new double *[j];
c = new double *[j];
Q1 = new double **[i];
Q2 = new double **[j];
Q3 = new double **[k];
v = new double **[j];
vnew = new double **[j];
vold = new double **[j];
vn = new double **[j];
beta = new double **[j];
f = new double **[j];
d = new double **[j];
for (j=0;j<i;j++)
{| Q1[j][j] = new double *[j];
Q2[j][j] = new double *[j];
Q3[j][j] = new double *[j];
v[j] = new double *[j];
vnew[j] = new double *[j];
vold[j] = new double *[j];
vn[j] = new double *[j];
beta[j] = new double *[j];
f[j] = new double *[j];
d[j] = new double *[j];
for (k=0;k<j;k++)
{| Q1[j][j][k] = new double [j];
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Q2[][k] = new double [high1];
Q3[][k] = new double [high1];
v[][k] = new double [high1];
vn[][k] = new double [high1];
vold[][k] = new double [high1];
vn[][k] = new double [high1];
beta[][k] = new double [high1];
f[][] = new double [high1];
d[][] = new double [high1];
}
}
for(i=0;i<l;i++)
{
for(j=0;j<w;j++)
{
for(k=0;k<high1;k++)
{
Q[][i][j][k]=0;
Q[][i][j][k]=0;
Q[][i][j][k]=0;
v[][i][j][k]=0;
vn[][i][j][k]=0;
vold[][i][j][k]=0;
vn[][i][j][k]=0;
vn[][i][j][k]=0;
vn[][i][j][k]=0;
vn[][i][j][k]=0;
}
}
Sigma= 0.1; Alpha1=1.0; Alpha2=0.8;
Alpha3=0.4; Ref1=0.93; Ref2=0.93;
Ref3=0.93; p=3.14159265358979;
CenterX = 0; CenterY = 0;
t=2; n=10; p1=1.2; p2=1.2; p3=1.0;
qcl=3.6; qc2=3.4; qc3=3.06;
k1=0.0026; k2=0.0052; k3=0.0021;
w1=0.0; w2=0.0005; w3=0.0005;
ch1=0.0; ch2=4.2; ch3=4.2; e=0.001;
deltaPhi = double(2*pi/(double)NPhi);
deltaR = double(0.5/(double)NR);
deltaZ=0.001; deltaT=0.1;

~zlihcp()
{
 delete []a;
delete []b;
delete []c;
delete []Q1;
delete []Q2;
delete []Q3;
delete []v;
delete []vn;
delete []vold;
delete []vn;
delete []beta;
delete []f;
delete []d;
};
void InitQ(double,double,double);
double IntrnTrsy(double);
double RunAll (double);
void FileWrit(int);
void Clear(void);
};

void zlihcp::FileWrit(int time)
{
 int i;k;


ofstream fout1, fout2, fout21, fout3, fout31;
sprintf(str, "%d", timel);
strcat(strl, str); strcat(str2, str); strcat(str21, str);
strcat(str3, str); strcat(str31, str);

fout1.open(str, ios::out);
ofout1 << "TITLE = Y'Example: Simple ZT-Volume DataY' " << endl;
ofout1 << "VARIABLES = Y'Z', Y'TemperatureY' " << endl;
ofout1 << "ZONE T=1209, F=POINTY" << endl;
for(k=0; k<NZ3; k++)
    fout1 << double(k*deltaZ) << "<" new[0][0][k] << endl;
ofout1.close();

fout2.open(strl, ios::out);
ofout2 << "TITLE = Y'Example: Simple ZT-Volume DataY' " << endl;
ofout2 << "VARIABLES = Y'R', Y'Z', Y'TemperatureY' " << endl;
ofout2 << "ZONE T=" right", J=1209, J=31, F=POINTY" << endl;
for(i=0; i<NR; i++)
{
    for(k=0; k<NZ3; k++)
    {
        fout2 << double(i*deltaR) << "<" double(k*deltaZ) << "<" new[i][0][k] << endl;
    }
}
ofout2.close();

fout3.open(strl, ios::out);
ofout3 << "TITLE = Y'Example: Simple ZT-Volume DataY' " << endl;
ofout3 << "VARIABLES = Y'R', Y'Z', Y'TemperatureY' " << endl;
ofout3 << "ZONE T=" right", J=1209, J=31, F=POINTY" << endl;
for(i=0; i<NR; i++)
{
    for(k=0; k<NZ3; k++)
    {
        fout3 << double(i*deltaR) << "<" double(k*deltaZ) << "<" new[i][NPhi/4][k] << endl;
    }
}
ofout3.close();

fout21.open(strl, ios::out);
ofout21 << "TITLE = Y'Example: Simple ZT-Volume DataY' " << endl;
ofout21 << "VARIABLES = Y'R', Y'Z', Y'TemperatureY' " << endl;
ofout21 << "ZONE T=" reverse", J=1209, J=31, F=POINTY" << endl;
for(i=0; i<NR; i++)
{
    for(k=0; k<NZ3; k++)
    {
        fout21 << double(-i*deltaR) << "<" double(k*deltaZ) << "<" new[i][3*NPhi/4][k] << endl;
    }
}
ofout21.close();

fout31.open(strl, ios::out);
ofout31 << "TITLE = Y'Example: Simple ZT-Volume DataY' " << endl;
ofout31 << "VARIABLES = Y'R', Y'Z', Y'TemperatureY' " << endl;
ofout31 << "ZONE T=" reverse", J=1209, J=31, F=POINTY" << endl;
for(i=0; i<NR; i++)
{
    for(k=0; k<NZ3; k++)
    {
        fout31 << double(-i*deltaR) << "<" double(k*deltaZ) << "<" new[i][NPhi/2][0] << endl;
    }
}
ofout31.close();
fout31.open(str31,ios::out);
fout31<<"TITLE = Example: Simple 2D-Volume Data" "<<endl;
fout31<<"VARIABLES = "R", "Temperature" "<<endl;
fout31<<"ZONE T="right",I=31,F=POINT"<<endl;
for(i=0;i<=NR;i++)
{
    fout31<<double(i*deltaR)<<" "<<new[0][0][0]"<<endl;
}
fout31<<"TITLE = Example: Simple 2D-Volume Data" "<<endl;
fout31<<"VARIABLES = "R", "Temperature" "<<endl;
fout31<<"ZONE T="reverse",I=31,F=POINT"<<endl;
for(i=0;i<=NR;i++)
    fout31<<double(i*deltaR)<<" "<<new[0][NPhi/4][0]"<<endl;
fout31.close();

void zlihcp::InitQ(double P0,double Cr,double Cp) // Initialize the laser power;
{
    int i,j,z;
    CenterX = Cr * cos(Cp);
    CenterY = Cr * sin(Cp);
    for(i=0;i<=NR;i++)
    {
        for(j=0;j<=NPhi;j++)
            for(z=0;z<=NZ1;z++)
            {
                Q1[i][j][z]= Alpha1*exp(-Alpha1*z*deltaZ)/(sqrt(2*pi)*Sigma)*exp(-pow((i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*P0*(1-Refl);
            }
        for(z=NZ1+1;z<=NZ2;z++)
            {
                Q2[i][j][z]= Alpha2*exp(-Alpha2*(z-NZ1)*deltaZ)*exp(-Alpha1*deltaZ*NZ1)/(sqrt(2*pi)*Sigma)*exp(-pow((i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*P0*(1-Refl2);
            }
        for(z=NZ2+1;z<=NZ3;z++)
            {
                Q3[i][j][z]= Alpha3*exp(-Alpha3*(z-NZ2)*deltaZ)*exp(-Alpha1*deltaZ*NZ2)/(sqrt(2*pi)*Sigma)*exp(-pow((i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*P0*(1-Refl3);
            }
    }
}

double zlihcp::IntmTrsy(double P0) // Time Iteration and Tri-diagonal syste
{
    MaxErr=1.0; nt=0;
    while(1+nt<=(NPhi+1)*20-10)
    {
        if(nt == 1)
            InitQ(P0,0,0);
        t=0;
        for(i=0;i<=NR;i++)
        {
            for(j=0;j<=NPhi;j++)
                for(z=0;z<=NZ3;z++)
                {
                    vn[i][j][z]=old[i][j][z];
                }
        }
        cout<<nt<<"new cycle"<<endl;
        MaxErr=1.0;
        while(MaxErr>e)
        {
            MaxErr=0.0;
for (i=1; i<=NR-1; i++)
{
    for (j=1; j<=NPhi; j++)
    {  ///////////The first layer///////////
        for (z=1; z<=NZ1-1; z++)
        {  ////////////////The first layer////////////////////////
            R change = k1*deltaT*(((i+0.5)*vold[i+1][j][z-2]*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/deltaR*deltaR);
                if (j==NPhi)
                {
                    Phichange = k1*deltaT*vold[i][j][z-2]*vold[i][j][z]+vold[i][j-1][z]]/pow(*deltaR*deltaPhi,2);
                }
                else
                {
                    Phichange = k1*deltaT*vold[i][j+1][z-2]*vold[i][j][z]+vold[i][j-1][z]]/pow(*deltaR*deltaPhi,2);
                }
            Z change = k1*deltaT*(((i+0.5)*vold[i][j][z+1-2]*vold[i][j][z]+vold[i][j-1][z]]/deltaZ);
            ///////////////The n state ///////////////

            R changel = k1*deltaT*(((i+0.5)*vold[i+1][j][z]+(i-0.5)*vold[i-1][j][z]))/(deltaR*deltaR);
                if (j==NPhi)
                {
                    Phichangel = k1*deltaT*vold[i][j][z-2]*vold[i][j][z]+vold[i][j-1][z]]/pow(*deltaR*deltaPhi,2);
                }
                else
                {
                    Phichangel = k1*deltaT*vold[i][j+1][z-2]*vold[i][j][z]+vold[i][j-1][z]]/pow(*deltaR*deltaPhi,2);
                }
            Z changel = k1*deltaT*(((i+0.5)*vold[i][j][z+2]*vold[i][j][z]+vold[i][j-1][z]])/(i*deltaR*deltaR);
                if (j==NPhi)
                {
                    Phichange1 = k1*deltaT*(((i+0.5)*vold[i][j][z-2]*vold[i][j][z]+vold[i][j-1][z]])/pow(*deltaR*deltaPhi,2);
                }
                else
                {
                    Phichange1 = k1*deltaT*(((i+0.5)*vold[i][j][z]+(i-0.5)*vold[i-1][j][z]))/(i*deltaR*deltaR);
                }
            ///////////Prepare the coefficient for Thomas way\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\n
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
110

d[i][j][z]=R change+Phichange+Z change+2*deltaT*Q 1[i][j][z]+(2*p2*qc2-wb2*cb2*deltaT)*vn[i][j][z];
b[z]=k2*deltaT/(deltaZ*deltaZ);
a[i][j][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT *((i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
+c[z]=k2*deltaT/(deltaZ*deltaZ);
d[i][j][z]=R change+Phichange+
k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
+f[i][j][z];

b[NZ2]=k2;
a[i][NZ2]=k2+k3;
c[NZ2]=k3;
d[i][j][NZ2]=0;


for(z= NZ2+1;z<=NZ3-1;z++)

R change = k3*deltaT*((i+0.5)*vold[i+l][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-l][j][z])/(i*deltaR*deltaR);
if(j==NPhi)
Phichange = k3*deltaT*((i+0.5)*vn[i+l][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-l][j][z])/(i*deltaR*deltaPhi,2);
else
Phichange = k3*deltaT*((i+0.5)*vn[i+l][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-l][j][z])/(i*deltaR*deltaPhi,2);

Z change = k3*deltaT*((i+0.5)*vn[i][j+1][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i][j-1][z])/(i*deltaR*deltaR);

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\n
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
judge=(vnew[i][j][z]-vold[i][j][z]);
if(judge>0)
judge=max(judge*(1);
if(judge>MaxErr)
MaxErr=judge;
void[i][j][z]=vnew[i][j][z];
}
}
}
for(i=0;i<=NR;i++)
{
for(j=0;j<=NPhi;j++)
{
for(z=0;z<=NZ3;z++)
{
vnew[i][j][0]=vnew[i][j][1];
vnew[i][j][NZ3]=vnew[i][j][NZ3-1];
vnew[i][j][z]=vnew[i][NPhi][j][z];
vnew[0][j][z]=vnew[i][j][z];
vnew[NR][j][z]=vnew[NR-1][j][z];
vold[i][j][z]=vnew[i][j][z];
}
}
//File write
if(nt==10)InitQ(P0,deltaR,1*deltaPhi);
if(nt==20)InitQ(P0,deltaR,2*deltaPhi);
if(nt==30)InitQ(P0,deltaR,3*deltaPhi);
if(nt==40)InitQ(P0,deltaR,4*deltaPhi);
if(nt==50)InitQ(P0,deltaR,5*deltaPhi);
if(nt==60)InitQ(P0,deltaR,6*deltaPhi);
if(nt==70)InitQ(P0,deltaR,7*deltaPhi);
if(nt==80)InitQ(P0,deltaR,8*deltaPhi);
if(nt==90)InitQ(P0,deltaR,9*deltaPhi);
if(nt==100)InitQ(P0,deltaR,10*deltaPhi);
if(nt==110)InitQ(P0,deltaR,11*deltaPhi);
if(nt==120)InitQ(P0,deltaR,12*deltaPhi);
if(nt==130)InitQ(P0,deltaR,13*deltaPhi);
if(nt==140)InitQ(P0,deltaR,14*deltaPhi);
if(nt==150)InitQ(P0,deltaR,15*deltaPhi);
if(nt==160)InitQ(P0,deltaR,16*deltaPhi);
if(nt==170)InitQ(P0,deltaR,17*deltaPhi);
if(nt==180)InitQ(P0,deltaR,18*deltaPhi);
if(nt==190)InitQ(P0,deltaR,19*deltaPhi);
if(nt==200)InitQ(P0,deltaR,20*deltaPhi);
if(nt==210)InitQ(P0,0,0);
if(nt==220)InitQ(P0,deltaR,1*deltaPhi);
if(nt==230)InitQ(P0,deltaR,2*deltaPhi);
if(nt==240)InitQ(P0,deltaR,3*deltaPhi);
if(nt==250)InitQ(P0,deltaR,4*deltaPhi);
if(nt==260)InitQ(P0,deltaR,5*deltaPhi);
if(nt==270)InitQ(P0,deltaR,6*deltaPhi);
if(nt==280)InitQ(P0,deltaR,7*deltaPhi);
if(nt==290)InitQ(P0,deltaR,8*deltaPhi);
if(nt==300)InitQ(P0,deltaR,9*deltaPhi);
if(nt==310)InitQ(P0,deltaR,10*deltaPhi);
if(nt==320)InitQ(P0,deltaR,11*deltaPhi);
if(nt==330)InitQ(P0,deltaR,12*deltaPhi);
if(nt==340)InitQ(P0,deltaR,13*deltaPhi);
if(nt==350)InitQ(P0,deltaR,14*deltaPhi);
if(nt==360)InitQ(P0,deltaR,15*deltaPhi);
if(nt==370)InitQ(P0,deltaR,16*deltaPhi);
if(nt==380)InitQ(P0,deltaR,17*deltaPhi);
if(nt==390)InitQ(P0,deltaR,18*deltaPhi);
if(nt==400)InitQ(P0,deltaR,19*deltaPhi);
if(nt==410)InitQ(P0,deltaR,20*deltaPhi);
cout<<"temperature"<<vnew[0][0][0]<<endl;
void zlihcp::Clear(void)
{
    int i,j,k;
    for(i=0;i<NR+1;i++)
    {
        for(j=0;j<NPhi+1;j++)
        {
            for(k=0;k<NZ3+1;k++)
            {
                Q1[i][j][k]=0;
                Q2[i][j][k]=0;
                Q3[i][j][k]=0;
                v[i][j][k]=0;
                vnew[i][j][k]=0;
                vold[i][j][k]=0;
                vn[i][j][k]=0;
                beta[i][j][k]=0;
                f[i][j][k]=0;
                d[i][j][k]=0;
            }
        }
    }
}

double zlihcp::RunAll(double P0) //
{
    double TemRet = 0;
    Clear();
    TemRet = IntmTrsy(P0);
    return TemRet;
}

int main(void)
{
    zlihcp zl(NR+1,NPhi+1,NZ3+1);
    long double P0m, TIm, T2m, deltaP, X, Tpoint;
    long double S, Snew, Pnew, error1;
    ofstream fout14; fout14.open("pnew.txt",ios::out);
    P0m=16.0; Pnew=16.0;
    TIm=0; T2m=0;
    Tpoint=8; S=0;
    Snew=0; error1=0.001;
    do
    {
        P0m=Pnew; deltaP=P0m/100; S=Snew;
        TIm=zl.RunAll(P0m);
        T2m=zl.RunAll(P0m+deltaP);
        X=(T2m-TIm)/deltaP;
        Pnew = P0m+X*(X*S)*(Tpoint-TIm );
        Snew = (Tpoint-TIm)*(Tpoint-TIm);
        fout14<<"PNEW"<<Pnew<<endl;
        fout14<<"SNEW"<<Snew<<endl;
    } while ((Snew-S)/Snew > error1 );
    fout14<<"end"<<endl;
    fout14.close();
    return 0;
}
Le Zhang
4/11/05

This program is about heat transfer in the skin of a human being. There are three layers in the skin. The first layer is epidermis, the second one is dermis and the last one is subcutaneous.

#include <fstream.h>
#include<string.h>
#include <iostream.h>
#include <math.h>
#include<stdio.h>

#define NZ1 8
#define NZ2 208
#define NZ3 1208
#define NR 30
#define NPhi 20
#define CIRCLE 1
#define EndTemp 2
#define ConTemp 8
#define LSS4 0.04

class zlihcp
{
private:
    double ***Q1, ***Q2, ***Q3;
    double ***v, ***vnew, ***vold, ***vn;
    double ***beta, ***r, ***d;
    double *b, *a, *c;
    double MaxErr, h, e;
    double deltaZ, deltaT, deltaPhi, deltaR;
    double Rchange, Phichange, Zchange, Rchange1, Phichange1, Zchange1;
    double p1, p2, p3, p4, q1, q2, q3, k1, k2, k3, wb1, wb2, wb3, cb1, cb2, cb3;
    double Sigma, Alpha1, Alpha2, Alpha3, Reff1, Reff2, Reff3;
    double P0, pi, judge, CenterX, CenterY;
    int i, j, k;
    int MaxLen, MaxWid, MaxHig;

public:
    double point[5];
    double LSS, MLS1;
    double LSS4, MLS1;
    int TimeRec[100], FlagRec[100], nt, flag, CountNum;
    zlihcp(int l, int w, int high1)
    {
        int ik;
        MaxLen = l;
        MaxWid = w;
        MaxHig = high1;
        CountNum = 0;
        LSS = 0;
        MLS1 = 10000000;
        flag = 0;
        for (i = 0; i < 100; i++)
            TimeRec[i] = FlagRec[i] = -1;
        a = new double *[l];
        for (i = 0; i < l) ++
            a[i] = new double[high1];
        b = new double[high1];
        c = new double[high1];
        Q1 = new double *[l];
        Q2 = new double *[l];
        Q3 = new double *[l];
        v = new double *[l];
        vnew = new double *[l];
        void = new double *[l];
        vn = new double *[l];
        beta = new double *[l];
        f = new double *[l];
};
d = new double **[1];
for (i=0;i<l;i++)
{
    Q1[i] = new double *[w];
    Q2[i] = new double *[w];
    Q3[i] = new double *[w];
    v[i] = new double *[w];
    vnew[i] = new double *[w];
    vn[i] = new double *[w];
    beta[i] = new double *[w];
    d[i] = new double *[w];
    for (k=0;k<w;k++)
    {
        Q1[i][k] = new double [high1];
        Q2[i][k] = new double [high1];
        Q3[i][k] = new double [high1];
        v[i][k] = new double [high1];
        vnew[i][k] = new double [high1];
        vn[i][k] = new double [high1];
        beta[i][k] = new double [high1];
        d[i][k] = new double [high1];
    }
    }
for(i=0;i<l;i++)
{
    for(j=0;j<w;j++)
    {
        for(k=0;k<w;k++)
        {
            Q1[i][j][k]=0;
            Q2[i][j][k]=0;
            Q3[i][j][k]=0;
            v[i][j][k]=0;
            vnew[i][j][k]=0;
            vn[i][j][k]=0;
            beta[i][j][k]=0;
            d[i][j][k]=0;
        }
    }
}
Sigma= 0.1; Alpha1=1.0; Alpha2=0.8;
Alpha3=0.4; Refl1=0.93; Refl2=0.93;
Refl3=0.93; pi=3.14159265358979;
CenterX = 0; CenterY = 0;
p1=2; n1=10; p2=1.2; p2=1.2; p3=1.0;
q1=3.6; q2=3.4; q3=3.06;
k1=0.0026; k2=0.0052; k3=0.0021;
w1=0.0; wb2=0.0005; wb3=0.0005;
cb1=0.0; cb2=4.2; cb3=4.2; c=0.001;
deltaPhi = double(2*pi/(double)NPhi);
deltaR = double(0.5*(double)NR);
deltaZ=0.001; deltaT=0.1;
}
~zlihcp()
{
    delete [s];
    delete [b];
    delete [k];
    delete [Q1];
    delete [Q2];
    delete [Q3];
    delete [v];
    delete [vnew];
void InitQ(double,double,double);
void InitTry(double);
double RunAll (double);
void FileWrit(int);
void Clear(void);

void zlihcp::FileWrit(int time1)
{
    int i,k;
    ofstream fout1,fout2,fout3,fout4,fout5,fout6;
    sprintf(str,"%d",time1); strcat(str,str1); strcat(str,str2);
    strcat(str,str3); strcat(str,str4);
    fout1.open(str,ios::out);
    fout1<<"TITLE = 'Example: Simple ZT-Volume Data'" <<endl;
    fout1<<"VARIABLES = 'Z', 'Temperature'" <<endl;
    fout1<<"ZONE T=1209,F=POINT"<<endl;
    for(k=0;k<=NZ3;k++)
    fout1<<"<"<<new[0][0][k]"<<endl;  
fout1.close();
    fout2.open(str,ios::out);
    fout2<<"TITLE = 'Example: Simple ZT-Volume Data'" <<endl;
    fout2<<"VARIABLES = 'R', 'Z', 'Temperature'" <<endl;
    fout2<<"ZONE T="right",J=1209,J=31,F=POINT"<<endl;
    for(i=0;i<=N Phi/3;i++)
    {
        for(k=0;k<=NZ3;k++)
        fout2<<"<"<<double(k*deltaR)"<<"<"<<double(k*deltaZ)"<<"<"<<new[i][0][k]"<<endl;
    }
    fout2.close();
    fout3.open(str,ios::out);
    fout3<<"TITLE = 'Example: Simple ZT-Volume Data'" <<endl;
    fout3<<"VARIABLES = 'R', 'Z', 'Temperature'" <<endl;
    fout3<<"ZONE T="center",J=1209,J=31,F=POINT"<<endl;
    for(i=0;i<=N Phi/2;i++)
    {
        for(k=0;k<=NZ3;k++)
        fout3<<"<"<<double(i*deltaR)"<<"<"<<double(i*deltaZ)"<<"<"<<new[i][N Phi/2][k]"<<endl;
    }
    fout3.close();
    fout4.open(str,ios::out);
    fout4<<"TITLE = 'Example: Simple ZT-Volume Data'" <<endl;
    fout4<<"VARIABLES = 'R', 'Z', 'Temperature'" <<endl;
    fout4<<"ZONE T="center",J=1209,J=31,F=POINT"<<endl;
    for(i=0;i<=N Phi;i++)
    {
        for(k=0;k<=NZ3;k++)
        fout4<<"<"<<double(k*deltaR)"<<"<"<<double(k*deltaZ)"<<"<"<<new[i][N Phi/4][k]"<<endl;
    }
    fout4.close();
    fout5.open(str,ios::out);
    fout5<<"TITLE = 'Example: Simple ZT-Volume Data'" <<endl;
    fout5<<"VARIABLES = 'R', 'Z', 'Temperature'" <<endl;
    fout5<<"ZONE T="center",J=1209,J=31,F=POINT"<<endl;
    for(i=0;i<=N Phi;i++)
    {
        for(k=0;k<=NZ3;k++)
        fout5<<"<"<<double(k*deltaR)"<<"<"<<double(k*deltaZ)"<<"<"<<new[i][3*N Phi/4][k]"<<endl;
    }
    fout5.close();
    fout6.open(str,ios::out);
    fout6<<"TITLE = 'Example: Simple ZT-Volume Data'" <<endl;
    fout6<<"VARIABLES = 'R', 'Z', 'Temperature'" <<endl;
    fout6<<"ZONE T="center",J=1209,J=31,F=POINT"<<endl;
    for(i=0;i<=N Phi;i++)
    {
        for(k=0;k<=NZ3;k++)
        fout6<<"<"<<double(k*deltaR)"<<"<"<<double(k*deltaZ)"<<"<"<<new[i][1][k]"<<endl;
    }
    fout6.close();
    fout7.open(str,ios::out);
    fout7<<"TITLE = 'Example: Simple ZT-Volume Data'" <<endl;
    fout7<<"VARIABLES = 'R', 'Z', 'Temperature'" <<endl;
    fout7<<"ZONE T="center",J=1209,J=31,F=POINT"<<endl;
    for(i=0;i<=N Phi;i++)
    {
        for(k=0;k<=NZ3;k++)
        fout7<<"<"<<double(k*deltaR)"<<"<"<<double(k*deltaZ)"<<"<"<<new[i][N Phi/4][k]"<<endl;
    }
    fout7.close();
}
void zlihcp::InitQ(double P0, double Cr, double Cp) // Initialize the laser power;
{
  int i, j, z;
  CenterX = Cr * cos(Cp);
  CenterY = Cr * sin(Cp);
  for(i=0; i<=NR, +++)
  {
    for(j=0; j<=NPhi, +++)
    {
      for(z=0; z<=NZ1, +++)
      {
        Q1[i][j][z]= Alpha1 * exp( -(Alpha1 * z * deltaZ) / (sqrt(2*pi) * Sigma) * exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX, 2) + pow(i*sin(j*deltaPhi)*deltaR-CenterY, 2) / (2*Sigma*Sigma)) * P0*(1-Ref1));
      }
    }
  }
  for(z=0; z<=NZ2, +++)
  {
    Q2[i][j][z]= Alpha2 * exp( -(Alpha2 * (z-NZ1) * deltaZ) / (sqrt(2*pi) * Sigma) * exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX, 2) + pow(i*sin(j*deltaPhi)*deltaR-CenterY, 2) / (2*Sigma*Sigma)) * P0*(1-Ref2));
  }
  for(z=0; z<=NZ3, +++)
  {
    Q3[i][j][z]= Alpha3 * exp( -(Alpha3 * (z-NZ2) * deltaZ) / (sqrt(2*pi) * Sigma) * exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX, 2) + pow(i*sin(j*deltaPhi)*deltaR-CenterY, 2) / (2*Sigma*Sigma)) * P0*(1-Ref3));
  }
}

double zlihcp::IntmTrsy(double P0) // Time Iteration and Tri-diagonal system
{
  MaxErr=1.0; nt=0; MLSS=100000; MLSS1=100000;
  while(nt<1)
  {
    nt++;
    if(nt == 1) InitQ(P0,0,0);
    t=0;
    for(i=0; i<=NR, +++)
    {
      for(j=0; j<=NPhi, +++)
      {
        for(z=0; z<=NZ3, +++)
        {
          
        }
      }
    }
  }
vn[i][j][z]=vold[i][j][z];
}
}
cout<<"new cicle"<<endl;
MaxErr=1.0;
while(MaxErr>=e)
{
MaxErr=0.0;
for(i=1;i<=NR-1;i++)
{
for(j=1;j<=NPhi;j++)
{
///////////////////////////////////////////////////////////////////////////
The first layer///////////////////////////////////////////////////////////////////////////
for(z=1;z<=NZ1-1;z++)
{
    Rchange = k1*deltaT*((i+0.5)*vold[i+1][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/(i*deltaR*deltaR);
    if(j==NPhi)
    {
        Phichange = k1*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
    }
    else
    {
        Phichange = k1*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
    }
    Zchange = k1*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z-1])/pow(deltaZ,2);
    ///////////////////////////////////////////////////////////////////////////
The n state ///////////////////////////////////////////////////////////////////////////
    Rchangel = k1*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
    if(j==NPhi)
    {
        Phichangel = k1*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
    }
    else
    {
        Phichangel = k1*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
    }
    Zchangel = k1*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/pow(deltaZ,2);
    ///////////////////////////////////////////////////////////////////////////
    Prepare the coefficient for Thomas way ///////////////////////////////////////////////////////////////////////////
    fli[]=rchange+Phichangel+Zchangel+2*deltaT*Ql[i][j][z]+(2*pl*qc1-wb1*cbl*deltaT)*vn[i][j][z];
    b[]=k1*deltaT/(deltaZ*deltaZ);
    a[i][j][z]=2*pl*qc1+wb1*cbl*deltaT+4/(i*deltaR*deltaR)+k1*deltaT/2*deltaR*deltaR;
    c[z]=(k1*deltaT)/(deltaZ*deltaZ);
    d[i][j][z]=f[i][j][z]+Rchange+Phichange+k1*deltaT*((i+0.5)*deltaR*deltaR)+4*deltaT*Ql[i][j][z];
}
}

a[1][1]=a[1][1]-b[1];
b[1]=0;
b[NZ1]=k1;
a[]][NZ1]=k1+2;
c[NZ1]=k2;
d[]][NZ1]=0;
///////////////////////////////////////////////////////////////////////////The second layer///////////////////////////////////////////////////////////////////////////
for(z=NZ1+1;z<=NZ2-1;z++)
{
///////////////////////////////////////////////////////////////////////////The n+1 state///////////////////////////////////////////////////////////////////////////
    Rchange = k2*deltaT*((i+0.5)*vold[i+1][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/(i*deltaR*deltaR);
    if(j==NPhi)
    {
        Phichange = k2*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
    }
    else
    {
        Phichange = k2*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
    }
    Zchange = k2*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z-1])/pow(deltaZ,2);
    ///////////////////////////////////////////////////////////////////////////
The n state ///////////////////////////////////////////////////////////////////////////
    Rchangel = k2*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
    if(j==NPhi)
    {
        Phichangel = k2*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
    }
    else
    {
\[
\text{Phichangel} = k_2 \cdot \text{deltaT} \cdot (v_{n[i][j][z]} + 2 \cdot v_{n[i][j-l][z]} - 2 \cdot v_{n[i][j+l][z]}) / \text{pow}(i \cdot \text{deltaR} \cdot \text{deltaPhi}, 2);
\]

\[
\text{Zchangel} = k_2 \cdot \text{deltaT} \cdot (v_{n[i][j][z+l]} + 2 \cdot v_{n[i][j-l][z]} - 2 \cdot v_{n[i][j][z-l]}) / \text{pow}(\text{deltaZ}, 2);
\]

\[
\text{// Prepare the coefficient for Thom as way{//}
\]

\[
f[i][j][z] = R_{\text{change}} + \text{Phichangel} + \text{Zchangel} + 2 \cdot \text{deltaT} \cdot Q_2[i][j][z] + (2 \cdot p_2 \cdot q_2 - w_2 \cdot c_2 \cdot \text{deltaT}) \cdot v_{n[i][j][z]};
\]

\[
b[z] = k_2 \cdot \text{deltaT} / (\text{deltaZ} \cdot \text{deltaZ});
\]

\[
a[i][j][z] = 2 \cdot p_2 \cdot q_2 + w_2 \cdot c_2 \cdot \text{deltaT} \cdot k_2 \cdot \text{deltaT} \cdot (4i+1) / (i \cdot \text{deltaR} \cdot \text{deltaR}) + 4 \cdot \text{pow}(\text{deltaPhi} \cdot i \cdot \text{deltaR}, 2) / \text{pow}(\text{deltaPhi}, 2); \]
for( i = 1 ; i <= NR -1 ; i++ )
{
    for( j = 1 ; j <= NPhi ; j++ )
    {
        for( z = 1 ; z <= NZ3 -1 ; z++ )
        {
            vnew[ i ][ j ][ z ] = v[ i ][ j ][ z ] + beta[ i ][ j ][ z ] * vnew[ i ][ j ][ z -1 ];
            judge = vnew[ i ][ j ][ z ] - void[ i ][ j ][ z ];
            if( judge < 0 )
                judge = judge * (-1);
            if( judge > MaxErr )
                MaxErr = judge;
            void[ i ][ j ][ z ] = vnew[ i ][ j ][ z ];
        }
    }
}
t++;
cout << "number" << t << "<<MaxErr" << MaxErr << endl;
for( i = 0 ; i <= NR ; i++ )
{
    for( j = 0 ; j <= NPhi ; j++ )
    {
        for( z = 0 ; z <= NZ3 ; z++ )
        {
            vnew[ i ][ j ][ 0 ] = vnew[ i ][ j ][ 1 ];
            vnew[ i ][ j ][ NZ3 ] = vnew[ i ][ j ][ NZ3 -1 ];
            vnew[ i ][ j ][ z ] = vnew[ i ][ j ][ NPhi ][ z ];
            vnew[ 0 ][ j ][ z ] = vnew[ 1 ][ j ][ z ];
            vnew[ NR ][ j ][ z ] = vnew[ NR -1 ][ j ][ z ];
            void[ i ][ j ][ z ] = vnew[ i ][ j ][ z ];
        }
    }
}

/////////////////////five points code here///////////////////// 
point[0] = vnew[0][0][0]; point[1] = vnew[NR][0][0]; 
point[2] = vnew[NR][3*NPhi/4][0]; point[3] = vnew[NR][NPhi/2][0];
point[4] = vnew[NR][NPhi/4][0];

////////////////////////////////////////////////////
if( nt == 10 ) InitQP0,deltaR,1*deltaPhi);
if( nt == 20 ) InitQP0,deltaR,2*deltaPhi);
if( nt == 30 ) InitQP0,deltaR,3*deltaPhi);
if( nt == 40 ) InitQP0,deltaR,4*deltaPhi);
if( nt == 50 ) InitQP0,deltaR,5*deltaPhi);
if( nt == 60 ) InitQP0,deltaR,6*deltaPhi);
if( nt == 70 ) InitQP0,deltaR,7*deltaPhi);
if( nt == 80 ) InitQP0,deltaR,8*deltaPhi);
if( nt == 90 ) InitQP0,deltaR,9*deltaPhi);
if( nt == 100 ) InitQP0,deltaR,10*deltaPhi);
if( nt == 110 ) InitQP0,deltaR,11*deltaPhi);
if( nt == 120 ) InitQP0,deltaR,12*deltaPhi);
if( nt == 130 ) InitQP0,deltaR,13*deltaPhi);
if( nt == 140 ) InitQP0,deltaR,14*deltaPhi);
if( nt == 150 ) InitQP0,deltaR,15*deltaPhi);
if( nt == 160 ) InitQP0,deltaR,16*deltaPhi);
if( nt == 170 ) InitQP0,deltaR,17*deltaPhi);
if( nt == 180 ) InitQP0,deltaR,18*deltaPhi);
if( nt == 190 ) InitQP0,deltaR,19*deltaPhi);
if( nt == 200 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 210 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 220 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 230 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 240 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 250 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 260 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 270 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 280 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 290 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 300 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 310 ) InitQP0,deltaR,20*deltaPhi);
if( nt == 320 ) InitQP0,deltaR,20*deltaPhi);

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
if(nt==330)InitQ(P0,deltaR,12*deltaPhi);
if(nt==340)InitQ(P0,deltaR,13*deltaPhi);
if(nt==350)InitQ(P0,deltaR,14*deltaPhi);
if(nt==360)InitQ(P0,deltaR,15*deltaPhi);
if(nt==370)InitQ(P0,deltaR,16*deltaPhi);
if(nt==380)InitQ(P0,deltaR,17*deltaPhi);
if(nt==390)InitQ(P0,deltaR,18*deltaPhi);
if(nt==400)InitQ(P0,deltaR,19*deltaPhi);
if(nt==410)InitQ(P0,deltaR,20*deltaPhi);

if(nt>410)
{
LSS = pow((CentTemp-point[0]),2)/(CentTemp*CentTemp)+pow((EndTemp-point[1]),2)/(EndTemp*EndTemp)
+pow((EndTemp-point[2]),2)/(EndTemp*EndTemp)+pow((EndTemp-point[3]),2)/(EndTemp*EndTemp)
+pow((EndTemp-point[4]),2)/(EndTemp*EndTemp);
LSS_4 = pow((EndTemp-point[1]),2)/(EndTemp*EndTemp)+pow((EndTemp-point[2]),2)/(EndTemp*EndTemp)
+pow((EndTemp-point[3]),2)/(EndTemp*EndTemp)+pow((EndTemp-point[4]),2)/(EndTemp*EndTemp);
}

if(flag = 0) //stop heating
{
if(LSS_4<LSS4)
{

InitQ(0,CIRCLE*deltaR,20*deltaPhi);
if(\nLSS_4<LSS4)
{

TimeRec[CountNum] = nt;
FlagRec[CountNum] = flag;
CountNum ++;
flag = 2;
}
{
TimeRec[CountNum] = nt;
FlagRec[CountNum] = flag;
CountNum ++;
flag = 1;
FileWrit(nt);
}
}

if(flag = 1) //start heating
{
InitQ(P0,0*deltaR,0*deltaPhi);
if(LSS_4<LSS4)
{

TimeRec[CountNum] = nt;
FlagRec[CountNum] = flag;
CountNum ++;
flag = 2;
}
if(point[0]>CentTemp)
{
TimeRec[CountNum] = nt;
FlagRec[CountNum] = flag;
CountNum ++;
FileWrit(nt);
flag = 0;
}
}

if(flag == 2) //stop heating
{
InitQ(P0,0*deltaR,0*deltaPhi);
if(point[0]>CentTemp)
{

TimeRec[CountNum] = nt;
FlagRec[CountNum] = flag;
goto loopend;
}
}

}

loopend:
FileWrit(nt);

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
return vnew[0][0][0];
}

void zlihcp::Clear(void)
{
    int i,j,k;
    for(i=0;i<NR+1;i++)
    {
        for(j=0;j<NPhi+1;j++)
        {
            for(k=0;k<NZ3+1;k++)
            {
                Ql[i][j][k]=0;
                Q2[i][j][k]=0;
                Q3[i][j][k]=0;
                vnew[i][j][k]=0;
                vold[i][j][k]=0;
                vn[i][j][k]=0;
                beta[i][j][k]=0;
                f[i][j][k]=0;
                d[i][j][k]=0;
            }
        }
    }
}

double zlihcp::RunAll(double P0) //
{
    double TemRet = 0;
    Clear();
    TemRet = InmTrsy(P0);
    return TemRet;
}

int main(void)
{
    zlihcp zl(NR+1,NPhi+1,NZ3+1);
    long double P0m;
    int i;
    ofstream fout14;
    fout14.open("time.txt",ios::out);
    P0m=16.6159;
    zl.RunAll(P0m);
    fout14<<"LEAST SUM SQUARE "<<zl.LSS<<endl;
    fout14<<"4 LEAST SUM SQUARE "<<zl.LSS_4<<endl;
    for(i=0;i<100;i++)
    {
        if(zl.TimeRec[i]!=-1)
        {
            if(zl.FlagRec[i]==0)
            fout14<<"Number "<<zl.CoolTime "<<zl.TimeRec[i]<<endl;
            if(zl.FlagRec[i]==1)
            fout14<<"Number "<<zl.HeatTime "<<zl.TimeRec[i]<<endl;
            else
            fout14<<"Number "<<zl.EndTime "<<zl.TimeRec[i]<<endl;
        }
        fout14<<"END"<<endl;
    fout14.close();
    return 0;
}
/* Table A.3 Program 1: Source code of step 3 in Figure 3.2 is used for
the skin model without any blood vessels.

Le Zhang
4/11/05
This program is about heat transfer in the skin of a human being.
There are three layers in the skin. The first layer is epidermis,
the second one is dermis and the last one is Subcutaneous.
*/

#include <fstream.h>
#include <string.h>
#include <iostream.h>
#include <math.h>
#include<stdio.h>
#define NZ1 8
#define NZZ 208
#define NZ3 1208
#define NR 30
#define NPhi 20
#define CIRCLE 1
#define EndTemp 3
#define CentTemp 8
#define LSS 0.04
#define T1 965  //LSS is right
#define T2 1062  //end time

class zlihcp
{
private:
    double ***Q1,***Q2,***Q3;
    double ***v,***vnew,***vold,***vn,***vsave;
    double ***beta,***f,***d;
    double *b,**a,*c;
    double MaxErr,h,e;
    double deltaZ,deltaT,deltaPhi,deltaR;
    double R change,P hichange,Zchange,R changel,Phichangel,Z ch ange1;
    double p 1  ,p2 ,p3 ,qc 1  ,qc2 ,qc3 ,k 1  ,k2,k3 ,wb 1, w b2,w b3 ,cb 1  ,cb2 ,cb3;
    double Sigma,A lphal ,Alpha2 ,Alpha3 ,R effl ,Reff2 ,Reff3;
    double P0, pi,judge,CenterX,CenterY;
    int i,j,k;
    int MaxLen,MaxWid,MaxHig;
public:
    double point[5];
    double LSS,M LSS;
    double L SS_4,L SS1,M L SS1;
    int TimeRec[100],FlagRec[100],nt,flag,CountNum;
    zlihcp(int l, int w, int hight);
    
    private:
    }
    public:
    void = new double **[1];
    double *b,**a,*c;
    double MaxErr,h,e;
    double deltaZ,deltaT,deltaPhi,deltaR;
    double R change,P hichange,Zchange,R changel,Phichangel,Z ch ange1;
    double p 1  ,p2 ,p3 ,qc 1  ,qc2 ,qc3 ,k 1  ,k2,k3 ,wb 1, w b2,w b3 ,cb 1  ,cb2 ,cb3;
    double Sigma,A lphal ,Alpha2 ,Alpha3 ,R effl ,Reff2 ,Reff3;
    double P0, pi,judge,CenterX,CenterY;
    int i,j,k;
    int MaxLen,MaxWid,MaxHig;
    public:
    double point[5];
    double LSS,M LSS;
    double L SS_4,L SS1,M L SS1;
    int TimeRec[100],FlagRec[100],nt,flag,CountNum;
    zlihcp(int l, int w, int hight);
    
    private:
    }
    public:
    void = new double **[1];
beta = new double **[n];
f = new double **[l];
d = new double **[l];
for (j=0;j<l;j++)
{
    Q1[j] = new double *[w];
    Q2[j] = new double *[w];
    Q3[j] = new double *[w];
    v[j] = new double *[w];
    vold[j] = new double *[w];
    vn[j] = new double *[w];
    vsave[j] = new double *[w];
    beta[j] = new double *[w];
    f[j] = new double *[w];
    d[j] = new double *[w];
    for (k=0;k<w;k++)
    {
        Q1[j][k] = new double [h];
        Q2[j][k] = new double [h];
        Q3[j][k] = new double [h];
        v[j][k] = new double [h];
        vold[j][k] = new double [h];
        vn[j][k] = new double [h];
        vsave[j][k] = new double [h];
        beta[j][k] = new double [h];
        f[j][k] = new double [h];
        d[j][k] = new double [h];
    }
}
for(i=0;i<l;i++)
{
    for(j=0;j<w;j++)
    {
        for(k=0;k<h;k++)
        {
            Q1[i][j][k] = 0;
            Q2[i][j][k] = 0;
            Q3[i][j][k] = 0;
            v[i][j][k] = 0;
            vold[i][j][k] = 0;
            vn[i][j][k] = 0;
            vsave[i][j][k] = 0;
            beta[i][j][k] = 0;
            f[i][j][k] = 0;
            d[i][j][k] = 0;
        }
    }
}
Sigma = 0.1; Alpha1=1.0; Alpha2=0.8;
Alpha3=0.4; Refl=0.93; Refr=0.93;
Refd=0.93; pi=3.14159265358979;
CenterX = 0; CenterY = 0;
t=2; n=10; p=1.2; p2=1.2; p3=1.0;
qc1=3.0; qc2=3.4; qc3=3.06;
k1=0.0026; k2=0.0052; k3=0.0021;
wbl=0.0; wbl=0.0005; wbl=0.0005;
c=0.0; cb1=4.2; cb1=4.2; c=0.001;
deltaphi = double(2*pi/(double)Nphi);
deltar = double(0.5*(double)NR);
delta2=0.001; deltart=0.1;
};
-zihep()
{
    delete [a];
    delete [b];
    delete [c];
void zlihcp::FileWrit(int time)
{
    int i,k; ofstream fout1,fout21,fout21, fout3, fout31;
    char str[20],str1[20],str2[20],str21[20],str3[20],str31[20];
    sprintf((str, "%d",time1); strcat(str1,str);
    strcat(str2,str); strcat(str21,str);
    strcat(str3,str); strcat(str31,str);
    fout1.open(str1,ios::out);
    fout2 << TITLE = "Example: Simple ZT-Volume Data" << endl;
    fout2 << VARIABLES = "Z", "Temperature" << endl;
    fout2 << ZONE 1=1209,F=POINT<<endl;
    for(k=0;k<=NZ3;k++)
    fout2 << double(k*deltaZ)<< "<"<<new[0][0][k]<<endl;
    fout1.close();
    fout2.open(str2,ios::out);
    fout2 << TITLE = "Example: Simple 2D-Volume Data" << endl;
    fout2 << VARIABLES = "R", "Z", "Temperature" << endl;
    fout2 << ZONE T="right",I=1209, J=31, F=POINT<<endl;
    for(i=0;i<=NR;i++)
    {
        for(k=0;k<=NZ3;k++)
        fout2 << double(i*deltaR)<< "<"<<double(k*deltaZ)<< "<"<<new[i][0][k]<<endl;
    }
    fout2.close();
    fout21.open(str21,ios::out);
    fout21 << TITLE = "Example: Simple 2D-Volume Data" << endl;
    fout21 << VARIABLES = "R", "Z", "Temperature" << endl;
    fout21 << ZONE T="reverse",I=1209, J=31, F=POINT<<endl;
    for(i=0;i<=NR;i++)
    {
        for(k=0;k<=NZ3;k++)
        fout21 << double(i*deltaR)<< "<"<<double(k*deltaZ)<< "<"<<new[i][NPhi/2][k]<<endl;
    }
    fout21.close();
}
for(k=0;k<=NZ3;k++)
fout21<<double(-i*deltaR)<< " " << double(k*deltaZ)<< " " << new[i][3*NPhi/4][k]<<endl;
}
fout21.close();
///////////////Center cross////////////
fout3.open(str3,ios::out);
fout3<< " TITLE = V'Example: Simple 2D-Volume Data' " <<endl;
fout3<< " VARIABLES = V'R', V'Temperature' " <<endl;
fout3<< " ZONE T="right",I=31, F=POINT" <<endl;
for(i=0;i<=NR;i++)
fout3<<double(*deltaR)<< " " << new[i][0][0]<<endl;
fout3.close();
void zlihcp::InitQ(double PO,double Cr,double Cp) // Initialize the laser power
{
int i,j,z; CenterX = Cr * cos(Cp); CenterY = Cr * sin(Cp);
for(i=0;i<=NR;i++)
{
for(j=0;j<=NPhi;j++)
for(z= 0;z< = NZ1;z++)
Q1[i][j][z]= Alpha1*exp(-Alpha1*z*deltaZ)/(sqrt(2*pi)*Sigma)*exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*PO*(1-Refl);
for(z=NZ1+1 ;z<=NZ2 ;z++)
Q2[i][j][z]= Alpha2*exp(-Alpha2*(z-NZ1)*deltaZ)*exp(-Alpha1*deltaZ*NZ1)/(sqrt(2*pi)*Sigma)*exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*PO*(1-Refl2);
for(z=NZ2+1 ;z<=NZ3 ;z++)
Q3[i][j][z]= Alpha3*exp(-Alpha3*(z-NZ2) *deltaZ)*exp(-Alpha1 *deltaZ*NZ2)/(sqrt(2*pi)*Sigma)*exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*PO*(1-Refl3);
}
double zlihcp::IntmTrsy(double PO) // Time Iteration and Tri-diagonal systm
{x=1.0; nt=0;
while(nt> -1)
{
nt++;
if(nt = = 1) InitQ(PO,0,0);
t=0;
for(i=0;i<=NR;i++)
{...}}
for(j=0; j<=Nphi; j++)
{
for(z=0; z<=NZ3; z++)
{
vn[i][j][z]=vold[i][j][z];
vsave[i][j][z]=vold[i][j][z];
}
}
cout<<"new cycle"<<endl;
MaxErr=1.0;
while(MaxErr>=e)
{
MaxErr=0.0;
for(i=1; i<=NR-1; i++)
{
for(j=0; j<=Nphi; j++)
{
for(z=0; z<=NZ1-1; z++)
{
Rchange = k1*deltaT*((i+0.5)*vold[i+1][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/((deltaR*deltaR);
if(j==Nphi)
{
Phichange = k1*deltaT*(vold[i][i][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
else
{
Phichange = k1*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
Zchange = k1*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z-1])/pow(deltaZ,2);

Rchange1 = k1*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(deltaR);
if(j==Nphi)
{
Phichange1 = k1*deltaT*(vn[i][i][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
else
{
Phichange1 = k1*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
Zchange1 = k1*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/pow(deltaZ,2);

Rchange = k2*deltaT*((i+0.5)*vold[i+1][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/((deltaR*deltaR);
if(j==Nphi)
{
Phichange = k2*deltaT*(vold[i][i][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
else
{
Phichange = k2*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
Zchange = k2*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z-1])/pow(deltaZ,2);

b[z]=(k1*deltaT)/(deltaZ*deltaZ);
a[i][j]=2*pi*qi+wb1*cb1*deltaT*vn[i][j][z];
}}
}
\[
R_{\text{changel}} = k_2 \cdot \delta T \cdot (v_{i+0.5}^{n} + v_{i-0.5}^{n+1}) - 2 \cdot i \cdot v_{i}^{n} + (i-0.5) \cdot v_{i-1}^{n} + (i+0.5) \cdot v_{i+1}^{n}
\]

\[
f[i][j][z] = R_{\text{changel}} + P_{\text{chiangle}} + Z_{\text{changel}} + 2 \cdot \delta T \cdot Q_{2n}[i][j][z] + (2 \cdot p_2 \cdot q_{c2} - w_{b2} \cdot c_{b2} \cdot \delta T) \cdot v_{i}^{n} + \left(\frac{4 \cdot i \cdot \delta R \cdot \delta R}{(i \cdot \delta R)^2} + \frac{4 \cdot \delta T}{\delta Z^2}\right)\]

\[
b[NZ2] = k_2;
\]

\[
a[i][NZ2] = k_2 + k_3;
\]

\[
c[NZ2] = 0;
\]

\[
d[i][j][NZ2] = 0;
\]

\[
\text{// tri-diagonal system}\n\]
for (z = NZ2 - 1; z >= 1; z--)
{
  v[i][j][z] = a[i][j][z] + c[i][j][z] * v[i][j][z+1];
  beta[i][j][z] = b[i][j][z] * c[i][j][z] * beta[i][j][z+1];
}
for (i = 1; i <= NR - 1; i++)
{
  for (j = 0; j <= NPHI; j++)
  {
    for (z = 1; z <= NZ2 - 1; z++)
    {
      new[i][j][z] = v[i][j][z] + beta[i][j][z] * vnew[i][j][z+1];
      judge = (v[i][j][z] - vold[i][j][z]);
      if (judge < 0)
        judge = judge * (-1);
      if (judge > MaxErr)
        MaxErr = judge;
      vold[i][j][z] = vnew[i][j][z];
    }
  }
  t++; cout << "number" << t << " " << "MaxErr" << MaxErr << endl;
for (i = 0; i <= NR - 1; i++)
{
  for (j = 0; j <= NPHI; j++)
  {
    for (z = 0; z <= NZ2; z++)
    {
      vnew[i][j][0] =  vnew[i][j][1];
      vnew[i][j][NZ2] =  vnew[i][j][NZ2 - 1];
      vnew[i][j][0][z] =  vnew[i][j][NPHI][z];
      vnew[i][j][0][z] =  vnew[i][j][z];
      vnew[i][j][NPHI][z] =  vnew[i][j][z];
      vnew[i][j][NPHI/2][z] =  vnew[i][j][z];
      vold[i][j][z] =  vnew[i][j][z];
    }
  }
}

// five points code here

if (nt == 10) InitQ(P0, deltaR, 1 * deltaPhi);
if (nt == 20) InitQ(P0, deltaR, 2 * deltaPhi);
if (nt == 30) InitQ(P0, deltaR, 3 * deltaPhi);
if (nt == 40) InitQ(P0, deltaR, 4 * deltaPhi);
if (nt == 50) InitQ(P0, deltaR, 5 * deltaPhi);
if (nt == 60) InitQ(P0, deltaR, 6 * deltaPhi);
if (nt == 70) InitQ(P0, deltaR, 7 * deltaPhi);
if (nt == 80) InitQ(P0, deltaR, 8 * deltaPhi);
if (nt == 90) InitQ(P0, deltaR, 9 * deltaPhi);
if (nt == 100) InitQ(P0, deltaR, 10 * deltaPhi);
if (nt == 110) InitQ(P0, deltaR, 11 * deltaPhi);
if (nt == 120) InitQ(P0, deltaR, 12 * deltaPhi);
if (nt == 130) InitQ(P0, deltaR, 13 * deltaPhi);
if (nt == 140) InitQ(P0, deltaR, 14 * deltaPhi);
if (nt == 150) InitQ(P0, deltaR, 15 * deltaPhi);
if (nt == 160) InitQ(P0, deltaR, 16 * deltaPhi);
if (nt == 170) InitQ(P0, deltaR, 17 * deltaPhi);
if (nt == 180) InitQ(P0, deltaR, 18 * deltaPhi);
if (nt == 190) InitQ(P0, deltaR, 19 * deltaPhi);
if (nt == 200) InitQ(P0, deltaR, 20 * deltaPhi);
if (nt == 210) InitQ(P0, deltaR, 21 * deltaPhi);
if (nt == 220) InitQ(P0, deltaR, 22 * deltaPhi);
if (nt == 230) InitQ(P0, deltaR, 23 * deltaPhi);
if (nt == 240) InitQ(P0, deltaR, 24 * deltaPhi);
if (nt == 250) InitQ(P0, deltaR, 25 * deltaPhi);

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
if(nt==260)InitQ(P0,deltaR,5*deltaPhi);
if(nt==270)InitQ(P0,deltaR,6*deltaPhi);
if(nt==280)InitQ(P0,deltaR,7*deltaPhi);
if(nt==290)InitQ(P0,deltaR,8*deltaPhi);
if(nt==300)InitQ(P0,deltaR,9*deltaPhi);
if(nt==310)InitQ(P0,deltaR,10*deltaPhi);
if(nt==320)InitQ(P0,deltaR,11*deltaPhi);
if(nt==330)InitQ(P0,deltaR,12*deltaPhi);
if(nt==340)InitQ(P0,deltaR,13*deltaPhi);
if(nt==350)InitQ(P0,deltaR,14*deltaPhi);
if(nt==360)InitQ(P0,deltaR,15*deltaPhi);
if(nt==370)InitQ(P0,deltaR,16*deltaPhi);
if(nt==380)InitQ(P0,deltaR,17*deltaPhi);
if(nt==390)InitQ(P0,deltaR,18*deltaPhi);
if(nt==400)InitQ(P0,deltaR,19*deltaPhi);
if(nt==410)InitQ(P0,deltaR,20*deltaPhi);

if(nt>410)
{
    LSS = pow((CenTemp-point[0]),2)/(CenTemp*CenTemp)+pow((EndTemp-point[1]),2)/(EndTemp*EndTemp)
        +pow((EndTemp-point[2]),2)/(EndTemp*EndTemp)+pow((EndTemp-point[3]),2)/(EndTemp*EndTemp)
        +pow((EndTemp-point[4]),2)/(EndTemp*EndTemp);
    if(flag = 0) //stop heating
    {
        InitQ(0,CIRCLE*deltaR,20*deltaPhi);
        if(LSS_4<LSS4)
        {
            TimeRec[CountNum] = nt;
            FlagRec[CountNum] = flag;
            CountNum ++;
            goto loopend;
        }
        {
            TimeRec[CountNum] = nt;
            FlagRec[CountNum] = flag;
            CountNum ++;
            flag = 1; FileWrit(nt);
        }
    }
}
if(flag == 1) //start heating
{
    InitQ(P0,0*deltaR,0*deltaPhi);
    if(LSS_4<LSS4)
    {
        TimeRec[CountNum] = nt;
        FlagRec[CountNum] = flag;
        CountNum ++;
        goto loopend;
    }
    if(point[0]<CenTemp)
    {
        TimeRec[CountNum] = nt;
        FlagRec[CountNum] = flag;
        CountNum ++;
        FileWrit(nt);
        flag = 0;
    }
}
if(flag == 2) //stop heating
{
    goto loopend;
}
}
loopend:
FileWrit(nt);
return vnew[0][0][0];

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
void zlihcp::Clear(void)
{
    int i,j,k;
    for(i=0;i<NR+l;i++)
    {
        for(j=0;j<NPhi+l;j++)
        {
            for(k=0;k<NZ3+l;k++)
            {
                Q1[i][j][k]=0;
                Q2[i][j][k]=0;
                Q3[i][j][k]=0;
                v[i][j][k]=0;
                vnew[i][j][k]=0;
                void[i][j][k]=0;
                v[i][j][k]=0;
                beta[i][j][k]=0;
                f[i][j][k]=0;
                d[i][j][k]=0;
            }
        }
    }
}

void zlihcp::Clear1(void)
{
    int i,j,k;
    for(i=0;i<NR+1;i++)
    {
        for(j=0;j<NPhi+1;j++)
        {
            for(k=0;k<NZ3+1;k++)
            {
                Q1[i][j][k]=0;
                Q2[i][j][k]=0;
                Q3[i][j][k]=0;
                v[i][j][k]=vsave[i][j][k];
                v[i][j][k]=0;
                beta[i][j][k]=0;
                f[i][j][k]=0;
                d[i][j][k]=0;
                nt=T1; //4LSS is the minimum;
            }
        }
    }
}

double zlihcp::RunAll(double PO) //
{
    double TemRet = 0;
    Clear(); TemRet = IntmTrsyl(PO);
    return TemRet;
}

double zlihcp::IntmTrsyl(double PO) // Time Iteration and Tri-diagonal systme
{
    MaxErr=1.0;
    while(nt<T2) //need the center temp back to 8
    {
        nt++; InitQ(PO,0,0); t=0;
        for(i=0;i<NR;i++)
        {
            for(j=0;j<NPhi+j++)
            {
                for(z=0;z<NZ3;z++)
                {
                    v[i][j][z]=void[i][j][z];
                }
            }
        }
    }
}
MaxErr=1.0;
while(MaxErr>e) {
  MaxErr=0.0;
  for(i=1;i<=NR-1;i++) {
    for(j=1;j<=NPhi;j++) {
      // The first layer
      for(z=1;z<=NZ1-1;z++) {
        Rchange = k1*deltaT*((i+0.5)*vold[i][j][z]-2*i*vold[i-1][j][z]+(i-0.5)*vold[i+1][j][z])/(i*deltaR*deltaR);
        if(j==NPhi)
          Phichange = k1*deltaT*(vold[i][j][z]-2*vold[i][j-1][z])/(i*deltaR*deltaPhi);
        else
          Phichange = k1*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/(i*deltaR*deltaPhi);
        Zchange = k1*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z-1])/(i*deltaR*deltaZ);
        // The n state
        Rchange1 = k1*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
        if(j==NPhi)
          Phichange1 = k1*deltaT*(vn[i][j][z]-2*vn[i][j-1][z])/(i*deltaR*deltaPhi);
        else
          Phichange1 = k1*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/(i*deltaR*deltaPhi);
        Zchange1 = k1*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/(i*deltaR*deltaZ);
        // Prepare the coefficient for Thomas way
        f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*q1-wb1*cb1*deltaT)*vn[i][j][z];
        b[z]=(k1*deltaT)/(deltaZ*deltaZ);
        a[i][j][z]=2*p1*q1+w1*cb1*deltaT+k1*deltaT*(deltaPhi*deltaR);
        c[z]=(k1*deltaT)/(deltaZ*deltaZ);
        d[i][j][z]=f[i][j][z]-Rchange+Phichange;
        if(j==NPhi)
          Phichange = k2*deltaT*((i+0.5)*vold[i][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i][j-1][z])/(i*deltaR*deltaR);
        else
          Phichange = k2*deltaT*(vold[i][j][z]-2*vold[i][j-1][z])/(i*deltaR*deltaPhi);
        Zchange = k2*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z-1])/(i*deltaR*deltaZ);
        // The n state
        Rchange1 = k2*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
        if(j==NPhi)
          Phichange1 = k2*deltaT*(vn[i][j][z]-2*vn[i][j-1][z])/(i*deltaR*deltaPhi);
        else
          Phichange1 = k2*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/(i*deltaR*deltaPhi);
      }
    }
  }
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
else
{
  Phichangel = k2*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
  }
Zchangel = k2*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/pow(deltaz,2);

/****Prepare the coefficient for Thomas way*************/

f[i][j][z]=Rchangel+Phichangel+zchangel+2*deltaT*Q2[i][j][z]+(2*p2*qc-wb*cb*deltaT)*vn[i][j][z];
b[i][z]=k2*deltaT/(deltaz*deltaz);
a[i][z]=2*p2*qc-wb*cb*deltaT/(4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));
c[i][j][z]=k2*deltaT/(deltaz*deltaz);
d[i][j][z]=f[i][j][z]+Rchangel+Phichangel+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));
}

b[NZ2]=k2;
a[NZ2]=k2+k3;
c[NZ2]=k3;
d[NZ2]=0;
/****The third layer ***********/

for(z=NZ2+1;z<=NZ3-1;z++)
{

/****The n+1 state ***********/

Rchangel = k3*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
if(j==NPphi)
{
  Phichangel = k3*deltaT*(vn[i][l][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
  }
else
{
  Phichangel = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
  }
Zchangel = k3*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/pow(deltaz,2);

/****The n state ***********/

Rchangel = k3*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
if(j==NPphi)
{
  Phichangel = k3*deltaT*(vn[i][l][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
  }
else
{
  Phichangel = k3*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
  }
Zchangel = k3*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/pow(deltaz,2);

/****Prepare the coefficient for Thomas way*************/

f[i][j][z]=Rchangel+Phichangel+zchangel+2*deltaT*Q2[i][j][z]+(2*p2*qc-wb*cb*deltaT)*vn[i][j][z];
b[i][z]=k3*deltaT/(deltaz*deltaz);
a[i][z]=2*p2*qc-wb*cb*deltaT/(4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));
c[i][j][z]=k3*deltaT/(deltaz*deltaz);
d[i][j][z]=f[i][j][z]+Rchangel+Phichangel+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));
}
a[NZ3-1]=a[NZ3-1]-c[NZ3-1];
c[NZ3-1]=0;
}

// tri-diagonal system

for(i=1;i<=NR-1;i++)
{
  for(j=1;j<=NPphi+1;j++)
  {
    v[i][j][NZ]=0.0;
    beta[i][j][NZ]=0.0;
    for(z=NZ3-1;z>=1;z--)
    {
      v[i][j][z]=v[i][j][z]+z*v[i][j+1][z]/(a[i][j+l][z]-c[i][j][z]*beta[i][j][z+1]);
      beta[i][j][z]=a[i][j][z]/(a[i][j][z]-c[i][j][z]*beta[i][j][z+1]);
    }
  }
}
for(i=1;i<=NR-1;i++)
{
    for(j=1;j<=NPhi;j++)
    {
        for(z=1;z<=NZ3-1;z++)
        {
            vnew[i][j][z] = v[i][j][z] + beta[i][j][z]*vnew[i][j][z-1];
            judge = vnew[i][j][z] - vold[i][j][z];
            if(judge<0) judge = judge*(-1);
            if(judge>MaxErr) MaxErr = judge;
            vold[i][j][z] = vnew[i][j][z];
        }
    }
}
t++; cout<<"number"<<t<<" "<<"MaxErr"<<MaxErr<<endl;
for(0;i<=NR;i++)
{
    for(j=1;j<=NPhi;j++)
    {
        for(z=1;z<=NZ3-1;z++)
        {
            vnew[i][j][0] = vnew[i][j][1];
            vnew[i][j][NZ3] = vnew[i][j][NZ3-1];
            vnew[i][0][z] = vnew[i][NPhi][z];
            vnew[0][j][z] = vnew[1][j][z];
            vnew[NR][j][z] = vnew[NR-1][j][z];
            vold[i][j][z] = vnew[i][j][z];
        }
    }
}

///// five points code here /////
point[0] = vnew[0][0][0];
point[1] = vnew[NR][0][0];
point[2] = vnew[NR][3*NPhi/4][0];
point[3] = vnew[NR][NPhi/2][0];
point[4] = vnew[NR][NPhi/4][0];
FileWrite(nt);
return vnew[0][0][0];
}

double zlihcp::RunAll(double P0) //
{
    double TemRet = 0;
    Clear();
    TemRet = IntmTrsy1(P0);
    return TemRet;
}

int main(void)
{
    /////////////////
    zlihcp zl(NR+1,NPhi+1,NZ3+1);
    long double P0m,T1m,T2m, deltaP;
    long double S, Snew, Pnew, err1;
    double rec0[5], rec1[5], X[5], Scale1, Scale2;
    double Tpoint = CenTemp, Tpointa = EndTemp;
    int i;
    ofstream fout14;
    fout14.open("time.txt", ios::out);
    Pnew=16.6159;
    T1m = 0;
    T2m = 0;
    S = 0;
    Snew = 0;

    for(i=1;i<=NR-1;i++)
    {
        for(j=1;j<=NPhi;j++)
        {
            for(z=1;z<=NZ3-1;z++)
            {
                vnew[i][j][z] = v[i][j][z] + beta[i][j][z]*vnew[i][j][z-1];
                judge = vnew[i][j][z] - vold[i][j][z];
                if(judge<0) judge = judge*(-1);
                if(judge>MaxErr) MaxErr = judge;
                vold[i][j][z] = vnew[i][j][z];
            }
        }
    }
}
t++; cout<<"number"<<t<<" "<<"MaxErr"<<MaxErr<<endl;
for(0;i<=NR;i++)
{
    for(j=1;j<=NPhi;j++)
    {
        for(z=1;z<=NZ3-1;z++)
        {
            vnew[i][j][0] = vnew[i][j][1];
            vnew[i][j][NZ3] = vnew[i][j][NZ3-1];
            vnew[i][0][z] = vnew[i][NPhi][z];
            vnew[0][j][z] = vnew[1][j][z];
            vnew[NR][j][z] = vnew[NR-1][j][z];
            vold[i][j][z] = vnew[i][j][z];
        }
    }
}

///// five points code here /////
point[0] = vnew[0][0][0];
point[1] = vnew[NR][0][0];
point[2] = vnew[NR][3*NPhi/4][0];
point[3] = vnew[NR][NPhi/2][0];
point[4] = vnew[NR][NPhi/4][0];
FileWrite(nt);
return vnew[0][0][0];
}

double zlihcp::RunAll(double P0) //
{
    double TemRet = 0;
    Clear();
    TemRet = IntmTrsy1(P0);
    return TemRet;
}

int main(void)
{
    /////////////////
    zlihcp zl(NR+1,NPhi+1,NZ3+1);
    long double P0m,T1m,T2m, deltaP;
    long double S, Snew, Pnew, err1;
    double rec0[5], rec1[5], X[5], Scale1, Scale2;
    double Tpoint = CenTemp, Tpointa = EndTemp;
    int i;
    ofstream fout14;
    fout14.open("time.txt", ios::out);
    Pnew=16.6159;
    T1m = 0;
    T2m = 0;
    S = 0;
    Snew = 0;

    for(i=1;i<=NR-1;i++)
    {
        for(j=1;j<=NPhi;j++)
        {
            for(z=1;z<=NZ3-1;z++)
            {
                vnew[i][j][z] = v[i][j][z] + beta[i][j][z]*vnew[i][j][z-1];
                judge = vnew[i][j][z] - vold[i][j][z];
                if(judge<0) judge = judge*(-1);
                if(judge>MaxErr) MaxErr = judge;
                vold[i][j][z] = vnew[i][j][z];
            }
        }
    }
}
t++; cout<<"number"<<t<<" "<<"MaxErr"<<MaxErr<<endl;
for(0;i<=NR;i++)
{
    for(j=1;j<=NPhi;j++)
    {
        for(z=1;z<=NZ3-1;z++)
        {
            vnew[i][j][0] = vnew[i][j][1];
            vnew[i][j][NZ3] = vnew[i][j][NZ3-1];
            vnew[i][0][z] = vnew[i][NPhi][z];
            vnew[0][j][z] = vnew[1][j][z];
            vnew[NR][j][z] = vnew[NR-1][j][z];
            vold[i][j][z] = vnew[i][j][z];
        }
    }
}

///// five points code here /////
point[0] = vnew[0][0][0];
point[1] = vnew[NR][0][0];
point[2] = vnew[NR][3*NPhi/4][0];
point[3] = vnew[NR][NPhi/2][0];
point[4] = vnew[NR][NPhi/4][0];
FileWrite(nt);
return vnew[0][0][0];
}
errorl=0.001;
P0m=16.6159;
zl.RunAlll(P0m);
//fout14<<"LEAST SUM SQUARE "<<zl.LSS<<endl;
fout14<<"4 LEAST SUM SQUARE "<<zl.LSS_4<<endl;
for(i=0;i<100;i++)
{
  if(zl.TimeRec[i]!=-1)
  {
    if(zl.FlagRec[i]==0)
      fout14<<"Number "<<i<<" CoolTime "<<zl.TimeRec[i]<<endl;
    if(zl.FlagRec[i]==1)
      fout14<<"Number "<<i<<" HeatTime "<<zl.TimeRec[i]<<endl;
    if(zl.FlagRec[i]==2)
      fout14<<"Number "<<i<<" EndTime "<<zl.TimeRec[i]<<endl;
  }
}
do
{
  P0m=Pnew;
  deltaP=P0m/100;
  S=Snew;
  T1m=zl.RunAlll(P0m);
  for (i=0;i<5;i++)
    rec0[i] = zl.point[i];
  T2m=zl.RunAlll(P0m+deltaP);
  for (i=0;i<5;i++)
    rec1[i] = zl.point[i];
  /*Compute the Coefficient*/
  X[0] = (rec1[0]-rec0[0])/deltaP;
  X[1] = (rec1[1]-rec0[1])/deltaP;
  X[2] = (rec1[2]-rec0[2])/deltaP;
  X[3] = (rec1[3]-rec0[3])/deltaP;
  X[4] = (rec1[4]-rec0[4])/deltaP;
  Scale1= pow(X[0],2)+pow(X[1],2)+pow(X[2],2)+pow(X[3],2)+pow(X[4],2);
  Scale2=X[0]*(Tpoint-rec0[0])+X[1]*(Tpointa-rec0[1])+X[2]*(Tpointa-rec0[2])
    +X[3]*(Tpointa-rec0[3])+X[4]*(Tpointa-rec0[4]);
  Pnew = P0m+Scale2/Scale1;
  Snew = pow((Tpoint-rec1[0],2)/(CenTemp*CenTemp)+pow((Tpointa-rec1[1],2)/(EndTemp*EndTemp)
    +pow((Tpointa-rec1[2],2)/(EndTemp*EndTemp)+pow((Tpointa-rec1[3],2)/(EndTemp*EndTemp)
    +pow((Tpointa-rec1[4],2)/(EndTemp*EndTemp);
  fout14<<"PNEW "<<Pnew<<endl;
  fout14<<"LEAST SQUARE SUM "<<Snew<<endl;
}
while ((Snew-S)/Snew > error1);
fout14<<"END"<<endl;
fout14.close();
return 0;
APPENDIX B

SOURCE CODE FOR SOLVING THE 3D SKIN STRUCTURE EMBEDDED WITH A BLOOD VESSEL
/*Table B.1 Program 1: Source code of step 1 in Figure 5.3 is used for the skin model with a blood vessel.
Le Zhang
4/11/05
This program is about heat transfer in the skin of a human being.
There are three layers in the skin. The first layer is epidermis, the second one is dermis and the last one is Subcutaneous.
*/
#include <fstream.h>
#include <string.h>
#include <iostream.h>
#include <math.h>
#include<stdio.h>
define NZ1 8
#define NZ2 208
#define NZ3 1208
#define NR 30
#define NPhi 20
#define BLOODTEMP 1
#define Hi 2
class zlihcp
{
private:
    double ***Q1, ***Q2, ***Q3;
    double ***v, ***vnew, ***vold, ***vn;
    double ***beta, ***f, ***d;
    double *b, **a, *c;
    double MaxErr, h, e;
    double deltaZ, deltaT, deltaPhi, deltaR;
    double Rchange, Phichange, Zchange, Rchanged, Phichanged, Z changed;
    double p1, p2, p3, q1, q2, qc1, qc2, qc3, k1, k2, k3, wb1, wb2, wb3, ch1, ch2, ch3;
    double Sigma, Alpha1, Alpha2, Alpha3, Ref1, Ref2, Ref3;
    double P0, psijudge, CenterX, CenterY;
    double BSpeed, BP, BF, BAlpha, BCh, Bratio1, Bratio2, Ubl[NZ3-NZ2+1], Ub1[NZ3-NZ2+1];
    int i, j, t, n, nt;
    int MaxLen, MaxWid, MaxHig, loop1, count1, VR;
public:
    zlihcp(int l, int w, int high1)
    {
        int i, j, k;
        MaxLen = l;
        MaxWid = w;
        MaxHig = high1;
        a = new double *[l];
        for (j = 0; j < l + 1)
            a[j] = new double [high1];
        b = new double [high1];
        c = new double [high1];
        Q1 = new double *[l];
        Q2 = new double *[l];
        Q3 = new double *[l];
        v = new double *[l];
        vnew = new double *[l];
        vold = new double *[l];
        vn = new double *[l];
        f = new double *[l];
        d = new double *[l];
        for (j = 0; j < l + 1)
            {
                Q1[j] = new double *[w];
                Q2[j] = new double *[w];
                Q3[j] = new double *[w];
                v[j] = new double *[w];
                vnew[j] = new double *[w];
                vold[j] = new double *[w];
                vn[j] = new double *[w];
                beta[j] = new double *[w];
                f[j] = new double *[w];
                d[j] = new double *[w];
            }
for (k=0;k<h;k++)
{ 
    Q1[j][k] = new double [high1];
    Q2[j][k] = new double [high1];
    Q3[j][k] = new double [high1];
    v[j][k] = new double [high1];
    vnew[j][k] = new double [high1];
    vold[j][k] = new double [high1];
    beta[j][k] = new double [high1];
    f[j][k] = new double [high1];
    d[j][k] = new double [high1];
}

for(i=0;i<l;i++)
{
    for(j=0;j<w;j++)
    {
        for(k=0;k<h;k++)
        { 
            Q1[i][j][k]=0;
            Q2[i][j][k]=0;
            Q3[i][j][k]=0;
            v[i][j][k]=0;
            vnew[i][j][k]=0;
            vold[i][j][k]=0;
            beta[i][j][k]=0;
            f[i][j][k]=0;
            d[i][j][k]=0;
        }
    }
}

loop1 = 10; count1 = 1; Sigma= 0.1;
Alpha1=1.0; Alpha2=0.8; Alpha3=0.4;
Ref1=0.93; Ref2=0.93; Ref3=0.93;
p1=3.14159265358979;
CenterX = 0; CenterY = 0;
t=2; n=10;
p1=1.2; p2=1.2; p3=1.0;
qc1=3.6; qc2=3.4; qc3=3.06;
k1=0.0026; k2=0.0052; k3=0.0021;
w1=0.0; w2=0.0005; w3=0.0005;
cb1=0.0; cb2=4.2; cb3=4.2;
e=0.001;
deltaPhi = double(2*pi/(double)NPhi);
deltaR = double(0.5/(double)NR);
deltaZ=0.001; deltaT=0.1;
//Blood parameter
VR = 2;
BSpeed = 80;
BP = 2*pi*deltaR*double(VR);
BF = pi*deltaR*double(VR)*deltaR*double(VR);
BCb = 0.004134;
BAlpha = 0.002;
Bratio1 = 1/(BCb*BSpeed);
Bratio2 = BAlpha*BP/(BCb*BSpeed*BF);
}
~zlihec()
{
    delete []a;
    delete []b;
    delete []c;
    delete []Q1;
    delete []Q2;
    delete []Q3;
delete []v;
delete []vnew;
delete []vold;
delete []vn;
delete []beta;
delete []f;
delete []d;
};
void InitQ(double,double,double);
double Intm T rsy(double);.
void Vessel(void);
void FileWrite(int);
void Clear();

void zlihcp::Vessel(void)
{
  /* It is used to calculate the blood vessel.
     Runge-Kutta (order four) is applied for the function.
     All of the variable has a prefix RF
     The equation is like Cb*w*Dtb/Dz = -Alpha*P*(Tb-Tw)
     The entry temperature is 10 centigrade.
   */
  int RFi,RFj,RFk;
  double RFw[NZ3-NZ2+1],RFh,RFz,RFk1,RFk2,RFk3,RFk4,RFTw;
  for (RFi=0;RFi<=NZ3-NZ2;RFi++)
    Ub1[RFi]=RFw[RFi]=0;
  RFi = 0;
  RFh = deltaZ;
  RFz = 0;
  RFw[RFi] = BLOODTEMP;
  for (RFi=1;RFi<=NZ3-NZ2-1;RFi++)
    { /* Get the average wall temperature from the tissue part
        Here we only choose the four angle points average temperature
      */
      RFTw = void[VR][NPhi/4][NZ3+1-RFi] + void[VR][NPhi/2][NZ3+1-RFi];
      RFTw = RFTw/4;
      ubw[RFi] = RFTw;
      // Solve the Runge-Kutta equation
      RFk1 = RFh*(Bratio1*Q3[0][0][NZ3-(RFi-1)]-Bratio2*(RFw[RFi-1]-RFTw));
      RFk2 = RFh*(Bratio1*Q3[0][0][NZ3-(RFi-1)]-Bratio2*(RFw[RFi-2]-RFTw));
      RFk3 = RFh*(Bratio1*Q3[0][0][NZ3-(RFi-1)]-Bratio2*(RFw[RFi-1]+RFk1/2-RFTw));
      RFk4 = RFh*(Bratio1*Q3[0][0][NZ3-(RFi-1)]-Bratio2*(RFw[RFi-1]+RFk2/2-RFTw));
      RFw[RFi] = RFw[RFi-1]+(RFk1+2*RFk2+2*RFk3+RFk4)/6;
      RFz = RFh*RFw;
    }
  // Receive the data from the function
  for (RFj=0;RFj<=NZ3-NZ2;RFj++)
    { for (RFi=0;RFi<=NZ3-NZ2-1;RFi++)
      Ub1[NZ3-NZ2] = RFw[NZ3-NZ2+1];
      for (RFi=0;RFi<=VR-1;RFi++)
        { for (RFj=0;RFj<=NPhi;RFj++)
          { for (RFk=NL2+1;RFk<=NZ3;RFk++)
            void[RF][RFj][RFk] = Ub1[NZ3-RFk];
          }
        }
    }
}

void zlihcp::FileWrite(int time1)
{
  int ik; ofstream fout1,fout2,fout21,fout3,fout31;
  sprintf(str,"%d",time1); strcat(str1,str); strcat(str2,str); strcat(str1,str);
  fout1.open(str1,ios::out);
  fout1<<" TITLE = "Example: Simple ZT-Volume Data" <<endl;
foutil << VARIABLES = "Z", "Temperature" <<endl;
foutil << ZONE "1209,F=POINT" <<endl;
for(k=0;k<N3;z++)
  foutil << double(k*deltaZ) << "<<<new[0][0][k][10][k]<<<<endl;
foutil close();

/////////////contour curve///////////
foutil2.open(str2,ios::out);
foutil2 << TITLE = "Example: Simple 2D-Volume Data" "<<<<endl;
foutil2 << VARIABLES = "R", "Z", "Temperature" <<endl;
foutil2 << ZONE "right", J=1209, J=31, F=POINT" <<endl;
for(i=0;i<=NR;i++)
  }
  foutil2 << double(i*deltaR) << "<<<new[i][0][0][k][10][k]<<<<endl;
}
for(k=0;k<=N3;z++)
  foutil2 << double(k*deltaZ) << "<<<new[i][0][0][k][10][k]<<<<endl;
}
foutil2.close();

///////////////////Contour curve cross///////////
foutil21.open(str21,ios::out);
foutil21 << TITLE = "Example: Simple 2D-Volume Data" "<<<<endl;
foutil21 << VARIABLES = "R", "Z", "Temperature" <<endl;
foutil21 << ZONE "right", J=1209, J=31, F=POINT" <<endl;
for(i=0;i<=NR;i++)
  }
  foutil21 << double(i*deltaR) << "<<<new[i][N Phi/4][4][k][10][k]<<<<endl;
}
for(k=0;k<=N3;z++)
  foutil21 << double(k*deltaZ) << "<<<new[i][N Phi/4][4][k][10][k]<<<<endl;
}
foutil21.close();

/////////////Center///////////
foutil3.open(str3,ios::out);
foutil3 << TITLE = "Example: Simple 2D-Volume Data" "<<<<endl;
foutil3 << VARIABLES = "R", "Temperature" <<endl;
foutil3 << ZONE "right", J=31, F=POINT" <<endl;
for(i=0;i<=NR;i++)
  foutil3 << double(i*deltaR) << "<<<new[i][0][0][k][10][k]<<<<endl;
}
for(k=0;k<=N3;z++)
  foutil3 << double(k*deltaZ) << "<<<new[i][0][0][k][10][k]<<<<endl;
}
foutil3.close();

///////////////////Center cross///////////
foutil31.open(str31,ios::out);
foutil31 << TITLE = "Example: Simple 2D-Volume Data" "<<<<endl;
foutil31 << VARIABLES = "R", "Temperature" <<endl;
foutil31 << ZONE "right", J=31, F=POINT" <<endl;
for(i=0;i<=NR;i++)
  foutil31 << double(i*deltaR) << "<<<new[i][N Phi/4][4][k][10][k]<<<<endl;
}
for(k=0;k<=N3;z++)
  foutil31 << double(k*deltaZ) << "<<<new[i][N Phi/4][4][k][10][k]<<<<endl;
}
foutil31.close();
void zlihcp::InitQ(double PO,double Cr,double Cp) // Initialize the laser power:
{
    int i,j,z;
    CenterX = Cr * cos(Cp); CenterY = Cr * sin(Cp);
    for(i=0;i<=NR;i++)
    {
        for(j=0;j<NPphi;j++)
        {
            for(z=0;z<=NZ1;z++)
            {
                Q1[i][j][z] = Alpha1 * exp(-Alpha1 * z * deltaZ / (sqrt(2*pi) * Sigma) * exp(-pow((i*cos(j*deltaPhi) * deltaR - CenterX,2) + pow((i*sin(j*deltaPhi) * deltaPhi - CenterY,2))) / (2 * Sigma * Sigma)) * P0 * (l - Refl1);
            }
        }
    }
}

double zlihcp::IntmTrsy(double PO) // Time Iteration and Tri-diagonal system
{
    MaxErr=1.0; nt=0;
    while(++nt<=(NPphi+1) * 20 - 10)
    {
        if(nt == 1)
            InitQ(PO,0,0);
        t=0;
        for(i=0;i<=NR;i++)
        {
            for(j=0;j<NPphi;j++)
            {
                for(z=0;z<=NZ3;z++)
                {
                    vn[i][j][z]=vold[i][j][z];
                }
            }
        }
        MaxErr=1.0;
        while(MaxErr>=e)
        {
            Vessel();
            MaxErr=0.0;
            for(i=1;i<=NR-1;i++)
            {
                for(j=1;j<NPphi;j++)
                {
                    // The first layer
                    Rchange = k1 * deltaT *(vold[i][j+1][z] - 2 * vold[i][j][z] + vold[i][j-1][z]) / (i * deltaR * deltaR);
                    Phichange = k1 * deltaT * (vold[i][j+1][z] - 2 * vold[i][j][z] + vold[i][j-1][z]) / pow(i*deltaPhi,2);
                    Zchange = k1 * deltaT * (vold[i][j][z+1] - 2 * vold[i][j][z] + vold[i][j][z-1]) / pow(deltaZ,2);
                    Rchange1 = k1 * deltaT *(vn[i][j+1][z] - 2 * vn[i][j][z] + vn[i][j-1][z]) / (i * deltaR * deltaR);
                    Phichange1 = k1 * deltaT * (vn[i][j+1][z] - 2 * vn[i][j][z] + vn[i][j-1][z]) / pow(i*deltaPhi,2);
                }
            }
        }
    }
}
\[
\begin{align*}
\text{Phichange1} &= k1*\text{deltaT}^*\text{vn}[i][j][0][z-2]*\text{vn}[i-1][j][z]+\text{vn}[i][j][1][z])\text{pow}(i*\text{deltaR}^*\text{deltaPhi},2); \\
\text{Zchange1} &= k1*\text{deltaT}^*\text{vn}[i][j][0][z-1]*2*\text{vn}[i-1][j][z]+\text{vn}[i][j][1][z]+\text{vn}[i][j][1][z])\text{pow}(\text{deltaZ},2); \\
\end{align*}
\]

//Prepare the coefficient for Thomas way\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\"
if (i<=VR)
    continue;
else
{  
    Rchange = k3*deltaT*((i+0.5)*vold[i+1][j][z]-2*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/deltaR*deltaR;
    if(j==NPhi)
        {  
            Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                }
    else
        {  
            Phichange = k3*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z-1])/pow(i*deltaR*deltaPhi,2);
                }  
    Zchange = k3*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z-1])/pow(deltaZ,2);
    // The n state // ///////////////////////////////////////////  
    Rchange1 = k3*deltaT*((i+0.5)*vn[i+1][j][z]-2*vn[i][j][z]+vn[i-1][j][z])/pow(i*deltaR*deltaR,2);
    if(j==NPhi)
        {  
            Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                }
    else
        {  
            Phichange1 = k3*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/pow(i*deltaR*deltaPhi,2);
                }
    Zchange1 = k3*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/pow(deltaZ,2);
    // Prepare the coefficient for Thomas way///////////////////////////////  
    b[i][j]=Rchange+Phichange+Zchange1+2*deltaT*Q[vold[i][j][z]+(2*p3*qc3+wh3*ch3*deltaT)*vn[i][j][z]+2*wh3*ch3*deltaT*Ubl[NZ3-z];
    a[i][j]=2*p3*qc3+wh3*ch3*deltaT+2*k3*deltaT/pow(deltaZ,2);
    c[i][j]=k3*deltaT/pow(deltaZ,2);
    for(i=1;i<=NR-1;i++)
    {  
        for(j=1;j<=NPhi;j++)
        {  
            for(z=NZ3-1;z>=1;z--)
            {  
                if((z>=NZ2)&& (i<=VR))
                    continue;
                else
                    {  
                        vnew[i][j][z]=v[i][j][z]+c[i][j][z]*beta[i][j][z+1];  
                        beta[i][j][z]=b[i][j][z]/(a[i][j][z]+c[i][j][z]*beta[i][j][z+1]);
                    }
            }
        }
    }
    for(i=1;i<=NR-1;i++)
    {  
        for(j=1;j<=NPhi;j++)
        {  
            for(z=1;z<=NZ3-1;z++)
            {  
                if(z>=NZ2)&& (i<=VR))
                    continue;
                else
                    {  
                        vnew[i][j][z]=v[i][j][z]+c[i][j][z]*beta[i][j][z+1];  
                        judge=vnew[i][j][z]-vold[i][j][z];
                        if(judge<0)
                            judge = judge*(-1);
                    }
            }
        }
    }
}
if(judge>MaxErr)
    MaxErr=judge;
void[i][j][z]=vnew[i][j][z];
}
}
t++; cout<<"number"<<""<<"MaxErr"<<MaxErr<<endl;
for(i=0;i<=VR-1;++i)
    { 
    for(j=0;j<=NPhi;++j)
        { 
    for(z=0;z<=NZ3;++z)
        
        vnew[i][j][z]=Ub1[NZ3-z];
    }
}
for(i=0;i<=NR;++i)
    { 
    for(j=0;j<=NPhi;++j)
        { 
    for(z=0;z<=NZ3;++z)
        { 
        vnew[i][j][0]=vnew[i][j][1]; 
        if(i>=VR+1) 
        vnew[i][j][NZ3]=vnew[i][j][NZ3-1]; 
        vnew[i][j][0][z]=vnew[i][j][NPhi][z]; 
        if(z>=NZ2) 
        vnew[VR][j][z]=(vnew[VR+1][j][z]+Bi*deltaR*vnew[0][j][z])/(1+deltaR*Bi); 
        else 
        vnew[0][j][z]=vnew[1][j][z]; 
        }
    }
    
    vnew[i][j][NZ2]=Ub1[NZ3-NZ2];
    vnew[NR][j][z]=vnew[NR-1][j][z];
    void[i][j][z]=vnew[i][j][z];
    }
}
if(nt==10)InitQ(P0,deltaR,1*deltaPhi);
if(nt==20)InitQ(P0,deltaR,2*deltaPhi);
if(nt==30)InitQ(P0,deltaR,3*deltaPhi);
if(nt==40)InitQ(P0,deltaR,4*deltaPhi);
if(nt==50)InitQ(P0,deltaR,5*deltaPhi);
if(nt==60)InitQ(P0,deltaR,6*deltaPhi);
if(nt==70)InitQ(P0,deltaR,7*deltaPhi);
if(nt==80)InitQ(P0,deltaR,8*deltaPhi);
if(nt==90)InitQ(P0,deltaR,9*deltaPhi);
if(nt==100)InitQ(P0,deltaR,10*deltaPhi);
if(nt==110)InitQ(P0,deltaR,11*deltaPhi);
if(nt==120)InitQ(P0,deltaR,12*deltaPhi);
if(nt==130)InitQ(P0,deltaR,13*deltaPhi);
if(nt==140)InitQ(P0,deltaR,14*deltaPhi);
if(nt==150)InitQ(P0,deltaR,15*deltaPhi);
if(nt==160)InitQ(P0,deltaR,16*deltaPhi);
if(nt==170)InitQ(P0,deltaR,17*deltaPhi);
if(nt==180)InitQ(P0,deltaR,18*deltaPhi);
if(nt==190)InitQ(P0,deltaR,19*deltaPhi);
if(nt==200)InitQ(P0,deltaR,20*deltaPhi);
if(nt==210)InitQ(P0,0,0);
if(nt==220)InitQ(P0,deltaR,1*deltaPhi);
if(nt==230)InitQ(P0,deltaR,2*deltaPhi);
if(nt==240)InitQ(P0,deltaR,3*deltaPhi);
if(nt==250)InitQ(P0,deltaR,4*deltaPhi);
if(nt==260)InitQ(P0,deltaR,5*deltaPhi);
if(nt==270)InitQ(P0,deltaR,6*deltaPhi);
if(nt==280)InitQ(P0,deltaR,7*deltaPhi);
if(nt==290)InitQ(P0,deltaR,8*deltaPhi);
if(nt==300)InitQ(P0,deltaR,9*deltaPhi);
if(nt==310)InitQ(P0,deltaR,10*deltaPhi);
if(nt==320)InitQ(P0,deltaR,11*deltaPhi);
if(nt==330)InitQ(P0,deltaR,12*deltaPhi);
if(nt==340)InitQ(P0,deltaR,13*deltaPhi);
if(nt==350)InitQ(P0,deltaR,14*deltaPhi);
if(nt==360)InitQ(P0,deltaR,15*deltaPhi);
if(nt==370)InitQ(P0,deltaR,16*deltaPhi);
if(nt==380)InitQ(P0,deltaR,17*deltaPhi);
if(nt==390)InitQ(P0,deltaR,18*deltaPhi);
if(nt==400)InitQ(P0,deltaR,19*deltaPhi);
if(nt==410)InitQ(P0,deltaR,20*deltaPhi);
cout<<"temperature"<<vn[0][0][0]<<endl;
}
FileWrite(nt);
return vn[0][0][0];
}

void zlihcp::Clear(void)
{
    int i,j,k;
    for(i=0;i<NR+1;i++)
    {
        for(j=0;j<NPhi+1;j++)
        {
            for(k=0;k<NZ3+1;k++)
            {
                Q1[i][j][k]=0;
                Q2[i][j][k]=0;
                Q3[i][j][k]=0;
                v[i][j][k]=0;
                vnew[i][j][k]=0;
                void[i][j][k]=0;
                vn[i][j][k]=0;
                beta[i][j][k]=0;
                f[i][j][k]=0;
                d[i][j][k]=0;
            }
        }
    }
}

double zlihcp::RunAll(double PO) //
{
    double TemRet = 0;
    Clear(); TemRet = IntmTrsy(PO);
    return TemRet;
}

int main(void)
{
    zlihcp zl(NR+1,NPhi+1,NZ3+1);
    long double P0m, T1m, T2m, deltaP, X,Tpoint;
    long double S, Snew, error1, judge1;
    ofstream fout14; fout14.open("pnew.txt",ios::out);
    P0m=16.0; Pnew=16.0; T1m=0; T2m=0;
    Tpoint=8; S=0; Snew=0; error1=0.001;
    do
    {
        P0m=Pnew; deltaP=P0m/100;
        S=Snew; T1m=zl.RunAll(P0m);
        T2m=zl.RunAll(P0m+deltaP);
        X=(T2m-T1m)*deltaP;
        Pnew = P0m+X/(X*X)*(Tpoint-T1m);
        Snew = (Tpoint-T1m)*X/(X*X);
        fout14<<Pnew<<endl;
        judge1 = Snew-S;
    } while (judge1/Snew > error1);
    fout14.close();
    return 0;
}
/*Table B.2 Program 2: Source code of step 2 in Figure 5.3 is used for the skin model with a blood vessel.

Le Zhang
4/11/05
This program is about heat transfer in the skin of a human being. There are three layers in the skin. The first layer is epidermis, the second on is dermis and the last one is Subcutaneous.
*/

#include <fstream.h>
#include <string.h>
#include <iostream.h>
#include <math.h>
#include <stdio.h>
#define NZ1 8
#define NZ2 208
#define NZ3 1208
#define NR 30
#define NPhi 20
#define CIRCLE 1
#define EndTemp 3
#define CenTemp 8
#define LSS4 0.04
#define Bi 2
#define BLOODTEMP 1

class zlihcp
{
private:
    double ***Q1,***Q2,***Q3;
    double ***v,***vnew,***vold,***vn;
    double ***beta,***f,***d;
    double *b,**a,*c;
    double MaxErr,h,e;
    double deltaZ,deltaT,deltaPhi,deltaR;
    double R change, Phichange, Zchange, R changel, P hic h ange2, Z ch ange2;
    double p1 ,p2,p3,p4,p5,p6,p7,p8;
    double Sigma,A l pha1,Alpha2,Alpha3,Refl,Refl2,Refl3;
    double P0; p1,judge,CenterX,CenterY ;
    double BSpeed,BP ,BF,Alpha,B ratio2,Uwl[NZ3-NZ2+1],Ubl[NZ3-NZ2+1];
    int i,j,k,t,n;
    int M axLen,M axW id,M axH ig,VR;

public:
    double point[5];
    double LSS,MLSS;
    double LSS_4,LSS1,MLSS1;
    int TimeRec[100],FlagRec[100],int,flag,CountNum;
    zlihcp(int i, int w , int highl)
    {
        int i,j;
        MaxLen = 1;
        MaxWid = w ;
        MaxHig = highl;
        CountNum = 0;
        LSS=0;
        MLSS=10000000;
        flag = 0;
        for (i=0;i<100;i++)
            TimeRec[i]=FlagRec[i] = -1;
        a = new double *[i];
        for(i=0;i<j;++i)
            a[i] = new double [highl];
        b = new double [highl];
        c = new double [highl];
        Q1 = new double *[i];
        Q2 = new double *[i];
        Q3 = new double *[i];
        v = new double *[i];
        vnew = new double *[i];
        vold = new double *[i];
        vn = new double *[i];

    }
beta = new double *[1];
f = new double *[1];
d = new double *[1];
for (j=0;j<l;j++)
{
  Q1[j] = new double *[w];
  Q2[j] = new double *[w];
  Q3[j] = new double *[w];
  v[j] = new double *[w];
  vnew[j] = new double *[w];
  vn[j] = new double *[w];
  beta[j] = new double *[w];
  f[j] = new double *[w];
  d[j] = new double *[w];
  for (k=0;k<w;k++)
  {
    Q1[j][k] = new double [high1];
    Q2[j][k] = new double [high1];
    Q3[j][k] = new double [high1];
    v[j][k] = new double [high1];
    vnew[j][k] = new double [high1];
    vn[j][k] = new double [high1];
    beta[j][k] = new double [high1];
    f[j][k] = new double [high1];
    d[j][k] = new double [high1];
  }
}
~zlihcp()
{
    delete []a;
    delete []b;
    delete []c;
    delete []Q1;
    delete []Q2;
    delete []Q3;
    delete []v;
    delete [vnew;]
    delete []vold;
    delete []vn;
    delete []beta;
    delete []f;
    delete []d;
};

void InitQ(double,double,double);
double IntmTrsy(double);
double RunAll(double);
void Vessel(void);
void FileWrit(int);
void Clear(void);

void zlihcp::FileWrit(int time1)
{
    int i,k; ofstream foutl,fout2,fout3,fout31;
    sprintf(str,"%d",time1); strcat(strl,str); strcat(str2,str);
    strcat(str21,str); strcat(str3,str); strcat(str31,str);
    ///////////zt curve/////////////////////
    fout1.open(str1,ios::out);
    fout1<<"  TITLE = "Example: Simple ZT-Volume Data""<<endl;
    fout1<<"  VARIABLES = \"Z\", \"Temperature\""<<endl;
    fout1<<"  ZONE T=1209,F=POINT""<<endl;
    for(k=0;k<=NZ3;k++)
        fout1<<double(k*deltaZ)<<"  "<<vnew[0][0][k]""<<endl;
    fout1.close();

    /////////// contour curve ///////////contour curve
    fout2.open(str2,ios::out);
    fout2<<"  TITLE = "Example: Simple 2D-Volume Data""<<endl;
    fout2<<"  VARIABLES = \"R\", \"Z\", \"Temperature\""<<endl;
    fout2<<"  ZONE T=\"right\",J=1209, I=31, F=POINT""<<endl;
    for(i=0;i<=NR;i++)
        for(k=0;k<=NZ3;k++)
            fout2<<double(i*deltaR)<<"  "<<double(k*deltaZ)<<"  "<<vnew[i][0][k]""<<endl;
    fout2.close();

    ////////////C ontour curve cross///////////////
    fout21.open(str1,ios::out);
    fout21<<"  TITLE = "Example: Simple 2D-Volume Data""<<endl;
    fout21<<"  VARIABLES = \"R\", \"Z\", \"Temperature\""<<endl;
    fout21<<"  ZONE T=\"reverse\",J=1209, I=31, F=POINT""<<endl;
    for(i=0;i<=NR;i++)
        for(k=0;k<=NZ3;k++)
            fout21<<double(i*deltaR)<<"  "<<double(k*deltaZ)<<"  "<<vnew[i][NP2]/2][k]""<<endl;
    fout21.close();
void zlihcp::InitQ(double PO,double Cr,double Cp) // Initialize the laser power;
{
    int i,j,z; CenterX = Cr * cos(Cp); CenterY = Cr * sin(Cp);
    for(i=0;i<=NR;i++)
    {
        for(j=0;j<=NPhi;j++)
        {
            for(z=0;z<=NZ1;z++)
            {
                Q1[i][j][z]= Alpha1*exp(-Alpha1*z*deltaZ)/(sqrt(2*pi)*Sigma)*exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*PO*(1-Ref1);
            }
            for(z=NZ1+1;z<=NZ2;z++)
            {
                Q2[i][j][z]= Alpha2*exp(-Alpha2*(z-NZ1)*deltaZ)*exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*PO*(1-Ref2);
            }
            for(z=NZ2+1;z<=NZ3;z++)
            {
                Q3[i][j][z]= Alpha3*exp(-Alpha3*(z-NZ2)*deltaZ)*exp(-Alpha1*deltaZ*NZ2)/(sqrt(2*pi)*Sigma)*exp(-(pow(i*cos(j*deltaPhi)*deltaR-CenterX,2)+pow(i*sin(j*deltaPhi)*deltaR-CenterY,2))/(2*Sigma*Sigma))*PO*(1-Ref3);
            }
        }
    }
}

void zlihcp::Vessel(void)
{
    /* It is used to calculate the blood vessel.
Runge-Kutta (order four) is applied for the function.
All of the variable has a prefix RF*/
The equation is like:
\[ C_b \cdot w \cdot F \cdot \frac{dT_b}{Dz} = -\alpha \cdot P \cdot (T_b - T_w) \]

The entry temperature is 10 centigrade.

```c
int RFi, RFj, RFk;
double RFw[N3-1=NZ2+1];
double RFh, RFz, RFk1, RFk2, RFk3, RFk4, RFTw;
for (RFi=0; RFi<=N3-NZ2; RFi++)
  Ub1[RFi]=RFw[RFi]=0;
RFi = 0;
RFh = deltaZ;
RFz = 0;
RFw[RFi] = BLOODTEMP;
for (RFi=1; RFi<=N3-NZ2+1; RFi++)
  { /* Get the average wall temperature from the tissue part */
    Here we only choose the four angle points average temperature */
    RFTw = vold[VR][NPphi/4][NZ3+1-RFi]+vold[VR][NPphi/2][NZ3+1-RFi]
      +vold[VR][3*NPphi/4][NZ3+1-RFi]+vold[VR][0][NZ3+1-RFi];
    RFTw = RFtw/4;
    // Solve the Runge-Kutta equation
    RFk1 = RFh*(-Bratio2*(RFw[RFi-l]-RFTw));
    RFk2 = RFh*(-Bratio2*(RFw[RFi-l]+RFk1/2-RFTw));
    RFk3 = RFh*(-Bratio2*(RFw[RFi-l]+RFk2/2-RFTw));
    RFk4 = RFh*(-Bratio2*(RFw[RFi-l]+RFk3-RFTw));
    RFw[RFi] = RFw[RFi-l]+(RFk1 + 2*RFk2 + 2*RFk3 + RFk4)/6;
    RFz = RFi*RFh;
  }
  // Receive the data from the function
  for (RFi=0;RFi<=N3-NZ2;RFi++)
    Ub1[RFi] = RFw[RFi];
for (RFi=0;RFi<=1=NZ2;RFi++)
  { /* Get the average wall temperature from the tissue part */
    Here we only choose the four angle points average temperature */
    RFTw = vold[VR][NPphi/4][NZ3+1-RFi]+vold[VR][NPphi/2][NZ3+1-RFi]
      +vold[VR][3*NPphi/4][NZ3+1-RFi]+vold[VR][0][NZ3+1-RFi];
    RFTw = RFtw/4;
    // Solve the Runge-Kutta equation
    RFk1 = RFh*(-Bratio2*(RFw[RFi-l]-RFTw));
    RFk2 = RFh*(-Bratio2*(RFw[RFi-l]+RFk1/2-RFTw));
    RFk3 = RFh*(-Bratio2*(RFw[RFi-l]+RFk2/2-RFTw));
    RFk4 = RFh*(-Bratio2*(RFw[RFi-l]+RFk3-RFTw));
    RFw[RFi] = RFw[RFi-l]+(RFk1 + 2*RFk2 + 2*RFk3 + RFk4)/6;
    RFz = RFi*RFh;
  }
  // Receive the data from the function
  for (RFi=0;RFi<=N3-NZ2;RFi++)
    Ub1[RFi] = RFw[RFi];
for (RFi=0;RFi<=1=NZ2;RFi++)
  { /* Get the average wall temperature from the tissue part */
    Here we only choose the four angle points average temperature */
    RFTw = vold[VR][NPphi/4][NZ3+1-RFi]+vold[VR][NPphi/2][NZ3+1-RFi]
      +vold[VR][3*NPphi/4][NZ3+1-RFi]+vold[VR][0][NZ3+1-RFi];
    RFTw = RFtw/4;
    // Solve the Runge-Kutta equation
    RFk1 = RFh*(-Bratio2*(RFw[RFi-l]-RFTw));
    RFk2 = RFh*(-Bratio2*(RFw[RFi-l]+RFk1/2-RFTw));
    RFk3 = RFh*(-Bratio2*(RFw[RFi-l]+RFk2/2-RFTw));
    RFk4 = RFh*(-Bratio2*(RFw[RFi-l]+RFk3-RFTw));
    RFw[RFi] = RFw[RFi-l]+(RFk1 + 2*RFk2 + 2*RFk3 + RFk4)/6;
    RFz = RFi*RFh;
  }
  // Receive the data from the function
  for (RFi=0;RFi<=N3-NZ2;RFi++)
    Ub1[RFi] = RFw[RFi];
for (RFi=0;RFi<=1=NZ2;RFi++)
  { /* Get the average wall temperature from the tissue part */
    Here we only choose the four angle points average temperature */
    RFTw = vold[VR][NPphi/4][NZ3+1-RFi]+vold[VR][NPphi/2][NZ3+1-RFi]
      +vold[VR][3*NPphi/4][NZ3+1-RFi]+vold[VR][0][NZ3+1-RFi];
    RFTw = RFtw/4;
    // Solve the Runge-Kutta equation
    RFk1 = RFh*(-Bratio2*(RFw[RFi-l]-RFTw));
    RFk2 = RFh*(-Bratio2*(RFw[RFi-l]+RFk1/2-RFTw));
    RFk3 = RFh*(-Bratio2*(RFw[RFi-l]+RFk2/2-RFTw));
    RFk4 = RFh*(-Bratio2*(RFw[RFi-l]+RFk3-RFTw));
    RFw[RFi] = RFw[RFi-l]+(RFk1 + 2*RFk2 + 2*RFk3 + RFk4)/6;
    RFz = RFi*RFh;
  }

double zlhecp::IntmTrys(double P0) // Time Iteration and Tri-diagonal systme
{  MaxErr=1.0;  nt=0;  MLSS=1000000;  MLSS1=100000;
while(nt<1)
  { /////////////Change center point///////////
    nt++;
    if(nt == 1)
      ImiQ(P0,0,0);
    t=0;
    for(i=0;i<=NR;i++)
      { for(j=0;j<=NPhiy;++j)
            { for(z=0;z<=NZ3;++z)
                  { vn[i][j][z]=vold[i][j][z];
                  }
            }
        }
    MaxErr=1.0;
    while(MaxErr>=e)
      { Vessel();
        MaxErr=0.0;
        for(i=1;i<=NR-1;++i)
          { for(j=1;j<=NPhij;++j)
            { }
          }
      }
  }
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The first layer

for (z = l; z <= NZ1 - 1; z++)

The n+1 state

Rchange = k1*deltaT*((i+0.5)*vold[i+1][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/(i*deltaR*deltaR);
if(j==NPphi)
{
    Phichange = k1*deltaT*(vold[i][j][z]+vold[i][j][z-1])/pow(i*deltaR*deltaPhi,2);
} else
{
    Phichange = k1*deltaT*(vold[i][j][z-1]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
Zchange = k1*deltaT*(vold[i][j][z-1]-2*vold[i][j][z]+vold[i][j-1][z])/pow(deltaZ,2);

The n state

Rchange = k1*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
if(j==NPphi)
{
    Phichange = k1*deltaT*(vn[i][j][z]+vn[i][j][z-1])/pow(i*deltaR*deltaPhi,2);
} else
{
    Phichange = k1*deltaT*(vn[i][j][z-1]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
Zchange = k1*deltaT*(vn[i][j][z-1]-2*vn[i][j][z]+vn[i][j-1][z])/pow(deltaZ,2);

Prepare the coefficient for Thomas way

f[i][j][z]= Rchange + Phichange + Zchange + 2*deltaT*Ql[i][j][z] + (2*pl*qcl-wbl*cbkl*deltaT)*vn[i][j][z];
b[z]=(kl*deltaT)/(deltaZ*deltaZ);
a[i][z]= 2*pl*qc1+wbl*cb1*deltaT+k1*deltaT*(d*+1)*(i*deltaR*deltaR);
Zchange = k1*deltaT*(vn[i][j][z-1]-2*vn[i][j][z]+vn[i][j-1][z])/pow(deltaZ,2);
d[i][j][z]= (k1*deltaT)/(deltaZ*deltaZ);
f[i][j][z]= Rchange+Phichange+k1*deltaT*((i+1)/(i*deltaR*deltaR))+(4*deltaPhi)*deltaR*Zchange+2*deltaT*vn[i][j][z];
}

The second layer

for (z = NZ1 - 1; z <= NZ2 - 1; z++)

The n+1 state

Rchange = k2*deltaT*((i+0.5)*vold[i+1][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/(i*deltaR*deltaR);
if(j==NPphi)
{
    Phichange = k2*deltaT*(vold[i][j][z]+vold[i][j][z-1])/pow(i*deltaR*deltaPhi,2);
} else
{
    Phichange = k2*deltaT*(vold[i][j][z-1]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
Zchange = k2*deltaT*(vold[i][j][z-1]-2*vold[i][j][z]+vold[i][j-1][z])/pow(deltaZ,2);

The n state

Rchange = k2*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
if(j==NPphi)
{
    Phichange = k2*deltaT*(vn[i][j][z]+vn[i][j][z-1])/pow(i*deltaR*deltaPhi,2);
} else
{
    Phichange = k2*deltaT*(vn[i][j][z-1]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
}
Zchange = k2*deltaT*(vn[i][j][z-1]-2*vn[i][j][z]+vn[i][j-1][z])/pow(deltaZ,2);

Prepare the coefficient for Thomas way

f[i][j][z]= Rchange+Phichange+k2*deltaT*(Q2[i][j][z]+(2*p2*qc2-wb2*cb2*deltaT)*vn[i][j][z];

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[
b[z] = k2*\Delta T/(\Delta Z*\Delta Z);
\]
\[
a[i][j]=2*p2*qc2+w b2*cb2*\Delta T+k2*\Delta T*((4*i+l)/(i*\Delta R*\Delta R)+4/pow(\Delta Phi*i*\Delta R,2)) +2*k2*\Delta T/pow(\Delta Z,2);
\]
\[
c[z] = k2*\Delta T/(\Delta Z*\Delta Z);
\]
\[
d[i][j][z]=f[i][j][z]+Rchange+Phichange+k2*\Delta T*((4*i+l)/(i*\Delta R*\Delta R)+4/pow(\Delta Phi*i*\Delta R,2)) *vold[i][j][z];
\]

```c
if (i>=VR+1)
{
    b[NZ2]=k2;
    a[i][NZ2]=k2+k3;
    c[NZ2]=k3;
    d[i][j][NZ2]=0;
}
if(i<=VR)
{
    // My code start here
    x = NZ2-1;
    a[i][NZ2-1]=2*p2*qc2+w b2*cb2*\Delta T+k2*\Delta T*((4*i+l)/(i*\Delta R*\Delta R)+4/pow(\Delta Phi*i*\Delta R,2)) +2*k2*\Delta T/pow(\Delta Z,2);
    c[NZ2-1]=0;
    d[i][j][NZ2-1]=f[i][j][z]+Rchange+Phichange+k2*\Delta T*((4*i+l)/(i*\Delta R*\Delta R)
    +4/pow(\Delta Phi*i*\Delta R,2)) *vold[i][j][z]+Ub1[NZ3-NZ2]*k2*\Delta T/(\Delta Z*\Delta Z);
}
```

```
// The third layer
for(i=NZ2+1;i<=NZ3-1;i++)
{
    if(i<=VR)
    continue;
    Rchange = k3*\Delta T*((i+0.5)*vold[i+1][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/(i*\Delta R*\Delta R);
    if(j==NPhi)
    Phichange = k3*\Delta T*(vold[i][j][z]-2*vold[i][j][z]+vold[i][j][z])/(i*\Delta R*\Delta R);
    else
    Phichange = k3*\Delta T*(vold[i][j][z]-2*vold[i][j][z]+vold[i][j][z])/(i*\Delta R*\Delta R);
    Zchange = k3*\Delta T*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][z])/(i*\Delta R*\Delta R);
    if(j==NPhi)
    Phichangel = k3*\Delta T*(vold[i][j][z]-2*vold[i][j][z]+vold[i][j][z])/(i*\Delta R*\Delta R);
    else
    Phichangel = k3*\Delta T*(vold[i][j][z]-2*vold[i][j][z]+vold[i][j][z])/(i*\Delta R*\Delta R);
    Rchangel = k3*\Delta T*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*\Delta R*\Delta R);
    if(j==NPhi)
    Phichangel = k3*\Delta T*(vn[i][j][z]-2*vn[i][j][z]+vn[i][j][z])/(i*\Delta R*\Delta R);
    else
    Phichangel = k3*\Delta T*(vn[i][j][z]-2*vn[i][j][z]+vn[i][j][z])/(i*\Delta R*\Delta R);
    Zchangel = k3*\Delta T*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z])/(i*\Delta R*\Delta R);

    // Prepare the coefficient for Thomas way
    f[i][j][z]=Rchangel+Phichangel+Zchangel+2*\Delta T*Q3[i][j][z]+2*\Delta T*Q3[i][j][z]+2*\Delta T*Q3[i][j][z]+2*\Delta T*Q3[i][j][z]+2*\Delta T*Q3[i][j][z]+2*\Delta T*Q3[i][j][z];
    b[z]=k3*\Delta T/(\Delta Z*\Delta Z);
    a[i][j]=2*p2*qc3+w b3*cb3*\Delta T+k3*\Delta T*((4*i+l)/(i*\Delta R*\Delta R)+4/pow(\Delta Phi*i*\Delta R,2)) +2*k3*\Delta T/pow(\Delta Z,2);
    c[z]=k3*\Delta T/(\Delta Z*\Delta Z);
    d[i][j][z]=f[i][j][z]+Rchange+Phichange+k3*\Delta T*((4*i+l)/(i*\Delta R*\Delta R)+4/pow(\Delta Phi*i*\Delta R,2)) *vold[i][j][z];
}
```

```c
a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1];
c[NZ3-1]=0;
```
\( v[i][j][NZ3]=0.0; \)
\( \beta[i][j][NZ3]=0.0; \)
for \( z=\text{NZ3}\rightarrow 1; z--> \)
\{ 
  if \((z>=\text{NZ2})\&\&(i<=VR))
  continue;
  else 
  \{ 
  \( v[i][j][z]=(d[i][j][z]+c[z]*v[i][j][z+1])/(a[i][z]-c[z]*\beta[i][j][z+1]); \)
  \( \beta[i][j][z]=b[z]/(a[i][z]-c[z]*\beta[i][j][z+1]); \)
  \}
\}
for \( i=1;i<=\text{NR}-1;i++ \)
\{ 
for \( j=1;j<=\text{NPhi};j++ \)
\{ 
for \( z=1;z<=\text{NZ3}-1;z++ \)
\{ 
  if \((z>=\text{NZ2})\&\&(i<=VR))
  continue;
  else 
  \{ 
  \( vnew[i][j][z]=v[i][j][z]+\beta[i][j][z]*vnew[i][j][z-1]; \)
  judge=(vnew[i][j][z]-vold[i][j][z]);
  if(judge<0) 
  judge = judge * (-1);
  if(judge>MaxErr) 
  MaxErr=judge;
  vold[i][j][z]=vnew[i][j][z];
  \}
}\}
++; cout<<"\text{number}<<"<<"\text{MaxErr}"<<MaxErr<<endl;
//\text{Boundary Condition}
for \( i=0;i<=\text{VR};i++ \)
\{ 
for \( j=0;j<=\text{NPhi};j++ \)
\{ 
for \( z=\text{NZ2}\rightarrow \text{NZ3};z++ \)
\{ 
  vnew[i][j][z]=Ub[\text{NZ3}-z];
  \}
\}
for \( i=0;i<=\text{NR};i++ \)
\{ 
for \( j=0;j<=\text{NPhi};j++ \)
\{ 
for \( z=0,z=\text{NZ3};z++ \)
\{ 
  vnew[i][j][0]=vnew[i][j][1];
  if \((z>=\text{VR}+1))
  vnew[i][j][NZ3]=vnew[i][j][NZ3-1];
  vnew[i][j][0]=(vnew[i][j][NPhij][z]);
  if \((z>=\text{NZ2}));
  vnew[VR][j][z]=(vnew[VR+1][j][z]+Bi*deltaR*vnew[0][j][z]+(1+deltaR*Bi));
  else 
  vnew[0][j][z]=vnew[1][j][z];
  if \((z<=\text{VR}+1))
  vnew[i][j][\text{NZ2}]=Ub[\text{NZ3}-\text{NZ2}];
  vnew[\text{NR}][j][z]=vnew[\text{NR}-1][j][z];
  vold[i][j][z]=vnew[i][j][z];
  \}
\}
\}
Point code here

point[0] = vnew[0][0][0];
point[1] = vnew[Nr][0][0];
point[2] = vnew[Nr][3*Phi/4][0];
point[3] = vnew[Nr][2*Phi/4][0];
point[4] = vnew[Nr][4*Phi/4][0];
if(nt==10)InitQ(Po,deltaR,1*deltaPhi);
if(nt==20)InitQ(Po,deltaR,2*deltaPhi);
if(nt==30)InitQ(Po,deltaR,3*deltaPhi);
if(nt==40)InitQ(Po,deltaR,4*deltaPhi);
if(nt==50)InitQ(Po,deltaR,5*deltaPhi);
if(nt==60)InitQ(Po,deltaR,6*deltaPhi);
if(nt==70)InitQ(Po,deltaR,7*deltaPhi);
if(nt==80)InitQ(Po,deltaR,8*deltaPhi);
if(nt==90)InitQ(Po,deltaR,9*deltaPhi);
if(nt==100)InitQ(Po,deltaR,10*deltaPhi);
if(nt==110)InitQ(Po,deltaR,11*deltaPhi);
if(nt==120)InitQ(Po,deltaR,12*deltaPhi);
if(nt==130)InitQ(Po,deltaR,13*deltaPhi);
if(nt==140)InitQ(Po,deltaR,14*deltaPhi);
if(nt==150)InitQ(Po,deltaR,15*deltaPhi);
if(nt==160)InitQ(Po,deltaR,16*deltaPhi);
if(nt==170)InitQ(Po,deltaR,17*deltaPhi);
if(nt==180)InitQ(Po,deltaR,18*deltaPhi);
if(nt==190)InitQ(Po,deltaR,19*deltaPhi);
if(nt==200)InitQ(Po,deltaR,20*deltaPhi);
if(nt==210)InitQ(Po,0,0);
if(nt==220)InitQ(Po,deltaR,1*deltaPhi);
if(nt==230)InitQ(Po,deltaR,2*deltaPhi);
if(nt==240)InitQ(Po,deltaR,3*deltaPhi);
if(nt==250)InitQ(Po,deltaR,4*deltaPhi);
if(nt==260)InitQ(Po,deltaR,5*deltaPhi);
if(nt==270)InitQ(Po,deltaR,6*deltaPhi);
if(nt==280)InitQ(Po,deltaR,7*deltaPhi);
if(nt==290)InitQ(Po,deltaR,8*deltaPhi);
if(nt==300)InitQ(Po,deltaR,9*deltaPhi);
if(nt==310)InitQ(Po,deltaR,10*deltaPhi);
if(nt==320)InitQ(Po,deltaR,11*deltaPhi);
if(nt==330)InitQ(Po,deltaR,12*deltaPhi);
if(nt==340)InitQ(Po,deltaR,13*deltaPhi);
if(nt==350)InitQ(Po,deltaR,14*deltaPhi);
if(nt==360)InitQ(Po,deltaR,15*deltaPhi);
if(nt==370)InitQ(Po,deltaR,16*deltaPhi);
if(nt==380)InitQ(Po,deltaR,17*deltaPhi);
if(nt==390)InitQ(Po,deltaR,18*deltaPhi);
if(nt==400)InitQ(Po,deltaR,19*deltaPhi);
if(nt==410)InitQ(Po,deltaR,20*deltaPhi);

\[ \text{LSS} = \frac{(\text{CenterTemp} - \text{point}[0])^2}{\text{CenterTemp} \cdot \text{CenterTemp}} + \frac{(\text{CenterTemp} - \text{point}[1])^2}{\text{CenterTemp} \cdot \text{CenterTemp}} + \frac{(\text{CenterTemp} - \text{point}[2])^2}{\text{CenterTemp} \cdot \text{CenterTemp}} + \frac{(\text{CenterTemp} - \text{point}[3])^2}{\text{CenterTemp} \cdot \text{CenterTemp}} + \frac{(\text{CenterTemp} - \text{point}[4])^2}{\text{CenterTemp} \cdot \text{CenterTemp}} \]

if(flag == 0) stop heating

InitQ(CIRCLE,deltaR,20*deltaPhi);
if(LSS_4 < LSS4)
{
    TimeRec[CountNum] = nt;
    FlagRec[CountNum] = flag;
    CountNum ++;
    flag = 2;
    goto BT;
}
{
    TimeRec[CountNum] = nt;
    FlagRec[CountNum] = flag;
    CountNum ++;
}
flag = 1;
FileWrt(nt);
}
}
if(flag == 1) //start heating
{
InitQ(PO, 0, deltaR, 0, deltaPhi);
if(LSS_4 < LSS_4)
{
TimeRec[CountNum] = nt;
FlagRec[CountNum] = flag;
CountNum++;
flag = 2;
goto BT;
}
if(point[0] < CenTemp)
{
TimeRec[CountNum] = nt;
FlagRec[CountNum] = flag;
CountNum++;
FileWrt(nt);
flag = 0;
}
BT:
if(flag == 2) //stop heating
{
InitQ(PO, 0, deltaR, 0, deltaPhi);
if(point[0] < CenTemp)
{
TimeRec[CountNum] = nt;
FlagRec[CountNum] = flag;
goto loopend;
}
}
loopend:
FileWrt(nt);
return vnew[0][0][0];

void zlihep::Clear(void)
{
int i, j, k;
for(i = 0; i < NR + l; i++)
{
for(j = 0; j < NPHi + l; j++)
{
for(k = 0; k < NZ3 + l; k++)
{
Q1[i][j][k] = 0;
Q2[i][j][k] = 0;
Q3[i][j][k] = 0;
v[i][j][k] = 0;
vnew[i][j][k] = 0;
void[i][j][k] = 0;
vold[i][j][k] = 0;
vn[i][j][k] = 0;
beta[i][j][k] = 0;
T[i][j][k] = 0;
d[i][j][k] = 0;
}
}
}

double zlihep::RunAll(double PO) //
{
double Temper = 0;
Clear();
TemRet = IntmTrsy(P0);
return TemRet;
}

int main(void)
{
    #include "N R[-1], NPhi+1, NZ3+1"); long double P0m;
    int i; ofstream fout14; fout14.open("time.txt", ios::out);
    P0m = 17.4119; zl.RunAll(P0m);
    fout14 << "LEAST SUM SQUARE " << zl.LSS << endl;
    fout14 << "4 LEAST SUM SQUARE " << zl.LSS_4 << endl;
    for(i = 0; i < 100; i++)
    {
        if(zl.TimeRec[i] != -1)
        {
            if(zl.FlagRec[i] == 0)
            {
                fout14 << "Number " "CoolTime " "z1.TimeRec[i]" << endl;
            }
            if(zl.FlagRec[i] == 1)
            {
                fout14 << "Number " "HeatTime " "z1.TimeRec[i]" << endl;
            }
            if(zl.FlagRec[i] == 2)
            {
                fout14 << "Number " "EndTime " "z1.TimeRec[i]" << endl;
            }
        }
        fout14 << "END" << endl;
        fout14.close();
        return 0;
    }
}
/*Table B.3 Program 3: Source code of step 3 in Figure 5.3 is used for the skin model with a blood vessel.
Le Zhang
4/11/05
This program is about heat transfer in the skin of a human being. There are three layers in the skin. The first layer is epidermis, the second on is dermis and the last one is Subcutaneous.
*/

#include <fstream.h>
#include <cstring.h>
#include <stdio.h>
#include <math.h>

#define NZ1 8
#define NZ2 208
#define NZ3 1208
#define NR 30
#define NPhi 20
define CIRCLE 1
#define EndTemp 4
#define CenTemp 8
#define LSS4 0.04
#define Bi 2
#define BLOODTEMP 1
#define T1 1422 //LSS4 is right
#define T2 1508 //end time

class zlihcp
{
  private:
    double ***Q1,***Q2,***Q3;
    double ***v,***vnew,***vold,***vn,***vsave;
    double ***beta,***f,***d;
    double *b,**a,*c;
    double MaxErr,h,e;
    double deltaZ,deltaT,deltaPhi,deltaR;
    double R change,P hichange,Z change,R changel,P hichangel,Z changel;
    double p l,p 2,p 3,q cl,q c2,q c3,k l,k 2,k 3,w b l,w b 2,w b 3,c b l,c b 2,c b 3;
    double Sigma,Alpha1,Alpha2,Alpha3,Reff1,Reff2,Reff3;
    double P 0,pi jude,CenterX,CenterY;
    int i,j,t,n;
    int MaxLen,MaxWid,MaxHig,VR;
    double BSpeed,BP,BF,BA lpha,BC b,B ratio2,U w l[NZ3-NZ2+1],U b l[NZ3-NZ2+1];
  public:
    double point[5];
    double LSS,MLSS;
    double LSS_4,LSS1,MLSS1;
    int TimeRec[100],FlagRec[100],nt,flag,CountNum;
    zlihcp(int l, int w, int high1)
    {
      int i,j,k; MaxLen = l; MaxWid = w; MaxHig = high1;
      CountNum = 0; LSS=0; MLSS=10000000; flag = 0;
      for (i=0;i<100;++i)
        TimeRec[i]=FlagRec[i] = -1;
      for(j=0;j<2;++j++)
        for(i=0;i<2;++j++)
          a[i] = new double [high1];
      b = new double [high1];
      c = new double [high1];
      Q1 = new double **[l];
      Q2 = new double **[l];
      Q3 = new double **[l];
      v = new double **[l];
      vnew = new double **[l];
      vold = new double **[l];
      vn = new double **[l];
      vsave = new double **[l];
      beta = new double **[l];
      f = new double **[l];
      d = new double **[l];
    }

for (j=0;j<w;j++)
{
    Q1[j] = new double*[w];
    Q2[j] = new double*[w];
    Q3[j] = new double*[w];
    v[j] = new double*[w];
    vnew[j] = new double*[w];
    vold[j] = new double*[w];
    vsave[j] = new double*[w];
    beta[j] = new double*[w];
    f[j] = new double*[w];
    d[j] = new double*[w];
    for (k=0;k<w;k++)
    {
        Q1[j][k] = new double[high1];
        Q2[j][k] = new double[high1];
        Q3[j][k] = new double[high1];
        v[j][k] = new double[high1];
        vnew[j][k] = new double[high1];
        vold[j][k] = new double[high1];
        vsave[j][k] = new double[high1];
        beta[j][k] = new double[high1];
        f[j][k] = new double[high1];
        d[j][k] = new double[high1];
    }
}
for(i=0;i<l;i++)
{
    for(j=0;j<w;j++)
    {
        for(k=0;k<high1;k++)
        {
            Q1[i][j][k]=0;
            Q2[i][j][k]=0;
            Q3[i][j][k]=0;
            v[i][j][k]=0;
            vnew[i][j][k]=0;
            vold[i][j][k]=0;
            vsave[i][j][k]=0;
            beta[i][j][k]=0;
            f[i][j][k]=0;
            d[i][j][k]=0;
        }
    }
}
Sigma= 0.1; Alpha1=1.0; Alpha2=0.8; Alpha3=0.4; Refl1=0.93; Refl2=0.93; Refl3=0.93; pi=3.14159265358979; CenterX = 0; CenterY = 0; t=2; n=10; p1=1.2; p2=1.2; p3=1.0; qcl=3.6; qc2=3.4; qc3=3.06; k1=0.0026; k2=0.0052; k3=0.0021; wb1=0.0; wb2=0.0005; wb3=0.0005; ch1=0.0; ch2=4.2; ch3=4.2; w=0.001; deltaPhi = double(2*pi/(double)NPhi);
    deltaR = double(0.5/(double)NR);
    deltaZ=0.001;
    deltaT=0.1;
    //Blood parameter
    VR = 2;
    BSpeed = 80;
    BP = 2*pi*double(VR);
    BF = pi*double(VR)*double(VR);
    BCb = 0.004134;
    BAlpha = 0.002;
    Bratio2 = BAlpha*BP/(BCh*BSpeed*BF);
}
void zlihcp::FileWrit(int timel)
{
    int i,k; ofstream fout1,fout2,fout3,fout31;
    sprintf(str,"%d",timel); strcat(str1,str); strcat(str2,str);
    strcat(str3,str); strcat(str31,str);

    ////////////zt curve/////////////////////
    fout1.open(str1,ios::out);
    fout1<< "TITLE = " "Example: Simple ZT-Volume Data" "<<endl;
    fout1<< "VARIABLES = Z, Temperature" ""<<endl;
    fout1<< "ZONE T="right",i=1209, J=31, F=POINT" ""<<endl;
    for(k=0;k<=NZ3;k++)
    fout1<< "new[0][0][k]" ""<<endl;
    fout1.close();

    ////////////contour curve/////////////////////////////
    fout2.open(str2,ios::out);
    fout2<< "TITLE = " "Example: Simple 2D-Volume Data" ""<<endl;
    fout2<< "VARIABLES = R, Z, Temperature" ""<<endl;
    fout2<< "ZONE T="right",i=1209, J=31, F=POINT" ""<<endl;
    for(i=0;i<=NR;i++)
    { for(k=0;k<=NZ3;k++)
    fout2<< "new[0][i][k]" ""<<endl;
    }
    fout2.close();

    ////////////Contour curve cross////////////////////////
    fout21.open(str21,ios::out);
    fout21<< "TITLE = " "Example: Simple 2D-Volume Data" ""<<endl;
    fout21<< "VARIABLES = R, Z, Temperature" ""<<endl;
    fout21<< "ZONE T="right",i=1209, J=31, F=POINT" ""<<endl;
    for(i=0;i<=NR;i++)
    { for(k=0;k<=NZ3;k++)
    fout21<< "new[i][i][k]" ""<<endl;
    }
    fout21.close();

};
void ziilhcp::InitQ(double P0, double Cr, double Cp) // Initialize the laser power;
{
    int i, j, z; CenterX = Cr * cos(Cp); CenterY = Cr * sin(Cp);
    for(i=0; i< NR; i++)
    {
        for(j=0; j< NPhi; j++)
        {
            for(z=0; z< NZ1; z++)
            {
                Q1[i][j][z] = Alpha1 * exp(-Alpha1 * (z * deltaZ) / (sqrt(2 * pi) * Sigma)) * exp(-pow(i * cos(j * deltaPhi) * deltaR - CenterX, 2) + pow(i * sin(j * deltaPhi) * deltaR - CenterY, 2) / (2 * Sigma * Sigma)) * P0 * (1 - Refl1);
            }
            for(z= NZ1 + 1; z< NZ2; z++)
            {
                Q2[i][j][z] = Alpha2 * exp(-Alpha2 * (z - NZ1) * deltaZ) * exp(-Alpha1 * deltaZ * NZ1) / (sqrt(2 * pi) * Sigma) * exp(-pow(i * cos(j * deltaPhi) * deltaR - CenterX, 2) + pow(i * sin(j * deltaPhi) * deltaR - CenterY, 2) / (2 * Sigma * Sigma)) * P0 * (1 - Refl2);
            }
            for(z= NZ2 + 1; z< NZ3; z++)
            {
                Q3[i][j][z] = Alpha3 * exp(-Alpha3 * (z - NZ2) * deltaZ) * exp(-Alpha1 * deltaZ * NZ1) * exp(-Alpha2 * deltaZ * (NZ2 - NZ1)) / (sqrt(2 * pi) * Sigma) * exp(-pow(i * cos(j * deltaPhi) * deltaR - CenterX, 2) + pow(i * sin(j * deltaPhi) * deltaR - CenterY, 2) / (2 * Sigma * Sigma)) * P0 * (1 - Refl3);
            }
        }
    }
    fout31.close();
}

void ziilhcp::Vessel(void)
{
    /* It is used to calculate the blood vessel.

    Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.*/
}
Runge-Kutta (order four) is applied for the function.
All of the variable has a prefix RF
The equation is like \( Cb \times w \times P \times dB \, dDz = -Alpha \times P \times(Tb - Tw) \)
The entry temperature is 1 centigrade.

```c
int RFi, RFj, RFk;
double RFw[NZ3 - NZ2 + 1];
```

```c
double RFh, RFz, RFk1, RFk2, RFk3, RFk4, RFTw;
for (RFi = 0; RFi <= NZ3 - NZ2; RFi++)
Ub1[RFi] = RFw[RFi] = 0;
RFi = 0;
RFh = deltaZ;
RFz = 0;
RFw[RFi] = BLOODTEMP;
for (RFi = 1; RFi <= NZ3 - NZ2 + 1; RFi++)
{
    /* Get the average wall temperature from the tissue part
    Here we only choose the four angle points average temperature */
    RFTw = vold[VR][NPphi4][NZ3 + 1 - RFi] + vold[VR][NPphi2][NZ3 + 1 - RFi] + 
    vold[VR][3 * NPphi4][NZ3 + 1 - RFi] + vold[VR][0][NZ3 + 1 - RFi];
    RFTw = RFTw / 4;
    Uwl[RFi] = RFTw;
    // Solve the Runge-Kutta equation
    RFk1 = RFh * (-Bratio2 * (RFw[RFi - 1] - RFTw));
    RFk2 = RFh * (-Bratio2 * (RFw[RFi - 1] + RFk1/2 - RFTw));
    RFk3 = RFh * (-Bratio2 * (RFw[RFi - 1] + RFk2/2 - RFTw));
    RFk4 = RFh * (-Bratio2 * (RFw[RFi - 1] + RFk3 - RFTw));
    RFw[RFi] = RFw[RFi - 1] + (RFk1 + 2 * RFk2 + 2 * RFk3 + RFk4)/6;
    RFz = RFi * RFh;
}
```

```c
// Receive the data from the function
for (RFi = 0; RFi <= NZ3 - NZ2; RFi++)
Ub1[RFi] = RFw[RFi];
for (RFi = 0; RFi <= VR - 1; RFi++)
{
    for (RFj = 0; RFj <= NPphi; RFj++)
    {
        for (RFk = NZ2; RFk <= NZ3; RFk++)
        {
            vold[RFi][RFj][RFk] = Ub1[NZ3 - RFk];
        }
    }
}
```

```c
double zlhcpc::intTsy(double P0) // Time Iteration and Tri-diagonal systme
{
    MaxErr = 1.0; t = 0;
    while (nt < nbar - 1)
    {
        nth++;
        if (nth == 1)
            InitQ(P0, 0, 0);
        t = 0;
        for (i = 0; i <= NR; i++)
        {
            for (j = 0; j <= NPphi; j++)
            {
                for (z = 0; z <= NZ3; z++)
                {
                    vn[i][j][z] = vold[i][j][z];
                    vsave[i][j][z] = vold[i][j][z];
                }
            }
            cout << " new cycle" << endl;
            MaxErr = 1.0;
            while (MaxErr <= e)
            {
                Vessel();
            }
        }
    }
    return MaxErr;
}
```
MaxErr=0.0;
for(i=1;i<=NR-1;i++)
{
    for(j=1;j<=NPHI;j++)
    {
        ///////////////The first layer////////////////////////
        for(z=1;z<=NZL-1;z++)
        {
            /////The n+1 state////////////////////
            Rchange = kl*deltaT*((i+0.5)*vold(i+1,j)[z]-2*i*vold(i,j)[z]+(i-0.5)*vold(i-1,j)[z])/i*deltaR*deltaR);
            if(j==NPHI)
            {
                Phichange = kl*deltaT*(vold(i)[j+1][z]-2*vold(i)[j][z]+vold(i)[j-1][z])/i*deltaR*deltaPhi,2);
            } else
            {
                Phichange = kl*deltaT*(vold(i)[j][z]-2*vold(i)[j][z]+vold(i)[j-1][z])/i*deltaR*deltaPhi,2);
            }
            Zchange = kl*deltaT*(vold(i)[j][z+1]-2*vold(i)[j][z]+vold(i)[j][z-1])/pow(i*deltaR*deltaPhi,2);
        } /(*deltaR*deltaR),
        if(j==NPHI)
        {
            Phichange1 = kl*deltaT*(vn(i)[j][z]-2*vn(i)[j][z]+vn(i)[j-1][z])/i*deltaR*deltaPhi,2);
        } else
        {
            Phichange1 = kl*deltaT*(vn(i)[j+1][z]-2*vn(i)[j][z]+vn(i)[j-1][z])/pow(i*deltaR*deltaPhi,2);
        }
        Zchange1 = kl*deltaT*(vn(i)[j][z+1]-2*vn(i)[j][z]+vn(i)[j][z-1])/pow(i*deltaR*deltaPhi,2);
        /////Prepare the coefficient for Thomas way////////////////////////
        f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*(Q[i][j][z]+2*Q1[i][j][z]+2*Q2[i][j][z])
        a[i][j][z]=k1*deltaT/((deltaZ*deltaZ);
        b[i][j][z]=k2*deltaT/((deltaZ*deltaZ);
        d[i][j][z]=f[i][j][z]+Rchange1+Phichange1+Zchange1+2*deltaT*(Q[i][j][z]+2*Q1[i][j][z]+2*Q2[i][j][z])
        k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+2*k1*deltaT/pow(deltaZ,2);
        c[i][j][z]=k2*deltaT/((deltaZ*deltaZ);
        d[i][j][z]=f[i][j][z]+Rchange1+Phichange1+Zchange1+2*deltaT*(Q[i][j][z]+2*Q1[i][j][z]+2*Q2[i][j][z])
        k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+4/pow(deltaPhi*i*deltaR,2))vold(i)[j][z];
    } a[i][j]=a[i][j]-b[i][j];
    b[i][j]=0;
    b[NZL]=k1;
    a[i][j][NZL]=k1+k2;
    c[NZL]=k2;
    d[i][j][NZL]=0;
    /////The second layer////////////////////////
    for(z=NZL+1;z<=NZL+2+z++)
    {
        ///////////////The n+1 state////////////////////////
        Rchange = k2*deltaT*((i+0.5)*vold(i+1,j)[j][z]-2*i*vold(i,j)[j][z]+(i-0.5)*vold(i-1,j)[j][z])
        /i*deltaR*deltaR);
        if(j==NPHI)
        {
            Phichange = k2*deltaT*(vold(i)[j+1][j][z]-2*vold(i)[j][j][z]+vold(i)[j-1][j][z])/pow(i*deltaR*deltaPhi,2);
        } else
        {
            Phichange = k2*deltaT*(vold(i)[j][j][z]-2*vold(i)[j][j][z]+vold(i)[j-1][j][z])/pow(i*deltaR*deltaPhi,2);
        }
        Zchange = k2*deltaT*(vold(i)[j][j][z+1]-2*vold(i)[j][j][z]+vold(i)[j][j][z-1])/pow(deltaZ,2);
        /////The n state////////////////////////
        Rchange1 = k2*deltaT*(i+0.5)*vn[i][j][j][z]+(i-0.5)*vn[i-1][j][j][z])
        /i*deltaR*deltaR);
        if(j==NPHI)
        {
            Phichange1 = k2*deltaT*(vn[i][j+1][j][z]-2*vn[i][j][j][z]+vn[i][j-1][j][z])/i*deltaR*deltaPhi,2);
        } else
        {
            Phichange1 = k2*deltaT*(vn[i][j][j+1][j][z]-2*vn[i][j][j][z]+vn[i][j][j-1][j][z])/i*deltaR*deltaPhi,2);
        }
    }
Zchange1 = k2*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][z-1])/pow(deltaZ,2);

// Prepare the coefficient for Thomas way

f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q2[i][j][z]+(2*p2*qc2-wb2*cb2*deltaT)*vn[i][j][z];
b[i][j][z]=k2*deltaT/(deltaZ*deltaZ);
d[i][j][z]=2*p2*qc2+w2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));
c[i][j][z]=k2*deltaT/pow(deltaZ,2);
d[i][j][z]=f[i][j][z]-Rchange1-Phichange1-k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));

if (i>=VR+1)
{
  b[NZ2]=k2;
  a[i][NZ2]=k2+k3;
  c[NZ2]=k3;
d[i][j][NZ2]=0;
}
if(i<=VR)
{
  b[NZ2]=-k2;
  a[i][j][NZ2]=2*p2*qc2+w2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));
  c[NZ2]=k2*deltaT/pow(deltaZ,2);
  d[i][j][NZ2]=f[i][j][z];
}

if((i==VR+1)
{
  b[NZ2]=k2;
  a[i][j][NZ2]=k2+k3;
  c[NZ2]=k3;
d[i][j][NZ2]=0;
}

if((z>=NZ2)&& (i<=VR ))
continue;

else
{
  Rchange = k3*deltaT*((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
  if(j==NPphi)
  Phichange = k3*deltaT*(vn[i][l][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
  else
  Phichange = k3*deltaT*(vn[i][j+l][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
  Zchange = k3*deltaT*(vn[i][j][z+l]-2*vn[i][j][z]+vn[i][j-l][z])/pow(deltaZ,2);
  // Prepare the coefficient for Thomas way
  f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q2[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z];
  b[i][j][z]=k3*deltaT/(deltaZ*deltaZ);
  a[i][j][z]=2*p3*qc3+w3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));
  c[i][j][z]=k3*deltaT/pow(deltaZ,2);
  d[i][j][z]=f[i][j][z];
}
}

a[i][j][NZ3-1]=a[i][j][NPphi-1];
c[NZ3-1]=0;
}

// tri-diagonal system
for(i=1;i<=NR-1;i++)
{
  for(j=1;j<=NPphi;j++)
  {
    v[i][j][NZ3]=0;
    beta[i][j][NZ3]=0;
    if((z>=NZ3)&&(i<=VR ))
    continue;
    else
    {
      // The n+1 state
      Rchange = k3*deltaT*((i+0.5)*vn[i][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
      if(j==NPphi)
      Phichange = k3*deltaT*(vn[i][l][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
      else
      Phichange = k3*deltaT*(vn[i][j+l][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
      Zchange = k3*deltaT*(vn[i][j][z+l]-2*vn[i][j][z]+vn[i][j-l][z])/pow(deltaZ,2);
      // Prepare the coefficient for Thomas way
      f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q2[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z];
      b[i][j][z]=k3*deltaT/(deltaZ*deltaZ);
      a[i][j][z]=2*p3*qc3+w3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2));
      c[i][j][z]=k3*deltaT/pow(deltaZ,2);
      d[i][j][z]=f[i][j][z];
    }
  }
}
\[
v[i][j][z] = (d[i][j][z]+c[i][j][z]*v[i][j][z+1])/(a[i][j][z]+c[i][j][z]*\text{beta}[i][j][z+1]),
\]
\[
\text{beta}[i][j][z] = b[i][j][z]/(a[i][j][z]+c[i][j][z]*\text{beta}[i][j][z+1]);
\]

for (i = 1; i <= N_R-1; i++)
{
    for (j = 1; j <= N_Phi; j++)
    {
        for (z = 1; z <= N_Z3-1; z++)
        {
            if ((z >= N_Z2) && (i < VR))
                continue;
        }
    }
}

for (i = 0; i <= N_R-1; i++)
{
    for (j = 0; j <= N_Phi; j++)
    {
        for (z = N_Z2; z <= N_Z3; z++)
        {
            vnew[i][j][z] = Ub1[N_Z3-z];
        }
    }
}

for (i = 0; i <= N_R; i++)
{
    for (j = 0; j <= N_Phi; j++)
    {
        for (z = 0; z <= N_Z3; z++)
        {
            vnew[i][j][0] = vnew[i][j][1];
            if (i == VR+1)
                vnew[i][j][N_Z3] = vnew[i][j][N_Z3-1];
            vnew[i][0][z] = vnew[i][N_Phi][z];
            if (z == N_Z2)
                vnew[VR][j][z] = (vnew[VR+1][j][z]*Bi*deltaR*vnew[0][j][z])/(1+deltaR*Bi);
            else
                vnew[0][i][z] = vnew[1][i][z];
            if (i == VR-1)
                vnew[1][i][N_Z2] = Ub1[N_Z3-N_Z2];
            vnew[N_R][j][z] = vnew[N_R-1][j][z];
            vold[i][j][z] = vnew[i][j][z];
        }
    }
}

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\n
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
if(nt==40)InitQ(P0,deltaR,4*deltaPhi);
if(nt==50)InitQ(P0,deltaR,5*deltaPhi);
if(nt==60)InitQ(P0,deltaR,6*deltaPhi);
if(nt==70)InitQ(P0,deltaR,7*deltaPhi);
if(nt==80)InitQ(P0,deltaR,8*deltaPhi);
if(nt==90)InitQ(P0,deltaR,9*deltaPhi);
if(nt==100)InitQ(P0,deltaR,10*deltaPhi);
if(nt==110)InitQ(P0,deltaR,11*deltaPhi);
if(nt==120)InitQ(P0,deltaR,12*deltaPhi);
if(nt==130)InitQ(P0,deltaR,13*deltaPhi);
if(nt==140)InitQ(P0,deltaR,14*deltaPhi);
if(nt==150)InitQ(P0,deltaR,15*deltaPhi);
if(nt==160)InitQ(P0,deltaR,16*deltaPhi);
if(nt==170)InitQ(P0,deltaR,17*deltaPhi);
if(nt==180)InitQ(P0,deltaR,18*deltaPhi);
if(nt==190)InitQ(P0,deltaR,19*deltaPhi);
if(nt==200)InitQ(P0,deltaR,20*deltaPhi);
if(nt==210)InitQ(P0,0,0);
if(nt==220)InitQ(P0,deltaR,1*deltaPhi);
if(nt==230)InitQ(P0,deltaR,2*deltaPhi);
if(nt==240)InitQ(P0,deltaR,3*deltaPhi);
if(nt==250)InitQ(P0,deltaR,4*deltaPhi);
if(nt==260)InitQ(P0,deltaR,5*deltaPhi);
if(nt==270)InitQ(P0,deltaR,6*deltaPhi);
if(nt==280)InitQ(P0,deltaR,7*deltaPhi);
if(nt==290)InitQ(P0,deltaR,8*deltaPhi);
if(nt==300)InitQ(P0,deltaR,9*deltaPhi);
if(nt==310)InitQ(P0,deltaR,10*deltaPhi);
if(nt==320)InitQ(P0,deltaR,11*deltaPhi);
if(nt==330)InitQ(P0,deltaR,12*deltaPhi);
if(nt==340)InitQ(P0,deltaR,13*deltaPhi);
if(nt==350)InitQ(P0,deltaR,14*deltaPhi);
if(nt==360)InitQ(P0,deltaR,15*deltaPhi);
if(nt==370)InitQ(P0,deltaR,16*deltaPhi);
if(nt==380)InitQ(P0,deltaR,17*deltaPhi);
if(nt==390)InitQ(P0,deltaR,18*deltaPhi);
if(nt==400)InitQ(P0,deltaR,19*deltaPhi);
if(nt==410)InitQ(P0,deltaR,20*deltaPhi);

LSS = pow((CenTemp-point[0]),2)/(CenTemp*CenTemp)+pow((EndTemp-point[1]),2)/(EndTemp*EndTemp)
+pow((EndTemp-point[2]),2)/(EndTemp*EndTemp)+pow((EndTemp-point[3]),2)/(EndTemp*EndTemp)
+pow((EndTemp-point[4]),2)/(EndTemp*EndTemp);
LSS_4 = pow((EndTemp-point[1]),2)/(EndTemp*EndTemp)+pow((EndTemp-point[2]),2)/(EndTemp*EndTemp)
+pow((EndTemp-point[3]),2)/(EndTemp*EndTemp)+pow((EndTemp-point[4]),2)/(EndTemp*EndTemp);

if(flag = = 0) //stop heating
    {InitQ(0,CIRCLE*deltaR,0*deltaPhi);
     if(LSS_4<LSS4)
        {
            TimeRec[CountNum] = nt;
            FlagRec[CountNum] = flag;
            CountNum ++; goto loopend;
        }
    }

if(flag = = 1) //start heating
    {InitQ(P0,0*deltaR,0*deltaPhi);
     if(LSS_4<LSS4) 
        {TimeRec[CountNum] = nt;
         FlagRec[CountNum] = flag;
         CountNum ++; flag +=1; FileWrit(nt);
        }
    }
if(point[0]>CenTemp)
{
    TimeRec[CountNum] = nt;
    FlagRec[CountNum] = flag;
    CountNum ++;
    FileWrit(nt);
    flag = 0;
}
}
if(flag == 2) //stop heating
    goto looend;
}
}

loopend:
FileWrit(nt);
return vnew[0][0][0];

void zlihcp::Clear(void)
{
    int i,j,k;
    for(i=0;i<NR+1;i++)
    {
        for(j=0;j<NPHI+1;j++)
        {
            for(k=0;k<NZ3+1;k++)
            {
                Q1[i][j][k] = 0;
                Q2[i][j][k] = 0;
                Q3[i][j][k] = 0;
                v[i][j][k] = 0;
                vnew[i][j][k] = 0;
                vold[i][j][k] = 0;
                vn[i][j][k] = 0;
                beta[i][j][k] = 0;
                f[i][j][k] = 0;
                d[i][j][k] = 0;
            }
        }
    }
}

void zlihcp::Clearl(void)
{
    int i,j,k;
    for(i=0;i<NR+1;i++)
    {
        for(j=0;j<NPHI+1;j++)
        {
            for(k=0;k<NZ3+1;k++)
            {
                Q1[i][j][k] = 0;
                Q2[i][j][k] = 0;
                Q3[i][j][k] = 0;
                v[i][j][k] = 0;
                vnew[i][j][k] = 0;
                vold[i][j][k] = vsave[i][j][k];
                vn[i][j][k] = 0;
                beta[i][j][k] = 0;
                f[i][j][k] = 0;
                d[i][j][k] = 0;
                nt = T1; //4LSS is the minimum;
            }
        }
    }
}

double zlihcp::RunAll(double P0) //
{
    double TemRet = 0; Clear();
    TemRet = IntmTrsy(P0); return TemRet;
}
double zlhcp::IntmTrsyl(double P0) // Time Iteration and Tri-diagonal system
{
    MaxErr=1.0;
    while(nt<=T2) // need the center temp back to 8
    {
        nt++;
        InitQ(PO,0,0);
        t=0;
        for(i=0;i<NR;i++)
        {
            for(j=0;j<NP;i++)
            {
                for(z=0;z<NZ;z++)
                {
                    vn[i][j][z]=vold[i][j][z];
                }
            }
        }
        MaxErr=1.0;
        while(MaxErr>=e)
        {
            Vessel();
            MaxErr=0.0;
            for(i=1;i<=NR-1;i++)
            {
                for(j=1;j<=NP-1;j++)
                {
                    // The first layer
                    for(z=1;z<=NZ-1;z++)
                    {
                        Rchange = k1*deltaT*((i+0.5)*vold[i+1][j][z]-2*i*vold[i][j][z]+(i-0.5)*vold[i-1][j][z])/i*deltaR*deltaR;
                        if(j==NP)
                        {
                            Phichange = k1*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
                        } else
                        {
                            Phichange = k1*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
                        }
                        Zchange = k1*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j][1])/pow(deltaZ,2);
                        // The n+1 state
                        Rchange1 = k1*deltaT*(((i+0.5)*vn[i+1][j][z]-2*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
                        if(j==NP)
                        {
                            Phichange1 = k1*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
                        } else
                        {
                            Phichange1 = k1*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-l][z])/pow(i*deltaR*deltaPhi,2);
                        }
                        Zchange1 = k1*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j][1])/pow(deltaZ,2);
                        // Prepare the coefficient for Thomas way
                        f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*(Q1[i][j][z]+2*p1*q1-wb1*cbl*deltaT)*vn[i][j][z];
                        b[z]=(k1*deltaT)*(deltaZ*deltaZ);
                        a[i][j][z]=2*p1*q1+wb1*cbl*deltaT+k1*deltaT*(4*i+1)/(2*k1*deltaT)*pow(deltaPhi*deltaPhi,2);
                        e[z]=(k1*deltaT)*pow(deltaPhi*deltaPhi,2);
                        d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                        k1*deltaT*(4*i+1)/(deltaR*deltaR)+4*deltaT*(pow(deltaPhi*deltaPhi,2)*vold[i][j][z];
                    }
                }
            }
        }
        a[i][1]=a[i][1]-b[1];
        b[1]=0;
        b[NZ1]=1;
        a[i][NZ1]=k1+k2;
        c[NZ1]=k2;
        d[i][j][NZ1]=0;
        // The second layer
        for(i=NZ1;i<=NZ2-1;i++)
        {
            Rchange = k2*deltaT*((i+0.5)*vold[i+1][j][z]-2*i*vdold[i][j][z]+(i-0.5)*vold[i-1][j][z])/i*deltaR*deltaR;
        }
    }
}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
if(j==N Phi)
    Phichange = k2*deltaT*(vold[i][j][z]-2*vold[i][j-1][z]+vold[i][j-1][z-1]/pow(i*deltaR*deltaPhi,2);
else
    Phichange = k2*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
Zchange = k2*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j-1][z-1]/pow(deltaZ,2);

Rchange1 = k2*deltaT*((i+0.5)*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])
    /((i*deltaR*deltaR));

if(j==N Phi)
    Phichange1 = k2*deltaT*(vn[i][j][z]-2*vn[i][j-1][z]+vn[i][j-1][z-1]/pow(i*deltaR*deltaPhi,2);
else
    Phichange1 = k2*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
Zchange1 = k2*deltaT*(vn[i][j][z+1]-2*vn[i][j][z]+vn[i][j-1][z-1]/pow(deltaZ,2);

// The third layer
for(z=2; z<=NZ3-1; z++)
{
    // The n+1 state
    Rchange = k3*deltaT*((i+0.5)*vold[i+1][j][z]-2*vold[i][j][z]+vold[i-1][j][z]/(i*deltaR*deltaR);
    if(j==N Phi)
        Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
    else
        Phichange = k3*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j-1][z])/pow(deltaZ,2);
    Zchange = k3*deltaT*(vold[i][j][z+1]-2*vold[i][j][z]+vold[i][j-1][z]/pow(deltaZ,2);
}

// tri-diagonal system

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
for(i=1;i<=NR-1;i++)
{
  for(j=1;j<=NPhi;j++)
  {
    v[i][j][NZ2]=0.0;
    beta[i][j][NZ2]=0.0;
    for(z=NZ3-1;z>=1;z--)
      {
        if((z>=NZ2)&(i<=VR))
          continue;
        else
          {
            v[i][j][z]=(d[i][j][z]+c[i][z]*v[i][j][z+1])/(a[i][z]-c[i][z]*beta[i][j][z+1]);
            beta[i][j][z]=b[z]/(a[i][z]-c[i][z]*beta[i][j][z+1]);
          }
      }
  }
}
for(i=1;i<=NR-1;i++)
{
  for(j=1;j<=NPhi;j++)
  {
    for(z=1;z<=NZ3-1;z++)
      {
        if((z>=NZ2)&(i<=VR))
          continue;
        else
          {
            vnew[i][j][z]=v[i][j][z]+beta[i][j][z]*vnew[i][j][z-1];
            judge=(vnew[i][j][z]-vold[i][j][z]);
            if(judge<0)
              judge = judge*(-1);
            if(judge>MaxErr)
              MaxErr=judge;
            vold[i][j][z]=vnew[i][j][z];
          }
      }
  }
}
for(i=0;i<=NR;i++)
{
  for(j=0;j<=NPhi;j++)
  {
    for(z=0;z<=NZ3;z++)
      {
        vnew[i][j][0]=vnew[i][j][1];
        vnew[i][j][NZ3]=vnew[i][j][NZ3-1];
        if(i>(VR+1))
          vnew[i][j][NZ2]=vnew[i][j][NPhi];
        else
          vnew[i][j][0][z]=vnew[i][j][NPhi][z];
        vnew[0][j][0]=vnew[i][j][0];
        vnew[0][j][z]=vnew[i][j][1][z];
        vnew[NR-1][j][z]=vnew[i][j][NZ2][z];
        vnew[NR][j][0]=vnew[i][j][NZ3][z];
        vold[i][j][0]=vnew[i][j][0];
        vold[i][j][z]=vnew[i][j][z];
    }
  }
}
//Boundary Condition
for(i=0;i<=VR-1;i++)
{
  for(j=0;j<=NPhi,j++)
  {
    for(z=NZ2;z<=NZ3;z++)
      {
        vnew[i][j][z]=Ub1[NZ3-z];
      }
  }
}
for(i=0;i<=NR;i++)
{
  for(j=0;j<=NPhi;j++)
  {
    for(z=0;z<=NZ3;z++)
      {
        vnew[0][j][0]=vnew[i][j][1];
        if(i>(VR+1))
          vnew[0][j][NZ3]=vnew[i][j][NZ3-1];
        if(i<VR)
          vnew[0][j][NZ2]=Ub1[NZ3-NZ2];
        else
          vnew[0][j][0][z]=vnew[i][j][NPhi][z];
        vnew[0][j][0][z]=vnew[i][j][0][z]+Bi*deltaR*vnew[0][j][0][z]/(1+deltaR*Bi);
      }
  }
}
FileWrite(nt);
return vnew[0][0][0];
}
double zlihcp::RunAll1(double P0) //
{
  double TempRet = 0; Clear1();
}
TemRet = InTrysy1(P0); return TemRet;
}

int main(void)
{
    zlihcp zl(NR+1,Nphi+1,NZ+1);
    long double P0m, T1m, T2m, deltaP;
    long double S, Snew, Pnew, error1;
    double rec[5], rec[1][5], X[5], Scale, Scale2;
    double Tpoint=CenTemp, Tpointa=EndTemp;
    int i; ofstream fout14; fout14.open("time.txt", ios::out);
    Pnew=17.4119; T1m= 0, T2m=0;
    S=0, Snew=0, error1=0.001; P0m=17.4119;
    zl.RunAll(P0m);
    fout14<<"LEAST SUM SQUARE "<<zl.LSS_4<<endl;
    for(i=0;i<100;i++)
    {
        if(zl.TimeRec[i]!=-1)
            fout14<<"Number "<<i<<" CoolTime "<<zl.TimeRec[i]<<endl;
        if(zl.FlagRec[i] == 1)
            fout14<<"Number "<<i<<" HeatTime "<<zl.TimeRec[i]<<endl;
        if(zl.FlagRec[i] == 2)
            fout14<<"Number "<<i<<" EndTime "<<zl.TimeRec[i]<<endl;
    }
    do
    {
        P0m=Pnew; deltaP=P0m/100;
        S=Snew; T1m=zl.RunAll(P0m);
        for(i=0;i<5;i++)
            rec[i] = zl.point[i];
        T2m=zl.RunAll(P0m+deltaP);
        for(i=0;i<5;i++)
            rec[1][i] = zl.point[i];
        /*Compute the Coefficient*/
        X[0] = (rec[1][0]-rec[0])/deltaP;
        X[1] = (rec[1][1]-rec[1][0])/deltaP;
        X[2] = (rec[1][2]-rec[2])/deltaP;
        X[3] = (rec[1][3]-rec[3])/deltaP;
        X[4] = (rec[1][4]-rec[4])/deltaP;
        Scale1=pow(X[0],2)+pow(X[1],2)+pow(X[2],2)+pow(X[3],2)+pow(X[4],2);
        Scale2=X[0]*(Tpoint-rec[0])+X[1]*(Tpointa-rec[1])+X[2]*(Tpointa-rec[2])
            +X[3]*(Tpointa-rec[3])+X[4]*(Tpointa-rec[4]);
        Pnew = P0m+Scale2/Scale1;
        Snew = pow((Tpoint-rec[0]),2)+(CenTemp*CenTemp)+pow((Tpointa-rec[1]),2)+(EndTemp*EndTemp)
            +pow((Tpointa-rec[2]),2)+(EndTemp*EndTemp)+pow((Tpointa-rec[3]),2)+(EndTemp*EndTemp)
            +pow((Tpointa-rec[4]),2)+(EndTemp*EndTemp);
        fout14<<"LEAST SQUARE SUM "<<Snew<<endl;
    }
    while ((Snew-S)/Snew > error1);
    fout14<<"END"<<endl;
    fout14.close();
    return 0;
}
REFERENCES


170


