Spring 2005

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

Le Zhang

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

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

03/30/2005
Date

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

Weizhong Dai
Supervisor of Dissertation Research

[illegible]
Head of Department

CAM
Department

Recommendation concurred in:

[illegible]

[illegible]

[illegible]

[illegible]

Advisory Committee

Approved:
[illegible]
Director of Graduate Studies

[illegible]
Dean of the College

Approved:
[illegible]
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 [illegible]

Date 4/18/05

GS Form 14
(5/03)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

B_i	Biot number
C_l	specific heat of layer 1
C_b^l	specific heat of blood in layer 1
I	iterative index
$\mathbf{I}$	identity matrix
i, j, k	index of grid point
k_l	heat conductivity of layer l
L_{pre}^l	pre-conditioned Richardson operator
L_l	depth of layer l
N_r, N_φ, N_l^z	numbers of grid points in the r, φ, z directions, respectively
n	time level
P_0	laser intensity
Q_r^l	heat source in layer l
R	number of grid point for the radius of the skin structure
R_{bv}	number of grid point for the radius of the blood vessel
Reff_l	laser reflectivity in layer l
r, φ, z	cylindrical coordinates
S	least squares sum
t	Time
$u_{ijk}^n, (u_b)_k$	numerical solutions of tissue and blood, respectively
W_b^l	blood perfusion rate of layer l
α_l	laser absorptivity of layer l
$P_r^2, \delta_z^2, \delta_\varphi^2$	second order finite difference operators
ρ_l	density of layer l
ω	relaxation parameter
σ	standard deviation of laser beam width
$\theta_b, \theta_l, \theta_w$	elevated blood, tissue and vessel periphery temperatures, respectively
$\Delta r, \Delta\varphi, \Delta z$	mesh sizes in the r, φ, z directions, respectively
Δt	time increment

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 $42-46^{\circ}C$) 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 Disseratation

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

BACKGROUD 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 $T_j(\hat{p}_i)$, $j=1,2,...,M$, computed from the solution of the direct problem by using the estimated values of the heat source $\hat{p}_i$, $i=1,2,...,M$, should match the measured temperatures Y_j, $j=1,2,...,M$, as closely as possible over a specified time domain $0<t<t_f$. Here, the superscript ^ 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

$$S(\hat{\mathrm{p}})=\sum_{i=1}^{N}[Y_i-\hat{T}_i(\hat{\mathrm{p}})]^2+\alpha^*\sum_{j=1}^{M}\hat{p}_j^2\,, \tag{2.1}$$

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.

$T_i(\hat{\mathrm{p}})$ = estimated temperature obtained from the solution of the direct problem by using the estimated values of the unknown parameters $\hat{\mathrm{p}}=\{\hat{p}_1,\hat{p}_2,...,\hat{p}_M\}$.

$\hat{p}_j$ = element of the estimated parameter vector $\hat{\mathrm{p}} = \{\hat{p}_1, \hat{p}_2, ..., \hat{p}_M\}$.

α^* = 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 α^* 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 α^*, the solution is damped and deviates from the exact results. By proper selection of α^*, instability can be alleviated [Beck 1985]. Thus, selection of α^* is crucial while the number of parameters is large. Because only optimized laser power interests us in this dissertation, α^* 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.

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

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

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

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].

Equation (2.2) can be rearranged in the form

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

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

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

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

$$X^T(Y-T)=\alpha^* p \tag{2.4a}$$

where

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

$$X=\frac{\partial \mathrm{T}}{\partial \mathrm{p}^t}=\begin{bmatrix}\frac{\partial \hat{T}_1}{\partial \hat{p}_1} & \frac{\partial \hat{T}_1}{\partial \hat{p}_2} & \cdots & \frac{\partial \hat{T}_1}{\partial \hat{p}_M}\\ \frac{\partial \hat{T}_2}{\partial \hat{p}_1} & \frac{\partial \hat{T}_2}{\partial \hat{p}_2} & \cdots & \frac{\partial \hat{T}_2}{\partial \hat{p}_M}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{\partial \hat{T}_N}{\partial \hat{p}_1} & \frac{\partial \hat{T}_N}{\partial \hat{p}_2} & \cdots & \frac{\partial \hat{T}_N}{\partial \hat{p}_M}\end{bmatrix}, \tag{2.4c}$$

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

$$X_{ji}\equiv\frac{\partial \hat{T}_i}{\partial \hat{p}_j},\qquad i=1,2,...,N \text{ and } j=1,2,...,\mathrm{M}. \tag{2.5}$$

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 $\hat{T}_i(\mathrm{p})$ in a Taylor series with respect to an arbitrary value of a parameter as

$$\hat{T}_i = \hat{T}_{0i} + \sum_{h=1}^{N} \frac{\partial \hat{T}_i}{\partial \hat{p}_h} (\hat{p}_h - \hat{p}_0). \tag{2.6a}$$

This result is expressed in matrix form as

$$T = T_0 + \frac{\partial T}{\partial p^t} (p - p_0). \tag{2.6b}$$

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

$$\hat{T}_j = \sum_{h=1}^{N} \frac{\partial \hat{T}_i}{\partial \hat{p}_h} \hat{p}_h \tag{2.7a}$$

and

$$T = \frac{\partial T}{\partial p^t} p \equiv X_p. \tag{2.7b}$$

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

$$\sum_{i=1}^{N} \frac{\partial \hat{T}_i}{\partial \hat{p}_j} \left(Y_i - \sum_{h=1}^{N} \frac{\partial \hat{T}_i}{\partial \hat{p}_h} \hat{p}_h \right) = \alpha^* \sum_{k=1}^{M} \hat{p}_k \frac{\partial \hat{p}_k}{\partial \hat{p}_j}. \tag{2.8a}$$

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

$$X^t (Y - Xp) = \alpha^* p. \tag{2.8b}$$

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

$$p = (X^t X + \alpha^* I)^{-1} X^t Y. \tag{2.9}$$

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

$$p^{k+1} = p^k + (X^t X + \alpha^* I)^{-1} X^t (Y - T). \tag{2.10}$$

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 $\alpha^* \to 0$, Eq. (2.10) reduces to the Newton's method and for $\alpha^* \to \infty$, 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:

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

let T_{ijk} denote the approximation to $T(i\Delta x, j\Delta y, k\Delta z)$, where $\Delta x, \Delta y$ and Δz 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:

$$\frac{1}{\Delta x^2}\delta_x^2 T_{ijk}=\frac{1}{\Delta x^2}(T_{i+1jk}-2T_{ijk}+T_{i-1jk}) \tag{2.12}$$

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

$$-(\frac{1}{\Delta x^2}\delta_x^2+\frac{1}{\Delta y^2}\delta_y^2+\frac{1}{\Delta z^2}\delta_z^2)T_{ijk}=f_{ijk}. \tag{2.13}$$

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

$$(A_x + A_y + A_z)\overline{T} = \overline{f}. \tag{2.14}$$

It can seen [Li 1979] that the eigenvalues of A_z are $\frac{4}{\Delta z^2}\sin^2\frac{k\pi\Delta z}{2}$, $k = 1,\ldots,N_z - 1$. Since Δz is very small compared with Δx and Δy, the ratio $\frac{\lambda_{\max}(A_z)}{\lambda_{\min}(A_z)} = O(\frac{1}{\Delta z^2})$ is very large, where $\lambda_{\max}(A_z)$ and $\lambda_{\min}(A_z)$ 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

$$L_{pre}\overline{T}^{(n+1)} = L_{pre}\overline{T}^{(n)} - \alpha[(A_x + A_y + A_z)\overline{T}^{(n)} - \overline{f}], \tag{2.15}$$

where the precondtioner is chosen as follows:

$$L_{pre} \equiv A_z + (\frac{4}{\Delta x^2} + \frac{4}{\Delta y^2})I, \tag{2.16}$$

and α is a relaxation parameter. It is well known from the numerical linear algebra that the iteration process converges if the iteration operator

$$B = I - \alpha L_{pre}^{-1}(A_x + A_y + A_z) \tag{2.17}$$

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 $L_{pre}^{-1}(A_x + A_y + A_z)$ has the form

$$\lambda_{ijk} = \frac{\frac{4}{\Delta x^2}\sin^2\frac{i\pi\Delta x}{2} + \frac{4}{\Delta y^2}\sin^2\frac{j\pi\Delta y}{2} + \frac{4}{\Delta z^2}\sin^2\frac{k\pi\Delta z}{2}}{\frac{4}{\Delta x^2} + \frac{4}{\Delta y^2} + \frac{4}{\Delta z^2}\sin^2\frac{k\pi\Delta z}{2}}. \tag{2.18}$$

When Δz is very small compared with Δx and Δy, λ_{ijk} is dominated by $\frac{4}{\Delta z^2}\sin^2\frac{k\pi\Delta z}{2}$. Thus, λ_{ijk} is close to 1. If one chooses a relaxation parameter α 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 $42-46^\circ C$) 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 $42^\circ C$, 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

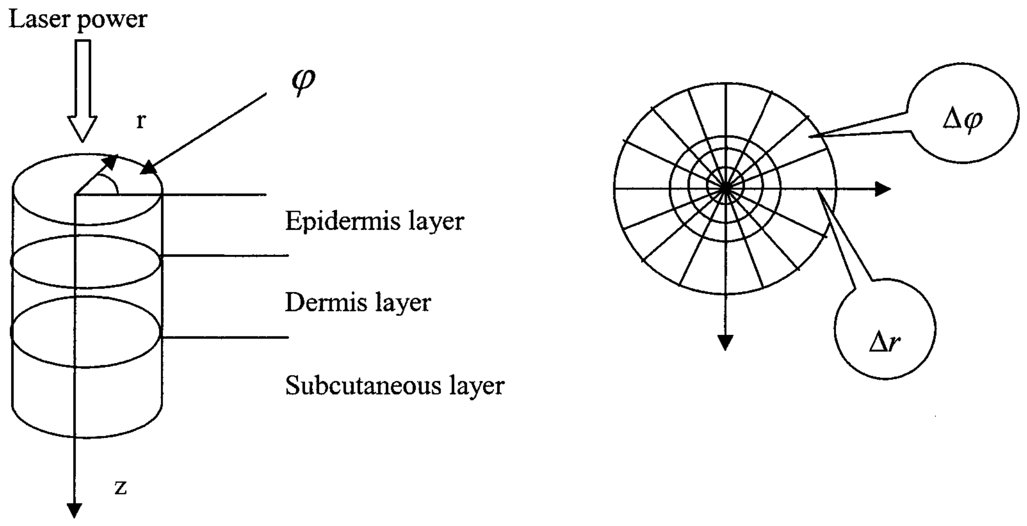

Figure 3.1 Schematic configuration of a 3D triple-layered skin structure irradiated by a laser and grid configuration on the surface

Figure 3.1 shows the 3D cylindrical coordinates, where r is the radius of the target region, ranging from 0 to 0.5mm; φ is the angle between the project of r on the xy-plane with positive x-axis, ranging from 0 to 2π ; and z is the depth of the region.

<u>3.1.3 Governing Equation Introduction</u>

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:

$$\rho_l C_l \frac{\partial \theta_l}{\partial t} + W_b^l C_b^l \theta_l - k_l [\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial \theta_l}{\partial r}) + \frac{1}{r^2}\frac{\partial^2 \theta_l}{\partial \varphi^2} + \frac{\partial^2 \theta_l}{\partial z^2}] = Q_r^l, \quad l = 1,2,3 \tag{3.1}$$

where θ_l is the elevated tissue temperature above the ambient temperature due to heating by a laser, ρ_l C_l and k_l denote density, specific heat, and thermal conductivity of tissue, respectively, furthermore 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 described as follows [Jaesung 1994]:

$$\begin{aligned} Q_1 &= \alpha_1 e^{-\alpha_1 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r\cos\varphi - x_0)^2 + (r\sin\varphi - y_0)^2}{2\sigma^2}} P_0(1-\mathrm{Reff}_1)f(t), \\ Q_2 &= \alpha_2 e^{-\alpha_1 L_1 - \alpha_2 Z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r\cos\varphi - x_0)^2 + (r\sin\varphi - y_0)^2}{2\sigma^2}} P_0(1-\mathrm{Reff}_2)f(t), \\ Q_3 &= \alpha_3 e^{-\alpha_1 L_1 - \alpha_2 L_2 - \alpha_3 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r\cos\varphi - x_0)^2 + (r\sin\varphi - y_0)^2}{2\sigma^2}} P_0(1-\mathrm{Reff}_3)f(t), \end{aligned} \tag{3.2}$$

where $\alpha_1, \alpha_2, \alpha_3$ are laser absorbtivity of the three layers respectively; $\mathrm{Reff}_1, \mathrm{Reff}_2, \mathrm{Reff}_3$ are laser reflectivity of three layers of the skin, respectively; σ 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:

$$\frac{\partial \theta_1}{\partial z} = 0, \qquad z = 0, \tag{3.3}$$

$$\theta_1 = \theta_2,\ k_1 \frac{\partial \theta_1}{\partial z} = k_2 \frac{\partial \theta_2}{\partial z}, \qquad z = L_1, \tag{3.4}$$

$$\theta_2 = \theta_3,\ k_2 \frac{\partial \theta_2}{\partial z} = k_3 \frac{\partial \theta_3}{\partial z}, \qquad z = L_1 + L_2, \tag{3.5}$$

$$\frac{\partial \theta_3}{\partial z} = 0, \qquad z = L_1 + L_2 + L_3. \tag{3.6}$$

On the lateral walls we assume that

$$\frac{\partial \theta_l}{\partial r} = 0, \qquad r = R, \tag{3.7a}$$

and

$$\theta_l(r, \varphi, z) = \theta_l(r, \varphi + 2m\pi, z). \tag{3.7b}$$

The initial condition is assumed to be

$$\theta_l = 0,\ t = 0, \qquad l = 1,2,3. \tag{3.8}$$

3.2 Finite Difference Scheme

3.2.1 Notations and Scheme

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

Here i, j, k are chosen to be $1 \le i \le N_r, 1 \le j \le N_\varphi, 1 \le k \le N_l^z$,so that $N_r \cdot \Delta r = R,\ N_\varphi \cdot \Delta\varphi = 2\pi,\ N_l^z \cdot \Delta z = L_l,\ l = 1, 2, 3.$

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

$$\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial\theta_l}{\partial r}) \approx \frac{(i+\frac{1}{2})\theta_{i+1jk} - 2i\theta_{ijk} + (i-\frac{1}{2})\theta_{i-1jk}}{r_i\Delta r},$$

$$\frac{\partial^2\theta_l}{\partial z^2} \approx \frac{\theta_{ijk+1} - 2\theta_{ijk} + \theta_{ijk-1}}{\Delta z^2},$$

$$\frac{1}{r^2}\frac{\partial^2\theta_l}{\partial\varphi^2} \approx \frac{\theta_{ij+1k} - 2\theta_{ijk} + \theta_{ij-1k}}{r_i^2 \cdot \Delta\varphi^2}.$$

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:

$$\begin{aligned}&\rho_l C_l \frac{(u_l)^{n+1}_{ijk} - (u_l)^n_{ijk}}{\Delta t} + \frac{W^l_b C^l_b}{2}[(u_l)^{n+1}_{ijk} + (u_l)^n_{ijk}] \\ &-\frac{k_l}{2}\{\frac{r_{i+\frac{1}{2}}[(u_l)^{n+1}_{i+1jk} + (u_l)^n_{i+1jk}] - 2r_i[(u_l)^{n+1}_{ijk} + (u_l)^n_{ijk}] + r_{i-\frac{1}{2}}[(u_l)^{n+1}_{i-1jk} + (u_l)^n_{i-1jk}]}{r_i \cdot \Delta r^2} \\ &+\frac{[(u_l)^{n+1}_{ij+1k} + (u_l)^n_{ij+1k}] - 2[(u_l)^{n+1}_{ijk} + (u_l)^n_{ijk}] + [(u_l)^{n+1}_{ij-1k} + (u_l)^n_{ij-1k}]}{r_i^2 \cdot \Delta\varphi^2} \\ &+\delta_z^2[(u_l)^{n+1}_{ijk} + (u_l)^n_{ijk}]\} = (Q^l_r)^{n+\frac{1}{2}}_{ijk}, \qquad l = 1, 2, 3,\end{aligned} \tag{3.9}$$

where $\delta_z^2 u^n_{ijk} = \frac{u_{ijk+1} - 2u_{ijk} + u_{ijk-1}}{\Delta z^2}$, and so on. The discrete interfacial equations are assumed to be, for any time level

$$k_1 \frac{(u_1)^n_{ijN_1^z} - (u_1)^n_{ijN_1^z-1}}{\Delta z} = k_2 \frac{(u_2)^n_{ij1} - (u_2)^n_{ij0}}{\Delta z}, \qquad (u_1)^n_{ijN_1^z} = (u_2)^n_{ij0}, \tag{3.10}$$

and

$$k_2 \frac{(u_2)^n_{ijN_2^z} - (u_1)^n_{i,j,N_{21}^z-1}}{\Delta z} = k_3 \frac{(u_3)^n_{ij1} - (u_3)^n_{ij0}}{\Delta z}, \qquad (u_2)^n_{ijN_2^z} = (u_3)^n_{ij0}. \tag{3.11}$$

The initial and boundary condition are chosen to be

$$(u_l)^0_{ijk} = 0, \tag{3.12}$$

$$(u_1)^n_{ij0} = (u_1)^n_{ij1}, \qquad (u_3)^n_{ijN_3^z} = (u_3)^n_{ijN_3^z-1}, \tag{3.13}$$

$$(u_l)^n_{0jk} = (u_l)^n_{1jk}, \qquad (u_l)^n_{N_rjk} = (u_l)^n_{N_r-1jk}, \qquad (u_l)^n_{ijk} = (u_l)^n_{ij+N\Delta\varphi k}, \tag{3.14}$$

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_o, 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 $(u_{cal})_i (i = 1,2,3 \cdots M)$ at the given locations are obtained, a least squares approach is employed to minimize the difference between the pre-specified elevated temperature θ_{pre} and the temperature distribution u_{cal} as follows:

$$S(P_0) = \sum_{i=0}^{M} [(\theta_{pre}^i - u_{cal}^i)]^2, \; i = 0, 1, \cdots, M. \tag{3.15}$$

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

$$\frac{d}{dP_0} S(P_0) = 2 \sum_{i=0}^{M} (\frac{d(u_{cal})^i}{dP_0})[(\theta_{pre})^i - (u_{cal})^i] = 0. \tag{3.16}$$

Hence, a new P_o can be calculated iteratively as follows:

$$(P_0^{(k+1)}) = (P_0^{(k)}) + (X'X + \alpha^* I)^{-1} X'[(\theta_{pre})^i - (u_{cal})^i], \tag{3.17}$$

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

$$X = \left[\frac{\partial(u_{cal})^0}{\partial P_0} \; \frac{\partial(u_{cal})^1}{\partial P_0} \cdots \frac{\partial(u_{cal})^M}{\partial P_0} \right]^t, \tag{3.18}$$

and

$$\overrightarrow{\theta_{pre}} = \begin{bmatrix} \theta_{pre}^0 \\ \theta_{pre}^1 \\ \vdots \\ \theta_{pre}^M \end{bmatrix}, \quad \overrightarrow{u_{cal}} = \begin{bmatrix} u_{cal}^0 \\ u_{cal}^1 \\ \vdots \\ u_{cal}^M \end{bmatrix}. \tag{3.19}$$

Hence, an algorithm for calculating the required laser power P_o 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 $\theta_{pre}^{i} (i = 1, 2, 3, \cdots, M)$ 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_o and $P_o+\Delta P_o$, 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:

$$\left| \frac{S(P_0^{k+1}) - S(P_0^{k})}{S(P_0^{k+1})} \right| < \varepsilon. \tag{3.20}$$

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:

$$\begin{aligned} L_{pre}^{l}((u_l)_{ijk}^{n+1})^{(I+1)} &= L_{pre}^{l}((u_l)_{ijk}^{n+1})^{(I)} - \omega\{((u_l)_{ijk}^{n+1})^{I} - (u_l)_{ijk}^{n} + \frac{W_b^l C_b^l \Delta t}{2\rho_l C_l}[((u_l)_{ijk}^{n+1})^{I} + (u_l)_{ijk}^{n}] \\ &- \frac{k_l \Delta t}{2\rho_l C_l}[\frac{r_{i+\frac{1}{2}}[((u_l)_{i+1jk}^{n+1})^{I} + (u_l)_{i+1jk}^{n}] - 2r_i[((u_l)_{ijk}^{n+1})^{I} + (u_l)_{ijk}^{n}] + r_{i-\frac{1}{2}}[((u_l)_{i-1jk}^{n+1})^{I} + (u_l)_{i-1jk}^{n}]}{r_i \cdot \Delta r^2} \\ &+ \delta_\varphi^2[((u_l)_{ijk}^{n+1})^{I} + (u_l)_{ijk}^{n}] + \delta_z^2[((u_l)_{ijk}^{n+1})^{I} + (u_l)_{ijk}^{n}]]\} + \frac{\omega \Delta t}{\rho_l C_l}(Q_r^l)_{ijk}^{n+\frac{1}{2}}, \end{aligned} \tag{3.21}$$

where $l = 1, 2, 3,\quad I = 1, 2, 3, ..., \text{n}$ and $\delta_\varphi^2 = \frac{1}{r_i^2} \cdot \frac{(u_l)_{ij+1k}^n - 2(u_l)_{ijk}^n + (u_l)_{ij-1k}^n]}{\Delta\varphi^2}$,

$\delta_z^2 = \frac{\theta_{ijk+1} - 2\theta_{ijk} + \theta_{ijk-1}}{\Delta z^2}$ and the preconditioned operator

$$L_{pre}^l = 1 + \frac{W_b^l C_b^l \Delta t}{2\rho_l C_l} + \frac{k_l \Delta t}{2\rho_l C_l}\left(\frac{2r_i + r_{i+\frac{1}{2}} + r_{i-\frac{1}{2}}}{r_i \Delta r^2} + \frac{4}{\Delta\varphi^2 \cdot r_i^2}\right)I - \frac{k_l \Delta t}{2\rho_l C_l}\delta, \tag{3.22}$$

and ω is a relaxation parameter $(0 \le \omega \le 1)$. 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 $A\vec{u} = \vec{d}$.

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 $(u_l)_{0jk}^n = (u_l)_{1jk}^n$ and introduce the definitions of the inner products and norms between the mesh functions u_{ijk}^n and v_{ijk}^n as follows:

$$\begin{aligned}
&(u^n, v^n)_l = \Delta r \Delta\varphi \Delta z \sum_{i=1}^{N_r - 1} \sum_{j=1}^{N\varphi} \sum_{k=1}^{N_l^z - 1} u_{ijk}^n v_{ijk}^n, \quad \left\|u^n\right\|_l^2 = (u^n, u^n)_l, \\
&\| \nabla_{\bar{r}} u^n \|_l = (\nabla_{\bar{r}} u^n, \nabla_{\bar{r}} u^n)_l = \Delta r \Delta\varphi \Delta z \sum_{i=1}^{N_r - 1} \sum_{j=1}^{N\varphi} \sum_{k=1}^{N_l^z - 1} (\nabla_{\bar{r}} u_{ijk}^n)_l^2, \\
&\| \nabla_{\bar{r}} u^n \|_{l,1} = (\nabla_{\bar{r}} u^n, \nabla_{\bar{r}} u^n)_{l,1} = \Delta r \Delta\varphi \Delta z \sum_{i=1}^{N_r} \sum_{j=1}^{N\varphi} \sum_{k=1}^{N_l^z - 1} (\nabla_{\bar{r}} u_{ijk}^n)_{l,1}^2
\end{aligned} \tag{3.23}$$

where $l = 1,2,3$, and $\nabla_{\bar{r}}$ is the first-order backward finite difference operator such that

$\nabla_{\bar{r}} u_{ijk}^n = \frac{u_{ijk}^n - u_{i-1jk}^n}{\Delta r}$, and so on for the φ and z directions.

LEMMA 1. If $(u_l)_{ijk}^n, l = 1,2,3,$ is the solution of Eqs. (3.9)-(3.14), then

$$k_1\sum_{k=1}^{N_1^z-1}\delta_z^2[(u_1)_{ijk}^{n+1}+(u_1)_{ijk}^n]\cdot[(u_1)_{ijk}^{n+1}+(u_1)_{ijk}^n]$$
$$+k_2\sum_{k=1}^{N_2^z-1}\delta_z^2[(u_2)_{ijk}^{n+1}+(u_2)_{ijk}^n]\cdot[(u_2)_{ijk}^{n+1}+(u_2)_{ijk}^n]$$
$$+k_3\sum_{k=1}^{N_3^z-1}\delta_z^2[(u_3)_{ijk}^{n+1}+(u_3)_{ijk}^n]\cdot[(u_3)_{ijk}^{n+1}+(u_3)_{ijk}^n] \tag{3.24}$$
$$=-k_1\sum_{k=1}^{N_1^z}\nabla_{\bar z}[(u_1)_{ijk}^{n+1}+(u_1)_{ijk}^n]^2-k_2\sum_{k=1}^{N_2^z}\nabla_{\bar z}[(u_2)_{ijk}^{n+1}+(u_2)_{ijk}^n]^2$$
$$-k_3\sum_{k=1}^{N_3^z}\nabla_{\bar z}[(u_3)_{ijk}^{n+1}+(u_3)_{ijk}^n]^2,$$

$$\sum_{i=1}^{N_r-1}r_iP_r^2[(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^n]\cdot[(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^n]=-\sum_{i=1}^{N_r-1}r_{i-\frac{1}{2}}\nabla_{\bar r}[(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^n]^2 \tag{3.25}$$

and

$$\sum_{j=1}^{N_r}r_i\delta_\varphi^2[(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^n]\cdot[(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^n]=-\sum_{j=1}^{N_r}\frac{1}{r_i}\nabla_{\bar\varphi}[(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^n]^2. \tag{3.26}$$

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

$$LHS=k_1\sum_{k=1}^{N_1^z-1}\delta_z^2U_k^{(1)}\cdot U_k^{(1)}+k_2\sum_{k=1}^{N_2^z-1}\delta_z^2U_k^{(2)}\cdot U_k^{(2)}+k_3\sum_{k=1}^{N_3^z-1}\delta_z^2U_k^{(3)}\cdot U_k^{(3)}$$
$$=\frac{1}{\Delta z^2}k_1\sum_{k=1}^{N_1^z-1}[(U_{k+1}^{(1)}-U_k^{(1)})-(U_k^{(1)}-U_{k-1}^{(1)})]\cdot U_k^{(1)}$$
$$+\frac{1}{\Delta z^2}k_2\sum_{k=1}^{N_2^z-1}[(U_{k+1}^{(2)}-U_k^{(2)})-(U_k^{(2)}-U_{k-1}^{(2)})]\cdot U_k^{(2)}$$
$$+\frac{1}{\Delta z^2}k_3\sum_{k=1}^{N_3^z-1}[(U_{k+1}^{(3)}-U_k^{(3)})-(U_k^{(3)}-U_{k-1}^{(3)})]\cdot U_k^{(3)}$$

$$
\begin{aligned}
&= \frac{1}{\Delta z^2} k_1 [\sum_{k=2}^{N_1^z} (U_k^{(1)} - U_{k-1}^{(1)}) \cdot U_{k-1}^{(1)} - \sum_{k=1}^{N_1^z - 1} (U_k^{(1)} - U_{k-1}^{(1)}) \cdot U_k^{(1)}] \\
&+ \frac{1}{\Delta z^2} k_2 [\sum_{k=2}^{N_2^z} (U_k^{(2)} - U_{k-1}^{(2)}) \cdot U_{k-1}^{(2)} - \sum_{k=1}^{N_2^z - 1} (U_k^{(2)} - U_{k-1}^{(2)}) \cdot U_k^{(2)}] \\
&+ \frac{1}{\Delta z^2} k_3 [\sum_{k=2}^{N_3^z} (U_k^{(3)} - U_{k-1}^{(3)}) \cdot U_{k-1}^{(3)} - \sum_{k=1}^{N_3^z - 1} (U_k^{(3)} - U_{k-1}^{(3)}) \cdot U_k^{(3)}].
\end{aligned}
\tag{3.27}
$$

Based on Eq. (3.13), the LHS can be further written as follows:

$$
\begin{aligned}
LHS &= \frac{1}{\Delta z^2} k_1 [\sum_{k=2}^{N_1^z - 1} (U_k^{(1)} - U_{k-1}^{(1)}) \cdot U_{k-1}^{(1)} - \sum_{k=1}^{N_1^z - 1} (U_k^{(1)} - U_{k-1}^{(1)}) \cdot U_k^{(1)}] \\
&+ \frac{1}{\Delta z^2} k_2 [\sum_{k=1}^{N_2^z - 1} (U_k^{(2)} - U_{k-1}^{(2)}) \cdot U_{k-1}^{(2)} - \sum_{k=1}^{N_2^z - 1} (U_k^{(2)} - U_{k-1}^{(2)}) \cdot U_k^{(2)}] \\
&+ \frac{1}{\Delta z^2} k_3 [\sum_{k=1}^{N_3^z - 1} (U_k^{(3)} - U_{k-1}^{(3)}) \cdot U_{k-1}^{(3)} - \sum_{k=1}^{N_3^z - 1} (U_k^{(3)} - U_{k-1}^{(3)}) \cdot U_k^{(3)}] \\
&+ \frac{1}{\Delta z^2} k_1 (U_{N_1^z}^{(1)} - U_{N_1^z - 1}^{(1)}) \cdot U_{N_1^z - 1}^{(1)} - \frac{1}{\Delta z^2} k_2 (U_1^{(2)} - U_0^{(2)}) \cdot U_0^{(2)} \\
&+ \frac{1}{\Delta z^2} k_2 (U_{N_2^z}^{(2)} - U_{N_2^z - 1}^{(2)}) \cdot U_{N_2^z - 1}^{(2)} - \frac{1}{\Delta z^2} k_3 (U_1^{(3)} - U_0^{(3)}) \cdot U_0^{(3)}.
\end{aligned}
\tag{3.28}
$$

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

$$
\begin{aligned}
LHS &= -k_1 \sum_{k=1}^{N_1^z - 1} \nabla_{\bar{z}} U_k^{(1)} \cdot \nabla_{\bar{z}} U_k^{(1)} - k_2 \sum_{k=1}^{N_2^z - 1} \nabla_{\bar{z}} U_k^{(2)} \cdot \nabla_{\bar{z}} U_k^{(2)} - k_3 \sum_{k=1}^{N_3^z - 1} \nabla_{\bar{z}} U_k^{(3)} \cdot \nabla_{\bar{z}} U_k^{(3)} \\
&+ \frac{1}{\Delta z^2} k_1 (U_{N_1^z}^{(1)} - U_{N_1^z - 1}^{(1)}) \cdot U_{N_1^z - 1}^{(1)} - \frac{1}{\Delta z^2} k_1 (U_{N_1^z}^{(1)} - U_{N_1^z - 1}^{(1)}) \cdot U_{N_1^z}^{(1)} \\
&+ \frac{1}{\Delta z^2} k_2 (U_{N_2^z}^{(2)} - U_{N_2^z - 1}^{(2)}) \cdot U_{N_2^z - 1}^{(2)} - \frac{1}{\Delta z^2} k_2 (U_{N_2^z}^{(2)} - U_{N_2^z - 1}^{(2)}) \cdot U_{N_2^z}^{(2)} \\
&= -k_1 \sum_{k=1}^{N_1^z} \nabla_{\bar{z}} U_k^{(1)} \cdot \nabla_{\bar{z}} U_k^{(1)} - k_2 \sum_{k=1}^{N_2^z} \nabla_{\bar{z}} U_k^{(2)} \cdot \nabla_{\bar{z}} U_k^{(2)} - k_3 \sum_{k=1}^{N_3^z} \nabla_{\bar{z}} U_k^{(3)} \cdot \nabla_{\bar{z}} U_k^{(3)},
\end{aligned}
\tag{3.29}
$$

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 $(u_l)_{ijk}^n$ and $(v_l)_{ijk}^n, l = 1, 2, 3,$ 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, $(Q_1)_r^l$ and $(Q_2)_r^l$. We let $(\varepsilon_l)_{ijk}^n = (u_l)_{ijk}^n - (v_l)_{ijk}^n$ and $\sigma_l = (Q_1)_r^l - (Q_2)_r^l$. One may see that $(\varepsilon_l)_{ijk}^n$ satisfies Eqs. (3.10)-(3.14) and the following equation:

$$\begin{aligned}&\rho_l C_l \frac{(\varepsilon_l)_{ijk}^{n+1} - (\varepsilon_l)_{ijk}^n}{\Delta t} + W_b^l C_b^l \frac{(\varepsilon_l)_{ijk}^{n+1} + (\varepsilon_l)_{ijk}^n}{2} \\ &- k_l (P_r^2 + \delta_\varphi^2 + \delta_z^2) \frac{(\varepsilon_l)_{ijk}^{n+1} + (\varepsilon_l)_{ijk}^n}{2} = (\sigma_l)_{ijk}^{n+\frac{1}{2}}, \qquad l = 1,\ 2,\ 3.\end{aligned} \tag{3.30}$$

Multiply Eq. (3.30) with $l=1$ by $2r_i \Delta r \Delta\varphi \Delta z \Delta t[(\varepsilon_1)_{ijk}^{n+1} + (\varepsilon_1)_{ijk}^n]$, Eq. (3.30) with $l=2$ by $2r_i \Delta r \Delta\varphi \Delta z \Delta t[(\varepsilon_2)_{ijk}^{n+1} + (\varepsilon_2)_{ijk}^n]$, and Eq. (3.46) with $l=3$ by $2r_i \Delta r \Delta\varphi \Delta z \Delta t[(\varepsilon_3)_{ijk}^{n+1} + (\varepsilon_3)_{ijk}^n]$.

Then, sum over i,j,k from $1 \le i \le N_r - 1, 1 \le j \le N_\varphi, 1 \le k \le N_l^z - 1$, respectively, adding them together and then using lemma 1, one obtains

$$\begin{aligned}&\sum_{l=1}^{3} \rho_l C_l [\| \sqrt{r}(\varepsilon_l)^{n+1} \|^2 - \| \sqrt{r}(\varepsilon_l)^n \|^2] + \Delta t \sum_{l=1}^{3} W_b^l C_b^l \| \sqrt{r}[(\varepsilon_l)^{n+1} + (\varepsilon_l)^n] \|^2 \\ &+ \Delta t \sum_{l=1}^{3} k_l \left\| \sqrt{E^{-\frac{1}{2}} r}[\nabla_{\bar{r}}(\varepsilon_l)^{n+1} + \nabla_{\bar{r}}(\varepsilon_l)^n] \right\|_l^2 + \Delta t \sum_{l=1}^{3} k_l \left\| \sqrt{\frac{1}{r}}[\nabla_{\bar{\varphi}}(\varepsilon_l)^{n+1} + \nabla_{\bar{\varphi}}(\varepsilon_l)^n] \right\|_l^2 \\ &+ \Delta t \sum_{l=1}^{3} k_l \left\| \sqrt{r}[\nabla_{\bar{z}}(\varepsilon_l)^{n+1} + \nabla_{\bar{z}}(\varepsilon_l)^n] \right\|_{l,1}^2 = 2\Delta t \sum_{l=1}^{3} ((\sigma_l)^{n+\frac{1}{2}}, r[(\varepsilon_l)^{n+1} + (\varepsilon_l)^n]),\end{aligned} \tag{3.31}$$

where $E^{-\frac{1}{2}}$ is a shift operator such that $E^{-\frac{1}{2}} r_i = r_{i-\frac{1}{2}}$. By the generalized Cauchy-Schwarz's inequality, we have

$$2((\sigma_l)^{n+\frac{1}{2}}, r[(\varepsilon_l)^{n+1} + (\varepsilon_l)^n]) \le \varepsilon \left\| \sqrt{r}[(\varepsilon_l)^{n+1} + (\varepsilon_l)^n] \right\|^2 + \varepsilon^{-1} \left\| \sqrt{r}(\sigma_l)^{n+\frac{1}{2}} \right\|^2$$
$$\le 2\varepsilon \left\| \sqrt{r}(\varepsilon_l)^{n+1} \right\|^2 + 2\varepsilon \left\| \sqrt{r}(\varepsilon_l)^n \right\|^2 + \varepsilon^{-1} \left\| \sqrt{r}(\sigma_l)^{n+\frac{1}{2}} \right\|^2, \tag{3.32}$$

where ε is a positive constant. Substituting Eq. (3.32) in to Eq.(3.31), we obtain

$$\sum_{l=1}^{3}(2\rho_l C_l - 2\varepsilon\Delta t) \| \sqrt{r}(\varepsilon_l)^{n+1} \|^2 + \Delta t \sum_{l=1}^{3} W_b^l C_b^l \| \sqrt{r}[(\varepsilon_l)^{n+1} + (\varepsilon_l)^n] \|^2$$
$$+ \Delta t \sum_{l=1}^{3} k_l \left\| \sqrt{E^{-\frac{1}{2}} r} [\nabla_{\bar{r}}(\varepsilon_l)^{n+1} + \nabla_{\bar{r}}(\varepsilon_l)^n] \right\|_l^2 + \Delta t \sum_{l=1}^{3} k_l \left\| \sqrt{\frac{1}{r}} [\nabla_{\bar{\varphi}}(\varepsilon_l)^{n+1} + \nabla_{\bar{\varphi}}(\varepsilon_l)^n] \right\|_l^2$$
$$+ \Delta t \sum_{l=1}^{3} k_l \left\| \sqrt{r} [\nabla_{\bar{z}}(\varepsilon_l)^{n+1} + \nabla_{\bar{z}}(\varepsilon_l)^n] \right\|_{l,1}^2 \le \sum_{l=1}^{3} (2\rho_l C_l + 2\varepsilon\Delta t) \left\| \sqrt{r}(\varepsilon_l)^n \right\|^2$$
$$+ \Delta t \sum_{l=1}^{3} \varepsilon^{-1} \left\| \sqrt{r}(\sigma_l)^{n+\frac{1}{2}} \right\|^2, \tag{3.33}$$

We denote $F(n) = \sum_{l=1}^{3} 2\rho_l C_l \left\| \sqrt{r}(\varepsilon_l)^n \right\|^2$. 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:

$$(1 - \Delta t)F(n+1) \le (1 + \Delta t)F(n) + \Delta t \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \left\| \sqrt{r}(\sigma_l)^{n+\frac{1}{2}} \right\|^2. \tag{3.34}$$

Thus, we obtain

$$F(n+1) \le \frac{1+\Delta t}{1-\Delta t} F(n) + \frac{\Delta t}{1-\Delta t} \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \left\| \sqrt{r}(\sigma_l)^{n+\frac{1}{2}} \right\|^2$$
$$\le \frac{1+\Delta t}{1-\Delta t} \left[\frac{1+\Delta t}{(1-\Delta t)} F(n-1) + \frac{\Delta t}{1-\Delta t} \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \left\| \sqrt{r}(\sigma_l)^{n-\frac{1}{2}} \right\|^2 \right]$$
$$+ \frac{\Delta t}{1-\Delta t} \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \left\| \sqrt{r}(\sigma_l)^{n+\frac{1}{2}} \right\|^2 \le \cdots$$

$$\le \left(\frac{1+\Delta t}{1-\Delta t}\right)^{n+1} F(0) + \frac{\Delta t}{1-\Delta t}[1 + \frac{1+\Delta t}{1-\Delta t} + \cdots$$

$$+ \left(\frac{1+\Delta t}{1-\Delta t}\right)^{n}]\sum_{l=1}^{3} \frac{1}{\rho_l C_l} \max_{0\le m\le n} \left\| \sqrt{r}(\sigma_l)^{m+\frac{1}{2}} \right\|^2$$

$$\le \left(\frac{1+\Delta t}{1-\Delta t}\right)^{n+1} F(0) + \frac{\Delta t}{1-\Delta t} \left[\frac{1-(\frac{1+\Delta t}{1-\Delta t})^{n+1}}{1-(\frac{1+\Delta t}{1-\Delta t})} \right] \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \max_{0\le m\le n} \left\| \sqrt{r}(\sigma_l)^{m+\frac{1}{2}} \right\|^2 \qquad (3.35)$$

$$\le \left(\frac{1+\Delta t}{1-\Delta t}\right)^{n+1} [F(0) + \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \max_{0\le m\le n} \left\| \sqrt{r}(\sigma_l)^{m+\frac{1}{2}} \right\|^2].$$

Using the inequalities $(1+\varepsilon)^n \le e^{n\varepsilon}$ for $\varepsilon > 0$, and $(1-\varepsilon)^{-1} \le e^{2\varepsilon}$ when $0 < \varepsilon \le \frac{1}{2}$, we obtain

$$F(n+1) \le e^{3(n+1)\Delta t}[F(0) + \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \max_{0\le m\le n} \left\| \sqrt{r}(\sigma_l)^{m+\frac{1}{2}} \right\|^2], \qquad (3.36)$$

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

$$F(n+1) \le e^{3t_0} \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \max_{0\le m\le n} \left\| \sqrt{r}(\sigma_l)^{m+\frac{1}{2}} \right\|^2 , \qquad (3.37)$$

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

THEOREM. Assume that solution $(u_l)^n_{ijk}$ and $(v_l)^n_{ijk}, l = 1,2,3$ 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, $(Q_1)^l_r$ and $(Q_2)^l_r$. Let $(\varepsilon_l)^n_{ijk} = (u_l)^n_{ijk} - (v_l)^n_{ijk}$ and $\sigma_l = (Q_1)^l_r - (Q_2)^l_r$. Then $(\varepsilon_l)^n_{ijk}$ satisfies, for $0 \le n\Delta t \le t_0$,

$$\sum_{l=1}^{3} 2\rho_l C_l \left\| \sqrt{r}(\varepsilon_l)^n \right\|^2 \le e^{3t_0} \sum_{l=1}^{3} \frac{1}{\rho_l C_l} \max_{0 \le m \le n-1} \left\| \sqrt{r}(\sigma_l)^{m+\frac{1}{2}} \right\|, \tag{3.38}$$

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 $S_p^{Specified}$, 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:

Step1. 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 counter-clockwise 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.

Step2. 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 $u_{cal}^{Center} < \theta_{pre}^{Perimeter}$ or $u_{cal}^{Perimeter} > \theta_{pre}^{Perimeter}$ {
        if $S_p(P_0) < S_p^{specified}$, 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 $S_p(P_0) < S_p^{specified}$, record time T1 and begin step 3;
```

if $u_{cal}^{Center} > \theta_{pre}^{Center}$, set flag to 0 and go to procedure one

}

Step3. Heat the center with power P_0 till $u_{cal}^{Center} > \theta_{pre}^{Center}$ and record ΔT (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 ΔT. The above algorithm is illustrated by a flowchart shown in Figure 3.2.

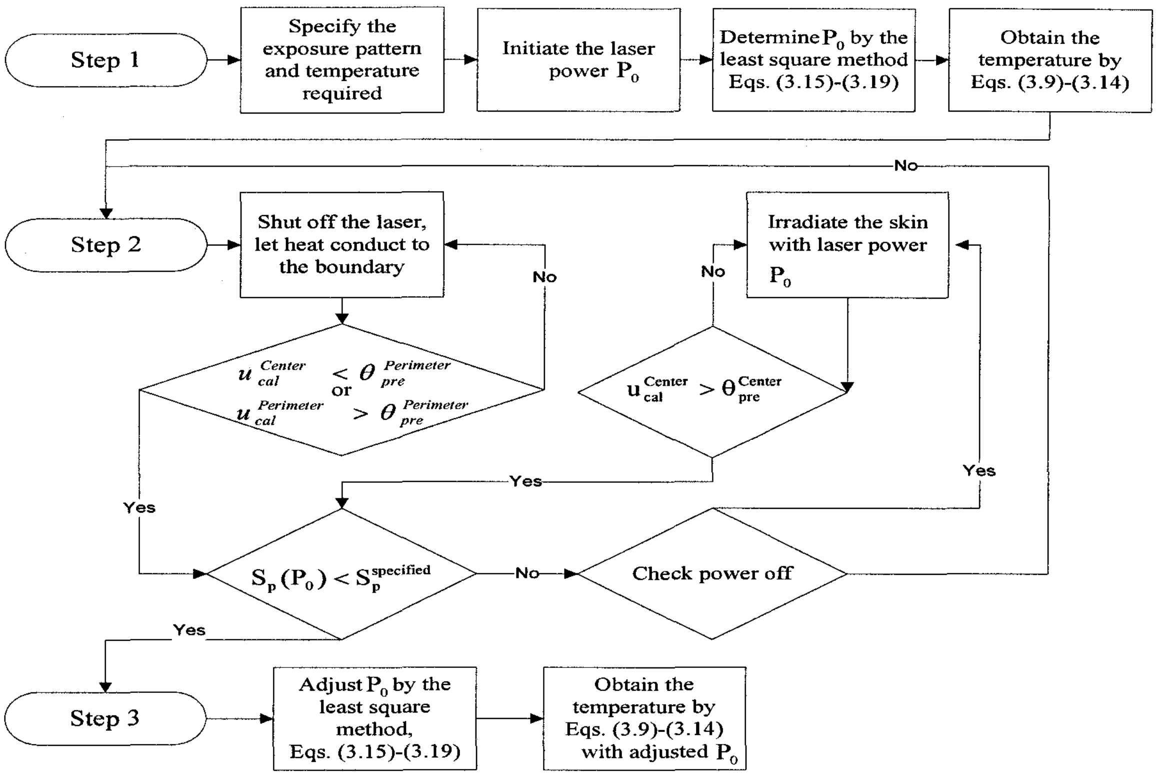
Step 1
Specify the exposure pattern and temperature required
Initiate the laser power P_0
Determine P_0 by the least square method Eqs. (3.15)-(3.19)
Obtain the temperature by Eqs. (3.9)-(3.14)
No
Step 2
Shut off the laser, let heat conduct to the boundary
No
No
Irradiate the skin with laser power P_0
$u_{cal}^{Center} < \theta_{pre}^{Perimeter}$ or $u_{cal}^{Perimeter} > \theta_{pre}^{Perimeter}$
$u_{cal}^{Center} > \theta_{pre}^{Center}$
Yes
Yes
Yes
$S_p(P_0) < S_p^{specified}$
No
Check power off
Yes
Step 3
Adjust P_0 by the least square method, Eqs. (3.15)-(3.19)
Obtain the temperature by Eqs. (3.9)-(3.14) with adjusted P_0

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, φ, 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 Δr 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.

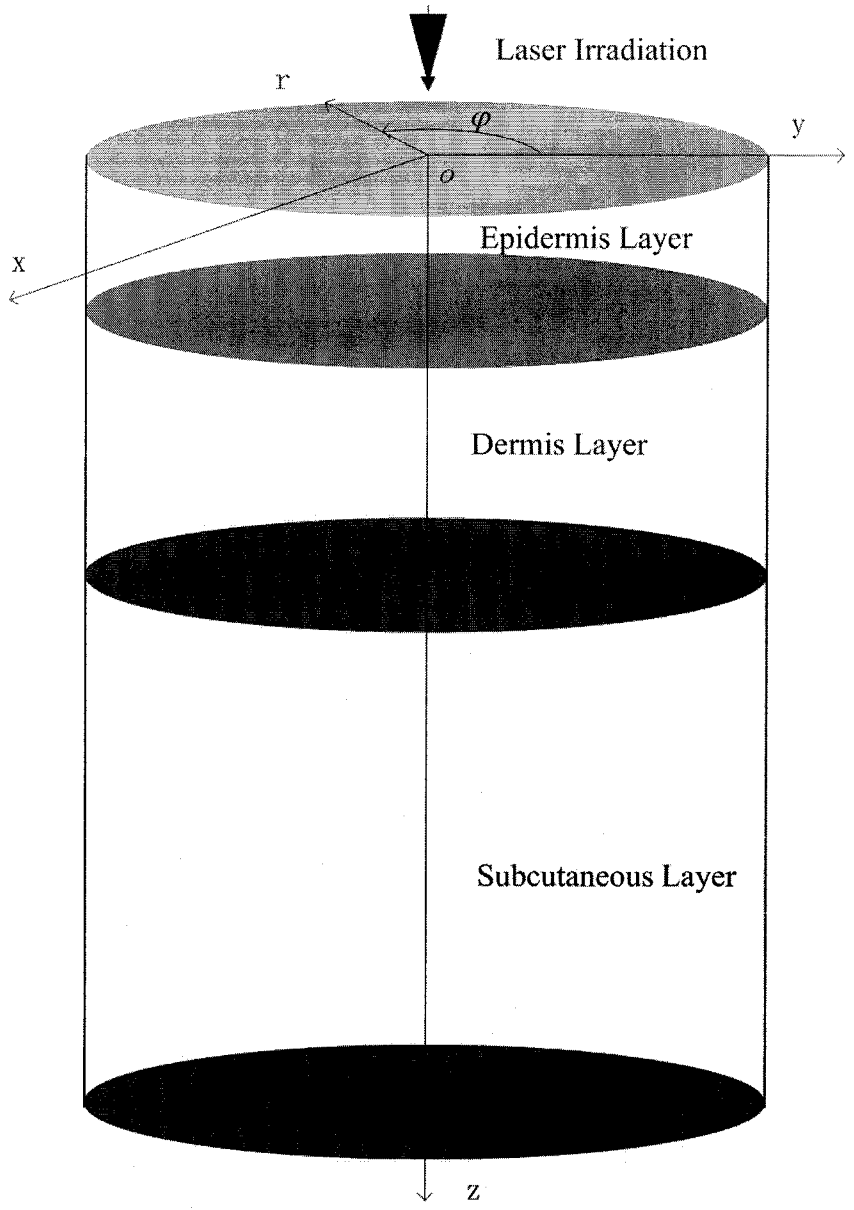

Figure 4.1 A 3D skin structure

Table 4.1 Parameters for a 3D skin structure

Parameter	**Value**
$C_1\ (J/g^oC)$	3.6
$C_2\ (J/g^oC)$	3.4
$C_3\ (J/g^oC)$	3.06
$Cb^1\ (J/g^oC)$	0
$Cb^2\ (J/g^oC)$	4.2
$Cb^3\ (J/g^oC)$	4.2
$C_B((J/m^3K)$	4.134
K1 $(W/cm\ ^oC)$	0.0026
K2 $(W/cm\ ^oC)$	0.0052
K3 $((W/cm\ ^oC)$	0.0021
Reff_1	0.93
Reff_2	0.93
Reff_3	0.93
$Wb^1\ (\ (g/cm^3)$	0
$Wb^2\ (g/cm^3)$	0.0005
$Wb^3\ (g/cm^3)$	0.0005
$\alpha((w/m^2K)$	2000
α_1	1
α_2	0.8
α_3	0.4
$\rho_1\,(g/cm^3)$	1.2
$\rho_2\,(g/cm^3)$	1.2
$\rho_3\,(g/cm^3)$	1
σ (cm)	0.1
$v((m/s)$	0.08

Note: The data is coming from [Liu 1997].

Table 4.2 Pre-specified geometry parameters for the skin structure

Parameter	**Value**
Bi	2
$R(cm)$	0.5
$L_1(cm)$	0.008
$L_2(cm)$	0.02
$L_3(cm)$	1
ω	1
$S_P^{Specified}$	0.04
Δr	1/30
Δt	0.1
Δz	0.001
$\Delta\varphi$	$\frac{2\pi}{20}$
N_1^z	8
N_2^z	208
N_2^z	1208
R_{bv}	2
N_r	30
N_φ	20

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 $\sum_{i=1}^{4}\left[\frac{\theta_{pre}^{i} - u_{cal}^{i}}{\theta_{pre}^{i}}\right]^2$, which shows the difference between the pre-specified temperatures and the calculated temperatures, is 0.513821.

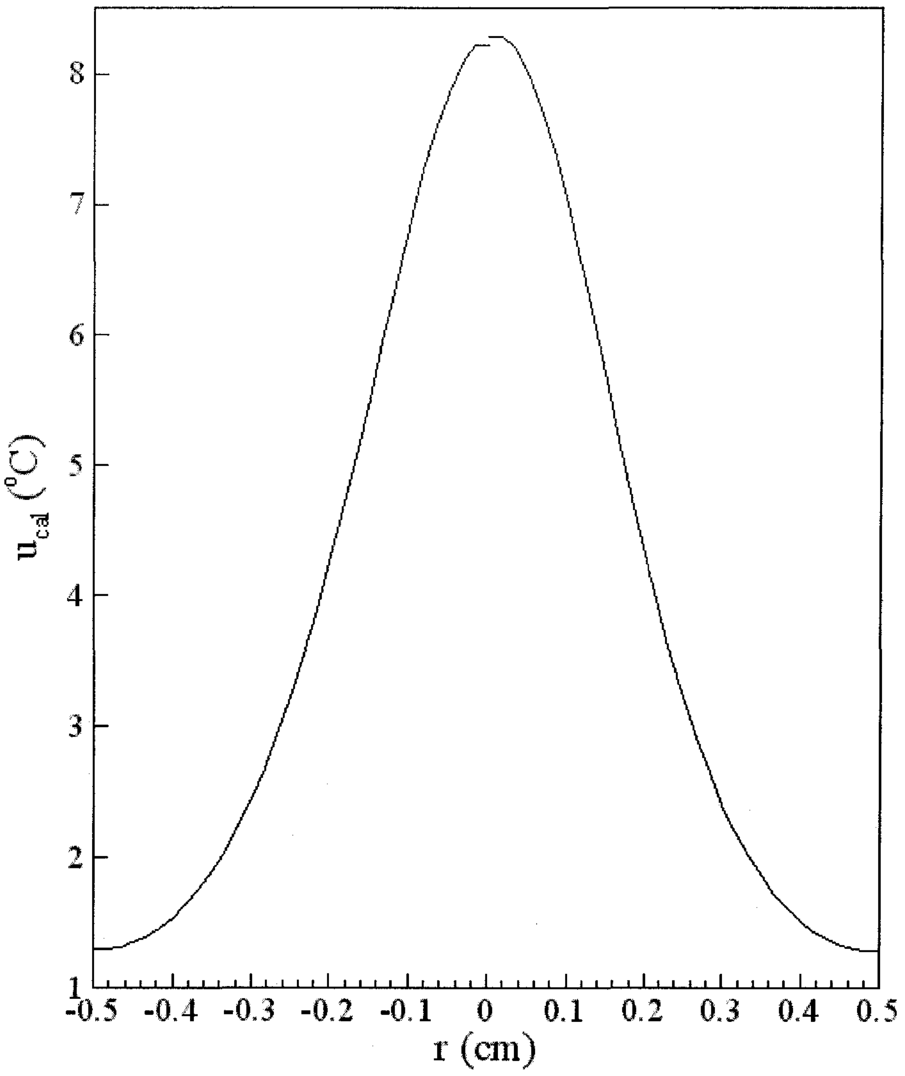

Figure 4.2 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ seconds without any blood vessels

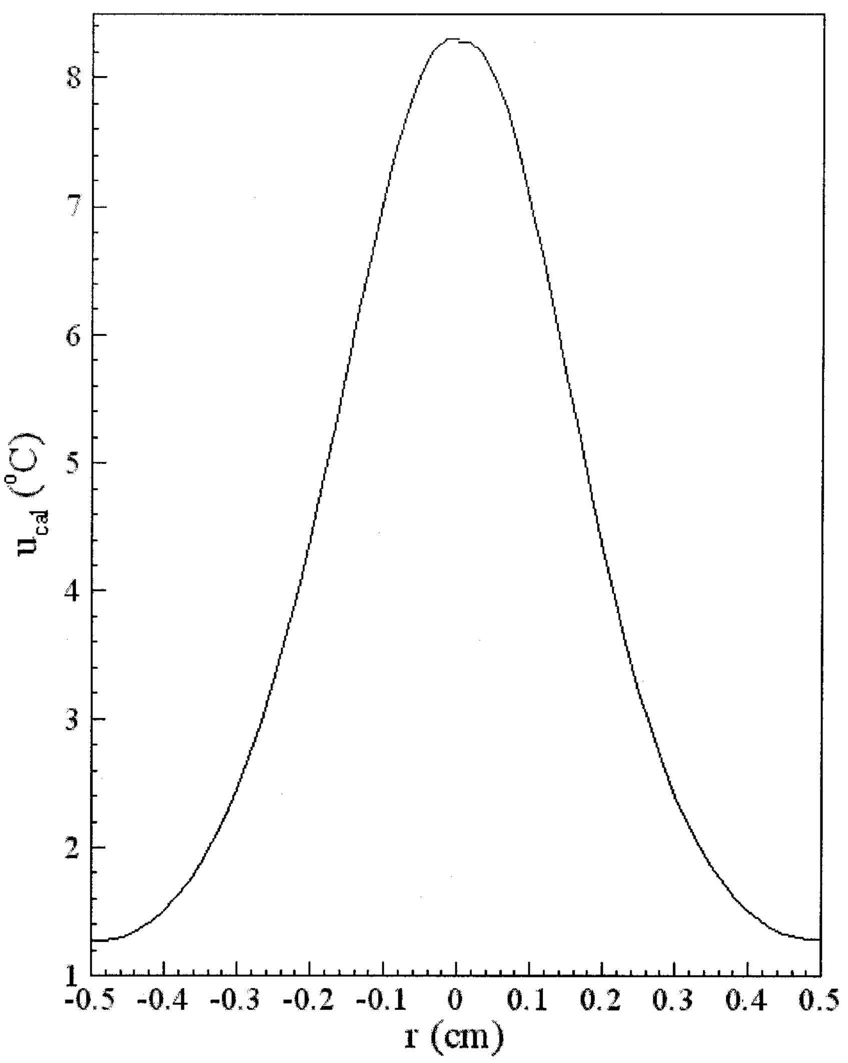

Figure 4.3 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 410$ seconds without any blood vessels

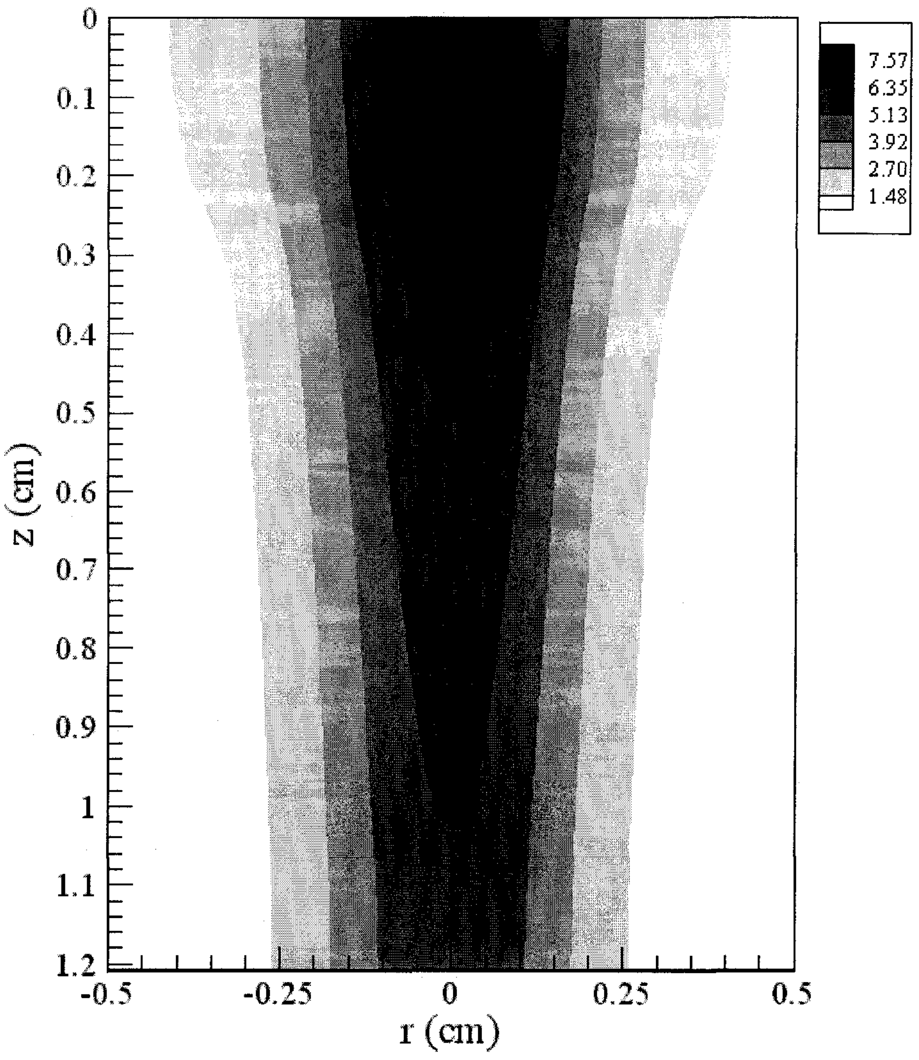

Figure 4.4 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ seconds without any blood vessels

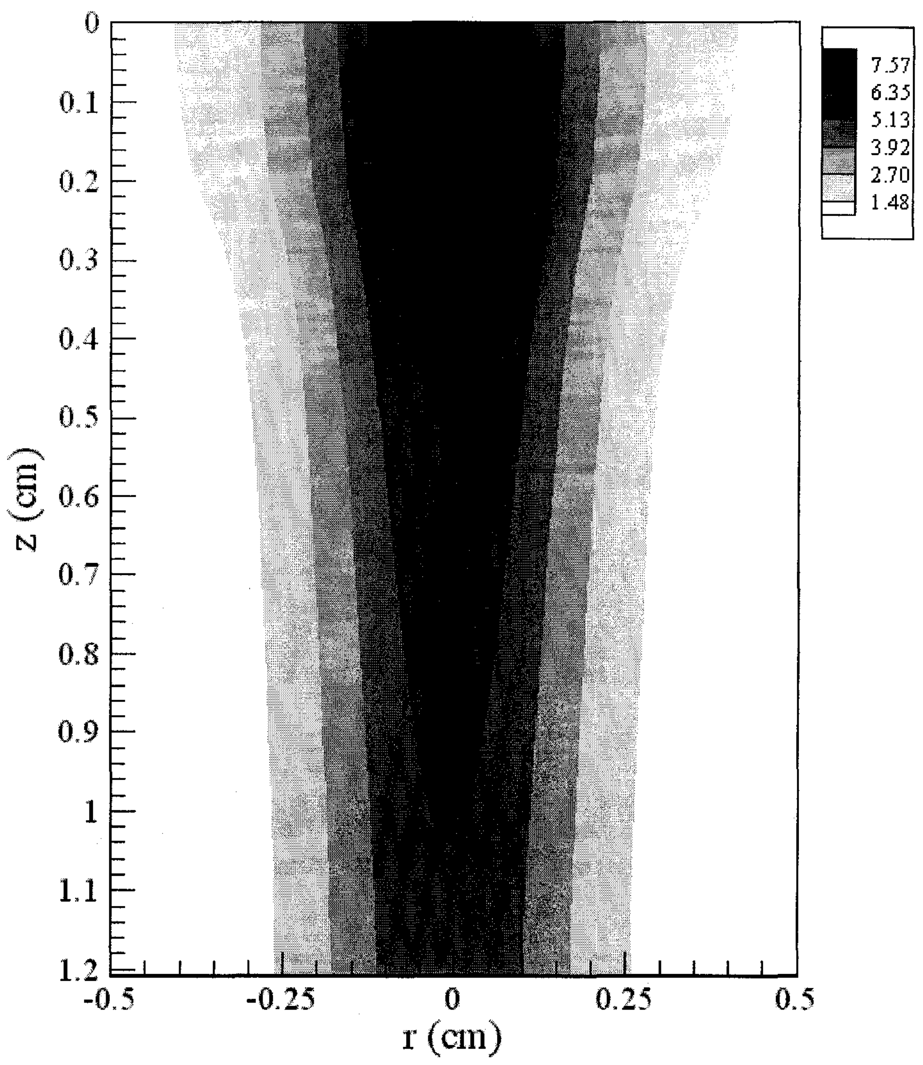

Figure 4.5 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 410$ seconds without any blood vessels

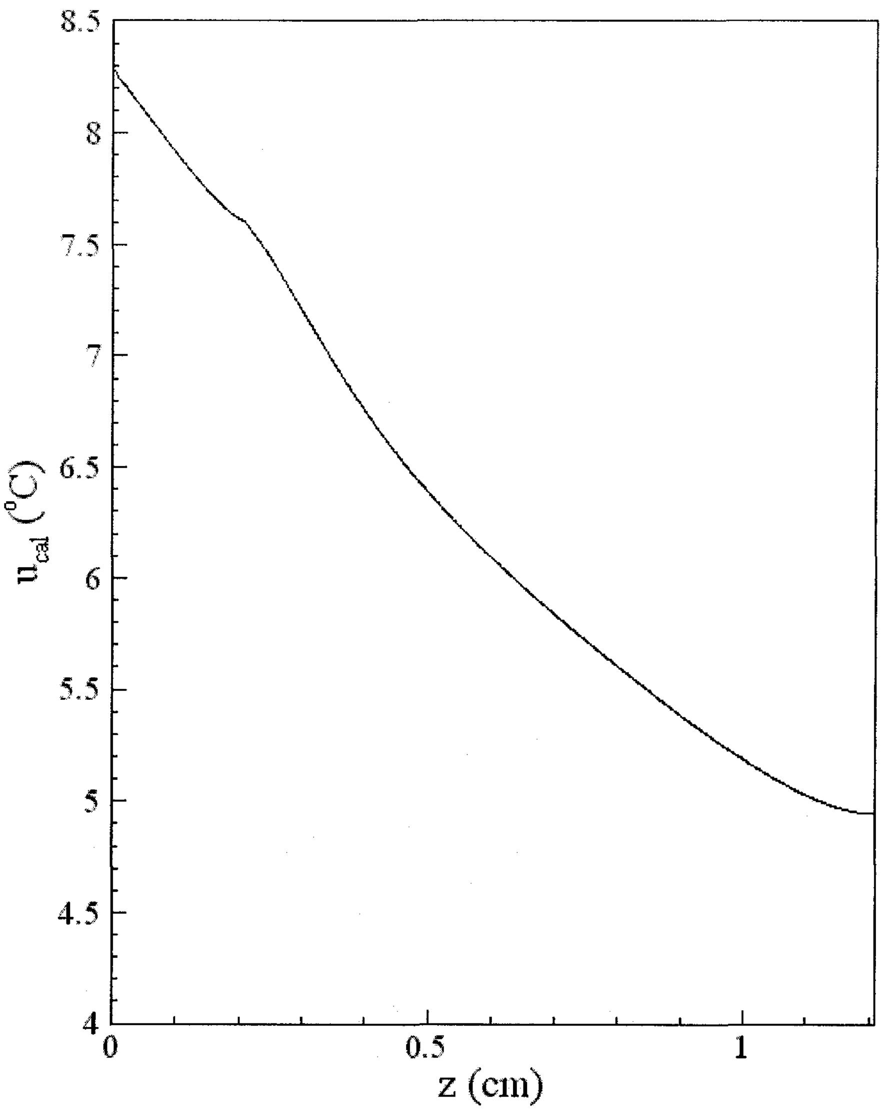

Figure 4.6 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 410$ seconds without any blood vessels

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.

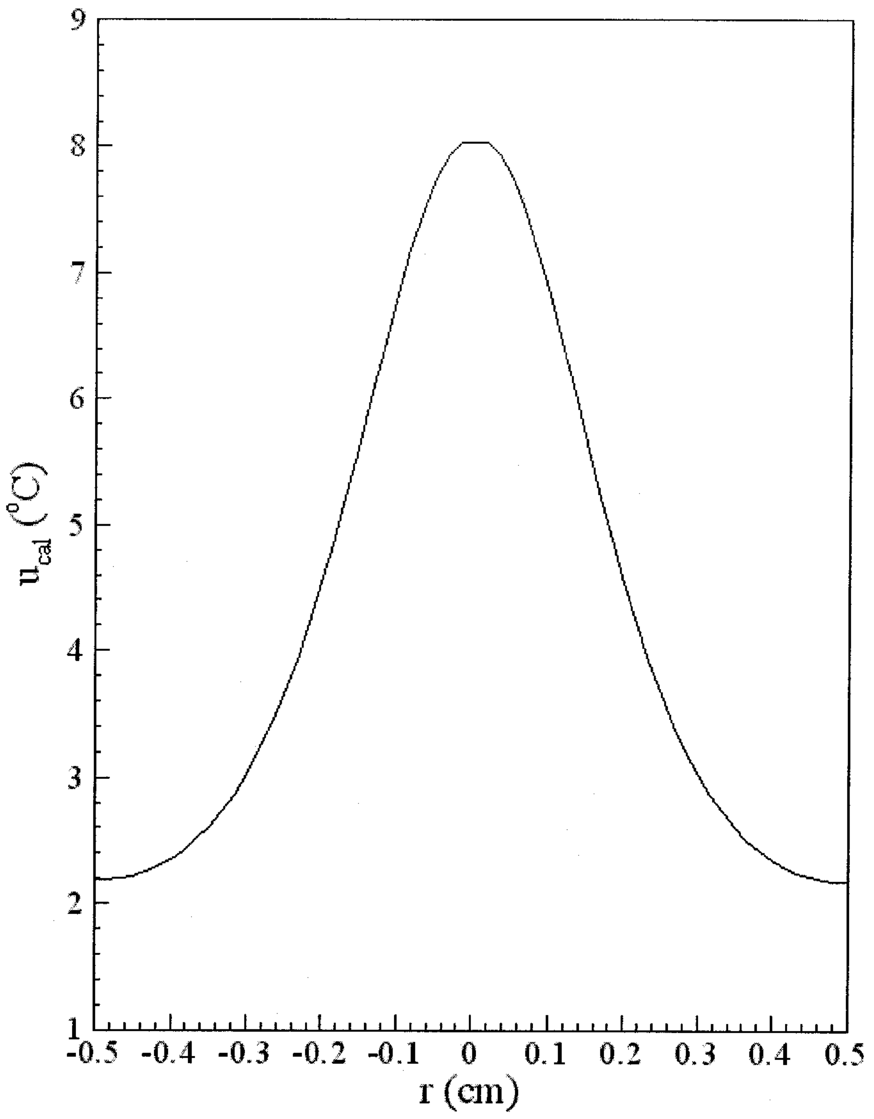

Figure 4.7 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 674$ seconds without any blood vessels

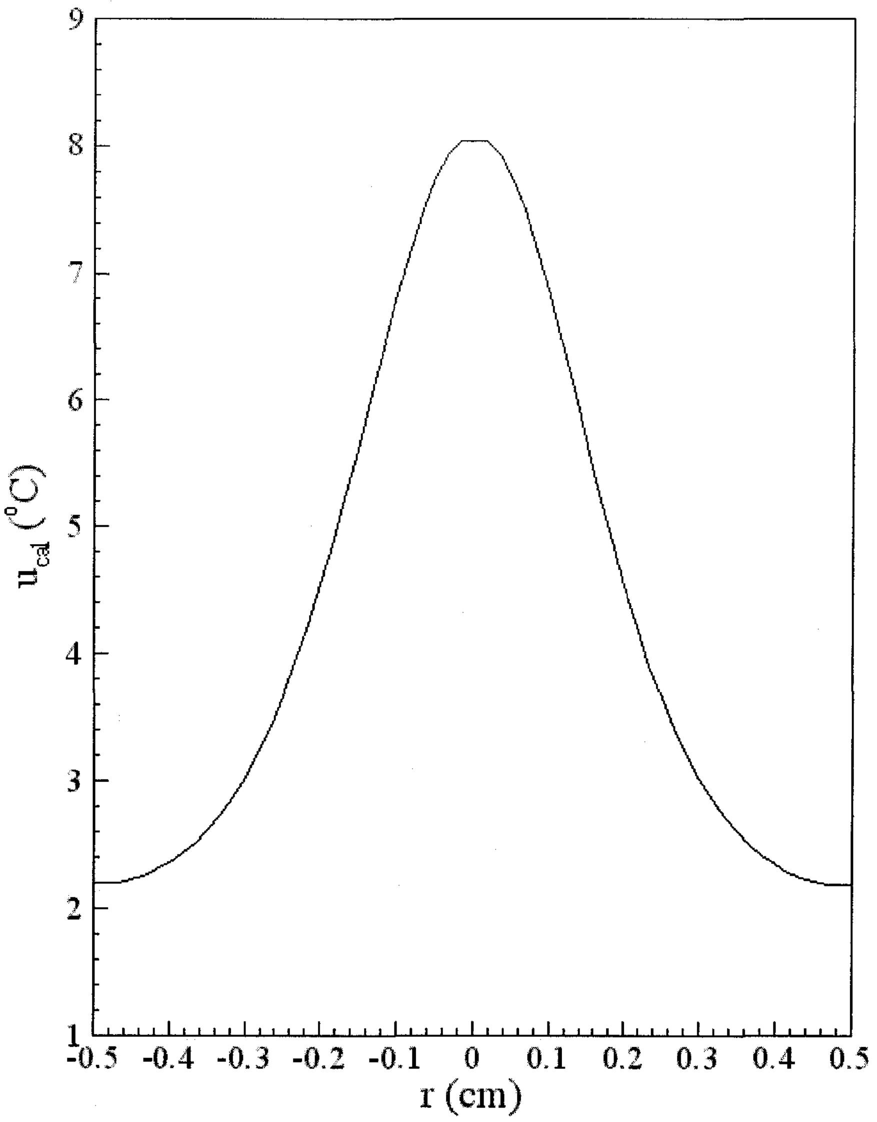

Figure 4.8 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 674$ seconds without any blood vessels

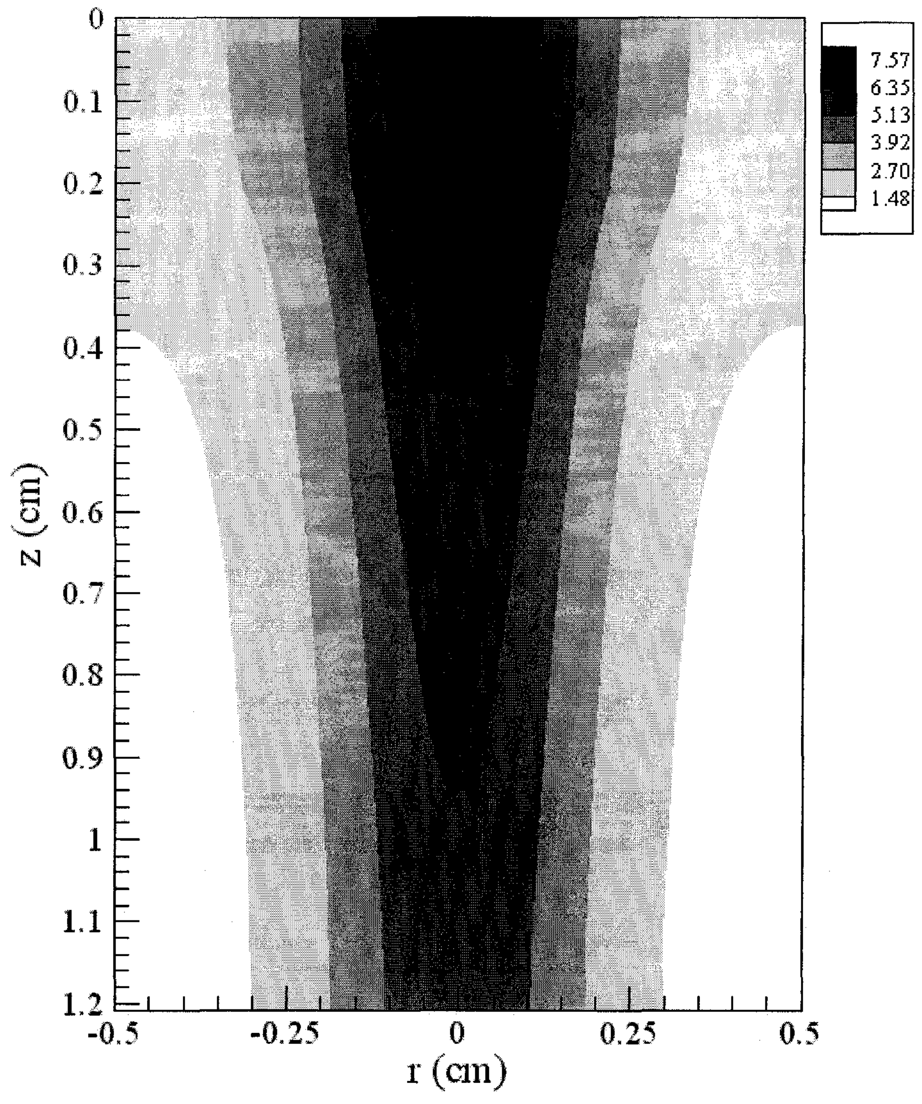

Figure 4.9 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 674$ seconds without any blood vessels

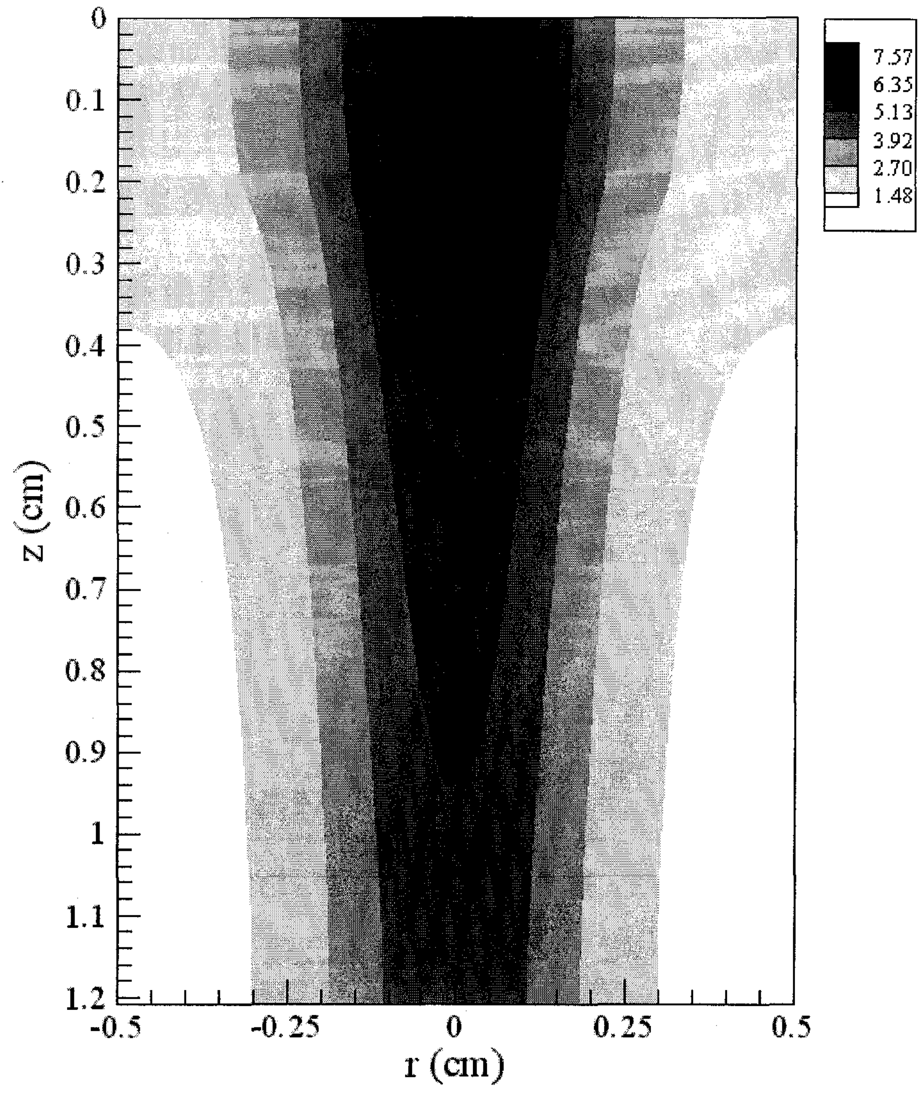

Figure 4.10 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 674$ seconds without any blood vessels

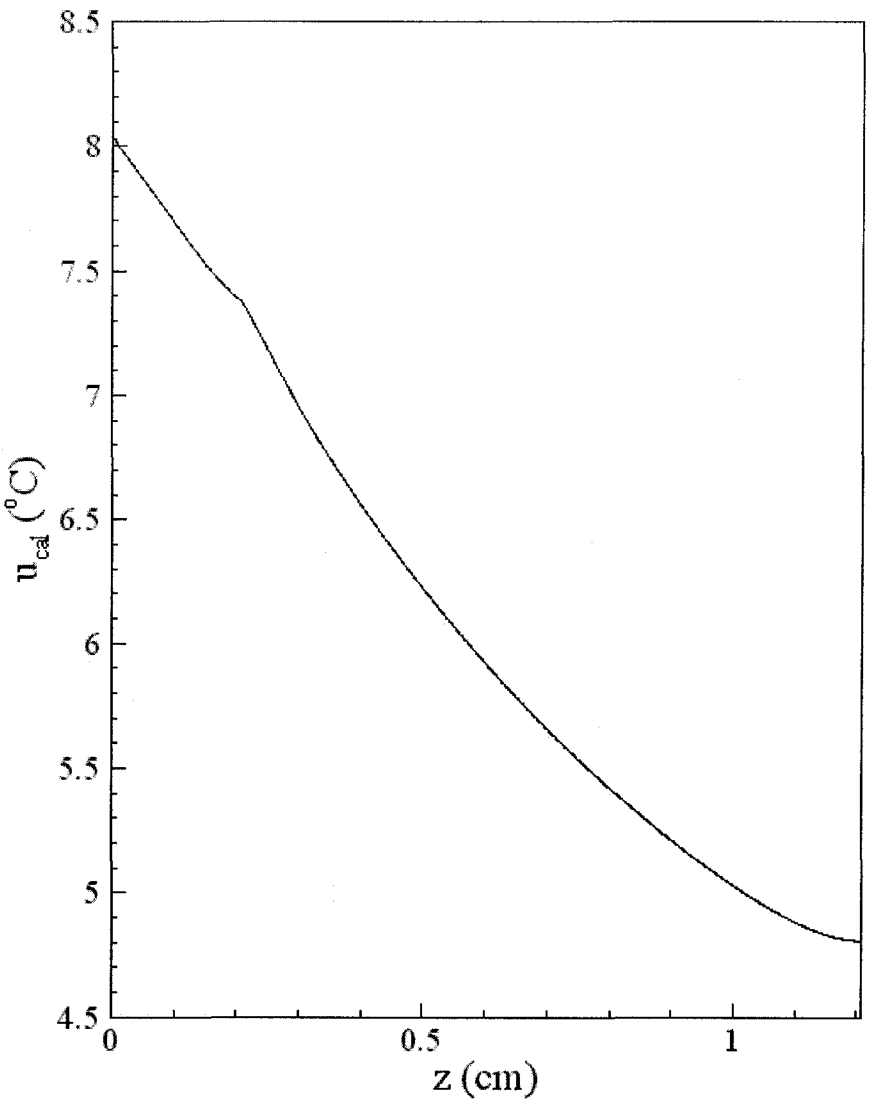

Figure 4.11 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 674$ seconds without any blood vessels

4.2.2 Calculation Case 2

The elevated temperatures at the center and perimeter are $8^{\circ}C$ and $3^{\circ}C$, 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 1.30985. 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.

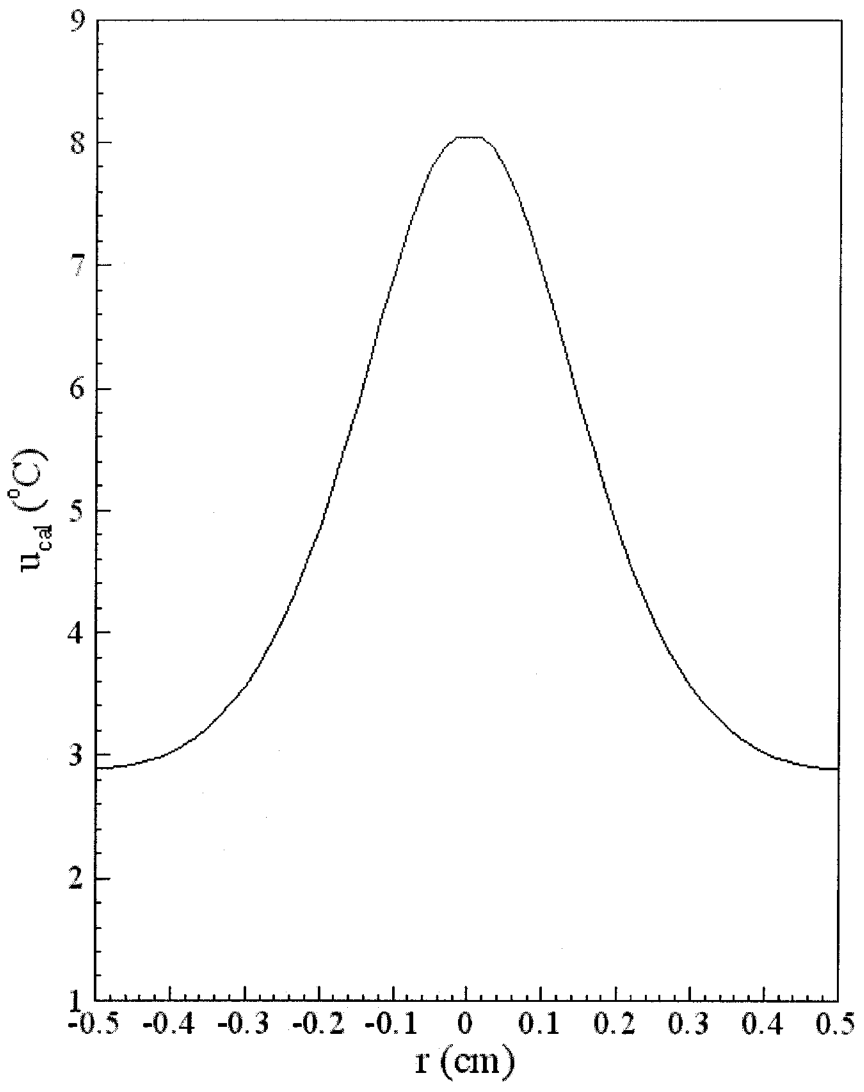

Figure 4.12 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 1062$ seconds without any blood vessels

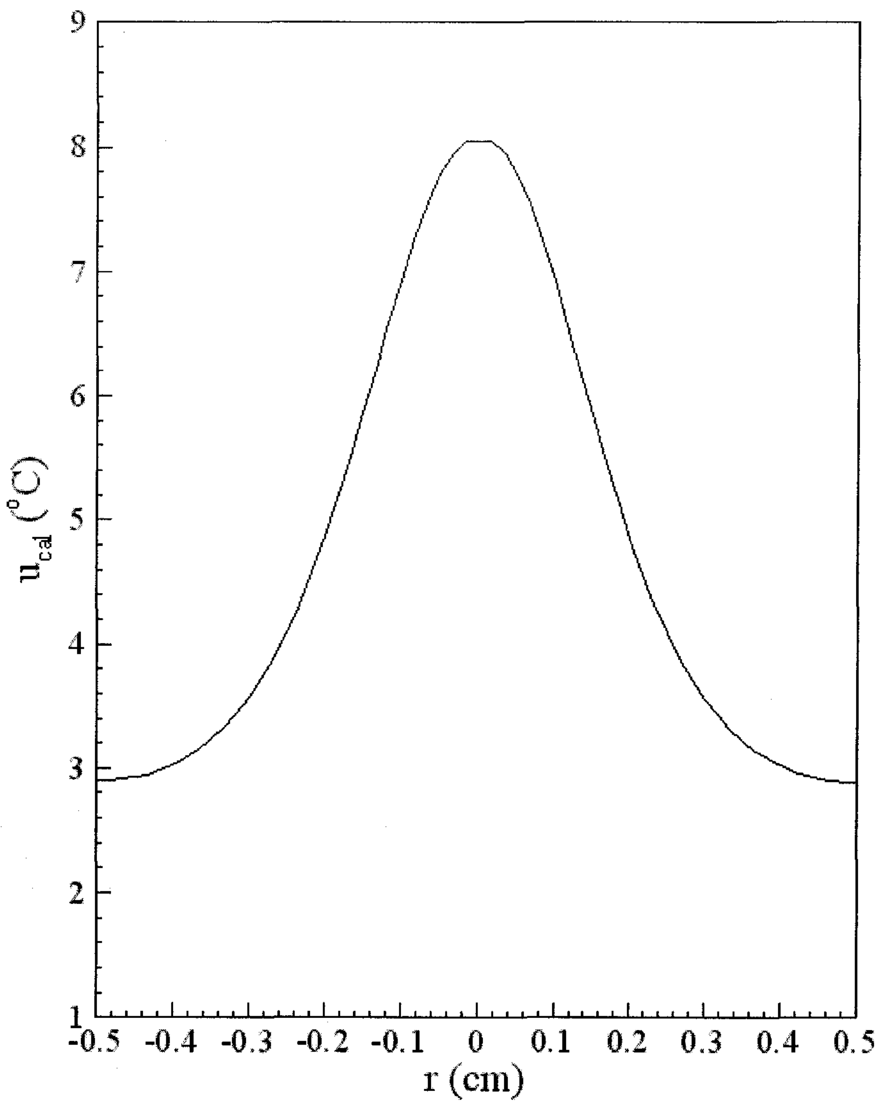

Figure 4.13 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1062$ seconds without any blood vessels

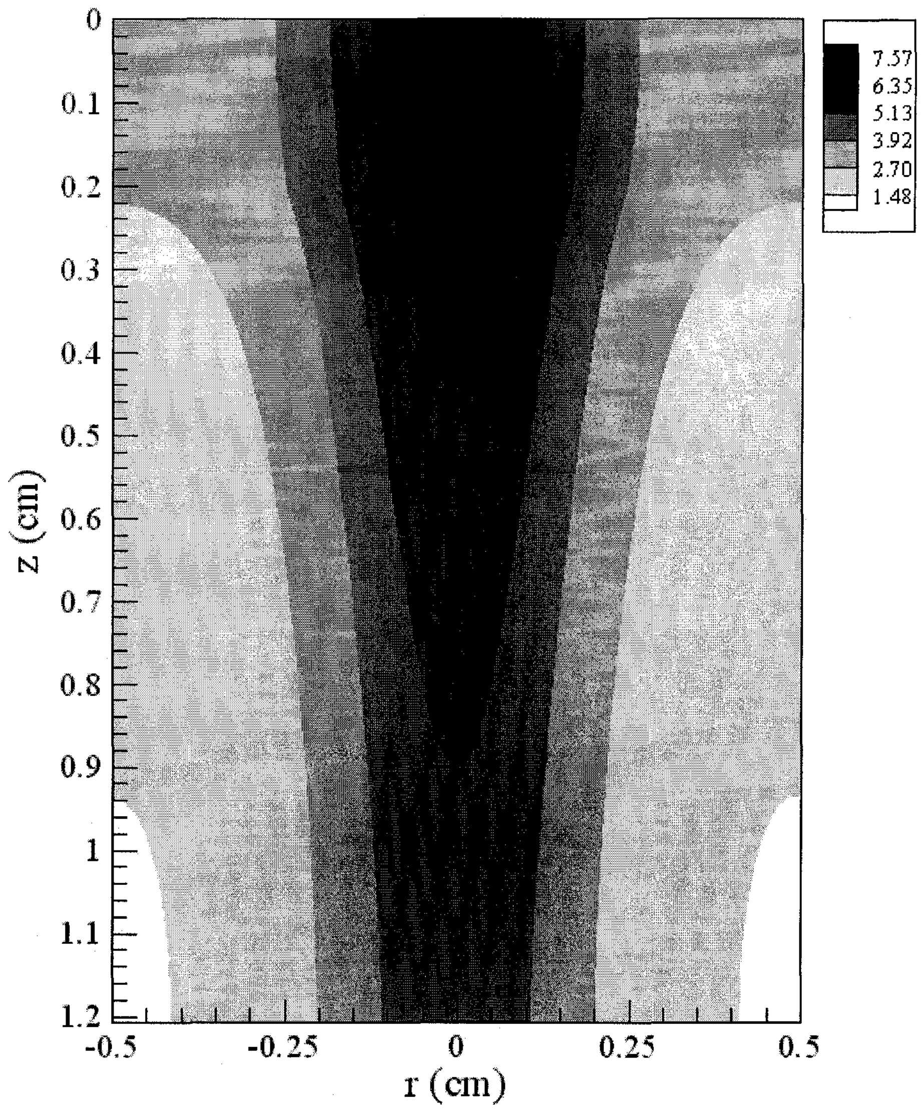

Figure 4.14 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 1062$ seconds without any blood vessels

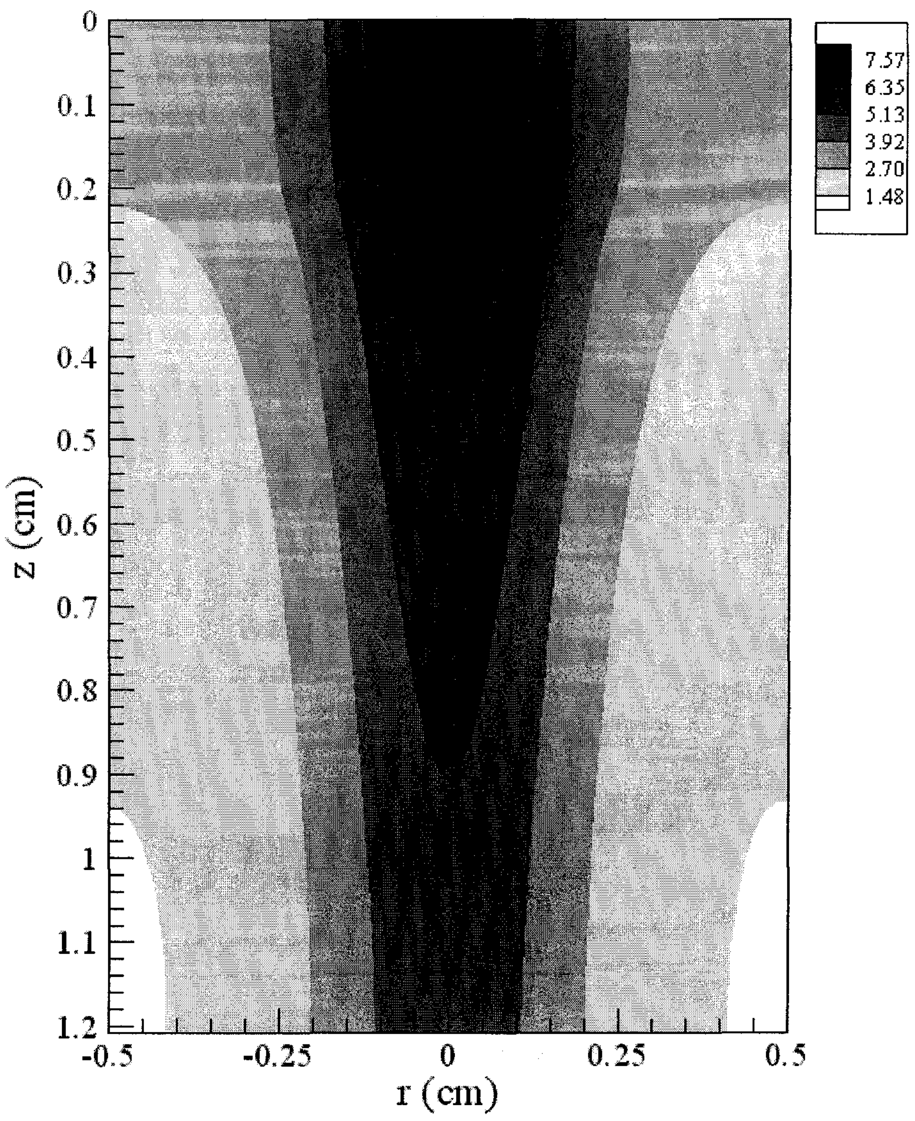

Figure 4.15 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1062$ seconds without any blood vessels

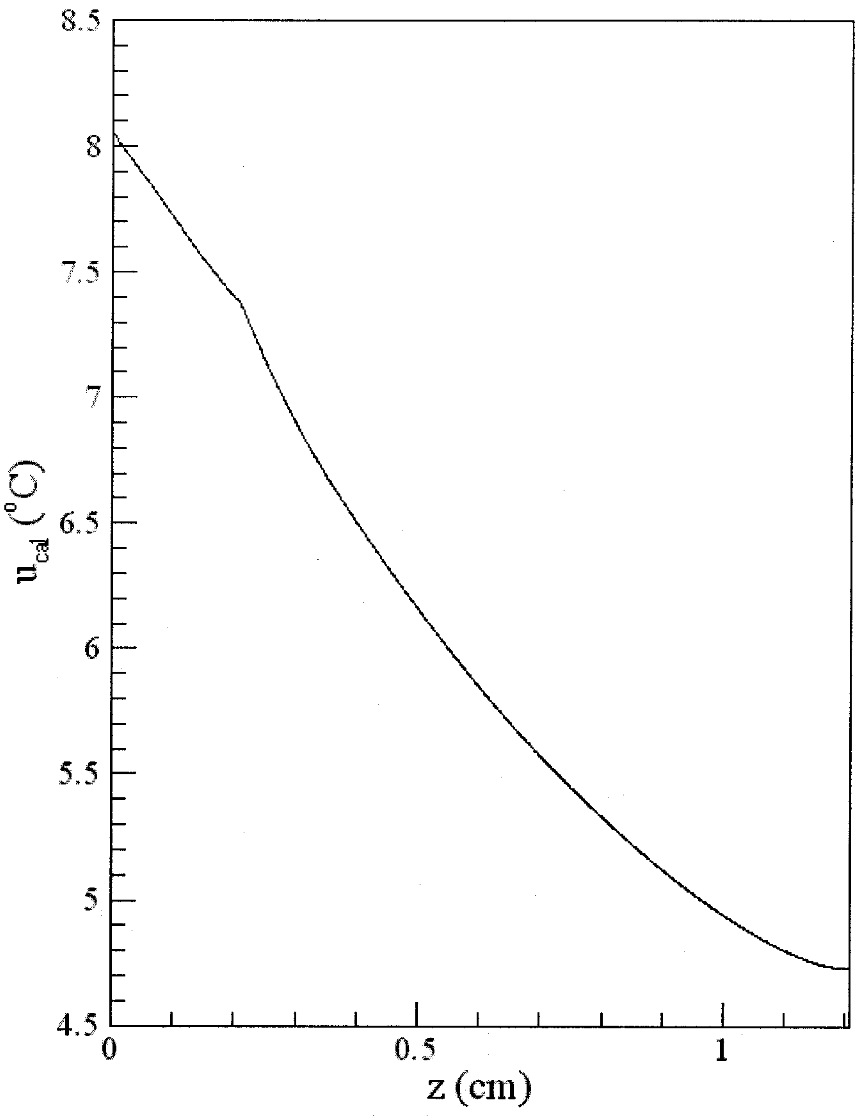

Figure 4.16 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1062$ seconds without any blood vessels

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.

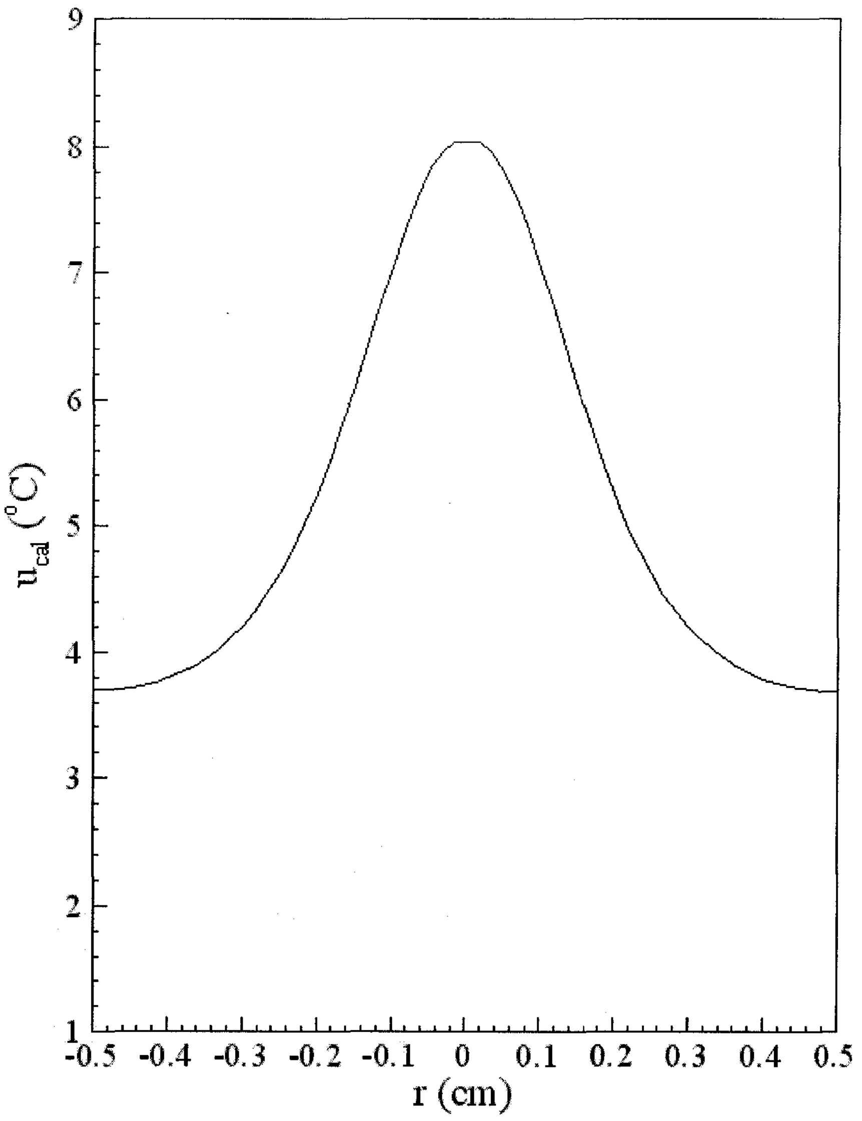

Figure 4.17 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 1536$ seconds without any blood vessels

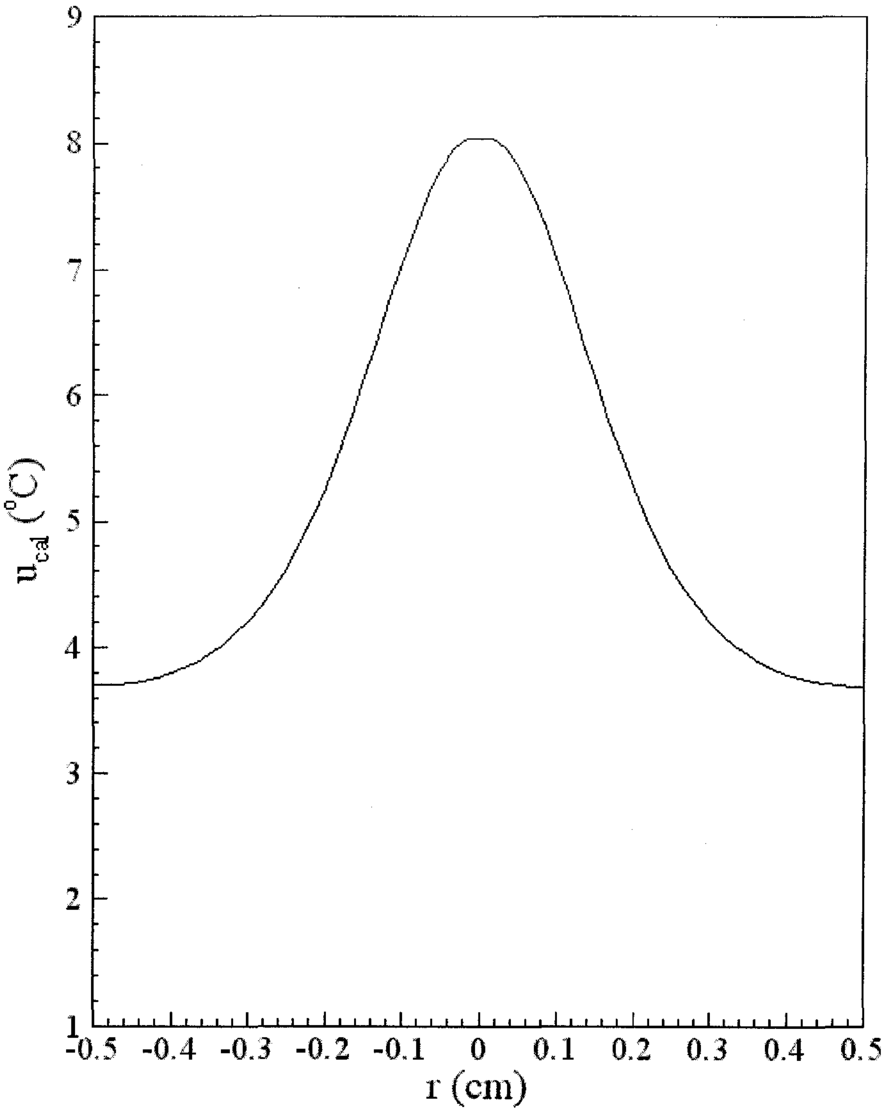

Figure 4.18 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1536$ seconds without any blood vessels

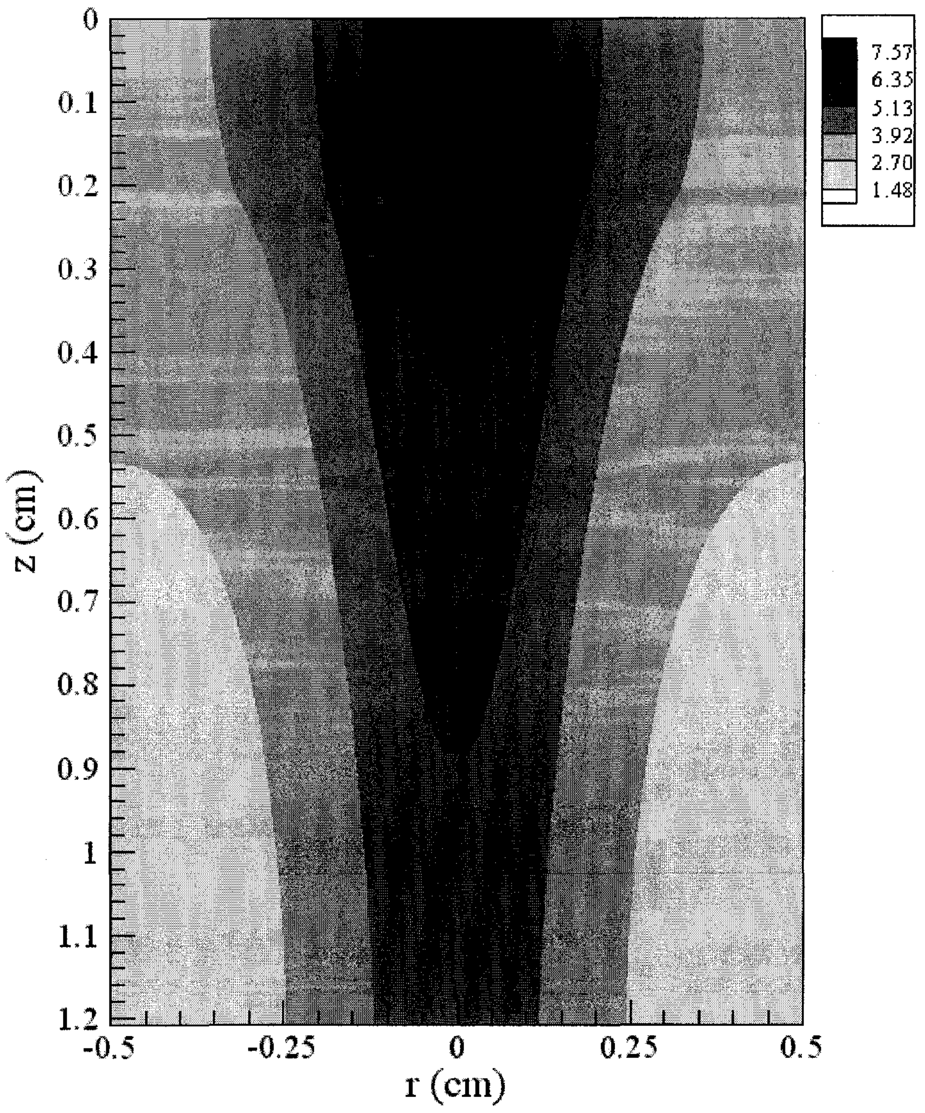

Figure 4.19 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 1536$ seconds without any blood vessels

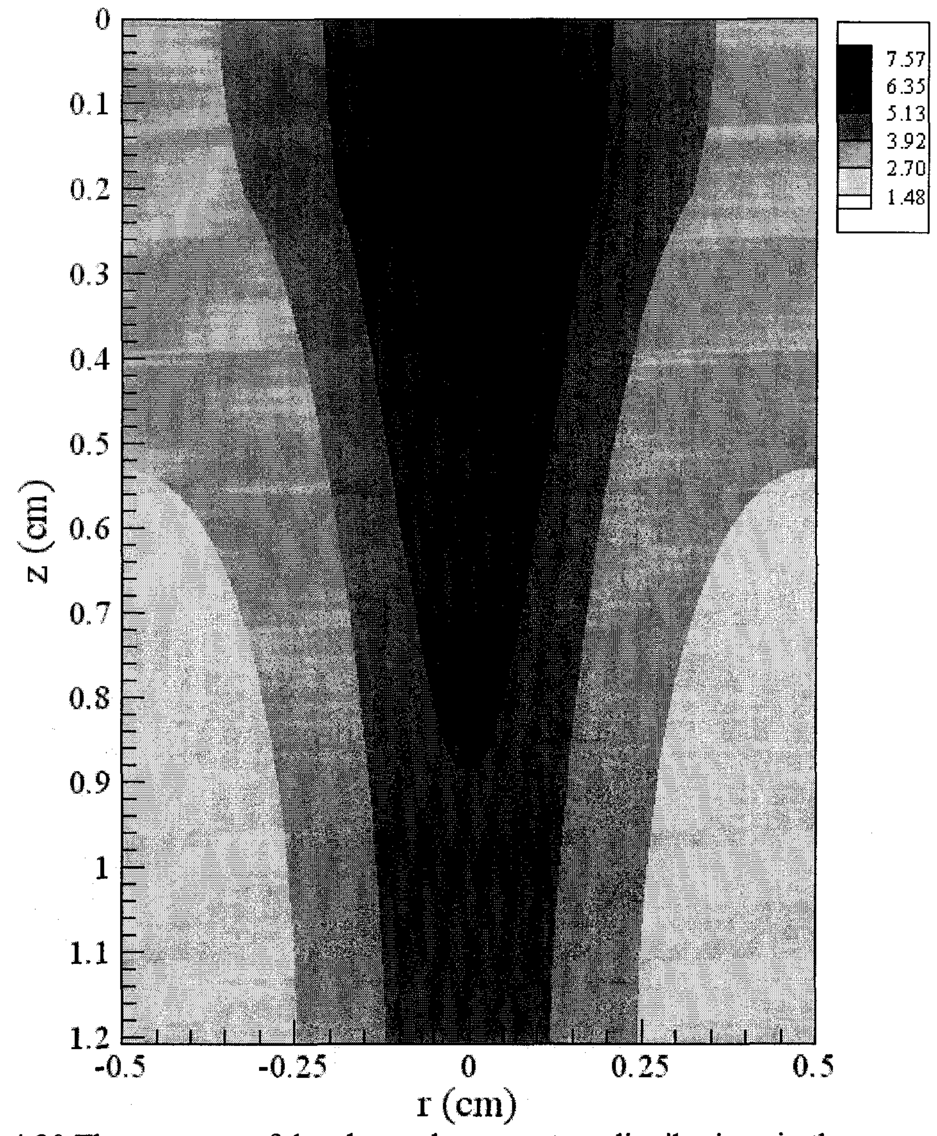

Figure 4.20 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1536$ seconds without any blood vessels

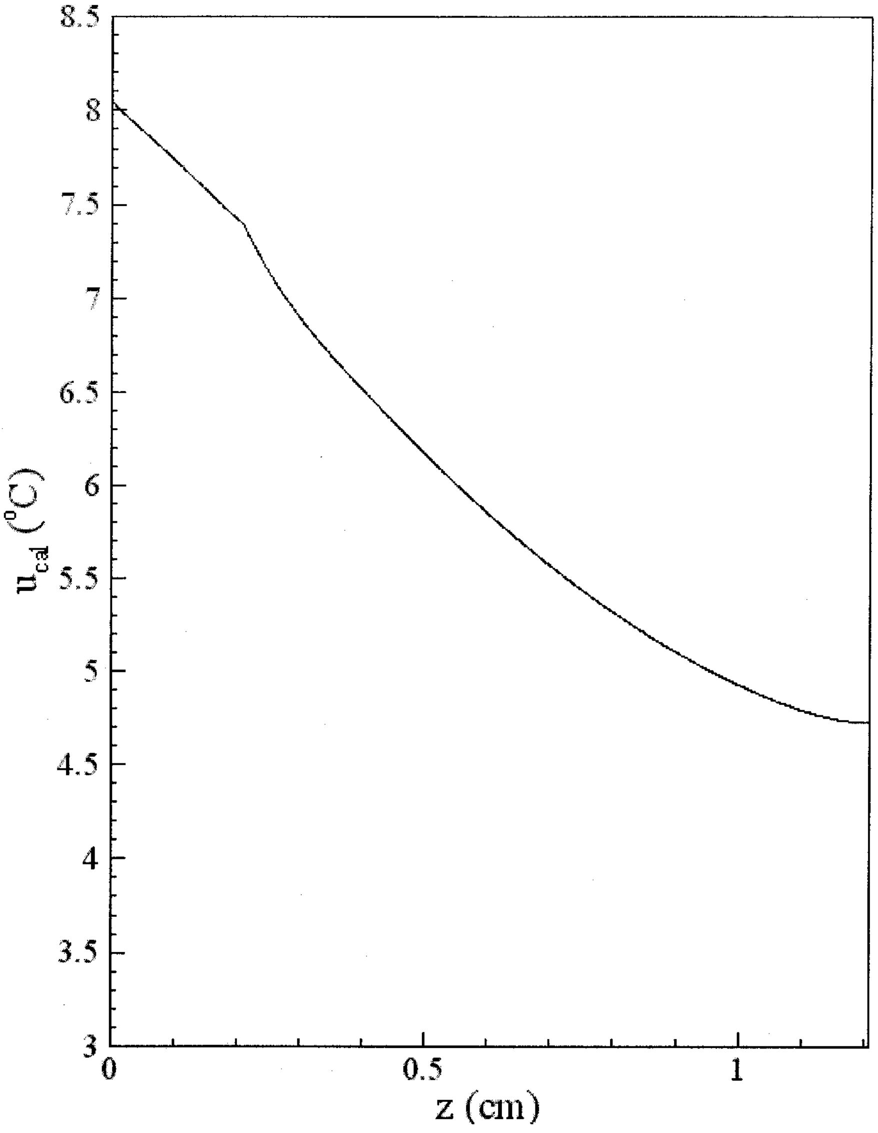

Figure 4.21 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1536$ seconds without any blood vessels

4.3 Grid Independent

Finally, to test our algorithm to be independent of the grid size, there are three additional meshes (r, φ, 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.

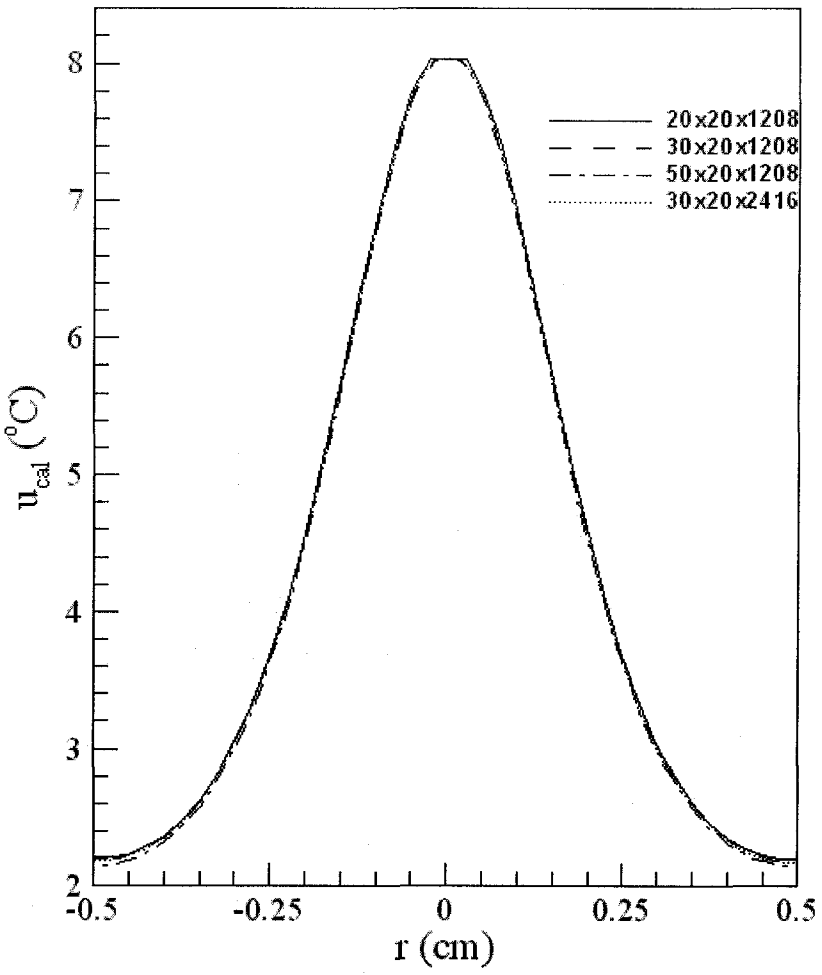

Figure 4.22 Elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 674$ without any blood vessels

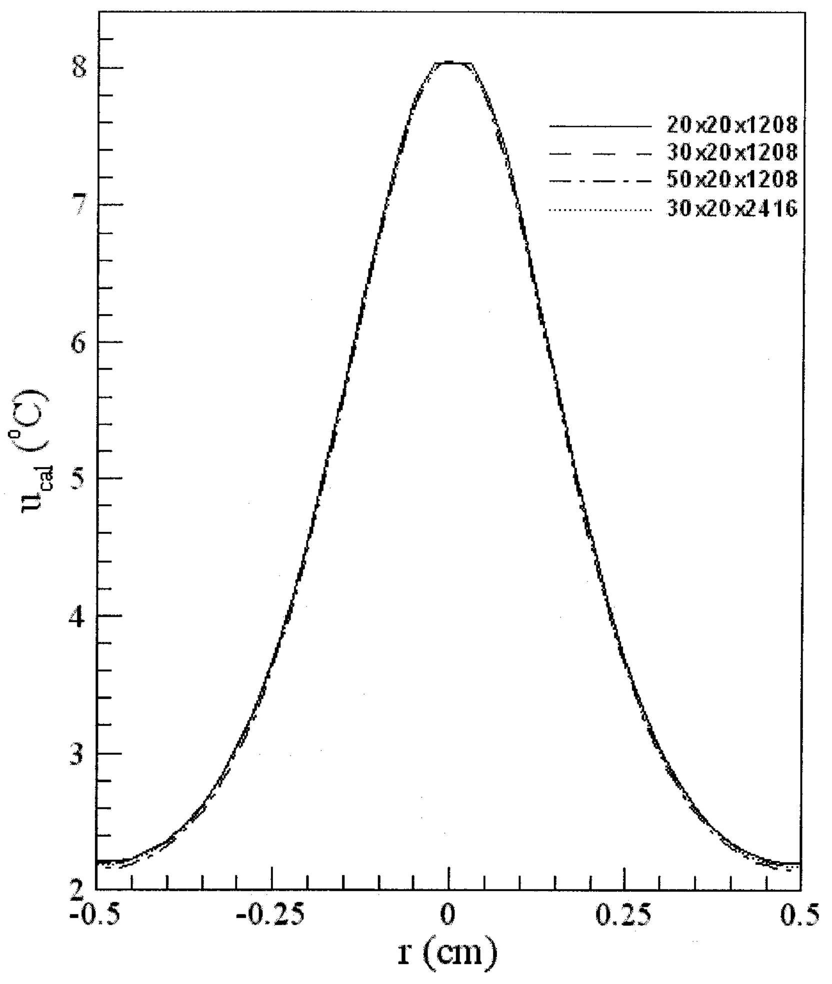

Figure 4.23 Elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at t = 674 without any blood vessels

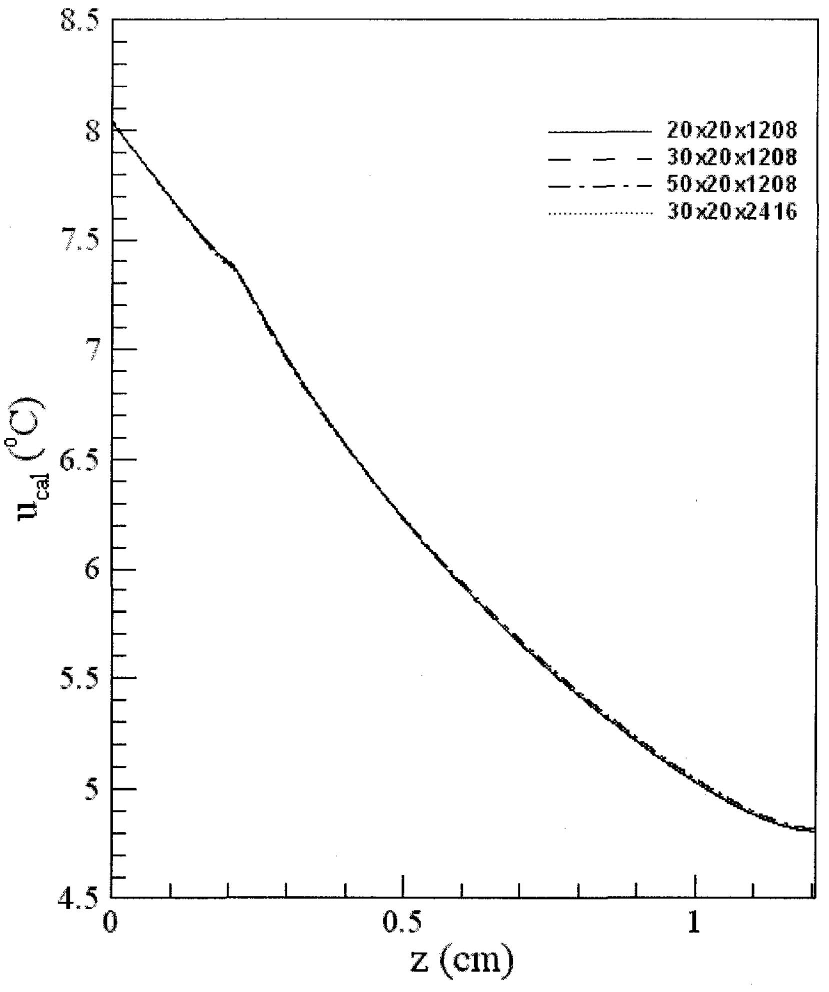

Figure 4.24 Elevated temperature profiles along the depth (the z-direction) at the center of the skin surface, at t = 674 without any blood vessels

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.

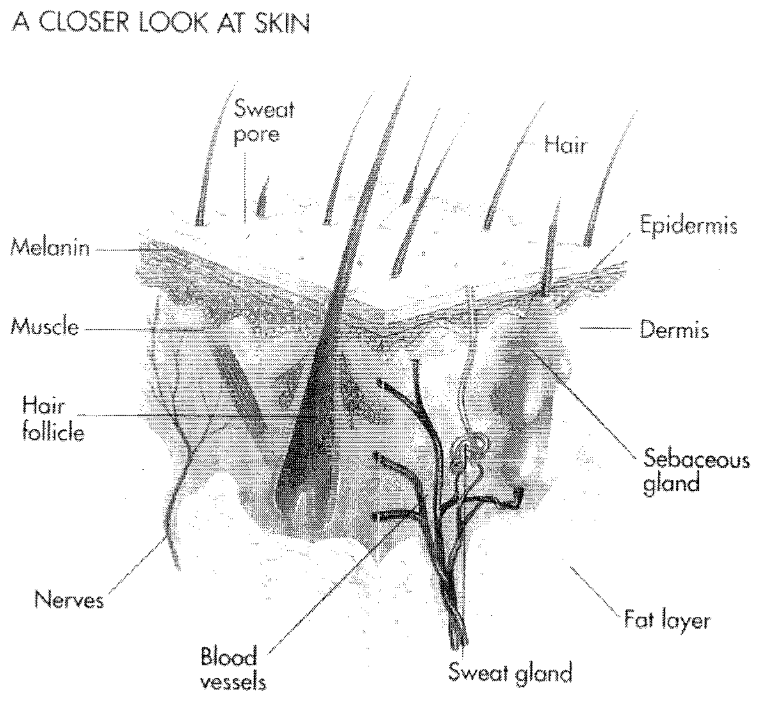

Figure 5.1 Skin structure [Lorimer 1999]

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].

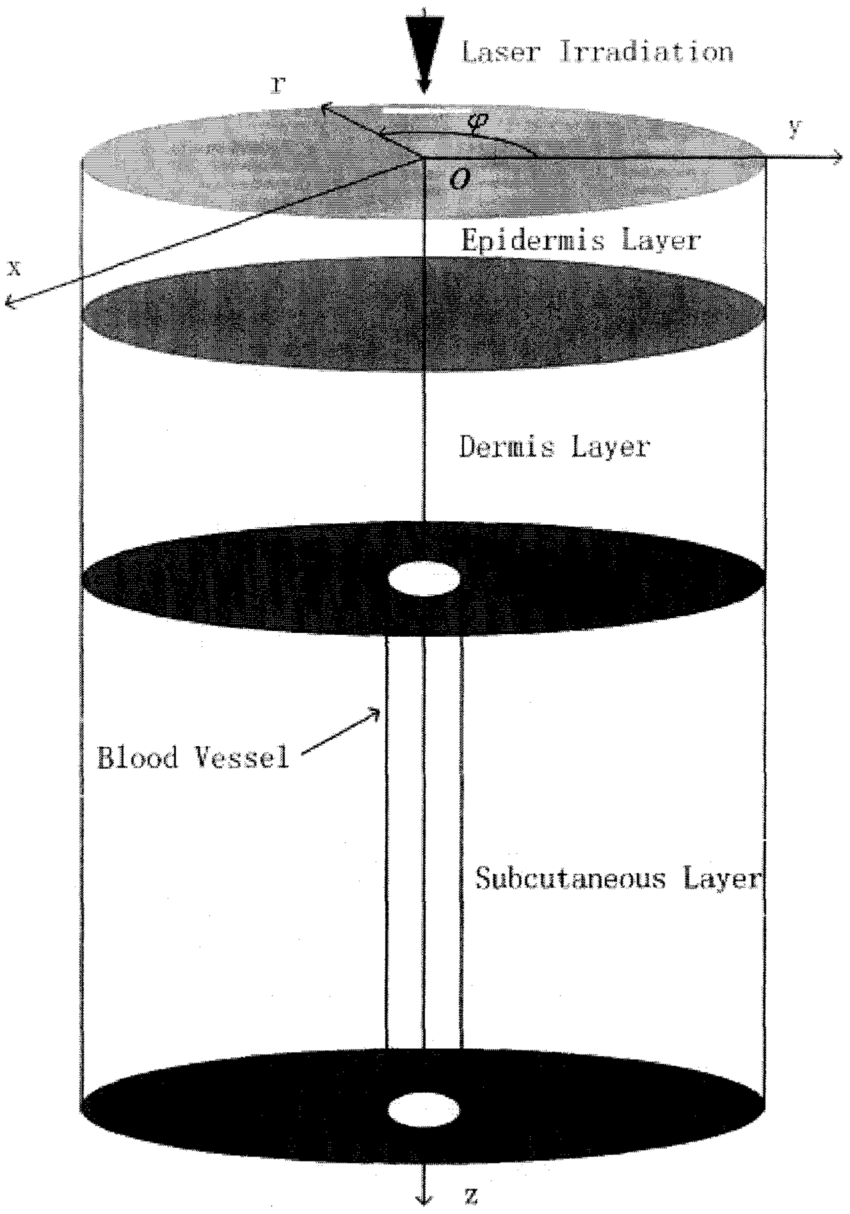

Figure 5.2 Configuration of a 3D skin structure

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]:

$$C_B vF\frac{\partial(\theta_b)}{\partial z} - \alpha P(\theta_w - \theta_b) = 0, \tag{5.1}$$

where C_B is the specific heat of blood, v the velocity of the blood, F the vessel lateral section, and α , P the heat transfer coefficients between blood and tissue, and vessel perimeter respectively. Further, θ_w is the vessel periphery elevated temperature and θ_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]:

$$\rho_l C_l \frac{\partial\theta_l}{\partial t} + W_b^l C_b^l(\theta_l - \theta_b) - k_l[\frac{1}{r}\frac{\partial}{\partial r}(r\frac{\partial\theta_l}{\partial r}) + \frac{1}{r^2}\frac{\partial^2\theta_l}{\partial\varphi^2} + \frac{\partial^2\theta_l}{\partial z^2}] = Q_r^l, \qquad l = 1,2,3 \tag{5.2}$$

where θ_l is the elevated tissue due to heating by a laser; ρ_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.

$$Q_1 = \alpha_1 e^{-\alpha_1 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r\cos\varphi - x_0)^2 + (r\sin\varphi - y_0)^2}{2\sigma^2}} P_0(1 - \mathrm{Reff}_1) f(t),$$
$$Q_2 = \alpha_2 e^{-\alpha_1 L_1 - \alpha_2 Z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r\cos\varphi - x_0)^2 + (r\sin\varphi - y_0)^2}{2\sigma^2}} P_0(1 - \mathrm{Reff}_2) f(t), \qquad (5.3)$$
$$Q_3 = \alpha_3 e^{-\alpha_1 L_1 - \alpha_2 L_2 - \alpha_3 z} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r\cos\varphi - x_0)^2 + (r\sin\varphi - y_0)^2}{2\sigma^2}} P_0(1 - \mathrm{Reff}_3) f(t),$$

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

$$\frac{\partial \theta_1}{\partial z} = 0, \quad z = 0, \qquad (5.4)$$

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

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

$$\frac{\partial \theta_3}{\partial z} = 0, \quad z = L_1 + L_2 + L_3. \qquad (5.7)$$

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

$$\theta_2 = \theta_b, \quad z = L_1 + L_2, \qquad (5.8)$$

$$\theta_3 = \theta_b, \quad z = L_1 + L_2 + L_3, \qquad (5.9)$$

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

$$\frac{\partial \theta_b}{\partial r} = Bi(\theta_w - \theta_b). \qquad (5.10)$$

The boundary conditions at the lateral wall of the tissue are

$$\frac{\partial \theta_l}{\partial r} = 0, \quad r = R, \tag{5.11}$$

and

$$\theta_l(r,\varphi,z) = \theta_l(r,\varphi + 2m\pi, z), \tag{5.12}$$

where m is an integer.

The initial condition is

$$\theta_l = 0, \ t = 0, \quad l = 1,2,3. \tag{5.13}$$

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

To obtain a temperature distribution numerically, we first let $(u_l)_{ijk}^n$ and u_b be the numerical approximation of $(\theta_l)(i\Delta r, j\Delta\varphi, k\Delta z, n\Delta t)$ and θ_b, and $\Delta r, \Delta\varphi, \Delta z$, and Δt be the spatial and temporal mesh sizes, respectively. Here, i, j, k are chosen to be $1 \le i \le N_r$, $1 \le j \le N_\varphi$, $1 \le k \le N_l^z$, so that $N_r \cdot \Delta r = R$, $N_\varphi \cdot \Delta\varphi = 2\pi$ and $N_l^z \cdot \Delta z = L_l$, $l = 1,2,3$. We also let R_{bv} 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:

$$\begin{aligned}&\rho_l C_l \frac{(u_l)_{ijk}^{n+1} - (u_l)_{ijk}^n}{\Delta t} + \frac{W_b^l C_b^l}{2}[(u_l)_{ijk}^{n+1} + (u_l)_{ijk}^n - 2(u_b)_k] \\ &- k_l(P_r^2 + \delta_\varphi^2 + \delta_z^2)\frac{(u_l)_{ijk}^{n+1} + (u_l)_{ijk}^n}{2} = (Q_r^l)_{ijk}^{n+0.5}, \quad l = 1,2,3.\end{aligned} \tag{5.14}$$

where $P_r^2 u_{ijk} = \dfrac{r_{i+\frac{1}{2}}(u_{i+1jk} - u_{ijk}) - r_{i-\frac{1}{2}}(u_{ijk} - u_{i-1jk})}{r_i \Delta r^2}$, $\delta_\varphi^2 u_{ijk} = \dfrac{u_{ij+1k} - 2u_{ijk} + u_{ij-1k}}{r_i^2 \Delta\varphi^2}$, $\delta_z^2 u_{ijk}^n = \dfrac{u_{ijk+1} - 2u_{ijk} + u_{ijk-1}}{\Delta z^2}$,

and $r_{i+\frac{1}{2}} = (i + \dfrac{1}{2})\Delta r$. The discrete interfacial equations are assumed to be, for any time level,

$$k_1 \frac{(u_1)_{ijN_1^z}^n - (u_1)_{i,j,N_1^z-1}^n}{\Delta z} = k_2 \frac{(u_2)_{ij1}^n - (u_2)_{ij0}^n}{\Delta z}, \quad (u_1)_{ijN_1^z}^n = (u_2)_{ij0}^n, \tag{5.15}$$

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

$$k_2 \frac{(u_2)_{ijN_1^z}^n - (u_2)_{i,j,N_1^z-1}^n}{\Delta z} = k_3 \frac{(u_3)_{ij1}^n - (u_3)_{ij0}^n}{\Delta z}, \quad (u_2)_{ijN_1^z}^n = (u_3)_{ij0}^n, \tag{5.16}$$

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

$$(u_2)_{ijN_2^z}^n = (u_b)_{N_2^z}. \tag{5.17}$$

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

$$(u_l)_{ijk}^{n+1} = ((u_l)_{i+1jk}^{n+1} + Bi \cdot \Delta r \cdot (u_l)_{i-1jk}^{n+1})/(1 + Bi \cdot \Delta r), \tag{5.18}$$

where $i = R_{bv}$.

The initial and other boundary conditions are discretized as follows:

$$(u_l)_{ijk}^0 = 0, \tag{5.19}$$

$$(u_1)_{ij0}^n = (u_1)_{ij1}^n, \tag{5.20}$$

$$if \;\; i > R_{bv}, \;\; (u_3)_{ijN_3^z}^n = (u_3)_{ijN_3^z-1}^n, \tag{5.21}$$

$$if \;\; i \le R_{bv}, \;\; (u_3)_{ijN_3^z}^n = u_b, \tag{5.22}$$

$$(u_l)_{N_r jk}^n = (u_l)_{N_r-1jk}^n, \tag{5.23}$$

$$(u_l)^n_{ijk} = (u_l)^n_{ij+mN_\varphi\Delta\varphi k}, \tag{5.24}$$

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 θ^{center}_{pre} is satisfied. It should be pointed out that in step 1, we calculate θ_l from Eqs. (5.1)-(5.14) starting at a guessed θ_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^{center}_{cal}) and perimeter ($u^{primeter}_{cal}$) and pre-specified temperature (θ^{center}_{pre}, $\theta^{primeter}_{pre}$), the laser may need to be turned on and off until $S_p(P_0)$ is less than a pre-specified value, $S^{Specified}_p$. 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).

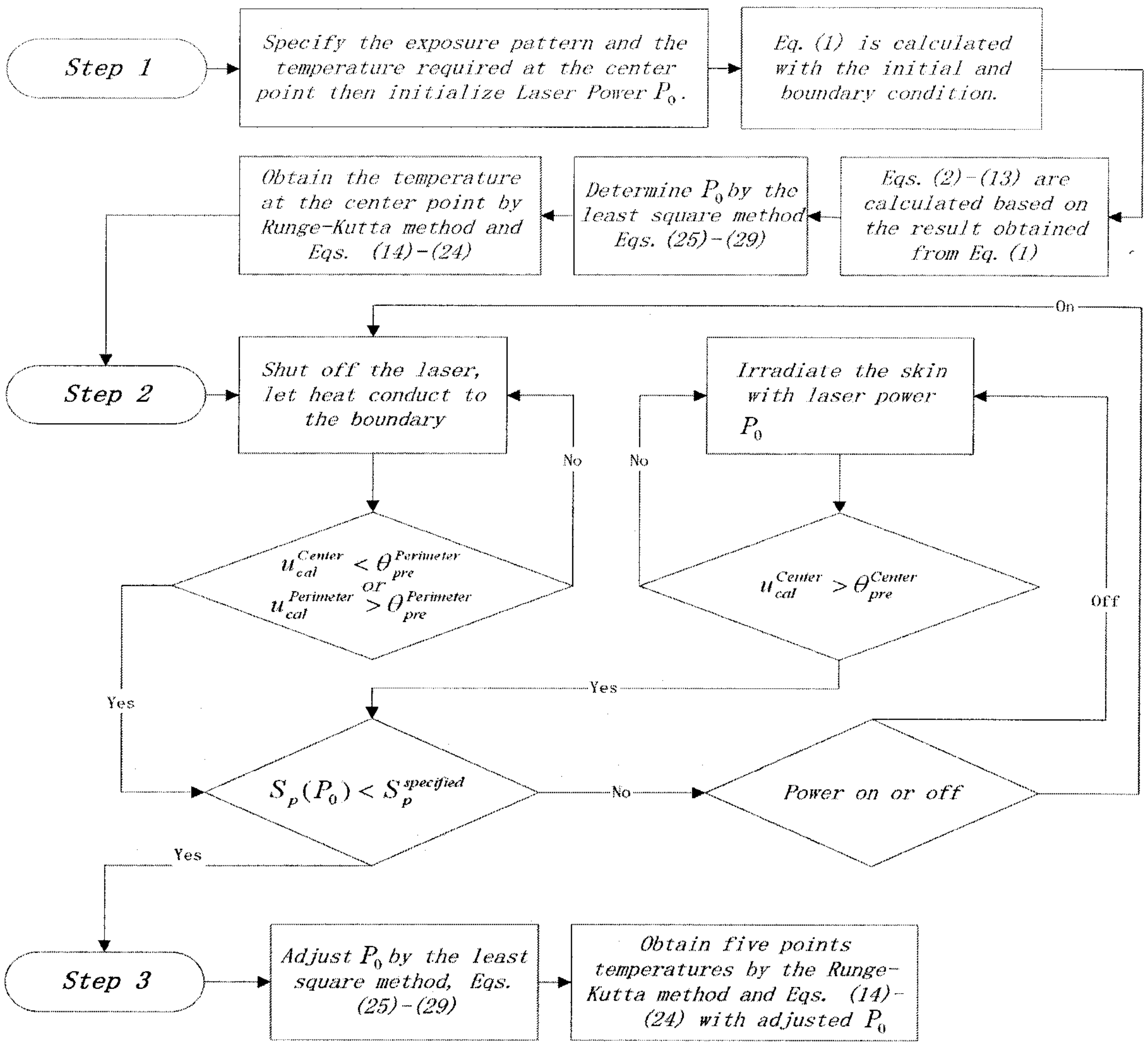
Step 1
Specify the exposure pattern and the temperature required at the center point then initialize Laser Power P_0.
Eq. (1) is calculated with the initial and boundary condition.
Eqs. (2)-(13) are calculated based on the result obtained from Eq. (1)
Determine P_0 by the least square method Eqs. (25)-(29)
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
Irradiate the skin with laser power P_0
$u_{cal}^{Center} < \theta_{pre}^{Perimeter}$ or $u_{cal}^{Perimeter} > \theta_{pre}^{Perimeter}$
$u_{cal}^{Center} > \theta_{pre}^{Center}$
No
No
Yes
Yes
On
Off
$S_p(P_0) < S_p^{specified}$
No
Power on or off
Yes
Step 3
Adjust P_0 by the least square method, Eqs. (25)-(29)
Obtain five points temperatures by the Runge-Kutta method and Eqs. (14)-(24) with adjusted P_0

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,φ,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 $1\,^oC$.

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 Δr 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 $8^\circ C$ and $2^\circ C$, 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 $8^\circ C$. However, the temperature at the perimeter is much lower than the required temperature. A relative error $\sum_{i=1}^{4}\left[\frac{\theta_{pre}^{i} - u_{cal}^{i}}{\theta_{pre}^{i}}\right]^2$, which shows the difference between the pre-specified temperature and the calculated temperature, is 0.505088.

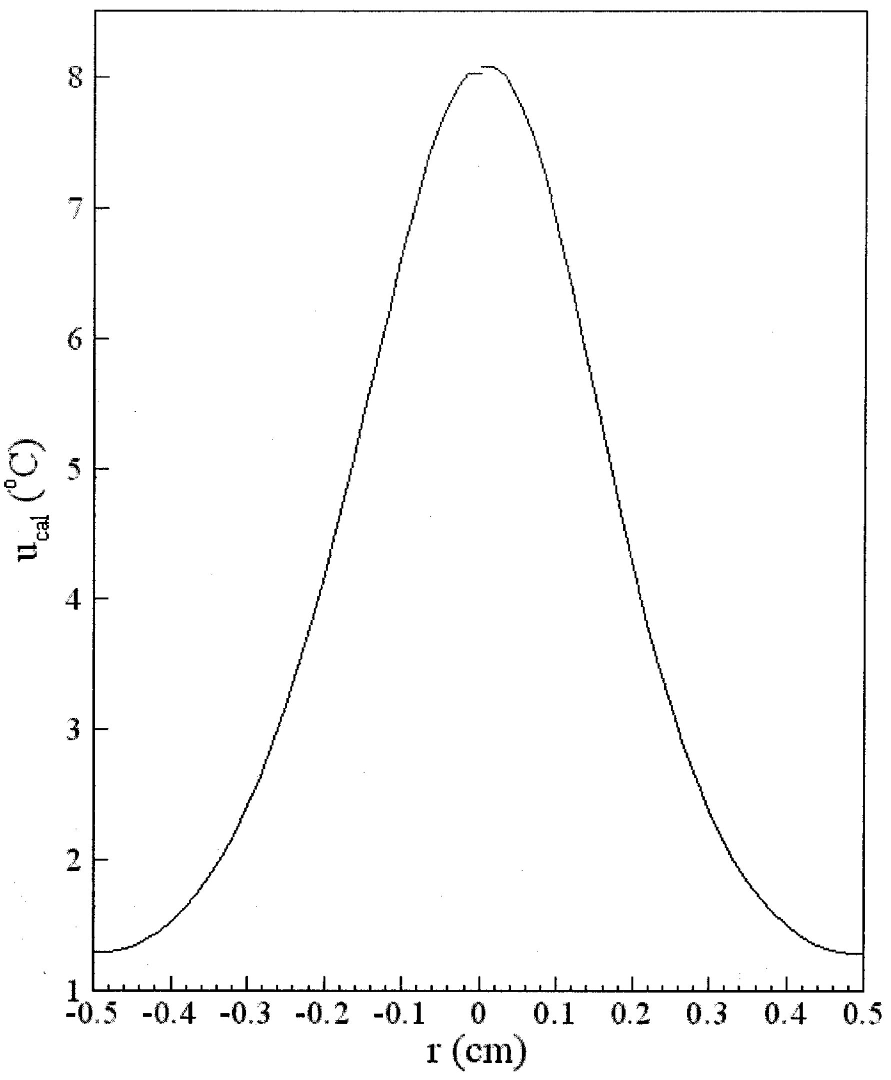

Figure 5.4 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ seconds with a blood vessel

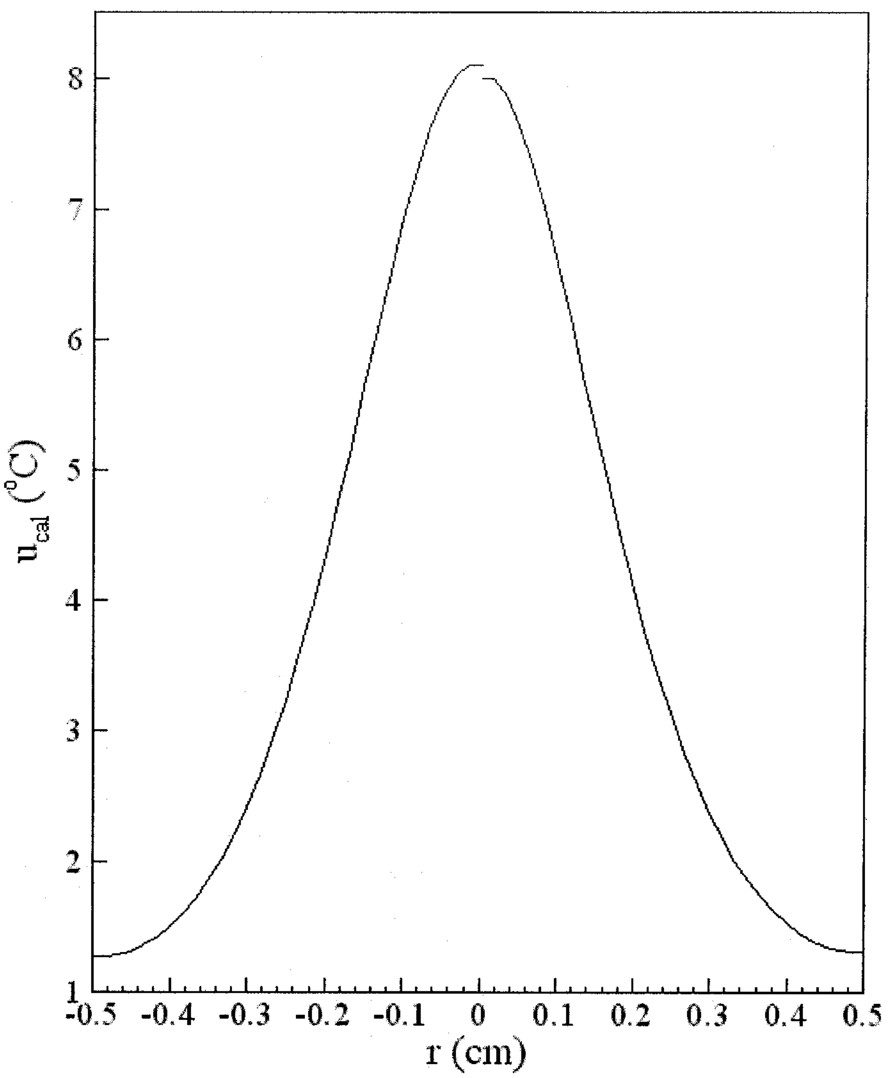

Figure 5.5 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 410$ seconds with a blood vessel

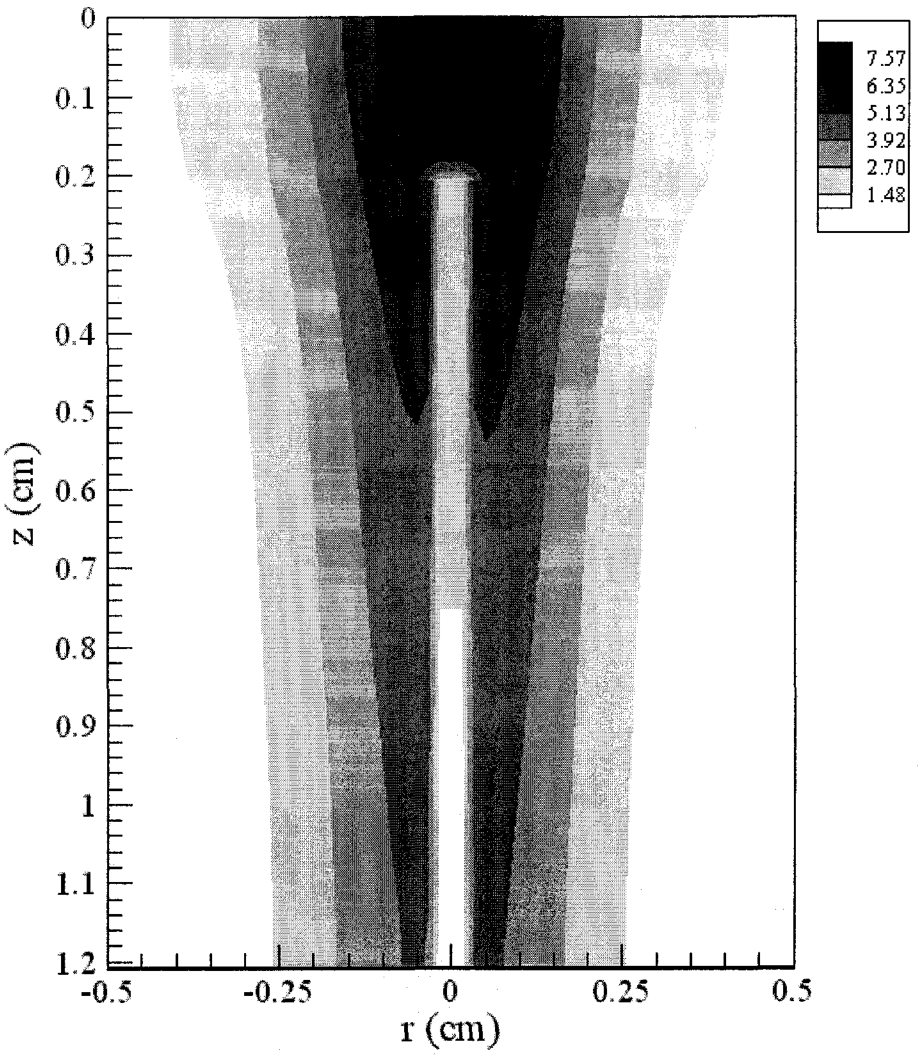

Figure 5.6 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ seconds with a blood vessel

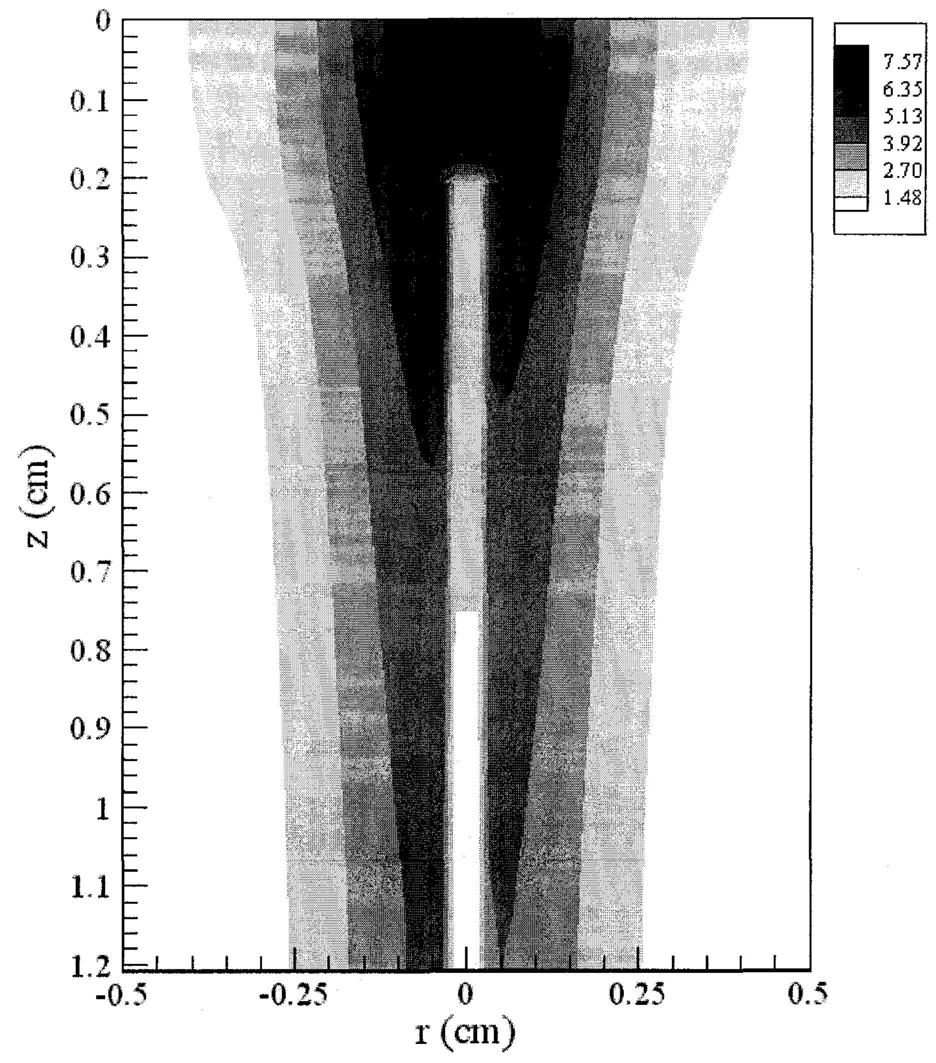

Figure 5.7 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 410$ seconds with a blood vessel

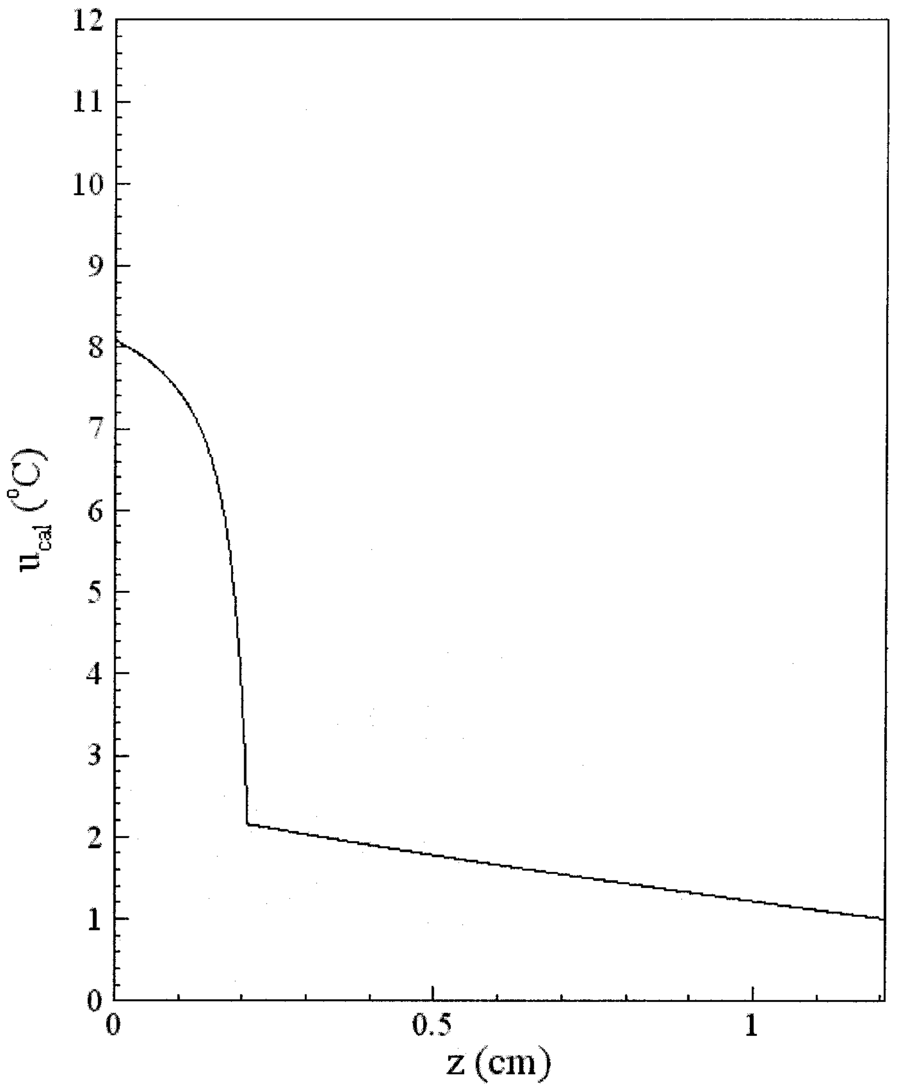

Figure 5.8 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 410$ seconds with a blood vessel

The computation shows that the laser was automatically shut off between $t = 410$ seconds and $t = 529$ seconds, and then on with a modified $P_0 = 17.2661$ (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 $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 5.11 and 5.12 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 5.13 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.

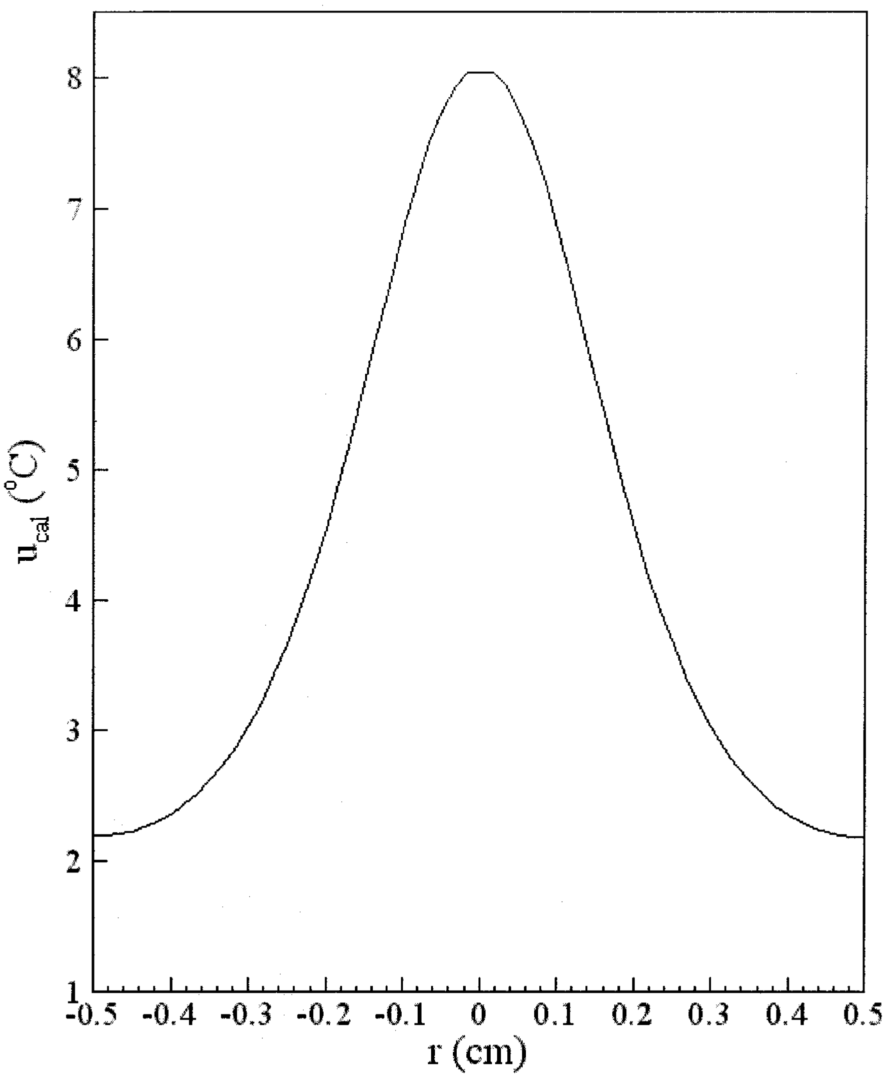

Figure 5.9 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 672$ seconds with a blood vessel

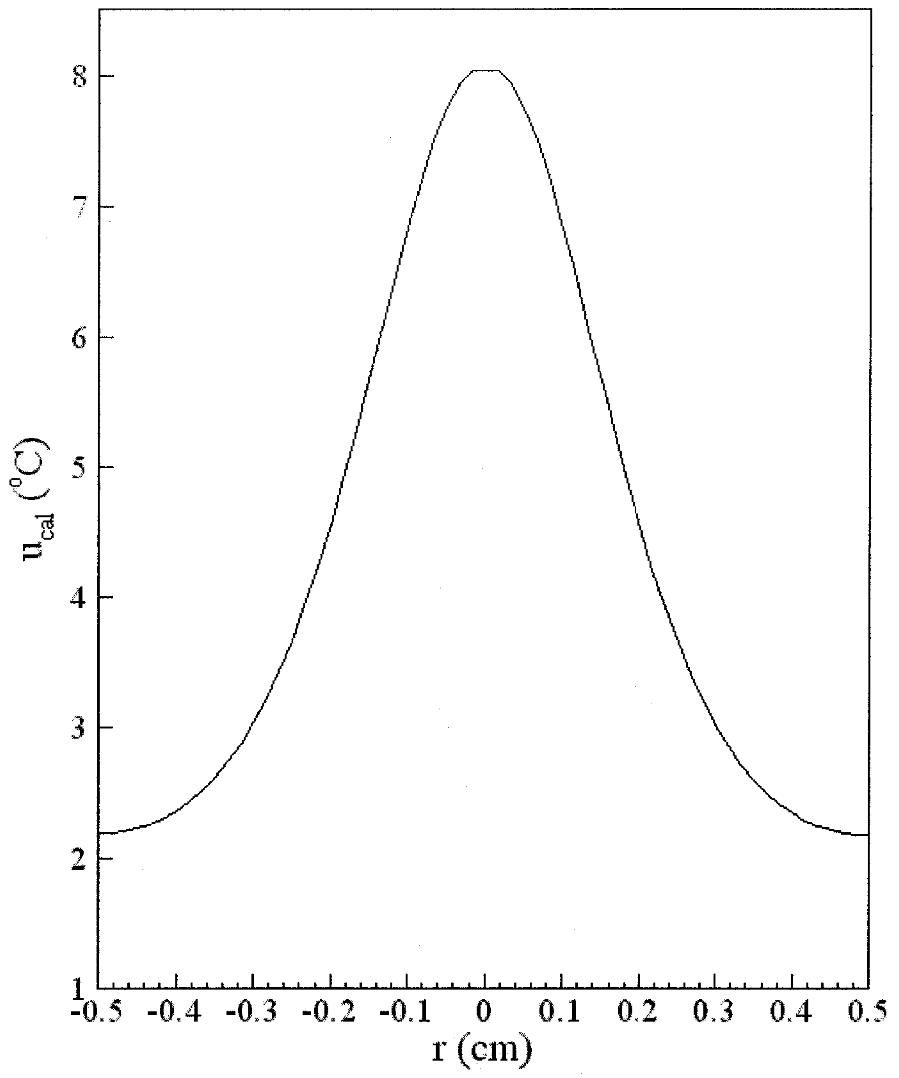

Figure 5.10 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at t = 672 seconds with a blood vessel

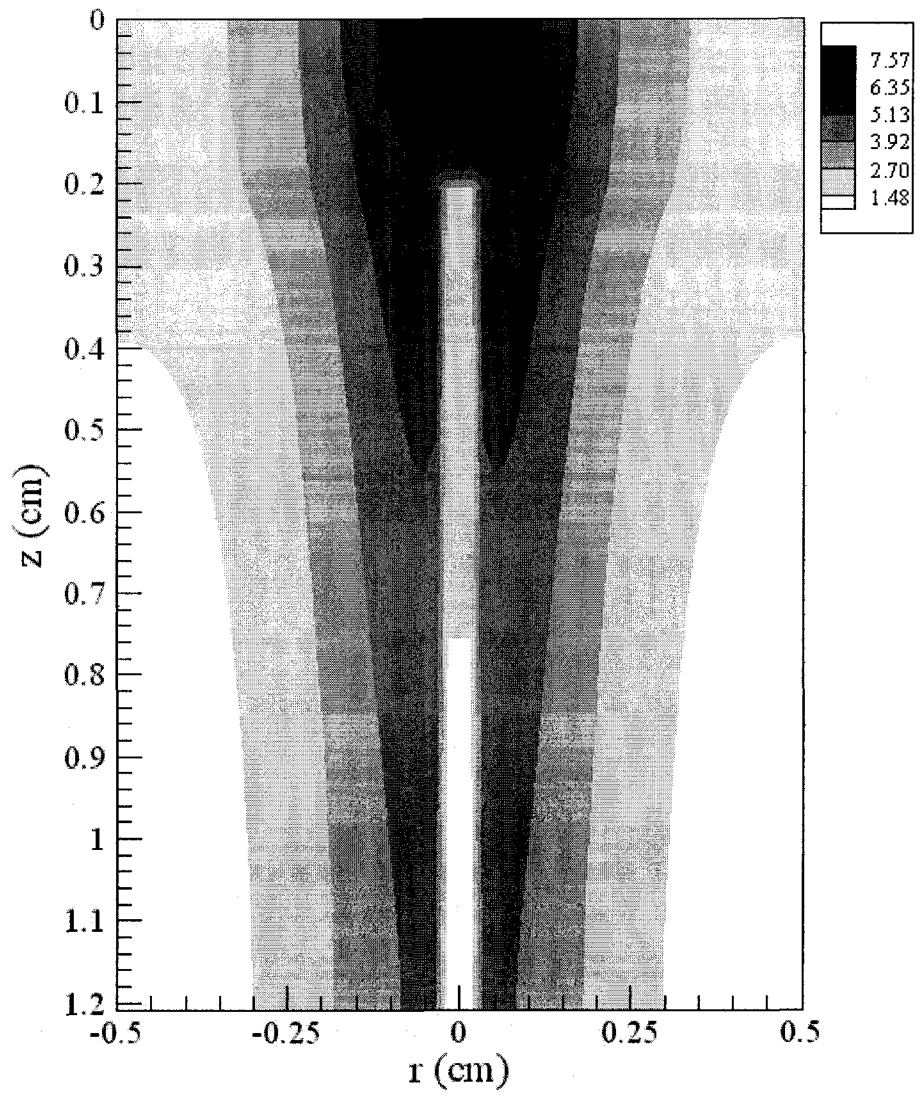

Figure 5.11 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 672$ seconds with a blood vessel

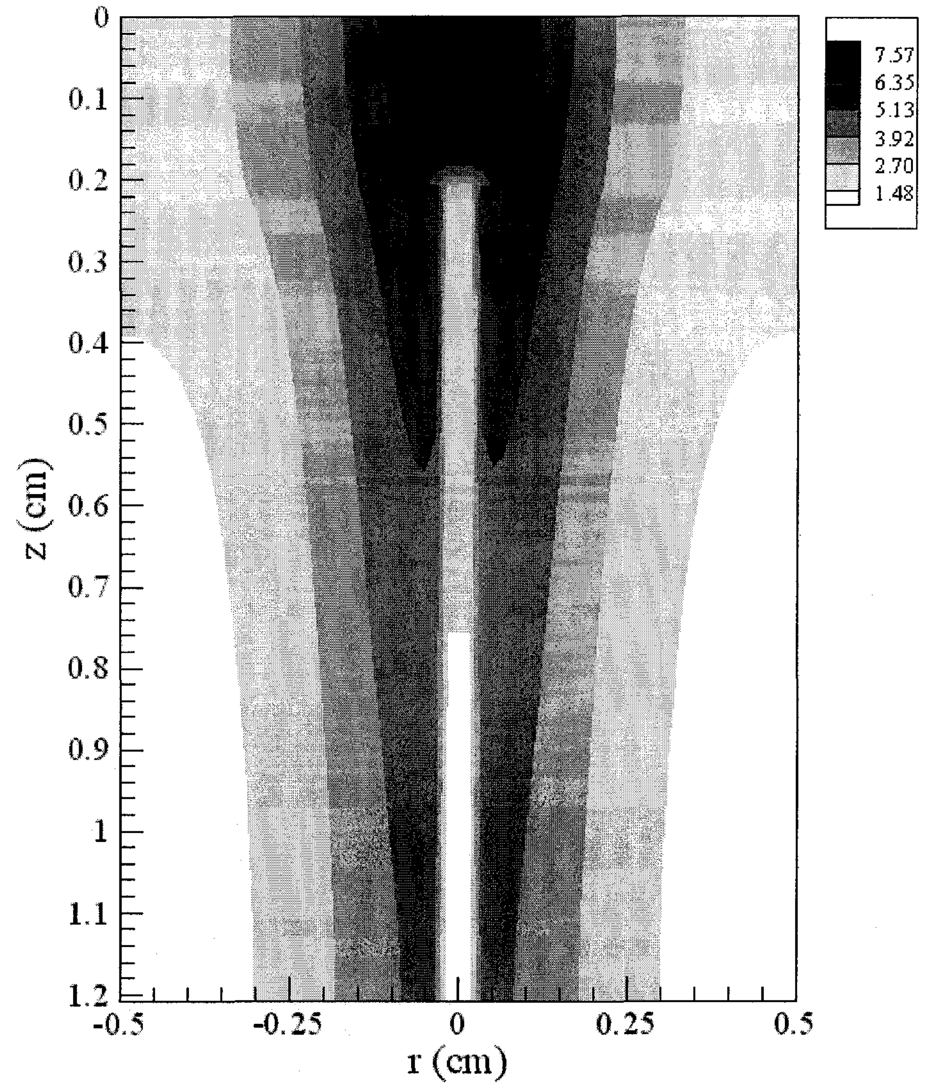

Figure 5.12 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 672$ seconds with a blood vessel

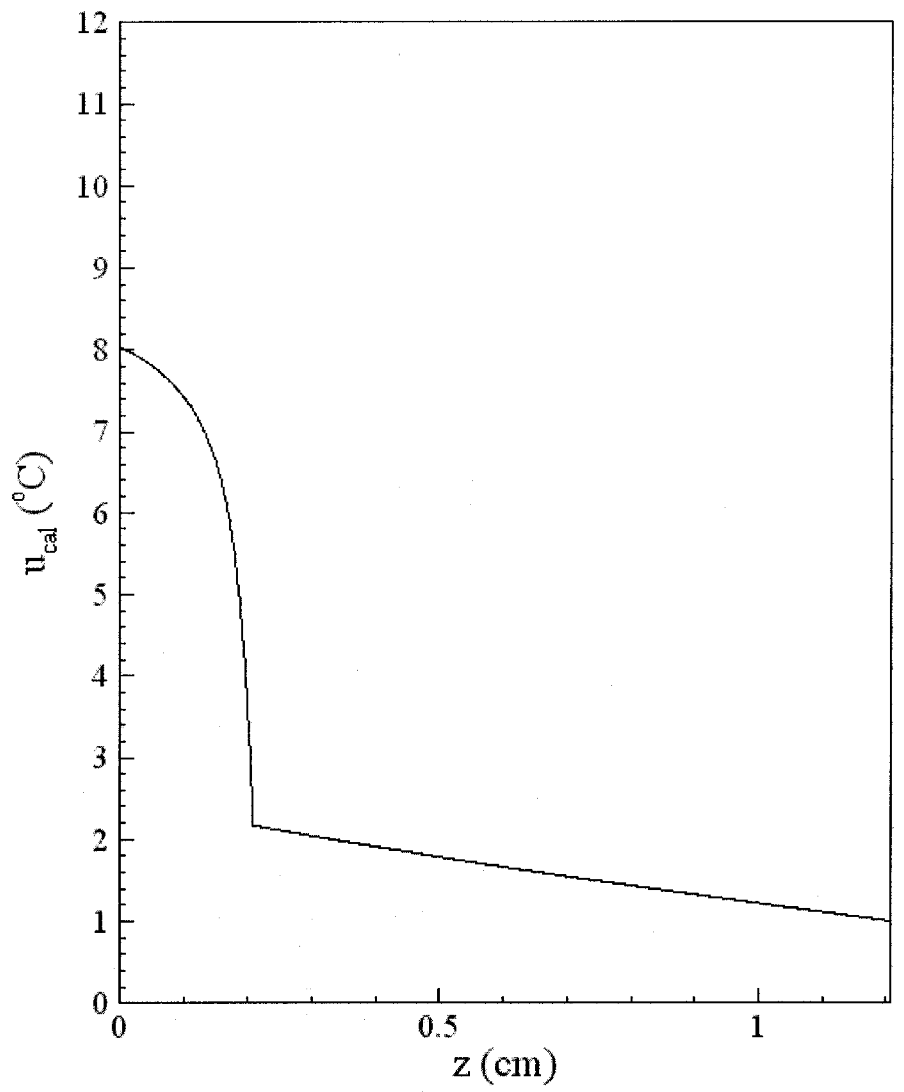

Figure 5.13 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 672$ seconds with a blood vessel

5.3.2 Calculation Case 2

The elevated temperatures at the center and perimeter are $8^{\circ}C$ and $3^{\circ}C$, 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 $\varphi = 0$ and $\varphi = \pi$ and with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, respectively. Figure 5.16 and 5.17 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 5.18 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.

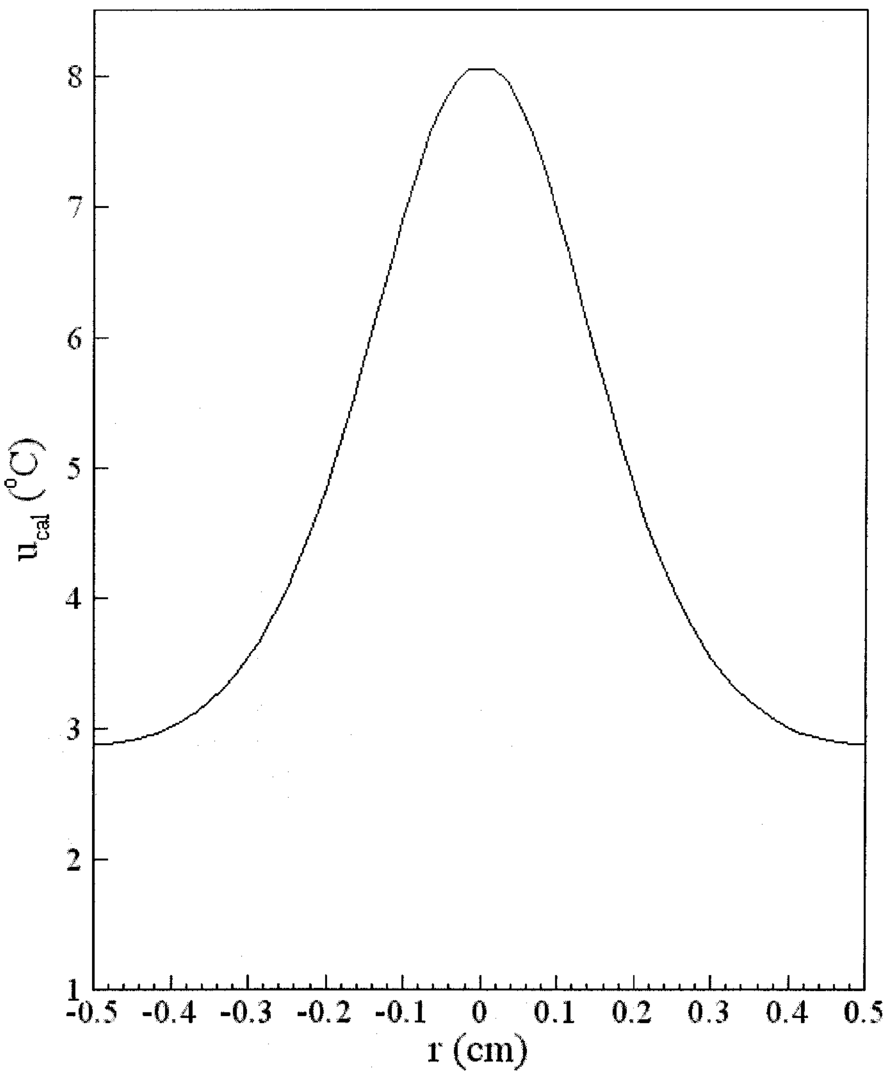

Figure 5.14 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 1044$ seconds with a blood vessel

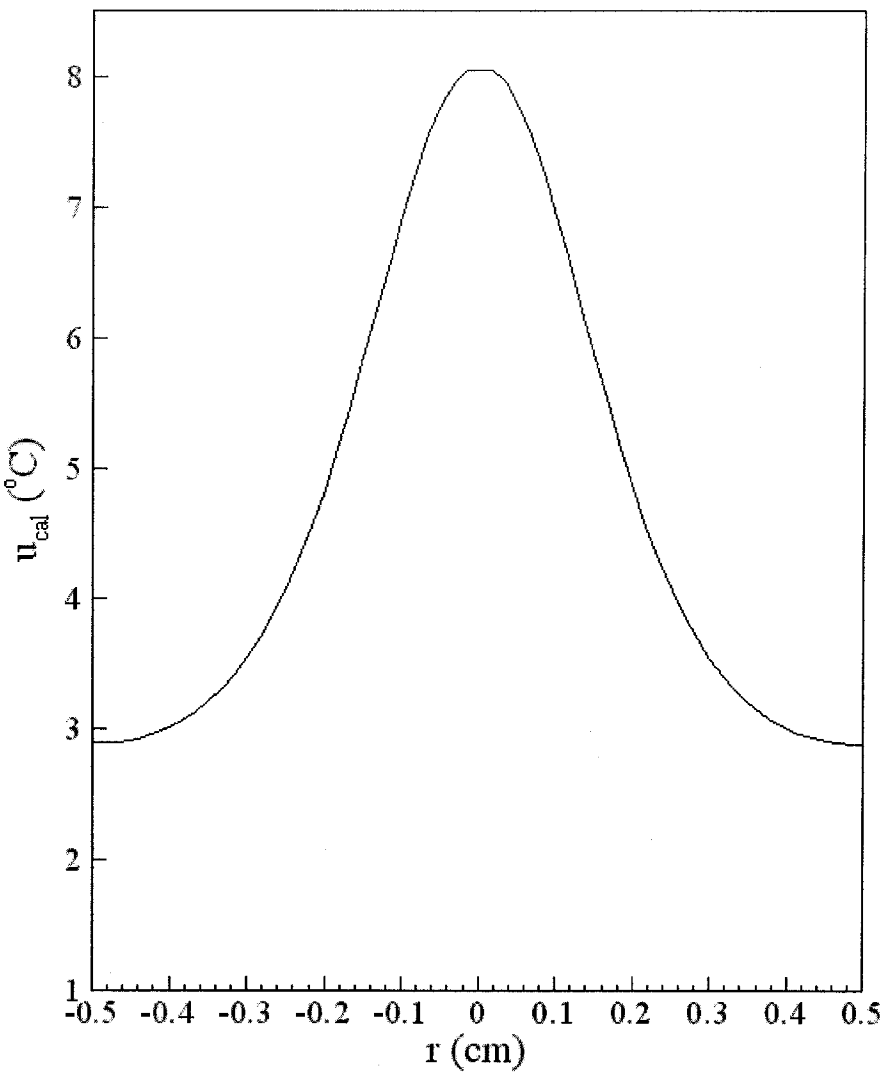

Figure 5.15 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1044$ seconds with a blood vessel

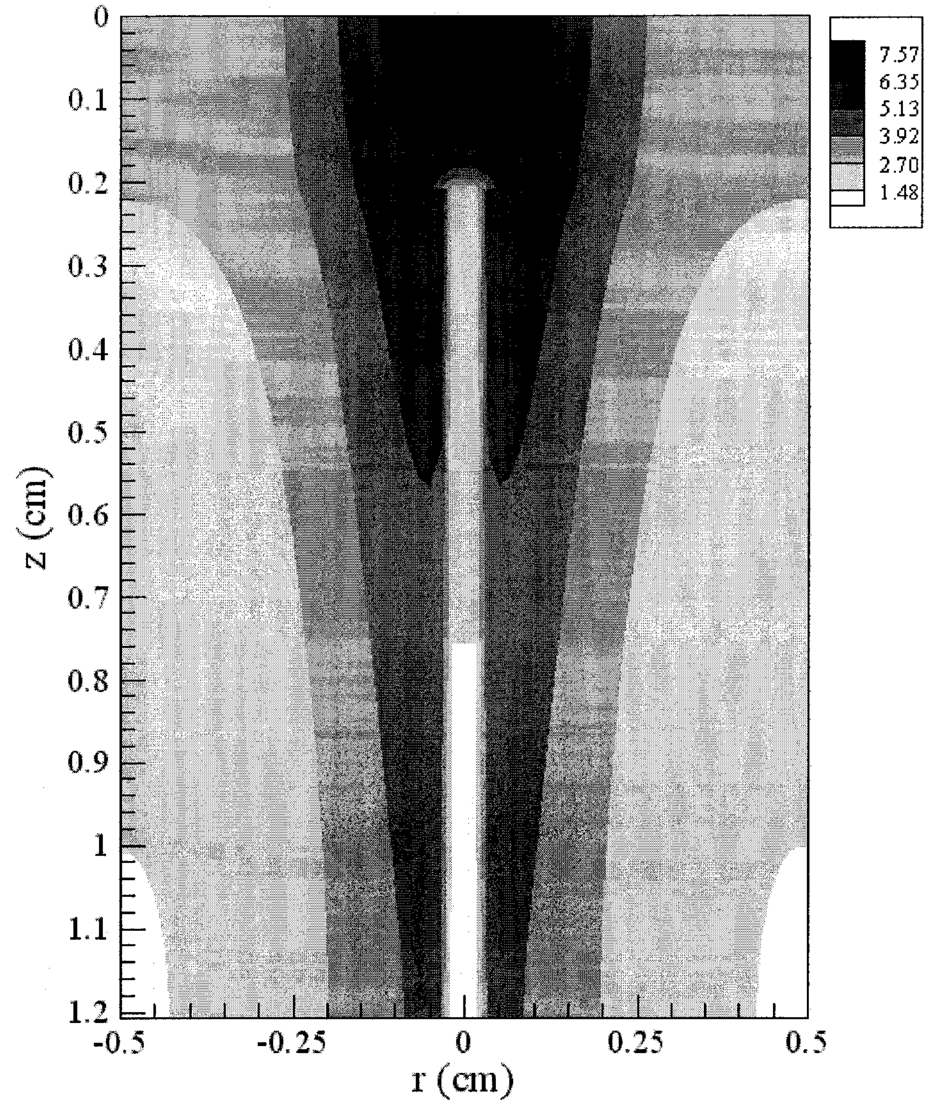

Figure 5.16 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 1044$ seconds with a blood vessel

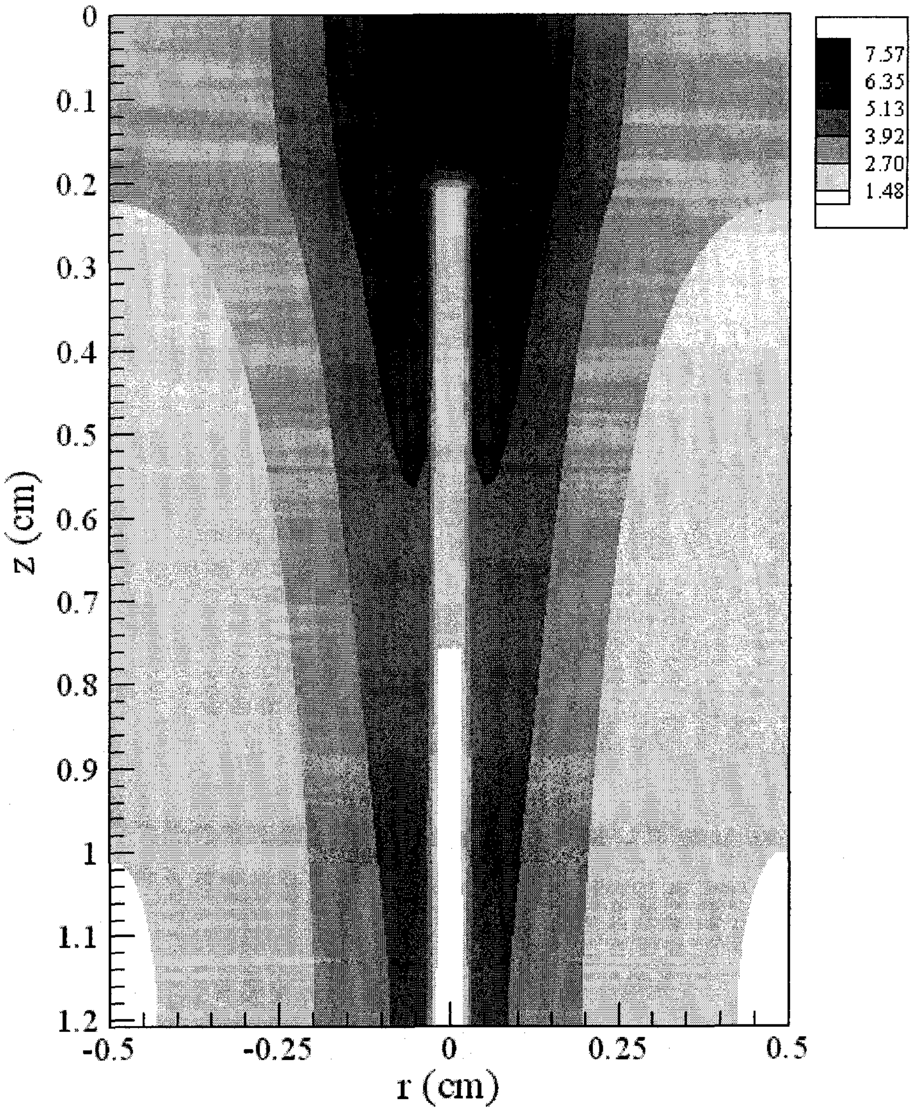

Figure 5.17 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1044$ seconds with a blood vessel

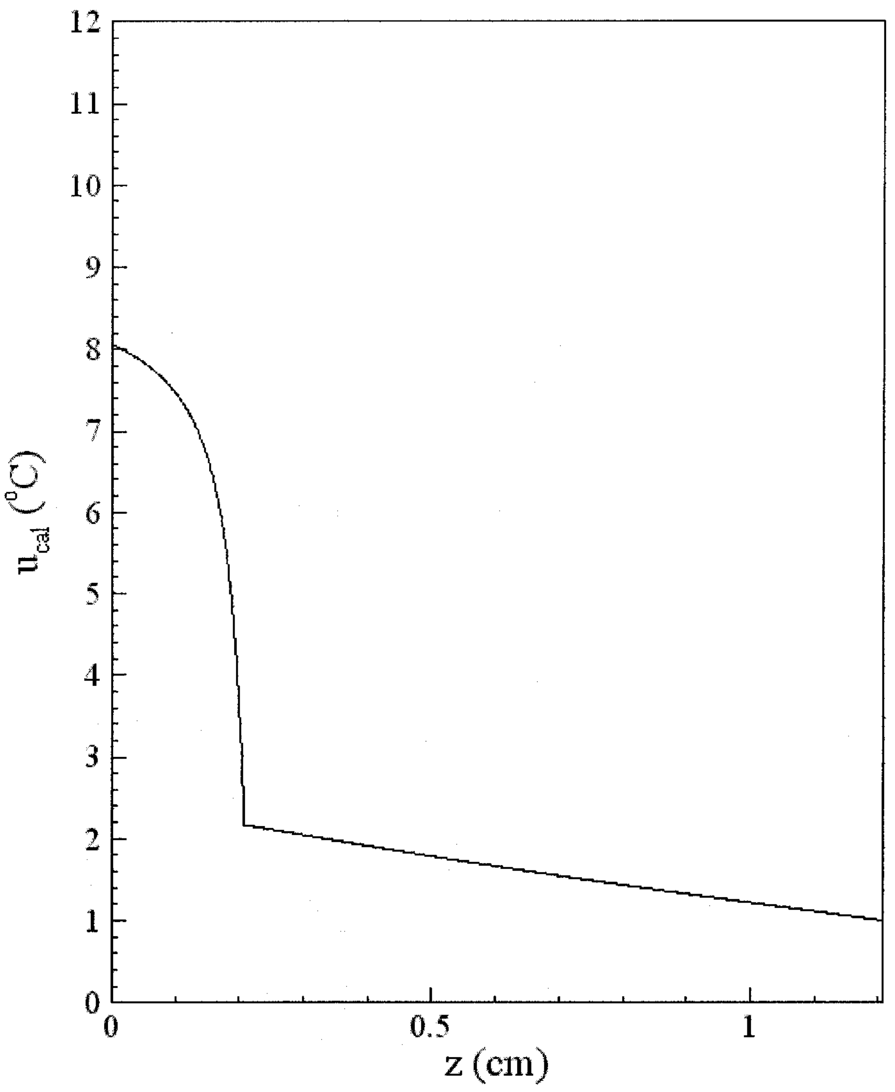

Figure 5.18 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1044$ seconds with a blood vessel

5.3.3 Calculation Case 3

The elevated temperatures at the center and perimeter are $8^{\circ}C$ and $4^{\circ}C$, 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.

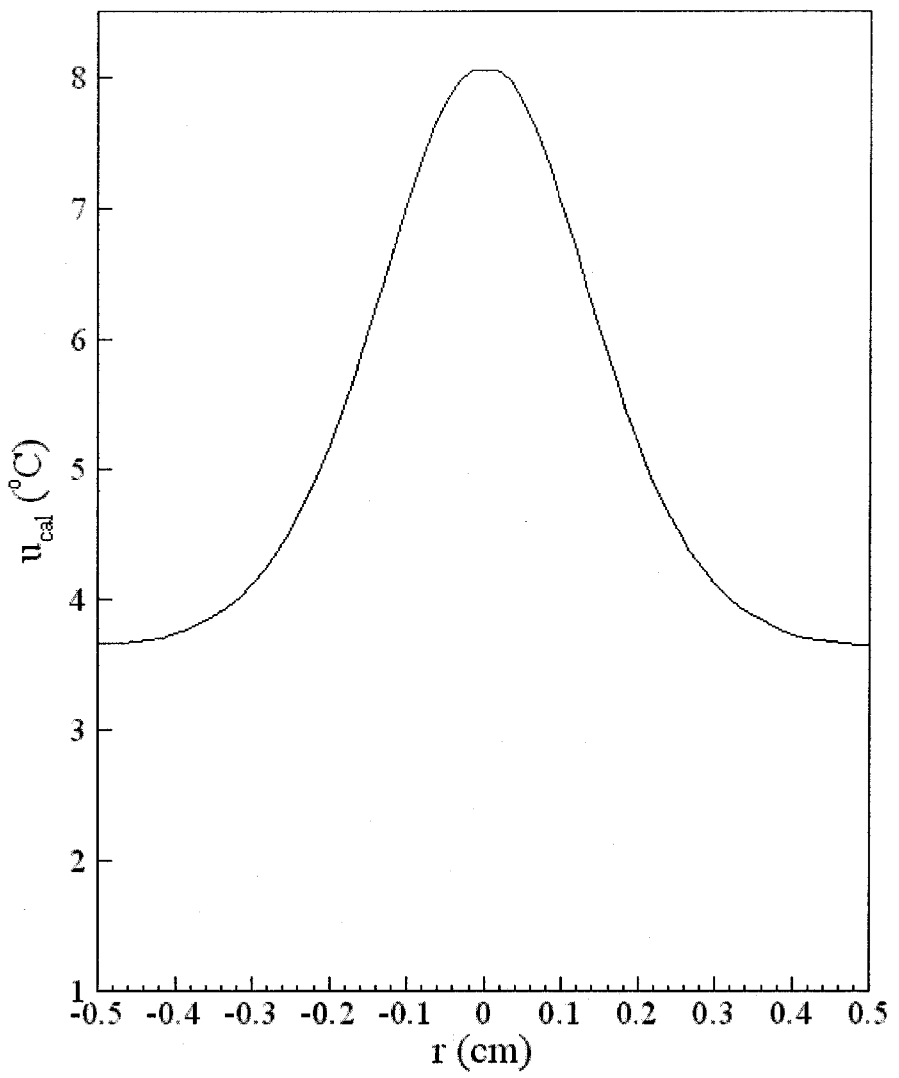

Figure 5.19 The elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 1509$ seconds with a blood vessel

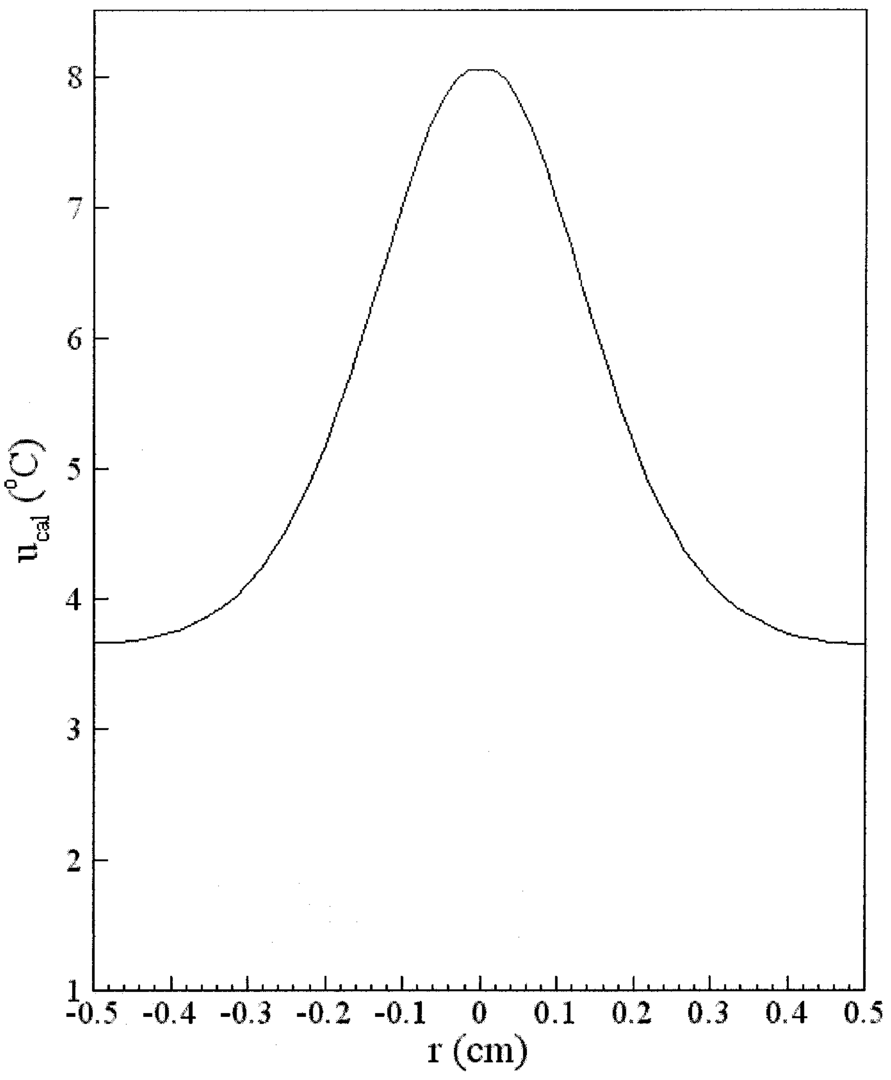

Figure 5.20 The elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1509$ seconds with a blood vessel

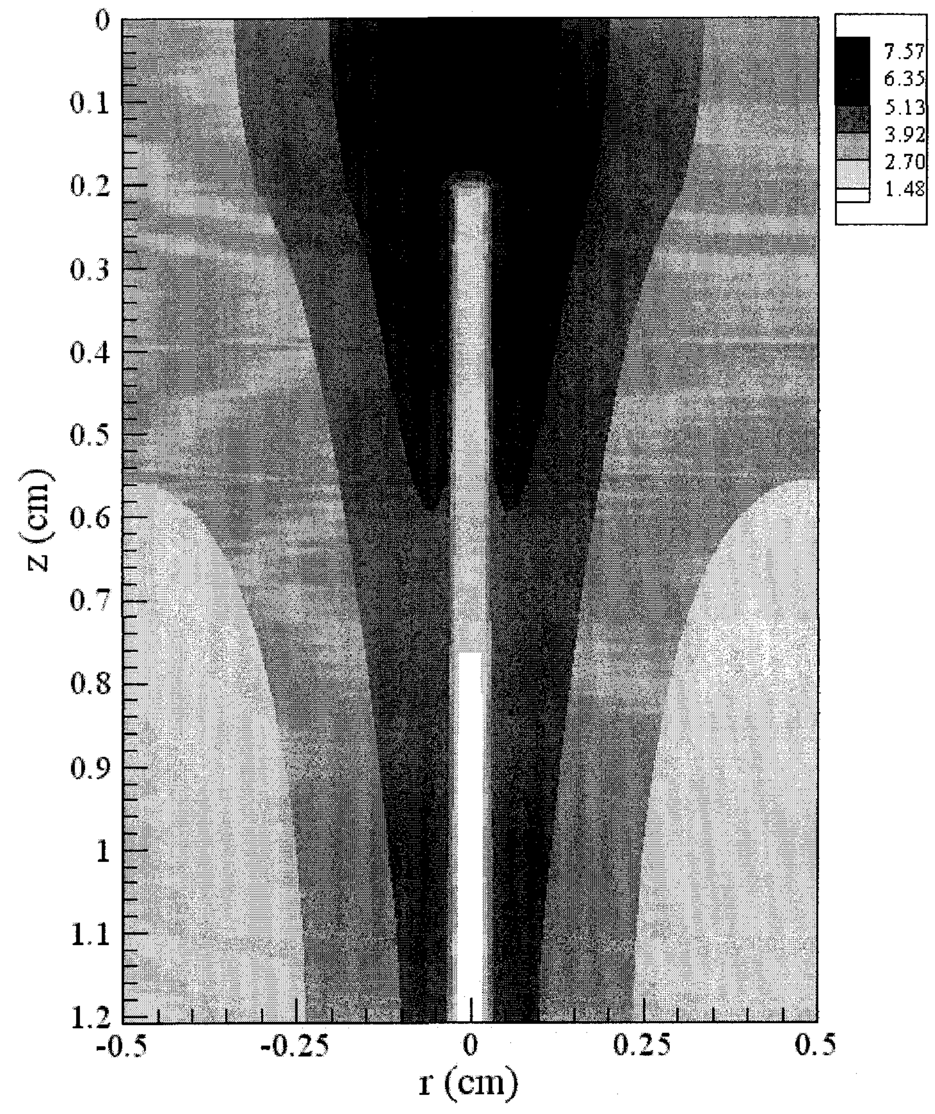

Figure 5.21 The contours of the elevated temperature distributions in the cross section with $\varphi = 0$ and $\varphi = \pi$, at $t = 1509$ seconds with a blood vessel

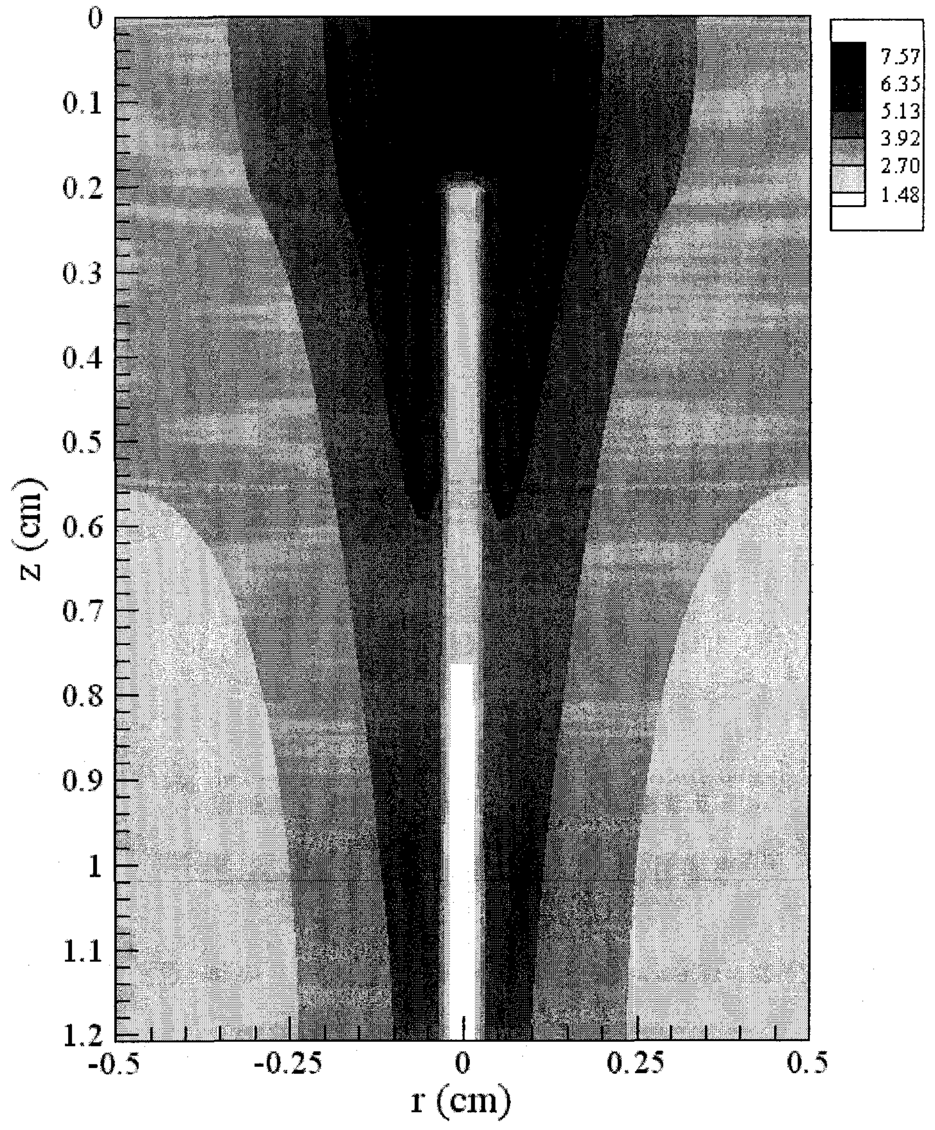

Figure 5.22 The contours of the elevated temperature distributions in the cross section with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 1509$ seconds with a blood vessel

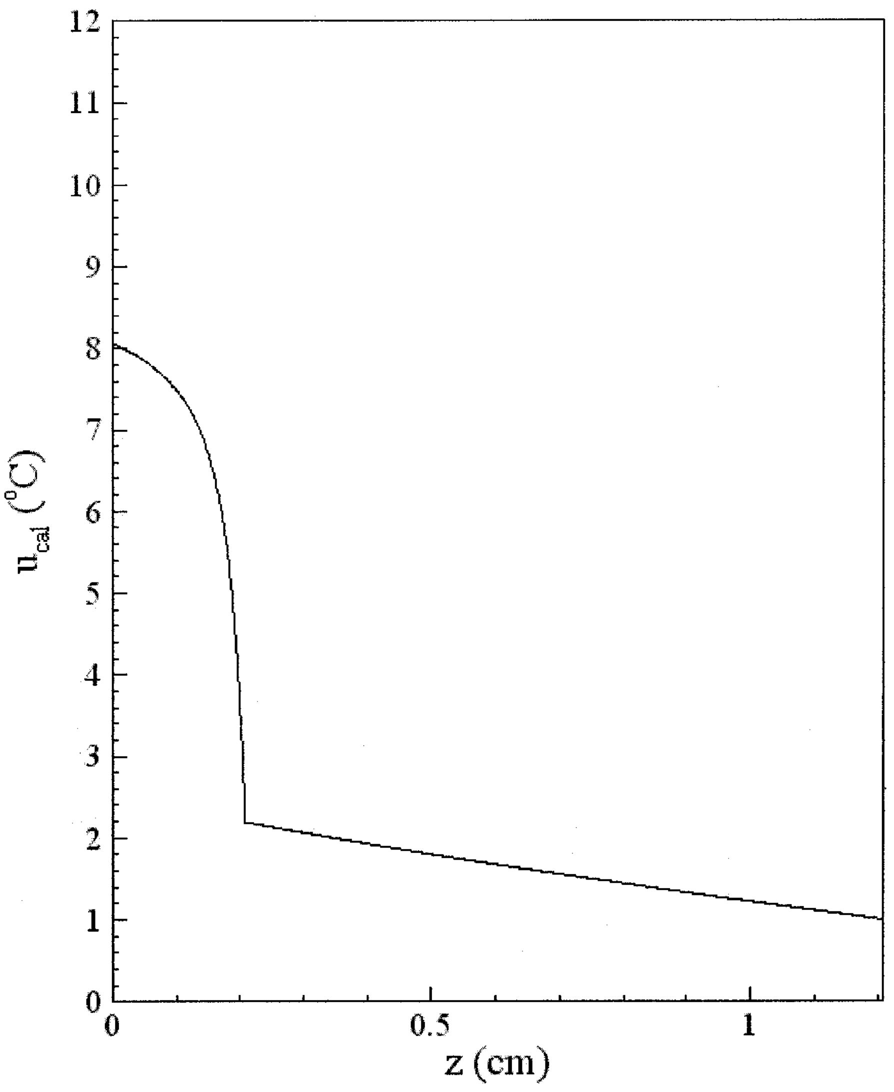

Figure 5.23 The elevated temperature profile along the depth (the z-direction) at the center of the skin surface, at $t = 1509$ seconds with a blood vessel

5.4 Grid Independent Experiment

Finally, to test our algorithm to be independent of the grid size, there are two additional meshes (r,φ,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.

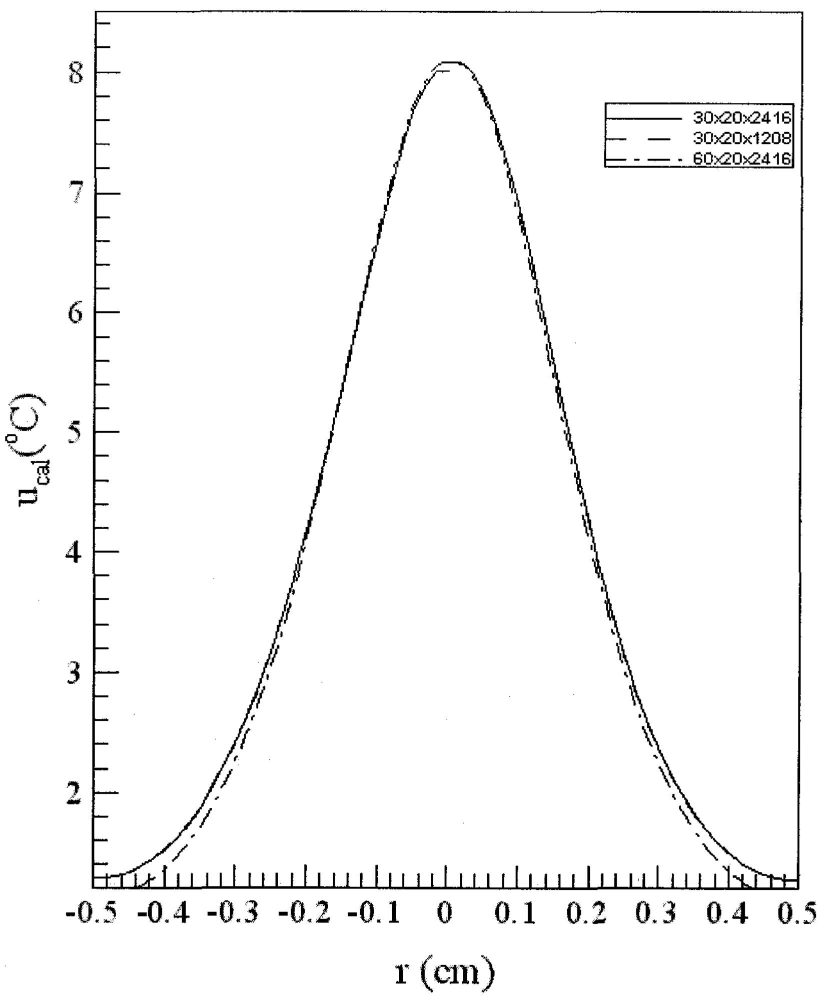

Figure 5.24 Elevated temperature profiles along the diameter on the skin surface with $\varphi = 0$ and $\varphi = \pi$, at $t = 410$ with a blood vessel

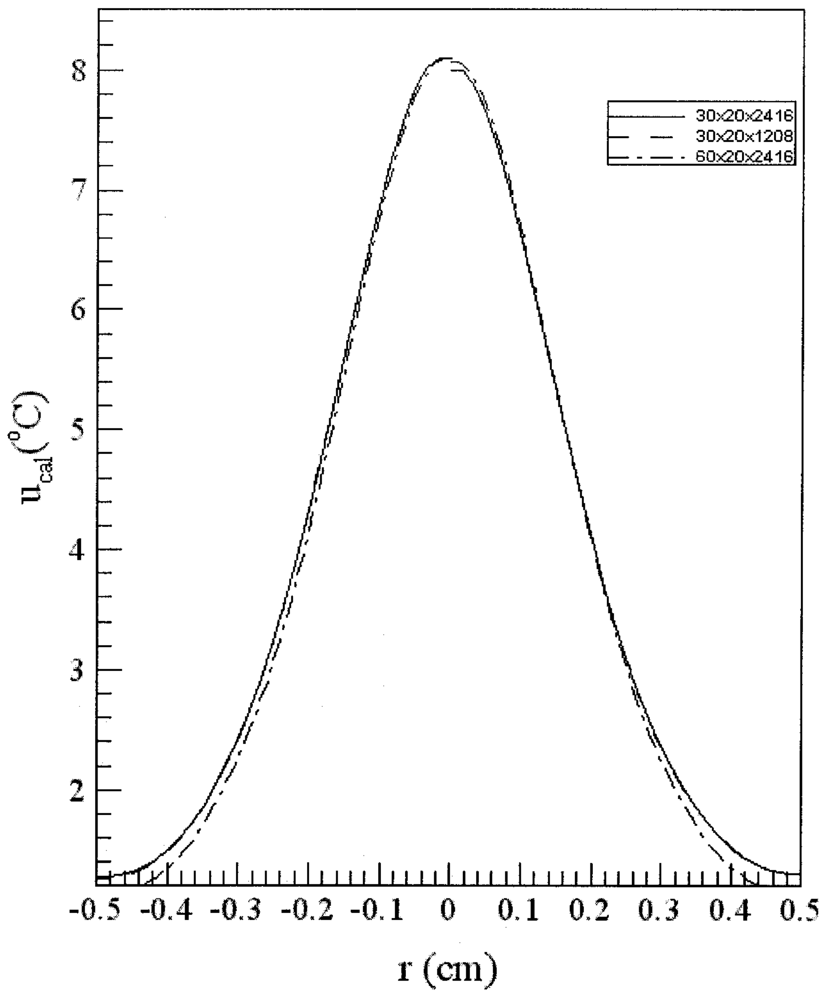

Figure 5.25 Elevated temperature profiles along the diameter on the skin surface with $\varphi = \frac{\pi}{2}$ and $\varphi = \frac{3\pi}{2}$, at $t = 410$ with a blood vessel

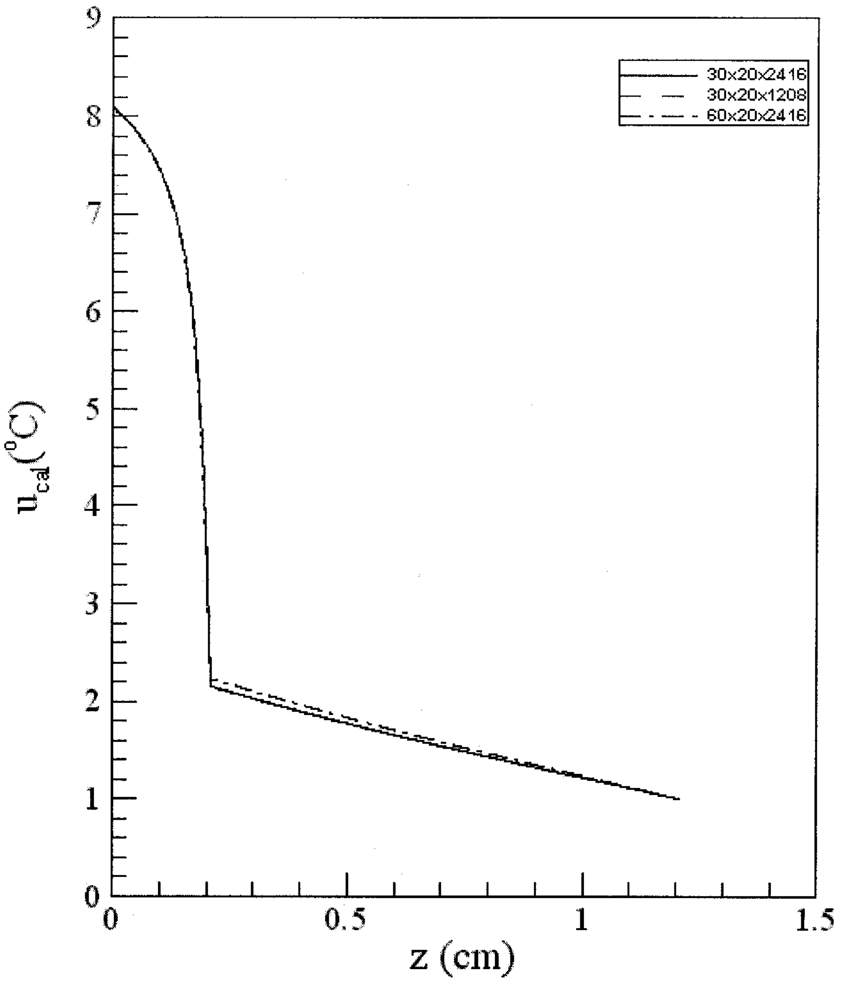

Figure 5.26 Elevated temperature profiles along the depth (the z-direction) at the center of the skin surface, at $t = 410$ with a blood vessel

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.

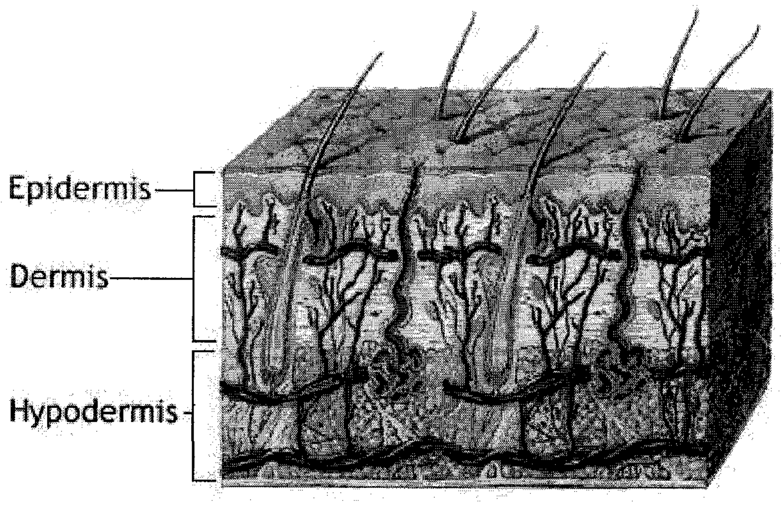

Figure 6.1 Skin structure embedded with multi-level blood vessels

APPENDIX A

SOURCE CODE FOR SOLVING THE 3D SKIN STURCTURE 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 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
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,Rchange1,Phichange1,Zchange1;
      double p1,p2,p3,qc1,qc2,qc3,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,z,t,n,nt ;
      int MaxLen,MaxWid,MaxHig;
  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;j++)
     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];
    beta = new double **[l];
    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];
      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<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];
          vn[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;
          vn[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; Reff1=0.93; Reff2=0.93;
 Reff3=0.93; pi=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;
  wb1=0.0; wb2=0.0005; wb3=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;
};

    ~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 FileWrit(int);
    void Clear(void);
 };

 void zlihcp::FileWrit(int time1)
    {
     int i,k;
      ofstream fout1,fout2,fout21,fout3,fout31;
      char str[20],str1[20]="zt",str2[20]="rzt",str21[20]="rztc",str3[20]="center",str31[20]="centerc";
      sprintf(str,"%d",time1);
      strcat(str1,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 I=1209,F=POINT"<<endl;
      for(k=0;k<=NZ3;k++)
       fout1<<double(k*deltaZ)<<" "<<vnew[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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][0][k]<<endl;
          }
        }
      fout2<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
      fout2<<"  VARIABLES = \"R\", \"Z\", \"Temperature\" "<<endl;
      fout2<<" ZONE T=\"reverse\",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)<<" "<<vnew[i][NPhi/2][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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][NPhi/4][k]<<endl;
        }
      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)<<" "<<vnew[i][3*NPhi/4][k]<<endl;
        }
      fout21.close();
      //////////////Center///////////////
      fout3.open(str3,ios::out);
      fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
      fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
      fout3<<" ZONE T=\"right\",I=31, F=POINT"<<endl;
      for(i=0;i<=NR;i++)
        fout3<<double(i*deltaR)<<" "<<vnew[i][0][0]<<endl;
      fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
      fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
      fout3<<" ZONE T=\"reverse\",I=31, F=POINT"<<endl;
      for(i=0;i<=NR;i++)
        {
        fout3<<-double(i*deltaR)<<" "<<vnew[i][NPhi/2][0]<<endl;
      }
      fout3.close();
```

```
///////////////Center cross////////////////
      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)<<" "<<vnew[i][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)<<" "<<vnew[i][3*NPhi/4][0]<<endl;
      fout31.close();
  }
void zlihcp::InitQ(double P0,double Cr,double Cp) // Initilize 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-Reff1 );
             }
           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-Reff2);
            }
           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-Reff3 );
            }
        }
       }
     }

double zlihcp::IntmTrsy(double P0) // Time Iteration and Tri-diagonal systme
     {
      MaxErr=1.0;  nt=0;
      while(++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]=vold[i][j][z];
             }
           }
          }
       cout<<nt<<"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++)
      { /////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==NPhi)
          {
            Phichange = k1*deltaT*(vold[i][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 //////////////
        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][1][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);
////////////Prepare the coefficient for Thomas way///////////////////////
        f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*qc1-wb1*cb1*deltaT)*vn[i][j][z];
        b[z]=(k1*deltaT)/(deltaZ*deltaZ);
        a[i][z]=2*p1*qc1+wb1*cb1*deltaT+k1*deltaT*(4*i+1)/(i*deltaR*deltaR)+k1*deltaT*4/pow(deltaPhi*i*deltaR,2)
                +2*k1*deltaT/pow(deltaZ,2);
        c[z]=(k1*deltaT)/(deltaZ*deltaZ);
        d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                    k1*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
      }
    a[i][1]= a[i][1]-b[1];
    b[1] = 0;
    b[NZ1]=k1;
    a[i][NZ1]=k1+k2;
    c[NZ1]=k2;
    d[i][j][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][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////////////////////////
           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][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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][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[z]=k2*deltaT/(deltaZ*deltaZ);
                    a[i][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                        +2*k2*deltaT/pow(deltaZ,2);
                    c[z]=k2*deltaT/(deltaZ*deltaZ);
                    d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                             k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
                  }
                 b[NZ2]=k2;
                 a[i][NZ2]=k2+k3;
                 c[NZ2]=k3;
                 d[i][j][NZ2]=0;
          ///////////////The third layer /////////////////////////
         for(z=NZ2+1;z<=NZ3-1;z++)
         { //////////////The n+1 state ////////////////////////
                    Rchange = k3*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 = k3*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                     }
                    else
                     {
                      Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/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*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
                    if(j==NPhi)
                     {
                     Phichange1 = k3*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                     }
                    else
                     {
                      Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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////////////////////////
                    f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q3[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z];
                    b[z]=k3*deltaT/(deltaZ*deltaZ);
                    a[i][z]=2*p3*qc3+wb3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                        +2*k3*deltaT/pow(deltaZ,2);
                    c[z]=k3*deltaT/(deltaZ*deltaZ);
                    d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                           k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
          }
          a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1];
          c[NZ3-1]=0;
              }
     }
// tri-diagonal system
         for(i=1;i<=NR-1;i++)
          {
          for(j=1;j<=NPhi;j++)
           {
             v[i][j][NZ3]=0.0;
            beta[i][j][NZ3]=0.0;
            for(z=NZ3-1;z>=1;z--)
             {
              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<=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(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][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];
           }
         }
       }
   }
 //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<<"temperation"<<vnew[0][0][0]<<endl;
 }
```

```
        FileWrit(1000);
        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;
          }
        }
       }
 }

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, T1m, T2m, deltaP, X,Tpoint;
      long double S, Snew, Pnew,error1;
      ofstream foutl4;  foutl4.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)*(Tpoint-T1m);
      foutl4<<"PNEW"<<Pnew<<endl;
      foutl4<<"SNEW"<<Snew<<endl;
      }
     while ((Snew-S)/Snew > error1 );
     foutl4<<"end"<<endl;
     foutl4.close();
     return 0;
  }
```

```
/*Table A.2 Program 2: Source code of step 2 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 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 2
#define CenTemp 8
#define LSS4 0.04

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,Rchange1,Phichange1,Zchange1;
      double p1,p2,p3,qc1,qc2,qc3,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,z,t,n ;
      int MaxLen,MaxWid,MaxHig;

 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;
    a = new double *[l];
    for(j=0;j<l;j++)
     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];
    beta = new double **[l];
    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];
      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<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];
        vn[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;
          vn[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; Reff1=0.93; Reff2=0.93;
 Reff3=0.93; pi=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;
  wb1=0.0; wb2=0.0005; wb3=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;
};

~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 FileWrit(int);
    void Clear(void);
  };

void zlihcp::FileWrit(int time1)
   {
    int i,k;
    ofstream fout1,fout2,fout21,fout3,fout31;
    char str[20],str1[20]="zt",str2[20]="rzt",str21[20]="rztc",str3[20]="center",str31[20]="centerc";
    sprintf(str,"%d",time1); strcat(str1,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 I=1209,F=POINT"<<endl;
    for(k=0;k<=NZ3;k++)
     fout1<<double(k*deltaZ)<<" "<<vnew[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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][0][k]<<endl;
    }
    fout2<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout2<<"  VARIABLES = \"R\", \"Z\", \"Temperature\" "<<endl;
    fout2<<" ZONE T=\"reverse\",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)<<" "<<vnew[i][NPhi/2][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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][NPhi/4][k]<<endl;
    }
    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)<<" "<<vnew[i][3*NPhi/4][k]<<endl;
     }
    fout21.close();
    ///////////////Center////////////////
    fout3.open(str3,ios::out);
    fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
    fout3<<" ZONE T=\"right\",I=31, F=POINT"<<endl;
    for(i=0;i<=NR;i++)
      fout3<<double(i*deltaR)<<" "<<vnew[i][0][0]<<endl;
    fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
    fout3<<" ZONE T=\"reverse\",I=31, F=POINT"<<endl;
    for(i=0;i<=NR;i++)
      fout3<<-double(i*deltaR)<<" "<<vnew[i][NPhi/2][0]<<endl;
    fout3.close();
   ///////////////Center cross/////////////
    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)<<" "<<vnew[i][NPhi/4][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)<<" "<<vnew[i][3*NPhi/4][0]<<endl;
    fout31.close();
   }

  void zlihcp::InitQ(double P0,double Cr,double Cp) // Initilize 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-Reff1 );
            }

          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-Reff2);
           }
          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-Reff3 );
           }
          }
        }
    }

  double zlihcp::IntmTrsy(double P0) // Time Iteration and Tri-diagonal systme
   {
    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;i++)
         {
          for(j=0;j<=NPhi;j++)
           {
            for(z=0;z<=NZ3;z++)
             {
            vn[i][j][z]=vold[i][j][z];
           }
          }
         }
      cout<<nt<<"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++)
              {
                /////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==NPhi)
                    {
                    Phichange = k1*deltaT*(vold[i][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 //////////////
                 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][1][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);
                  ////////////Prepare the coefficient for Thomas way//////////////////////
                  f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*qc1-wb1*cb1*deltaT)*vn[i][j][z];
                  b[z]=(k1*deltaT)/(deltaZ*deltaZ);
                  a[i][z]=2*p1*qc1+wb1*cb1*deltaT+k1*deltaT*(4*i+1)/(i*deltaR*deltaR)
                        +k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+2*k1*deltaT/pow(deltaZ,2);
                   c[z]=(k1*deltaT)/(deltaZ*deltaZ);
                   d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                               k1*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
              }
            a[i][1]= a[i][1]-b[1];
            b[1] = 0;
            b[NZ1]=k1;
            a[i][NZ1]=k1+k2;
            c[NZ1]=k2;
            d[i][j][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][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////////////////////////
          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][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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][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[z]=k2*deltaT/(deltaZ*deltaZ);
                   a[i][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                       +2*k2*deltaT/pow(deltaZ,2);
                   c[z]=k2*deltaT/(deltaZ*deltaZ);
                   d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                           k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
                  }
            b[NZ2]=k2;
            a[i][NZ2]=k2+k3;
            c[NZ2]=k3;
            d[i][j][NZ2]=0;
          ///////////////The third layer //////////////////////////
         for(z=NZ2+1;z<=NZ3-1;z++)
         { ///////////////The n+1 state ////////////////////////
                   Rchange = k3*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 = k3*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                    }
                   else
                    {
                    Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/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*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
                   if(j==NPhi)
                    {
                    Phichange1 = k3*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                    }
                   else
                    {
                    Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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////////////////////////
                   f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q3[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z];
                   b[z]=k3*deltaT/(deltaZ*deltaZ);
                   a[i][z]=2*p3*qc3+wb3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                       +2*k3*deltaT/pow(deltaZ,2);
                   c[z]=k3*deltaT/(deltaZ*deltaZ);
                   d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                          k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
                  }
            a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1]; c[NZ3-1]=0;
          }
      }
   // tri-diagonal system
      for(i=1;i<=NR-1;i++)
       {
        for(j=1;j<=NPhi;j++)
          {
            v[i][j][NZ3]=0.0;
            beta[i][j][NZ3]=0.0;
            for(z=NZ3-1;z>=1;z--)
               {
                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<=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(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][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];
////////////////////////////////////////////////////////
        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);
    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);
      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,20*deltaPhi);
       if(LSS_4<LSS4)
        {
         TimeRec[CountNum] = nt;
         FlagRec[CountNum] = flag;
         CountNum ++;
         flag =2;
        }
       if((point[1]>EndTemp)||(point[2]>EndTemp)||(point[3]>EndTemp)||(point[4]>EndTemp)||(point[0]<EndTemp))
        {
         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]>CenTemp)
       {
        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]>CenTemp)
      {
       TimeRec[CountNum] = nt;
       FlagRec[CountNum] = flag;
       goto loopend;
      }
     }
    }
 }
 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;
          }
        }
       }
}

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;
      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 "<<i<<" CoolTime "<<zl.TimeRec[i]<<endl;
          if(zl.FlagRec[i] ==1)
           fout14<<"Number "<<i<<" HeatTime "<<zl.TimeRec[i]<<endl;
          else
           fout14<<"Number "<<i<<" 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 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 T1 965    //LSS4 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 Rchange,Phichange,Zchange,Rchange1,Phichange1,Zchange1;
      double p1,p2,p3,qc1,qc2,qc3,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,z,t,n ;
      int MaxLen,MaxWid,MaxHig;
   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;
     a = new double *[l];
     for(j=0;j<l;j++)
      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];
     vsave = new double **[l];
```

```
        beta = new double **[l];
        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];
          vnew[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 [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];
            vn[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;
             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; Reff1=0.93; Reff2=0.93;
    Reff3=0.93; pi=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;
     wb1=0.0; wb2=0.0005; wb3=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;
   } ;

  ~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);
    double IntmTrsy1(double);
    double RunAll1 (double);
    void Clear1(void);
    void FileWrit(int);
    void Clear(void);
  };

void zlihcp::FileWrit(int time1)
   {
    int i,k;  ofstream fout1,fout2,fout21,fout3,fout31;
    char str[20],str1[20]="zt",str2[20]="rzt",str21[20]="rztc",str3[20]="center",str31[20]="centerc";
    sprintf(str,"%d",time1);  strcat(str1,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 I=1209,F=POINT"<<endl;
    for(k=0;k<=NZ3;k++)
     fout1<<double(k*deltaZ)<<" "<<vnew[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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][0][k]<<endl;
      }
    fout2<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout2<<"  VARIABLES = \"R\", \"Z\", \"Temperature\" "<<endl;
    fout2<<" ZONE T=\"reverse\",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)<<" "<<vnew[i][NPhi/2][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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][NPhi/4][k]<<endl;
     }
    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)<<" "<<vnew[i][3*NPhi/4][k]<<endl;
      }
     fout21.close();
    ////////////////Center/////////////////
    fout3.open(str3,ios::out);
    fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
    fout3<<" ZONE T=\"right\",I=31, F=POINT"<<endl;
    for(i=0;i<=NR;i++)
      fout3<<double(i*deltaR)<<" "<<vnew[i][0][0]<<endl;
    fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
    fout3<<" ZONE T=\"reverse\",I=31, F=POINT"<<endl;
    for(i=0;i<=NR;i++)
      fout3<<-double(i*deltaR)<<" "<<vnew[i][NPhi/2][0]<<endl;
    fout3.close();
   ///////////////Center cross/////////////
    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)<<" "<<vnew[i][NPhi/4][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)<<" "<<vnew[i][3*NPhi/4][0]<<endl;
    fout31.close();
   }

void zlihcp::InitQ(double P0,double Cr,double Cp) // Initilize 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-Reff1 );
        }
         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-Reff2);
          }
         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-Reff3 );
          }
        }
     }
   }

double zlihcp::IntmTrsy(double P0) // Time Iteration and Tri-diagonal systme
   {
    MaxErr=1.0;  nt=0;
    while(nt>-1)
     {
       nt++;
       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]=vold[i][j][z];
        vsave[i][j][z]=vold[i][j][z];
       }
     }
    }
 cout<<nt<<"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++)
       { /////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==NPhi)
              {
              Phichange = k1*deltaT*(vold[i][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 //////////////

           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][1][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);


            ////////////Prepare the coefficient for Thomas way///////////////////////
            f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*qc1-wb1*cb1*deltaT)*vn[i][j][z];

      b[z]=(k1*deltaT)/(deltaZ*deltaZ);
      a[i][z]=2*p1*qc1+wb1*cb1*deltaT+k1*deltaT*(4*i+1)/(i*deltaR*deltaR)
              +k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+2*k1*deltaT/pow(deltaZ,2);
       c[z]=(k1*deltaT)/(deltaZ*deltaZ);
       d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                   k1*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
     }
    a[i][1]= a[i][1]-b[1];
     b[1] = 0;
     b[NZ1]=k1;
     a[i][NZ1]=k1+k2;
     c[NZ1]=k2;
     d[i][j][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][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/////////////////////////
            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][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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][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[z]=k2*deltaT/(deltaZ*deltaZ);
            a[i][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                +2*k2*deltaT/pow(deltaZ,2);
            c[z]=k2*deltaT/(deltaZ*deltaZ);
            d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                    k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
            }
       b[NZ2]=k2;
       a[i][NZ2]=k2+k3;
       c[NZ2]=k3;
        d[i][j][NZ2]=0;
     //////////////The third layer /////////////////////////
     for(z=NZ2+1;z<=NZ3-1;z++)
      { //////////////The n+1 state ////////////////////////
            Rchange = k3*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 = k3*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
             }
            else
             {
              Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/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*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
            if(j==NPhi)
             {
             Phichange1 = k3*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
             }
            else
             {
              Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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///////////////////////
            f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q3[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z];
            b[z]=k3*deltaT/(deltaZ*deltaZ);
            a[i][z]=2*p3*qc3+wb3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                +2*k3*deltaT/pow(deltaZ,2);
            c[z]=k3*deltaT/(deltaZ*deltaZ);
            d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                     k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
         }
      a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1];
      c[NZ3-1]=0;
     }
   }
// tri-diagonal system
     for(i=1;i<=NR-1;i++)
       {
       for(j=1;j<=NPhi;j++)
        {
          v[i][j][NZ3]=0.0;
          beta[i][j][NZ3]=0.0;
```

```
                for(z=NZ3-1;z>=1;z--)
                 {
                  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<=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(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][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];
// 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);
         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);
          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,20*deltaPhi);
            if(LSS_4<LSS4)
             {
              TimeRec[CountNum] = nt;
              FlagRec[CountNum] = flag;
              CountNum ++;
              goto loopend;
             }
            if((point[1]>EndTemp)||(point[2]>EndTemp)||(point[3]>EndTemp)||(point[4]>EndTemp)||(point[0]<EndTemp))
             {
              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];
}
```

```
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::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]=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 zlihcp::IntmTrsy1(double P0) // Time Iteration and Tri-diagonal systme
      {
       MaxErr=1.0;
       while(nt<T2) //need the center temp back to 8
        {  ////////////Change center point////////////////
          nt++;  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]=vold[i][j][z];
                }
              }
            }
```

```
cout<<nt<<"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++)
       {
        /////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==NPhi)
                 {
                 Phichange = k1*deltaT*(vold[i][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 //////////////
              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][1][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);
     ////////////Prepare the coefficient for Thomas way//////////////////////
     f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*qc1-wb1*cb1*deltaT)*vn[i][j][z];
     b[z]=(k1*deltaT)/(deltaZ*deltaZ);
     a[i][z]=2*p1*qc1+wb1*cb1*deltaT+k1*deltaT*(4*i+1)/(i*deltaR*deltaR)
            +k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+2*k1*deltaT/pow(deltaZ,2);
     c[z]=(k1*deltaT)/(deltaZ*deltaZ);
     d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                        k1*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
     }
  a[i][1]= a[i][1]-b[1];
  b[1] = 0;
  b[NZ1]=k1;
  a[i][NZ1]=k1+k2;
  c[NZ1]=k2;
  d[i][j][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][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////////////////////////
                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][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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][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[z]=k2*deltaT/(deltaZ*deltaZ);
                    a[i][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                        +2*k2*deltaT/pow(deltaZ,2);
                    c[z]=k2*deltaT/(deltaZ*deltaZ);
                    d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                              k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
                    }
                b[NZ2]=k2;
                 a[i][NZ2]=k2+k3;
                 c[NZ2]=k3;
                 d[i][j][NZ2]=0;
            //////////////The third layer ///////////////////////
            for(z=NZ2+1;z<=NZ3-1;z++)
             {
             /////////////The n+1 state ///////////////////////
                    Rchange = k3*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 = k3*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                     }
                    else
                     {
                      Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/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*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
                    if(j==NPhi)
                     {
                     Phichange1 = k3*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                     }
                    else
                     {
                      Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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//////////////////////
                    f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q3[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z];
                    b[z]=k3*deltaT/(deltaZ*deltaZ);
                    a[i][z]=2*p3*qc3+wb3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                        +2*k3*deltaT/pow(deltaZ,2);
               c[z]=k3*deltaT/(deltaZ*deltaZ);
               d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                            k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
              }
             a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1];
             c[NZ3-1]=0;
            }
        }
  // tri-diagonal system
        for(i=1;i<=NR-1;i++)
        {
         for(j=1;j<=NPhi;j++)
          {
           v[i][j][NZ3]=0.0;
           beta[i][j][NZ3]=0.0;
           for(z=NZ3-1;z>=1;z--)
            {
             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<=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(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][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];
   }
             FileWrit(nt);
             return vnew[0][0][0];
}

double zlihcp::RunAll1(double P0) //
  {

   double TemRet = 0;
   Clear1();
   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,error1;
    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;
  error1=0.001;
  P0m=16.6159;
  zl.RunAll(P0m);
  //foutl4<<"LEAST SUM SQUARE "<<zl.LSS<<endl;
  foutl4<<"4 LEAST SUM SQUARE "<<zl.LSS_4<<endl;
  for(i=0;i<100;i++)
   {
   if(zl.TimeRec[i]!=-1)
    {
    if(zl.FlagRec[i] ==0)
     foutl4<<"Number "<<i<<" CoolTime "<<zl.TimeRec[i]<<endl;
    if(zl.FlagRec[i] ==1)
     foutl4<<"Number "<<i<<" HeatTime "<<zl.TimeRec[i]<<endl;
    if(zl.FlagRec[i] ==2)
    foutl4<<"Number "<<i<<" EndTime "<<zl.TimeRec[i]<<endl;
    }
  }
 do
  {
    P0m=Pnew;
   deltaP=P0m/100;
    S=Snew;
  T1m=zl.RunAll1(P0m);
  for (i=0;i<5;i++)
  rec0[i] = zl.point[i];
  T2m=zl.RunAll1(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);
  foutl4<<"PNEW "<<Pnew<<endl;
  foutl4<<"LEAST SQUARE SUM "<<Snew<<endl;
  }
 while ((Snew-S)/Snew > error1 );
 foutl4<<"END"<<endl;
 foutl4.close();
return 0;
}
```

APPENDIX B

SOURCE CODE FOR SOLVING THE 3D SKIN STURCTURE 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 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 BLOODTEMP 1
#define Bi 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,Rchange1,Phichange1,Zchange1;
     double p1,p2,p3,qc1,qc2,qc3,k1,k2,k3,wb1,wb2,wb3,cb1,cb2,cb3;
     double Sigma,Alpha1,Alpha2,Alpha3,Reff1,Reff2,Reff3;
     double P0, pi,judge,CenterX,CenterY ;
     double BSpeed,BP,BF,BAlpha,BCb,Bratio1,Bratio2,Uw1[NZ3-NZ2+1],Ub1[NZ3-NZ2+1];
     int i,j,z,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;j++)
     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];
    beta = new double **[l];
    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];
      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<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];
      vn[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;
        vn[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;
     Reff1=0.93;  Reff2=0.93; Reff3=0.93;
      pi=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;
      wb1=0.0;  wb2=0.0005; wb3=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);
 } ;

 ~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);
      void Vessel(void);
      double RunAll (double);
      void FileWrit(int);
      void Clear(void);
    };

    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*F*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 temerature
                */
                RFTw =vold[VR][NPhi/4][NZ3+1-RFi] +vold[VR][NPhi/2][NZ3+1-RFi]
                        +vold[VR][3*NPhi/4][NZ3+1-RFi]+vold[VR][0][NZ3+1-RFi];
                RFTw = RFTw/4;
                Uw1[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-1]+RFk1/2-RFTw));
                RFk3 = RFh*(Bratio1*Q3[0][0][NZ3-(RFi-1)]-Bratio2*(RFw[RFi-1]+RFk2/2-RFTw));
                RFk4 = RFh*(Bratio1*Q3[0][0][NZ3-(RFi-1)]-Bratio2*(RFw[RFi-1]+RFk3-RFTw));
                RFw[RFi] = RFw[RFi-1]+(RFk1+2*RFk2+2*RFk3+RFk4)/6;
                RFz = RFi*RFh;
              }
             // Receive the data from the function
                  for (RFi=0;RFi<=NZ3-NZ2-1;RFi++)
                      Ub1[RFi] = RFw[RFi];
                  Ub1[NZ3-NZ2] = RFw[NZ3-NZ2-1];
                  for (RFi=0;RFi<=VR-1;RFi++)
                    {
                     for (RFj=0;RFj<=NPhi;RFj++)
                       {
                        for (RFk=NZ2+1;RFk<=NZ3;RFk++)
                            vold[RFi][RFj][RFk] = Ub1[NZ3-RFk];
                       }
                    }
              }
  void zlihcp::FileWrit(int time1)
     {
      int i,k;  ofstream fout1,fout2,fout21,fout3,fout31;
      char str[20],str1[20]="zt",str2[20]="rzt",str21[20]="rztc",str3[20]="center",str31[20]="centerc";
      sprintf(str,"%d",time1);  strcat(str1,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 I=1209,F=POINT"<<endl;
 for(k=0;k<=NZ3;k++)
   fout1<<double(k*deltaZ)<<" "<<vnew[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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][0][k]<<endl;
  }
 fout2<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
 fout2<<" VARIABLES = \"R\", \"Z\", \"Temperature\" "<<endl;
 fout2<<" ZONE T=\"reverse\",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)<<" "<<vnew[i][NPhi/2][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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][NPhi/4][k]<<endl;
  }
 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)<<" "<<vnew[i][3*NPhi/4][k]<<endl;
  }
  fout21.close();
///////////////Center////////////////
 fout3.open(str3,ios::out);
 fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
 fout3<<" VARIABLES = \"R\", \"Temperature\" "<<endl;
 fout3<<" ZONE T=\"right\",I=31, F=POINT"<<endl;
 for(i=0;i<=NR;i++)
  fout3<<double(i*deltaR)<<" "<<vnew[i][0][0]<<endl;
 fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
 fout3<<" VARIABLES = \"R\", \"Temperature\" "<<endl;
 fout3<<" ZONE T=\"reverse\",I=31, F=POINT"<<endl;
 for(i=0;i<=NR;i++)
  fout3<<-double(i*deltaR)<<" "<<vnew[i][NPhi/2][0]<<endl;
 fout3.close();
 //////////////Center cross////////////
 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)<<" "<<vnew[i][NPhi/4][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)<<" "<<vnew[i][3*NPhi/4][0]<<endl;
 fout31.close();
}
```

```
void zlihcp::InitQ(double P0,double Cr,double Cp) // Initilize 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-Reff1 );
             }
            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-Reff2);
             }
            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-Reff3 );
             }
          }
       }
    }

double zlihcp::IntmTrsy(double P0) // Time Iteration and Tri-diagonal systme
    {
     MaxErr=1.0;  nt=0;
     while(++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]=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<=NPhi;j++)
          { ////////////////The first layer///////////////////////
           for(z=1;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==NPhi)
                 Phichange = k1*deltaT*(vold[i][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 //////////////
             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][1][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);
              ////////////Prepare the coefficient for Thomas way//////////////////////
           f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*qc1-wb1*cb1*deltaT)*vn[i][j][z];
           b[z]=(k1*deltaT)/(deltaZ*deltaZ);
           a[i][z]=2*p1*qc1+wb1*cb1*deltaT+k1*deltaT*(4*i+1)/(i*deltaR*deltaR)
                +k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+2*k1*deltaT/pow(deltaZ,2);
           c[z]=(k1*deltaT)/(deltaZ*deltaZ);
           d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                         k1*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
        }
     a[i][1]= a[i][1]-b[1];
     b[1] = 0;
     b[NZ1]=k1;
     a[i][NZ1]=k1+k2;
     c[NZ1]=k2;
     d[i][j][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][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////////////////////////
              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][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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][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[z]=k2*deltaT/(deltaZ*deltaZ);
              a[i][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                   +2*k2*deltaT/pow(deltaZ,2);
              c[z]=k2*deltaT/(deltaZ*deltaZ);
              d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                        k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
             }
            if (i>=VR+1)
             {
              b[NZ2]=k2;
              a[i][NZ2]=k2+k3;
              c[NZ2]=k3;
              d[i][j][NZ2]=0;
             }
            if((i<=VR)&&(z==NZ2-1))
             {
              a[i][NZ2-1]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                   +2*k2*deltaT/pow(deltaZ,2);
              c[NZ2-1]=0;
              d[i][j][NZ2-1]= f[i][j][z]+Rchange+Phichange+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)
                   +4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z]+Ub1[NZ3-NZ2]*k2*deltaT/(deltaZ*deltaZ);
             }
    /////////////The third layer ///////////////////////
     for(z=NZ2+1;z<=NZ3-1;z++)
     {  /////////////The n+1 state //////////////////////
```

```
                    if (i<=VR)
                     continue;
                    else
                     {
                      Rchange = k3*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 = k3*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                          }
                          else
                          {
                           Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/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*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
                         if(j==NPhi)
                          {
                          Phichange1 = k3*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                          }
                         else
                          {
                           Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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/////////////////////////
                         f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q3[i][j][z]
                                 +(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z]+2*wb3*cb3*deltaT*Ub1[NZ3-z];
                         b[z]=k3*deltaT/(deltaZ*deltaZ);
                         a[i][z]=2*p3*qc3+wb3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                              +2*k3*deltaT/pow(deltaZ,2);
                         c[z]=k3*deltaT/(deltaZ*deltaZ);
                         d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                                 k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
                       }
                   }
                  a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1];
                  c[NZ3-1]=0;
                 }
             }
// tri-diagonal system
            for(i=1;i<=NR-1;i++)
             {
             for(j=1;j<=NPhi;j++)
              {
                 v[i][j][NZ3]=0.0;
                 beta[i][j][NZ3]=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[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<=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];
                   }
                 }
               }
           }
       t++;  cout<<"number"<<t<<"  "<<"MaxErr"<<MaxErr<<endl;
       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[i][j][0] = vnew[i][j][1];
                  if(i>=VR+1)
                  vnew[i][j][NZ3] = vnew[i][j][NZ3-1];
                  vnew[i][0][z] = vnew[i][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];
                 if(i<=VR-1)
                  vnew[i][j][NZ2] = Ub1[NZ3-NZ2];
                 vnew[NR][j][z] = vnew[NR-1][j][z];
                 vold[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<<"temperation"<<vnew[0][0][0]<<endl;
    }
   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;
         }}}}

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, T1m, T2m, deltaP, X,Tpoint;
     long double S, Snew, Pnew,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)*(Tpoint-T1m);
     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 Rchange,Phichange,Zchange,Rchange1,Phichange1,Zchange1;
      double p1,p2,p3,qc1,qc2,qc3,k1,k2,k3,wb1,wb2,wb3,cb1,cb2,cb3;
      double Sigma,Alpha1,Alpha2,Alpha3,Reff1,Reff2,Reff3;
      double P0, pi,judge,CenterX,CenterY ;
      double BSpeed,BP,BF,BAlpha,BCb,Bratio2,Uw1[NZ3-NZ2+1],Ub1[NZ3-NZ2+1];
      int i,j,z,t,n ;
      int MaxLen,MaxWid,MaxHig,VR;
 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;
     a = new double *[l];
     for(j=0;j<l;j++)
      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];
     beta = new double **[l];
     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];
       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<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];
         vn[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;
          vn[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;
        Reff1=0.93;  Reff2=0.93; Reff3=0.93;
        pi=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;
        wb1=0.0;  wb2=0.0005;  wb3=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;
        Bratio2 = BAlpha*BP/(BCb*BSpeed*BF);
      };
```

```
  ~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 fout1,fout2,fout21,fout3,fout31;
    char str[20],str1[20]="zt",str2[20]="rzt",str21[20]="rztc",str3[20]="center",str31[20]="centerc";
    sprintf(str,"%d",time1);  strcat(str1,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 I=1209,F=POINT"<<endl;
    for(k=0;k<=NZ3;k++)
      fout1<<double(k*deltaZ)<<" "<<vnew[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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][0][k]<<endl;
     }
    fout2<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout2<<"  VARIABLES = \"R\", \"Z\", \"Temperature\" "<<endl;
    fout2<<" ZONE T=\"reverse\",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)<<" "<<vnew[i][NPhi/2][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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][NPhi/4][k]<<endl;
     }
    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)<<" "<<vnew[i][3*NPhi/4][k]<<endl;
     }
     fout21.close();
    ///////////////Center////////////////
    fout3.open(str3,ios::out);
    fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
    fout3<<" ZONE T=\"right\",I=31, F=POINT"<<endl;
    for(i=0;i<=NR;i++)
        fout3<<double(i*deltaR)<<" "<<vnew[i][0][0]<<endl;
    fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
    fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
    fout3<<" ZONE T=\"reverse\",I=31, F=POINT"<<endl;
    for(i=0;i<=NR;i++)
        fout3<<-double(i*deltaR)<<" "<<vnew[i][NPhi/2][0]<<endl;
    fout3.close();
   //////////////Center cross////////////
    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)<<" "<<vnew[i][NPhi/4][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)<<" "<<vnew[i][3*NPhi/4][0]<<endl;
    fout31.close();
 }

 void zlihcp::InitQ(double P0,double Cr,double Cp) // Initilize 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-Reff1 );
            }

           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-Reff2);
            }
          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-Reff3 );
         }
        }
     }
    }

 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*F*dTb/Dz = -Alpha*P*(Tb-Tw)
                The entry temperature is 10 centigrade.
              */
          int RFi,RFj,RFk;
          double RFw[NZ3-NZ2+1];
          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 temerature
                */
               RFTw =vold[VR][NPhi/4][NZ3+1-RFi]+vold[VR][NPhi/2][NZ3+1-RFi]
                      +vold[VR][3*NPhi/4][NZ3+1-RFi]+vold[VR][0][NZ3+1-RFi];
               RFTw = RFTw/4;
               Uw1[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;
      }
   // 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<=NPhi;RFj++)
            {
              for (RFk=NZ2;RFk<=NZ3;RFk++)
                vold[RFi][RFj][RFk] = Ub1[NZ3-RFk];
            }
          }
    }

    double zlihcp::IntmTrsy(double P0) // Time Iteration and Tri-diagonal systme
       {
        MaxErr=1.0; nt=0; MLSS=100000; MLSS1=100000;
        while(nt>-1)
         {  /////////////Change center point////////////////
           nt++;
           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]=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<=NPhi;j++)
             {
```

```
        ////////////////The first layer///////////////////////
    for(z=1;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==NPhi)
             {
             Phichange = k1*deltaT*(vold[i][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 //////////////
           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][1][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);
           /////////////Prepare the coefficient for Thomas way///////////////////////
          f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*qc1-wb1*cb1*deltaT)*vn[i][j][z];
          b[z]=(k1*deltaT)/(deltaZ*deltaZ);
          a[i][z]=2*p1*qc1+wb1*cb1*deltaT+k1*deltaT*(4*i+1)/(i*deltaR*deltaR)
                  +k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+2*k1*deltaT/pow(deltaZ,2);
          c[z]=(k1*deltaT)/(deltaZ*deltaZ);
          d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                      k1*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
     }
     a[i][1]= a[i][1]-b[1];
     b[1] = 0;
     b[NZ1]=k1;
     a[i][NZ1]=k1+k2;
     c[NZ1]=k2;
     d[i][j][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][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////////////////////////
            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][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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][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[z]=k2*deltaT/(deltaZ*deltaZ);
                a[i][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                    +2*k2*deltaT/pow(deltaZ,2);
                c[z]=k2*deltaT/(deltaZ*deltaZ);
                d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                        k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
                }
        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
         z = NZ2-1;
         a[i][NZ2-1]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                    +2*k2*deltaT/pow(deltaZ,2);
         c[NZ2-1]=0;
         d[i][j][NZ2-1]= f[i][j][z]+Rchange+Phichange+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)
                +4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z]+Ub1[NZ3-NZ2]*k2*deltaT/(deltaZ*deltaZ);
         }
      //////////////The third layer /////////////////////////
        for(z=NZ2+1;z<=NZ3-1;z++)
        { //////////////The n+1 state ////////////////////////
           if (i<=VR)
            continue;
          Rchange = k3*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 = k3*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                 }
                else
                 {
                  Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/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*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
                if(j==NPhi)
                 {
                 Phichange1 = k3*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                 }
                else
                 {
                  Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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///////////////////////
                f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q3[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z]
                                                +2*wb3*cb3*deltaT*Ub1[NZ3-z];
                b[z]=k3*deltaT/(deltaZ*deltaZ);
                a[i][z]=2*p3*qc3+wb3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                    +2*k3*deltaT/pow(deltaZ,2);
                c[z]=k3*deltaT/(deltaZ*deltaZ);
                d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                        k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
              }
        a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1];
         c[NZ3-1]=0;
        }
      }
  // tri-diagonal system
        for(i=1;i<=NR-1;i++)
         {
         for(j=1;j<=NPhi;j++)
          {
```

```
    v[i][j][NZ3]=0.0;
    beta[i][j][NZ3]=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[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<=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];
            }
         }
       }
     }
   t++; cout<<"number"<<t<<"  "<<"MaxErr"<<MaxErr<<endl;
  //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[i][j][0] = vnew[i][j][1];
          if(i>=VR+1)
            vnew[i][j][NZ3] = vnew[i][j][NZ3-1];
          vnew[i][0][z] = vnew[i][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];
          if(i<=VR-1)
            vnew[i][j][NZ2] = Ub1[NZ3-NZ2];
          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];
        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);
        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);
         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,20*deltaPhi);
           if(LSS_4<LSS4)
            {
            TimeRec[CountNum] = nt;
            FlagRec[CountNum] = flag;
            CountNum ++;
            flag =2;
            goto BT;
            }
           if((point[1]>EndTemp)||(point[2]>EndTemp)||(point[3]>EndTemp)||(point[4]>EndTemp)||(point[0]<EndTemp))
            {
            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;
              goto BT;
             }
            if(point[0]>CenTemp)
             {
              TimeRec[CountNum] = nt;
              FlagRec[CountNum] = flag;
              CountNum ++;
              FileWrit(nt);
              flag =0;
             }
           }
           BT:
           if(flag == 2) //stop heating
            {
           InitQ(P0,0*deltaR,0*deltaPhi);
           if(point[0]>CenTemp)
            {
             TimeRec[CountNum] = nt;
             FlagRec[CountNum] = flag;
             goto loopend;
             }
           }
          }
        }
               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;
          }
        }
       }
}

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;
    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 "<<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;
       }
    }
    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<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 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 Rchange,Phichange,Zchange,Rchange1,Phichange1,Zchange1;
    double p1,p2,p3,qc1,qc2,qc3,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,z,t,n ;
    int MaxLen,MaxWid,MaxHig,VR;
    double BSpeed,BP,BF,BAlpha,BCb,Bratio2,Uw1[NZ3-NZ2+1],Ub1[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;
    a = new double *[l];
    for(j=0;j<l;j++)
     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];
    vsave = new double **[l];
    beta = new double **[l];
    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];
       vnew[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 [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];
         vn[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;
        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;
       Reff1=0.93;  Reff2=0.93; Reff3=0.93;
       pi=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;
       wb1=0.0;  wb2=0.0005;  wb3=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;
       Bratio2 = BAlpha*BP/(BCb*BSpeed*BF);
};
```

```
~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);
    double IntmTrsy1(double);
    double RunAll1 (double);
    void Clear1(void);
};

  void zlihcp::FileWrit(int time1)
     {
      int i,k;  ofstream fout1,fout2,fout21,fout3,fout31;
      char str[20],str1[20]="zt",str2[20]="rzt",str21[20]="rztc",str3[20]="center",str31[20]="centerc";
      sprintf(str,"%d",time1);  strcat(str1,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 I=1209,F=POINT"<<endl;
      for(k=0;k<=NZ3;k++)
        fout1<<double(k*deltaZ)<<" "<<vnew[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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][0][k]<<endl;
       }
      fout2<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
      fout2<<"  VARIABLES = \"R\", \"Z\", \"Temperature\" "<<endl;
      fout2<<" ZONE T=\"reverse\",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)<<" "<<vnew[i][NPhi/2][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<<double(i*deltaR)<<" "<<double(k*deltaZ)<<" "<<vnew[i][NPhi/4][k]<<endl;
       }
      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)<<" "<<vnew[i][3*NPhi/4][k]<<endl;
       }

       fout21.close();
      ///////////////Center////////////////
      fout3.open(str3,ios::out);
      fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
      fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
      fout3<<" ZONE T=\"right\",I=31, F=POINT"<<endl;
      for(i=0;i<=NR;i++)
        fout3<<double(i*deltaR)<<" "<<vnew[i][0][0]<<endl;
      fout3<<" TITLE = \"Example: Simple 2D-Volume Data\" "<<endl;
      fout3<<"  VARIABLES = \"R\", \"Temperature\" "<<endl;
      fout3<<" ZONE T=\"reverse\",I=31, F=POINT"<<endl;
      for(i=0;i<=NR;i++)
        fout3<<-double(i*deltaR)<<" "<<vnew[i][NPhi/2][0]<<endl;
      fout3.close();
     //////////////Center cross////////////
      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)<<" "<<vnew[i][NPhi/4][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)<<" "<<vnew[i][3*NPhi/4][0]<<endl;
      fout31.close();
     }

void zlihcp::InitQ(double P0,double Cr,double Cp) // Initilize 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-Reff1 );
             }
            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-Reff2);
              }
          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-Reff3 );
        }
       }
       }
     }

  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*F*dTb/Dz = -Alpha*P*(Tb-Tw)
    The entry temperature is 1 centigrade.
  */
  int RFi,RFj,RFk;
  double RFw[NZ3-NZ2+1];

          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 temerature
           */
            RFTw =vold[VR][NPhi/4][NZ3+1-RFi] +vold[VR][NPhi/2][NZ3+1-RFi]
                      +vold[VR][3*NPhi/4][NZ3+1-RFi]+vold[VR][0][NZ3+1-RFi];
            RFTw = RFTw/4;
            Uw1[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;
         }
// 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<=NPhi;RFj++)
          {
            for (RFk=NZ2;RFk<=NZ3;RFk++)
           vold[RFi][RFj][RFk] = Ub1[NZ3-RFk];
          }
        }
  }

double zlihcp::IntmTrsy(double P0) // Time Iteration and Tri-diagonal systme
    {
     MaxErr=1.0; nt=0;
     while(nt>-1)
      {
        nt++;
         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]=vold[i][j][z];
              vsave[i][j][z]=vold[i][j][z];
             }
            }
          }
       cout<<nt<<"new cicle"<<endl;
       MaxErr=1.0;
       while(MaxErr>=e)
        {
         Vessel();
```

```
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 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==NPhi)
           {
           Phichange = k1*deltaT*(vold[i][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 ///////////////
         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][1][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);
          ////////////Prepare the coefficient for Thomas way//////////////////////
        f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*qc1-wb1*cb1*deltaT)*vn[i][j][z];
        b[z]=(k1*deltaT)/(deltaZ*deltaZ);
        a[i][z]=2*p1*qc1+wb1*cb1*deltaT+k1*deltaT*(4*i+1)/(i*deltaR*deltaR)
               +k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+2*k1*deltaT/pow(deltaZ,2);
        c[z]=(k1*deltaT)/(deltaZ*deltaZ);
        d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                    k1*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
     }
    a[i][1]= a[i][1]-b[1];
     b[1] = 0;
     b[NZ1]=k1;
     a[i][NZ1]=k1+k2;
     c[NZ1]=k2;
     d[i][j][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][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/////////////////////////
          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][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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][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[z]=k2*deltaT/(deltaZ*deltaZ);
                    a[i][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                        +2*k2*deltaT/pow(deltaZ,2);
                    c[z]=k2*deltaT/(deltaZ*deltaZ);
                    d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                           k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
                    }
            if (i>=VR+1)
              {
               b[NZ2]=k2;
               a[i][NZ2]=k2+k3;
               c[NZ2]=k3;
               d[i][j][NZ2]=0;
              }
             if(i<=VR)
              {
               z = NZ2-1;
              a[i][NZ2-1]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                        +2*k2*deltaT/pow(deltaZ,2);
              c[NZ2-1]=0;
              d[i][j][NZ2-1]= f[i][j][z]+Rchange+Phichange+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)
                    +4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z]+Ub1[NZ3-NZ2]*k2*deltaT/(deltaZ*deltaZ);
              }
        //////////////The third layer ////////////////////////
            for(z=NZ2+1;z<=NZ3-1;z++)
            { //////////////The n+1 state ///////////////////////
               if (i<=VR)
                 continue;
              Rchange = k3*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 = k3*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                    else
                     Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/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*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
                    if(j==NPhi)
                     Phichange1 = k3*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
                    else
                     Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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///////////////////////
                    f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q3[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z]
                                                       +2*wb3*cb3*deltaT*Ub1[NZ3-z];
                    b[z]=k3*deltaT/(deltaZ*deltaZ);
                    a[i][z]=2*p3*qc3+wb3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                        +2*k3*deltaT/pow(deltaZ,2);
                    c[z]=k3*deltaT/(deltaZ*deltaZ);
                    d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                          k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
              }
             a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1];
             c[NZ3-1]=0;
            }
        }
   // tri-diagonal system
          for(i=1;i<=NR-1;i++)
            {
            for(j=1;j<=NPhi;j++)
             {
              v[i][j][NZ3]=0.0;
              beta[i][j][NZ3]=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[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<=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];
                   }
                 }
              }
            }
          t++; cout<<"number"<<t<<"  "<<"MaxErr"<<MaxErr<<endl;
         //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[i][j][0] = vnew[i][j][1];
                   if(i>=VR+1)
                     vnew[i][j][NZ3] = vnew[i][j][NZ3-1];
                   vnew[i][0][z] = vnew[i][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];
                  if(i<=VR-1)
                    vnew[i][j][NZ2] = Ub1[NZ3-NZ2];
                  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];
        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);
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);
  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,20*deltaPhi);
   if(LSS_4<LSS4)
    {
    TimeRec[CountNum] = nt;
    FlagRec[CountNum] = flag;
    CountNum ++;  goto loopend;
    }
   if((point[1]>EndTemp)||(point[2]>EndTemp)||(point[3]>EndTemp)||(point[4]>EndTemp)||(point[0]<EndTemp))
    {
    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];
  }

 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::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]=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 zlihcp::IntmTrsy1(double P0) // Time Iteration and Tri-diagonal systme
  {
   MaxErr=1.0;
   while(nt<=T2) //need the center temp back to 8
    {
      nt++;
      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]=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<=NPhi;j++)
         { ////////////////The first layer///////////////////////
           for(z=1;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==NPhi)
                 {
                   Phichange = k1*deltaT*(vold[i][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 //////////////
                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][1][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);
                 ////////////Prepare the coefficient for Thomas way///////////////////////
               f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q1[i][j][z]+(2*p1*qc1-wb1*cb1*deltaT)*vn[i][j][z];
               b[z]=(k1*deltaT)/(deltaZ*deltaZ);
               a[i][z]=2*p1*qc1+wb1*cb1*deltaT+k1*deltaT*(4*i+1)/(i*deltaR*deltaR)
                      +k1*deltaT*4/pow(deltaPhi*i*deltaR,2)+2*k1*deltaT/pow(deltaZ,2);
               c[z]=(k1*deltaT)/(deltaZ*deltaZ);
               d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                         k1*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
              }
   a[i][1]= a[i][1]-b[1];
   b[1] = 0;
               b[NZ1]=k1;
               a[i][NZ1]=k1+k2;
               c[NZ1]=k2;
               d[i][j][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][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/////////////////////////
            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][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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][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[z]=k2*deltaT/(deltaZ*deltaZ);
            a[i][z]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                +2*k2*deltaT/pow(deltaZ,2);
            c[z]=k2*deltaT/(deltaZ*deltaZ);
            d[i][j][z]= f[i][j][z]+Rchange+Phichange+
                      k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
          }
      if (i>=VR+1)
       {
         b[NZ2]=k2;
         a[i][NZ2]=k2+k3;
         c[NZ2]=k3;
         d[i][j][NZ2]=0;
       }
      if(i<=VR)
       {
         z = NZ2-1;
        a[i][NZ2-1]=2*p2*qc2+wb2*cb2*deltaT+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
               +2*k2*deltaT/pow(deltaZ,2);
         c[NZ2-1]=0;
         d[i][j][NZ2-1]= f[i][j][z]+Rchange+Phichange+k2*deltaT*((4*i+1)/(i*deltaR*deltaR)
               +4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z]+Ub1[NZ3-NZ2]*k2*deltaT/(deltaZ*deltaZ);
     }
    //////////////The third layer //////////////////////////
    for(z=NZ2+1;z<=NZ3-1;z++)
    { //////////////The n+1 state //////////////////////////
          Rchange = k3*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 = k3*deltaT*(vold[i][1][z]-2*vold[i][j][z]+vold[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
            else
             Phichange = k3*deltaT*(vold[i][j+1][z]-2*vold[i][j][z]+vold[i][j-1][z])/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*i*vn[i][j][z]+(i-0.5)*vn[i-1][j][z])/(i*deltaR*deltaR);
            if(j==NPhi)
             Phichange1 = k3*deltaT*(vn[i][1][z]-2*vn[i][j][z]+vn[i][j-1][z])/pow(i*deltaR*deltaPhi,2);
            else
             Phichange1 = k3*deltaT*(vn[i][j+1][z]-2*vn[i][j][z]+vn[i][j-1][z])/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//////////////////////
            f[i][j][z]=Rchange1+Phichange1+Zchange1+2*deltaT*Q3[i][j][z]+(2*p3*qc3-wb3*cb3*deltaT)*vn[i][j][z]
                                          +2*wb3*cb3*deltaT*Ub1[NZ3-z];
            b[z]=k3*deltaT/(deltaZ*deltaZ);
            a[i][z]=2*p3*qc3+wb3*cb3*deltaT+k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))
                +2*k3*deltaT/pow(deltaZ,2);
            c[z]=k3*deltaT/(deltaZ*deltaZ);
            d[i][j][z]=f[i][j][z]+Rchange+Phichange+
                    k3*deltaT*((4*i+1)/(i*deltaR*deltaR)+4/pow(deltaPhi*i*deltaR,2))*vold[i][j][z];
          }
    a[i][NZ3-1]=a[i][NZ3-1]-c[NZ3-1];
    c[NZ3-1]=0;
   }
 }
// tri-diagonal system
```

```
        for(i=1;i<=NR-1;i++)
          {
           for(j=1;j<=NPhi;j++)
             {
              v[i][j][NZ3]=0.0;
              beta[i][j][NZ3]=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[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<=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];}}}}
        t++;  cout<<"number"<<t<<"  "<<"MaxErr"<<MaxErr<<endl;
//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[i][j][0] = vnew[i][j][1];
                     if(i>=VR+1)
                       vnew[i][j][NZ3] = vnew[i][j][NZ3-1];
                     vnew[i][0][z] = vnew[i][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];
                   if(i<=VR-1)
                     vnew[i][j][NZ2] = Ub1[NZ3-NZ2];
                   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];
}
   FileWrit(nt);
   return vnew[0][0][0];
}

double zlihcp::RunAll1(double P0) //
   {
    double TemRet = 0;  Clear1();
```

```
  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,error1;
   double rec0[5],rec1[5],X[5],Scale1,Scale2;
   double Tpoint=CenTemp,Tpointa=EndTemp;
   int i;  ofstream foutl4;  foutl4.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);
  foutl4<<"4 LEAST SUM SQUARE "<<zl.LSS_4<<endl;
   for(i=0;i<100;i++)
     {
     if(zl.TimeRec[i]!=-1)
      {
      if(zl.FlagRec[i] ==0)
       foutl4<<"Number "<<i<<" CoolTime "<<zl.TimeRec[i]<<endl;
      if(zl.FlagRec[i] ==1)
        foutl4<<"Number "<<i<<" HeatTime "<<zl.TimeRec[i]<<endl;
      if(zl.FlagRec[i] ==2)
        foutl4<<"Number "<<i<<" EndTime "<<zl.TimeRec[i]<<endl;}}
   do
    {
      P0m=Pnew;  deltaP=P0m/100;
      S=Snew;  T1m=zl.RunAll1(P0m);
     for (i=0;i<5;i++)
      rec0[i] = zl.point[i];
    T2m=zl.RunAll1(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);
    foutl4<<"PNEW "<<Pnew<<endl;
    foutl4<<"LEAST SQUARE SUM "<<Snew<<endl;
    }
    while ((Snew-S)/Snew > error1 );
   foutl4<<"END"<<endl;
   foutl4.close();
   return 0;
 }
```

REFERENCES

[Beck 1977] J. V. Beck and K. J. Arnold, "Parameter Estimation in Engineering and Science," New York: John Wiley and Sons, Inc., 1977.

[Beck 1985] J. V. Beck, B. Blackwell, C.R. St. Clair, Jr., "Inverse Heat Conduction: Ill-posed Problems," New York: John Wiley and Sons, Inc., 1985.

[Bejan 2000] A. Bejan, "The tree of convective heat streams: its thermal insulation function and the predicted ¾-power relation between body heat loss and body size," Internal Journal of Heat and Mass Transfer, vol. 44, pp. 699-704, 2001

[Chatterjee 1994] I. Chatterjee and R. A. Adams, "Finite element thermal modeling of the human body under hyperthermia treatment for cancer," Int. J. of Computer Applications in Technology., vol. 7, pp. 151-159, 1994.

[Clegg 1989] S. T. Clegg and R. B. Roemer, "Predictions of three-dimensional temperature distributions during hyperthermia experiments," ASME, Heat Transfer Division., vol. 126, pp. 29-35, 1989.

[Dai 1998] W. Dai and R. Nassar, “A three-dimensional numerical method for thermal analysis in X-ray lithography,” International Journal of Numerical Methods for Heat & Fluid Flow, vol. 8, No.4, pp.409-423, 1998.

[Dai 2003a] W. Dai, Q. Li, R. Nassar and T. Zhu, “A domain decomposition method for solving the Pennes' bioheat transfer in a 3d triple-layered skin structure,” Proceedings of the Second M.I.T. Conference on Computational Fluid and Solid Mechanics, MIT, Boston, June 17-20, vol. 2, pp. 1650-1659, 2003.

[Dai 2003b] W. Dai, H. Yu, R. Nassar and T. Zhu, “A fourth-order compact finite difference scheme for solving a 1-D Pennes' bioheat transfer equation in a uniform tissue,” Proceedings of the 2003 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences, Las Vegas, Nevada, June 23-26, pp. 336-339, 2003.

[Gartner 2000] L. P. Gartner, “Color atlas of histology,” 3th ed., Philadelphia: Lippincott, Williams&Wilkins, 2000.

[Ham 1965] A. W. Ham, “Histology,” 5th ed., Philadelphia: Lippincott, Williams&Wilkins , 1965.

[Hall 1984] E. J. Hall and L. Roizin-Towle, “Biological effects of heat,” Cancer Res., vol. 44, pp. 4708-4713, 1984.

[Hensel 1991] E. Hensel, "Inverse Theory and Applications for Engineers," Englewood Cliffs, NJ: Prentice-Hall, 1991.

[Huang 1994] H. W. Huang, C.L. Chan and R.B. Roemer, "Analytical solutions of Pennes bioheat transfer equation with a blood vessel," Journal of Biomechanical Engineering, vol. 116, pp. 208-212, 1994.

[Jaesung 1994] H. Jaesung and F.J. Klavs, "Combined experimental and modeling studies of laser-assisted chemical vapor depositio of copper from copper (I)-hexafluoroacetylacetonate trimethylvinylsilane," J. Appl. Phys., vol. 75, pp. 2240-2250, 1994.

[Lorimer 1999]L. T. Lorimer, "The Human Body," New York: Reader's Digest Children's Publishing, Inc., 1999.

[Li 1979] R. Li and G. Hong, "Numerical Solutions for Partial Differential Equations," People's Educational Publisher, Peking (Chinese), 1979.

[Liu 1995] J. Liu, Z. Ren, C. Wang, "Interpretation of living tissue's temperature oscillations by thermal wave theory," Chinese Sci. Bull., vol. 40, pp.1493-1495, 1995.

[Liu 1997] J. Liu. X. Zhang and C. Wang, "Generalized Time delay Bioheat equation and preliminary Analysis on its Wave. Nature," Chinese Sci Bull., vol. 42, pp 289-292, 1997.

[Liu 1999] J. Liu, X. Chen and L. X. Xu, "New thermal wave aspects on burn evaluation of skin subjected to instantaneous heating," IEEE Transaction on Biomedical Engineering, vol. 46, pp. 420-428, 1999.

[Liu 2000a] J. Liu, "Preliminary survey on the mechanisms of the wave-like behaviors of heat transfer in living tissues," Forschung im Ingenieurwesenm., vol. 66, pp. 1-10, 2000.

[Liu 2000b] J. Liu and L. X. Xu, "Boundary information based diagnostics on the thermal states of biological bodies," Int. J. of Heat and Mass Transfer, vol. 43, pp. 2827-2839, 2000.

[Lu 1998] W. Q. Lu, J. Liu and Y. Zeng, "Simulation of the thermal wave propagation in biological tissues by the dual reciprocity boundary element method," Engineering Analysis with Boundary Elements, vol. 22, pp. 167-174, 1998.

[Majchrzak 1999] E. Majchrzak and B. Mochnacki, "Numerical model of heat transfer between blood vessel and biological tissue," Computer Assisted Mechanics and Engineering Sciences, vol. 6, pp. 439-447, 1999.

[Martin 1989] G. T. Martin and F. Bowman, "The temperature distribution in laser irradiated tissue with blood perfusion," ASME, Heat Transfer Division, vol. 126, pp. 97-102, 1989.

[Moroz 2002] P. Moroz, S. K. Jones and B. N. Gary, “Magnetically mediated hyperthermia: current status and future directions,” Int. J. Hyperthermia., vol. 18, pp. 267-284, 2002.

[Muralidharan 2002] V. Muralidharan, C. Malcontenti-Wilson and C. Cristophi, “Interstitial laser hyperthermia for colorectal liver metastases: the effect of thermal sensitization and the use of a cylindrical diffuser tip on tumor necrosis,” J. Clin. Laser Med. Surg., vol. 20, pp. 189-196, 2002.

[Ozisik 1993] M.N.Ozisik, “Heat Conduction,” 2nd ed., New York: John & Wiley, 1993, Chap.14

[Payne 1999] A. Payne, M. Mattingly, R. B. Roemer and E. P. Scott, “A model for a thin layer phantom with application to hyperthermia cancer therapy,” Bioengineering Conference, ASME., vol. 42, pp. 197-198, 1999.

[Pennes 1948] H. H. Pennes, “Analysis of tissue and arterial temperature in the resting human forearm,” J. Appl. Physiol., vol. 1, pp. 93-122, 1948.

[Roemer 1989] R. B. Roemer, E. G. Moros and K. Hynynen, “A comparison of bioheat transfer and effective conductivity equation predictions to experimental hyperthermia data,” ASME, Heat Transfer Division, vol. 126, pp. 11-15, 1989.

[Roemer 1991] R. B. Roemer, "Optimal power deposition in hyperthermia I. The treatment goal: the ideal temperature distribution: the role of large blood vessels," ASME, Heat Transfer Division, vol. 7, pp. 317-341, 1991.

[Silva 2004] A. K. da Silva, S. Lorente and A. Bejan, "Constructal Multi-Scale Tree-Shaped Heat Exchangers", Journal of Applied Physics, vol. 96(3), pp. 1709-1718, Aug. 2004.

[Steffer 1987] C. Streffer, "Biological basis for the use of hyperthermia in tumor therapy," Strahlentherapie und Onkologie, vol. 163, pp. 416-419, 1987.

[Tikhonov 1977] A. N. Tikhonov and Y. Arsenin, "Solutions of Ill-Posed Problems," Winston &Sons, Washington, DC, 1977.

[Tsuda 1996] N. Tsuda and K. Kuroda, "An inverse method to optimize heating conditions in RF-capacitive hyperthermia," IEEE Transaction on Biomedical Engineering, vol. 43, pp. 1029-1037, 1996.

[Usatoff 2001] V. Usatoff and N. A. Habib, "Update of laser-induced thermotherapy for liver tumors," Hapatogastroenterology, vol. 48, pp. 330-332, 2001.

[Waldow 1988] S. Waldow, P. Morrison, and L. Grossweiner, "Nd:YAG laser-induced hyperthermia in a mouse tumor model," Lasers Surg. Med., vol 8, pp. 510-514, 1988.

[Wang 1992] M-J. Wang, J.O Naim, D. W. Rogers and R.J. Lanzafame, "The effect of Nd:YAG laser-induced hyperthermia on local tumor recurrence in experimental rat mammary tumors," Journal of Clinical Medical & Surgery, vol. 10, pp. 265-272, 1992.

[Wust 2002] P. Wust, B. Hildebrandt, et al., "Hyperthermia in combined treatment of cancer," Lancet Oncol., vol. 3, pp. 487-497, 2002.

[Zhang 2005] L. Zhang, W. Dai, R. Nassar, "A numerical modeling for optimizing laser power irradiating on a 3D triple layered cylindrical skin structure," Numerical Heat Transfer, Part A, to be published.

[Zhou 2004] J. H. Zhou and J. Liu, "Numerical study on 3-D light and heat transport in biological tissues embedded with large blood vessels during laser-induced thermotherapy," Numerical Heat Transfer, Part A, vol. 45, pp. 415-499, 2004.