Winter 2011

Vascular countercurrent network for 3D triple-layered skin structure with radiation heating

Xiaoqi Zeng

VASCULAR COUNTERCURRENT NETWORK FOR 3D TRIPLE-LAYERED SKIN STRUCTURE WITH RADIATION HEATING

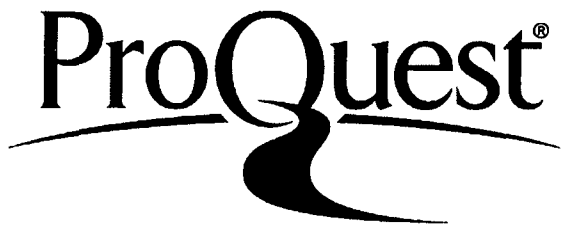
by

Xiaoqi Zeng, B.S., M.S.

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

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

February 2011



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

October 15, 2010

Date

We hereby recommend that the dissertation prepared under our supervision by Xiaoqi Zeng

entitled

VASCULAR COUNTERCURRENT NETWORK FOR 3D TRIPLE-LAYERED SKIN STRUCTURE WITH RADIATION HEATING

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Supervisor of Dissertation Research

Head of Department

Computational Analysis and Modeling

Department

Recommendation concurred in:

Advisory Committee

Approved:

Director of Graduate Studies

Approved:

Dean of the Graduate School

Dean of the College

GS Form 13a
(6/07)

ABSTRACT

Heat transfer in living tissue has become more and more attention for researchers, because high thermal radiation produced by intense fire, such as wild fires, chemical fires, accidents, warfare, terrorism, etc, is often encountered in human's daily life. Living tissue is a heterogeneous organ consisting of cellular tissue and blood vessels, and heat transfer in cellular tissue and blood vessel is quite different, because the blood vessels provide channels for fast heat transfer. The metabolic heat generation, heat conduction and blood perfusion in soft tissue, convection and perfusion of the arterial-venous blood through the capillary, and interaction with the environment should be also considered in heat transfer in living tissue. To understand the effect of high thermal radiation on biological tissues, specifically, thermo-mechanical damage to the tissue, a mathematical model for skin injury induced by radiation heating has been developed by W. Dai et al. in 2008 [19], where the skin was considered to be a 3D triple-layered structure with an embedded three-level dendritic countercurrent vascular network.

Since there are up to seven layers of blood vessels in the skin tissues [25], the motivation of this dissertation research is to extend the mathematical model developed by W. Dai et al. in 2008 [19] to the case that considers a seven-level dendritic countercurrent vascular network, where the dimensions and blood flow of the blood vessels are determined based on the constructal theory of multi-scale tree-shaped heat exchangers. As such, the number of the blood vessels is increased from eight to one

hundred and twenty eight. This makes the computation much more complicated. To this end, blood flow oriented coordinates system was first designed, so that a simple energy equation in blood vessels can be obtained and solved using the fourth-order Runge-Kutta method. Coupled with the mathematical model and numerical schemes developed in W. Dai's paper [19], the temperature distribution in a living skin tissue embedded with a seven layered dendritic countercurrent vascular network is able to be predicted, and hence the skin burn injury induced by radiation heating can also be predicted. Furthermore, the numerical scheme is proved to be unconditional stable and the Preconditioned Richardson iteration developed for the computation is convergent. Unconditional stable scheme (no restriction on mesh ratio) is particularly important in this research since the thickness of the first layer of the skin structure is small and hence the grid size in the thickness direction can be very small. The developed Precondition Richardson iteration allows us to transform a complicated solution system to a tridiagonal linear system, so that the conventional Thomas Algorithm can be easily used, and hence the computation cost can be reduced.

Numerical results show that there is no difference between the current study and the previous study regarding the area of the high degree burn injury. However, the areas of the first and second degree burn injury are different from those obtained in W. Dai's paper [19], because of the more complex countercurrent vascular network that is used in the present model. The obtained model and numerical method in this dissertation could be used in future studies: e.g., by considering a larger area of skin structure with complicated dendritic countercurrent multi-level blood vessels, as well as modeling such well documented effects of thermal damage as skin wrinkles and tissue shrinkage.

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 Yiwei Zeng

Date 2/18/2011

DEDICATION

I dedicate this dissertation to my parents, my sister and my son, Max. Especially to my beloved one, even though she does not have a name, gives me loyal, noble, pure and unique love!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I am grateful to the people who have helped me to fulfill this dissertation. Especially, to my advisor, Dr. Weizhong Dai, for his generous advice and guidance with great patience. It is my honor to be his student. Without his guidance and advice, this dissertation could not have been completed. I would like to thank Dr. Don Liu for his support and for his Partial Differential Equation class. Sincere acknowledgement is also extended to Dr. Sumeet Dua, for his Data Mining in bio-informatics course. Also thanks to Drs. Songming Hou, and Katie Evans for their kindness of serving as advisory committee members.

Many thanks to Dr. Terry M. McConathy, Executive Vice President & Graduate School Dean, for her real leadership ability and dedication to the education and research at Louisiana Tech University.

I owe a lot to my sister, Xiaoqin Zeng, for her supporting me and having done her best to provide me comfortable conditions for living and studying. Finally, I want to express my appreciation to all my friends. This dissertation is dedicated to all the above mentioned.

CHAPTER ONE

INTRODUCTION

1.1 General Overview

Heat transfer in living tissue has become more and more attention for researchers, because high thermal radiation produced by intense fire, such as wild fires, chemical fires, accidents, warfare, terrorism, etc, is often encountered in human's daily life. Living tissue is a heterogeneous organ consisting of cellular tissue and blood vessels, and heat transfer cellular tissue and blood vessel is quite different, because the blood vessels provide channels for fast heat transfer. The metabolic heat generation, heat conduction and blood perfusion in soft tissue, convection and perfusion of the arterial-venous blood through the capillary, and interaction with the environment should be also considered in heat transfer in living tissue.

Recently, a mathematical model for skin injury induced by radiation heating has been developed by W. Dai et al. [19]. The model was obtained by modifying the Pennes equation [41] and by taking into account the thermal relaxation time of biological tissue, where the skin was considered to be a 3D triple-layered structure with embedded three-level dendritic countercurrent vascular network. The Fourier's Law and the governing equation of heat transfer in biological tissue are stated as follows:

$$\vec{q} = -k\nabla T, \tag{1.1}$$

$$\rho C \frac{\partial T}{\partial t} = -k\nabla \cdot \vec{q} + W_b C_b (T_b - T) + Q, \tag{1.2}$$

where $\vec{q}$, T and k denote the thermal flux vector, tissue temperature and thermal conductivity, respectively, W_b is the blood perfusion rate; C_b is the specific heat of blood; T_b is the blood temperature; and Q is the volumetric heat.

After replacing $\vec{q}$ in Eq. (1.2) with Eq. (1.1), the Pennes equation [41] is obtained and now is widely known as the conventional bio-heat transfer equation,

$$\rho C \frac{\partial T}{\partial t} = k\Delta T + W_b C_b (T_b - T) + Q. \tag{1.3}$$

As pointed out by Dai and his colleagues in their paper [19], as early as in1990s, scientists discovered that living tissue along with a number of other common materials, exhibited a relatively long thermal relaxation (or lag) time (denoted by τ). Quantitatively, once a temperature gradient has been imposed across a material volume element, τ, which could range up to 100 s, represents the time required to establish steady thermal conduction in the domain considered. The implication of such large values of τ is that Fourier's law is generally not used to describe the heat conduction in organic media, the classic expression for the thermal flux vector $\vec{q}$, instead the Maxwell-Cattaneo flux law,

$$(1+\tau\frac{\partial}{\partial t})\vec{q} = -\mathrm{k}\nabla T, \tag{1.4}$$

is widely used [8]. By substituting $\vec{q}$ into Eq. (1.2), it obtains the modified Pennes equation as the governing equation of heat transfer in living tissue,

$$\rho C \left(\frac{\partial T}{\partial t} + \tau\frac{\partial^2 T}{\partial t^2}\right) + \tau W_b C_b \frac{\partial T}{\partial t} = k\Delta T + W_b C_b (T_b - T) + Q, \tag{1.5}$$

which is a hyperbolic type partial differential equation. Thus, Eq. (1.4) –(1.5) predict that heat conduction in such media occurs in a wave-like manner, a phenomena now is known as "second sound" [11], and not by diffusion.

In Dai et al's paper [19], the skin was considered to be a 3D triple-layered structure embedded with three level dendrical countercurrent vascular network, where the dimensions and blood flow rates of the multi-level blood vessels were determined based on the recently developed constructal theory of multi-scale tree-shaped heat exchangers [2], [3], [43], [4].

1.2 Research Objectives

Since there are up to seven levels of blood vessels in the skin tissue pointed out by R. B. Roemer and his colleagues in 1996 [42], the objective of this dissertation is to extend the mathematical and its numerical scheme developed by Dai and his colleagues in 2008 [19] to a more complex architecture consisting of a hierarchical seven-level countercurrent vascular network embedded in a 3D triple-layered skin tissue. The seven-level countercurrent vascular network proposed in this dissertation is different from the one considered in R. B. Roemer's paper [42], because the dimensions and blood flow rates of the multi-level blood vessels are determined (predicted) based on the constructal theory of multi-scale tree-shaped heat exchangers [2], [3], [43], [4].

To achieve this objective, the following steps will be followed:

Step 1. Consider a 3D triple-layered skin tissue structure in Cartesian coordinates domain, and design the diameters and lengths of the seven level of countercurrent blood vessels, arteries and veins, which is based on the constructal theory of multi-scale tree-shaped heat exchangers.

Step 2. Set up the governing equations, including Maxwell-Cattaneo flux law which takes into account the thermal relaxation time of biological tissue, the modified Pennes equation for skin tissue coupled with the fourth power law for radiation

heating and the energy equation for blood flow which obtains the temperature distribution in the 3D triple-layered skin structure, and the equation for predicting thermal damage of the skin exposed to high thermal radiation on the triple-layered skin structure embedded with seven layers of countercurrent of blood vessels.

Step 3. Develop a numerical method for solving the established mathematical model. To this end, the finite difference method will be used for solving the modified Pennes equation and the fourth-order Runge-Kutta method will be employed for obtaining the blood temperature. The numerical scheme will be then solved by a preconditioned Richardson iteration in order to accelerate the convergence of the solution, and hence to reduce the computational cost. Based on the obtained temperature distribution, the skin burn injury will be calculated. Furthermore, the finite difference scheme will be proved to be unconditionally stable, and the Preconditioned Richardson iteration will be shown to be convergent.

Step 4. Test the established mathematical model and the obtained numerical method by considering a 3D triple-layered skin tissue with size of 1.62 cm by 1.62 cm by 1.542 cm, and a seven-level of countercurrent blood vessel network. Various meshes will be employed to this study.

1.3 Organization

Chapter One gives a general description of the research objectives in this dissertation.

Chapter Two introduces the main literature regarding the heat transfer in living tissues, particular models that represent the skin tissue, constructal theory of multi-scale tree-shape heat exchangers and previous work on heat transfer in 3D triple-layered skin structure embedded with three levels of blood vessels.

Chapter Three sets up the bio-heat transfer model of the 3D triple-layered skin structure embedded with seven-levels of countercurrent blood vessels, arteries and veins. which is based on the constructal theory.

Chapter Four gives the numerical method used in this research for solving the governing equations set up in Chapter Three.

Chapter Five tests the mathematical model and its numerical method in a 3D triple-layered skin tissue embedded with a seven-level of countercurrent network. Based on the obtained temperature distribution, skin burn injury will be calculated.

Finally, Chapter Six summarizes this dissertation research and suggests some possible future works to further the research.

CHAPTER TWO

LITERATURE REVIEW AND PREVIOUS WORK

Chapter Two will review the main literature related to the bio-heat transfer, including vascular countercurrent net work, constructal theory of multi-scale tree-shaped heat exchangers, and the previous work on the bio-heat transfer in skin tissue.

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

It shall point out that this section cites some results from previous literatures [42], [34], [35], [3], [43]. Heat transfer in living tissue is a complex process, which involves a great number of hierarchy levels from bio-macromolecules up to total organisms functioning as a whole. A microscopic view of living tissue considers the living tissue as consisting of two subsystems: the cellular tissue and a highly branching hierarchical vascular network involving arterial and venous beds. Blood flow through arteries supplies the cellular tissue with oxygen, nutrition products, etc. and controls heat balance in the system. The venous bed blood flow withdraws products resulting from life activity of the cellular tissue. The vascular network is embedded into cellular tissue. In spite of its small relative volume, the vascular network mainly determines heat and mass propagation. This is the case due to the fast convection transport with blood flow in vessels.

Blood vessels make up a complex network, which is practically a fractal. The larger a vessel, the faster the blood motion in it and, so, the stronger the effect of blood flow in the given vessel on heat transfer. Blood flow in capillaries practically does not affect heat propagation whereas blood inside large vessels moves so fast that its heat interaction with the surrounding cellular tissue is negligible.

As is known, several geometric models have been developed to simulate the vascular countercurrent network [42], [35]. In particular, H.W. Huang and R.B. Roemer [42] proposed a vascular countercurrent network as shown in Figure 2.1, which was plotted based on the figure in R.B.Roemer's paper as shown in Figure 2.2.

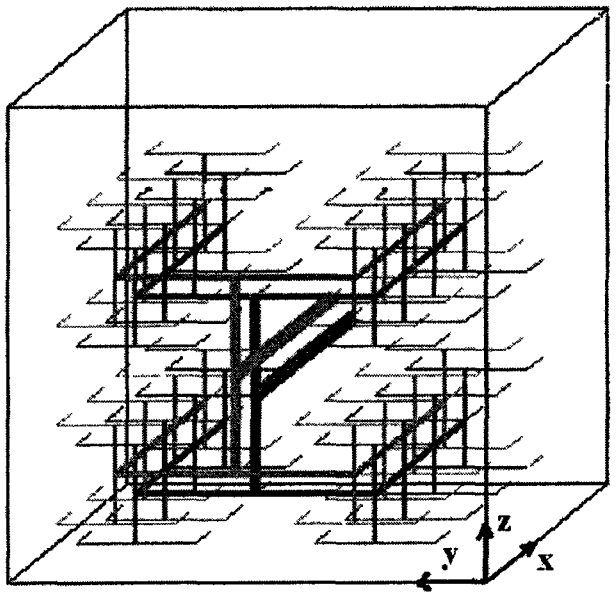


Figure 2.1 A proposed geometric model used consists of a regular, branching seven level of blood vessels of network of arteries (black) and veins (gray) that is embedded in a cubic control volume.

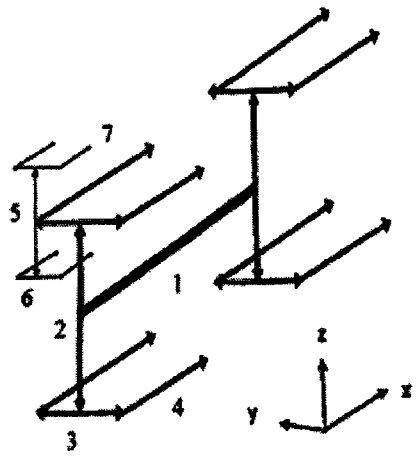


Figure 2.2 A figure showing the location of a subset of the vessels in the whole control volume. All of the level one, two, three and four vessels are shown, but only two sets of the connected level five-six-seven vessels are shown. A total of 64 such sets are used in the model, and they are regularly and uniformly spaced in the control volume [42].

In Figure 2.2, all vessels are designed to lie along the coordinates. There are up to seven levels of arteries and veins, beginning with the main artery and vein (level one), the biggest ones, where the diameter of the arteries (the same as veins) are decreased by a constant ratio γ between successive levels of branched vessels,

$$\gamma = \frac{D_{i+1}}{Di}. \tag{2.1}$$

Here D_{i+1} and D_i are the diameters of two successive levels of branching arteries. When two successive levels of numbered vessels do not branch but only change direction (i.e., levels six and seven in this model), the vessel diameter does not change, as shown in Figure 2.2.

It can be seen from Figure 2.2 that the artery network consists of the large central vessel (level one) running lengthwise (x) along the control volume. This vessel has two pairs of symmetric, vertical (z) vessels (level two) branching from it, one pair just inside the beginning of the cube (x=0) and one pair at the central plane (x=L/2). The mass flow

rate in the main arterial vessel decreases in a stepwise manner at both of those locations due to the blood removed by the second level vessels. In all cases studied in this paper, the level one artery is assumed to terminate at the central plane where the second set of level two vessels branch off of it. Each of the second level vessels branches into two level three vessels, which run crosswise (y). Each level three vessel feeds into a level four vessel which runs lengthwise (x), with each level four vessel extending (almost) one-half of the control volume length. Each of these level four vessels has four pairs of vertical (z) level five vessels periodically branching off of it, including the pairs at the corners where vessel levels three and four meet. Each level five vessel branches into two level six vessels which run crosswise (y). Each level six vessel then changes direction and becomes a level seven vessel which runs lengthwise (x). All vessels parallel to the x direction (i.e. the level one, four, and seven vessels) have been symmetrically located within the y, z planes, i.e. the level one artery is in the y, z center of the control volume, the level four vessels are located at centers of the four y, z quadrants of the control volume, and the level seven vessels are located at centers of the 16 squares that regularly divide up any y, z plane of the control volume. Similarly, the vessels parallel to the y, and z directions are uniformly spaced in the x direction.

In 2002, A. Lubashevsky and V. V. Gafiychuk [35] considered the living tissue to be a heterogeneous medium. They think that blood flow in capillaries practically does not affect heat propagation, whereas blood inside large vessels moves so fast that its heat interaction with the surrounding cellular tissue is negligible. They believe that there should be vessels of a certain length ℓ_v that are the smallest ones among the vessels wherein blood flow affects heat transfer remarkably.

The value of ℓ_v can be estimated as:

$$\ell_v \sim \sqrt{\frac{D}{jfL_n}}, \tag{2.2}$$

where $D = \kappa/(c\rho)$ is the temperature diffusivity of the cellular tissue determined by its thermal conductivity κ, specific heat c, and density ρ, the value j is the blood perfusion rate (the volume of blood going through tissue region of unit volume per unit time), and the factor $L_n \sim ln(l/a)$ is a logarithm of the mean ratio of the individual length to radius of blood vessels forming peripheral circulation. For the vascular networks, made up of the paired artery and vein trees where all the vessels are grouped into the pairs of the closely-spaced arteries and veins with opposite blood currents, the coefficient $f \sim L_n^{-1/2}$ accounts for the counter-current effect.

In the mean-field approximation the effect of blood flow on heat transfer is reduced to the renormalization of the temperature diffusivity, $D \rightarrow D_{eff}$, and the appearance of the effective heat sink fj in the bio-heat equation:

$$\frac{\partial T}{\partial t} = \nabla(D_{eff}\nabla T) - fj(T - T_a) + q_T\,. \tag{2.3}$$

Here T is the tissue temperature field averaged over scales about ℓ_v, the parameter T_a is the blood temperature inside the systemic circulation arteries, and the summand q_T called below the temperature generation rate is specified by the heat generation rate q as $q_T = q/(c\rho)$. The renormalization of the temperature diffusivity is mainly determined by the blood vessels of lengths about ℓ_v and, due to the fractal structure of vascular networks, the renormalization coefficient $F = D_{eff}/D$ is practically a constant of unity order, $F \gtrsim 1$. In their research, they proposed two types of blood vessel networks as shown in Figure 2.3 obtained from A. Lubashevsky's book published in 2002 [35].

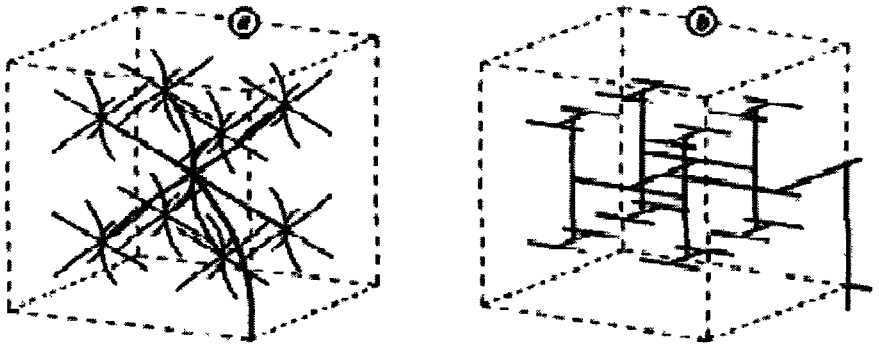


Figure 2.3 Models of the peripheral artery network embedding into the cellular tissue, (a) four-fold node model and (b) a more realistic dichotomic artery tree uniformly embedded into a cellular tissue domain. In the qualitative description of heat transfer both the models lead to the same result [35].

Figure 2.3a shows a simplified model for the vascular network, where the vessel lengths l_n and l_{n+1} of the neighboring hierarchy levels n and n+1 are related as $l_n = 2l_{n+1}$. Figure 2.3b demonstrates a more adequate model for the peripheral artery tree which, within the framework of the present qualitative analysis, may be reduced to the former one by combining three sequent two-fold nodes into one effective four-fold node at all the levels. In this case, the cubic domain of volume l_n^3 falls per each artery of level n.

Recently, A. Bejan has proposed a new geometric model, which is called "The Constructal Theory of Multi-scale Tree-shape Heat Exchangers" [3], [43]. In the constructal theory, they pointed out that the lengths of blood vessels follow an optical relationship as

$$L_m = 2^{\frac{1}{2}} L_{m+1}, \quad m = 1, \ldots, n. \tag{2.4}$$

While the number of blood vessels, $N_{m,}$ in each level is given by

$$N_m = 2^{m-1}, \quad m = 1, \ldots, n. \tag{2.5}$$

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

$$\dot{m}_i = 2^{-i}\dot{m}_1, \qquad i = 1, \ldots, n. \tag{2.6}$$

The ratio of diameters of the blood vessels satisfies:

$$\frac{D_{i+1}}{Di} = 2^{-\frac{1}{3}}, \qquad i = 1,2, \ldots, n. \tag{2.7}$$

2.2 Bio-heat Transfer Review

This review section is based on X. Tang's dissertation [46]. From her dissertation, it is known that the most utilized model for hyperthermia treatment planning involves the Pennes bio-heat transfer model (BHTE), in which the heat transfer between the blood vessels and tissue is assumed to occur mainly across the capillaries when the blood velocity is low [41]. The blood in the capillary bed instantly thermally equilibrates with the temperature of the surrounding tissue and enters the venous circulation at the local tissue temperature. Therefore, the contribution of the blood flow was modeled as a heat sink whose magnitude is proportional to the difference between the arterial supply temperature and the local tissue temperature.

As Tang pointed out, there are many numerical and experimental methods developed based on the Pennes bio-heat transfer model. Among them, Clegg and Roemer [10] performed hyperthermia sessions on a normal canine thigh to test the ability of a state and parameter estimation method to accurately predict the complete three-dimensional temperature distribution in experimental situations. They employed the Pennes equation as the system model and an optimization algorithm, which is based on a least squares error objective function, used for predicting certain unknown model parameters, such as the blood perfusion and the power deposition. Martin and Bowman [37] presented the exact steady state and transient solutions for the temperature distribution in laser irradiated and perfused tissue using the Pennes equation in

cylindrical coordinates. The obtained solutions were used to evaluate the significance of blood perfusion during continuous wave laser heating. Liauh and Roemer [27] presented a semilinear state and parameter estimation algorithm that decreases the total computational time required to accurately reconstruct complete hyperthermia temperature fields, since the relationship between the temperature and the blood perfusion based on the Pennes bio-heat transfer equation is generally nonlinear in the hyperthermia temperature estimation problem.

Using the automatic mesh generation capabilities of the software package ANSYS, Chatterjee and Adams [9] generated a 2D finite element thermal model of the prostate region of the human body based on the Pennes equation. The results show how selective heating can be obtained in the tumor region and what the effects of varying blood flow rates are.

Huang [24] considered the heat transfer within a perfused tissue in the presence of a vessel.The Pennes bio-heat transfer equation was used for the perfused tissue, and a lumped capacitance analysis was used for the convection in the vessel with a constant Nusselt number. Analytical solutions of the Pennes equation with a blood vessel were obtained. Payne [40] derived a design of the phantom from a combination of the convective fin equation and the Pennes BHTE, and developed a phantom model using an inverse technique applied to experimental data from a thin layer phantom to determine model parameters. Majchrzak and Mochnacki [36] considered the thermal processes proceeding within a perfused tissue in the presence of a vessel. The Pennes bio-heat transfer equation governs the steady state temperature field in the tissue sub-domain, while the ordinary differential equation resulting from the energy balance describes the

change of blood temperature along the vessel. The problem was solved using the combined numerical algorithm, in particular the boundary element method (for the tissue sub-domain) and the finite difference method (for the blood vessel sub-domain).

Liu and his co-workers [28], [29], [30] introduced a general form of the thermal wave model of Pennes bio-heat transfer in living tissues. The model was obtained based on a modified unsteady conduction equation (the CV equation). A general heat flux criterion was established to determine when the thermal wave propagation dominates the principal heat transfer process. This model can be used for tissue temperature prediction. Liu and Lu [31], [33] also used the dual reciprocity boundary element method to solve the integral inverse or direct bio-heat transfer problems.

Zhou and Liu [51] calculated temperature distributions based on the continuity, momentum and energy equations used in the fluid dynamics. Dai [15], [16] developed a domain decomposition method for solving the 3D Pennes bio-heat transfer equation in a triple-layered skin structure. Recently, Dai and Zhang [17] developed a numerical method for obtaining an optimal temperature distribution in a triple-layered cylindrical skin structure. It is the first time that the triple-layered skin structure, composed of epidermis, dermis and subcutaneous, was considered in the numerical model for the laser-induced hyperthermia. The method proved to be useful in optimizing laser power for a given laser irradiation pattern. However, the influence of blood vessels in the study was ignored. The presence of thermally significant vessels can have a dramatic impact on the temperature distribution in hyperthermia applications [26].

2.3 Previous Work on Skin Burn Injury by Radiation Heating

W. Dai and his colleagues in 2008 developed a mathematical model for skin burn injury in a triple-layered skin structure embedded with three-levels of countercurrent blood vessels with radiation heating [19]. The 3D triple-layered skin structure is shown in Figure 2.4, which is copied from Figure 2 in W. Dai's paper [19] with the author's permission.

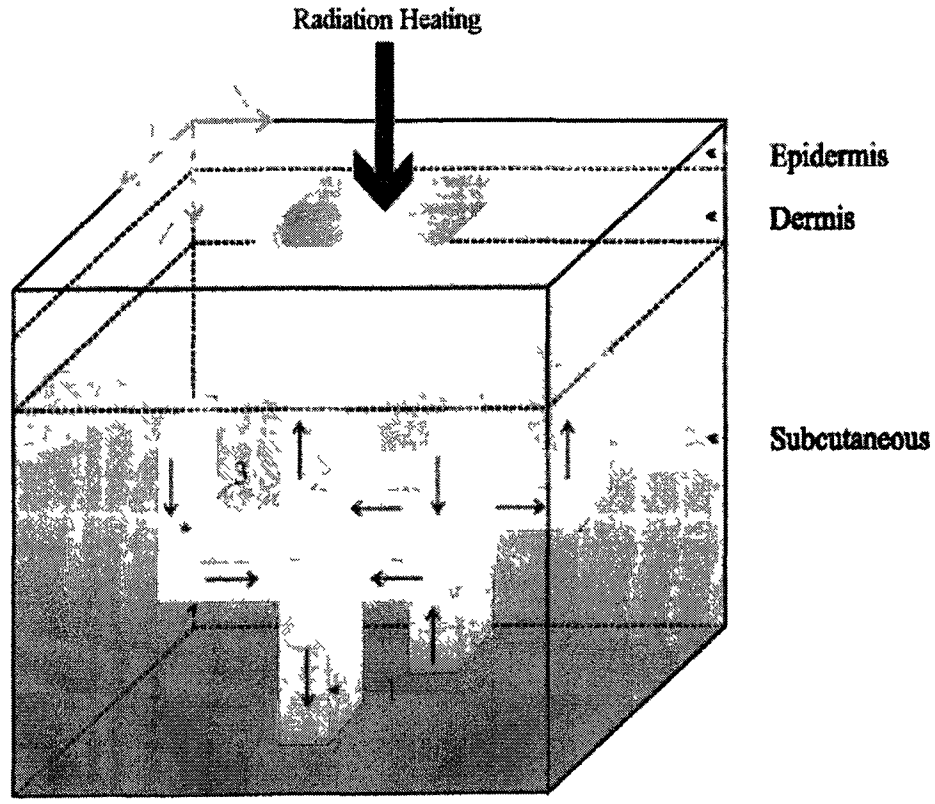


Figure 2.4 A three-dimensional triple-layered skin structure embedded with countercurrent vasculature [19].

In their model, the dimensions and blood flow rates of the multi-level blood vessels were determined based on A. Bejan's constructal theory of multi-scale tree-shaped heat exchangers [2], [3], [43], [4].

Figure 2.4, shows the basic arterial model consisting of the large central vessel (level 1) running lengthwise (in the z-direction) along the control volume. This vessel has a horizontal (in the x-direction) vessel (level 2) branching from it. The second vessel goes to third vessel (level 3) which runs again lengthwise (in the z-direction). The second vessel does not branch into two third vessels, and the diameters of these are also the same as suggested in [25]. In their vascular countercurrent network, these vessels are modeled as slim cuboids for simplicity.

To reduce the computational cost, they considered the target region to be a rectangular structure embedded with two countercurrent multi-level blood vessels that cross through the subcutaneous layer from the bottom to the top.

In their model, the Maxwell–Cattaneo flux law [8] was used to replace the Fourier's law, namely

$$\left(1+\tau\frac{\partial}{\partial t}\right)\vec{q} = -k\nabla T, \tag{2.8}$$

where $\vec{q}$ is the thermal flux vector, k and T denote the thermal conductivity and tissue temperature, respectively. The τ represents the time required to establish steady thermal conduction in a material volume element once a temperature gradient has been imposed across it, which could range up to 100 s (see [1]). Together with the Eq. (1.2), a modified (i.e., hyperbolic) form of the Pennes equation is obtained as the partial differential equation, which governs bio-heat transport.

$$\rho_l C_l\left(\frac{\partial T_l}{\partial t}+\tau\frac{\partial^2 T_l}{\partial t^2}\right)+\tau W_b^l C_b^l\frac{\partial T_l}{\partial t}$$
$$=k_l\left[\frac{\partial^2 T_l}{\partial x^2}+\frac{\partial^2 T_l}{\partial y^2}+\frac{\partial^2 T_l}{\partial z^2}\right]+W_b^l C_b^l(T_{out}-T_l)+Q_l,$$
$$l=1,2,3, \tag{2.9}$$

where T_l is the temperature of the lth tissue layer; T_{out} is the blood temperature at exit or entrance of the third level vessel for the artery or vein; ρ_l, C_b, and k_l denote the density, specific heat, and thermal conductivity of the lth tissue layer, respectively; τ is the thermal relaxation time; C_b^l is the specific heat of blood; and W_b^l is the blood perfusion rate. From Eq. (2.9), one may see that when τ is zero, Eq. (2.9) reduces to the Pennes equation [41], Eq. (1.3).

To test their model, they considered a 3D composite skin structure with the dimensions of 1 cm × 1 cm × 1.208 cm, where the values of the physical parameters used are given in Table 1. In their computation, they considered a heat transfer by convection from the skin's surface with the convection coefficient h = 0.001 W/cm^2, where the surface is exposed to an ambient temperature of 200 °C as the radiation heating. In addition, the thermal relaxation time and emissivity were taken to be τ = 20 s [29] and ε = 0.9 [20], respectively. Three meshes of 50 × 50 × 1208, 50 × 100 × 1208, and 100 × 100 × 1208 were chosen in order to test the convergence of the solution. Other thermal parameters used in the computation are listed in Tables 2 and 3. Because the results from this dissertation will be compared with their results, particularly their numerical results and Figure 2.4-2.13 from W. Dai's article in [19] are copied with the author's permission.

Figure 2.5 shows the temperature profiles at t = 200 s along the lines (a) y = 0.5 cm and (b) x = 0.5 cm on the skin surface, and (c) along the depth (the z-direction) at the

center of the skin surface, respectively, as well as (d) the temperature profiles over time at the point, where x = 0.5 cm, y = 0.5 cm, and z = 0.1 cm. It can be seen from this figure that there are no significant differences in the solutions obtained based on these three meshes, implying the solution is independent of the mesh size.

Figure 2.6 shows the contours of the temperature distributionsin these xz -cross-sections at y = 0.40 cm, where theartery is located, at various times (a) t = 100 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s. Figure 2.7 shows the contours of the temperature distributions in the xz-crosssection at y = 0.5 cm at various times (a) t = 100 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s. Figure 2.8 shows the contours of the temperature distributions at y = 0.56 cm, where the vein is located, at various times (a) t = 100 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s. Figure 2.9 shows the contours of the temperature distributions at the yz-cross-section at x = 0.5 cm, at various times (a) t = 100 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s. It can be seen from these figures that the temperature profiles are symmetric in the xz-cross-section at y = 0.5 cm, and the temperature elevations around the region where the vein is located are higher than those around the region where the artery is located. This implies that the vein is carrying the heat out, away from the heated area, and into the body core.

Figures 2.10–2.13 show the contours of the damage corresponding to Figures 2.5–2.9 showing the temperature distributions. Since values of x = 0.53, 1.0, 104 correspond to the first, second, and third degree burn injuries, respectively [20], one may see from these figures that the skin appears to be second degree burns at t = 200 s, and the third degree burn at t = 300 s and t = 400 s.

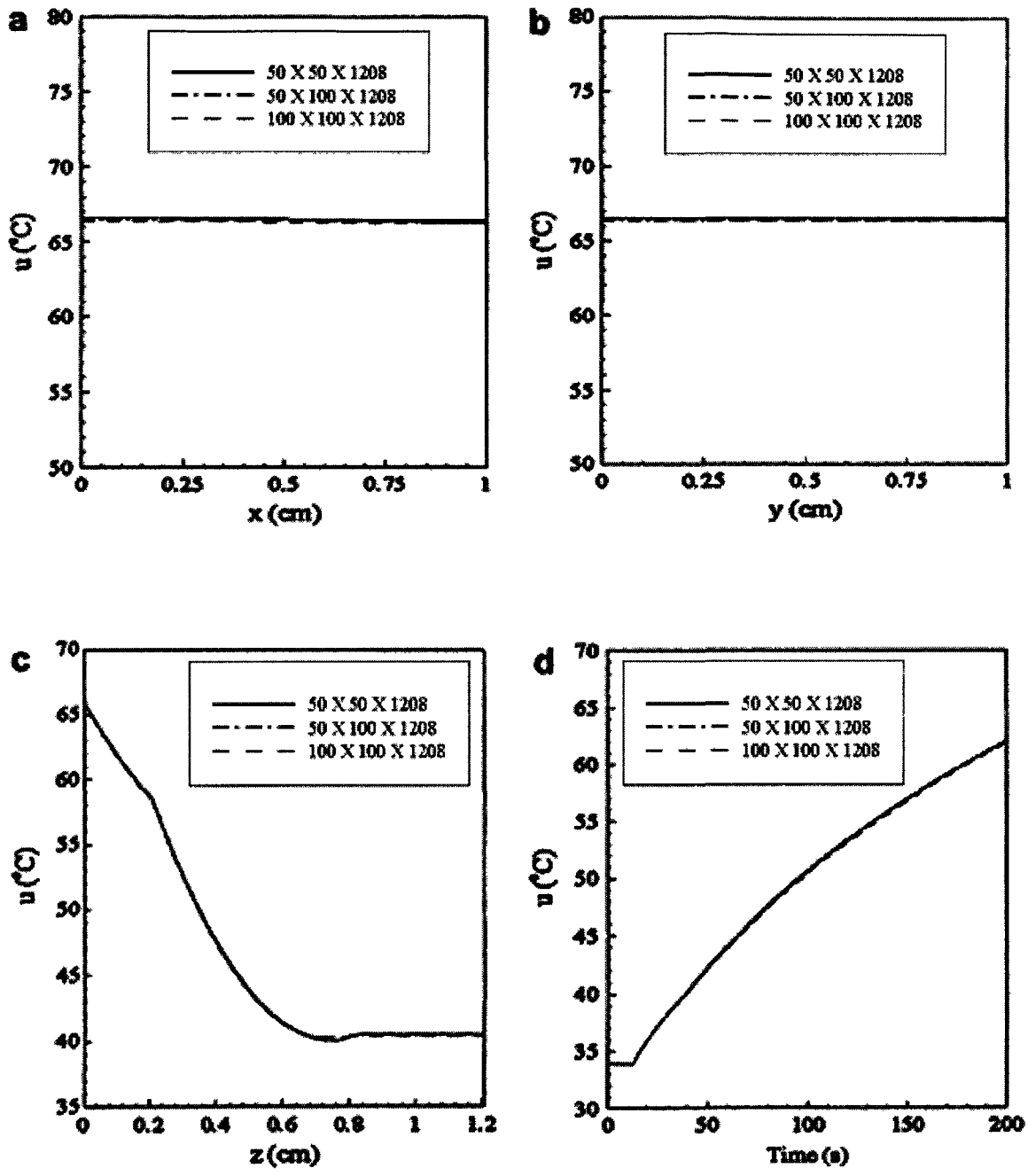


Figure 2.5 Profiles of temperature at t = 200 s along lines: (a) at y = 0.5 cm, (b) at x = 0.5 cm, on the skin surface, and (c) along the depth (the z-direction) at the center of the skin surface, as well as (d) at the point with x = 0.5 cm, y = 0.5 cm, and z = 0.1 cm over time [19].

Although W. Dai and his colleagues have developed a very promising mathematical model for skin burn injury induced by radiation heating, the considered vascular countercurrent network is only three-level of vessels. As pointed out by R.B. Roemer and his colleagues in 1996, the vascular countercurrent network could be up to seven-levels of vessels. This will increase the complex of the model and its computational cost. Thus, the purpose of this dissertation is to extend their research to the case that considers a 3D triple-layer skin structure embedded with a seven-level countercurrent vascular network and induced by radiation heating, and to develop a model and its numerical method that may be more accurate in predicting the skin burn injury induced by radiation heating.

CHAPTER THREE

BIO-HEAT TRANSFER MODEL

This chapter considers a 3D triple-layered skin structure consisting of epidermis, dermis and subcutaneous layer, where a dendritic seven level countercurrent vascular network is embedded. The governing equations for obtaining the temperature distribution in the skin structure are then set up, which will predict the skin burn injury induced by radiation heating.

3.1 Skin Structure and Vascular Network

This dissertation research considers a 3D triple-layered skin structure consisting of epidermis, dermis and subcutaneous layer as shown in Figure 3.1. Here a dendritic seven level countercurrent vascular network is embedded in the subcutaneous layer, here the blood vessels are considered as slim cuboids for simplicity. It should be noted that only large blood vessels can be seen in the subcutaneous tissue, because the dermis is very sparingly supplied with capillaries, and the capillary beds of skin lie immediately below the epidermis [23], and thus, the contribution of these small vessels to the heat transfer can be ignored [36], [44].

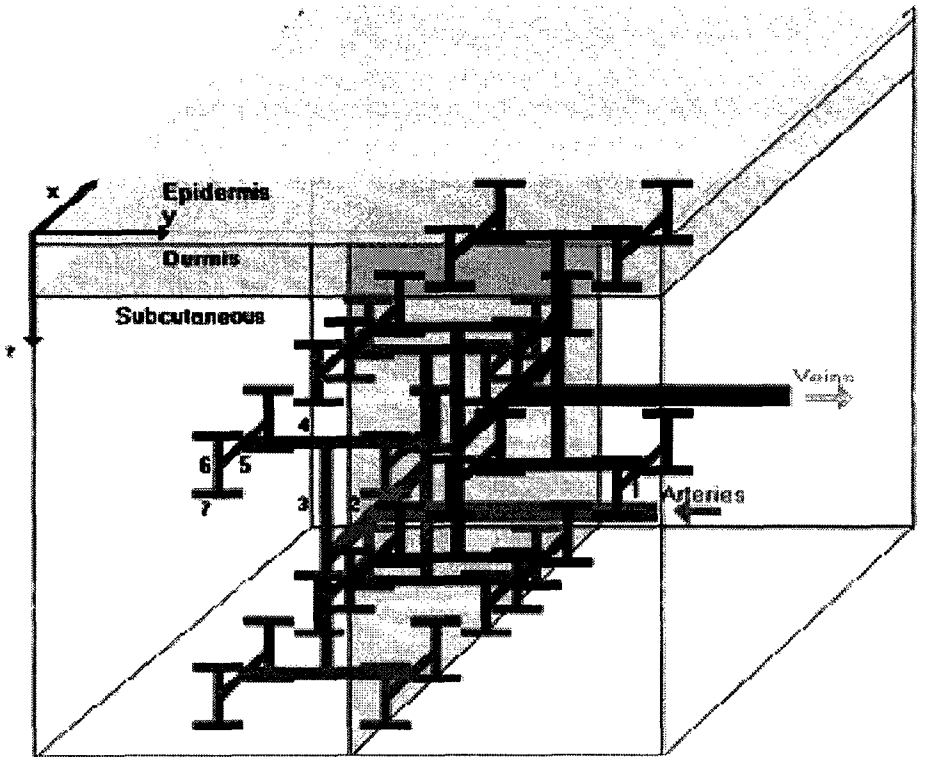


Figure 3.1 A 3D triple-layered skin structure embedded with seven levels of countercurrent vasculature network.

In Figure 3.1, the dark black color vessels is used to represent the artery dendritic network and the gray color vessels for the vein dendritic network. Levels of arteries are designed such that the first-level artery comes from right to left, running lengthwise in the y-coordinate along the skin structure, the second-level artery branches from the left end of the first-level artery running lengthwise in the x-coordinate, and the third-level artery has two vessels branching from the two ends of the second-level artery running lengthwise in the z-coordinate. There are four fourth-level arteries branching from the four ends of the third-level arteries running lengthwise in the y-coordinate. The fifth-level artery has eight arteries branching from the eight ends of fourth-level arteries running lengthwise in the x-direction. There are sixteen vessels in the sixth-level artery

running length wise in z-direction, and, finally, there are 32 vessels in the seventh-level artery running length wise in the x-direction.

The venous network is assumed to be similar to the arterial network, except that the blood flow direction in each vein is opposite of that in the artery; i.e., counter current flow occurs in these two kinds of vessels (see Figure 3.1). Symmetrically, the vein network has the same number of blood vessels as its counterpart artery in corresponding levels. As such, there are 128 blood vessels in total in the considered skin structure.

It should be pointed out that the above geometric structure of the dendric seven-level countercurrent vascular network is designed based on the suggestions by I. A. Lubashevsky [35] and R.B. Roemer [42]. However, the dimension and blood flow rates of the multi-level vessels, which will be determined based on the constructal theory of multi-scale tree-shaped heat exchangers [2], [3], [43], 4], are different from those used in [42].

Based on the constractal theory, the diameters of arteries satisfy an optimal rule and are assumed to be decreasing by a constant ratio γ between successive levels of branched vessels:

$$\gamma = \frac{NL_b^{m+1}}{NL_b^m} = \frac{NW_b^{m+1}}{NW_b^m} = 2^{-\frac{1}{3}}, \qquad m = 1, 2, \dots, 6, \tag{3.1}$$

where NL_b^m and NW_b^m are the length and width of the cross section of a blood vessel in level m, respectively, while the length of blood vessel is assumed to be double after two consecutive construction steps, which can be expressed as

$$L_b^m = 2^{\frac{1}{2}} L_b^{m+1}, \qquad m = 1, 2, \dots, 6, \tag{3.2}$$

where L_b^m is the length of the blood vessel in level m. Furthermore, the mass flow of blood in the mth level vessel, $M_m = v_m F_m$, is assumed to satisfy

$$M_m = 2M_{m+1} \qquad m = 1, 2, \ldots, 6, \tag{3.3}$$

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

Likewise, the diameter ratio, length ratio, and mass flow ratio between the successive levels of the branched blood as described in equations (3.1)-(3.3) can be used for veins.

3.2 Governing Equations

To obtain accurately the temperature distribution in the Figure 3.1 skin tissue structure, the thermal relaxation time of biological tissue in the skin tissue during the radiation heating will first be considered. As such, the Maxwell-Cattaneo flux law [8] is employed

$$\left(1 + \tau \frac{\partial}{\partial t}\right) \vec{q} = -k \nabla T, \tag{3.4}$$

where $\vec{q}$ represents the thermal flux vector, T is the tissue temperature and k denote the thermal conductivity of the tissue, τ is the thermal relaxation (or lag) time [Mitra 1995], which represents the time required to establish steady thermal conduction in the considering domain when a temperature gradient is imposed across a material volume element. Coupling Eq. (3.4) with the governing equation of heat transfer in biological tissue, Eq. (1.2),

$$\rho C \frac{\partial T}{\partial t} = -k \nabla \cdot \vec{q} + W_b C_b (T_b - T) + Q,$$

and to eliminate the flux vector $\vec{q}$, then a modified Pennes equation is obtained as follow

$$\rho C\left(\frac{\partial T}{\partial t}+\tau\frac{\partial^2 T}{\partial t^2}\right)+\tau W_b C_b\frac{\partial T}{\partial t}=k\Delta T+W_b C_b(T_b-T)+Q, \tag{3.5}$$

where T_b is the blood temperature; ρ and C denote the density, specific heat of the tissue, respectively; Q is the volumetric heat; C_b is the specific heat of blood; and W_b is the blood perfusion rate. The above modified Pennes equation (3.5) is then applied to the 3D triple-layered skin structure shown in Figure 3.1 and can be written as follows [19], [29], [30], [45]:

$$\begin{aligned}&\rho_l C_l\left(\frac{\partial T_l}{\partial t}+\tau\frac{\partial^2 T_l}{\partial t^2}\right)+\tau W_{\mathrm{b}}^l C_{\mathrm{b}}^l\frac{\partial T_l}{\partial t}\\&\quad=k_l\left[\frac{\partial^2 T_l}{\partial x^2}+\frac{\partial^2 T_l}{\partial y^2}+\frac{\partial^2 T_l}{\partial z^2}\right]+W_{\mathrm{b}}^l C_{\mathrm{b}}^l(T_{\mathrm{out}}-T_l)+Q_l,\\&\quad l=1,2,3,\end{aligned} \tag{3.6}$$

where T_l is the temperature of the *lth* skin tissue layer; T_{out} is the blood temperature at exit or entrance of the seventh level vessel for the artery or vein; $\rho_l C_l$ and $k_l$denote the density, specific heat, and thermal conductivity of the *l*th skin tissue layer, respectively; τ is the thermal relaxation time; C_b^l is the specific heat of blood; and W_b^l is the blood perfusion rate. Because in this model only radiation heating is considered, all other internal heat sources Q_l are assumed negligible.

The blood temperature in the cross-section of a vessel is assumed to be uniform. Thus, the steady-state energy balance in the blood vessel can be reached because the length of the considered blood vessel is relatively short, and the blood velocity is relatively high. However, one may use a transient heat transfer equation for a more accurate solution. Hence, the convective energy balance equations, which are used to calculate the artery (levels 1 through 6) blood temperatures, can be expressed as [18], [19], [23], [36], [49], [50], [46].

$$C_B M_1 \frac{\partial T_b^1}{\partial y} - \alpha P_1 (T_w^1 - T_b^1) = 0, \tag{3.7a}$$

$$C_B M_2 \frac{\partial T_b^2}{\partial x} - \alpha P_2 (T_w^2 - T_b^2) = 0, \tag{3.7b}$$

$$C_B M_3 \frac{\partial T_b^3}{\partial z} - \alpha P_3 (T_w^3 - T_b^3) = 0 \tag{3.7c}$$

$$C_B M_4 \frac{\partial T_b^4}{\partial y} - \alpha P_4 (T_w^4 - T_b^4) = 0, \tag{3.7d}$$

$$C_B M_5 \frac{\partial T_b^5}{\partial x} - \alpha P_5 (T_w^5 - T_b^5) = 0, \tag{3.7e}$$

$$C_B M_6 \frac{\partial T_b^6}{\partial z} - \alpha P_6 (T_w^6 - T_b^6) = 0, \tag{3.7f}$$

where C_B is the heat capacity of blood, M_l is the mass flow in the blood, (l is 1 to 7) and α is the heat transfer coefficient between blood and tissue, and P_m is the vessel perimeter. In addition, T_w^m and T_b^m are the wall temperature and the blood temperature in the mth level vessel. For the smallest, terminal arterial vessels (level 7), a decreased blood flow rate ($\dot{P}$) is included in the energy balance equation [19], [23], [36], [49], [50], [18], [46].

$$C_B M_7 \frac{\partial T_b^7}{\partial y} - \alpha P_7 (T_w^7 - T_b^7) - \dot{P} C_B F_7 T_b^7 = 0. \tag{3.8}$$

The convective energy balance equations (3.7)-(3.8) used to calculate the blood temperature in the artery domain is applied to the vein domain at the corresponding levels.

Assume that exchange on the surface of the skin with the surroundings includes the heat loss from convection, and radiation [44], [48], and is expressed as follows:

$$-k_1 \frac{\partial T_1}{\partial z} = h(T_a - T_1) + \varepsilon\sigma(T_a^4 - T_1^4), \quad z = 0, \tag{3.9}$$

where h is the convective heat transfer coefficient, T_a is the ambient temperature, σ is the Stefan-Boltzmann constant, and ε is the emissivity. Because radiation heating (such as

$T_a = 200$ °C) is considered, the ambient temperature T_a is much higher than T_1, it can be further assumed that the heat flux approaches zero as the tissue depth increases, which is realistic for a biological body [29]. For other boundaries, there is no heat loss is assumed, which gives the boundary condition:

$$\frac{\partial T}{\partial \vec{n}} = 0 \tag{3.10}$$

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

$$T_b^1 = T_{in}\,, \tag{3.11}$$

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

$$T_b^7 = T_{out}\,. \tag{3.12}$$

The continuity of the heat transfer between the lateral blood vessel and the tissues can be expressed as follows [19], [24]:

$$\frac{\partial T_b^m}{\partial \vec{n}} = B_i(T_w^m - T_b^m),\ m = 1, 2, 3, \ldots, 7. \tag{3.13}$$

The interfacial conditions between skin layers are assumed to be perfectly thermal contact and are given by [19]

$$T_1 = T_2,\ k_1\frac{\partial T_1}{\partial z} = k_2\frac{\partial T_2}{\partial z}\,,\ z = L_1\,, \tag{3.14}$$

$$T_2 = T_3,\ k_2\frac{\partial T_2}{\partial z} = k_3\frac{\partial T_3}{\partial z}\,,\ \mathrm{z} = L_1 + L_2\,. \tag{3.15}$$

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

$$T_l = T_0, \; t = 0, \; l = 1,2,3, \tag{3.16}$$

where T_1, T_2 and T_3 are the tissue temperature in skin layer one, two and three, respectively, and T_0 is the initial temperature in the tissue. The above equations, Eqs. (3.6) - (3.16), will give the temperature solution in the skin tissue domain. Once the temperature distribution is obtained, as suggested by Henriques and Mortiz [22], a quantitative description of thermal damage to skin, Ω, can be written as

$$\frac{\partial \Omega}{\partial t} = \zeta \exp\left(-\frac{\Delta E}{RT_l}\right), \; l = 1,2,3, \tag{3.17}$$

where ζ is the frequency factor, ΔE is the activation energy controlling the development of tissue injury (the quantities of ζ and ΔE are stated in Table 3). R (= 8.314 J K^{-1} mol^{-1}) is the gas constant. The temperature T_l denotes the temperature at layer ℓ. As suggested by Diller in 1992 [20] that Ω = 0.53, 1.0, 10^4 are corresponding to the first, second, and third degree burn injuries, respectively.

It can be seen that finding an analytical solution of Eqs. (3.6)-(3.16) is very difficult because of the complicated system and the nonlinear equation (see Eq. (3.9)). Thus, a numerical method for solving the above governing equations is necessary, in order to obtain the temperature distribution in the skin tissue structure.

CHAPTER FOUR

NUMERICAL METHODS

This chapter will develop a numerical method, which including a Crank-Nicolson type of finite difference scheme for solving the modified Pennes equation, and the fourth order Runge-Kutta method for obtaining the temperature of the blood vessels. The stability of the finite difference scheme will be proved based on the idea of W. Dai [12]. Then a preconditioned Richardson iteration for the numerical method is developed so that the solution system can be reduced to a tridiagonal linear system. The preconditioned Richardson iteration will be proved to be convergent.

4.1 Finite Difference Scheme

To develop a numerical method for obtaining the temperature distribution, $(u_l)_{ijk}^n$ and u_b are denoted to be the numerical approximations of $(T_l)(i\Delta x, j\Delta y, k\Delta z, n\Delta t)$ and T_b, where $\Delta x, \Delta y, \Delta z$, and Δt are the spatial and temporal mesh sizes, and i,j,k are integers with $0 \le i \le N_x$, $0 \le j \le N_y$, $0 \le k \le N_l^z$, so that $N_x \Delta x = NX$, $N_y \Delta y = NY$, and $N_l^z \Delta z = L_l$ $l=1,2,3$. In this mesh, it is assumed that $(u_3)_{ijk}^n = (u_b^m)_{ijk}$ when the grid point (i,j,k) is in the mth level blood vessel.

Since Eqs. (3.7)-(3.8) are first-order ordinary differential equations once T_w^m is determined, they can be solved by using the fourth-order Runge-Kutta method [7]. For example, the Runge-Kutta scheme for equation (3.7a) can be written as:

$$(T_b^1)_{j+1} = (T_b^1)_j + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4),$$

$$K_1 = \frac{\alpha\rho_1}{C_B M_1}\Delta y\left[(T_w^1)_j - (T_b^1)_j\right],$$

$$K_2 = \frac{\alpha\rho_1}{C_B M_1}\Delta y\left[(T_w^1)_j - (T_b^1)_j - \frac{1}{2}K_1\right],$$

$$K_3 = \frac{\alpha\rho_1}{C_B M_1}\Delta y\left[(T_w^1)_j - (T_b^1)_j - \frac{1}{2}K_2\right],$$

$$K_4 = \frac{\alpha\rho_1}{C_B M_1}\Delta y\left[(T_w^1)_j - (T_b^1)_j - K_3\right],$$

where α, ρ_1, C_B, M_1 are the heat transfer coefficient between blood and tissue, the vessel perimeters of level one, heat capacity of blood and the mass flow of blood in level one respectively, $(T_b^1)_j$ is blood temperature of level one at point j, $(T_w^1)_j$ is wall temperature of blood of level one at point j. Other equations in Eq. (3.7) and Eq. (3.8) are similar to apply.

Eq. (3.6) is solved by an approach developed in W. Dai's papers [12], [14]. Introducing

$$S_l = \left(1 + \tau\frac{W_b^l C_b^l}{\rho_l C_l}\right)T_l + \tau\frac{\partial T_l}{\partial t}, \tag{4.1}$$

Then Eq. (3.6) becomes

$$\rho_l C_l \frac{\partial S_l}{\partial t} = k_l\left[\frac{\partial^2 T_l}{\partial x^2} + \frac{\partial^2 T_l}{\partial y^2} + \frac{\partial^2 T_l}{\partial z^2}\right] + W_b^l C_b^l (T_{out} - T_l)\,. \tag{4.2}$$

Using the modified Euler method, Eq. (4.1) is discretized as

$$\frac{(S_l)_{ijk}^{n+1} + (S_l)_{ijk}^{n}}{2}$$

$$= \left(1 + \tau\frac{W_b^l C_b^l}{\rho_l C_l}\right)\frac{(u_l)_{ijk}^{n+1} + (u_l)_{ijk}^{n}}{2} + \tau\frac{(u_l)_{ijk}^{n+1} - (u_l)_{ijk}^{n}}{\Delta t} \tag{4.3}$$

For Eq. (4.2), the Crank-Nicolson method is employed and discretized as

$$\rho_l C_l \frac{(S_l)_{ijk}^{n+1}-(S_l)_{ijk}^{n}}{\Delta t} + W_b^l C_b^l \left[\frac{(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^{n}}{2} - (u_b)_{out}\right]$$

$$= k_l\left(\delta_x^2 + \delta_y^2 + \delta_z^2\right)\frac{(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^{n}}{2}, \quad l = 1, 2, 3. \tag{4.4}$$

Here $(s_l)_{ijk}^n$ is the numerical approximation of $(S_l)(i\Delta x, j\Delta y, k\Delta z, n\Delta t)$ defined by Eq. (4.1), $\delta_x^2 u_{ijk} = \frac{1}{\Delta x^2}(u_{i+1jk} - 2u_{ijk} + u_{i-1,jk})$ is a second-order central difference, and so on for the y and z directions.

It can be seen that the truncate error of Eqs. (4.3) and (4.4) are the order of $\Delta t^2 + \Delta x^2 + \Delta y^2 + \Delta z^2$. For the interfacial equations, Eqs. (3.14) and (3.15), the first-order forward and backward finite differences are used and discretized as

$$k_1 \frac{(u_1)_{ijN_1^z}^n - (u_1)_{ijN_1^x-1}^n}{\Delta z} = k_2 \frac{(u_2)_{ij1}^n - (u_2)_{ij0}^n}{\Delta z},$$

$$(u_1)_{ijN_1^z}^n = (u_2)_{ij0}^n, \tag{4.5a}$$

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

$$k_2 \frac{(u_2)_{ijN_2^z}^n - (u_2)_{ijN_2^x-1}^n}{\Delta z} = k_3 \frac{(u_3)_{ij1}^n - (u_3)_{ij0}^n}{\Delta z},$$

$$(u_2)_{ijN_2^z}^n = (u_3)_{ij0}^n. \tag{4.5b}$$

Similarly, the interfacial condition, Eq. (3.13), between the tissue and the lateral blood vessel is discretized as follows:

$$(u_3)_{ijk}^{n+1} = \frac{(u_3)_{i+1jk}^{n+1} + B_i \cdot \Delta x (u_3)_{i-1jk}^{n+1}}{1 + B_i \cdot \Delta x}, \tag{4.6a}$$

$$(u_3)_{ijk}^{n+1} = \frac{(u_3)_{ij+1k}^{n+1} + B_i \cdot \Delta y (u_3)_{ij-1k}^{n+1}}{1 + B_i \cdot \Delta y}, \tag{4.6b}$$

$$(u_3)_{ijk}^{n+1} = \frac{(u_3)_{ijk+1}^{n+1} + B_i \cdot \Delta z (u_3)_{ijk-1}^{n+1}}{1 + B_i \cdot \Delta z}, \tag{4.6c}$$

where the grid point (i, j ,k) is on the lateral walls of the blood vessel in the x, y, z directions, respectively.

To discretize the surface boundary condition, Eq. (3.9), the fourth power term is factored into

$$\begin{aligned}&(T_a^4 - T_1^4)\\&= (T_a^2 + T_1^2)(T_a^2 - T_1^2)\\&= (T_a^2 + T_1^2)(T_a + T_1)(T_a - T_1),\end{aligned} \tag{4.7a}$$

and then is discretize as

$$\begin{aligned}&k_1 \frac{(u_1)_{ij1}^{n+1} - (u_1)_{ij0}^{n+1}}{\Delta z}\\&= h\left[(u_1)_{ij0}^{n+1} - T_{air}\right] + \epsilon\sigma\left\{T_a^2 + \left[(u_1)_{ij0}^{n}\right]^2\right\}\\&\times \left[T_a + (u_1)_{ij0}^{n}\right]\left[T_a - (u_1)_{ij0}^{n+1}\right],\end{aligned} \tag{4.7b}$$

for any time level n.

Once the temperature distribution is obtained, the thermal damage of skin is then calculated by discretizing Eq. (3.17) as follows:

$$\frac{\Omega_{ijk}^{n+1} - \Omega_{ijk}^{n}}{\Delta t} = \zeta exp\left(-\frac{\Delta E}{R\frac{(T_l)_{ijk}^{n+1} + (T_l)_{ijk}^{n}}{2}}\right). \tag{4.8}$$

In order to reduce the computational cost, the solution method developed by Dai and his colleague [19] is further followed to solve for $(s_l)_{ijk}^{n+1}$ from Eq. (4.3), then substitute it into Eq. (4.4) to obtain an equation without the unknown $(s_l)_{ijk}^{n+1}$:

$$\frac{\rho_l C_l}{\Delta t}\left[\left(1 + r\frac{W_b^l C_b^l}{\rho_l C_l} + \frac{2\tau}{\Delta t}\right)(u_l)_{ijk}^{n+1} + \left(1 + r\frac{W_b^l C_b^l}{\rho_l C_l} - \frac{2\tau}{\Delta t}\right)(u_l)_{ijk}^{n} - 2(s_l)_{ijk}^{n}\right]$$

$$+W_b^l C_b^l \left[\frac{(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^{n}}{2} - (u_b)_{out}\right]$$

$$= k_l\left(\delta_x^2 + \delta_y^2 + \delta_z^2\right)\frac{(u_l)_{ijk}^{n+1}+(u_l)_{ijk}^{n}}{2},$$

$$l = 1, 2, 3. \tag{4.9}$$

It is noted that there are seven unknown $(u_l)_{ijk}^{n+1}$ in Eq. (4.9), which needs to solve iteratively. Here, a preconditioned Richardson iteration will be employed and described as follows:

Multiplying Eq. (4.9) by $\frac{\Delta t}{\rho_l C_l}$, choosing the coefficient of the each term $(u_l)_{ijk}^{n+1}$, and double those related to x and y to give preconditioner

$$L_{pre}^l = 1 + r\frac{W_b^l C_b^l}{\rho_l C_l} + \frac{2\tau}{\Delta t} + \frac{W_b^l C_b^l \Delta t}{2\rho_l C_l} + \frac{2k_l \Delta t}{\rho_l C_l}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) - \frac{k_l \Delta t}{2\rho_l C_l}\delta_z^2 \tag{4.10}$$

Then applying it to the Richardson Iteration method, $\vec{X}^{(I+1)} = \vec{X}^{(I)} - \omega(A\vec{X}^{(I)} - \vec{b})$ for $A\vec{X} = \vec{b}$, one may obtain a preconditioned Richardson iteration method as:

$$L_{pre}^l\left[(u_l)_{ijk}^{n+1}\right]^{(I+1)}$$

$$= L_{pre}^l\left[(u_l)_{ijk}^{n+1}\right]^{(I)} - \omega\left\{\left(1 + r\frac{W_b^l C_b^l}{\rho_l C_l} + \frac{2\tau}{\Delta t}\right)\left[(u_l)_{ijk}^{n+1}\right]^{(I)}\right.$$

$$+\left(1 + r\frac{W_b^l C_b^l}{\rho_l C_l} - \frac{2\tau}{\Delta t}\right)(u_l)_{ijk}^{n} - 2(s_l)_{ijk}^{n}$$

$$+\frac{W_b^l C_b^l \Delta t}{\rho_l C_l}\left[\frac{\left[(u_l)_{ijk}^{n+1}\right]^{(I)}+(u_l)_{ijk}^{n}}{2} - (u_l)_{out}\right]$$

$$\left.-\frac{k_l \Delta t}{\rho_l C_l}\left(\delta_x^2 + \delta_y^2+\delta_z^2\right)\frac{\left[(u_l)_{ijk}^{n+1}\right]^{(I)}+(u_l)_{ijk}^{n}}{2}\right\}, \tag{4.11}$$

where $0 \leq \omega \leq 1$ is the relaxation factor. It can be seen that

$$L_{pre}^{l}\left[(u_l)_{ijk}^{n+1}\right]^{(I+1)} =$$

$$\left\{1 + r\frac{W_b^l C_b^l}{\rho_l C_b} + \frac{2\tau}{\Delta t} + \frac{W_b^l C_b^l \Delta t}{2\rho_l C_l} + \frac{2k_l \Delta t}{\rho_l C_l}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) - \frac{k_l \Delta t}{2\rho_l C_l}\delta_z^2\right\}\left[(u_l)_{ijk}^{n+1}\right]^{(I+1)},$$

which represents a tri-diagonal matrix. In fact, let

$$\mathrm{P} = 1 + r\frac{W_b^l C_b^l}{\rho_l C_b} + \frac{2\tau}{\Delta t} + \frac{W_b^l C_b^l \Delta t}{2\rho_l C_l} + \frac{2k_l \Delta t}{\rho_l C_l}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right),$$

$$\mathrm{Q} = \frac{k_l \Delta t}{2\rho_l C_l}.$$

Thus,

$$\begin{aligned} L_{pre}^{l}\left[(u_l)_{ijk}^{n+1}\right]^{(I+1)} &= \mathrm{P}\left[(u_l)_{ijk}^{n+1}\right]^{(I+1)} - \mathrm{Q}\delta_z^2\left[(u_l)_{ijk}^{n+1}\right]^{(I+1)} \\ &= \mathrm{P}\left[(u_l)_{ijk}^{n+1}\right]^{(I+1)} - \mathrm{Q}\left[(\mathrm{u_l})_{\mathrm{jik-1}}^{\mathrm{n+1}} - 2(\mathrm{u_l})_{\mathrm{jik}}^{\mathrm{n+1}} + (\mathrm{u_l})_{\mathrm{jik+1}}^{\mathrm{n+1}}\right]^{(\mathrm{I+1})} \\ &= \left[-\mathrm{Q}(\mathrm{u_l})_{\mathrm{jik-1}}^{\mathrm{n+1}} + (\mathrm{P}+2\mathrm{Q})(\mathrm{u_l})_{\mathrm{jik}}^{\mathrm{n+1}} - \mathrm{Q}(\mathrm{u_l})_{\mathrm{jik+1}}^{\mathrm{n+1}}\right]^{(I+1)} \\ &= \left[-\mathrm{a}(\mathrm{u_l})_{\mathrm{jik-1}}^{\mathrm{n+1}} + \mathrm{b}(\mathrm{u_l})_{\mathrm{jik}}^{\mathrm{n+1}} - \mathrm{c}(\mathrm{u_l})_{\mathrm{jik+1}}^{\mathrm{n+1}}\right]^{(I+1)}, \end{aligned}$$

where a = Q, b = P+2Q, c = Q. There are three unknowns in the z-direction, which gives a tridiagonal linear system for solving $(u_l)_{ijk-1}^{n+1}$, $(u_l)_{ijk}^{n+1}$, and $(u_l)_{ijk+1}^{n+1}$.

If $(u_3)_{ijk}^{n}$=$(u_b^m)_{ijk}$ is denoted where the grid point (*i* ,*j*, *k*) is in the *m*th level blood vessel, then the Thomas algorithm can be used line by line along the *z*-direction.

From the knowledge of numerical solutions for partial differential equations, if a finite difference scheme is consistent (implying that the truncation error goes to zero when the grid size tends to zero) and stable (small changing the initial data, cause only a small changing the numerical result), then the numerical solution obtained based on the finite difference scheme is convergent to the exact solution of the partial different equation.

As pointed out in the previous section, development of an unconditionally stable finite difference scheme is particularly important for this research, since the first layer of the skin tissue is very small. The next section shows that the finite difference scheme, Eq. (4.3) and (4.4), is unconditionally stable. Furthermore, it will prove the preconditioned Richardson iteration, Eq. (4.11), will be proved to be convergent.

4.2 Stability

To show the stability of the obtained finite difference scheme, Eqs. (4.3) and (4.4), the idea in W. Dai's paper published in 1999 is followed [12]. For simplicity, only one-dimensional case is considered. Notations, inner products, and norms are introduced as follows:

$$\nabla_x u_j^n = \frac{u_{j+1}^n - u_j^n}{\Delta x}, \tag{4.12a}$$

$$\delta_x^2 u_j^n = \frac{1}{\Delta x^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n), \tag{4.12b}$$

$$(u^n, v^n) = \Delta x \sum_{j=1}^{N} u_j^n v_j^n , \tag{4.12c}$$

$$\|u^n\|^2 = (u^n, u^n) , \tag{4.12d}$$

$$\|\nabla_x u^n\|_1^2 = (\nabla_x u^n, \nabla_x u^n)_1 = \Delta x \sum_{j=1}^{N} (\nabla_x u_j^n)^2. \tag{4.12e}$$

It can be seen that for any mesh functions, u_j^n, v_j^n, with $u_0^n = u_1^n$, $u_N^n = u_{N-1}^n$, $v_0^n = v_1^n$ and $v_N^n = v_{N-1}^n$,

$$(\delta_x^2 u^2, v^n) = -(\nabla_x u^n, \nabla_x v^n)_1 . \tag{4.12f}$$

The proof of Eq. (4.12f) can be seen in [12].

It can be seen that Eq. (4.3) and Eq. (4.4) in one-dimensional case can be rewritten as:

$$\frac{s_j^{n+1}+s_j^n}{2} = (1+\tau\frac{W_bC_b}{\rho C})\frac{u_j^{n+1}+u_j^n}{2} + \tau\frac{u_j^{n+1}-u_j^n}{\Delta t}, \tag{4.13a}$$

$$\frac{s_j^{n+1}-s_j^n}{\Delta t} + \frac{W_bC_b}{\rho C}\left[\frac{u_j^{n+1}+u_j^n}{2} - (u_b)_{out}\right] = \frac{k}{\rho C}\delta_x^2\frac{u_j^{n+1}+u_j^n}{2}, \tag{4.13b}$$

where j=1, 2, …, N-1. Assume that $\{(s_1)_j^n, (u_1)_j^n\}$ and $\{(s_2)_j^n, (u_2)_j^n\}$ are two solutions of equation (4.13a) and (4.13b) with different initial conditions, but the same boundary conditions. Let $w_j^n = (s_1)_j^n - (s_2)_j^n$ and $v_j^n = (u_1)_j^n - (u_2)_j^n$. It can be obtained from Eq. (4.13a) and (4.13b)

$$\tau\frac{v_j^{n+1}-v_j^n}{\Delta t} = -(1+\tau\frac{W_bC_b}{\rho C})\frac{v_j^{n+1}+v_j^n}{2} + \frac{w_j^{n+1}+w_j^n}{2}, \tag{4.13c}$$

$$\frac{w_j^{n+1}-w_j^n}{\Delta t} + \frac{W_bC_b}{\rho C}\left[\frac{v_j^{n+1}+v_j^n}{2}\right] = \frac{k}{\rho C}\delta_x^2\frac{v_j^{n+1}+v_j^n}{2}. \tag{4.13d}$$

Then multiply Eq. (4.13d) by $\Delta x(\, w_j^{n+1} + w_j^n)$ and summing j from 1 to N.

$$\frac{\rho C}{k}\frac{\|w^{n+1}\|^2-\|w^n\|^2}{\Delta t} + \frac{W_bC_b}{2k}\left((v^{n+1}+v^n),(w^{n+1}+w^n)\right)$$

$$= \frac{\Delta x}{2}\sum_{j=1}^{N}\delta_x^2\,(v_j^{n+1}+v_j^n)\left(\, w_j^{n+1}+w_j^n\right). \tag{4.13e}$$

On the other hand, multiply Eq. (4.13c) by $\Delta x(v_j^{n+1} + v_j^n)$, and sum j from 1 to N. This gives

$$\frac{((w^{n+1}+w^n),(v^{n+1}+v^n))}{2} =$$

$$\tau\frac{\|v^{n+1}\|^2-\|v^n\|^2}{\Delta t} + (1+\tau\frac{W_bC_b}{\rho C})\frac{\|v^{n+1}+v^n\|^2}{2}. \tag{4.13f}$$

Substituting $\frac{1}{2}((w^{n+1} + w^n), (v^{n+1} + v^n))$ in Eq. (4.13f) into Eq. (4.13e) obtains

$$\frac{\rho C}{k}\frac{\|w^{n+1}\|^2-\|w^n\|^2}{\Delta t} + \frac{W_bC_b\tau}{k}\frac{\|v^{n+1}\|^2-\|v^n\|^2}{\Delta t} + \frac{W_bC_b}{k}(1+\tau\frac{W_bC_b}{\rho C})\frac{\|v^{n+1}+v^n\|^2}{2}$$

$$= \frac{\Delta x}{2}\sum_{j=1}^{N} \delta_x^2 (v_j^{n+1} + v_j^n)\left(w_j^{n+1} + w_j^n \right). \tag{4.13g}$$

Furthermore, multiplying Eq. (4.13c) by $\Delta x \delta_x^2 (v_j^{n+1} + v_j^n)$ and summing j from 1 to N gives

$$\tau \frac{\left(\delta_x^2 (v^{n+1}+v^n),(v^{n+1}-v^n)\right)}{\Delta t} + \frac{1}{2}\left(1 + \tau \frac{W_b C_b}{\rho C}\right)\left(\delta_x^2 (v^{n+1} + v^n), (v^{n+1} + v^n)\right)$$

$$= \frac{\Delta x}{2}\sum_{j=1}^{N} \delta_x^2 (v_j^{n+1} + v_j^n)\left(w_j^{n+1} + w_j^n \right). \tag{4.13h}$$

Based on Eq. (4.12e) and Eq. (4.12f), the above equation can be simplified to

$$-\tau \frac{\|\nabla_x v^{n+1}\|_1^2 - \|\nabla_x v^n\|_1^2}{\Delta t} - \frac{1}{2}\left(1 + \tau \frac{W_b C_b}{\rho C}\right)\left(\|\nabla_x (v^{n+1} + v^n)\|_1^2\right)$$

$$= \frac{\Delta x}{2}\sum_{j=1}^{N} \delta_x^2 (v_j^{n+1} + v_j^n)\left(w_j^{n+1} + w_j^n \right). \tag{4.13i}$$

Subtracting Eq. (4.13i) from Eq. (4.13g) gives

$$\frac{\rho C}{k}\frac{\|w^{n+1}\|^2 - \|w^n\|^2}{\Delta t} + \frac{W_b C_b \tau}{k}\frac{\|v^{n+1}\|^2 - \|v^n\|^2}{\Delta t} + \frac{W_b C_b}{k}\left(1 + \tau \frac{W_b C_b}{\rho C}\right)\frac{\|v^{n+1}+v^n\|^2}{2}$$

$$+ \tau \frac{\|\nabla_x v^{n+1}\|_1^2 - \|\nabla_x v^n\|_1^2}{\Delta t} + \frac{1}{2}\left(1 + \tau \frac{W_b C_b}{\rho C}\right)\left(\|\nabla_x (v^{n+1} + v^n)\|_1^2\right) = 0. \tag{4.13j}$$

Note that $\|v^{n+1} + v^n\|^2 \geq 0$, $\|\nabla_x (v^{n+1} + v^n)\|_1^2 \geq 0$, one may drop them out from Eq. (4.13j) and obtain

$$\frac{\rho C}{k}\frac{\|w^{n+1}\|^2 - \|w^n\|^2}{\Delta t} + \frac{W_b C_b \tau}{k}\frac{\|v^{n+1}\|^2 - \|v^n\|^2}{\Delta t} + \tau \frac{\|\nabla_x v^{n+1}\|_1^2 - \|\nabla_x v^n\|_1^2}{\Delta t} \leq 0. \tag{4.13k}$$

This indicates that

$$\rho C\|w^{n+1}\|^2 + W_b C_b \tau \|v^{n+1}\|^2 + k\tau \|\nabla_x v^{n+1}\|_1^2$$

$$\leq \rho C\|w^n\|^2 + W_b C_b \tau \|v^n\|^2 + k\tau \|\nabla_x v^n\|_1^2$$

$$\leq \rho C\|w^{n-1}\|^2 + W_b C_b \tau \|v^{n-1}\|^2 + k\tau \|\nabla_x v^{n-1}\|_1^2$$

$$\leq \cdots$$

$$\leq \rho C\|w^0\|^2 + W_b C_b \tau \|v^0\|^2 + k\tau \|\nabla_x v^0\|_1^2 \tag{4.13l}$$

which indicates further that $\|w^{n+1}\|^2$, $\|v^{n+1}\|^2$ and $\|\nabla_x v^{n+1}\|_1^2$ are controlled by the initial differences $\|w^0\|^2$, $\|v^0\|^2$ and $\|\nabla_x v^0\|_1^2$, implying that the scheme is stable. Since there is no restriction on the mesh ratio, $\frac{\Delta t}{\Delta x^2}$, in the proof, the finite difference scheme is unconditionally stable.

4.3 Convergence of the Preconditioned Richardson Iteration

To solve iteratively a linear system, $A\vec{u} = \vec{b}$, one may employ the Richardson

$$u^{(I+1)} = u^{(I)} - \omega(Au^{(I)} - \vec{b}) . \quad I = 0, 1, 2, 3, \ldots \tag{4.14}$$

where ω is a relaxation factor. However, the iteration may not be convergent for all cases.

To overcome this difficulty, one often introduces a matrix L to the Richardson iteration as

$$Lu^{(I+1)} = Lu^{(I)} - \omega(Au^{(I)} - \vec{b}) , \tag{4.15}$$

where L is chosen so that the system (4.15) is convergent and is easily computed. Such a L is called a preconditioner.

The preconditioned Richardson iteration, Eq. (4.11) , will be shown to be convergence. To this end, $\boldsymbol{A_x}, \boldsymbol{A_y}$ and $\boldsymbol{A_z}$ matrices and $(\boldsymbol{u_l})^{\boldsymbol{n+1}}$ vector are introduced as follows:

$$(\boldsymbol{A_x}(\boldsymbol{u_l})^{\boldsymbol{n+1}})_{ijk} = -\frac{k_l \Delta t}{2\rho_l C_l} \delta_x^2 (u_l)_{ijk}^{n+1} ,$$

$$(\boldsymbol{A_y}(\boldsymbol{u_l})^{\boldsymbol{n+1}})_{ijk} = -\frac{k_l \Delta t}{2\rho_l C_l} \delta_y^2 (u_l)_{ijk}^{n+1} ,$$

$$(\boldsymbol{A_z}(\boldsymbol{u_l})^{\boldsymbol{n+1}})_{ijk} = -\frac{k_l \Delta t}{2\rho_l C_l} \delta_z^2 (u_l)_{ijk}^{n+1} ,$$

and write system (4.11) in vector form as:

$$\boldsymbol{L}_{pre}^{l}[(\boldsymbol{u}_l)^{\mathbf{n+1}}]^{(\mathrm{I}+1)}$$

$$= \boldsymbol{L}_{pre}^{l}[(\boldsymbol{u}_l)^{\mathbf{n+1}}]^{(\mathrm{I})} - \omega\{(1+\tau\frac{\mathrm{W_b^l C_b^l}}{\rho_l C_l}+\frac{2\tau}{\Delta t}+\frac{W_b^l C_b^l \Delta t}{2\rho_l C_l})[(\boldsymbol{u}_l)^{\mathbf{n+1}}]^{(\mathrm{I})}$$

$$+(\boldsymbol{A}_x+\boldsymbol{A}_y+\boldsymbol{A}_z)[(\boldsymbol{u}_l)^{\mathbf{n+1}}]^{(\mathrm{I})}+\mathbf{f}\}\,, \tag{4.16a}$$

where the preconditioner is written as

$$\boldsymbol{L}_{pre}^{l}=\left[1+r\frac{W_b^l C_b^l}{\rho_l C_b}+\frac{2\tau}{\Delta t}+\frac{W_b^l C_b^l \Delta t}{2\rho_l C_l}+\frac{2k_l\Delta t}{\rho_l C_l}\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\right]\mathbf{I}+\boldsymbol{A}_x$$

and

$$(\mathbf{f})_{ijk} = (1+\tau\frac{W_b^l C_b^l}{\rho_l C_l}-\frac{2\tau}{\Delta t})(u_l)_{ijk}^{n}-2(s_l)_{ijk}^{n}$$

$$+\frac{W_b^l C_b^l \Delta t}{\rho_l C_l}\left[\frac{(u_l)_{ijk}^{n}}{2}-(u_{l)_{out}}\right]$$

$$-\frac{k_l\Delta t}{\rho_l C_l}(\delta_x^2+\delta_y^2+\delta_z^2)\frac{(u_l)_{ijk}^{n}}{2}\,. \tag{4.16b}$$

Eq. (4.16a) can be further written as

$$[(\boldsymbol{u}_l)^{\mathbf{n+1}}]^{(\mathrm{I}+1)} = \boldsymbol{R}[(\boldsymbol{u}_l)^{\mathbf{n+1}}]^{(\mathrm{I})}-\omega\,\boldsymbol{L}_{pre}^{-1}\vec{\boldsymbol{f}}, \tag{4.17}$$

where $\boldsymbol{R}$ is an iteration matrix:

$$\boldsymbol{R}\equiv\boldsymbol{I}-\omega\boldsymbol{L}_{pre}^{-1}\left[(1+\tau\frac{\mathrm{W_b^l C_b^l}}{\rho_l C_l}+\frac{2\tau}{\Delta t}+\frac{W_b^l C_b^l \Delta t}{2\rho_l C_l})\boldsymbol{I}+(\boldsymbol{A}_x+\boldsymbol{A}_y+\boldsymbol{A}_z)\right]. \tag{4.18}$$

From the numerical linear algebra [7], it is known that the iteration in (4.17) converges if the iteration matrix $\boldsymbol{R}$ has a spectral radius $\rho(\boldsymbol{R})<1$. It can be seen that the eigenvalues of

$$\boldsymbol{L}_{pre}^{-1}\left[(1+\tau\frac{\mathrm{W_b^l C_b^l}}{\rho_l C_l}+\frac{2\tau}{\Delta t}+\frac{W_b^l C_b^l \Delta t}{2\rho_l C_l})\boldsymbol{I}+(\boldsymbol{A}_x+\boldsymbol{A}_y+\boldsymbol{A}_z)\right]$$

have the form:

$$\lambda_{ijk} = \frac{A+r_x sin^2(i\pi\Delta x/2)+r_y sin^2(j\pi\Delta y/2)+r_z sin^2(k\pi\Delta z/2)}{A+r_x+r_y+r_z sin^2(k\pi\Delta z/2)}, \tag{4.19}$$

where $A = 1 + \tau\frac{W_b^l C_b^l}{\rho_l C_l} + \frac{2\tau}{\Delta t} + \frac{W_b^l C_b^l \Delta t}{2\rho_l C_l}$, $r_x = \frac{2k_l\Delta t}{\Delta x^2 \rho_l C_l}$, $r_y = \frac{2k_l\Delta t}{\Delta y^2 \rho_l C_l}$, and $r_z = \frac{2k_l\Delta t}{\Delta z^2 \rho_l C_l}$.

Since the numerator is smaller than the denominator in Eq. (4.19), then it obtains $0 < \lambda_{ijk} \leq 1$. If one chooses the relaxation parameter ω to be 1, then from Eq. (4.17), the spectral $\rho(\boldsymbol{R}) = 1 - \omega \min(\lambda_{ijk}) < 1$, imply that the iteration method (4.17) is convergent when $\omega = 1$.

4.4 The Computation

Based on the obtained numerical scheme in the previous section, one may obtain the temperature distribution in a 3D triple-layered skin structure embedded with a seven-level of vascular countercurrent network, which is exposed to the radiation heating, and hence predict the skin burning. The detailed computations can be described into several steps as follows:

Step 1. Initiate the temperature of the tissue u, the blood u_w^m and the wall of blood u_b^m.

Obtain the blood temperature u_b^m,by solving Eqs. (3.7)-(3.8) using the fourth-order Runge Kutta method. It should be pointed out that at each program loop, the temperature of each level of blood vessel must first be calculated, where the coordinates of blood vessels are independent of the coordinates of consideration of skin domain.

Step 2. Update the wall temperature of the blood vessel u_w^m by Eqs. (4.6a)- (4.6c).

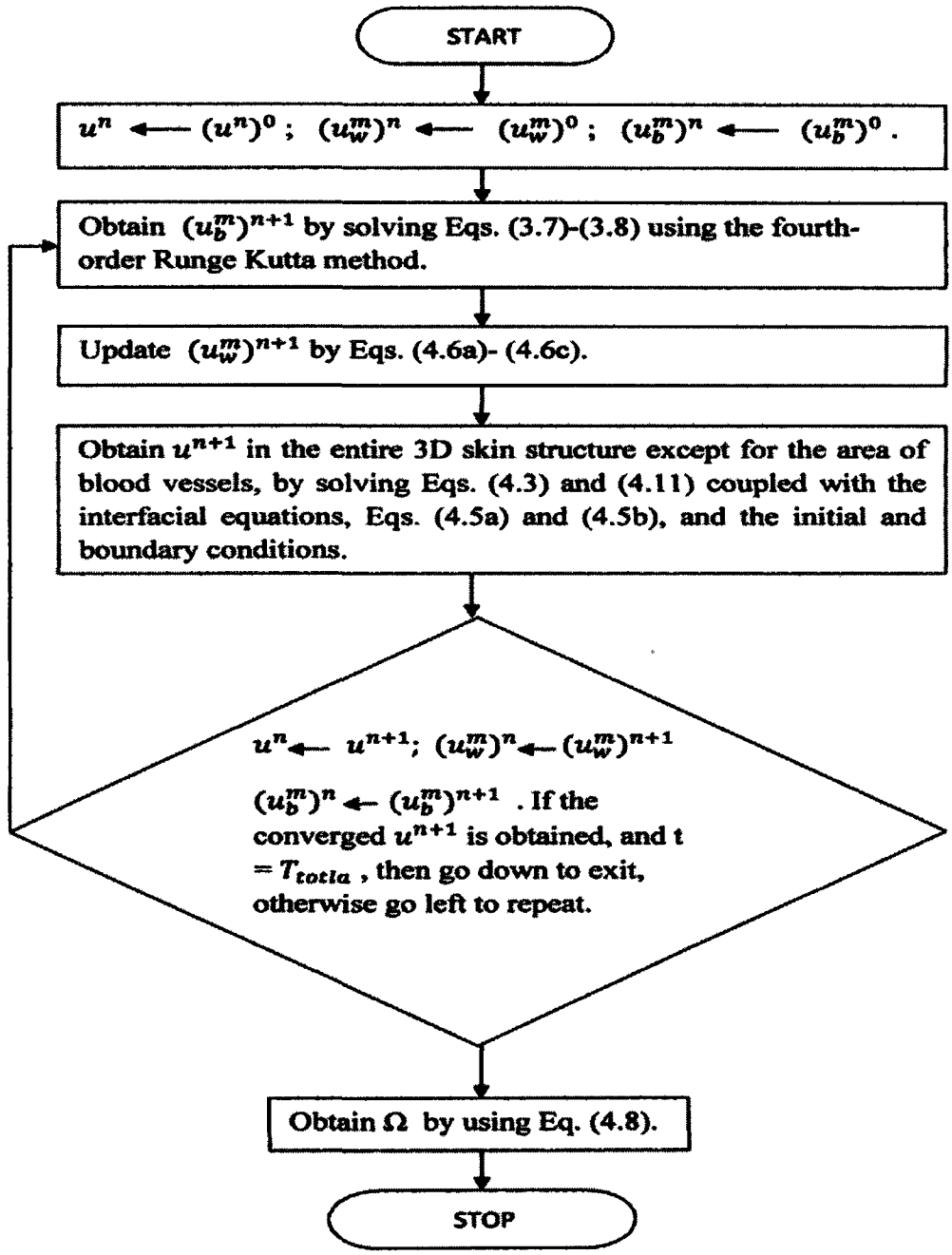
Step 3. Obtain the temperature distribution u in the entire 3D skin structure except for the area of blood vessels, by solving Eqs. (4.3) and (4.11) coupled with the

interfacial equations, Eqs. (4.5a) and (4.5b), and the initial and boundary conditions.

Step 4. Repeat steps 1 through 3 until a convergent solution, u, at time level n+1 is obtained.

Step 5. Calculate the damage function Ω by using Eq. (4.8).

The flow chart of the computation can be plotted as follows:



In the computation, the temperature of blood vessels is first obtained, then the wall temperature of the blood vessel u_w^m is updated, and the tissue temperature of the 3D domain is calculated. In particular, the coordinates of the blood vessels is independent on the coordinates of the tissue domain. Moreover, the direction of the coordinates of blood

vessels are chosen to coincide with the direction of the blood flow. As such, for the first level artery, its coordinate coincide with the opposite of the y direction of the tissue domain. The second level branches into two vessels. One is along the x direction, the other is opposite to the x direction. The level three artery consists four parts, with two parts run lengthwise with the z direction, and the other two run opposite of the z direction. There are eight parts in a level four artery, with four parts running lengthwise with the y direction, and the other four running opposite of the y direction. Level five has sixteen parts. There are eight parts running lengthwise with the x direction, and the other eight run opposite of the x direction. The level six has thirty-two parts: sixteen of them run lengthwise with the z direction, the other sixteen run opposite of the z direction. The seven level artery has sixty-four parts, with thirty-two running lengthwise with the y direction, and the other thirty-two run opposite of the y direction.

The coordinates of the veins are similar with corresponding level of arteries, except that the direction is the opposite.

CHAPTER FIVE

NUMERICAL EXAMPLE

This chapter will test the applicability of the model and its numerical scheme by considering a 3D triple-layered skin structure embedded with a seven-level dendritic countercurrent vascular network, as shown in Figure 3.1. Results will be discussed and compared with the previous work in W. Dai's paper [19].

5.1 Example Description

First, a 3D skin structure is chosen to be 1.62 cm ×1.62 cm ×1.542 cm in dimensions, where the thickness of layer one, two and three are 0.008 cm, 0.2 cm and 1.334 cm, respectively. The size of blood vessels at each level are listed in Table 1. Here, a larger dimensions of the 3D skin structure than the one (1 cm × 1 cm × 1.208 cm) in W. Dai's paper [19] was chosen, so that the seven-level dendritic countercurrent vascular network could be embedded completely into the 3D skin structure. In the computation, heat convection occurring on the skin's surface was considered (where the convection coefficient h=0.001 W/cm² [25]), and the surface is exposed to an ambient temperature of 200 °C as the radiation heating. The thermal relaxation time and emissivity for radiation heating were taken to be τ=20 s [29] and ε=0.9 [20], respectively. Three meshes of 162 × 162 × 771, 162 ×324×771, and 162 ×162×1542 were chosen in order to test the grid independence and the convergence of the numerical solution.

Other thermal parameters used in the computation are listed in Tables 2 and 3. The criterion of convergence at each time step was chosen to be $\left|T^{(new)} - T^{(old)}\right| < 10^{-3}$. The computer codes were written in C++ format which can be seen in Appendix B, and were run in a supercomputer in Louisiana Optical Network Initiative (LONI).

5.2 Results and Discussion

Figure 5.1 shows the temperature profiles at t=200 s in the case of (a) y=0.81 cm (along the central line on the skin surface as shown on (e)), (b) x=0.81 cm on the skin surface (along the central line as shown on (f)), and (c) along the depth of z-direction at the center of skin surface as shown on (g), as well as (d) the temperature profiles over time at the point as shown on (h) (where x=0.81cm, y=0.81 cm and z=0.1 cm). The numerical results were obtained based on three different meshes of 162 × 162 × 1542, 162 × 162 × 771, 162 × 324 × 771. It can be seen from these figures that there are no significant differences in the solutions obtained based on these three meshes, implying that the solution is independent on the mesh size. Because of symmetry, Figure 5.1(a) and 5.1(b) show an uniform temperature distribution with 66 °C. Figure 5.1(d) indicates that the temperature at the central point of skin surface is 62 °C . Compared with Figure 2.5 in Chapter Two, which were obtained by W. Dai in [19], one may see that numerical results in Figure 5.1(a), 5.1(b) and 5.1(d) are not different from those corresponding results in Figure 2.5. However, the curve in Figure 5.1(c) is slightly different when $z \geq 0.5$ cm. This is probably because the main artery is placed in a different location in this case.

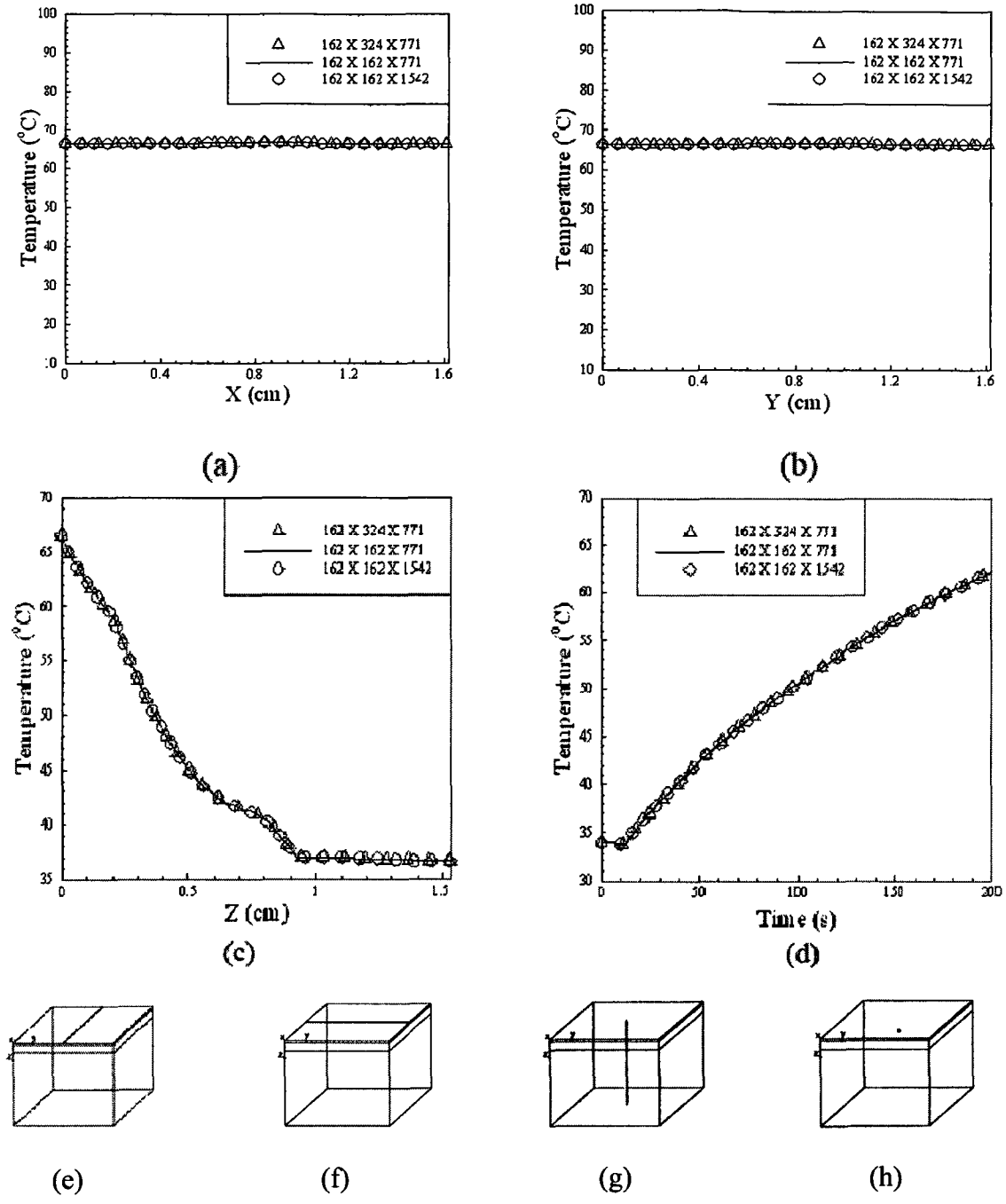
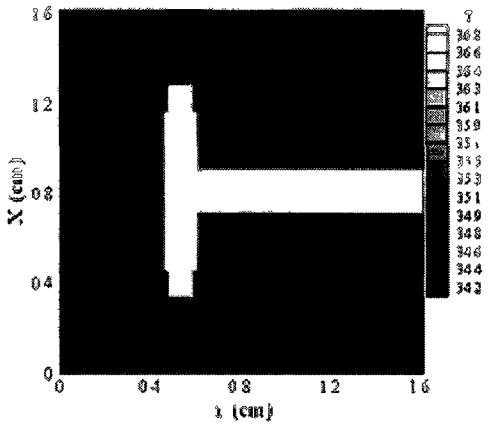


Figure 5.1 Temperature profile at t = 200 s along lines (a) at y = 0.81 cm, (b) at x = 0.81 cm on the surface of skin , and (c) along the z direction from the top to bottom at the central point of skin surface, and (d) at the point with x = 0.81cm, y = 0.81cm and z = 0.1 cm over time.

Figures 5.2-5.9 show contours of the temperature distribution of the skin structure with embedded seven-level dendritic countercurrent vascular network at t=0, 200, 300 and 400 seconds, respectively. In particular, Figure 5.2 shows the xy cross-sections at z = 1.025 cm as shown in (e) where levels 1, 2 and 3 of the artery and levels 4, 5 and 6 of the vein appear. From Figure 5.2, one may see that at t = 0, the blood temperature is 37 °C, which is hotter than the tissue With time increase, the skin tissue is exposed to the radiation. The heat transfers into the tissue, and increases the tissue temperature. At t = 400 s, the tissue is hotter than the blood. Part of level 6 blood vessels are also heat up. Figure 5.3 shows the xy cross-sections at z = 0.725 cm as shown in (e), where levels 4, 5 and 6 of the artery and levels 1, 2 and 3 of the vein appear. Similar results are obtained.

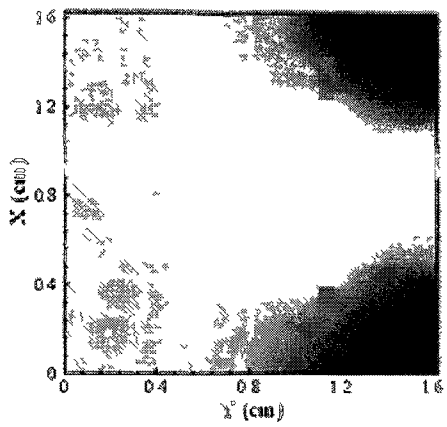
Figure 5.4 shows the xz cross-sections at y=0.54 cm as shown in (e), where levels 2, 3 and 4 of the artery appear. Figure 5.5 shows the xz cross-sections at y=0.92 cm as shown in (e), where level 1 of the artery and levels 2, 3 and 4 of the vein appear. Figure 5.6 shows the yz cross-sections at x=0.39 cm as shown in (e), where levels 3, 4 and 5 of the artery and levels 3, 4 and 5 of the vein appear. Figure 5.7 shows the yz cross-sections at x=0.55 cm as shown in (e), where levels 2, 6 and 7 of the artery and levels 2, 6 and 7 of the vein appear. Figure 5.8 shows the xz cross-sections at y=0.76 cm as shown in (e), where levels 1, 5, 6 and 7 of the artery and levels 4 and 7 of the vein appear. Figure 5.9 shows the xy cross-section at y=0.7 cm as shown in (e), where levels 1, 4 and 7 of the artery and levels 5, 6 and 7 of the vein appear. It can be seen from these figures that the temperature distributions are symmetric if the blood vessels appear to be symmetric in the cross-section. If the temperature of the artery is higher than the

temperature of the surrounding tissue, then the heat will transfer from the artery to the skin tissue. On the other hand, if the temperature of the artery is lower than the temperature of the surrounding tissue, then the blood in the artery will cool the tissue. In addition, the vein is carrying the heat out, away from the heated area, and into the body core. From Figures 5.4-5.9, one may see that the top potion of tissue is exposed to the radiation heating and is gradually heated up with time increase. At = 400 s, the hottest temperature reaches 80 °C, which is in the bright area. This may cause skin burning. By comparing with Figures 2.6 - 2.9, it can be seen that Figures 5.4 – 5.9 show a similar hottest portion. Due to the complex vascular network, the temperature distribution in the blood vessel area cannot be seen as clearly as these in Figures 2.6 -2.9.



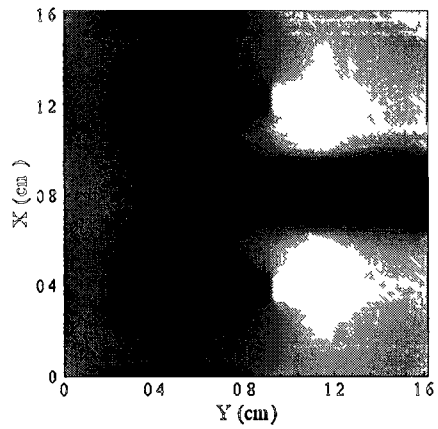
(a)

(Levels 1, 2 and 3 of artery (white) and levels 4, 5 and 6 of vein (light black) appear)



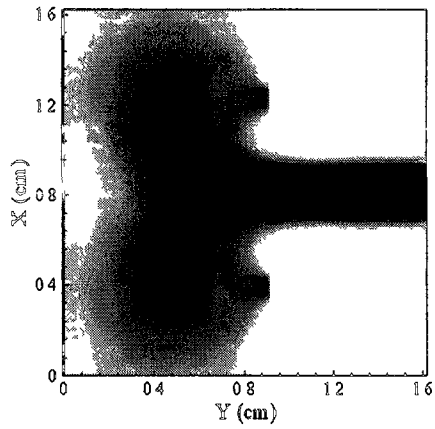
(b)

(35.2 °C (darkest) ≤ T ≤ 37.6 °C (brightest))



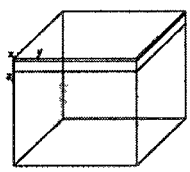
(c)

(37 °C (darkest) ≤ T ≤ 39.2 °C (brightest))



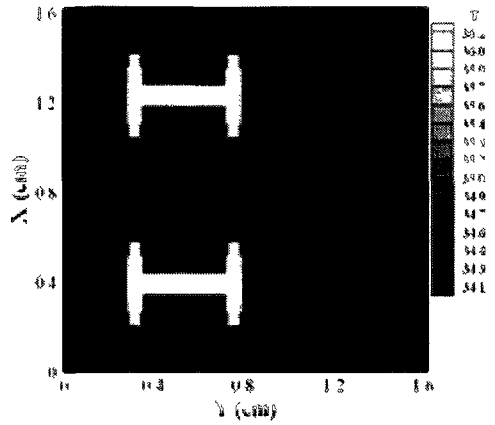
(d)

(37 °C (darkest) ≤ T ≤ 40.9 °C (brightest))

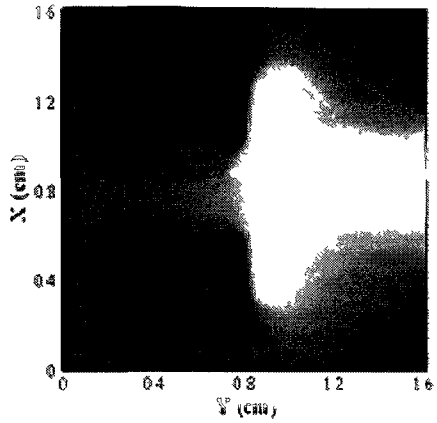


(e)

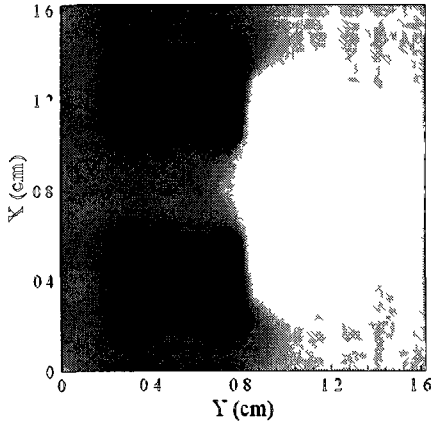
Figure 5.2 Contours of temperature distribution in the xy cross-section at z = 1.025 cm as shown on (e), when (a) t = 0 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.



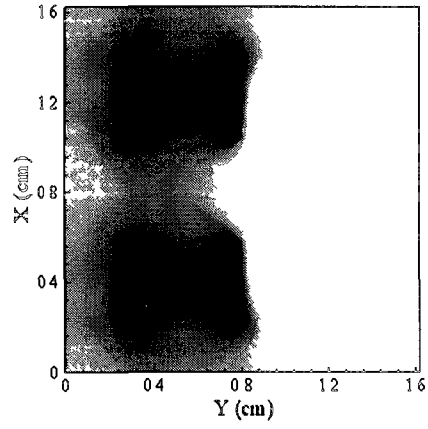
(a)
(Levels 4, 5 and 6 of artery (light black) and levels 1, 2 and 3 of vein (white) appear)



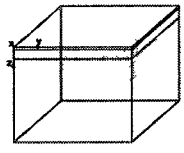
(b)
(37 °C (darkest) ≤ T ≤ 42.6 °C (brightest))



(c)
(37 °C (darkest) ≤ T ≤ 46.1 °C (brightest))

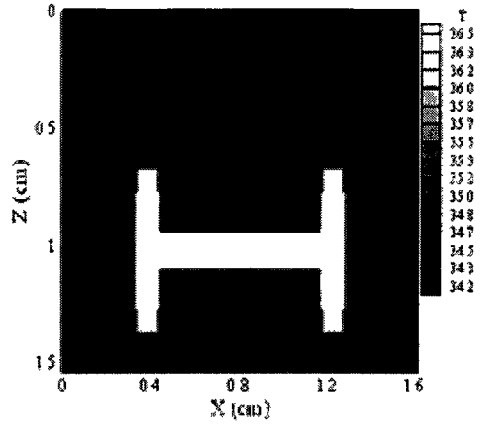


(d)
(37.1 °C (darkest) ≤ T ≤ 48.5 °C (brightest))

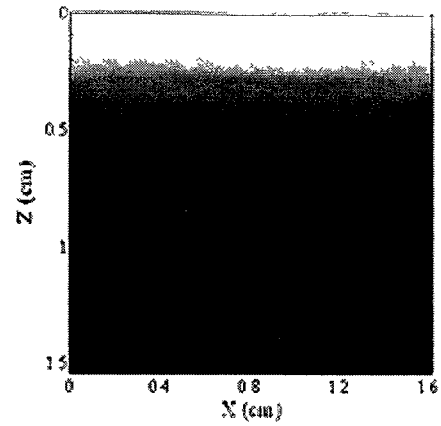


(e)

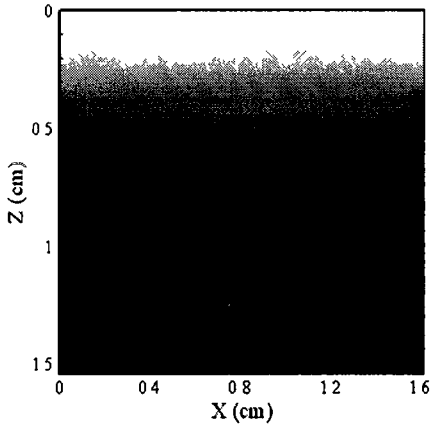
Figure 5.3 Contours of temperature distribution in the xy cross-section at z = 0.725 cm as shown in (e), when (a) t = 0 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.



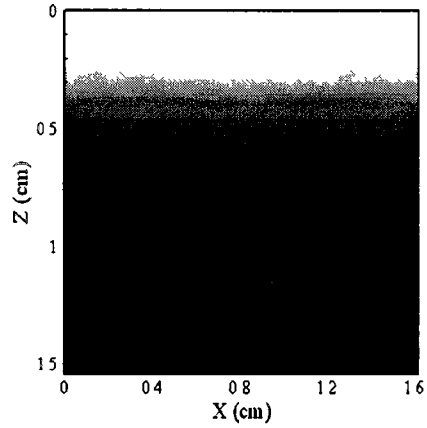
(a)
(Levels 2, 3 and 4 of artery appear)



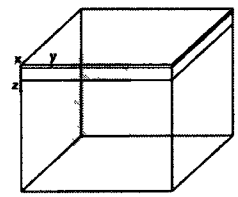
(b)
(36.8 °C (darkest) ≤ T ≤ 67 °C (brightest))



(c)
(37 °C (darkest) ≤ T ≤ 74.6 °C (brightest))

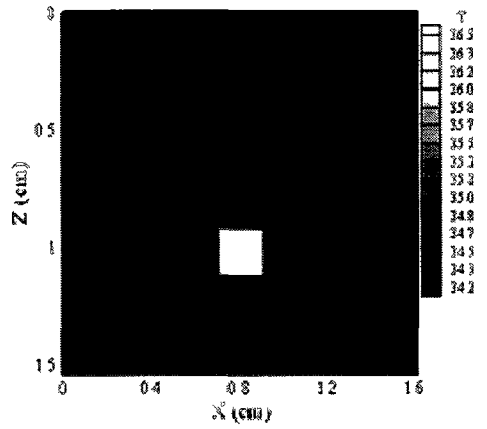


(d)
(37.1 °C (darkest) ≤ T ≤ 79.4 °C (brightest))

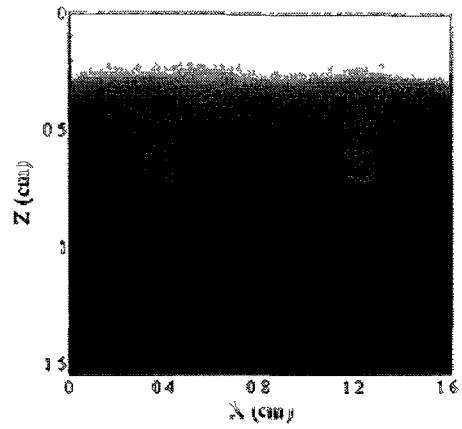


(e)

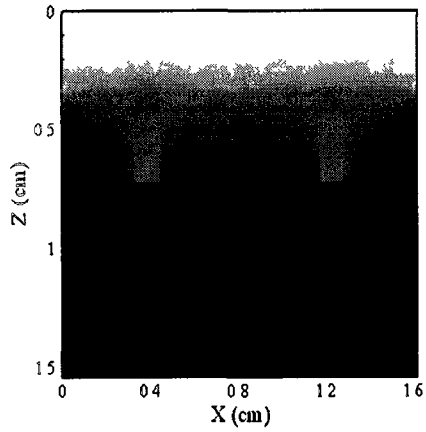
Figure 5.4 Contours of temperature distribution in the xz cross-section at y = 0.54 cm as shown in (e), when (a) t = 0 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.



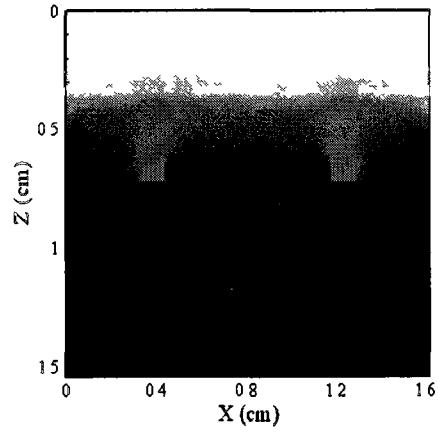
(a)
(Level 1 of artery (white) and levels 2, 3 and 4 of vein (light black) appear)



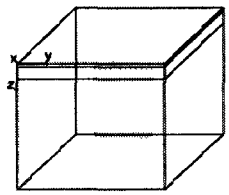
(b)
(36.2 °C (darkest) ≤ T ≤ 66.1 °C (brightest))



(c)
(36.7 °C (darkest) ≤ T ≤ 74.8 °C (brightest))

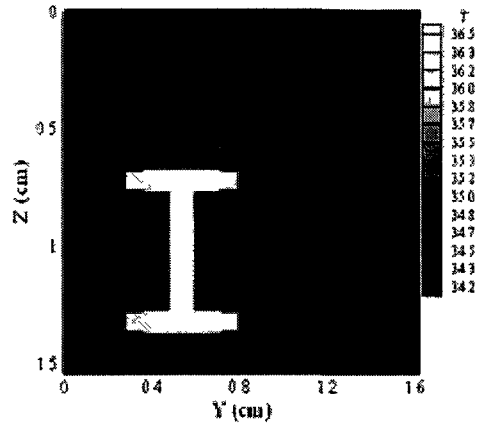


(d)
(37 °C (darkest) ≤ T ≤ 80 °C (brightest))



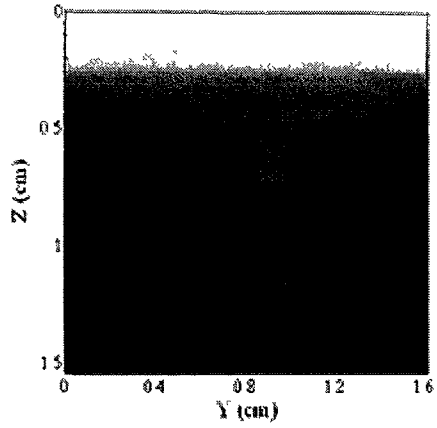
(e)

Figure 5.5 Contours of temperature distribution in the xz cross-section at y = 0.92 cm as shown in (e), when (a) t = 0 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.



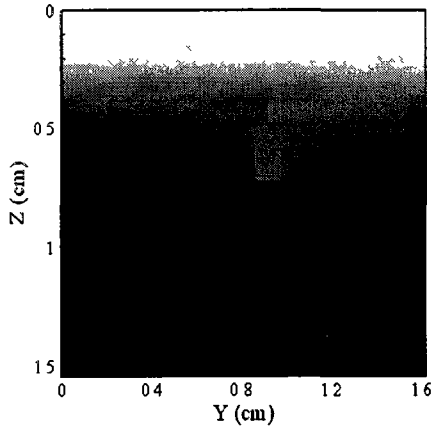
(a)

(Levels 3, 4 and 5 of artery (white) and levels 3, 4 and 5 of vein (light black) appear)



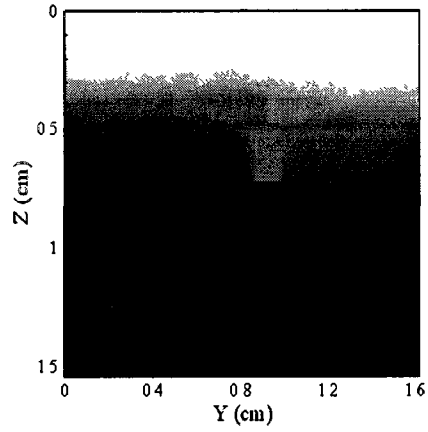
(b)

(35.1 °C (darkest) ≤ T ≤ 67 °C (brightest))



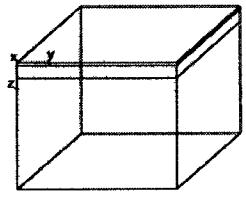
(c)

(36.2 °C (darkest) ≤ T ≤ 74.9 °C (brightest))



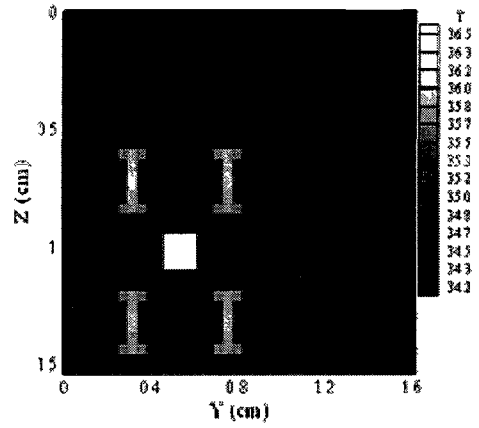
(d)

(37.1 °C (darkest) ≤ T ≤ 80.6 °C (brightest))

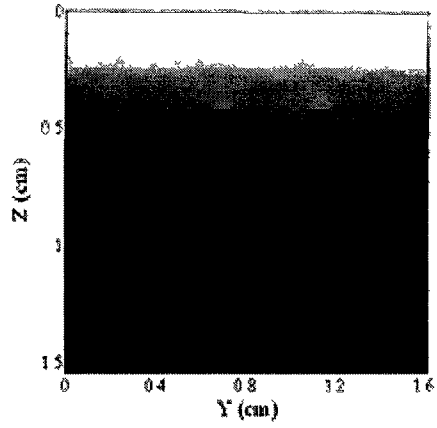


(e)

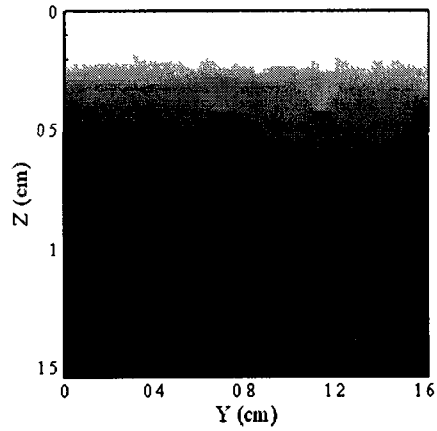
Figure 5.6 Contours of temperature distribution in the yz cross-section at x = 0.39 cm as shown in (e), when (a) t = 0 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.



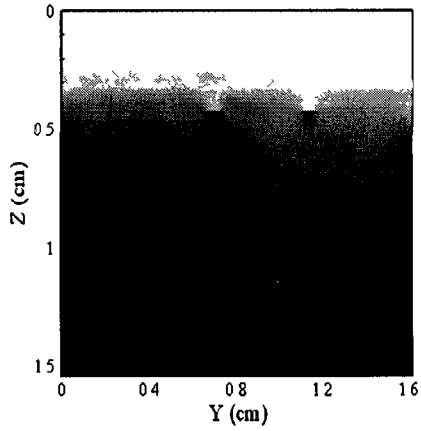
(a)
(Levels 2 (white) and 6, 7 (gray) of artery and levels 2, 6, 7 of vein (light black) appear)



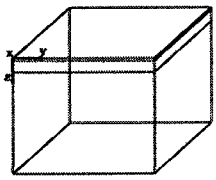
(b)
(35.4 °C (darkest) ≤ T ≤ 67.1 °C (brightest))



(c)
(36.3 °C (darkest) ≤ T ≤ 74.9 °C (brightest))

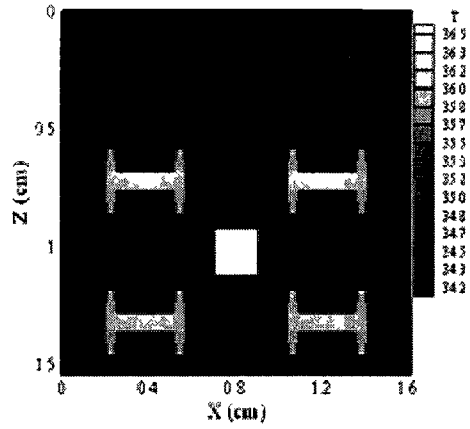


(d)
(37.1 °C (darkest) ≤ T ≤ 80.6 °C (brightest))

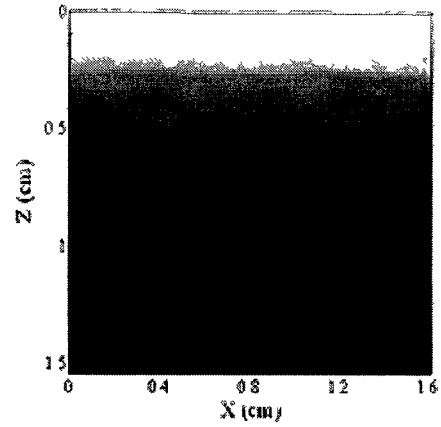


(e)

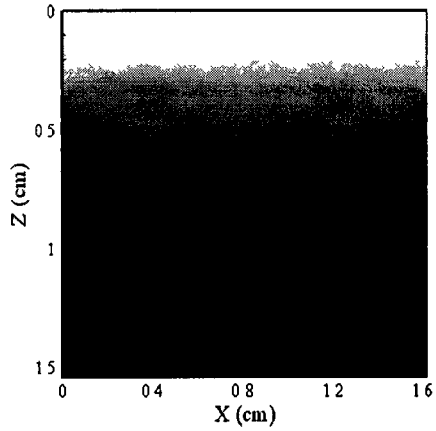
Figure 5.7 Contours of temperature distribution in the yz cross-section at x = 0.55 cm as shown in (e), when (a) t = 0 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.



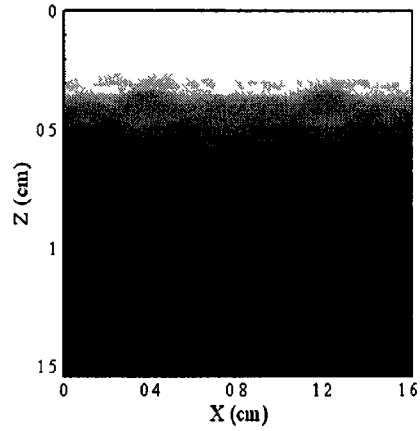
(a)
(Levels 1 (white), 5, 6 and 7 (grey) of artery and levels 4 and 7 of vein (light black) appear)



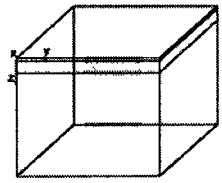
(b)
(36.6 °C (darkest) ≤ T ≤ 67.1 °C (brightest))



(c)
(36.9 °C (darkest) ≤ T ≤ 74.7 °C (brightest))

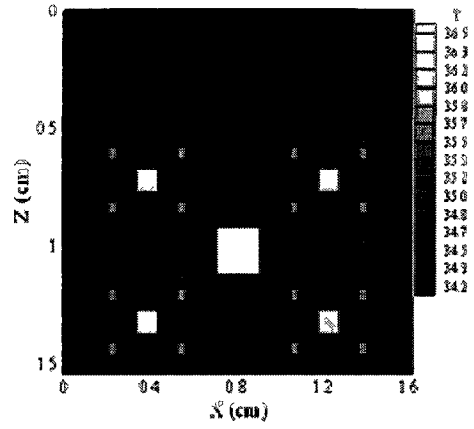


(d)
(37 °C (darkest) ≤ T ≤79.7 °C (brightest))

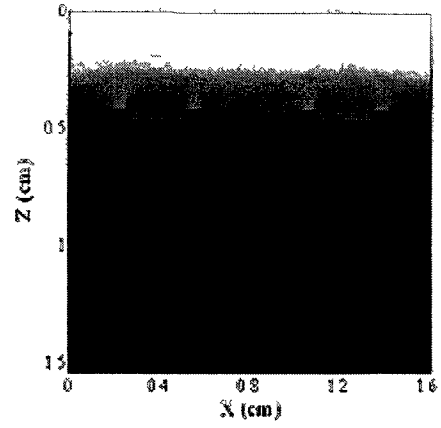


(e)

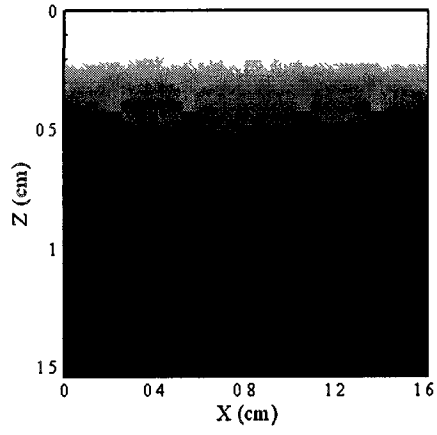
Figure 5.8 Contours of temperature distribution in the xz cross-section at y = 0.76 cm as shown in (e), when (a) t = 0 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.



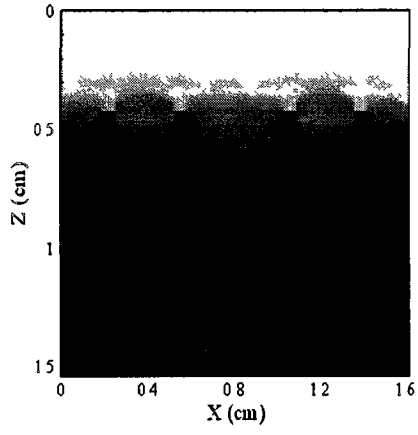
(a)
(Levels 1 and 4 (white), 7 (grey) of artery and levels 5, 6 and 7 of vein (light black) appear)



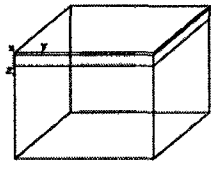
(b)
(36.7 °C (darkest) ≤ T ≤ 67.1 °C (brightest))



(c)
(36 9 °C (darkest) ≤ T ≤ 74.6 °C (brightest))



(d)
(37.1 °C (darkest) ≤ T ≤ 79.6 °C (brightest))



(e)

Figure 5.9 Contours of temperature distribution in the xz cross-section at y = 0.70 cm as shown in (e), when (a) t = 0 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.

Figures 5.10-5.13 show the contours of the skin burning distributions. Noted that values of $\Omega = 0.53, 1.0, 10^4$ are corresponding to the first, second, and third degree burn injuries, respectively [20], one may see from these figures that the skin appears to be a second degree burn at t = 200 s. With time increase, the tissue temperature increase, and the skin appears to be a third degree burning at t = 300 s and t = 400 s from skin surface. This indicates that if the tissue temperature reaches 74 °C as shown in Figures 5.4 – 5.9, the skin will then appear to be a third degree burning. Comparing with Figures 2.10 – 2.13, one may see that there is no difference between these two results regarding the area of the third degree burn injury. However, the areas of the first and second degree burn injuries are different from those obtained in [19] because of a more complex countercurrent vascular network in the case of this research.

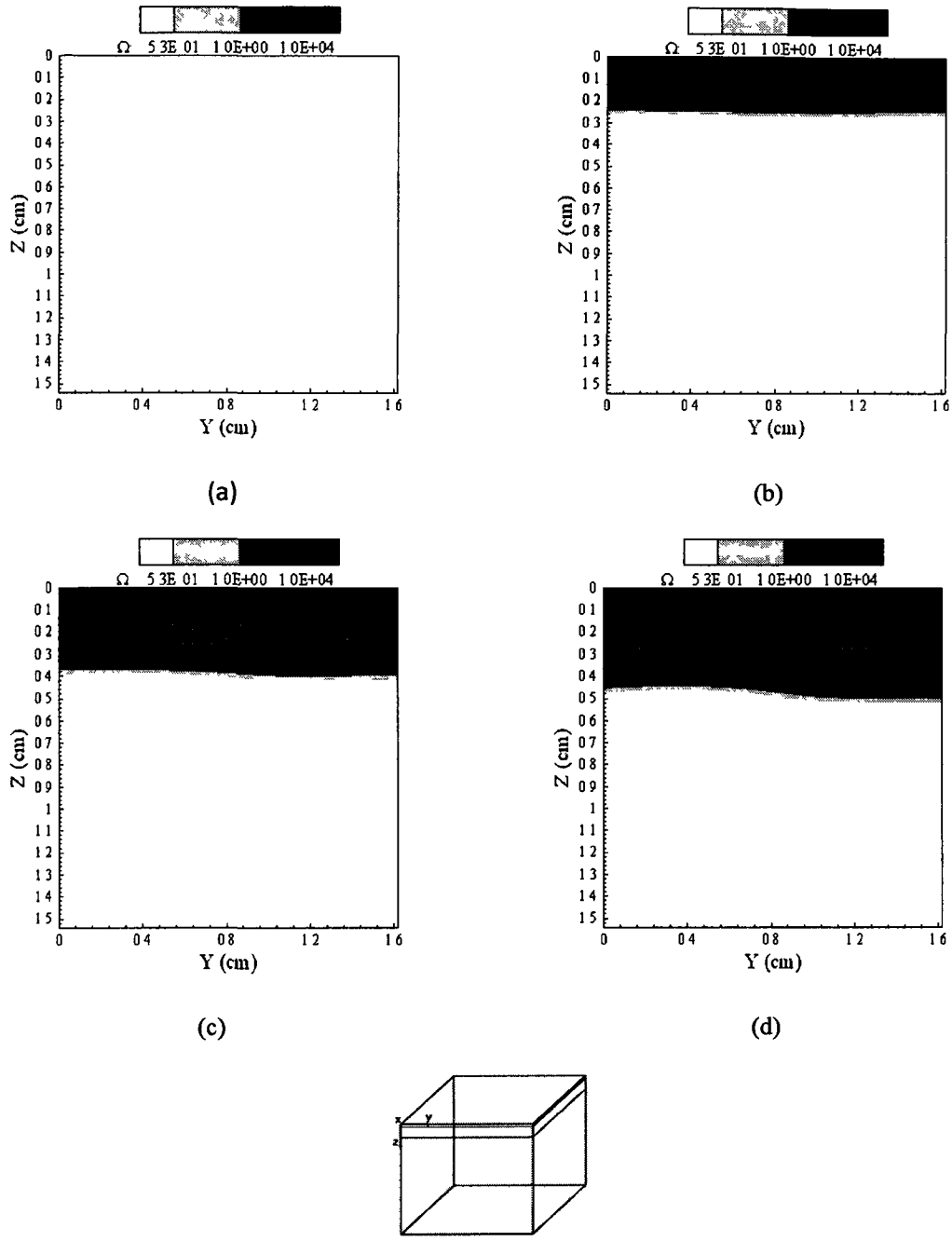


Figure 5.10 Contours of the skin burn distribution in the yz cross-section at x = 0.81 cm as shown in (e), when (a) t = 100 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.

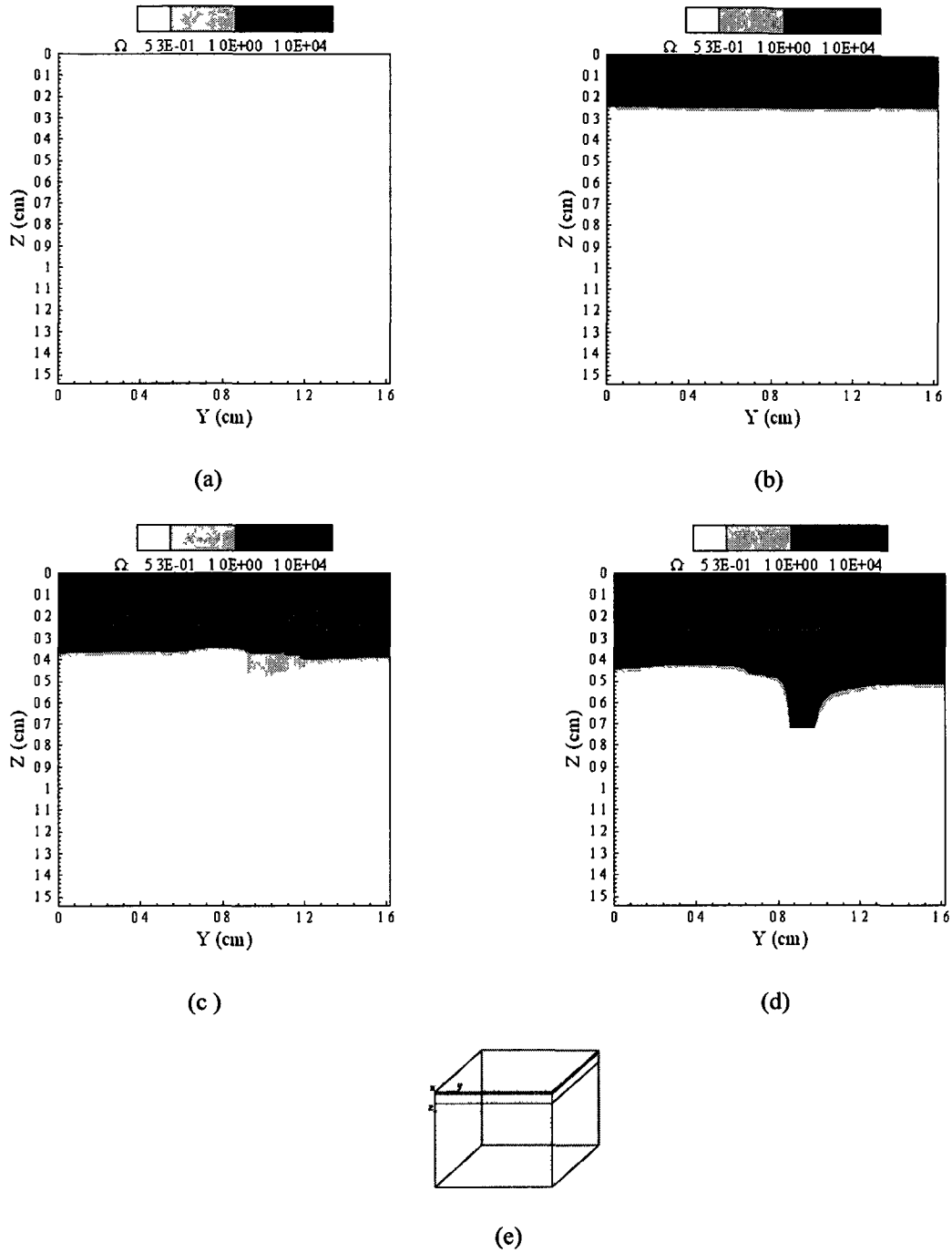


Figure 5.11 Contours of the skin burn distribution in the yz cross-section at x = 0.39 cm as shown in (e), where levels of 3, 4 and 5 arteries and vein appear, when (a) t = 100 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.

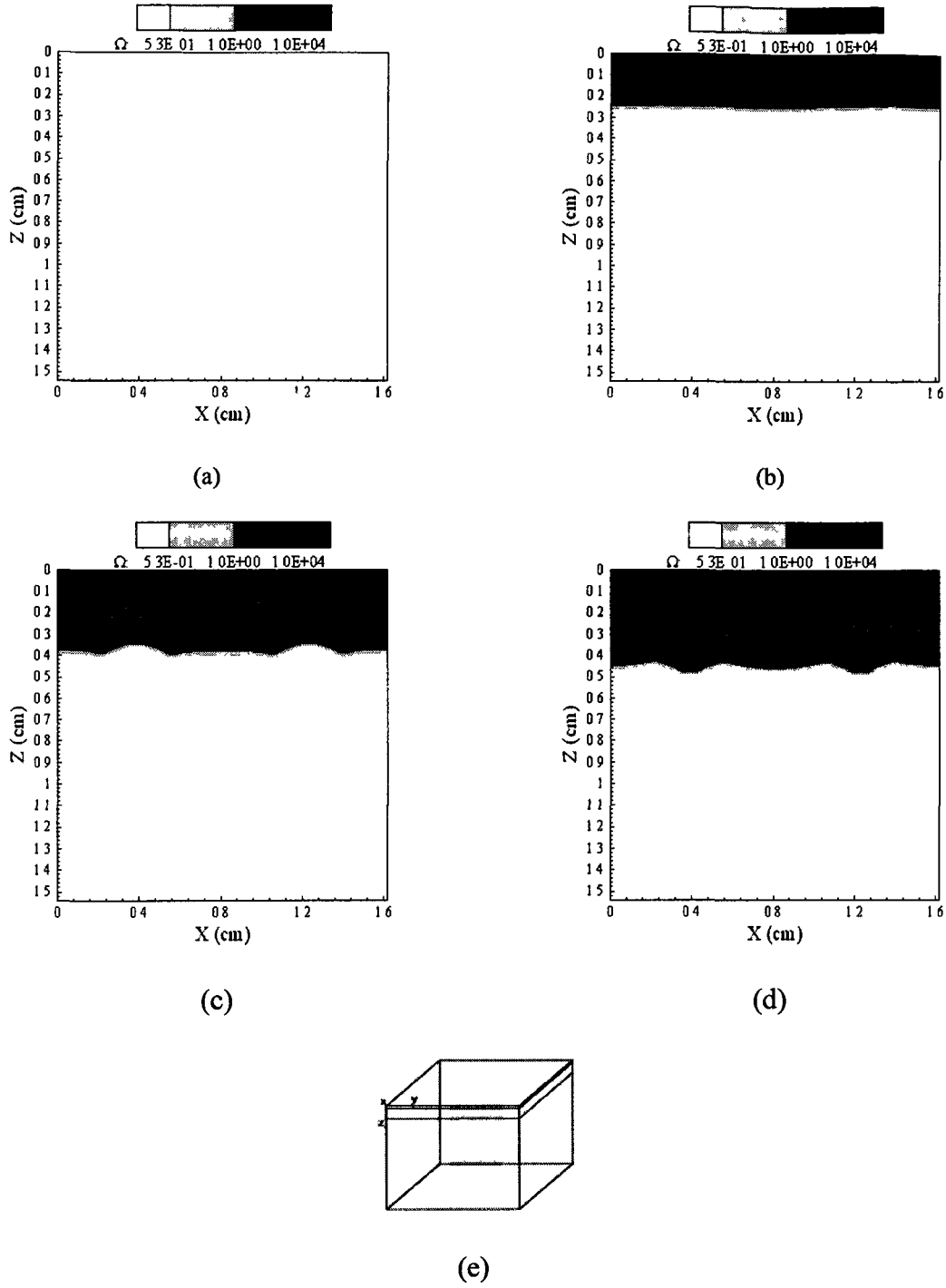


Figure 5.12 Contours of the skin burn distribution in the xz cross-section at y = 0.76 cm as shown in (e), where levels 1, 5, 6 and 7 of arteries and levels 4 and 7 of veins appear, when (a) t = 100 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.

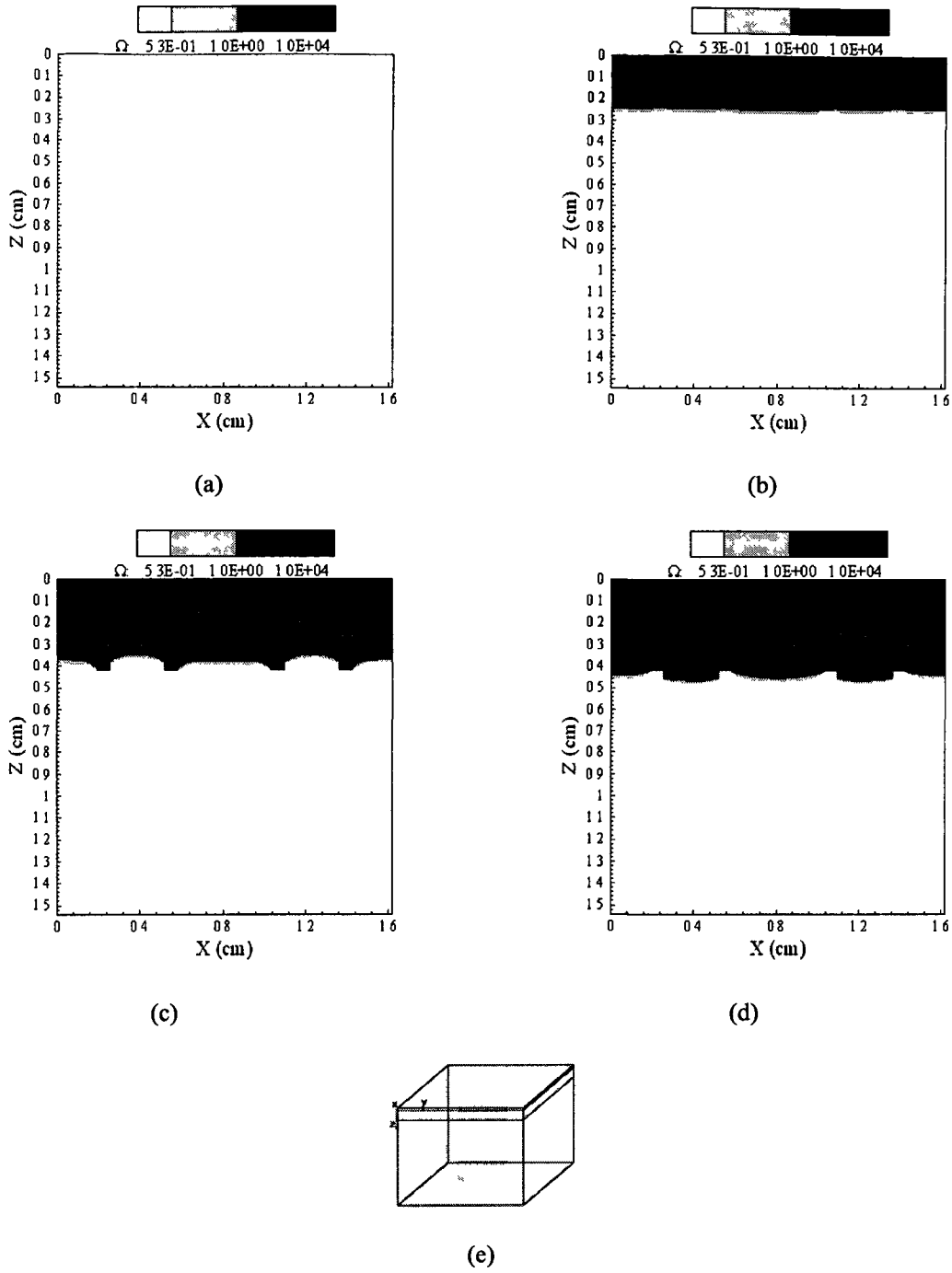


Figure 5.13 Contours of the skin burn distribution in the xz cross-section at y = 0.7 cm as shown in (e), where levels 1, 4 and 7 of arteries and levels 5, 6 and 7 of veins appear, when (a) t = 100 s, (b) t = 200 s, (c) t = 300 s, and (d) t = 400 s.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

This dissertation has developed a bio-heat transfer model for obtaining the temperature distribution and hence predicting thermal skin burning because of radiation heating. The skin tissue is considered to be a 3D triple-layer, consisting of epidermis, dermis and subcutaneous layers. A dendritic countercurrent vascular network is designed to embed into the subcutaneous layer, where the dimensions and blood flow rates are determined based on the constructal theory of multi-scale tree-shaped heat exchangers. The model takes into account the relatively large thermal relaxation time of biological tissue and the effects of high thermal radiation on such tissue using the Maxwell-Cattaneo thermal flux law in conjunction with the fourth power law. A finite difference scheme, together with the fourth-order Runge-Kutta method, is employed to solve the bio-heat transfer model, and hence to predict thermal damage in the skin structure. The finite difference scheme is proved to be unconditionally stable. The solution system is then solved by a preconditioned Richardson iteration, that is proved to be convergent. The bio-heat transfer model and its numerical method are then tested by a size of 1.62 cm by 1.62 cm by 1.542 cm skin tissue embedded with a seven-level countercurrent vascular network. As compared with the previous results in [19], the present results show that there is no difference between those found earlier regarding the area of the

third degree burn injury. However, the areas of the first and second degree burn injury are different from those obtained in W. Dai's paper [19], because of the more complex countercurrent vascular network that is used in the present model.

The numerical results show that heat transfer in living tissue is a highly heterogeneous and hierarchically organized medium. The blood vessels, arteries and vein played important roles in the distribution of heat transfer. These results can be used to explain live phenomena in humans. The exploratory approach developed in this dissertation could be used in future studies, for example, by considering a larger area of skin structure, or a tissue with tumors, as well as modeling such well documented effects of thermal damage as skin wrinkles and tissue shrinkage.

Table 1 and Table 2 show parameters of nomenclature on Appendix A. On Table 1, L_l (l = 1, 2, 3.) are the thickness of skin layer l. L_b^m ($m = 1, 2, \ldots, 7.$) are the length of the blood vessel in level m along the flowing direction of blood. NL_b^m and NW_b^m (m =1, 2,.., 7.) are the length and width of the cross-section of the mth level blood vessel. On Table 2, C_l and C_b^l (l = 1, 2, 3.) are the specific heat of tissue and blood in skin layer l. k_l (l = 1, 2, 3.) are the heat conductivity of skin layer l. ρ_l (l = 1, 2, 3.) are the density of skin layer l. W_b^l (l = 1, 2, 3.) are the blood perfusion rate in skin layer l. Table 3 shows the parameters used in Eq. 4.8, based on Takata's mode [20].

Table 1. Physical parameters used in computation.

Parameters	Values	Parameters	Values
$L_1(cm)$	0.008	*NZ (cm)*	1.542
$L_2(cm)$	0.2	$NL_b^1, NW_b^1(cm)$	0.2
$L_3(cm)$	1.334	$NL_b^2, NW_b^2(cm)$	0.16
$L_b^1(cm)$	1.0	$NL_b^3, NW_b^3(cm)$	0.12
$L_b^2(cm)$	0.72	$NL_b^4, NW_b^4(cm)$	0.1
$L_b^3(cm)$	0.5	$NL_b^5, NW_b^5(cm)$	0.08
$L_b^4(cm)$	0.36	$NL_b^6, NW_b^6(cm)$	0.06
$L_b^5(cm)$	0.26	$NL_b^7, NW_b^7(cm)$	0.04
$L_b^6(cm)$	0.18	$\Delta x(cm)$	0.01, 0.005
$L_b^7(cm)$	0.14	$\Delta y(cm)$	0.01, 0.005
NX, NY (cm)	1.62	$\Delta z(cm)$	0.001, 0.002

Table 2. Thermal parameters for a 3D skin structure [19], [29], [25], [18], [46]

Parameters	Values	Parameters	Values
α (W/°C · cm^2)	0.2	ω	1.0
$B_i = \alpha/k_3$	95.23	$\dot{P}(1/s)$	0.5× 10^{-3}
$C_1(J/g \cdot °C)$	3.6	$\rho_1\,(g/cm^3)$	1.2
$C_2(J/g \cdot °C)$	3.4	$\rho_2\,(g/cm^3)$	1.2
$C_3(J/g \cdot °C)$	3.06	$\rho_3\,(g/cm^3)$	1.0
$C_b^1(J/g \cdot °C)$	0.0	$\sigma(W/m^2K^4)$	5.669× 10^{-8}
$C_b^2(J/g \cdot °C)$	4.2	$\tau(s)$	20
$C_b^3(J/g \cdot °C)$	4.2	T_0(°C)	34
C_B	4.134	T_a(°C)	200
ϵ	0.9	T_{in}(°C)	37
$h(W/cm \cdot^2)$	0.001	$v_1\,(m/s)$	0.08
k_1(W/$cm \cdot$ °C)	0.0026	$W_b^1(g/cm^3 \cdot s)$	0.0
k_2(W/$cm \cdot$ °C)	0.0052	$W_b^2(g/cm^3 \cdot s)$	0.0005
k_3(W/$cm \cdot$ °C)	0.0021	$W_b^3(g/cm^3 \cdot s)$	0.0005

Table 3. Parameters used in Eq. (4.8) based on Takata's mode [20]

Temperature range (°C)	Activation energy ΔE $(J/K \cdot mol)$	Scaling factor ς $(1/s)$
$T \leq 50$	4.18×10^{8}	4.322×10^{64}
$T > 50$	6.69×10^{8}	6.69×10^{104}

REFERENCES

[1] P.J. Antaki, "Hotter than you think," *Mach Des*, July 13, 116, (1995).

[2] Bejan, *Shape and structure from engineering to nature,* Cambridge University Press, Cambridge, UK, (2000).

[3] A. Bejan, "The tree of convective heat streams: its thermal insulation function and the predicted 34-power relation between body heat loss and body size," *Int. J. Heat Mass Transfer*, vol. 44, 699, (2001).

[4] A. Bejan, S. Lorente, "Constructal theory of Generation of configuration in nature and engineering," *J. Appl. Phys*, vol. 100, 041301, (2006).

[5] R.C. Borkebak, "Heat transfer in biological systems," *Int. Rev. Gen. Exper. Zool.* vol. 2, 269, (1966).

[6] H. F. Bowman, E. G. Cravalho, Monty Woods, "Theory, measurement and application of properties of biomaterials," *Ann. Rev. Biophys. Bioeng.* vol. 4, 43, (1975).

[7] R.L. Burden, D.J. Faires, *Numerical analysis, 7th ed.*, Brooks/Cole, Chapter 5, (2001).

[8] C. Cattaneo, S.C. de Calore, *Atti del Seminar*, Mat. Fis. University, Modena, vol. 3, 83, (1948).

[9] I. Chatterjee and R. A. Adams, "Finite element thermal modeling of the human body under hyperthermia treatment for cancer," *Int. J. Com. Appls. Tech.*, vol. 7, 151, (1994).

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

[11] C.I. Christov, P.M. Jordan, "Heat conduction paradox involving second-sound propagation in moving media," *Phys. Rev. Lett*, 94, 54301 (2005).

[12] W. Dai, R. Nassar, "A finite difference scheme for solving the heat transport equation at the microscale," *Numer. Meth. Partial Diff. Eq.*, vol. 15, 597, (1999).

[13] W.Dai, R. Nassar, "A finite difference scheme for solving a three-dimensional heat transport equation in a thin film with microscale thickness," *Int. J. Numer. Meth. Engng.* vol 50, 1665, (2001).

[14] W. Dai, R. Nassar, "An unconditionally stable finite difference scheme for solving a 3D heat transport equation in a sub-microscale thin film," *J. Comp. Appl. Math.* vol. 145, 247, (2002).

[15] 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, vol. 2, 1650 (2003).

[16] W. Dai, H. Yu, R. Nassar, and T. Zhu, "A fourth-order compact finite difference scheme for solving a 1-D Pennes' bio-heat 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, 336, (2003).

[17] L. Zhang, W. Dai, R. Nassar, "A numerical modeling for optimizing laser power irradiating on a 3D triple layered cylindrical skin structure," *Numer. Heat Transfer, Part A*, vol. 48, 21, (2005).

[18] W. Dai, A. Bejan, X. Tang, L. Zhang, R. Nassar, "Optimal Temperature distribution in a 3D triple layered skin structure with embedded vasculature," *J. Appl. Phys.*, vol. 99, 104702, (2006).

[19] W. Dai, H. Wang, P.M. Jordan, R.E. Mickens, A. Bejan. "A Mathematical model for skin burn injury induced by radiation heating," *Int. J. Heat and Mass Transfe,* vol. 51, 5497, (2008).

[20] K.R. Diller, "Modeling of bioheat transfer processes at high and low temperature," in: Y. I Cho, J.P. Hartnett, T.F. IrvineJr. (Eds.), *Advanced in Heat Transfer, Academic Press*, New York, vol. 22, 157, (1992).

[21] L.P. Gartner, *Color Atlas of Histology, 3rd ed.*, Lippincott Williams & Wilkins, Philadelphia, (2000).

[22] F.C. Henriques, A.R. Mortiz, "Studies of thermal injury in the conduction of heat to and through skin and the temperature attained therein: a theoretical and experimental investigation," *Amer. J. Pathol*, vol. 23, 531, (1947).

[23] H.W. Huang, Convective thermal model formulation of a three dimensional vascular system with simplified blood flow paths: temperature distributions during hyperthermia, MS Thesis, University of Arizona, Tucson, AZ, (1992).

[24] H.W. Huang, C.L. Chan, and R.B. Roemer, "Analytical solutions of Pennes bioheat transfer equation with a blood vessel," *J. Biomech. Eng.* vol. 116, 208, (1994).

[25] H.W. Huang, Z.P. Chen, and R.B. Roemer, "A counter current vascular network model of heat transfer in tissues," *J. Biomech. Eng.* vol. 118, 120, (1996).

[26] S. G. Klemick, M. A. Jog, and P. S. Ayyaswamy, "Numerical evaluation of heat clearance properties of a radiatively heated biological tissue by adaptive grid scheme," *Numerical Heat Transfer, Part A*, vol. 31, 451, (1997).

[27] C. T. Liauh and R. B. Roemer, "A semilinear state and arameter estimation algorithm for inverse hyperthermia problems," *J. Biomechanical Engineering*, vol. 115, 257, (1993).

[28] J. Liu, Z. Ren, and C. Wang, "Interpretation of living tissue's temperature oscillations by thermal wave theory," *Chinese Science Bulletin*, vol. 40, 1493, (1995).

[29] J. Liu, X. Chen, and L.X. Xu, "New thermal wave aspects on burn evaluation of skin subjected to instantaneous heating," *IEEE Trans. Biomed. Eng.*, vol. 46, 420, (1999).

[30] J. Liu, "Preliminary survey on the mechanisms of the wave-like behaviors of heat transfer in living tissues," *Forsch. Ingenieurwesen*, vol. 66, 1, (2000).

[31] J. Liu and L. X. Xu, "Boundary information based diagnostics on the thermal states of biological bodies," *Int. J. Heat Mass Transfer,* vol. 43, 2827, (2000).

[32] L.T. Lorimer, *The human body*, Reader's Digest, New York, (1999).

[33] 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, 167, (1998).

[34] I. A. Lubashevsky, and Vasyl Gafiychuk, *Analysis of the optimality principles responsible for vascular network architectonics*, arXiv:adap-org/9909003 vol. 1 Sep (1999).

[35] I. A. Lubashevsky, V. V. Gafiychuk, and B. Y. Datsko, *Anomalous properties of heat diffusion in living tissue caused by branching artery network. Qualitative description*, arXiv:cond-mat/0201057 vol. 5 Jan (2002).

[36] E. Majchrzak, and B. Mochnacki, "Numerical model of heat transfer between blood vessel and biological tissue," *Comput. Assist. Mech. Eng. Sci.*, vol. 6, 439, (1999).

[37] G. T. Martin and H. F. Bowman, "The temperature distribution in laser irradiated tissue with blood perfusion," *ASME Heat Transfer Division*, vol. 126, 97, (1989).

[38] K. Mitra, S. Kumar, A. Vedevarz, and M. K. Moallemi, "Experimental evidence of hyperbolic heat conduction in processed meat," *Trans. ASME J. Heat Transfer*, vol. 117, 568, (1995).

[39] C. D. Murray, "The Physiological principle of minimal work in the vascular system and the cost of blood-volume," *Proceeding of the National Academy of Sciences,* vol. 12, 207, (1926).

[40] 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, 197, (1999).

[41] H.H. Pennes, "Analysis of tissue and arterial temperature in the resting human forearm," *J. Appl. Physiol.* vol. 1, 93, (1948).

[42] H.W.Huang, Z.P. Chen, and R.B. Roemer, "A counter current vascular network model of heat transfer in tissues," *J. Biomech, Eng*, vol. 118, 120, (1996).

[43] A.K. da Silva, S. Lorente, and A. Bejan, "Constructal multi-scale treeshaped heat exchangers," *J. Appl. Phys.*, vol. 96, 1709, (2004).

[44] C. Sturesson, and A. Andersson-Engels, "A mathematical model for predicting the temperature distribution in laser-induced hyperthermia: experimental evaluation and applications," *Phys. Med. Biol.*, vol. 40, 2037, (1995).

[45] J. Sun, A. Zhang, and L. X. Xu, "Evaluation of alternate cooling and heating for tumor treatment," *Int. J. Heat Mass Transfer*, vol. 51, 5478, (2008).

[46] X. Tang, W. Dai, R. Nassar, and A. Bejan, "Optimal temperature distribution in a 3D triple layered skin structure embedded with artery and vein vasculature," *Numer. Heat Transfer, Part A*, vol. 50, 809, (2006).

[47] D'A. W. Thompson, *On Growth and Form*, (Cambridge University Press, Cambridge, UK, 1942).

[48] S. Xiang and J. Liu, "Comprehensive evaluation on the heating capacities of four typical whole body hyperthermia strategies via compartmental model," *Int. J. Heat Mass Transfer,* vol. 51, 5497, (2008).

[49] L. Zhang, W. Dai, and R. Nassar, "A Numerical modeling for optimizing laser power irradiating on a 3D triple layered cylindrical skin structure," *Numer. Heat Transfer, Part A*, vol. 48, 21, (2005).

[50] L. Zhang, W. Dai, and R. Nassar, "A numerical method for obtaining an optimal temperature distribution in a 3D triple-layered cylindrical skin structure embedded with a blood vessel," *Numer. Heat Transfer, Part A*, vol. 49, 437, (2006).

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

APPENDIX A

NOMENCLATURE

B_i ..... Biot number

C_l, C_b^l ..... specific heat of tissue and blood in layer *l*

C_B ..... heat capacity of blood

F_m ..... area of cross-section in the *m*th level vessel

h ..... heat convection coefficient

k_l ..... heat conductivity of layer *l*

L_l ..... thickness of layer *l*

L_b^m ..... length of the blood vessel in level *m* along the flowing direction of blood

M_k ..... main flow of blood in the *k*th level vessel

N_x, N_y, N_l^z ..... numbers of grid points in the *x, y, z* directions, respectively

NX, NY, NZ ..... lengths of the skin structure in the *x, y, z* directions, respectively

NL_b^m, NW_b^m ..... length and width of the cross-section of the *m*th level vessel

P ..... vessel periphery

$\dot{P}$ ..... blood flow rate

$(s_l)_{ijk}^n$ ..... numerical solution of function S_l

T_b^m, T_l, T_w^m ..... temperatures in blood, tissue, and vessel wall, respectively

T_{in}, T_{out} ..... temperatures of blood at entrance and exit, respectively

T_a ..... ambient temperature

t ..... time

u_{ijk}^n ..... numerical solution of temperature of tissue

u_b^m ..... numerical solution of temperature of blood in the *m*th level vessel

v_m ..... velocity of blood flow in the *m*th level vessel

W_b^l ..... blood perfusion rate in layer *l*

x, y, z ..... Cartesian coordinates

α ..... heat transfer coefficient between blood and tissue

δ_x^2 , δ_y^2 , δ_z^2 ..... second-order finite difference (FD) operators

Δt ..... time increment used in calculating heat transfer

$\Delta x, \Delta y, \Delta z$... mesh sizes of FD scheme for bio-heat transfer model in the x ,y, z directions

ε ..... emissivity

ω ..... relaxation factor

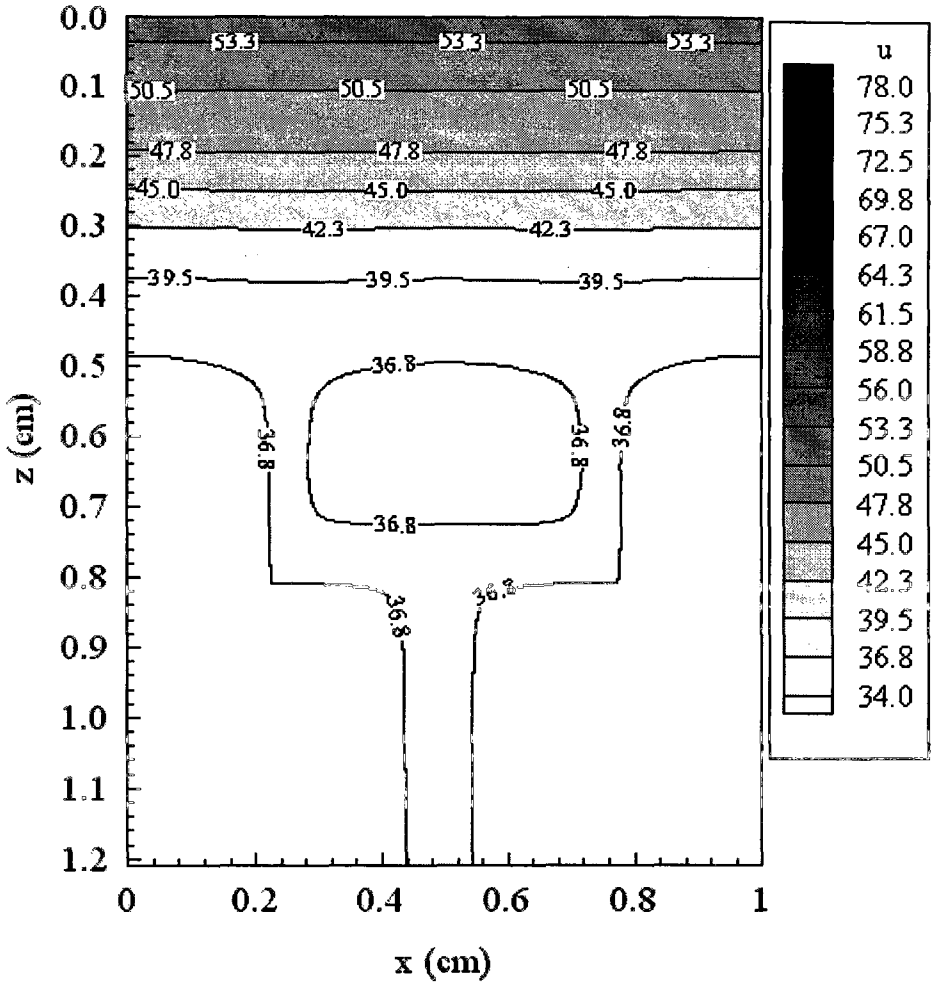
ρ_l ..... density of layer l

σ ..... Stefan-Boltzmann constant

τ ..... thermal lag time

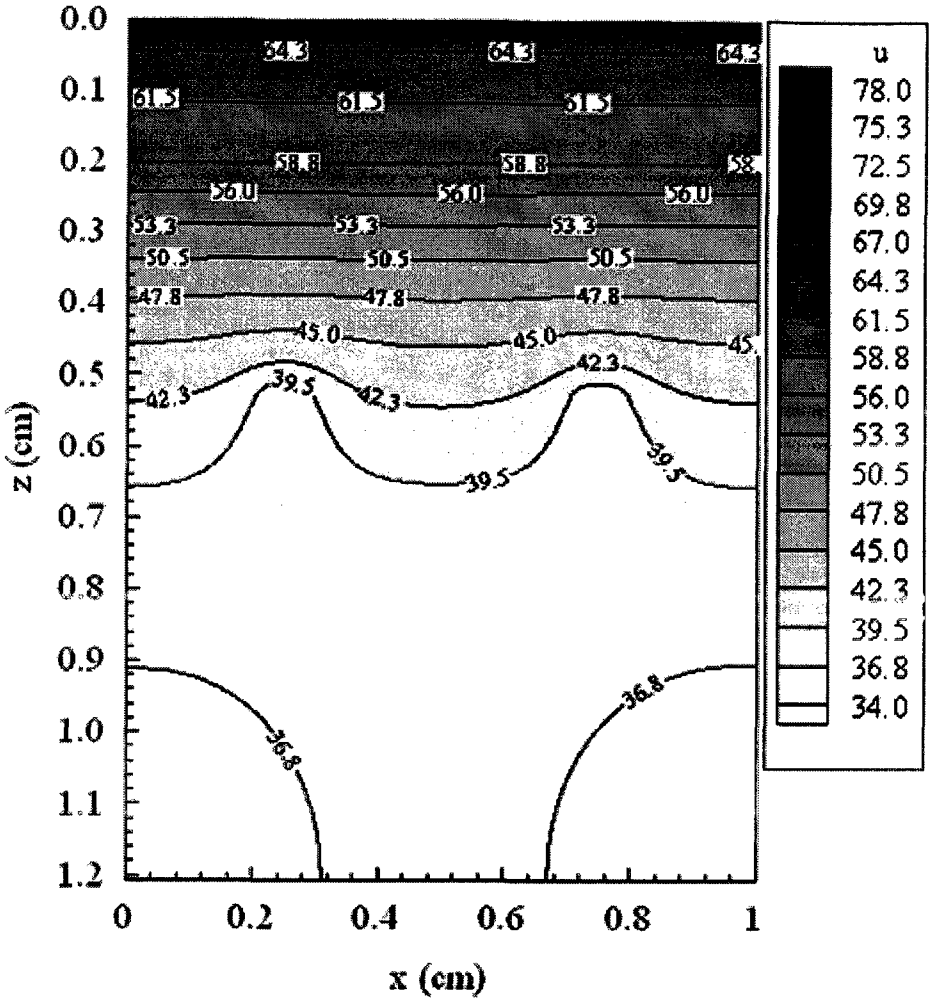
APPENDIX B

FIGURE 2.6 TO FIGURE 2.13



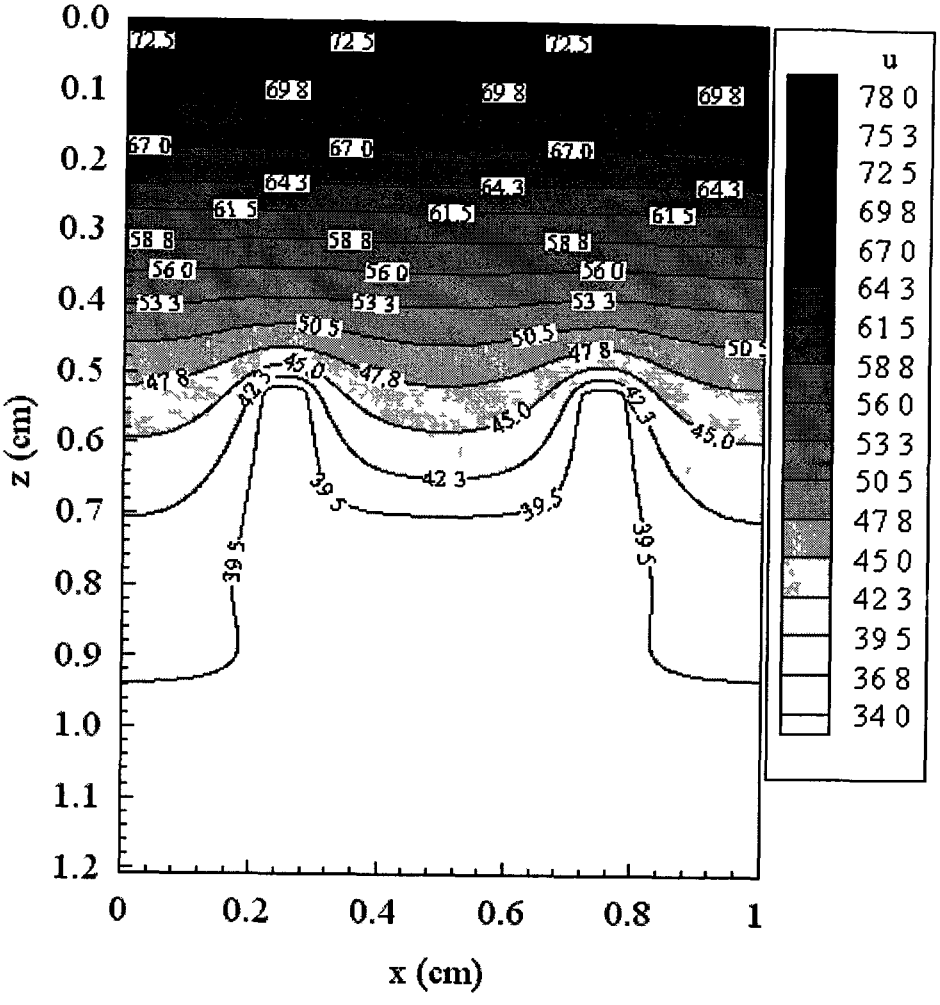
(a)

Figure 2.6.a Contours of the temperature distributions in the xz-cross-section at y = 0.4 cm, where the artery is located, at various times: (a) t = 100 s [19].



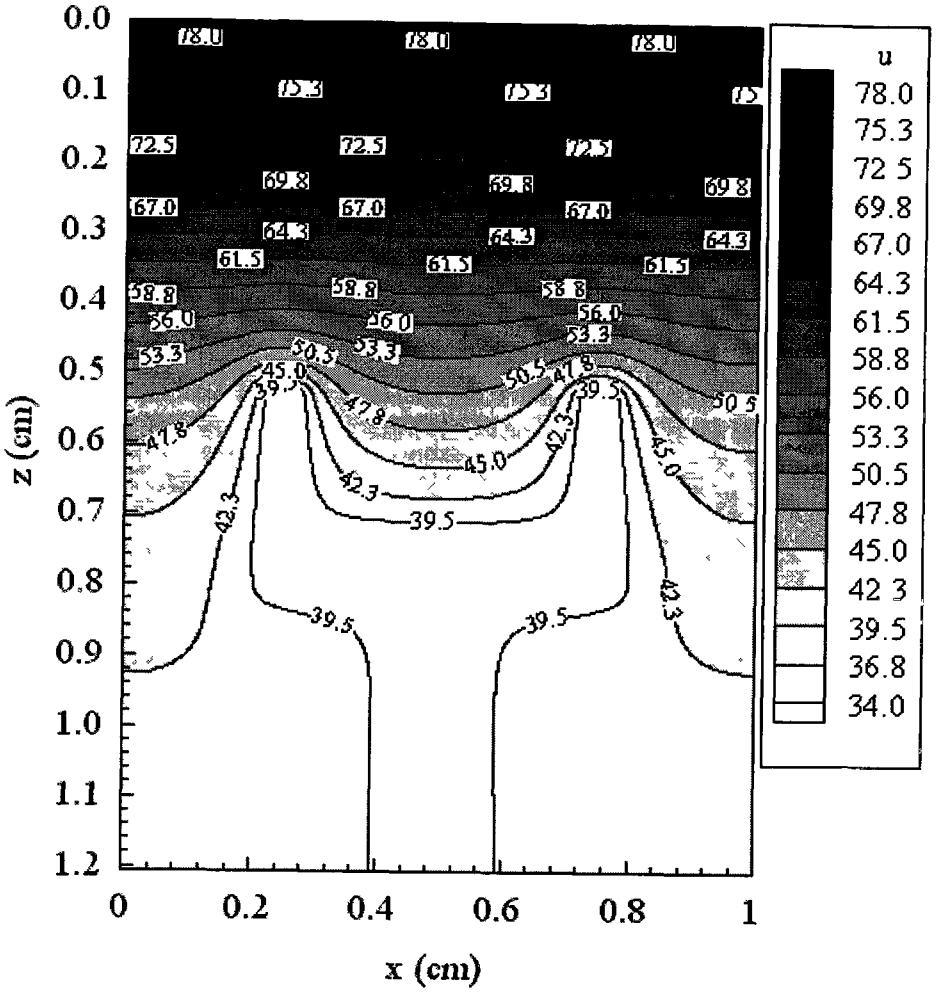
(b)

Figure 2.6.b Contours of the temperature distributions in the xz-cross-section at y = 0.4 cm, where the artery is located, at various times: (b) t = 200 s [19].



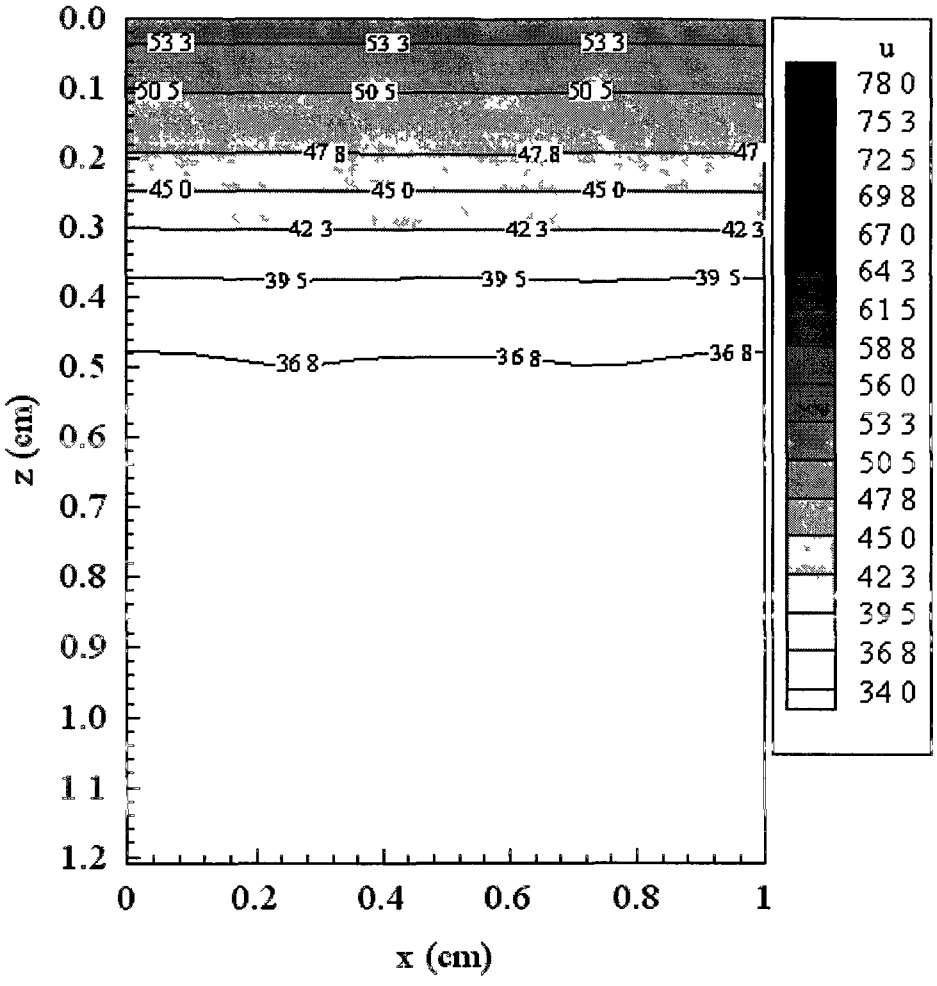
(c)

Figure 2.6.c Contours of the temperature distributions in the xz-cross-section at y = 0.4 cm, where the artery is located, at various times: (c) t = 300 s [19].



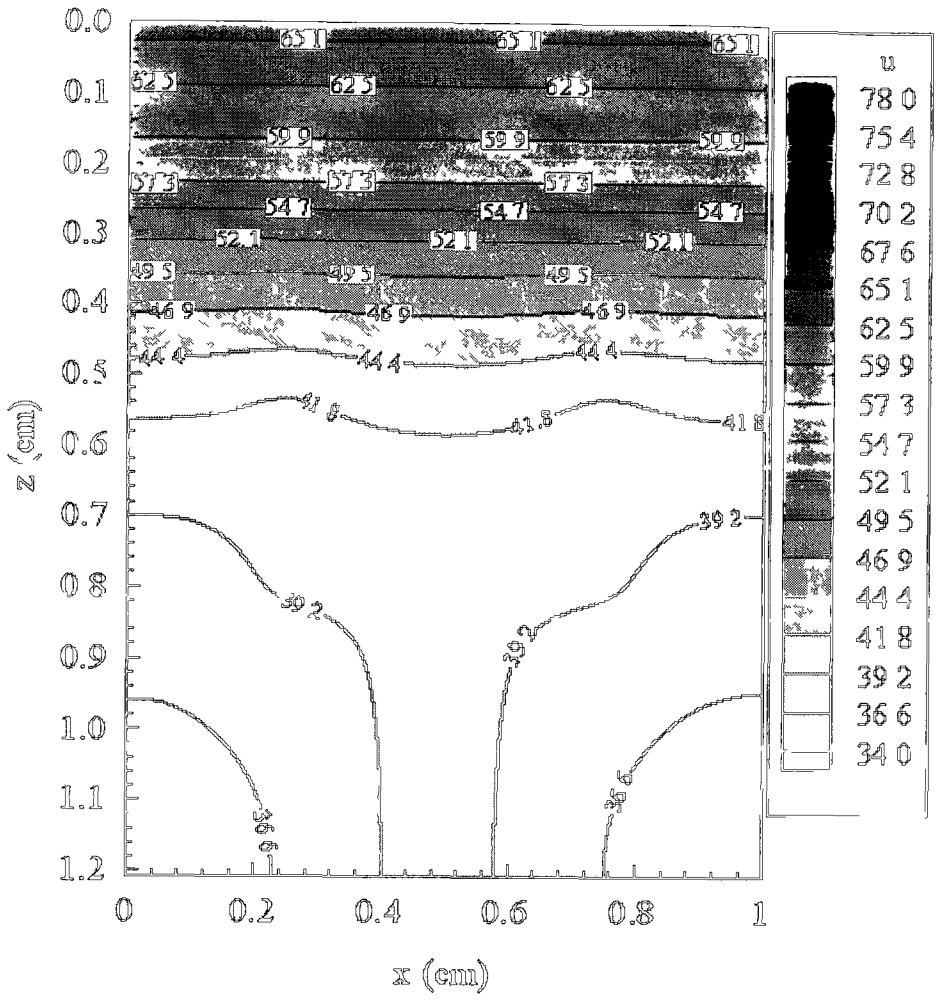
(d)

Figure 2.6.d Contours of the temperature distributions in the xz-cross-section at y = 0.4 cm, where the artery is located, at various times: (d) t = 400 s [19].



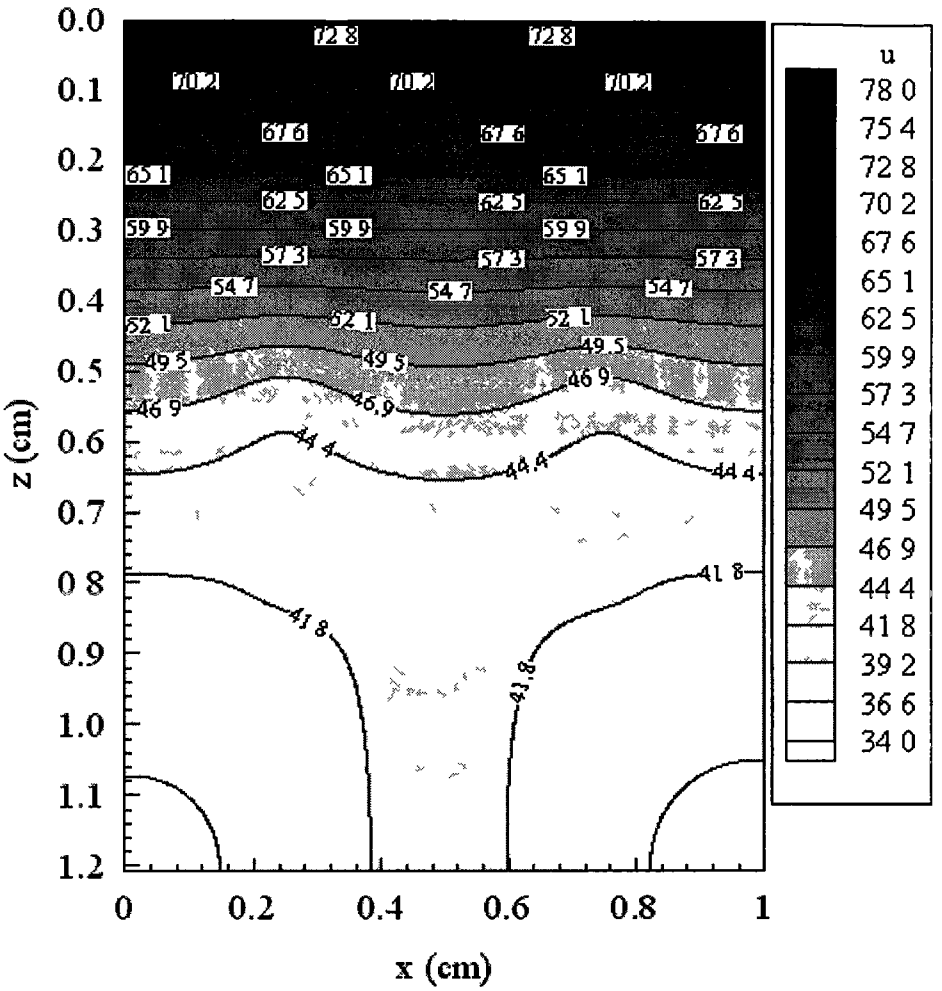
(a)

Figure 2.7.a Contours of the temperature distributions in the xz-cross-section at y = 0.5 cm at various times: (a) t = 100 s [19].



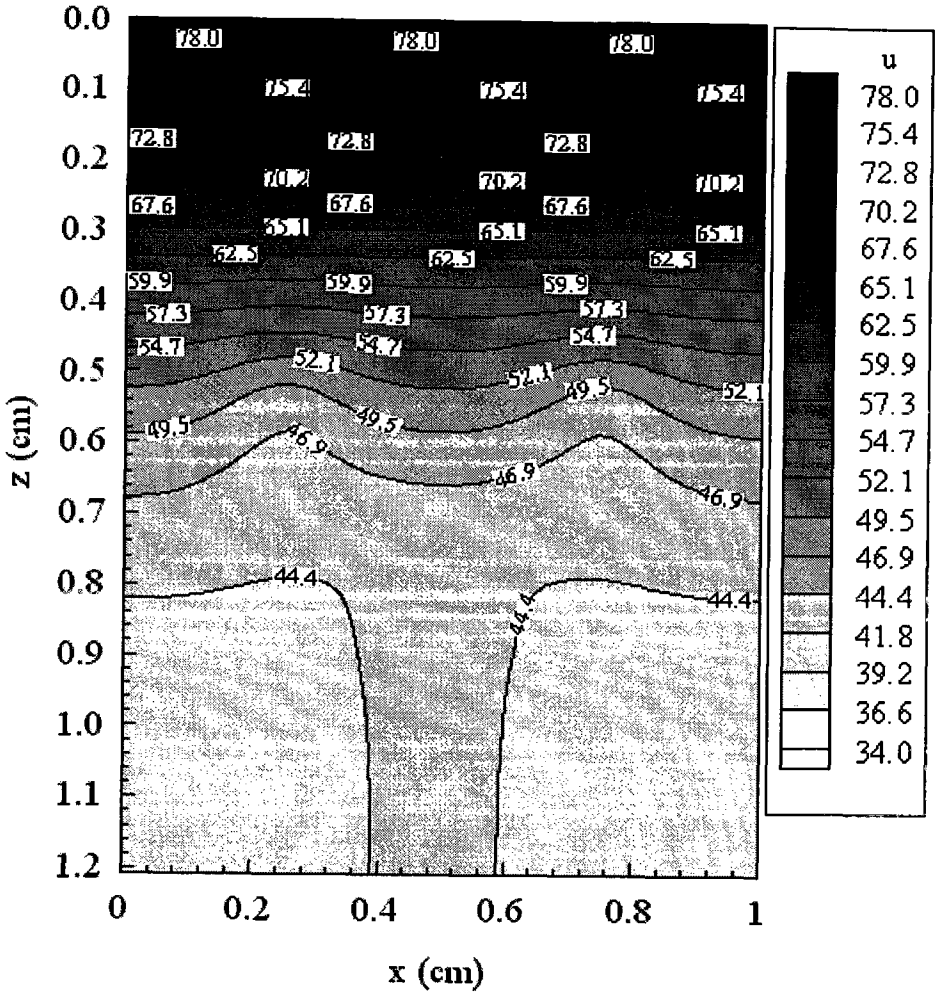
(b)

Figure 2.7.b Contours of the temperature distributions in the xz-cross-section at y = 0.5 cm at various times. (b) t = 200 s [19].



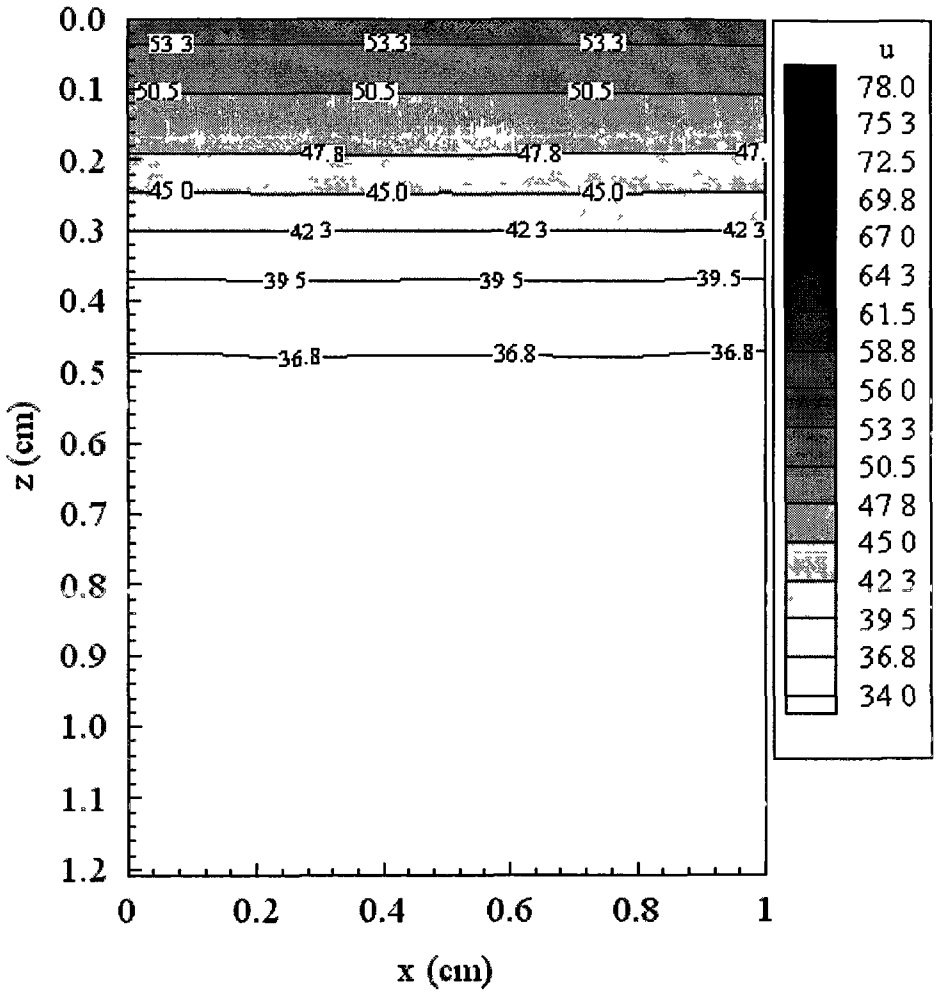
(c)

Figure 2.7.c Contours of the temperature distributions in the xz-cross-section at y = 0.5 cm at various times: (c) t = 300 s [19].



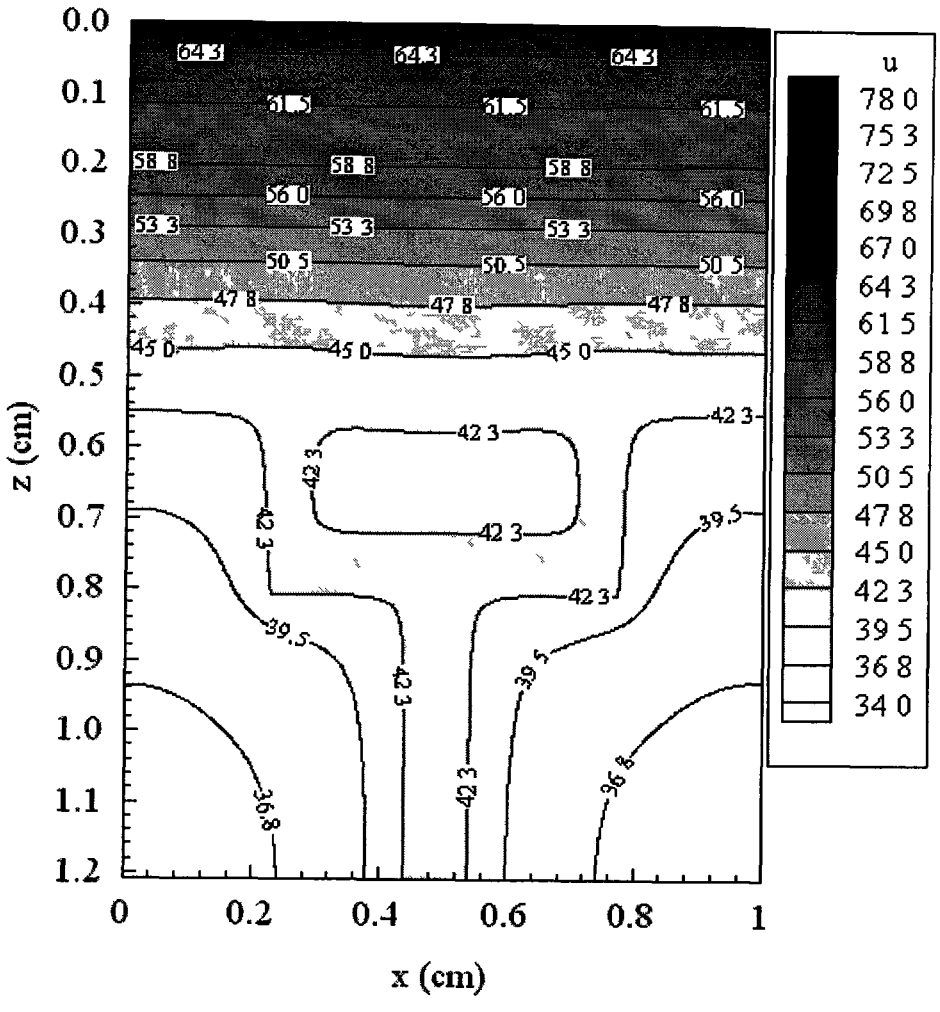
(d)

Figure 2.7.d Contours of the temperature distributions in the xz-cross-section at y = 0.5 cm at various times: (d) t = 400 s [19].



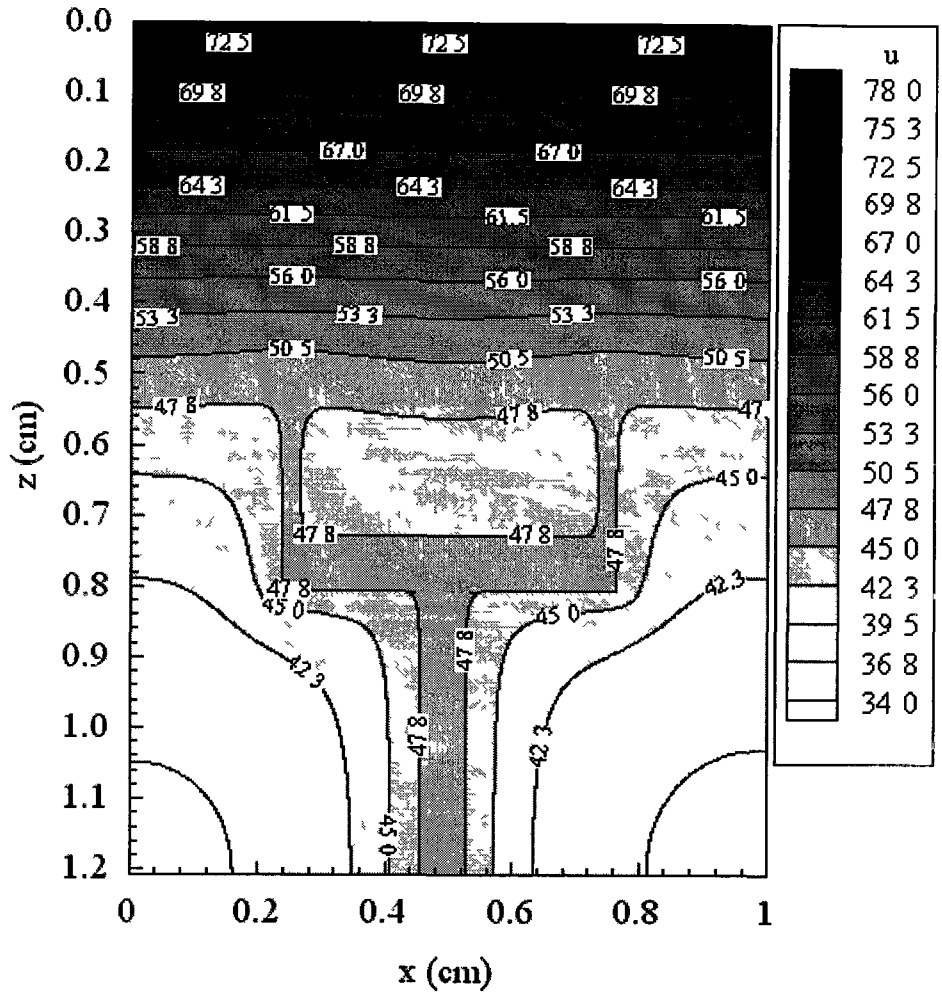
(a)

Figure 2.8.a Contours of the temperature distributions in the xz-cross-section at y = 0.56 cm, where the vein is located, at various times: (a) t = 100 s [19].



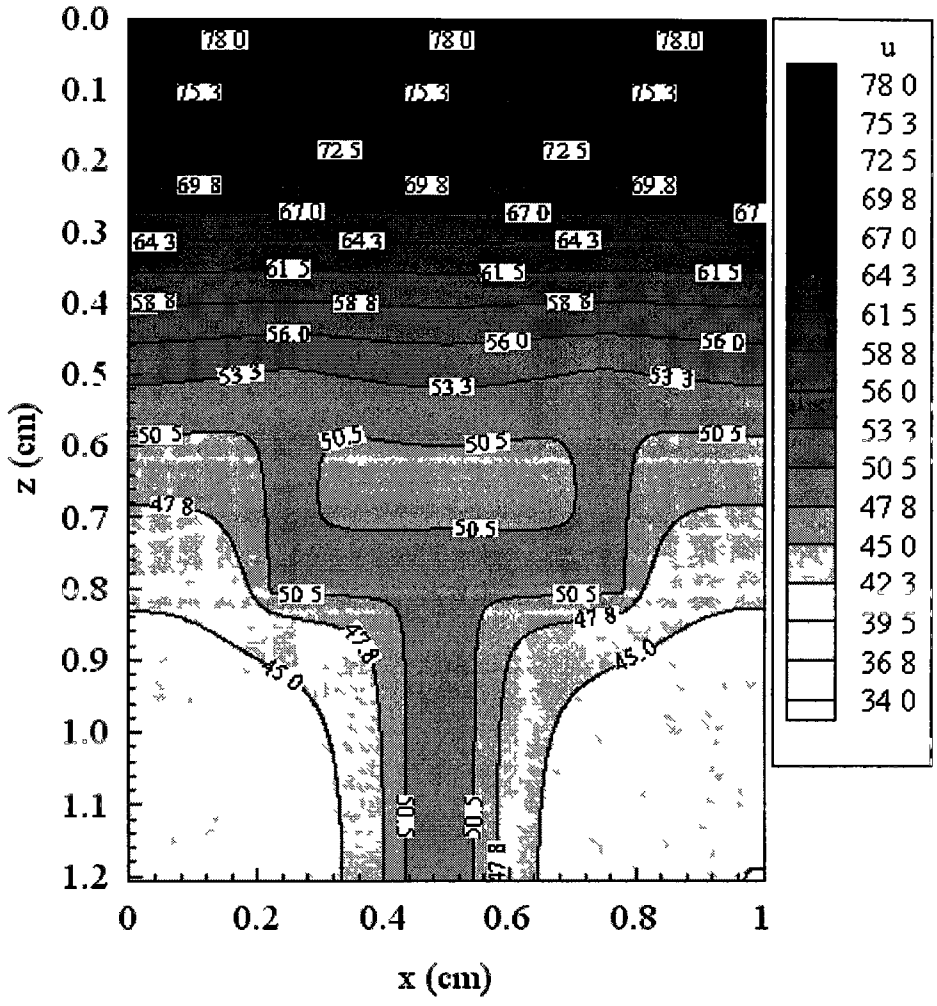
(b)

Figure 2.8.b Contours of the temperature distributions in the xz-cross-section at y = 0.56 cm, where the vein is located, at various times: (b) t = 200 s [19].



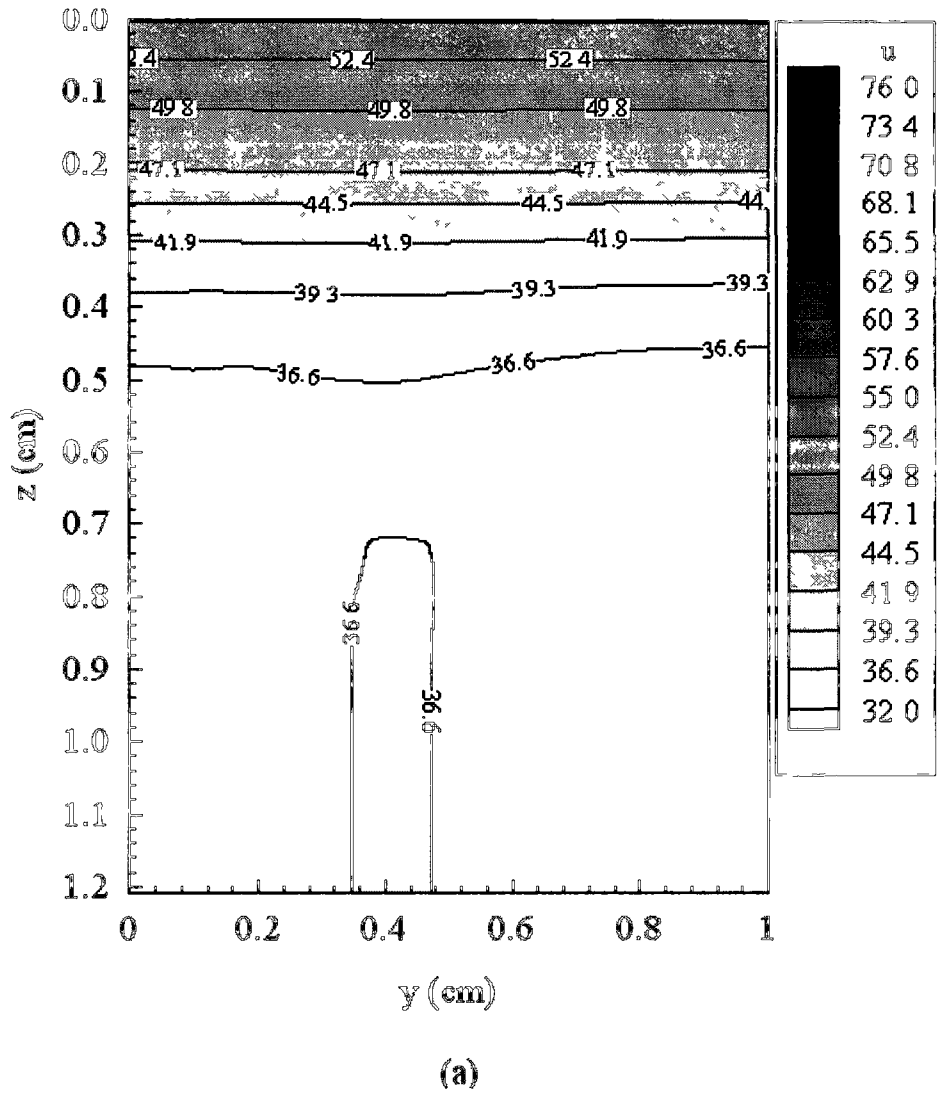
(c)

Figure 2.8.c Contours of the temperature distributions in the xz-cross-section at y = 0.56 cm, where the vein is located, at various times: (c) t = 300 s [19].



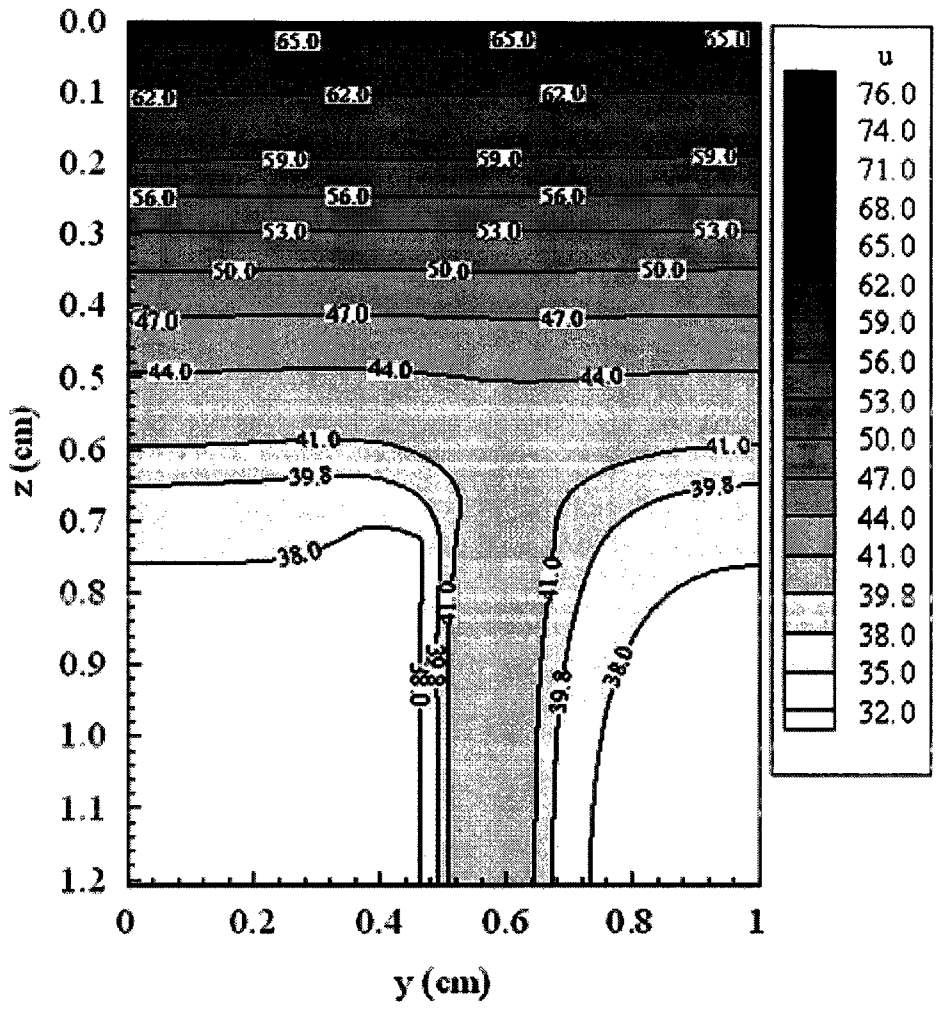
(d)

Figure 2.8.d Contours of the temperature distributions in the xz-cross-section at y = 0.56 cm, where the vein is located, at various times: (d) t = 400 s [19].



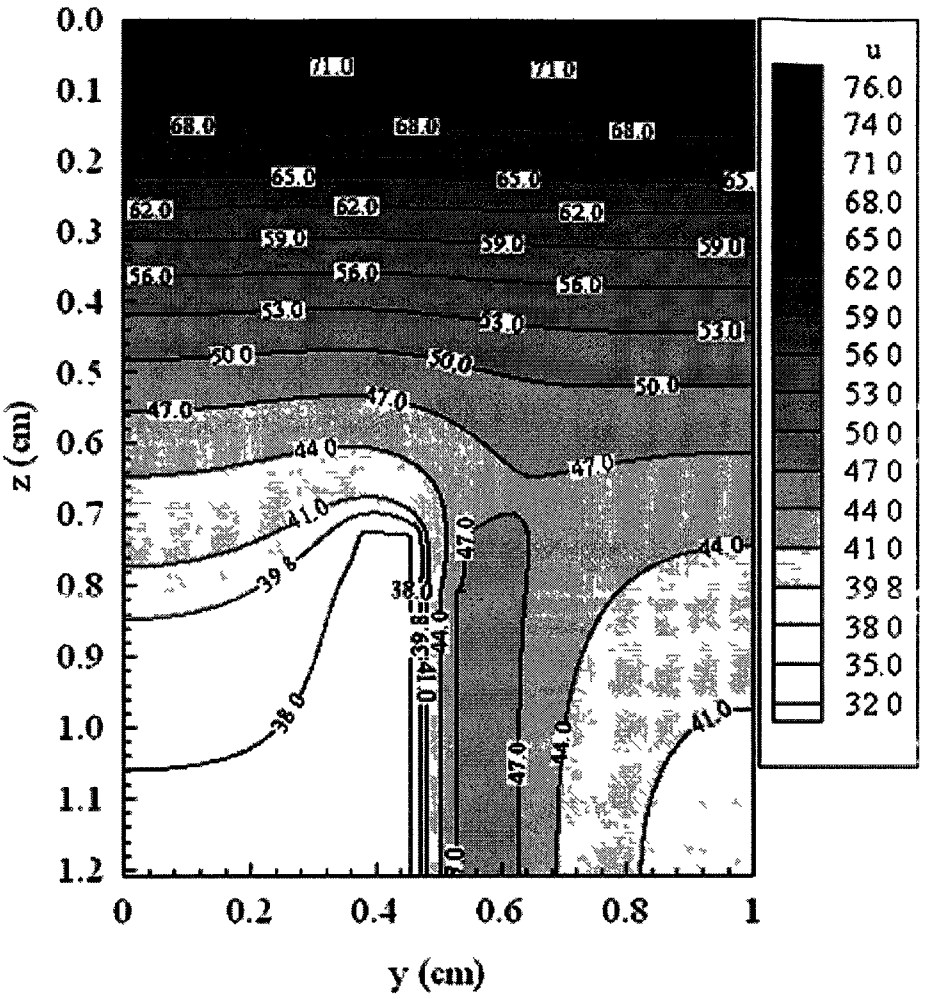
(a)

Figure 2.9.a Contours of the temperature distributions in the yz-cross-section at x = 0.5 cm at various times: (a) t = 100 s [19].



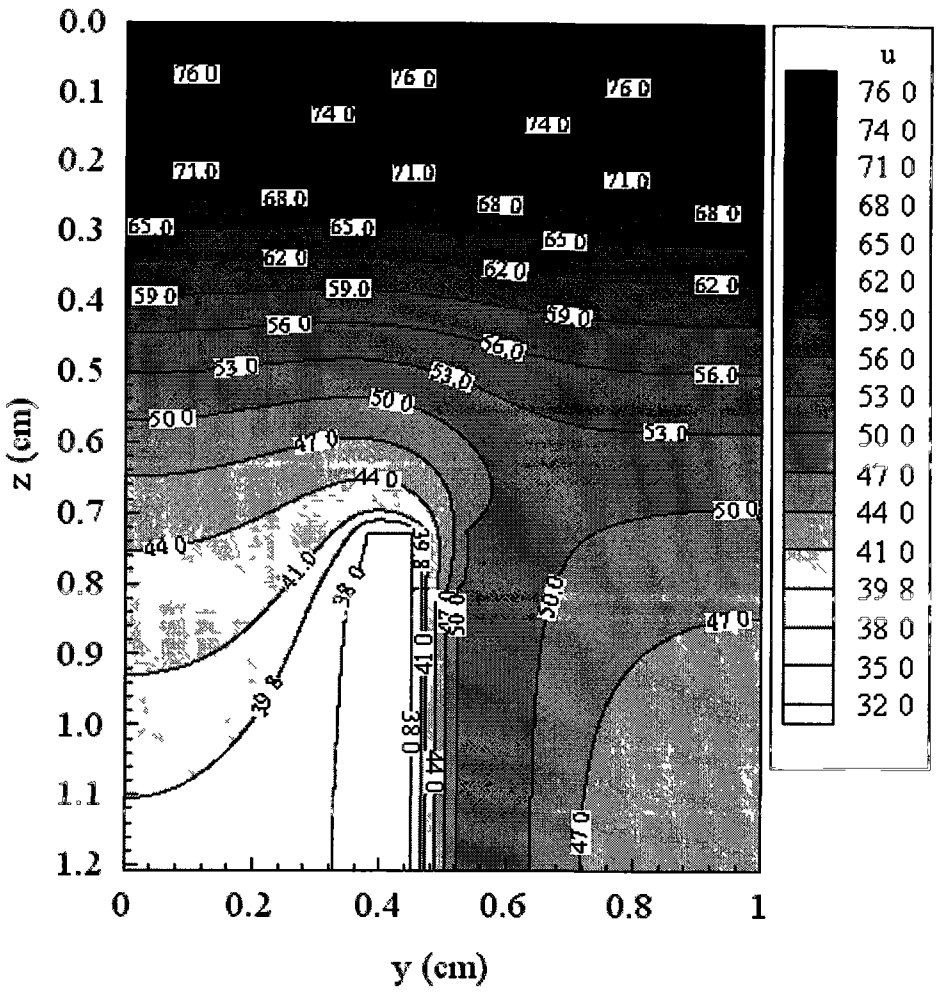
(b)

Figure 2.9.b Contours of the temperature distributions in the yz-cross-section at x = 0.5 cm at various times: (b) t = 200 s [19].



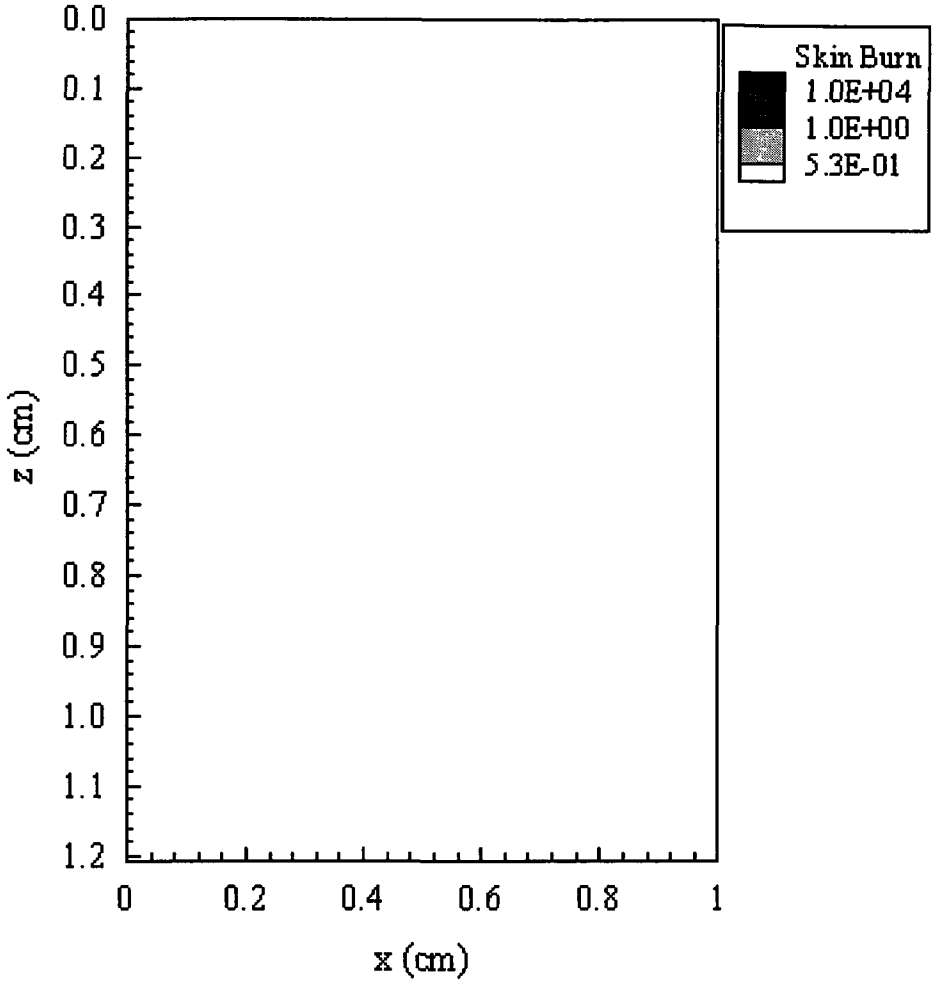
(c)

Figure 2.9.c Contours of the temperature distributions in the yz-cross-section at x = 0.5 cm at various times: (c) t = 300 s [19].



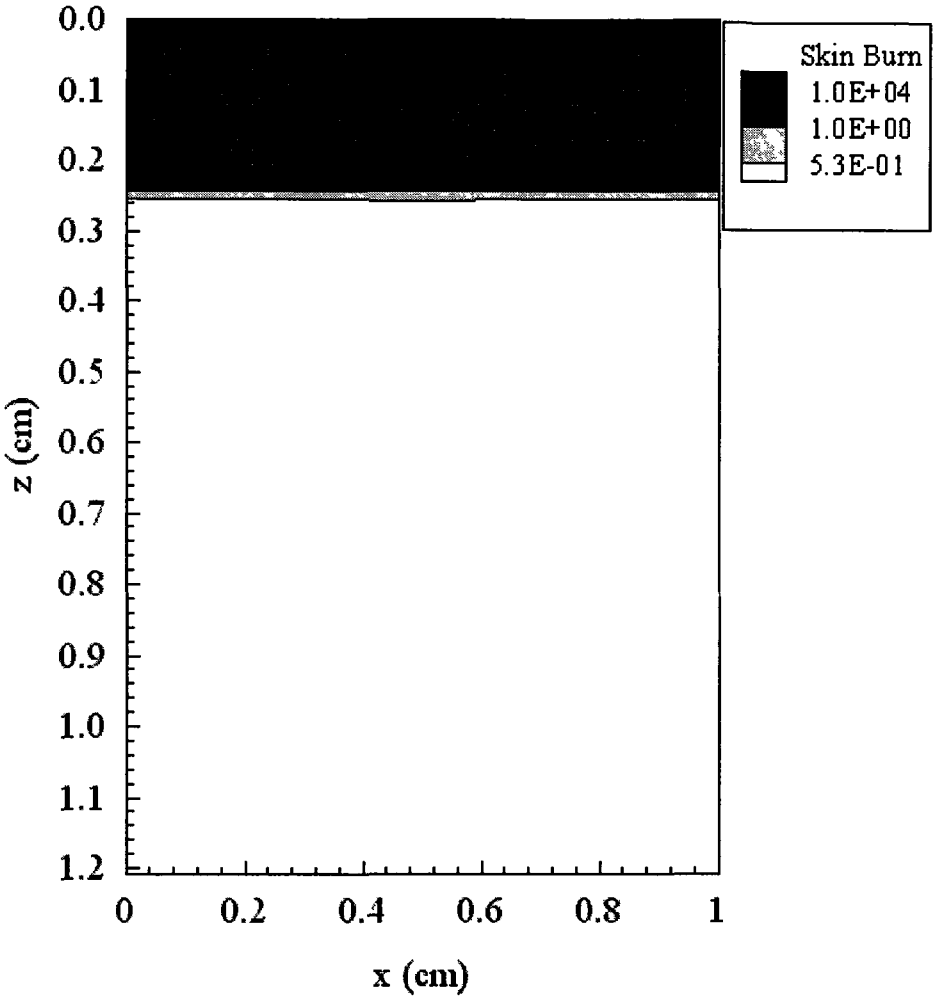
(d)

Figure 2.9.d Contours of the temperature distributions in the yz-cross-section at x = 0.5 cm at various times: (d) t = 400 s [19].



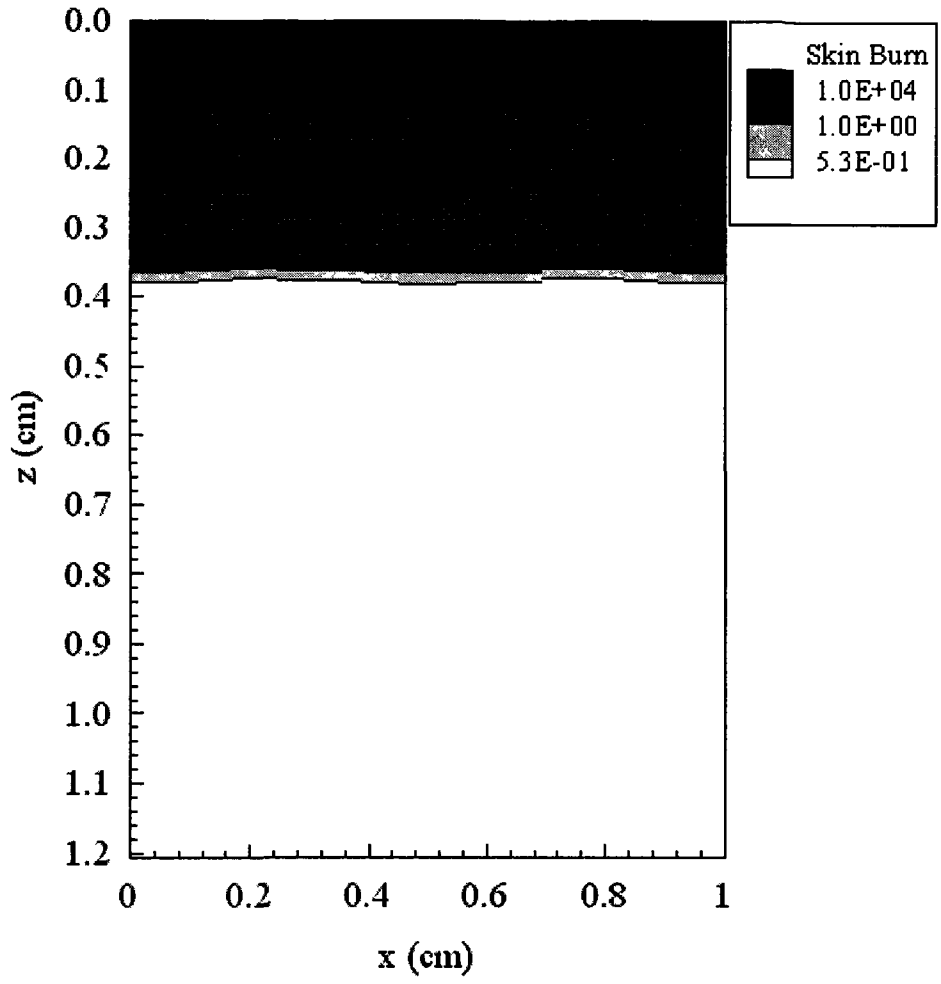
(a)

Figure 2.10.a Contours of the skin burn distributions in the xz-cross-section at y = 0.4 cm, where the artery is located, at various times: (a) t = 100 s [19].



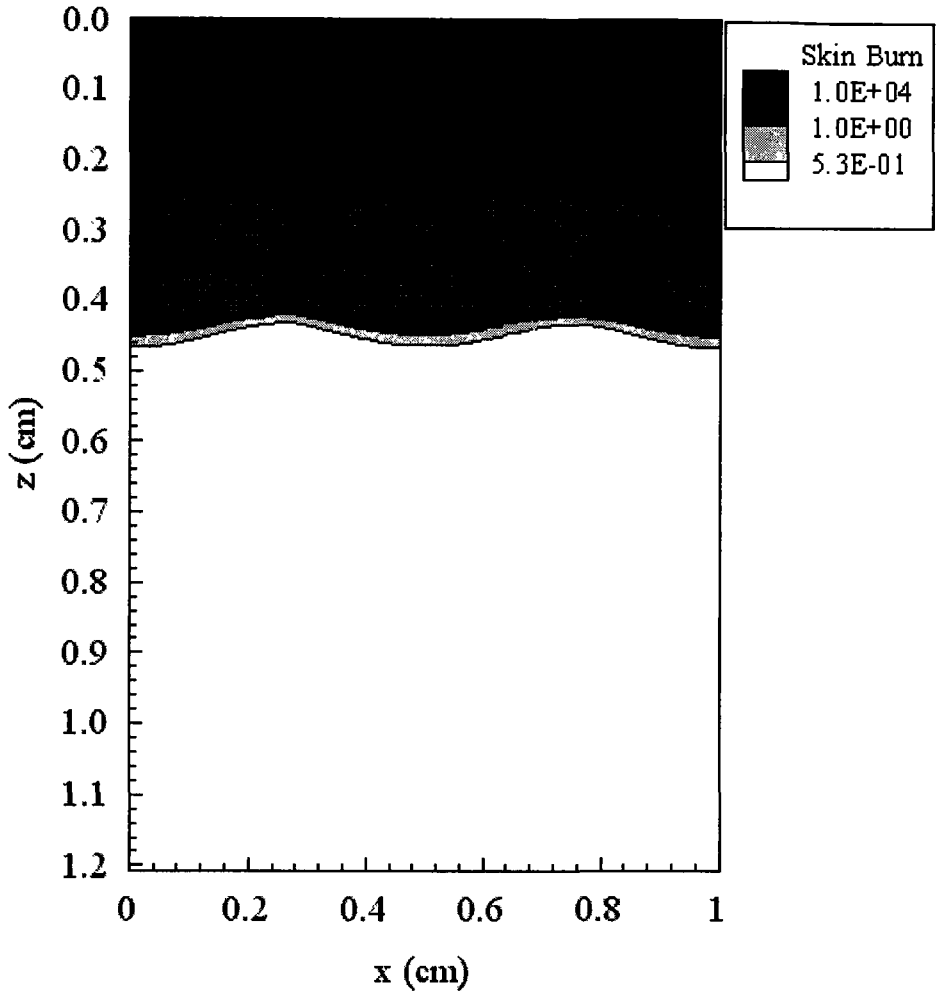
(b)

Figure 2.10.b Contours of the skin burn distributions in the xz-cross-section at y = 0.4 cm, where the artery is located, at various times: (b) t = 200 s [19].



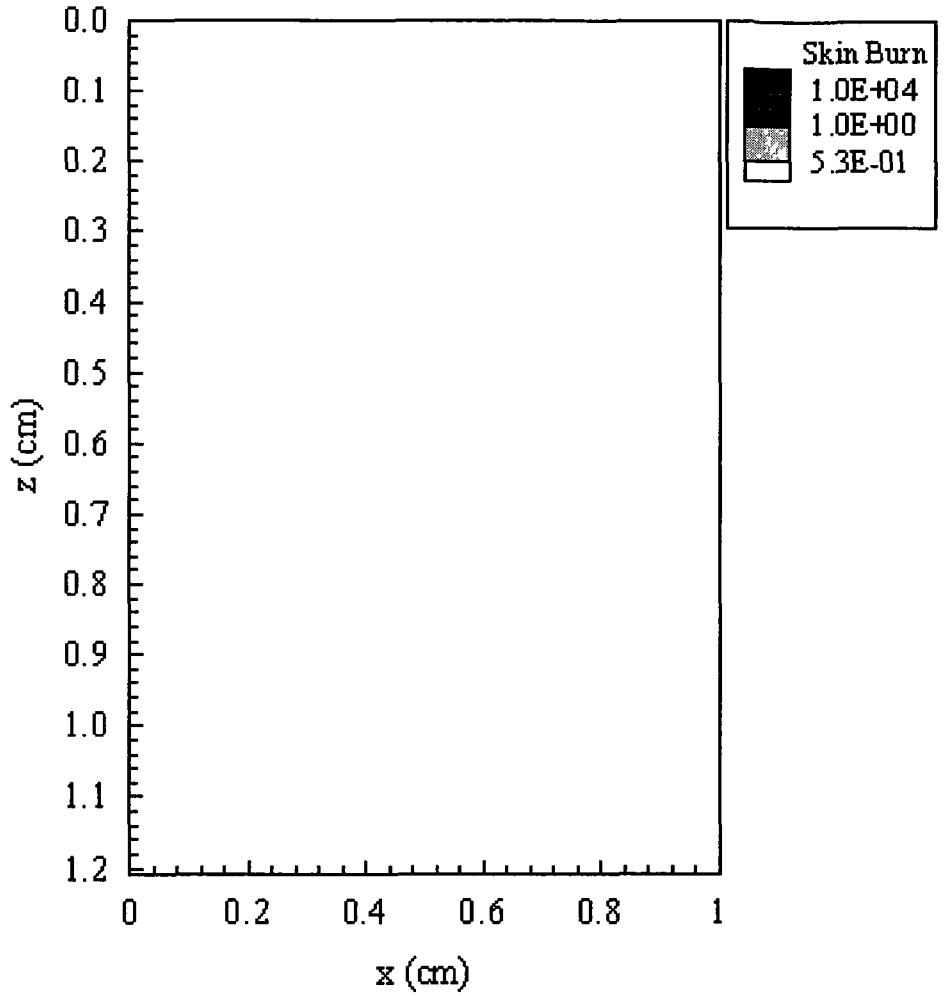
(c)

Figure 2.10.c Contours of the skin burn distributions in the xz-cross-section at y = 0.4 cm, where the artery is located, at various times: (c) t = 300 s [19].



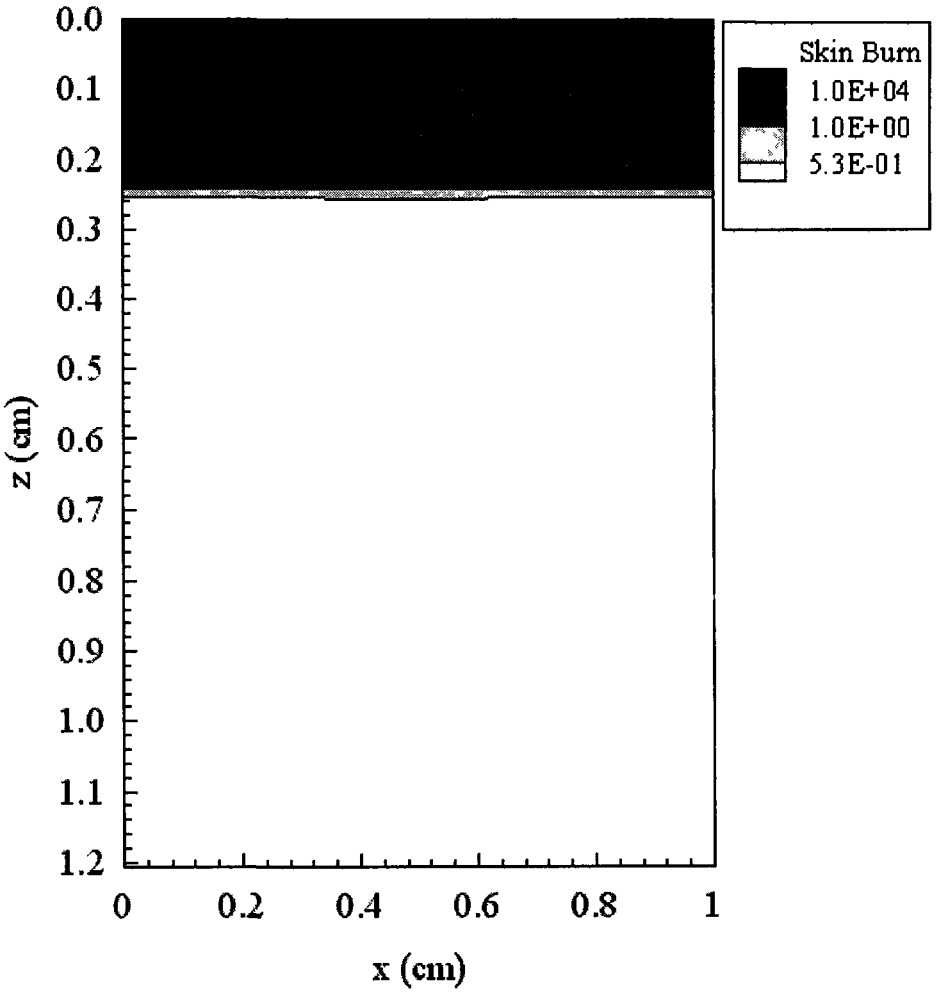
(d)

Figure 2.10.d Contours of the skin burn distributions in the xz-cross-section at y = 0.4 cm, where the artery is located, at various times: (d) t = 400 s [19].



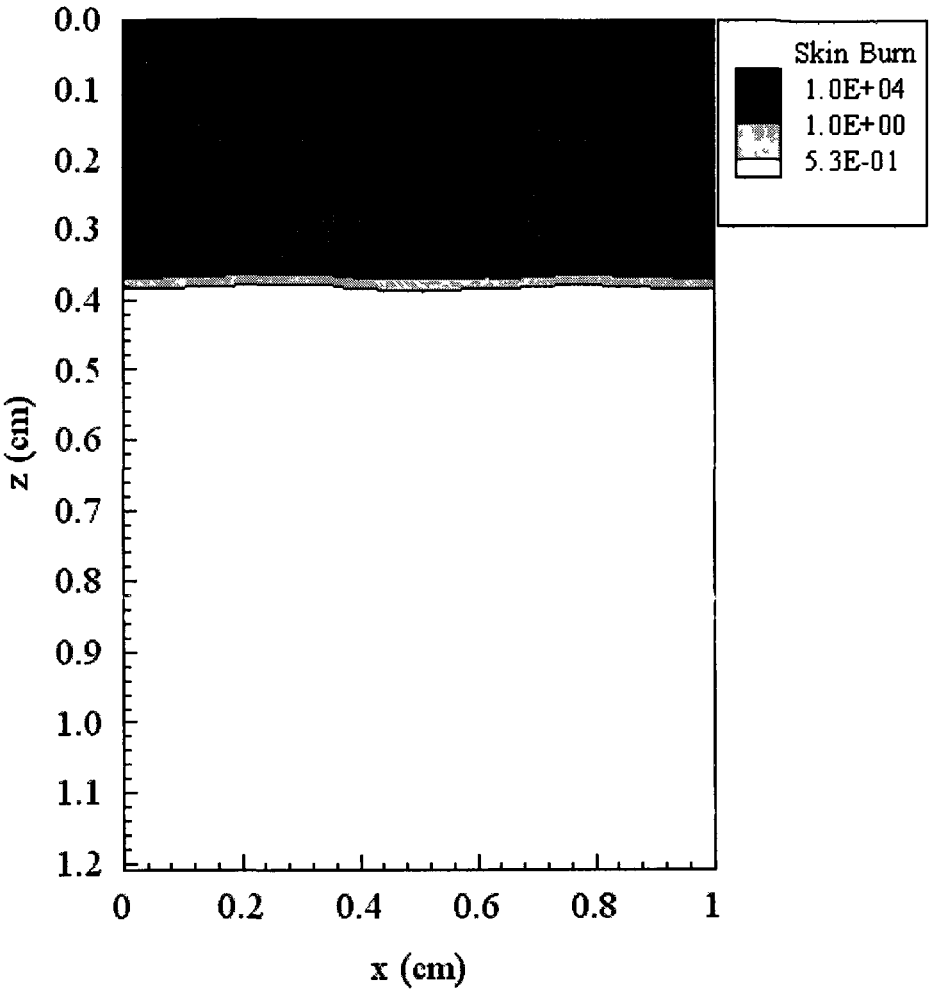
(a)

Figure 2.11.a Contours of the skin burn distributions in the xz-cross-section at y = 0.5 cm at various times: (a) t = 100 s [19].



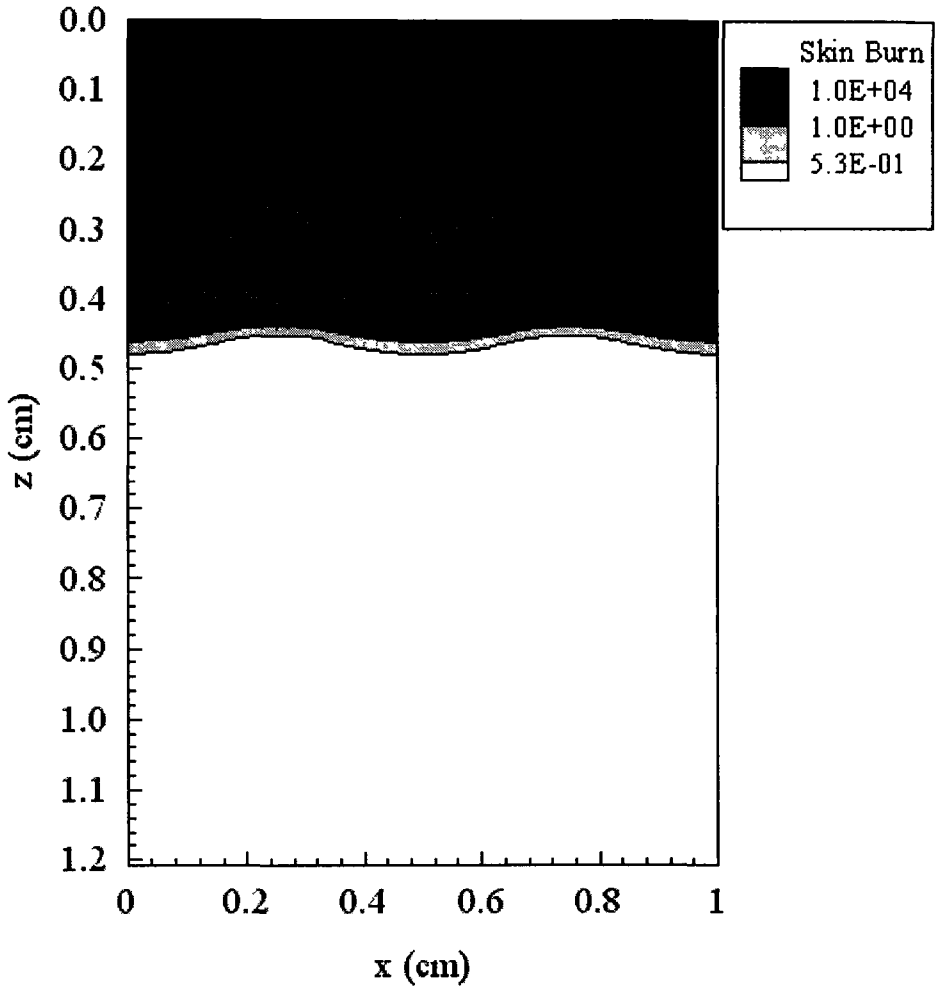
(b)

Figure 2.11.b Contours of the skin burn distributions in the xz-cross-section at y = 0.5 cm at various times: (b) t = 200 s [19].



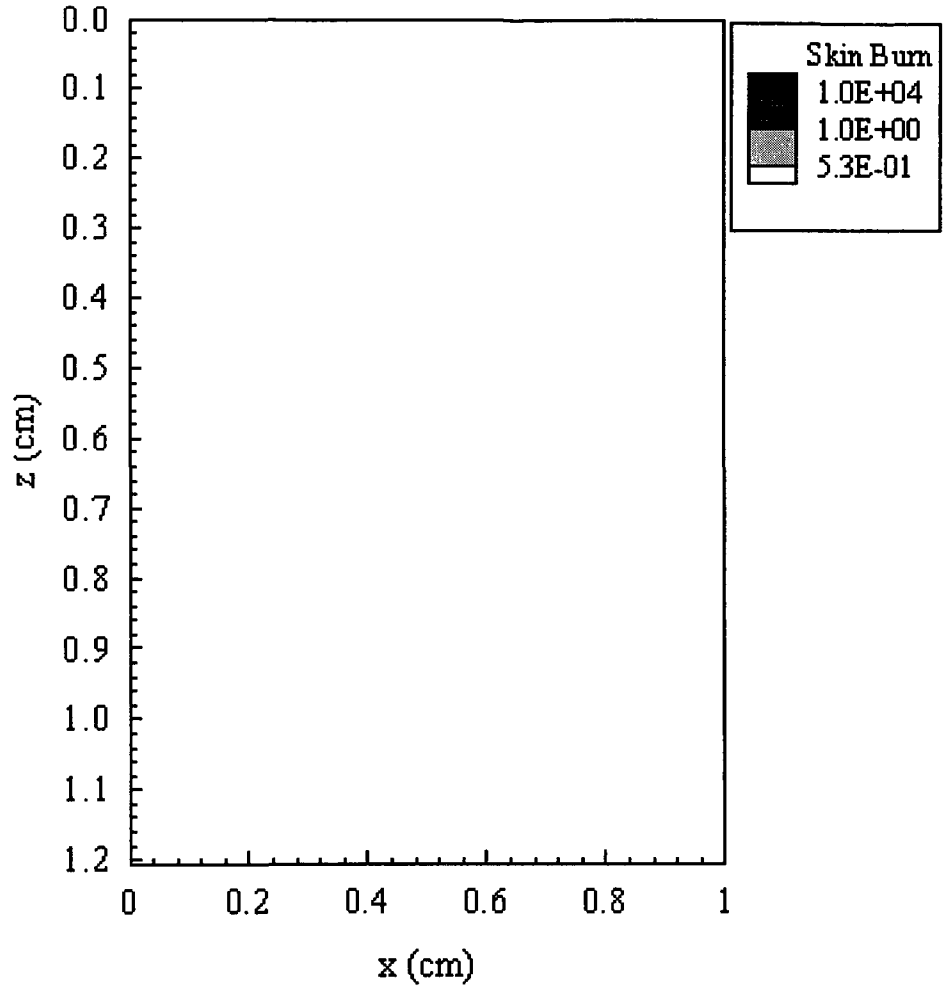
(c)

Figure 2.11.c Contours of the skin burn distributions in the xz-cross-section at y = 0.5 cm at various times: (c) t = 300 s [19].



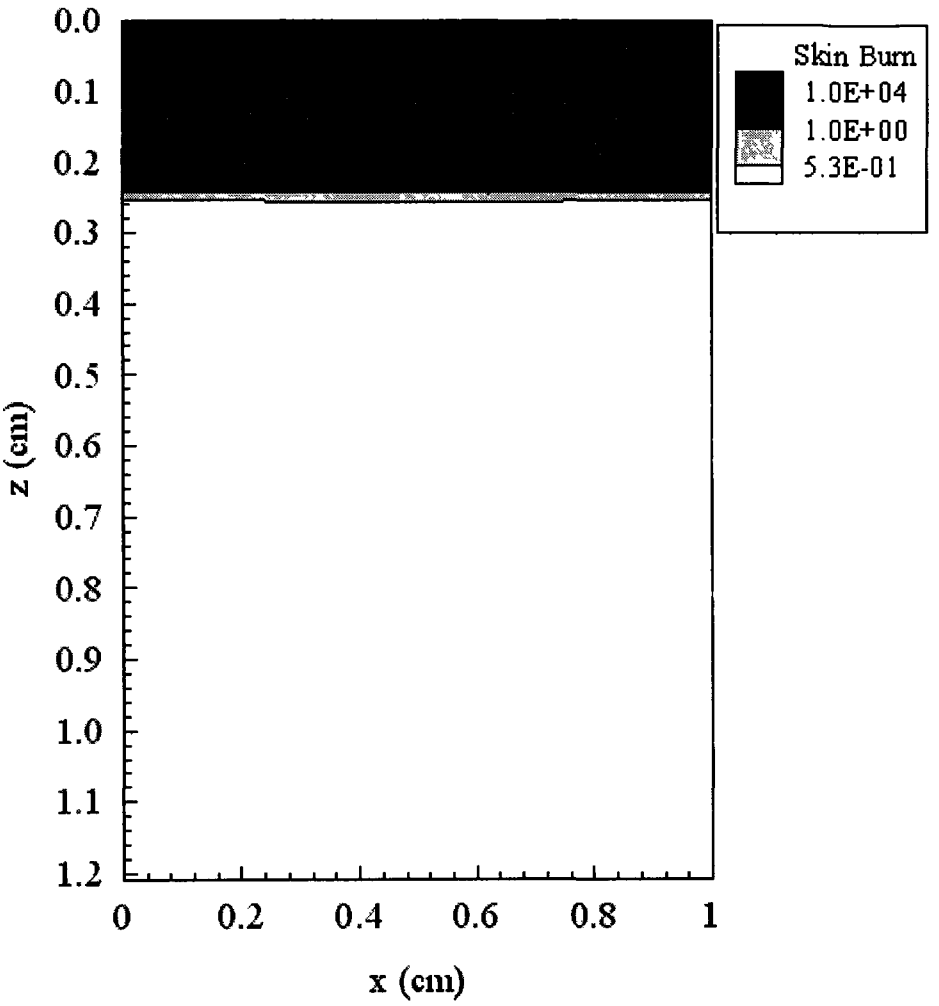
(d)

Figure 2.11.d Contours of the skin burn distributions in the xz-cross-section at y = 0.5 cm at various times: (d) t = 400 s [19].



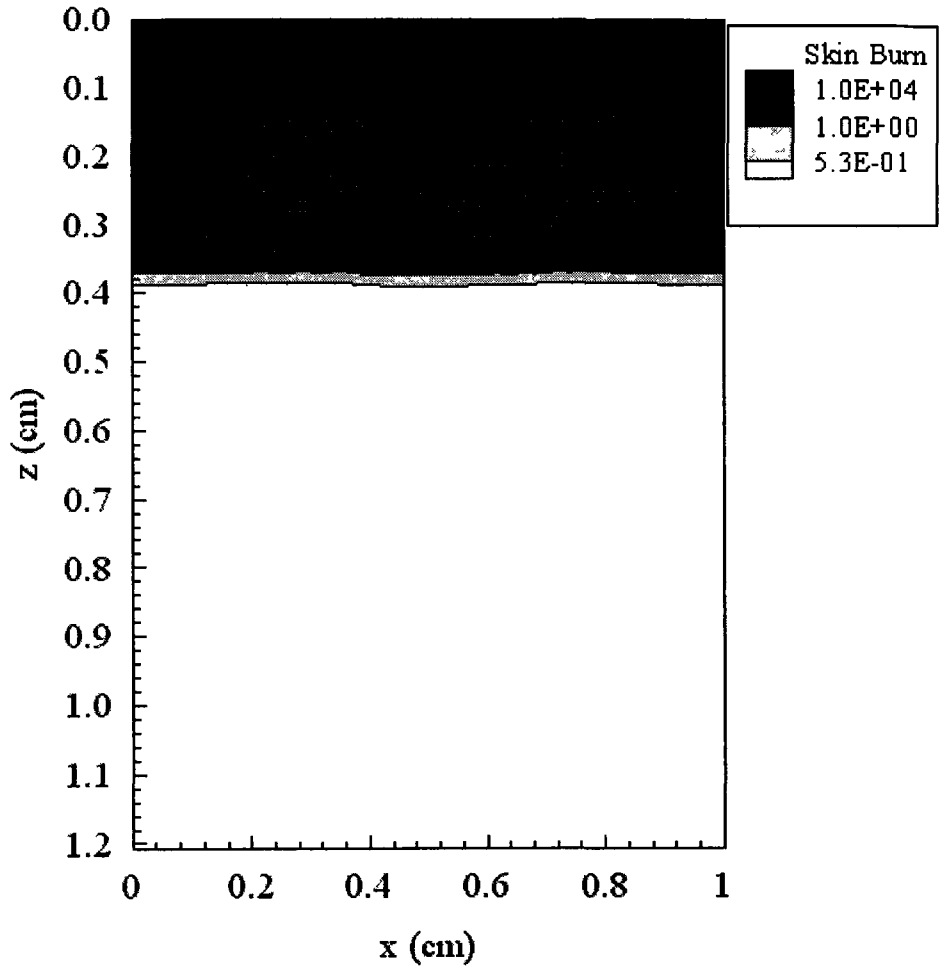
(a)

Figure 2.12.a Contours of the skin burn distributions in the xz-cross-section at y = 0.56 cm, where the vein is located, at various times: (a) t = 100 s [19].



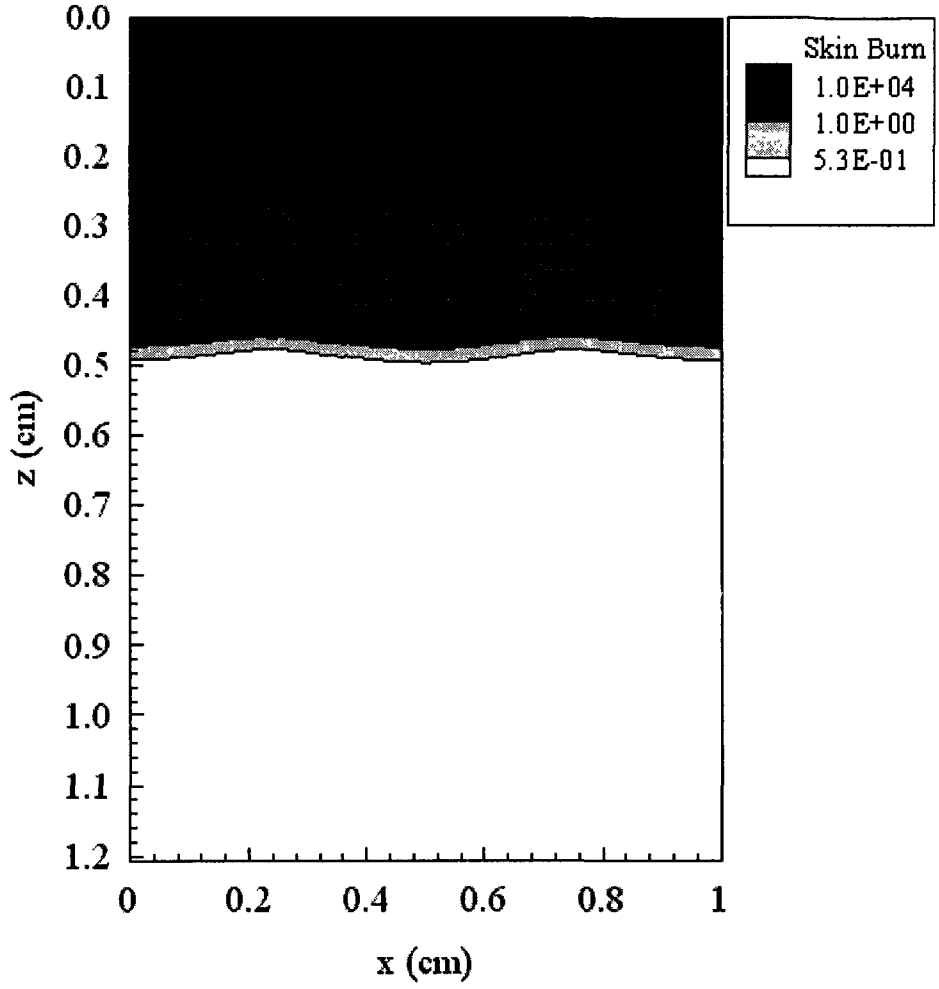
(b)

Figure 2.12.b Contours of the skin burn distributions in the xz-cross-section at y = 0.56 cm, where the vein is located, at various times: (b) t = 200 s [19].



(c)

Figure 2.12.c Contours of the skin burn distributions in the xz-cross-section at y = 0.56 cm, where the vein is located, at various times: (c) t = 300 s [19].



(d)

Figure 2.12.d Contours of the skin burn distributions in the xz-cross-section at y = 0.56 cm, where the vein is located, at various times: (d) t = 400 s [19].

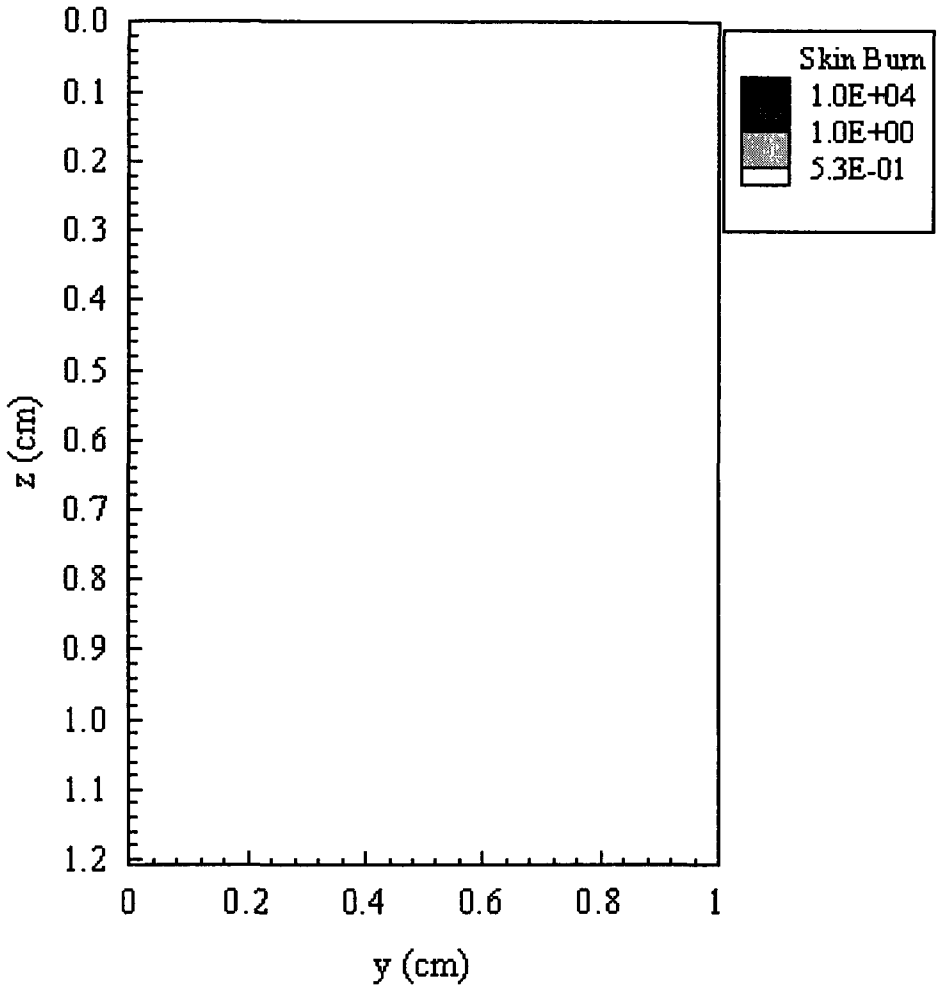
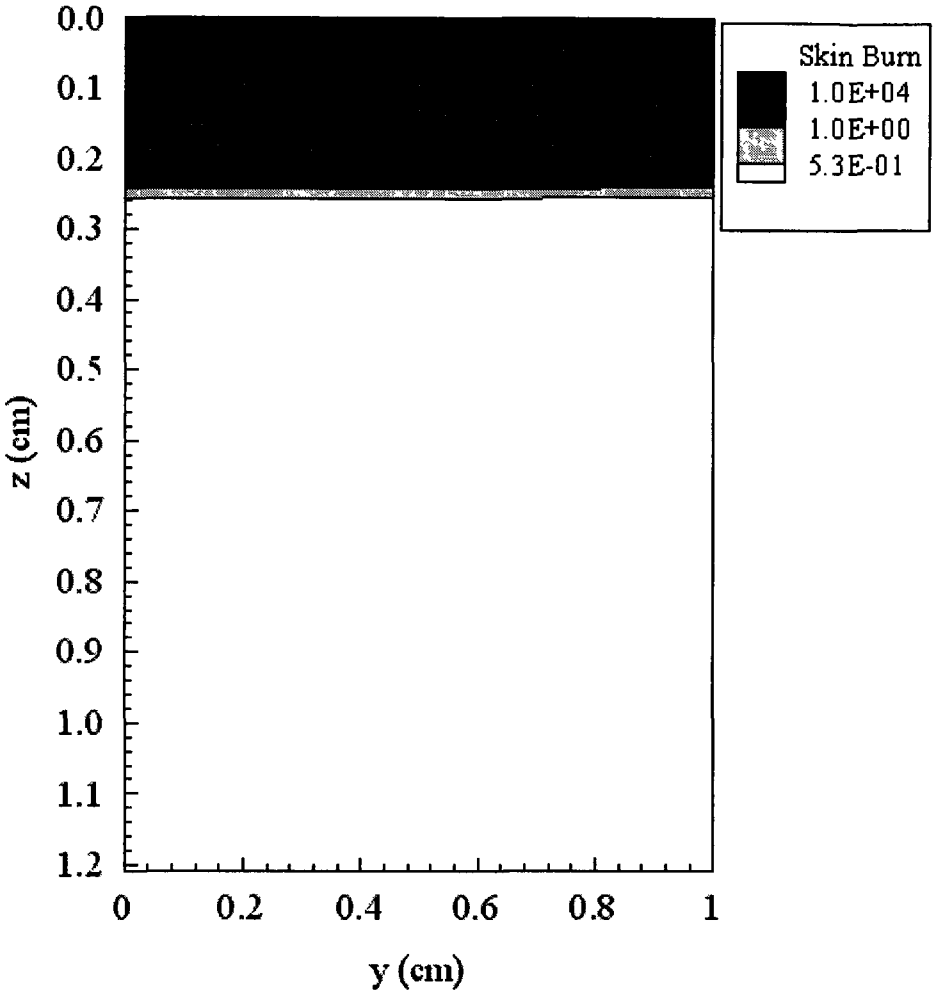
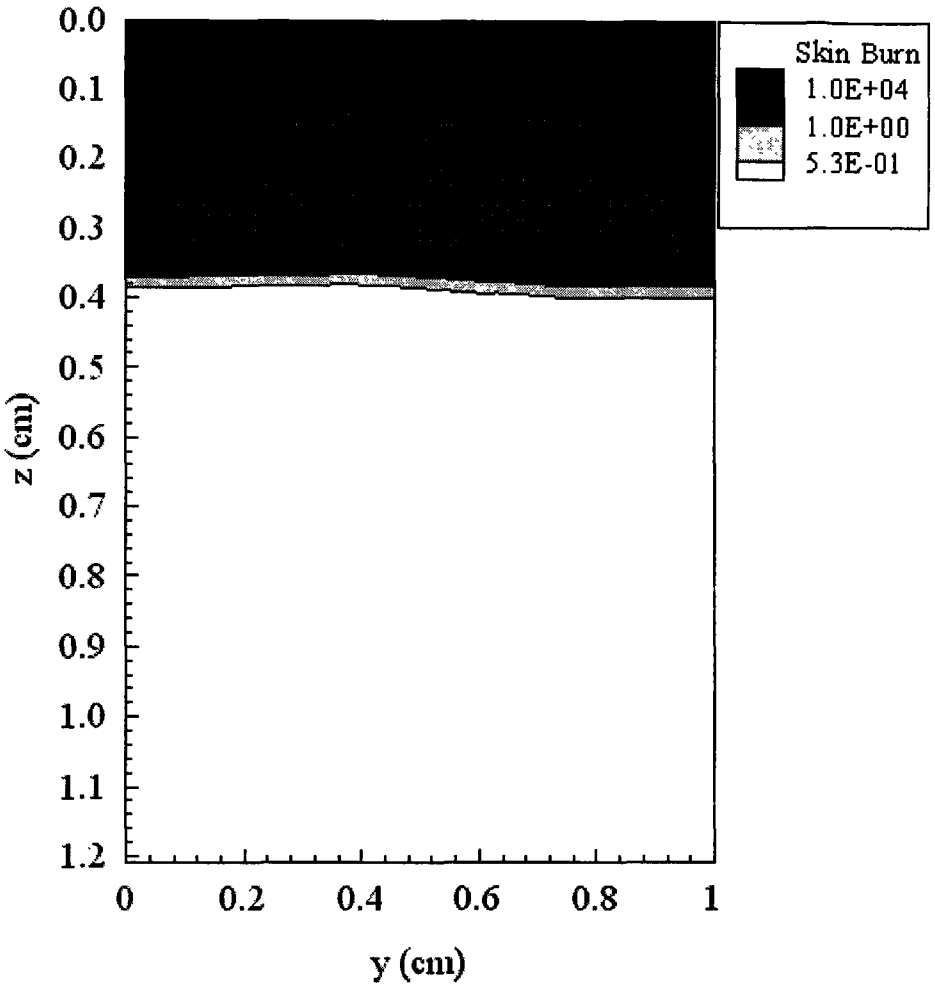


Figure 2.13.a Contours of the skin burn distributions in the yz-cross-section at x = 0.5 cm at various times: (a) t = 100 s [19].



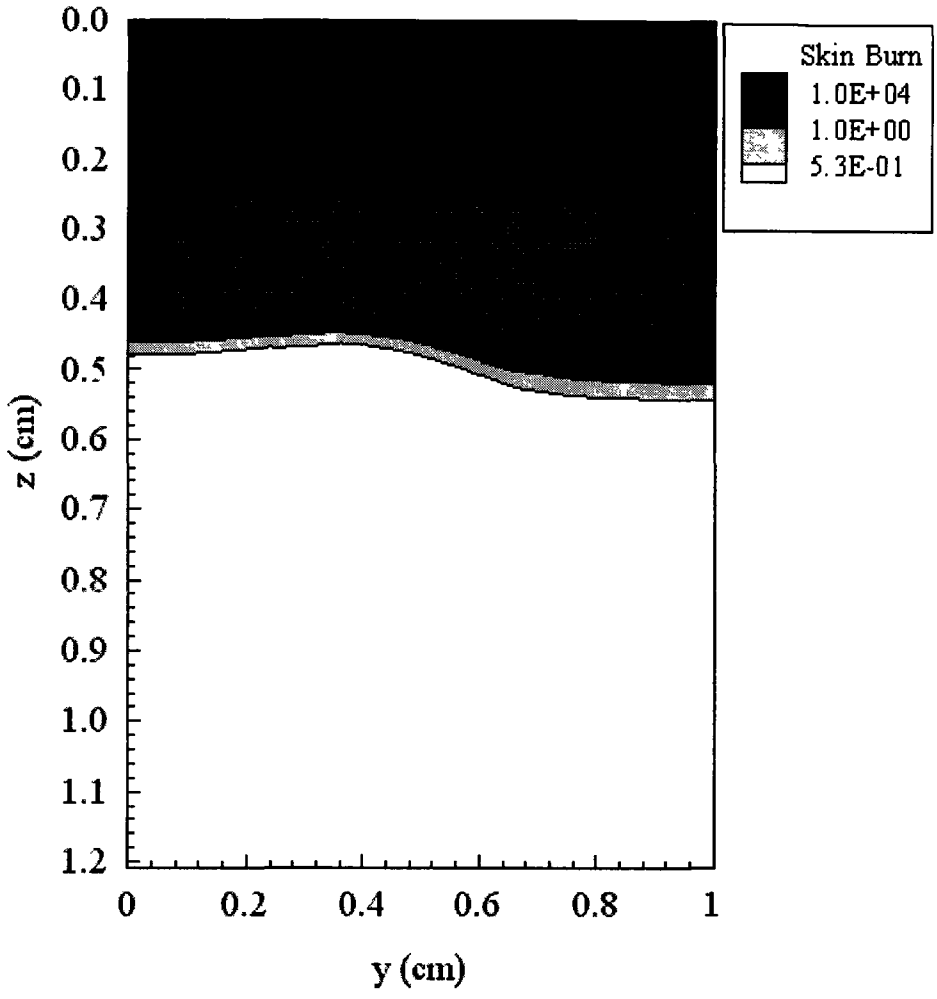
(b)

Figure 2.13.b Contours of the skin burn distributions in the yz-cross-section at x = 0.5 cm at various times: (b) t = 200 s [19].



(c)

Figure 2.13.c Contours of the skin burn distributions in the yz-cross-section at x = 0.5 cm at various times: (c) t = 300 s [19].



(d)

Figure 2.13.d Contours of the skin burn distributions in the yz-cross-section at x = 0.5 cm at various times: (d) t = 400 s [19].

APPENDIX C

SOURCE CODE OF EXAMPLE

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

#define SCREEN_OUT    //flag for screen output

#define SCALEX 1          //GRID POINT CHANGING SCALE
#define SCALEY 1

#define NZ1 8              //layer 1 skin grid points in z direction
#define NZ2 208            //layer 1+2   skin grid points in z direction
#define NZ3 1542           //layer 1+2+3 skin grid points in z direction

//#of grid points in x direction **MUST BE EVEN NUMBER**
#define NX  162 * SCALEX
//#of grid points in y direction **MUST BE EVEN NUMBER**
#define NY  162 * SCALEY

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

#define LX1B 62 * SCALEX
#define LY1B 20 * SCALEY
#define LZ1B 200
#define LX2B 16 * SCALEX
#define LY2B 72 * SCALEY  //whole length
#define LZ2B 160
#define LX3B 12 * SCALEX //each branch
#define LY3B 12 * SCALEY //each branch
#define LZ3B 500          //whole length
#define LX4B 36 * SCALEX   //whole length
#define LY4B 10 * SCALEY   //each branch
#define LZ4B 100           //each branch
#define LX5B 8 * SCALEX    //each branch
#define LY5B 26 * SCALEY   //whole length
#define LZ5B 80            //each branch
```

```
#define LX6B 6 * SCALEX     //each branch
#define LY6B 6 * SCALEY     //each branch
#define LZ6B 180            //whole length
#define LX7B 14 * SCALEX    //whole length
#define LY7B 4 * SCALEY     //each branch
#define LZ7B 50             //each branch

double deltaT = 0.1f;
double deltaX = 0.01f / SCALEX, deltaY = 0.01f / SCALEY;
double deltaZ = 0.001f;          //tissue and blood use same grid size
//double L1, L2, L3;             //depth of three skin layers

//double Lb1[2],Lb2[2],Lb3[2],Lb4[2],Lb5[2],Lb6[2],Lb7[2];
//Length of blood vessel
double P1[2],P2[2],P3[2],P4[2],P5[2],P6[2],P7[2];
//vessel perimeters of level 1 to 7.
double F1[2],F2[2],F3[2],F4[2],F5[2],F6[2],F7[2];
//area of the cross-section of blood vessel level 1 to 7.
double M1[2],M2[2],M3[2],M4[2],M5[2],M6[2],M7[2];
//the mass flow of blood in level 1 to 7.
double CB = 4.134f;              //heat capacity of blood
//double CB = 0.004134;
double Cb1 = 0.0f, Cb2 = 4.2f, Cb3 = 4.2f;
//specific heat of blood in skin later 1 to 3
double v1 = 8.0f;                //v2, v3. velocity of blood flow in level 1 ??
//double v1 = 80;//v2, v3. velocity of blood flow in level 1 ??
double alpha = 0.2f;             //heat transfer coefficient between blood and tissue
//double alpha = 0.002;          //heat transfer coefficient between blood and tissue
double Pdot = 0.5e-3f;           //decreased blood flow rate (for level 7 only)
double p1 = 1.2f, p2 = 1.2f, p3 = 1.0f;                  //density of tissue of layer 1,2,&3
double C1 = 3.6f, C2 = 3.4f, C3 = 3.06f;                 //specific heat of tissue
double k1 = 0.0026f, k2 = 0.0052f, k3 = 0.0021f;         //thermal conductivity of tissue
//double k1 = 0.0026,k2 = 0.0026, k3 = 0.0052;           //thermal conductivity of tissue
double Wb1 = 0.0f, Wb2 = 0.0005f, Wb3 = 0.0005f;         //blood perfusion rate
//double Wb1 = 0.0, Wb2 = 0.000, Wb3 = 0.000;            //blood perfusion rate

double Hf = 0.001;
//convective heat transfer coefficient between the environment and the skin
double Tf = 200.0;               //-17C;  //environment temperature
double tau = 20.0;               // delay of time
//double sigma = 5.669e-12;//Stefan - Boltzmann constant, unit: W/cm^2K^4
double sigma = 5.669e-12;  //Stefan - Boltzmann constant
double epsilon = 0.9;          //Reflectivity

double energy;
double frequency;
double gas = 8.314472;                   //Unit: J/K mol

int centerX = NX/2;
int centerY = NY/2;                      //center of the layer power

const double pai = 3.14159265358979f;
const double omega = 1.0f;
double Bi = alpha/k3;                    //BIOT number
//double Bi = 2;   //BIOT number
Double factor1[2],factor2[2],factor3[2],factor4[2];
```

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

double THETA0 = 0.0f;              //elavated blood temperature at entrance

double (*Tt)[NY+1][NZ3+1];         //tissue temperature at time level n
double (*Tt_n1)[NY+1][NZ3+1];      //tissue temperature at time level n+1 (loop I+1)
double (*Tt_n1_I)[NY+1][NZ3+1];    //tissue temperature at time level n+1 (loop I)
//double (*Tt_n_1)[NY+1][NZ3+1];   //tissue temperature at time level n-1
double (*Ut)[NY+1][NZ3+1];
double (*Ut_n1)[NY+1][NZ3+1];
double (*Damage_t)[NY+1][NZ3+1];
double (*Damage_n1)[NY+1][NZ3+1];

/*double **Tb1;                    //first  level blood boarder temperature
double **Tb2;                      //second level blood boarder temperature
double **Tb3;                      //third  level blood boarder temperature
double **Tb4;                      //fourth  level blood boarder temperature
double **Tb5;                      //fifth  level blood boarder temperature
double **Tb6;                      //sixth level blood boarder temperature
double **Tb7;                      //seventh  level blood boarder temperature*/

double *Tbd1[2][4];                //blood board temperature at time level 1
double *Tbd2[2][4];                //blood board temperature at time level 2
double *Tbd3[2][4];                //blood board temperature at time level 3
double *Tbd4[2][4];                //blood board temperature at time level 4
double *Tbd5[2][4];                //blood board temperature at time level 5
double *Tbd6[2][4];                //blood board temperature at time level 6
double *Tbd7[2][4];                //blood board temperature at time level 7

double **Tb1_n1;                   //first  level blood temperature at time level n+1
double **Tb2_n1;                   //second level blood temperature at time level n+1
double **Tb3_n1;                   //third  level blood temperature at time level n+1
double **Tb4_n1;                   //fourth  level blood temperature at time level n+1
double **Tb5_n1;                   //fifth level blood temperature at time level n+1
double **Tb6_n1;                   //sixth  level blood temperature at time level n+1
double **Tb7_n1;                   //seventh  level blood temperature at time level n+1
double **Tv1_n1;
//interpolated first  level blood vessel temperature at time level n+1
double **Tv2_n1;
//interpolated second level blood vessel temperature at time level n+1
double **Tv3_n1;
//interpolated third  level blood vessel temperature at time level n+1
double **Tv4_n1;
//interpolated fourth  level blood vessel temperature at time level n+1
double **Tv5_n1;
//interpolated fifth level blood vessel temperature at time level n+1
double **Tv6_n1;
//interpolated sixth  level blood vessel temperature at time level n+1
double **Tv7_n1;
//interpolated seventh  level blood vessel temperature at time level n+1

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

const int SPAN = 3;
```

```
//grid points index on boarder of blood vessel
int X1[2],X2a[2],X2b[2],X3a[2],X3b[2],X4a[2],X4b[2],X5a[2],X5b[2];
int X6a1[2],X6b1[2],X6a2[2],X6b2[2],X7a1[2],X7b1[2],X7a2[2],X7b2[2];
int Y1a[2],Y1b[2],Y2a[2],Y2b[2],Y3a[2],Y3b[2],Y4a1[2],Y4b1[2];
int Y4a2[2],Y4b2[2],Y5a1[2],Y5b1[2],Y5a2[2],Y5b2[2];
int Y6a1[2],Y6b1[2],Y6a2[2],Y6b2[2],Y7a1[2],Y7b1[2],Y7a2[2],Y7b2[2];
int Y7a3[2],Y7b3[2],Y7a4[2],Y7b4[2];
int Z1a[2],Z1b[2],Z2a[2],Z2b[2],Z3a[2],Z3b[2],Z4a[2],Z4b[2];
int Z5a1[2],Z5b1[2],Z5a2[2],Z5b2[2],Z6a1[2],Z6b1[2],Z6a2[2],Z6b2[2];
int Z7a1[2],Z7b1[2],Z7a2[2],Z7b2[2];

int i, j, x, y, z;      //x, y, z direction index
int x1, y7, z1, x2, y2, z2, x3, y3, z3, x4, y4, x5, y5, z4, z5, z6;
//y1 already defined in Math.h, temperary variables of x, y, z direction in all blood levels
int t, I, loopP; //used to keep record of loop and output result
int k, p, q, r, h; //repeat variable used for blood vessels of same layer

//blood boarder avaibles
int x1a,x2a,x2b,x3a,x3b,x4a,x4b,x5a,x5b,x6a1,x6b1,x6a2,x6b2,x7a1,x7b1,x7a2,x7b2;
int y1a,y1b,y2a,y2b,y3a,y3b,y4a1,y4b1,y4a2,y4b2,y5a1,y5b1,y5a2,y5b2,y6a1,y6b1;
int y6a2,y6b2,y7a1,y7b1,y7a2,y7b2,y7a3,y7b3,y7a4,y7b4;
int z1a,z1b,z2a,z2b,z3a,z3b,z4a,z4b,z5a1,z5b1,z5a2,z5b2,z6a1,z6b1;
int z6a2,z6b2,z7a1,z7b1,z7a2,z7b2;
int lx1,ly1,lz1,lx2,ly2,lz2,lx3,ly3,lz3,ly4,lz4,lx4,lx5,ly5,lz5;
int lx6,ly6,lz6,lx7,ly7,lz7;
int cenY,cenZ,cenX;

//bool bPowerOn; //flag indicating whether the power is on or off

void initialize();
int   CalcAll();
int   getTv_blood(int index);
int   CalcTb();
int   CalcTb2();
double CalcTt();
void   CalcVessel(int index);
void   Reset(void);
void   setVesselBorder();
void   AdjustMtx(int index);
void   clearMem();
void   setboarderVariable(int index);//added by me
void   reloadbloodeteperature(int index);//added by me

void writeSquareXZ(int x0, int z0, int x1, int z1, int y, int t, int I);
void writeSquareYZ(int y0, int z0, int y1, int z1, int x, int t, int I);
void writeSquareXY(int x0, int y0, int x1, int y1, int z, int t, int I);
void writeZCenter(int t);
void writeAll(int t, int I);
void writeLinearSys(int x, int y, int i, int I);
void writebloode(int i, double x);//test by zeng
void writeLog(char *line);
void writeSquareXZ_Damage(int x0, int z0, int x1, int z1, int y, int t, int I);
void writeSquareYZ_Damage(int y0, int z0, int y1, int z1, int x, int t, int I);
char tmp[256];

int   TOTAL_T   = 200;
```

```
//double TOTAL_T = 0.1f; //test value by zeng
double   Err_I = 0.001f;     //for I loop
double   Err_P = 0.01f;      //for P loop
double temp;
double maxErr1, f;
double aa, bb;
int inde;
double fk1, fk2, fk3, fk4;
int n;

int  main(void)
{
        printf("screen out\n");
        printf("initializing...\n");
        initialize(); //memory allocation & variable initalization
        printf("running_1\n");
        loopP = 0;//un-used variable by zeng
        //calculate tissue and blood temperature based on specified power level
        CalcAll();
        printf("CalcAll Finished!\n");
        clearMem();
        return 1;
}

//memory allocation & variable initalization
void initialize()
{
        Tt = new double [NX+1][NY+1][NZ3+1];
        Tt_n1    = new double [NX+1][NY+1][NZ3+1];
        Tt_n1_I = new double [NX+1][NY+1][NZ3+1];
        //Tt_n_1  = new double [NX+1][NY+1][NZ3+1];
        Ut                  = new double [NX+1][NY+1][NZ3+1];
        Ut_n1    = new double [NX+1][NY+1][NZ3+1];
        Damage_t = new double [NX+1][NY+1][NZ3+1];
        Damage_n1 = new double [NX+1][NY+1][NZ3+1];

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

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

        /*Lb1[0] = LX1A*deltaX;  //areries blood length in one to seven layer
        Lb2[0] = LY2A*deltaY;
        Lb3[0] = LZ3A*deltaZ;
        Lb4[0] = LX4A*deltaX;
        Lb5[0] = LY5A*deltaY;
        Lb6[0] = LZ6A*deltaZ;
        Lb7[0] = LX7A*deltaX;
```

```
Lb1[1] = LX1B*deltaX;  //vene blood length in one to seven layer
Lb2[1] = LY2B*deltaY;
Lb3[1] = LZ3B*deltaZ;
Lb4[1] = LX4B*deltaX;
Lb5[1] = LY5B*deltaY;
Lb6[1] = LZ6B*deltaZ;
Lb7[1] = LX7B*deltaX;

L1 = NZ1*deltaZ;      //first layer of skin length
L2 = (NZ2-NZ1)*deltaZ; //second layer of skin length
L3 = (NZ3-NZ2)*deltaZ; //seven layer of skin length */

setVesselBorder();
Tb1_n1 = new double*[2];
Tb2_n1 = new double*[2];
Tb3_n1 = new double*[2];
Tb4_n1 = new double*[2];
Tb5_n1 = new double*[2];
Tb6_n1 = new double*[2];
Tb7_n1 = new double*[2];
Tv1_n1 = new double*[2];
Tv2_n1 = new double*[2];
Tv3_n1 = new double*[2];
Tv4_n1 = new double*[2];
Tv5_n1 = new double*[2];
Tv6_n1 = new double*[2];
Tv7_n1 = new double*[2];

Tbd1[0][0] = new double[LX1A+1];
Tbd2[0][0] = new double[LY2A+1];
Tbd3[0][0] = new double[(LZ3A+1)*2];
Tbd4[0][0] = new double[(LX4A+1)*4];
Tbd5[0][0] = new double[(LY5A+1)*8];
Tbd6[0][0] = new double[(LZ6A+1)*16];
Tbd7[0][0] = new double[(LX7A+1)*32];

Tbd1[0][1] = new double[LX1A+1];
Tbd2[0][1] = new double[LY2A+1];
Tbd3[0][1] = new double[(LZ3A+1)*2];
Tbd4[0][1] = new double[(LX4A+1)*4];
Tbd5[0][1] = new double[(LY5A+1)*8];
Tbd6[0][1] = new double[(LZ6A+1)*16];
Tbd7[0][1] = new double[(LX7A+1)*32];

Tbd1[0][2] = new double[LX1A+1];
Tbd2[0][2] = new double[LY2A+1];
Tbd3[0][2] = new double[(LZ3A+1)*2];
Tbd4[0][2] = new double[(LX4A+1)*4];
Tbd5[0][2] = new double[(LY5A+1)*8];
Tbd6[0][2] = new double[(LZ6A+1)*16];
Tbd7[0][2] = new double[(LX7A+1)*32];

Tbd1[0][3] = new double[LX1A+1];
Tbd2[0][3] = new double[LY2A+1];
Tbd3[0][3] = new double[(LZ3A+1)*2];
Tbd4[0][3] = new double[(LX4A+1)*4];
```

```
Tbd5[0][3] = new double[(LY5A+1)*8];
Tbd6[0][3] = new double[(LZ6A+1)*16];
Tbd7[0][3] = new double[(LX7A+1)*32];

Tb1_n1[0] = new double[LX1A+1];
Tb2_n1[0] = new double[LY2A+1];
Tb3_n1[0] = new double[(LZ3A+1)*2];
Tb4_n1[0] = new double[(LX4A+1)*4];
Tb5_n1[0] = new double[(LY5A+1)*8];
Tb6_n1[0] = new double[(LZ6A+1)*16];
Tb7_n1[0] = new double[(LX7A+1)*32];

Tv1_n1[0] = new double[LX1A+1];
Tv2_n1[0] = new double[LY2A+1];
Tv3_n1[0] = new double[(LZ3A+1)*2];
Tv4_n1[0] = new double[(LX4A+1)*4];
Tv5_n1[0] = new double[(LY5A+1)*8];
Tv6_n1[0] = new double[(LZ6A+1)*16];
Tv7_n1[0] = new double[(LX7A+1)*32];

Tbd1[1][0] = new double[LX1B+1];
Tbd2[1][0] = new double[LY2B+1];
Tbd3[1][0] = new double[(LZ3B+1)*2];
Tbd4[1][0] = new double[(LX4B+1)*4];
Tbd5[1][0] = new double[(LY5B+1)*8];
Tbd6[1][0] = new double[(LZ6B+1)*16];
Tbd7[1][0] = new double[(LX7B+1)*32];

Tbd1[1][1] = new double[LX1B+1];
Tbd2[1][1] = new double[LY2B+1];
Tbd3[1][1] = new double[(LZ3B+1)*2];
Tbd4[1][1] = new double[(LX4B+1)*4];
Tbd5[1][1] = new double[(LY5B+1)*8];
Tbd6[1][1] = new double[(LZ6B+1)*16];
Tbd7[1][1] = new double[(LX7B+1)*32];

Tbd1[1][2] = new double[LX1B+1];
Tbd2[1][2] = new double[LY2B+1];
Tbd3[1][2] = new double[(LZ3B+1)*2];
Tbd4[1][2] = new double[(LX4B+1)*4];
Tbd5[1][2] = new double[(LY5B+1)*8];
Tbd6[1][2] = new double[(LZ6B+1)*16];
Tbd7[1][2] = new double[(LX7B+1)*32];

Tbd1[1][3] = new double[LX1B+1];
Tbd2[1][3] = new double[LY2B+1];
Tbd3[1][3] = new double[(LZ3B+1)*2];
Tbd4[1][3] = new double[(LX4B+1)*4];
Tbd5[1][3] = new double[(LY5B+1)*8];
Tbd6[1][3] = new double[(LZ6B+1)*16];
Tbd7[1][3] = new double[(LX7B+1)*32];

Tb1_n1[1] = new double[LX1B+1];
Tb2_n1[1] = new double[LY2B+1];
Tb3_n1[1] = new double[(LZ3B+1)*2];
Tb4_n1[1] = new double[(LX4B+1)*4];
```

```
Tb5_n1[1] = new double[(LY5B+1)*8];
Tb6_n1[1] = new double[(LZ6B+1)*16];
Tb7_n1[1] = new double[(LX7B+1)*32];

Tv1_n1[1] = new double[LX1B+1];
Tv2_n1[1] = new double[LY2B+1];
Tv3_n1[1] = new double[(LZ3B+1)*2];
Tv4_n1[1] = new double[(LX4B+1)*4];
Tv5_n1[1] = new double[(LY5B+1)*8];
Tv6_n1[1] = new double[(LZ6B+1)*16];
Tv7_n1[1] = new double[(LX7B+1)*32];

P1[0] = (LZ1A*deltaZ + LY1A*deltaY)*2;
P2[0] = (LX2A*deltaX + LZ2A*deltaZ)*2;
P3[0] = (LX3A*deltaX + LY3A*deltaY)*2;
P4[0] = (LZ4A*deltaZ + LY4A*deltaY)*2;
P5[0] = (LX5A*deltaX + LZ5A*deltaZ)*2;
P6[0] = (LX6A*deltaX + LY6A*deltaY)*2;
P7[0] = (LZ7A*deltaZ + LY7A*deltaY)*2;

P1[1] = (LZ1B*deltaZ + LY1B*deltaY)*2;
P2[1] = (LX2B*deltaX + LZ2B*deltaZ)*2;
P3[1] = (LX3B*deltaX + LY3B*deltaY)*2;
P4[1] = (LZ4B*deltaZ + LY4B*deltaY)*2;
P5[1] = (LX5B*deltaX + LZ5B*deltaZ)*2;
P6[1] = (LX6B*deltaX + LY6B*deltaY)*2;
P7[1] = (LZ7B*deltaZ + LY7B*deltaY)*2;

F1[0] = (LZ1A*deltaZ)*(LY1A*deltaY);
F2[0] = (LX2A*deltaX)*(LZ2A*deltaZ);
F3[0] = (LX3A*deltaX)*(LY3A*deltaY);
F4[0] = (LZ4A*deltaZ)*(LY4A*deltaY);
F5[0] = (LX5A*deltaX)*(LZ5A*deltaZ);
F6[0] = (LX6A*deltaX)*(LY6A*deltaY);
F7[0] = (LZ7A*deltaZ)*(LY7A*deltaY);

F1[1] = (LZ1B*deltaZ)*(LY1B*deltaY);
F2[1] = (LX2B*deltaX)*(LZ2B*deltaZ);
F3[1] = (LX3B*deltaX)*(LY3B*deltaY);
F4[1] = (LZ4B*deltaZ)*(LY4B*deltaY);
F5[1] = (LX5B*deltaX)*(LZ5B*deltaZ);
F6[1] = (LX6B*deltaX)*(LY6B*deltaY);
F7[1] = (LZ7B*deltaZ)*(LY7B*deltaY);

M1[0] = v1 * F1[0];
M2[0] = 0.5f * M1[0];
M3[0] = 0.5f * M2[0];
M4[0] = 0.5f * M3[0];
M5[0] = 0.5f * M4[0];
M6[0] = 0.5f * M5[0];
M7[0] = 0.5f * M6[0];

M1[1] = v1 * F1[1];
M2[1] = 0.5f * M1[1];
M3[1] = 0.5f * M2[1];
M4[1] = 0.5f * M3[1];
```

```
M5[1] = 0.5f * M4[1];
M6[1] = 0.5f * M5[1];
M7[1] = 0.5f * M6[1];

//factor[] is used for Runge-Kutta method

factor1[0] = deltaX*alpha*P1[0]/(M1[0]*CB);
factor2[0] = deltaY*alpha*P2[0]/(M2[0]*CB);
factor3[0] = deltaZ*alpha*P3[0]/(M3[0]*CB);
factor4[0] = deltaX*alpha*P4[0]/(M4[0]*CB);
factor5[0] = deltaY*alpha*P5[0]/(M5[0]*CB);
factor6[0] = deltaZ*alpha*P6[0]/(M6[0]*CB);
factor7[0] = deltaX*alpha*P7[0]/(M7[0]*CB);

factor1[1] = deltaX*alpha*P1[1]/(M1[1]*CB);
factor2[1] = deltaY*alpha*P2[1]/(M2[1]*CB);
factor3[1] = deltaZ*alpha*P3[1]/(M3[1]*CB);
factor4[1] = deltaX*alpha*P4[1]/(M4[1]*CB);
factor5[1] = deltaY*alpha*P5[1]/(M5[1]*CB);
factor6[1] = deltaZ*alpha*P6[1]/(M6[1]*CB);
factor7[1] = deltaX*alpha*P7[1]/(M7[1]*CB);

writeLog("Initialization...");

//initialize tri-diangonal system, left side (fixed)
for(i=1;i<=NX-1;i++)
{
        for(j=1;j<=NY-1;j++)
        {
                for(z=1;z<=NZ1-1;z++)
                {
                        //Laser_case: Cranknicolson-scheme
                        /*b0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);
                        a0[i][j][z] = 2*p1*C1 + Wb1*Cb1*deltaT +
                              (4.0*k1*deltaT)*(1.0f/(deltaX*deltaX)
                              +1.0f/(deltaY*deltaY))
                               + (2*k1*deltaT)/(deltaZ*deltaZ);
                        c0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);*/

                        //Laser_case: implicit-scheme
                        /*b0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);
                        a0[i][j][z] = p1*C1 + Wb1*Cb1*deltaT +
                                      (4.0*k1*deltaT)*(1.0f/(deltaX*deltaX)
                                       +1.0f/(deltaY*deltaY))
                                       + (2.0*k1*deltaT)/(deltaZ*deltaZ);
                        c0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);*/

                        //Radiation_case: Dr dai's scheme
                        b0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);
                        a0[i][j][z] = 2.0*p1*C1*(1.0+tau*Wb1*Cb1/(p1*C1)
                                       +2.0*tau/deltaT) + Wb1*Cb1*deltaT
                                       + (4.0*k1*deltaT)*(1.0/(deltaX*deltaX)
                                      +1.0/(deltaY*deltaY))
                                      + (2.0*k1*deltaT)/(deltaZ*deltaZ);
                        c0[i][j][z] = -(k1*deltaT)/(deltaZ*deltaZ);
                }
```

```
b0[i][j][NZ1]=-k1;
a0[i][j][NZ1]=k1+k2;
c0[i][j][NZ1]=-k2;

for(z=NZ1+1;z<=NZ2-1;z++)
{
        //Laser_case: Cranknicolson-scheme
        /*b0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);
        a0[i][j][z] = 2*p2*C2 + Wb2*Cb2*deltaT +
                      (4*k2*deltaT)*(1.0f/(deltaX*deltaX)
                      +1.0f/(deltaY*deltaY))
                      + (2*k2*deltaT)/(deltaZ*deltaZ);
        c0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);*/

        //Laser_case: implicit-scheme
        /*b0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);
        a0[i][j][z] = p2*C2 + Wb2*Cb2*deltaT +
                      (4.0*k2*deltaT)*(1.0f/(deltaX*deltaX)
                      +1.0f/(deltaY*deltaY))
                      + (2.0*k2*deltaT)/(deltaZ*deltaZ);
        c0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);*/

        //Radiation_case: Dr Dai's scheme
        b0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);
        a0[i][j][z] = 2.0*p2*C2*(1.0+tau*Wb2*Cb2/(p2*C2)
                      +2.0*tau/deltaT) + Wb2*Cb2*deltaT
                      + (4.0*k2*deltaT)*(1.0/(deltaX*deltaX)
                      +1.0/(deltaY*deltaY))
                      + (2.0*k2*deltaT)/(deltaZ*deltaZ);
        c0[i][j][z] = -(k2*deltaT)/(deltaZ*deltaZ);
}

b0[i][j][NZ2]=-k2;
a0[i][j][NZ2]=k2+k3;
c0[i][j][NZ2]=-k3;

// third skin layer
for(z=NZ2+1;z<=NZ3-1;z++)
{
        //Laser_case: Cranknicolson-scheme
        /*b0[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);
        a0[i][j][z] = 2*p3*C3 + + Wb3*Cb3*deltaT +
                      (4*k3*deltaT)*(1.0f/(deltaX*deltaX)
                      +1.0f/(deltaY*deltaY))
                      + (2*k3*deltaT)/(deltaZ*deltaZ);
        c0[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);*/

        //Laser_case: implicit-scheme
        /*b0[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);
        a0[i][j][z] = p3*C3 + Wb3*Cb3*deltaT +
                    (4.0*k3*deltaT)*(1.0f/(deltaX*deltaX)
                    +1.0f/(deltaY*deltaY))
                    + (2.0*k3*deltaT)/(deltaZ*deltaZ);
        c0[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);*/

        //Radiation_case: Dr Dai's scheme
```

```
                                        b0[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);
                                        a0[i][j][z] = 2.0*p3*C3*(1.0+tau*Wb3*Cb3/(p3*C3)
                                                +2.0*tau/deltaT) + Wb3*Cb3*deltaT
                                                + (4.0*k3*deltaT)*(1.0/(deltaX*deltaX)
                                                +1.0/(deltaY*deltaY))
                                                + (2.0*k3*deltaT)/(deltaZ*deltaZ);
                                        c0[i][j][z] = -(k3*deltaT)/(deltaZ*deltaZ);
                                }
                        }//j
                }//i

                return;
}

void clearMem()
{
        if(Tt) delete [] Tt;
        if(Tt_n1) delete [] Tt_n1;
        if(Tt_n1_I) delete [] Tt_n1_I;
        //if(Tt_n_1) delete [] Tt_n_1;
        if(Ut) delete [] Ut;
        if(Ut_n1) delete [] Ut_n1;

        if(a) delete [] a;
        if(b) delete [] b;
        if(c) delete [] c;
        if(d) delete [] d;

        if(Tb1_n1) {delete [] Tb1_n1[0]; delete [] Tb1_n1[1]; delete Tb1_n1;}
        if(Tb2_n1) {delete [] Tb2_n1[0]; delete [] Tb2_n1[1]; delete Tb2_n1;}
        if(Tb3_n1) {delete [] Tb3_n1[0]; delete [] Tb3_n1[1]; delete Tb3_n1;}
        if(Tb4_n1) {delete [] Tb4_n1[0]; delete [] Tb4_n1[1]; delete Tb4_n1;}
        if(Tb5_n1) {delete [] Tb5_n1[0]; delete [] Tb5_n1[1]; delete Tb5_n1;}
        if(Tb6_n1) {delete [] Tb6_n1[0]; delete [] Tb6_n1[1]; delete Tb6_n1;}
        if(Tb7_n1) {delete [] Tb7_n1[0]; delete [] Tb7_n1[1]; delete Tb7_n1;}

        if(Tv1_n1) {delete [] Tv1_n1[0]; delete [] Tv1_n1[1]; delete Tv1_n1;}
        if(Tv2_n1) {delete [] Tv2_n1[0]; delete [] Tv2_n1[1]; delete Tv2_n1;}
        if(Tv3_n1) {delete [] Tv3_n1[0]; delete [] Tv3_n1[1]; delete Tv3_n1;}
        if(Tv4_n1) {delete [] Tv4_n1[0]; delete [] Tv4_n1[1]; delete Tv4_n1;}
        if(Tv5_n1) {delete [] Tv5_n1[0]; delete [] Tv5_n1[1]; delete Tv5_n1;}
        if(Tv6_n1) {delete [] Tv6_n1[0]; delete [] Tv6_n1[1]; delete Tv6_n1;}
        if(Tv7_n1) {delete [] Tv7_n1[0]; delete [] Tv7_n1[1]; delete Tv7_n1;}

        writeLog("Memory released.");
        return;
}

void setVesselBorder()
{
        X1[0] = NX;
        X2b[0] = NX-LX1A;
        X2a[0] = X2b[0]-LX2A;
        X3a[0] = X2b[0]-LX2A/2-LX3A/2;
        X3b[0] = X2b[0]-LX2A/2+LX3A/2;
        X4a[0] = X2b[0]-LX2A/2-LX4A/2;
```

```
X4b[0] = X2b[0]-LX2A/2+LX4A/2;
X5a[0] = X4a[0]-LX5A;
X5b[0] = X4b[0]+LX5A;
X6a1[0]= X4a[0]-LX5A/2-LX6A/2;
X6b1[0]= X4a[0]-LX5A/2+LX6A/2;
X6a2[0]= X4b[0]+LX5A/2-LX6A/2;
X6b2[0]= X4b[0]+LX5A/2+LX6A/2;
X7a1[0]= X4a[0]-LX5A/2-LX7A/2;
X7b1[0]= X4a[0]-LX5A/2+LX7A/2;
X7a2[0]= X4b[0]+LX5A/2-LX7A/2;
X7b2[0]= X4b[0]+LX5A/2+LX7A/2;

Y1a[0] = centerY-LY1A/2;
Y1b[0] = centerY+LY1A/2;
Y2a[0] = centerY-LY2A/2;
Y2b[0] = centerY+LY2A/2;
Y3a[0] = Y2a[0]-LY3A;
Y3b[0] = Y2b[0]+LY3A;
Y4a1[0]= Y2a[0]-LY3A/2-LY4A/2;
Y4b1[0]= Y2a[0]-LY3A/2+LY4A/2;
Y4a2[0]= Y2b[0]+LY3A/2-LY4A/2;
Y4b2[0]= Y2b[0]+LY3A/2+LY4A/2;
Y5a1[0]= Y2a[0]-LY3A/2-LY5A/2;
Y5b1[0]= Y2a[0]-LY3A/2+LY5A/2;
Y5a2[0]= Y2b[0]+LY3A/2-LY5A/2;
Y5b2[0]= Y2b[0]+LY3A/2+LY5A/2;
Y6a1[0]= Y5a1[0]-LY6A;
Y6b1[0]= Y5b1[0]+LY6A;
Y6a2[0]= Y5a2[0]-LY6A;
Y6b2[0]= Y5b2[0]+LY6A;
Y7a1[0]= Y5a1[0]-LY6A/2-LY7A/2;
Y7b1[0]= Y5a1[0]-LY6A/2+LY7A/2;
Y7a2[0]= Y5b1[0]+LY6A/2-LY7A/2;
Y7b2[0]= Y5b1[0]+LY6A/2+LY7A/2;
Y7a3[0]= Y5a2[0]-LY6A/2-LY7A/2;
Y7b3[0]= Y5a2[0]-LY6A/2+LY7A/2;
Y7a4[0]= Y5b2[0]+LY6A/2-LY7A/2;
Y7b4[0]= Y5b2[0]+LY6A/2+LY7A/2;

Z1a[0] = (NZ3+NZ2)/2-LZ1A/2;
Z1b[0] = Z1a[0]+LZ1A;
Z2a[0] = (NZ3+NZ2)/2-LZ2A/2;
Z2b[0] = Z2a[0]+LZ2A;
Z3a[0] = (NZ3+NZ2)/2-LZ3A/2;
Z3b[0] = Z3a[0]+LZ3A;
Z4a[0] = Z3a[0]-LZ4A;
Z4b[0] = Z3b[0]+LZ4A;
Z5a1[0]= Z3a[0]-LZ4A/2-LZ5A/2;
Z5b1[0]= Z3a[0]-LZ4A/2+LZ5A/2;
Z5a2[0]= Z3b[0]+LZ4A/2-LZ5A/2;
Z5b2[0]= Z3b[0]+LZ4A/2+LZ5A/2;
Z6a1[0]= Z3a[0]-LZ4A/2-LZ6A/2;
Z6b1[0]= Z3a[0]-LZ4A/2+LZ6A/2;
Z6a2[0]= Z3b[0]+LZ4A/2-LZ6A/2;
Z6b2[0]= Z3b[0]+LZ4A/2+LZ6A/2;
Z7a1[0]= Z6a1[0]-LZ7A;
```

```
Z7b1[0]= Z6b1[0]+LZ7A;
Z7a2[0]= Z6a2[0]-LZ7A;
Z7b2[0]= Z6b2[0]+LZ7A;

X1[1]  = NX;
X2b[1] = NX-LX1A;
X2a[1] = X2b[1]-LX2A;
X3a[1] = X2b[1]-LX2A/2-LX3A/2;
X3b[1] = X2b[1]-LX2A/2+LX3A/2;
X4a[1] = X2b[1]-LX2A/2-LX4A/2;
X4b[1] = X2b[1]-LX2A/2+LX4A/2;
X5a[1] = X4a[1]-LX5A;
X5b[1] = X4b[1]+LX5A;
X6a1[1]= X4a[1]-LX5A/2-LX6A/2;
X6b1[1]= X4a[1]-LX5A/2+LX6A/2;
X6a2[1]= X4b[1]+LX5A/2-LX6A/2;
X6b2[1]= X4b[1]+LX5A/2+LX6A/2;
X7a1[1]= X4a[1]-LX5A/2-LX7A/2;
X7b1[1]= X4a[1]-LX5A/2+LX7A/2;
X7a2[1]= X4b[1]+LX5A/2-LX7A/2;
X7b2[1]= X4b[1]+LX5A/2+LX7A/2;

Y1a[1] = centerY-LY1A/2;
Y1b[1] = centerY+LY1A/2;
Y2a[1] = centerY-LY2A/2;
Y2b[1] = centerY+LY2A/2;
Y3a[1] = Y2a[1]-LY3A;
Y3b[1] = Y2b[1]+LY3A;
Y4a1[1]= Y2a[1]-LY3A/2-LY4A/2;
Y4b1[1]= Y2a[1]-LY3A/2+LY4A/2;
Y4a2[1]= Y2b[1]+LY3A/2-LY4A/2;
Y4b2[1]= Y2b[1]+LY3A/2+LY4A/2;
Y5a1[1]= Y2a[1]-LY3A/2-LY5A/2;
Y5b1[1]= Y2a[1]-LY3A/2+LY5A/2;
Y5a2[1]= Y2b[1]+LY3A/2-LY5A/2;
Y5b2[1]= Y2b[1]+LY3A/2+LY5A/2;
Y6a1[1]= Y5a1[1]-LY6A;
Y6b1[1]= Y5b1[1]+LY6A;
Y6a2[1]= Y5a2[1]-LY6A;
Y6b2[1]= Y5b2[1]+LY6A;
Y7a1[1]= Y5a1[1]-LY6A/2-LY7A/2;
Y7b1[1]= Y5a1[1]-LY6A/2+LY7A/2;
Y7a2[1]= Y5b1[1]+LY6A/2-LY7A/2;
Y7b2[1]= Y5b1[1]+LY6A/2+LY7A/2;
Y7a3[1]= Y5a2[1]-LY6A/2-LY7A/2;
Y7b3[1]= Y5a2[1]-LY6A/2+LY7A/2;
Y7a4[1]= Y5b2[1]+LY6A/2-LY7A/2;
Y7b4[1]= Y5b2[1]+LY6A/2+LY7A/2;

Z1a[1] = (NZ3+NZ2)/2-LZ1A/2;
Z1b[1] = Z1a[1]+LZ1A;
Z2a[1] = (NZ3+NZ2)/2-LZ2A/2;
Z2b[1] = Z2a[1]+LZ2A;
Z3a[1] = (NZ3+NZ2)/2-LZ3A/2;
Z3b[1] = Z3a[1]+LZ3A;
Z4a[1] = Z3a[1]-LZ4A;
```

```
Z4b[1] = Z3b[1]+LZ4A;
Z5a1[1]= Z3a[1]-LZ4A/2-LZ5A/2;
Z5b1[1]= Z3a[1]-LZ4A/2+LZ5A/2;
Z5a2[1]= Z3b[1]+LZ4A/2-LZ5A/2;
Z5b2[1]= Z3b[1]+LZ4A/2+LZ5A/2;
Z6a1[1]= Z3a[1]-LZ4A/2-LZ6A/2;
Z6b1[1]= Z3a[1]-LZ4A/2+LZ6A/2;
Z6a2[1]= Z3b[1]+LZ4A/2-LZ6A/2;
Z6b2[1]= Z3b[1]+LZ4A/2+LZ6A/2;
Z7a1[1]= Z6a1[1]-LZ7A;
Z7b1[1]= Z6b1[1]+LZ7A;
Z7a2[1]= Z6a2[1]-LZ7A;
Z7b2[1]= Z6b2[1]+LZ7A;

//seperate the blood vessels
int dX0 = 0; //offset of arteries
int dX1 = 38;  //offset of veins
int dZ = 150;
X1[0]  -= dX0;
X2a[0] -= dX0;
X2b[0] -= dX0;
X3a[0] -= dX0;
X3b[0] -= dX0;
X4a[0] -= dX0;
X4b[0] -= dX0;
X5a[0] -= dX0;
X5b[0] -= dX0;
X6a1[0]-= dX0;
X6b1[0]-= dX0;
X6a2[0]-= dX0;
X6b2[0]-= dX0;
X7a1[0]-= dX0;
X7b1[0]-= dX0;
X7a2[0]-= dX0;
X7b2[0]-= dX0;

X1[1]  += 0;//if dX1>0, then X1[1] will be out of range, so LX1B-dX1
X2a[1] += dX1;
X2b[1] += dX1;
X3a[1] += dX1;
X3b[1] += dX1;
X4a[1] += dX1;
X4b[1] += dX1;
X5a[1] += dX1;
X5b[1] += dX1;
X6a1[1]+= dX1;
X6b1[1]+= dX1;
X6a2[1]+= dX1;
X6b2[1]+= dX1;
X7a1[1]+= dX1;
X7b1[1]+= dX1;
X7a2[1]+= dX1;
X7b2[1]+= dX1;

Z1a[0] += dZ;
Z1b[0] += dZ;
```

```
            Z2a[0] += dZ;
            Z2b[0] += dZ;
            Z3a[0] += dZ;
            Z3b[0] += dZ;
            Z4a[0] += dZ;
            Z4b[0] += dZ;
            Z5a1[0]+= dZ;
            Z5b1[0]+= dZ;
            Z5a2[0]+= dZ;
            Z5b2[0]+= dZ;
            Z6a1[0]+= dZ;
            Z6b1[0]+= dZ;
            Z6a2[0]+= dZ;
            Z6b2[0]+= dZ;
            Z7a1[0]+= dZ;
            Z7b1[0]+= dZ;
            Z7a2[0]+= dZ;
            Z7b2[0]+= dZ;

            Z1a[1] -= dZ;
            Z1b[1] -= dZ;
            Z2a[1] -= dZ;
            Z2b[1] -= dZ;
            Z3a[1] -= dZ;
            Z3b[1] -= dZ;
            Z4a[1] -= dZ;
            Z4b[1] -= dZ;
            Z5a1[1]-= dZ;
            Z5b1[1]-= dZ;
            Z5a2[1]-= dZ;
            Z5b2[1]-= dZ;
            Z6a1[1]-= dZ;
            Z6b1[1]-= dZ;
            Z6a2[1]-= dZ;
            Z6b2[1]-= dZ;
            Z7a1[1]-= dZ;
            Z7b1[1]-= dZ;
            Z7a2[1]-= dZ;
            Z7b2[1]-= dZ;

            return;
}

void Reset()
{
            memset(a,   0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            memset(b,   0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            memset(c,   0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            memset(d,   0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));

            //memset(Tt ,    37, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            //memset(Tt_n1,  37, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            //memset(Tt_n1_I,37, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
            for (int i=0; i<=NX; i++)
                        for (int j=0; j<=NY; j++)
                                    for (int k=0; k<=NZ3; k++)
```

```
                {
                        Tt[i][j][k] = 34.0;
                        Tt_n1[i][j][k] = 34.0;
                        Tt_n1_1[i][j][k] = 34.0;
                        //Tt_n_1[i][j][k] = 34.0;
                        Ut[i][j][k] = 34.0;
                        Ut_n1[i][j][k] = 34.0;
                        Damage_t[i][j][k] = 0.0;
                        Damage_n1[i][j][k] = 0.0;
                }
//memset(Tb1[0], 37, sizeof(double)*(LZ1A+1));
//memset(Tb1_n1[0], 37, sizeof(double)*(LZ1A+1));
//memset(Tv1_n1[0], 37, sizeof(double)*(LZ1A+1));
for (j=0; j<=LX1A; j++)
{
        Tbd1[0][0][j] = 34.0;
        Tbd1[0][1][j] = 34.0;
        Tbd1[0][2][j] = 34.0;
        Tbd1[0][3][j] = 34.0;
        Tb1_n1[0][j] = 37.0;
        Tv1_n1[0][j] = 37.0;
}
//memset(Tb1[1], 37, sizeof(double)*(LZ1B+1));
//memset(Tb1_n1[1], 37, sizeof(double)*(LZ1B+1));
//memset(Tv1_n1[1], 37, sizeof(double)*(LZ1B+1));
for (j=0; j<=LX1B; j++)
{
        Tbd1[1][0][j] = 34.0;
        Tbd1[1][1][j] = 34.0;
        Tbd1[1][2][j] = 34.0;
        Tbd1[1][3][j] = 34.0;
        Tb1_n1[1][j] = 37.0;
        Tv1_n1[1][j] = 37.0;
}
//memset(Tb2[0], 37, sizeof(double)*(LX2A+1));
//memset(Tb2_n1[0], 37, sizeof(double)*(LX2A+1));
//memset(Tv2_n1[0], 37, sizeof(double)*(LX2A+1));
for (j=0; j<=LY2A; j++)
{
        Tbd2[0][0][j] = 34.0;
        Tbd2[0][1][j] = 34.0;
        Tbd2[0][2][j] = 34.0;
        Tbd2[0][3][j] = 34.0;
        Tb2_n1[0][j] = 37.0;
        Tv2_n1[0][j] = 37.0;
}
//memset(Tb2[1], 37, sizeof(double)*(LX2B+1));
//memset(Tb2_n1[1], 37, sizeof(double)*(LX2B+1));
//memset(Tv2_n1[1], 37, sizeof(double)*(LX2B+1));
for (j=0; j<=LY2B; j++)
{
        Tbd2[1][0][j] = 34.0;
        Tbd2[1][1][j] = 34.0;
        Tbd2[1][2][j] = 34.0;
        Tbd2[1][3][j] = 34.0;
        Tb2_n1[1][j] = 37.0;
```

```
            Tv2_n1[1][j] = 37.0;
        }
        //memset(Tb3[0], 37, sizeof(double)*(LZ3A+1)*2);
        //memset(Tb3_n1[0], 37, sizeof(double)*(LZ3A+1)*2);
        //memset(Tv3_n1[0], 37, sizeof(double)*(LZ3A+1)*2);
        for (j=0; j<=2*LZ3A+1; j++)
        {
            Tbd3[0][0][j] = 34.0;
            Tbd3[0][1][j] = 34.0;
            Tbd3[0][2][j] = 34.0;
            Tbd3[0][3][j] = 34.0;
            Tb3_n1[0][j] = 37.0;
            Tv3_n1[0][j] = 37.0;
        }
        //memset(Tb3[1], 37, sizeof(double)*(LZ3B+1)*2);
        //memset(Tb3_n1[1], 37, sizeof(double)*(LZ3B+1)*2);
        //memset(Tv3_n1[1], 37, sizeof(double)*(LZ3B+1)*2);
        for (j=0; j<=2*LZ3B+1; j++)
        {
            Tbd3[1][0][j] = 34.0;
            Tbd3[1][1][j] = 34.0;
            Tbd3[1][2][j] = 34.0;
            Tbd3[1][3][j] = 34.0;
            Tb3_n1[1][j] = 37.0;
            Tv3_n1[1][j] = 37.0;
        }

        for (j=0; j<=4*LX4A+3; j++)
        {
            Tbd4[0][0][j] = 34.0;
            Tbd4[0][1][j] = 34.0;
            Tbd4[0][2][j] = 34.0;
            Tbd4[0][3][j] = 34.0;
            Tb4_n1[0][j] = 37.0;
            Tv4_n1[0][j] = 37.0;
        }

        for (j=0; j<=4*LX4B+3; j++)
        {
            Tbd4[1][0][j] = 34.0;
            Tbd4[1][1][j] = 34.0;
            Tbd4[1][2][j] = 34.0;
            Tbd4[1][3][j] = 34.0;
            Tb4_n1[1][j] = 37.0;
            Tv4_n1[1][j] = 37.0;
        }

    for (j=0; j<=8*LY5A+7; j++)
        {
            Tbd5[0][0][j] = 34.0;
            Tbd5[0][1][j] = 34.0;
            Tbd5[0][2][j] = 34.0;
            Tbd5[0][3][j] = 34.0;
            Tb5_n1[0][j] = 37.0;
            Tv5_n1[0][j] = 37.0;
        }
```

```
        for (j=0; j<=8*LY5B+7; j++)
        {
                Tbd5[1][0][j] = 34.0;
                Tbd5[1][1][j] = 34.0;
                Tbd5[1][2][j] = 34.0;
                Tbd5[1][3][j] = 34.0;
                Tb5_n1[1][j] = 37.0;
                Tv5_n1[1][j] = 37.0;
        }

        for (j=0; j<=16*LZ6A+15; j++)
        {
                Tbd6[0][0][j] = 34.0;
                Tbd6[0][1][j] = 34.0;
                Tbd6[0][2][j] = 34.0;
                Tbd6[0][3][j] = 34.0;
                Tb6_n1[0][j] = 37.0;
                Tv6_n1[0][j] = 37.0;
        }

        for (j=0; j<=16*LZ6B+15; j++)
        {
                Tbd6[1][0][j] = 34.0;
                Tbd6[1][1][j] = 34.0;
                Tbd6[1][2][j] = 34.0;
                Tbd6[1][3][j] = 34.0;
                Tb6_n1[1][j] = 37.0;
                Tv6_n1[1][j] = 37.0;
        }

        for (j=0; j<=32*LX7A+31; j++)
        {
                Tbd7[0][0][j] = 34.0;
                Tbd7[0][1][j] = 34.0;
                Tbd7[0][2][j] = 34.0;
                Tbd7[0][3][j] = 34.0;
                Tb7_n1[0][j] = 37.0;
                Tv7_n1[0][j] = 37.0;
        }

        for (j=0; j<=32*LX7B+31; j++)
        {
                Tbd7[1][0][j] = 34.0;
                Tbd7[1][1][j] = 34.0;
                Tbd7[1][2][j] = 34.0;
                Tbd7[1][3][j] = 34.0;
                Tb7_n1[1][j] = 37.0;
                Tv7_n1[1][j] = 37.0;
        }

        //Tb1[0][0] = 37.0 + THETA0;
        Tb1_n1[0][0] = 37.0 + THETA0;

        return;
}
```

```
int     CalcAll()
{
        double maxErr, oldE; //sum of square error of tissue temperatures
        FILE *fp1;
        FILE *fp2;
        fp1 = fopen("T_center_t_0.txt","w");
        fp2 = fopen("T_center_t_100.txt","w");
        Reset();
        t = 0;

        ////////////////////////////////////////////////////////////
        getTv_blood(0);
        getTv_blood(1);
        //calculate blood temperature based on given vessel temperature
        CalcTb();
        CalcTb2();
        reloadbloodeteperature(0);
        reloadbloodeteperature(1);
        CalcVessel(0);
        CalcVessel(1);

        ////////////////////////////////////////////////////////////
        while((t*deltaT < TOTAL_T))
        {       //while loop begin
                t++;
                I = 0;
                maxErr = 0.0;
                oldE = 99999999.0f;

                do{ //I iteration, do while loop begin
                        I++;
                        //Calculate tissue temperature
                        maxErr = CalcTt();
                        printf("t:%2d I:%d Err:%5.4lf T0:%10.6f T1:%10.6f
                        T2:%10.6f\n", t, I, maxErr, Tt_n1[NX/2][NY/2][0],
                        Tt_n1[0][NY/2][0], Tt_n1[NX/2][NY/2][100]);

                  if(maxErr>=oldE)
                  {
                    writeLog("=======unstable======");
                    #ifdef SCREEN_OUT
                        printf("============\n");
                        #endif
                        writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
                        break;
                      }
                      oldE = maxErr;
                }while(maxErr>Err_I);  //do while I loop end, Err_I=0.001

fprintf(fp2,"%5d   %10.6f\n",t, Tt_n1[NX/2][NY/2][100]);

for (i=0; i<=NX; i++){
  for (j=0; j<=NY; j++){
     for (z=0; z<=NZ1; z++){
        Ut_n1[i][j][z] = - Ut[i][j][z] + (1.0+tau*Wb1*Cb1/(p1*C1)
```

```
                +2.0*tau/deltaT)*Tt_n1[i][j][z]
                +(1.0+tau*Wb1*Cb1/(p1*C1)-2.0*tau/deltaT)*Tt[i][j][z];
        }
        for (z=NZ1+1; z<=NZ2; z++){
          Ut_n1[i][j][z] = - Ut[i][j][z] + (1.0+tau*Wb2*Cb2/(p2*C2)
                +2.0*tau/deltaT)*Tt_n1[i][j][z]
                +(1.0+tau*Wb2*Cb2/(p2*C2)-2.0*tau/deltaT)*Tt[i][j][z];
        }
    for (z=NZ2+1; z<=NZ3; z++){
          Ut_n1[i][j][z] = - Ut[i][j][z] + (1.0+tau*Wb3*Cb3/(p3*C3)
                +2.0*tau/deltaT)*Tt_n1[i][j][z]
                +(1.0+tau*Wb3*Cb3/(p3*C3)-2.0*tau/deltaT)*Tt[i][j][z];
    }
   }
}

//Calculation of damage
for (i=0; i<=NX; i++){
   for (j=0; j<=NY; j++){
      for (z=0; z<=NZ3; z++){
        if (Tt_n1[i][j][z] <=50.0){
          energy = 4.18e+5;
          frequency = 4.322e+64;
        }
        else{
          energy = 6.69e+5;
          frequency = 9.389e+104;}
          Damage_n1[i][j][z] = Damage_t[i][j][z]
          + frequency*exp(- energy/(gas*(Tt_n1[i][j][z]+273.0)))*deltaT;
      }
    }
  }

  memcpy(Ut, Ut_n1, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
  memcpy(Tt, Tt_n1, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
  memcpy(Damage_t, Damage_n1, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));

} //Time while loop block end, upper bound TOTAL_T
      fclose(fp2);
      printf("Record the temperature of the last second\n");
      //Record the temperature of the last second
      writeSquareXZ(0, 0, NX, NZ3, NY/2, t, I);
      writeSquareXY(0, 0, NX, NY, (Z1a[0]+Z1b[0])/2, t, I);
      writeSquareXY(0, 0, NX, NY, (Z1a[1]+Z1b[1])/2, t, I);
      writeSquareXZ(0, 0, NX, NZ3, (Y1a[0]+Y1b[0])/2, t, I);
      writeSquareXZ(0, 0, NX, NZ3, (Y4a1[0]+Y4b1[0])/2, t, I);
      writeSquareXZ(0, 0, NX, NZ3, (Y7a2[0]+Y7b2[0])/2, t, I);
      writeSquareYZ(0, 0, NY, NZ3, NX/2, t, I);
      writeSquareYZ(0, 0, NY, NZ3, (X3a[0]+X3b[0])/2, t, I);
      writeSquareYZ(0, 0, NY, NZ3, (X3a[1]+X3b[1])/2, t, I);
      writeSquareYZ(0, 0, NY, NZ3, (X6a2[0]+X6b2[0])/2, t, I);
      writeSquareYZ(0, 0, NY, NZ3, (X6a1[1]+X6b1[1])/2, t, I);
      writeSquareXZ_Damage(0, 0, NX, NZ3, NY/2, t, I);
      writeSquareXZ_Damage(0, 0, NX, NZ3, (Y1a[0]+Y1b[0])/2, t, I);
      writeSquareXZ_Damage(0, 0, NX, NZ3, (Y4a1[0]+Y4b1[0])/2, t, I);
      writeSquareXZ_Damage(0, 0, NX, NZ3, (Y7a2[0]+Y7b2[0])/2, t, I);
```

```
        writeSquareYZ_Damage(0, 0, NY, NZ3, NX/2, t, I);
        writeSquareYZ_Damage(0, 0, NY, NZ3, (X3a[0]+X3b[0])/2, t, I);
        writeSquareYZ_Damage(0, 0, NY, NZ3, (X3a[1]+X3b[1])/2, t, I);
        writeSquareYZ_Damage(0, 0, NY, NZ3, (X6a2[0]+X6b2[0])/2, t, I);
        writeSquareYZ_Damage(0, 0, NY, NZ3, (X6a1[1]+X6b1[1])/2, t, I);

        return(1);
}//programm finish

void setboarderVariable(int index)
{
        x1a=X1[index];x2a=X2a[index];x2b=X2b[index];x3a=X3a[index];x3b=X3b[index];
        x4a=X4a[index];x4b=X4b[index];
        x5a =X5a[index];x5b=X5b[index];x6a1=X6a1[index];x6b1=X6b1[index];
        x6a2=X6a2[index];x6b2=X6b2[index];
        x7a1 =X7a1[index];x7b1 =X7b1[index];x7a2 =X7a2[index];x7b2 =X7b2[index];
        y1a =Y1a[index];y1b=Y1b[index];y2a=Y2a[index];y2b=Y2b[index];
        y3a=Y3a[index];y3b=Y3b[index];
        y4a1=Y4a1[index];y4b1=Y4b1[index];y4a2=Y4a2[index];y4b2=Y4b2[index];
        y5a1=Y5a1[index];y5b1=Y5b1[index];
        y5a2=Y5a2[index];y5b2=Y5b2[index];y6a1=Y6a1[index];y6b1=Y6b1[index];
        y6a2=Y6a2[index];y6b2=Y6b2[index];
        y7a1=Y7a1[index];y7b1=Y7b1[index];y7a2=Y7a2[index];y7b2=Y7b2[index];
        y7a3=Y7a3[index];y7b3=Y7b3[index];
        y7a4=Y7a4[index];y7b4=Y7b4[index];z1a=Z1a[index];z1b=Z1b[index];
        z2a=Z2a[index];z2b=Z2b[index];
        z3a=Z3a[index];z3b=Z3b[index];z4a=Z4a[index];z4b=Z4b[index];
        z5a1=Z5a1[index];z5b1=Z5b1[index];
        z5a2=Z5a2[index];z5b2=Z5b2[index];z6a1=Z6a1[index];z6b1=Z6b1[index];
        z6a2=Z6a2[index];
        z6b2=Z6b2[index];z7a1=Z7a1[index];z7b1=Z7b1[index];z7a2=Z7a2[index];
        z7b2=Z7b2[index];
        lx1 = index==0?LX1A:LX1B;
        ly1 = index==0?LY1A:LY1B;
        lz1 = index==0?LZ1A:LZ1B;
        lx2 = index==0?LX2A:LX2B;
        ly2 = index==0?LY2A:LY2B;
        lz2 = index==0?LZ2A:LZ2B;
        lx3 = index==0?LX3A:LX3B;
        ly3 = index==0?LY3A:LY3B;
        lz3 = index==0?LZ3A:LZ3B;
        ly4 = index==0?LY4A:LY4B;
        lz4 = index==0?LZ4A:LZ4B;
        lx4 = index==0?LX4A:LX4B;
        lx5 = index==0?LX5A:LX5B;
        ly5 = index==0?LY5A:LY5B;
        lz5 = index==0?LZ5A:LZ5B;
        lx6 = index==0?LX6A:LX6B;
        ly6 = index==0?LY6A:LY6B;
        lz6 = index==0?LZ6A:LZ6B;
        lx7 = index==0?LX7A:LX7B;
        ly7 = index==0?LY7A:LY7B;
        lz7 = index==0?LZ7A:LZ7B;
        cenY = (y1a + y1b)/2;//actual center y after seperate the arteries and veins
        cenZ = (z1a + z1b)/2;
        cenX = (x2a + x2b)/2;  //symetry center for x coordinate from level 2
```

```
    return;
}

int getTv_blood(int index)
{
//interpolate vessel temperature from the tissue temperature near the vessel
//****LEFT and RIGHT side could be different if the blood vessel has an offset to //the center

 setboarderVariable(index); //set the common blood boarder variables

 //first level
 for(i=0;i<lx1;i++)
    Tv1_n1[index][i] = ( Tbd1[index][0][i] + Tbd1[index][1][i]
    + Tbd1[index][2][i] + Tbd1[index][3][i] ) / 4.0f;
 //second level
 for( i=0;i<=ly2;i++) //i=0 & LY2 are on the blood vessels
    Tv2_n1[index][i] = ( Tbd2[index][0][i] + Tbd2[index][1][i]
    + Tbd2[index][2][i] + Tbd2[index][3][i] ) / 4.0f;
//third level
for (k=0;k<2;k++)
{
    for(i=0;i<=lz3;i++)
    {
            Tv3_n1[index][i+k*(lz3+1)] = (Tbd3[index][0][i+k*(lz3+1)] +
            Tbd3[index][1][i+k*(lz3+1)] + Tbd3[index][2][i+k*(lz3+1)] +
            Tbd3[index][3][i+k*(lz3+1)]  ) / 4.0f;
    }
 }
 //fourth level
 k=0;
 for(r=0;r<2;r++)
 {
     for(j=0;j<2;j++)
     {
         for(i=0;i<=lx4;i++)
         {
            Tv4_n1[index][i+k*(lx4+1)] = ( Tbd4[index][0][i+k*(lx4+1)] +
            Tbd4[index][1][i+k*(lx4+1)] + Tbd4[index][2][i+k*(lx4+1)] +
            Tbd4[index][3][i+k*(lx4+1)] ) / 4.0f;
         }
         k++;
     }
 }
 //fifth level
 k=0;
 for(r=0;r<2;r++)//repeat variable in z coordinate
 {
    for(j=0;j<2;j++)//repeat variable in y coordinate
    {
        for(p=0;p<2;p++)//repeat variable in x coordinate
        {
            for(i=0;i<=ly5;i++)
            {
                Tv5_n1[index][i+k*(ly5+1)] = ( Tbd5[index][0][i+k*(ly5+1)] +
                Tbd5[index][1][i+k*(ly5+1)] + Tbd5[index][2][i+k*(ly5+1)] +
```

```
                Tbd5[index][3][i+k*(ly5+1)] ) / 4.0f;
            }
            k++;
        }
    }
}
//sixth level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
{
    for(j=0;j<2;j++)//repeat variable in y coordinate
    {
        for(p=0;p<2;p++)//repeat variable in x coordinate
        {
            for(q=0;q<2;q++) //repeat variable in inner y coordinate
            {
                for(i=0;i<=lz6;i++)
                {
                    Tv6_n1[index][i+k*(lz6+1)] = ( Tbd6[index][0][i+k*(lz6+1)] +
                    Tbd6[index][1][i+k*(lz6+1)] + Tbd6[index][2][i+k*(lz6+1)] +
                    Tbd6[index][3][i+k*(lz6+1)] ) / 4.0f;
                }
                k++;
            }
        }
    }
}
//seventh level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
{
    for(j=0;j<2;j++)//repeat variable in y coordinate
    {
        for(p=0;p<2;p++)//repeat variable in x coordinate
        {
            for(q=0;q<2;q++) //repeat variable in inner y coordinate
            {
                for(h=0;h<2;h++)//repeat variable in inner z coordinate
                {
                    for(i=0;i<=lx7;i++)
                    {
                        Tv7_n1[index][i+k*(lx7+1)] = ( Tbd7[index][0][i+k*(lx7+1)] +
                        Tbd7[index][1][i+k*(lx7+1)] + Tbd7[index][2][i+k*(lx7+1)] +
                        Tbd7[index][3][i+k*(lx7+1)] ) / 4.0f;
                    }
                    k++;
                }
            }
        }
    }
}
return(1);
}

int CalcTb() // calculate the value for artery
{
```

```
//int index = 0;
//double fk1, fk2, fk3, fk4;
inde = 0;
setboarderVariable(inde);//set the common blood boarder variables

Tb1_n1[inde][0] = 37.0 + THETA0;

//first level blood
for(i=1;i<=lx1;i++)
{
        fk1 = factor1[inde]*(Tv1_n1[inde][i-1]- Tb1_n1[inde][i-1]);

        fk2 = factor1[inde]*(Tv1_n1[inde][i-1]-(Tb1_n1[inde][i-1]+fk1/2.0));

        fk3 = factor1[inde]*(Tv1_n1[inde][i-1]-(Tb1_n1[inde][i-1]+fk2/2.0));

        fk4 = factor1[inde]*(Tv1_n1[inde][i-1]-(Tb1_n1[inde][i-1]+fk3));

        Tb1_n1[inde][i] = Tb1_n1[inde][i-1] + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
}

//second level blood
Tb2_n1[inde][ly2/2] = Tb1_n1[inde][lx1];
//the interface grid point between level 1 and level 2

//left part
for(i=ly2/2-1;i>=0;i--)
{
        fk1 = factor2[inde]*(Tv2_n1[inde][i+1]- Tb2_n1[inde][i+1]);

        fk2 = factor2[inde]*(Tv2_n1[inde][i+1]-(Tb2_n1[inde][i+1]+fk1/2.0));

        fk3 = factor2[inde]*(Tv2_n1[inde][i+1]-(Tb2_n1[inde][i+1]+fk2/2.0));

        fk4 = factor2[inde]*(Tv2_n1[inde][i+1]-(Tb2_n1[inde][i+1]+fk3));

        Tb2_n1[inde][i] = Tb2_n1[inde][i+1] + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
 }

//right part
for(i=ly2/2+1;i<=ly2;i++)
{
        fk1 = factor2[inde]*(Tv2_n1[inde][i-1]- Tb2_n1[inde][i-1]);

        fk2 = factor2[inde]*(Tv2_n1[inde][i-1]-(Tb2_n1[inde][i-1]+fk1/2.0));

        fk3 = factor2[inde]*(Tv2_n1[inde][i-1]-(Tb2_n1[inde][i-1]+fk2/2.0));

        fk4 = factor2[inde]*(Tv2_n1[inde][i-1]-(Tb2_n1[inde][i-1]+fk3));

        Tb2_n1[inde][i] = Tb2_n1[inde][i-1] + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
}

//third level blood
for(j=0;j<2;j++)
{    //the interface grid point between level 2 and level 3
```

```
        Tb3_n1[inde][lz3/2+j*(lz3+1)] = Tb2_n1[inde][j*ly2];
        //upper part
        for(i=lz3/2-1;i>=0;i--)
        {
           fk1 = factor3[inde]*(Tv3_n1[inde][i+1+j*(lz3+1)]-
            Tb3_n1[inde][i+1+j*(lz3+1)]);

           fk2 = factor3[inde]*(Tv3_n1[inde][i+1+j*(lz3+1)]-
                (Tb3_n1[inde][i+1+j*(lz3+1)]+fk1/2.0));

           fk3 = factor3[inde]*(Tv3_n1[inde][i+1+j*(lz3+1)]-
                (Tb3_n1[inde][i+1+j*(lz3+1)]+fk2/2.0));

           fk4 = factor3[inde]*(Tv3_n1[inde][i+1+j*(lz3+1)]-
               (Tb3_n1[inde][i+1+j*(lz3+1)]+fk3));

           Tb3_n1[inde][i+j*(lz3+1)] = Tb3_n1[inde][i+1+j*(lz3+1)]
                + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
     }

     //lower part
     for(i=lz3/2+1;i<=lz3;i++)
     {
           fk1 = factor3[inde]*(Tv3_n1[inde][i-1+j*(lz3+1)]
               - Tb3_n1[inde][i-1+j*(lz3+1)]);

           fk2 = factor3[inde]*(Tv3_n1[inde][i-1+j*(lz3+1)]
                -(Tb3_n1[inde][i-1+j*(lz3+1)]+fk1/2.0));

           fk3 = factor3[inde]*(Tv3_n1[inde][i-1+j*(lz3+1)]
               -(Tb3_n1[inde][i-1+j*(lz3+1)]+fk2/2.0));

           fk4 = factor3[inde]*(Tv3_n1[inde][i-1+j*(lz3+1)]
               -(Tb3_n1[inde][i-1+j*(lz3+1)]+fk3));

           Tb3_n1[inde][i+j*(lz3+1)] = Tb3_n1[inde][i-1+j*(lz3+1)]
                + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
     }
}

//fourth level blood
 k=0;
 for(r=0;r<2;r++)
 {
    for(j=0;j<2;j++)
    {
       //the order of level 3 and leve4 are exception than others  !!!!!!!!!!
       Tb4_n1[inde][lx4/2+k*(lx4+1)] = Tb3_n1[inde][r*lz3+j*(lz3+1)];

        //left part
        for(i=lx4/2-1;i>=0;i--)
        {
           fk1 = factor4[inde]*(Tv4_n1[inde][i+1+k*(lx4+1)]
                - Tb4_n1[inde][i+1+k*(lx4+1)]);

           fk2 = factor4[inde]*(Tv4_n1[inde][i+1+k*(lx4+1)]
```

```
                -(Tb4_n1[inde][i+1+k*(lx4+1)]+fk1/2.0));

            fk3 = factor4[inde]*(Tv4_n1[inde][i+1+k*(lx4+1)]
                -(Tb4_n1[inde][i+1+k*(lx4+1)]+fk2/2.0));

            fk4 = factor4[inde]*(Tv4_n1[inde][i+1+k*(lx4+1)]
                -(Tb4_n1[inde][i+1+k*(lx4+1)]+fk3));

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

    //right part
    for(i=lx4/2+1;i<=lx4;i++)
    {
            fk1 = factor4[inde]*(Tv4_n1[inde][i-1+k*(lx4+1)]
                - Tb4_n1[inde][i-1+k*(lx4+1)]);

            fk2 = factor4[inde]*(Tv4_n1[inde][i-1+k*(lx4+1)]
                -(Tb4_n1[inde][i-1+k*(lx4+1)]+fk1/2.0));

            fk3 = factor4[inde]*(Tv4_n1[inde][i-1+k*(lx4+1)]
                -(Tb4_n1[inde][i-1+k*(lx4+1)]+fk2/2.0));

            fk4 = factor4[inde]*(Tv4_n1[inde][i-1+k*(lx4+1)]
                -(Tb4_n1[inde][i-1+k*(lx4+1)]+fk3));

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

//fifth level blood
k=0;
for(r=0;r<4;r++)
{
    for(j=0;j<2;j++)
    {  //the interface grid point between level 4 and level 5
       Tb5_n1[inde][ly5/2+k*(ly5+1)] = Tb4_n1[inde][j*lx4+r*(lx4+1)];
       //left part
       for(i=ly5/2-1;i>=0;i--)
       {
            fk1 = factor5[inde]*(Tv5_n1[inde][i+1+k*(ly5+1)]
                - Tb5_n1[inde][i+1+k*(ly5+1)]);

            fk2 = factor5[inde]*(Tv5_n1[inde][i+1+k*(ly5+1)]
                -(Tb5_n1[inde][i+1+k*(ly5+1)]+fk1/2.0));

            fk3 = factor5[inde]*(Tv5_n1[inde][i+1+k*(ly5+1)]
                -(Tb5_n1[inde][i+1+k*(ly5+1)]+fk2/2.0));

            fk4 = factor5[inde]*(Tv5_n1[inde][i+1+k*(ly5+1)]
                -(Tb5_n1[inde][i+1+k*(ly5+1)]+fk3));
```

```
            Tb5_n1[inde][i+k*(ly5+1)] = Tb5_n1[inde][i+1+k*(ly5+1)]
                + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
    }

    //right part
    for(i=ly5/2+1;i<=ly5;i++)
    {
            fk1 = factor5[inde]*(Tv5_n1[inde][i-1+k*(ly5+1)]
                - Tb5_n1[inde][i-1+k*(ly5+1)]);

            fk2 = factor5[inde]*(Tv5_n1[inde][i-1+k*(ly5+1)]
                -(Tb5_n1[inde][i-1+k*(ly5+1)]+fk1/2.0));
            fk3 = factor5[inde]*(Tv5_n1[inde][i-1+k*(ly5+1)]
                -(Tb5_n1[inde][i-1+k*(ly5+1)]+fk2/2.0));

            fk4 = factor5[inde]*(Tv5_n1[inde][i-1+k*(ly5+1)]
                -(Tb5_n1[inde][i-1+k*(ly5+1)]+fk3));

           Tb5_n1[inde][i+k*(ly5+1)] = Tb5_n1[inde][i-1+k*(ly5+1)]
                + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
    }
    k++;
  }
}

//sixth level blood
k=0;
for(r=0;r<8;r++)
{
   for(j=0;j<2;j++)
   {
      //the interface grid point between level 5 and level 6
      Tb6_n1[inde][lz6/2+k*(lz6+1)] = Tb5_n1[inde][j*ly5+r*(ly5+1)];

      //left part
       for(i=lz6/2-1;i>=0;i--)
       {
          fk1 = factor6[inde]*(Tv6_n1[inde][i+1+k*(lz6+1)]
                - Tb6_n1[inde][i+1+k*(lz6+1)]);

          fk2 = factor6[inde]*(Tv6_n1[inde][i+1+k*(lz6+1)]
                -(Tb6_n1[inde][i+1+k*(lz6+1)]+fk1/2.0));

          fk3 = factor6[inde]*(Tv6_n1[inde][i+1+k*(lz6+1)]
                 -(Tb6_n1[inde][i+1+k*(lz6+1)]+fk2/2.0));

           fk4 = factor6[inde]*(Tv6_n1[inde][i+1+k*(lz6+1)]
                -(Tb6_n1[inde][i+1+k*(lz6+1)]+fk3));

           Tb6_n1[inde][i+k*(lz6+1)] = Tb6_n1[inde][i+1+k*(lz6+1)]
                + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
       }

      //right part
      for(i=lz6/2+1;i<=lz6;i++)
      {
```

```
            fk1 = factor6[inde]*(Tv6_n1[inde][i-1+k*(lz6+1)]
                - Tb6_n1[inde][i-1+k*(lz6+1)]);

            fk2 = factor6[inde]*(Tv6_n1[inde][i-1+k*(lz6+1)]
                -(Tb6_n1[inde][i-1+k*(lz6+1)]+fk1/2.0));

            fk3 = factor6[inde]*(Tv6_n1[inde][i-1+k*(lz6+1)]
                -(Tb6_n1[inde][i-1+k*(lz6+1)]+fk2/2.0));

            fk4 = factor6[inde]*(Tv6_n1[inde][i-1+k*(lz6+1)]
                -(Tb6_n1[inde][i-1+k*(lz6+1)]+fk3));

            Tb6_n1[inde][i+k*(lz6+1)] = Tb6_n1[inde][i-1+k*(lz6+1)]
                + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
        }
        k++;
    }
}

//seventh level blood
k=0;
for(r=0;r<16;r++)
{
    for(j=0;j<2;j++)
    {   //the interface grid point between level 6 and level 7
        Tb7_n1[inde][lx7/2+k*(lx7+1)] = Tb6_n1[inde][j*lz6+r*(lz6+1)];
        //left part
        for(i=lx7/2-1;i>=0;i--)
        {
            fk1 = factor7[inde]*(Tv7_n1[inde][i+1+k*(lx7+1)]
                - Tb7_n1[inde][i+1+k*(lx7+1)])
                + deltaZ*F7[inde]*Pdot*Tb7_n1[inde][i+1+k*(lx7+1)]/M7[inde];

            fk2 = factor7[inde]*(Tv7_n1[inde][i+1+k*(lx7+1)]
                -(Tb7_n1[inde][i+1+k*(lx7+1)]+fk1/2.0))
                + deltaZ*F7[inde]*Pdot*(Tb7_n1[inde][i+1+k*(lx7+1)]
                +fk1/2.0)/M7[inde];

            fk3 = factor7[inde]*(Tv7_n1[inde][i+1+k*(lx7+1)]
                -(Tb7_n1[inde][i+1+k*(lx7+1)]+fk2/2.0))
                + deltaZ*F7[inde]*Pdot*(Tb7_n1[inde][i+1+k*(lx7+1)]
                +fk2/2.0)/M7[inde];

            fk4 = factor7[inde]*(Tv7_n1[inde][i+1+k*(lx7+1)]
                -(Tb7_n1[inde][i+1+k*(lx7+1)]+fk3))
                + deltaZ*F7[inde]*Pdot*(Tb7_n1[inde][i+1+k*(lx7+1)]+fk3)/M7[inde];

            Tb7_n1[inde][i+k*(lx7+1)] = Tb7_n1[inde][i+1+k*(lx7+1)]
                + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
    }

//right part
for(i=lx7/2+1;i<=lx7;i++)
{
        fk1 = factor7[inde]*(Tv7_n1[inde][i-1+k*(lx7+1)]
            - Tb7_n1[inde][i-1+k*(lx7+1)])+ deltaZ*F7[inde]
```

```
                *Pdot*Tb7_n1[inde][i-1+k*(lx7+1)]/M7[inde];

            fk2 = factor7[inde]*(Tv7_n1[inde][i-1+k*(lx7+1)]
                  -(Tb7_n1[inde][i-1+k*(lx7+1)]+fk1/2.0))
                  + deltaZ*F7[inde]*Pdot*(Tb7_n1[inde][i-1+k*(lx7+1)]
                  +fk1/2.0)/M7[inde];

            fk3 = factor7[inde]*(Tv7_n1[inde][i-1+k*(lx7+1)]
                  -(Tb7_n1[inde][i-1+k*(lx7+1)]+fk2/2.0))
                  + deltaZ*F7[inde]*Pdot*(Tb7_n1[inde][i-1+k*(lx7+1)]
                  +fk2/2.0)/M7[inde];

            fk4 = factor7[inde]*(Tv7_n1[inde][i-1+k*(lx7+1)]
                  -(Tb7_n1[inde][i-1+k*(lx7+1)]+fk3))
                  + deltaZ*F7[inde]*Pdot*(Tb7_n1[inde][i-1+k*(lx7+1)]+fk3)/M7[inde];

            Tb7_n1[inde][i+k*(lx7+1)] = Tb7_n1[inde][i-1+k*(lx7+1)]
                  + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
      }
      k++;
    }
  }
  return(1);
}


int CalcTb2()  // Calculate the blood temperature of vein
{
  //int index = 1;
  //double fk1, fk2, fk3, fk4;
  inde = 1;
  setboarderVariable(inde);//set the common blood boarder variables

  //seventh level blood
  n=0;
  k=0;
  for(r=0;r<2;r++)//repeat variable in z coordinate
  {
     z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2;
     for(j=0;j<2;j++)//repeat variable in y coordinate
     {
        for(p=0;p<2;p++)//repeat variable in x coordinate
        {
           x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx7/2;
           x3=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2+lx7/2;
           for(q=0;q<2;q++) //repeat variable in inner y coordinate
           {
              y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2;
              for(h=0;h<2;h++)//repeat variable in inner z coordinate
              {
                 //left part
                 Tb7_n1[inde][0+k*(lx7+1)] = Tt_n1[x2-2][y2][z2+(2*h-1)*lz6/2
                       +(2*h-1)*lz7/2]; //entry point

                 for(i=1;i<=lx7/2;i++)
                 {
```

```
                    fk1 = factor7[inde]*(Tv7_n1[inde][i-1+k*(lx7+1)]
                          - Tb7_n1[inde][i-1+k*(lx7+1)]);

                    fk2 = factor7[inde]*(Tv7_n1[inde][i-1+k*(lx7+1)]
                          -(Tb7_n1[inde][i-1+k*(lx7+1)]+fk1/2.0));

                    fk3 = factor7[inde]*(Tv7_n1[inde][i-1+k*(lx7+1)]
                          -(Tb7_n1[inde][i-1+k*(lx7+1)]+fk2/2.0));

                    fk4 = factor7[inde]*(Tv7_n1[inde][i-1+k*(lx7+1)]
                          -(Tb7_n1[inde][i-1+k*(lx7+1)]+fk3));

                    Tb7_n1[inde][i+k*(lx7+1)] = Tb7_n1[inde][i-1+k*(lx7+1)]
                          + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;

                }
                //store the middle temperature (intersection of blood level 6 & 7)
                Tb6_n1[inde][h*lz6+n*(lz6+1)] = Tb7_n1[inde][lx7/2+k*(lx7+1)];

                //right part
                Tb7_n1[inde][lx7+k*(lx7+1)] = Tt_n1[x3+2][y2][z2+(2*h-1)
                          *lz6/2+(2*h-1)*lz7/2]; //entry point

                for(i=lx7-1;i>=lx7/2;i--)
                {
                    fk1 = factor7[inde]*(Tv7_n1[inde][i+1+k*(lx7+1)]
                          - Tb7_n1[inde][i+1+k*(lx7+1)]);

                   fk2 = factor7[inde]*(Tv7_n1[inde][i+1+k*(lx7+1)]
                          -(Tb7_n1[inde][i+1+k*(lx7+1)]+fk1/2.0));

                   fk3 = factor7[inde]*(Tv7_n1[inde][i+1+k*(lx7+1)]
                          -(Tb7_n1[inde][i+1+k*(lx7+1)]+fk2/2.0));

                   fk4 = factor7[inde]*(Tv7_n1[inde][i+1+k*(lx7+1)]
                          -(Tb7_n1[inde][i+1+k*(lx7+1)]+fk3));

                   Tb7_n1[inde][i+k*(lx7+1)] = Tb7_n1[inde][i+1+k*(lx7+1)]
                          + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;

                }
                //take the average of calculation of left and right sides
                Tb6_n1[inde][h*lz6+n*(lz6+1)] = (Tb6_n1[inde][h*lz6+n*(lz6+1)]
                          +Tb7_n1[inde][lx7/2+k*(lx7+1)])/2.0;
               Tb7_n1[inde][lx7/2+k*(lx7+1)]= Tb6_n1[inde][h*lz6+n*(lz6+1)];
               k++;
            }//h loop end
           n++;
        }//q loop end
     }//p loop end
   } //j loop end
}//r loop end


//sixth level blood
k=0;
```

```
for(r=0;r<8;r++)
{
    for(j=0;j<2;j++)
    {
      //upper part
       for(i=1;i<=lz6/2;i++)
       {
                fk1 = factor6[inde]*(Tv6_n1[inde][i-1+k*(lz6+1)]
                          - Tb6_n1[inde][i-1+k*(lz6+1)]);

                fk2 = factor6[inde]*(Tv6_n1[inde][i-1+k*(lz6+1)]
                          -(Tb6_n1[inde][i-1+k*(lz6+1)]+fk1/2.0));

                fk3 = factor6[inde]*(Tv6_n1[inde][i-1+k*(lz6+1)]
                          -(Tb6_n1[inde][i-1+k*(lz6+1)]+fk2/2.0));

                fk4 = factor6[inde]*(Tv6_n1[inde][i-1+k*(lz6+1)]
                          -(Tb6_n1[inde][i-1+k*(lz6+1)]+fk3));

                Tb6_n1[inde][i+k*(lz6+1)] = Tb6_n1[inde][i-1+k*(lz6+1)]
               + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
        }
        //store the middle temperature (intersection of blood level 5 & 6)
        Tb5_n1[inde][j*ly5+r*(ly5+1)] = Tb6_n1[inde][lz6/2+k*(lz6+1)];

       //lower part
       //Tb6_n1[inde][lz6+k*(lz6+1)] = Tb7_n1[inde][lx7/2+(2*k+1)*(lx7+1)];
       //the interface grid point between level 6 and level 7

       for(i=lz6-1;i>=lz6/2;i--)
       {
                fk1 = factor6[inde]*(Tv6_n1[inde][i+1+k*(lz6+1)]
                      - Tb6_n1[inde][i+1+k*(lz6+1)]);

                fk2 = factor6[inde]*(Tv6_n1[inde][i+1+k*(lz6+1)]
                     -(Tb6_n1[inde][i+1+k*(lz6+1)]+fk1/2.0));

                fk3 = factor6[inde]*(Tv6_n1[inde][i+1+k*(lz6+1)]
                      -(Tb6_n1[inde][i+1+k*(lz6+1)]+fk2/2.0));

                fk4 = factor6[inde]*(Tv6_n1[inde][i+1+k*(lz6+1)]
                      -(Tb6_n1[inde][i+1+k*(lz6+1)]+fk3));

                Tb6_n1[inde][i+k*(lz6+1)] = Tb6_n1[inde][i+1+k*(lz6+1)]
                     + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
        }
        //take the average of calculation of left and right sides
        Tb5_n1[inde][j*ly5+r*(ly5+1)] = (Tb5_n1[inde][j*ly5+r*(ly5+1)]
                +Tb6_n1[inde][lz6/2+k*(lz6+1)])/2.0;
        Tb6_n1[inde][lz6/2+k*(lz6+1)] = Tb5_n1[inde][j*ly5+r*(ly5+1)];
        k++;
    }
}
//fifth level blood
k=0;
for(r=0;r<4;r++)
```

```
{
    for(j=0;j<2;j++)
    {
        //left part
        for(i=1;i<=ly5/2;i++)
        {
                fk1 = factor5[inde]*(Tv5_n1[inde][i-1+k*(ly5+1)]
                        - Tb5_n1[inde][i-1+k*(ly5+1)]);

                fk2 = factor5[inde]*(Tv5_n1[inde][i-1+k*(ly5+1)]
                        -(Tb5_n1[inde][i-1+k*(ly5+1)]+fk1/2.0));

                fk3 = factor5[inde]*(Tv5_n1[inde][i-1+k*(ly5+1)]
                        -(Tb5_n1[inde][i-1+k*(ly5+1)]+fk2/2.0));

                fk4 = factor5[inde]*(Tv5_n1[inde][i-1+k*(ly5+1)]
                        -(Tb5_n1[inde][i-1+k*(ly5+1)]+fk3));

                Tb5_n1[inde][i+k*(ly5+1)] = Tb5_n1[inde][i-1+k*(ly5+1)]
                        + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
        }
        //store the middle temperature (intersection of blood level 4 & 5)
        Tb4_n1[inde][j*lx4+r*(lx4+1)] = Tb5_n1[inde][ly5/2+k*(ly5+1)];

        //right part
        for(i=ly5-1;i>=ly5/2;i--)
        {
                fk1 = factor5[inde]*(Tv5_n1[inde][i+1+k*(ly5+1)]
                        - Tb5_n1[inde][i+1+k*(ly5+1)]);

                fk2 = factor5[inde]*(Tv5_n1[inde][i+1+k*(ly5+1)]
                        -(Tb5_n1[inde][i+1+k*(ly5+1)]+fk1/2.0));

                fk3 = factor5[inde]*(Tv5_n1[inde][i+1+k*(ly5+1)]
                        -(Tb5_n1[inde][i+1+k*(ly5+1)]+fk2/2.0));

                fk4 = factor5[inde]*(Tv5_n1[inde][i+1+k*(ly5+1)]
                        -(Tb5_n1[inde][i+1+k*(ly5+1)]+fk3));

                Tb5_n1[inde][i+k*(ly5+1)] = Tb5_n1[inde][i+1+k*(ly5+1)]
                        + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
        }
        //take the average of calculation of left and right sides
        Tb4_n1[inde][j*lx4+r*(lx4+1)] = (Tb4_n1[inde][j*lx4+r*(lx4+1)]
                +Tb5_n1[inde][ly5/2+k*(ly5+1)])/2.0;
        Tb5_n1[inde][ly5/2+k*(ly5+1)] = Tb4_n1[inde][j*lx4+r*(lx4+1)];
        k++;
    }
}
//fourth level blood
k=0;
for(r=0;r<2;r++)
{
    for(j=0;j<2;j++)
    {
        //left part
```

```
            for(i=1;i<=lx4/2;i++)
            {
                fk1 = factor4[inde]*(Tv4_n1[inde][i-1+k*(lx4+1)]
                    - Tb4_n1[inde][i-1+k*(lx4+1)]);

                fk2 = factor4[inde]*(Tv4_n1[inde][i-1+k*(lx4+1)]
                    -(Tb4_n1[inde][i-1+k*(lx4+1)]+fk1/2.0));

                fk3 = factor4[inde]*(Tv4_n1[inde][i-1+k*(lx4+1)]
                    -(Tb4_n1[inde][i-1+k*(lx4+1)]+fk2/2.0));

                fk4 = factor4[inde]*(Tv4_n1[inde][i-1+k*(lx4+1)]
                    -(Tb4_n1[inde][i-1+k*(lx4+1)]+fk3));

                Tb4_n1[inde][i+k*(lx4+1)] = Tb4_n1[inde][i-1+k*(lx4+1)]
                    + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
            }
            //store the middle temperature (intersection of blood level 4 & 3)
            //the order between 3 and 4 are exception to others
            Tb3_n1[inde][r*lz3+j*(lz3+1)] = Tb4_n1[inde][lx4/2+k*(lx4+1)];

            //right part
            for(i=lx4-1;i>=lx4/2;i--)
            {
                fk1 = factor4[inde]*(Tv4_n1[inde][i+1+k*(lx4+1)]
                    - Tb4_n1[inde][i+1+k*(lx4+1)]);

                fk2 = factor4[inde]*(Tv4_n1[inde][i+1+k*(lx4+1)]
                    -(Tb4_n1[inde][i+1+k*(lx4+1)]+fk1/2.0));

                fk3 = factor4[inde]*(Tv4_n1[inde][i+1+k*(lx4+1)]
                    -(Tb4_n1[inde][i+1+k*(lx4+1)]+fk2/2.0));

                fk4 = factor4[inde]*(Tv4_n1[inde][i+1+k*(lx4+1)]
                    -(Tb4_n1[inde][i+1+k*(lx4+1)]+fk3));

                Tb4_n1[inde][i+k*(lx4+1)] = Tb4_n1[inde][i+1+k*(lx4+1)]
                    + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
            }
            //take the average of calculation of left and right sides
            Tb3_n1[inde][r*lz3+j*(lz3+1)] = (Tb3_n1[inde][r*lz3+j*(lz3+1)]
                            +Tb4_n1[inde][lx4/2+k*(lx4+1)])/2.0;
            Tb4_n1[inde][lx4/2+k*(lx4+1)] = Tb3_n1[inde][r*lz3+j*(lz3+1)];
            k++;
        }
}

//third level blood
for(j=0;j<2;j++)
{
    //upper part
    for(i=1;i<=lz3/2;i++)
    {
            fk1 = factor3[inde]*(Tv3_n1[inde][i-1+j*(lz3+1)]
                - Tb3_n1[inde][i-1+j*(lz3+1)]);
```

```
            fk2 = factor3[inde]*(Tv3_n1[inde][i-1+j*(lz3+1)]
                  -(Tb3_n1[inde][i-1+j*(lz3+1)]+fk1/2.0));

            fk3 = factor3[inde]*(Tv3_n1[inde][i-1+j*(lz3+1)]
                  -(Tb3_n1[inde][i-1+j*(lz3+1)]+fk2/2.0));

            fk4 = factor3[inde]*(Tv3_n1[inde][i-1+j*(lz3+1)]
                  -(Tb3_n1[inde][i-1+j*(lz3+1)]+fk3));

            Tb3_n1[inde][i+j*(lz3+1)] = Tb3_n1[inde][i-1+j*(lz3+1)]
                  + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
        }
        //store the middle temperature (intersection of blood level 2 & 3)
        Tb2_n1[inde][j*ly2] = Tb3_n1[inde][lz3/2+j*(lz3+1)];

        //lower part
        for(i=lz3-1;i>=lz3/2;i--)
        {
            fk1 = factor3[inde]*(Tv3_n1[inde][i+1+j*(lz3+1)]
                  - Tb3_n1[inde][i+1+j*(lz3+1)]);

            fk2 = factor3[inde]*(Tv3_n1[inde][i+1+j*(lz3+1)]
                  -(Tb3_n1[inde][i+1+j*(lz3+1)]+fk1/2.0));

            fk3 = factor3[inde]*(Tv3_n1[inde][i+1+j*(lz3+1)]
                  -(Tb3_n1[inde][i+1+j*(lz3+1)]+fk2/2.0));

            fk4 = factor3[inde]*(Tv3_n1[inde][i+1+j*(lz3+1)]
                  -(Tb3_n1[inde][i+1+j*(lz3+1)]+fk3));

            Tb3_n1[inde][i+j*(lz3+1)] = Tb3_n1[inde][i+1+j*(lz3+1)]
                  + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
        }
        //take the average of calculation of left and right sides
        Tb2_n1[inde][j*ly2] = (Tb2_n1[inde][j*ly2]+Tb3_n1[inde][lz3/2+j*(lz3+1)])/2.0;
        Tb3_n1[inde][lz3/2+j*(lz3+1)] = Tb2_n1[inde][j*ly2];
    }

    //second level blood
    //left part
     for(i=1;i<=ly2/2;i++)
     {
            fk1 = factor2[inde]*(Tv2_n1[inde][i-1]- Tb2_n1[inde][i-1]);

            fk2 = factor2[inde]*(Tv2_n1[inde][i-1]-(Tb2_n1[inde][i-1]+fk1/2.0));

            fk3 = factor2[inde]*(Tv2_n1[inde][i-1]-(Tb2_n1[inde][i-1]+fk2/2.0));

            fk4 = factor2[inde]*(Tv2_n1[inde][i-1]-(Tb2_n1[inde][i-1]+fk3));

            Tb2_n1[inde][i] = Tb2_n1[inde][i-1] + (fk1 + 2.0*fk2 + 2.0*fk3 + fk4)/6.0;
     }
      /////////store the middle temperature (intersection of blood level 2 & 1)
     Tb1_n1[inde][lx1] = Tb2_n1[inde][ly2/2];

      //right part
```

```
    for(i=ly2-1;i>=ly2/2;i--)
    {
            fk1 = factor2[inde]*(Tv2_n1[inde][i+1]- Tb2_n1[inde][i+1]);

            fk2 = factor2[inde]*(Tv2_n1[inde][i+1]-(Tb2_n1[inde][i+1]+fk1/2.0));

            fk3 = factor2[inde]*(Tv2_n1[inde][i+1]-(Tb2_n1[inde][i+1]+fk2/2.0));

            fk4 = factor2[inde]*(Tv2_n1[inde][i+1]-(Tb2_n1[inde][i+1]+fk3));

            Tb2_n1[inde][i] = Tb2_n1[inde][i+1] + (fk1 + 2.0*fk2  + 2.0*fk3 + fk4)/6.0;
    }
    //take the average of calculation of left and right sides
    Tb1_n1[inde][lx1] = (Tb1_n1[inde][lx1] + Tb2_n1[inde][ly2/2])/2.0;
    Tb2_n1[inde][ly2/2] = Tb1_n1[inde][lx1];

    //first level blood
     for(i=lx1-1;i>=0;i--)
    {
            fk1 = factor1[inde]*(Tv1_n1[inde][i+1]- Tb1_n1[inde][i+1]);

            fk2 = factor1[inde]*(Tv1_n1[inde][i+1]-(Tb1_n1[inde][i+1]+fk1/2.0));

            fk3 = factor1[inde]*(Tv1_n1[inde][i+1]-(Tb1_n1[inde][i+1]+fk2/2.0));

            fk4 = factor1[inde]*(Tv1_n1[inde][i+1]-(Tb1_n1[inde][i+1]+fk3));

            Tb1_n1[inde][i] = Tb1_n1[inde][i+1] + (fk1 + 2.0*fk2  + 2.0*fk3 + fk4)/6.0;
    }
    return(1);
}


double CalcTt()   //Calculate the tissue temperature
{
 //double maxErr, f;  //defined as gloab varialble by zeng
 //double aa, bb;     //defined as gloab varialble by zeng

 //initialize tridiagonal system
 memcpy(a, a0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
 memcpy(b, b0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
 memcpy(c, c0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));
 memset(d, 0, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));

 ////////////////////////////////////////////////////////////
 //initialize tri-diangonal system
 for(i=1;i<=NX-1;i++)
 {
   for(j=1;j<=NY-1;j++)
   {
     for(z=1;z<=NZ1-1;z++)
     {
         //Crank_Nicolson scheme for Laser-case
         /*f =( (2*p1*C1+Wb1*Cb1*deltaT)*Tt_n1_I[i][j][z] +
         (-2*p1*C1+Wb1*Cb1*deltaT)*Tt[i][j][z]-
         2*Wb1*Cb1*deltaT*(Tb3_n1[0][LZ3A])
```

```
            -k1*deltaT*( (Tt_n1_I[i-1][j][z]+Tt_n1_I[i+1][j][z]
            -2*Tt_n1_I[i][j][z])/(deltaX*deltaX)
            +(Tt_n1_I[i][j-1][z]+Tt_n1_I[i][j+1][z]
            -2*Tt_n1_I[i][j][z])/(deltaY*deltaY)
            +(Tt_n1_I[i][j][z-1]+Tt_n1_I[i][j][z+1]
            -2*Tt_n1_I[i][j][z])/(deltaZ*deltaZ))
            -k1*deltaT*( (Tt[i-1][j][z]+Tt[i+1][j][z]
            -2*Tt[i][j][z])/(deltaX*deltaX)
            +(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaY*deltaY)
            +(Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ) ) );

            d[i][j][z] = ( 2*p1*C1 + Wb1*Cb1*deltaT +(4*k1*deltaT)*(1.0f/(deltaX*deltaX)
            +1.0f/(deltaY*deltaY)) + (2*k1*deltaT)/(deltaZ*deltaZ) )
            * Tt_n1_I[i][j][z]- k1*deltaT /(deltaZ*deltaZ) * (Tt_n1_I[i][j][z-1]
            + Tt_n1_I[i][j][z+1]) - omega * f; */

            //Implicit scheme for Laser-case
            /*f = ( ( p1*C1+Wb1*Cb1*deltaT)*Tt_n1_I[i][j][z] - p1*C1*Tt[i][j][z]
            -Wb1*Cb1*deltaT*(Tb3_n1[0][LZ3A]) ////////////????????????
            -k1*deltaT*( (Tt_n1_I[i-1][j][z]+Tt_n1_I[i+1][j][z]
            -2.0*Tt_n1_I[i][j][z])/(deltaX*deltaX)
            +(Tt_n1_I[i][j-1][z]+Tt_n1_I[i][j+1][z]
            -2.0*Tt_n1_I[i][j][z])/(deltaY*deltaY)
            +(Tt_n1_I[i][j][z-1]+Tt_n1_I[i][j][z+1]
            -2.0*Tt_n1_I[i][j][z])/(deltaZ*deltaZ)) );

            d[i][j][z] = ( p1*C1 + Wb1*Cb1*deltaT + (4.0*k1*deltaT)*(1.0f/(deltaX*deltaX)
            +1.0f/(deltaY*deltaY)) + (2.0*k1*deltaT)/(deltaZ*deltaZ))
            * Tt_n1_I[i][j][z] - k1*deltaT /(deltaZ*deltaZ)
            * (Tt_n1_I[i][j][z-1] + Tt_n1_I[i][j][z+1])
            - omega * f; */
            //radiation_case: Dr Dai's method
            f = ( (2.0*p1*C1*(1.0+tau*Wb1*Cb1/(p1*C1)+2.0*tau/deltaT)
            +Wb1*Cb1*deltaT)*Tt_n1_I[i][j][z] + (2.0*p1*C1*(1.0+tau*Wb1*Cb1/(p1*C1)
            -2.0*tau/deltaT)+Wb1*Cb1*deltaT)*Tt[i][j][z]  -  4.0*p1*C1*Ut[i][j][z]
            - 2.0*Wb1*Cb1*deltaT*(Tb7_n1[0][LX7A])
            - k1*deltaT*( (Tt_n1_I[i-1][j][z]+Tt_n1_I[i+1][j][z]
            -2*Tt_n1_I[i][j][z])/(deltaX*deltaX)+(Tt_n1_I[i][j-1][z]
            +Tt_n1_I[i][j+1][z]-2*Tt_n1_I[i][j][z])/(deltaY*deltaY)
            +(Tt_n1_I[i][j][z-1]+Tt_n1_I[i][j][z+1]
        -2*Tt_n1_I[i][j][z])/(deltaZ*deltaZ) )
        -k1*deltaT*( (Tt[i-1][j][z]+Tt[i+1][j][z]
        -2*Tt[i][j][z])/(deltaX*deltaX)
        +(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaY*deltaY)
        +(Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ) ) );

        d[i][j][z] = ( 2.0*p1*C1*(1.0+tau*Wb1*Cb1/(p1*C1)+2.0*tau/deltaT)
        + Wb1*Cb1*deltaT + (4.0*k1*deltaT)*(1.0f/(deltaX*deltaX)
        +1.0f/(deltaY*deltaY)) + (2.0*k1*deltaT)/(deltaZ*deltaZ))
        * Tt_n1_I[i][j][z]- k1*deltaT /(deltaZ*deltaZ)
        * (Tt_n1_I[i][j][z-1] + Tt_n1_I[i][j][z+1])- omega * f;
    }
    d[i][j][NZ1]=0; //d=0 from -k1*u-1 + (k1+k2)*u -k2*u+1 = 0

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

```
    //Crank_Nicolson scheme for Laser-case
     /*f =( (2*p2*C2+Wb2*Cb2*deltaT)*Tt_n1_I[i][j][z] + (-
     2*p2*C2+Wb2*Cb2*deltaT)*Tt[i][j][z] -2*Wb2*Cb2*deltaT*(Tb3_n1[0][LZ3A])
    -k2*deltaT*( (Tt_n1_I[i-1][j][z]+Tt_n1_I[i+1][j][z]-2*Tt_n1_I[i][j][z])
     /(deltaX*deltaX)+(Tt_n1_I[i][j-1][z]+Tt_n1_I[i][j+1][z]
    -2*Tt_n1_I[i][j][z])/(deltaY*deltaY)+(Tt_n1_I[i][j][z-1]+Tt_n1_I[i][j][z+1]
    -2*Tt_n1_I[i][j][z])/(deltaZ*deltaZ) )
    -k2*deltaT*( (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX)
    +(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaY*deltaY)
    +(Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ) ) );

    d[i][j][z] = ( 2*p2*C2 + Wb2*Cb2*deltaT + (4*k2*deltaT)*(1.0f/(deltaX*deltaX)
    +1.0f/(deltaY*deltaY)) + (2*k2*deltaT)/(deltaZ*deltaZ)  ) * Tt_n1_I[i][j][z]
    - k2*deltaT /(deltaZ*deltaZ) * (Tt_n1_I[i][j][z-1] + Tt_n1_I[i][j][z+1])
    - omega * f;*/

    //Implicit scheme for Laser-case
    /*f =  ( ( p2*C2+Wb2*Cb2*deltaT)*Tt_n1_I[i][j][z] - p2*C2*Tt[i][j][z]
    -Wb2*Cb2*deltaT*(Tb3_n1[0][LZ3A])
    -k2*deltaT*( (Tt_n1_I[i-1][j][z]+Tt_n1_I[i+1][j][z]-
    2.0*Tt_n1_I[i][j][z])/(deltaX*deltaX)+(Tt_n1_I[i][j-1][z]+Tt_n1_I[i][j+1][z]-
    2.0*Tt_n1_I[i][j][z])/(deltaY*deltaY)+(Tt_n1_I[i][j][z-1]+Tt_n1_I[i][j][z+1]-
    2.0*Tt_n1_I[i][j][z])/(deltaZ*deltaZ)) );

     d[i][j][z] = ( p2*C2 + Wb2*Cb2*deltaT +
     (4.0*k2*deltaT)*(1.0f/(deltaX*deltaX)+1.0f/(deltaY*deltaY))+
     (2.0*k2*deltaT)/(deltaZ*deltaZ)) * Tt_n1_I[i][j][z] - k2*deltaT
     /(deltaZ*deltaZ) * (Tt_n1_I[i][j][z-1] + Tt_n1_I[i][j][z+1])- omega * f; */

     //radiation_case
     //radiation_case: Dr Dai's method
     f = (   (2.0*p2*C2*(1.0+tau*Wb2*Cb2/(p2*C2)+2.0*tau/deltaT)
     +Wb2*Cb2*deltaT)*Tt_n1_I[i][j][z] + (2.0*p2*C2*(1.0+tau*Wb2*Cb2/(p2*C2)-
     2.0*tau/deltaT)+Wb2*Cb2*deltaT)*Tt[i][j][z]  -  4.0*p2*C2*Ut[i][j][z] –
      2.0*Wb2*Cb2*deltaT*(Tb7_n1[0][LX7A])  - k2*deltaT
      *( (Tt_n1_I[i-1][j][z]+Tt_n1_I[i+1][j][z]-2*Tt_n1_I[i][j][z])/(deltaX*deltaX)
      +(Tt_n1_I[i][j-1][z]+Tt_n1_I[i][j+1][z]-2*Tt_n1_I[i][j][z])/(deltaY*deltaY)
      +(Tt_n1_I[i][j][z-1]+Tt_n1_I[i][j][z+1]-2*Tt_n1_I[i][j][z])/(deltaZ*deltaZ)
      -k2*deltaT*( (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX)
      +(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaY*deltaY)
      +(Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ) ) );

      d[i][j][z] = ( 2.0*p2*C2*(1.0+tau*Wb2*Cb2/(p2*C2)+2.0*tau/deltaT)
      + Wb2*Cb2*deltaT  + (4.0*k2*deltaT)*(1.0f/(deltaX*deltaX)+1.0f/
      (deltaY*deltaY))  + (2.0*k2*deltaT)/(deltaZ*deltaZ) )* Tt_n1_I[i][j][z]
      - k2*deltaT /(deltaZ*deltaZ) * (Tt_n1_I[i][j][z-1] + Tt_n1_I[i][j][z+1])-omega * f;
}

d[i][j][NZ2]=0; //d=0 from -k1*u-1 + (k1+k2)*u -k2*u+1 = 0

// third skin layer
for(z=NZ2+1;z<=NZ3-1;z++)
{
       //Crank_Nicolson scheme for Laser-case
       /*f = ( (2*p3*C3+Wb3*Cb3*deltaT)*Tt_n1_I[i][j][z] + (-
       2*p3*C3+Wb3*Cb3*deltaT)*Tt[i][j][z] -2*Wb3*Cb3*deltaT*(Tb3_n1[0][LZ3A])
```

```
      -k3*deltaT*( (Tt_n1_I[i-1][j][z]+Tt_n1_I[i+1][j][z]-
      2*Tt_n1_I[i][j][z])/(deltaX*deltaX)+(Tt_n1_I[i][j-1][z]+Tt_n1_I[i][j+1][z]-
      2*Tt_n1_I[i][j][z])/(deltaY*deltaY)+(Tt_n1_I[i][j][z-1]+Tt_n1_I[i][j][z+1]-
      2*Tt_n1_I[i][j][z])/(deltaZ*deltaZ)) -k3*deltaT
      *( (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX)
      +(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaY*deltaY)
      +(Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ) ));

      d[i][j][z] = ( 2*p3*C3 + Wb3*Cb3*deltaT +(4*k3*deltaT)*(1.0f/(deltaX*deltaX)
      +1.0f/(deltaY*deltaY)) + (2*k3*deltaT)/(deltaZ*deltaZ) )* Tt_n1_I[i][j][z]
     - k3*deltaT /(deltaZ*deltaZ) * (Tt_n1_I[i][j][z-1] + Tt_n1_I[i][j][z+1]) - omega * f;*/

      //Implicit scheme for Laser-case
      /*f =( ( p3*C3+Wb3*Cb3*deltaT)*Tt_n1_I[i][j][z] - p3*C3*Tt[i][j][z]
      -Wb3*Cb3*deltaT*(Tb3_n1[0][LZ3A]) -k3*deltaT*( (Tt_n1_I[i-1][j][z]
      +Tt_n1_I[i+1][j][z]-2.0*Tt_n1_I[i][j][z])/(deltaX*deltaX)+(Tt_n1_I[i][j-1][z]
      +Tt_n1_I[i][j+1][z]-2.0*Tt_n1_I[i][j][z])/(deltaY*deltaY)+(Tt_n1_I[i][j][z-1]
      +Tt_n1_I[i][j][z+1]-2.0*Tt_n1_I[i][j][z])/(deltaZ*deltaZ)));

      d[i][j][z] = ( p3*C3 + Wb3*Cb3*deltaT +(4.0*k3*deltaT)*(1.0f/(deltaX*deltaX)
      +1.0f/(deltaY*deltaY)) + (2.0*k3*deltaT)/(deltaZ*deltaZ)) * Tt_n1_I[i][j][z]
      - k3*deltaT /(deltaZ*deltaZ) * (Tt_n1_I[i][j][z-1] + Tt_n1_I[i][j][z+1])- omega * f; */

      //radiation_case
      //radiation_case: Dr Dai's method
      f= (  (2.0*p3*C3*(1.0+tau*Wb3*Cb3/(p3*C3)+2.0*tau/deltaT)
      +Wb3*Cb3*deltaT)*Tt_n1_I[i][j][z] + (2.0*p3*C3*(1.0+tau*Wb3*Cb3/(p3*C3)-
      2.0*tau/deltaT)+Wb3*Cb3*deltaT)*Tt[i][j][z] -  4.0*p3*C3*Ut[i][j][z] –
     2.0*Wb3*Cb3*deltaT*(Tb7_n1[0][LX7A]) - k3*deltaT
     *( (Tt_n1_I[i-1][j][z]+Tt_n1_I[i+1][j][z]-
     2*Tt_n1_I[i][j][z])/(deltaX*deltaX)+(Tt_n1_I[i][j-1][z]+Tt_n1_I[i][j+1][z]-
     2*Tt_n1_I[i][j][z])/(deltaY*deltaY)+(Tt_n1_I[i][j][z-1]+Tt_n1_I[i][j][z+1]-
     2*Tt_n1_I[i][j][z])/(deltaZ*deltaZ) ) -k3*deltaT
     *( (Tt[i-1][j][z]+Tt[i+1][j][z]-2*Tt[i][j][z])/(deltaX*deltaX)
     +(Tt[i][j-1][z]+Tt[i][j+1][z]-2*Tt[i][j][z])/(deltaY*deltaY)
     +(Tt[i][j][z-1]+Tt[i][j][z+1]-2*Tt[i][j][z])/(deltaZ*deltaZ));

      d[i][j][z] = ( 2.0*p3*C3*(1.0+tau*Wb3*Cb3/(p3*C3)+2.0*tau/deltaT) + Wb3*Cb3*deltaT
      + (4.0*k3*deltaT)*(1.0f/(deltaX*deltaX)+1.0f/(deltaY*deltaY))
      + (2.0*k3*deltaT)/(deltaZ*deltaZ))* Tt_n1_I[i][j][z]
       - k3*deltaT /(deltaZ*deltaZ) * (Tt_n1_I[i][j][z-1] + Tt_n1_I[i][j][z+1]) - omega * f;
    }//for NZ2 to NZ3
  }//j
}//i

//writeLinearSys(NX/2, NY/2, t, I);
//ajust tissue start here

double tE[NZ3], tF[NZ3];
//solve the tria-diangonal system
for(i=1;i<=NX-1;i++)
  for(j=1;j<=NY-1;j++)
  {
     // Without convection on the surface
      /*tF[1] = d[i][j][1] / ( b[i][j][1] + a[i][j][1] );
      tE[1] = -c[i][j][1] / ( b[i][j][1] + a[i][j][1] ); */
```

```
      // With convection on the surface
      /*tF[1] = (d[i][j][1]-b[i][j][1]*Hf*deltaZ*Tf/(k1+deltaZ*Hf))
       / ( b[i][j][1]*k1/(k1+deltaZ*Hf) + a[i][j][1] );*/
      //tE[1] = -c[i][j][1] / ( b[i][j][1]*k1/(k1+deltaZ*Hf) + a[i][j][1] );

      // With radiation on the surface
      aa = k1 + Hf*deltaZ + sigma*epsilon*deltaZ*((Tf+273.0)*(Tf+273.0)
        + (Tt_n1_I[i][j][0]+273.0)*(Tt_n1_I[i][j][0]+273.0))*((Tf+273.0)
        + (Tt_n1_I[i][j][0]+273.0));

      bb = Hf*deltaZ*Tf + sigma*epsilon*deltaZ*((Tf+273.0)*(Tf+273.0)
        + (Tt_n1_I[i][j][0]+273.0)*(Tt_n1_I[i][j][0]+273.0))*((Tf+273.0)
        + (Tt_n1_I[i][j][0]+273.0))*Tf;

      tF[1] = (d[i][j][1] - b[i][i][1]*bb/aa) / (b[i][j][1]*k1/aa + a[i][j][1]);
      tE[1] = -c[i][j][1] / (b[i][j][1]*k1/aa + a[i][j][1]);

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

   Tt_n1[i][j][NZ3-1] = tF[NZ3-1] / (1-tE[NZ3-1]);

   for(z=NZ3-2;z>=1;z--)
    Tt_n1[i][j][z] = tF[z] + tE[z]*Tt_n1[i][j][z+1];
}

////////////////////////////////////////////
//assign tissue boundary grid points
for(i=0;i<=NX;i++)
  for(j=0;j<=NY;j++)
  {
      // With convection on the surface
      Tt_n1[i][j][0] = (Tt_n1[i][j][1]*k1+Hf*deltaZ*Tf
      +sigma*epsilon*deltaZ*Tf*Tf*Tf*Tf)
       /(k1+deltaZ*Hf+sigma*epsilon*deltaZ*Tt_n1_I[i][j][0]
       *Tt_n1_I[i][j][0]*Tt_n1_I[i][j][0]);
      Tt_n1[i][j][NZ3] = Tt_n1[i][j][NZ3-1];
  }

for(z=0;z<=NZ3;z++)
  for(j=0;j<=NY;j++)
  {
      Tt_n1[0][j][z] = Tt_n1[1][j][z];
      Tt_n1[NX][j][z] = Tt_n1[NX-1][j][z];
  }

for(z=0;z<=NZ3;z++)
  for(i=0;i<=NX;i++)
  {
      Tt_n1[i][0][z] = Tt_n1[i][1][z];
      Tt_n1[i][NY][z] = Tt_n1[i][NY-1][z];
   }
```

```
//ajust tissue start here//////////////////////////
getTv_blood(0);
getTv_blood(1);
//calculate blood temperature based on given vessel temperature
CalcTb();
CalcTb2();
reloadbloodeteperature(0);
reloadbloodeteperature(1);
CalcVessel(0);
CalcVessel(1);

//calculate sum of square error
maxErr1 = 0;

for(i=1;i<NX;i++)
    for(j=1;j<NY;j++)
     for(z=1;z<NZ3;z++)
     {
        temp = fabs(Tt_n1[i][j][z] - Tt_n1_I[i][j][z]);
        if(temp > maxErr1)
        maxErr1 = (double)temp;
     }

     //store result to loop I
     memcpy(Tt_n1_I, Tt_n1, sizeof(double)*(NX+1)*(NY+1)*(NZ3+1));

     return maxErr1;
  }

  void reloadbloodeteperature(int index)
  {
     setboarderVariable(index); //set the common blood boarder variables

     //reassign back the grid points in blood
     //level 7
     k=0;
     for(r=0;r<2;r++)//repeat variable in z coordinate
     {
        z3=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2;
       for(j=0;j<2;j++)//repeat variable in y coordinate
       {
         y3=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2;
         for(p=0;p<2;p++)//repeat variable in x coordinate
         {
           for(q=0;q<2;q++) //repeat variable in inner y coordinate
           {
             for(h=0;h<2;h++)//repeat variable in inner z coordinate
             {
                for(x=0;x<=lx7;x++)
                {
                  x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx7/2+x;
                  x3=x+k*(lx7+1);
                  for(y=1;y<ly7;y++)
                  {
                     y2=y3+(2*q-1)*ly5/2+(2*q-1)*ly6/2-ly7/2+y;
                     for(z=1;z<lz7;z++)
```

```
                    {
                        z2=z3+(2*h-1)*lz6/2+(2*h-1)*lz7/2-lz7/2+z;
                        Tt_n1[x2][y2][z2] = Tb7_n1[index][x3];
                    }
                }
            }
            k++;
        }
    }
  }
}

//sixth level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
{
  z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz6/2;
  for(j=0;j<2;j++)//repeat variable in y coordinate
  {
    y3=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2;
    for(p=0;p<2;p++)//repeat variable in x coordinate
    {
      x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx6/2;
      for(q=0;q<2;q++) //repeat variable in inner y coordinate
      {
        y2=y3+(2*q-1)*ly5/2+(2*q-1)*ly6/2-ly6/2;
        for(z=0;z<=lz6;z++)
        {
          for(y=1;y<ly6;y++)
          {
            for(x=1;x<lx6;x++)
            {
              Tt_n1[x2+x][y2+y][z2+z]= Tb6_n1[index][z+k*(lz6+1)];
                  }
          }
        }
        k++;
      }
    }
  }
}

//fifth level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
{
  z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz5/2;
  for(j=0;j<2;j++)//repeat variable in y coordinate
  {
      y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2-ly5/2;
      for(p=0;p<2;p++)//repeat variable in x coordinate
      {
        x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx5/2;
        for(y=0;y<=ly5;y++)
          for(x=1;x<lx5;x++)
```

```
                for(z=1;z<lz5;z++)
                {
                    Tt_n1[x2+x][y2+y][z2+z] = Tb5_n1[index][y+k*(ly5+1)];
                }
              k++;
            }
        }
    }

    //fourth level
    k=0;
    for(r=0;r<2;r++)
    {
        z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz4/2;
        for(j=0;j<2;j++)
        {
            y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2-ly4/2;
            for(x=0;x<=lx4;x++)
            {
                x2=x4a+x;
                for(y=1;y<ly4;y++)
                  for(z=1;z<lz4;z++)
                  {
                     Tt_n1[x2][y2+y][z2+z]=Tb4_n1[index][x+k*(lx4+1)];
                  }
            }
            k++;
        }
    }

    //third level
    for (k=0;k<2;k++)
    {
        y2=y3a+k*(ly3+ly2);
        for(z=0;z<=lz3;z++)
            for(y=1;y<ly3;y++)
                for(x=1;x<lx3;x++)
                    Tt_n1[x3a+x][y2+y][z3a+z]=Tb3_n1[index][z+k*(lz3+1)];
    }

    //second level
    for( y=0;y<=ly2;y++)
        for(z=1;z<lz2;z++)
            for(x=1;x<lx2;x++)
                    Tt_n1[x2a+x][y2a+y][z2a+z] = Tb2_n1[index][y];

    //first level
    for(x=0;x<=lx1;x++)
        for(y=1;y<ly1;y++)
            for(z=1;z<lz1;z++)
                    Tt_n1[x1a-x][y1a+y][z1a+z] = Tb1_n1[index][x];
    return;
}


void CalcVessel(int index)
```

```
{
  setboarderVariable(index); //set the common blood boarder variables

  ////////////////////////////////////////////
  //calculate blood vessel temperature on sides
  //first level
  for(y=y1a+1;y<y1b;y++)//x-y plane
   for(x=x2b+1;x<=x1a;x++)
   {
         Tt_n1[x][y][z1a] = ( Tt_n1[x][y][z1a-1] + Tb1_n1[index][x1a-x]
               * deltaZ*Bi ) / (1+deltaZ*Bi);
         Tt_n1[x][y][z1b] = ( Tt_n1[x][y][z1b+1] + Tb1_n1[index][x1a-x]
               * deltaZ*Bi ) / (1+deltaZ*Bi);
   }
  for(z=z1a+1;z<z1b;z++)
   for(x=x2b+1;x<=x1a;x++)//x-z plane
   {
         Tt_n1[x][y1a][z] = ( Tt_n1[x][y1a-1][z] + Tb1_n1[index][x1a-x]
               * deltaY*Bi ) / (1+deltaY*Bi);
         Tt_n1[x][y1b][z] = ( Tt_n1[x][y1b+1][z] + Tb1_n1[index][x1a-x]
               * deltaY*Bi ) / (1+deltaY*Bi);
   }
  for(z=z1a+1;z<z2a;z++)//left most sides
  {
   for(y=y1a+1;y<y1b;y++)
         Tt_n1[x2b][y][z] = ( Tt_n1[x2b-1][y][z] + Tb1_n1[index][lx1-1]
               * deltaX*Bi ) / (1+deltaX*Bi);
   //side edges
   Tt_n1[x2b][y1a][z] = ( Tt_n1[x2b+1][y1a][z] + Tt_n1[x2b][y1a+1][z] )/2;
                Tt_n1[x2b][y1b][z] = ( Tt_n1[x2b+1][y1b][z]
        + Tt_n1[x2b][y1b-1][z] )/2;
  }
  for(z=z2b+1;z<z1b;z++)
  {
   for(y=y1a+1;y<y1b;y++)
         Tt_n1[x2b][y][z] = ( Tt_n1[x2b-1][y][z] + Tb1_n1[index][lx1-1]
               * deltaX*Bi ) / (1+deltaX*Bi);
   //side edges
   Tt_n1[x2b][y1a][z] = ( Tt_n1[x2b+1][y1a][z] + Tt_n1[x2b][y1a+1][z] )/2;
   Tt_n1[x2b][y1b][z] = ( Tt_n1[x2b+1][y1b][z] + Tt_n1[x2b][y1b-1][z] )/2;
  }
  for(x=x2b+1;x<=x1a;x++)
  {
   Tt_n1[x][y1a][z1a] = ( Tt_n1[x][y1a+1][z1a] + Tt_n1[x][y1a][z1a+1] )/2;
   Tt_n1[x][y1a][z1b] = ( Tt_n1[x][y1a+1][z1b] + Tt_n1[x][y1a][z1b-1] )/2;
   Tt_n1[x][y1b][z1a] = ( Tt_n1[x][y1b-1][z1a] + Tt_n1[x][y1b][z1a+1] )/2;
   Tt_n1[x][y1b][z1b] = ( Tt_n1[x][y1b-1][z1b] + Tt_n1[x][y1b][z1b-1] )/2;
   //save blood boarder temperature
   Tbd1[index][0][x1a-x] = Tt_n1[x][y1a][z1a];
   Tbd1[index][1][x1a-x] = Tt_n1[x][y1b][z1a];
   Tbd1[index][2][x1a-x] = Tt_n1[x][y1a][z1b];
   Tbd1[index][3][x1a-x] = Tt_n1[x][y1b][z1b];
  }
  //vertix
  Tt_n1[x2b][y1a][z1a] = ( Tt_n1[x2b+1][y1a][z1a] + Tt_n1[x2b][y1a+1][z1a]
                + Tt_n1[x2b][y1a][z1a+1] )/3;
```

```
Tt_n1[x2b][y1a][z1b] = ( Tt_n1[x2b+1][y1a][z1b] + Tt_n1[x2b][y1a+1][z1b]
                + Tt_n1[x2b][y1a][z1b-1] )/3;
Tt_n1[x2b][y1b][z1a] = ( Tt_n1[x2b+1][y1b][z1a] + Tt_n1[x2b][y1b-1][z1a]
                + Tt_n1[x2b][y1b][z1a+1] )/3;
Tt_n1[x2b][y1b][z1b] = ( Tt_n1[x2b+1][y1b][z1b] + Tt_n1[x2b][y1b-1][z1b]
                + Tt_n1[x2b][y1b][z1b-1] )/3;
 //save blood boarder temperature
Tbd1[index][0][x1a-x2b] = Tt_n1[x2b][y1a][z1a];
Tbd1[index][1][x1a-x2b] = Tt_n1[x2b][y1b][z1a];
Tbd1[index][2][x1a-x2b] = Tt_n1[x2b][y1a][z1b];
Tbd1[index][3][x1a-x2b] = Tt_n1[x2b][y1b][z1b];

//level 2
for(y=y2a+1;y<y2b;y++)//x-y plane
 for(x=x2a+1;x<x2b;x++)
 {
        Tt_n1[x][y][z2a] = ( Tt_n1[x][y][z2a-1] + Tb2_n1[index][y-y2a]
                * deltaZ*Bi ) / (1+deltaZ*Bi);
        Tt_n1[x][y][z2b] = ( Tt_n1[x][y][z2b+1] + Tb2_n1[index][y-y2a]
                * deltaZ*Bi ) / (1+deltaZ*Bi);
 }
for(z=z2a+1;z<z2b;z++)
{ //y-z plane
 for(y=y2a+1;y<y2b;y++)
        Tt_n1[x2a][y][z] = ( Tt_n1[x2a-1][y][z] + Tb2_n1[index][y-y2a]
                * deltaX*Bi ) / (1+deltaX*Bi);
 for(y=y2a+1;y<y1a;y++)
        Tt_n1[x2b][y][z] = ( Tt_n1[x2b+1][y][z] + Tb2_n1[index][y-y2a]
                * deltaX*Bi ) / (1+deltaX*Bi);
 for(y=y1b+1;y<y2b;y++)
        Tt_n1[x2b][y][z] = ( Tt_n1[x2b+1][y][z] + Tb2_n1[index][y-y2a]
                * deltaX*Bi ) / (1+deltaX*Bi);
 //x-z plane
 for(x=x2a+1;x<x3a;x++)
 {
  Tt_n1[x][y2a][z] = ( Tt_n1[x][y2a-1][z] + Tb2_n1[index][1]
                * deltaY*Bi ) / (1+deltaY*Bi);
        Tt_n1[x][y2b][z] = ( Tt_n1[x][y2b+1][z] + Tb2_n1[index][ly2-1]
                * deltaY*Bi ) / (1+deltaY*Bi);
 }
 for(x=x3b+1;x<x2b;x++)
 {
  Tt_n1[x][y2a][z] = ( Tt_n1[x][y2a-1][z] + Tb2_n1[index][1]
                * deltaY*Bi ) / (1+deltaY*Bi);
  Tt_n1[x][y2b][z] = ( Tt_n1[x][y2b+1][z] + Tb2_n1[index][ly2-1]
                * deltaY*Bi ) / (1+deltaY*Bi);
 }
 //vertical edges
 Tt_n1[x2a][y2a][z] = ( Tt_n1[x2a+1][y2a][z] + Tt_n1[x2a][y2a+1][z] )/2;
 Tt_n1[x2a][y2b][z] = ( Tt_n1[x2a+1][y2b][z] + Tt_n1[x2a][y2b-1][z] )/2;
 Tt_n1[x2b][y2a][z] = ( Tt_n1[x2b-1][y2a][z] + Tt_n1[x2b][y2a+1][z] )/2;
 Tt_n1[x2b][y2b][z] = ( Tt_n1[x2b-1][y2b][z] + Tt_n1[x2b][y2b-1][z] )/2;
 //vertical boarder edges of level 2 and 1
 Tt_n1[x2b][y1a][z] = ( Tt_n1[x2b+1][y1a][z] + Tt_n1[x2b][y1a-1][z] )/2;
 Tt_n1[x2b][y1b][z] = ( Tt_n1[x2b+1][y1b][z] + Tt_n1[x2b][y1b+1][z] )/2;
}
```

```
//horizontal y direction side edges
for(y=y2a+1;y<y2b;y++)
{
 Tt_n1[x2a][y][z2a] = ( Tt_n1[x2a+1][y][z2a] + Tt_n1[x2a][y][z2a+1] )/2;
 Tt_n1[x2a][y][z2b] = ( Tt_n1[x2a+1][y][z2b] + Tt_n1[x2a][y][z2b-1] )/2;
 //save blood boarder temperature y2a+1 to y2b-1
 Tbd2[index][0][y-y2a] = Tt_n1[x2a][y][z2a];
 Tbd2[index][2][y-y2a] = Tt_n1[x2a][y][z2b];
}
for(y=y2a+1;y<y1a;y++)
{
 Tt_n1[x2b][y][z2a] = ( Tt_n1[x2b-1][y][z2a] + Tt_n1[x2b][y][z2a+1] )/2;
 Tt_n1[x2b][y][z2b] = ( Tt_n1[x2b-1][y][z2b] + Tt_n1[x2b][y][z2b-1] )/2;
 //save blood boarder temperature y2a+1 to y1a-1
 Tbd2[index][1][y-y2a] = Tt_n1[x2b][y][z2a];
 Tbd2[index][3][y-y2a] = Tt_n1[x2b][y][z2b];
}
for(y=y1b+1;y<y2b;y++)
{
 Tt_n1[x2b][y][z2a] = ( Tt_n1[x2b-1][y][z2a] + Tt_n1[x2b][y][z2a+1] )/2;
 Tt_n1[x2b][y][z2b] = ( Tt_n1[x2b-1][y][z2b] + Tt_n1[x2b][y][z2b-1] )/2;
 //save blood boarder temperature y1b+1 to y2b-1
 Tbd2[index][1][y-y2a] = Tt_n1[x2b][y][z2a];
 Tbd2[index][3][y-y2a] = Tt_n1[x2b][y][z2b];
}
//horizontal x direction edges
for(x=x2a+1;x<x3a;x++)
{
 Tt_n1[x][y2a][z2a] = ( Tt_n1[x][y2a+1][z2a] + Tt_n1[x][y2a][z2a+1] )/2;
 Tt_n1[x][y2a][z2b] = ( Tt_n1[x][y2a+1][z2b] + Tt_n1[x][y2a][z2b-1] )/2;
 Tt_n1[x][y2b][z2a] = ( Tt_n1[x][y2b-1][z2a] + Tt_n1[x][y2b][z2a+1] )/2;
 Tt_n1[x][y2b][z2b] = ( Tt_n1[x][y2b-1][z2b] + Tt_n1[x][y2b][z2b-1] )/2;
 }
for(x=x3b+1;x<x2b;x++)
{
        Tt_n1[x][y2a][z2a] = ( Tt_n1[x][y2a+1][z2a] + Tt_n1[x][y2a][z2a+1] )/2;
        Tt_n1[x][y2a][z2b] = ( Tt_n1[x][y2a+1][z2b] + Tt_n1[x][y2a][z2b-1] )/2;
        Tt_n1[x][y2b][z2a] = ( Tt_n1[x][y2b-1][z2a] + Tt_n1[x][y2b][z2a+1] )/2;
        Tt_n1[x][y2b][z2b] = ( Tt_n1[x][y2b-1][z2b] + Tt_n1[x][y2b][z2b-1] )/2;
}
//vertix
Tt_n1[x2a][y2a][z2a] = ( Tt_n1[x2a+1][y2a][z2a] + Tt_n1[x2a][y2a+1][z2a]
                + Tt_n1[x2a][y2a][z2a+1] )/3;
Tt_n1[x2a][y2a][z2b] = ( Tt_n1[x2a+1][y2a][z2b] + Tt_n1[x2a][y2a+1][z2b]
                + Tt_n1[x2a][y2a][z2b-1] )/3;
Tt_n1[x2a][y2b][z2a] = ( Tt_n1[x2a+1][y2b][z2a] + Tt_n1[x2a][y2b-1][z2a]
                + Tt_n1[x2a][y2b][z2a+1] )/3;
Tt_n1[x2a][y2b][z2b] = ( Tt_n1[x2a+1][y2b][z2b] + Tt_n1[x2a][y2b-1][z2b]
                + Tt_n1[x2a][y2b][z2b-1] )/3;
Tt_n1[x2b][y2a][z2a] = ( Tt_n1[x2b-1][y2a][z2a] + Tt_n1[x2b][y2a+1][z2a]
                + Tt_n1[x2b][y2a][z2a+1] )/3;
Tt_n1[x2b][y2a][z2b] = ( Tt_n1[x2b-1][y2a][z2b] + Tt_n1[x2b][y2a+1][z2b]
                + Tt_n1[x2b][y2a][z2b-1] )/3;
Tt_n1[x2b][y2b][z2a] = ( Tt_n1[x2b-1][y2b][z2a] + Tt_n1[x2b][y2b-1][z2a]
                + Tt_n1[x2b][y2b][z2a+1] )/3;
Tt_n1[x2b][y2b][z2b] = ( Tt_n1[x2b-1][y2b][z2b] + Tt_n1[x2b][y2b-1][z2b]
```

```
                + Tt_n1[x2b][y2b][z2b-1] )/3;
//save blood boarder temperature y2a
Tbd2[index][0][0] = Tt_n1[x2a][y2a][z2a];
Tbd2[index][1][0] = Tt_n1[x2b][y2a][z2a];
Tbd2[index][2][0] = Tt_n1[x2a][y2a][z2b];
Tbd2[index][3][0] = Tt_n1[x2b][y2a][z2b];
//save blood boarder temperature y2b
Tbd2[index][0][ly2] = Tt_n1[x2a][y2b][z2a];
Tbd2[index][1][ly2] = Tt_n1[x2b][y2b][z2a];
Tbd2[index][2][ly2] = Tt_n1[x2a][y2b][z2b];
Tbd2[index][3][ly2] = Tt_n1[x2b][y2b][z2b];

//horizontal boarder edges of level 2 and 1
for(y=y1a+1;y<y1b;y++)
{
  Tt_n1[x2b][y][z2a] = ( Tt_n1[x2b-1][y][z2a] + Tt_n1[x2b][y][z2a-1] )/2;
  Tt_n1[x2b][y][z2b] = ( Tt_n1[x2b-1][y][z2b] + Tt_n1[x2b][y][z2b+1] )/2;
  //save blood boarder temperature y1a+1 to y1b-1
  Tbd2[index][1][y-y2a] = Tt_n1[x2b][y][z2a];
  Tbd2[index][3][y-y2a] = Tt_n1[x2b][y][z2b];

}
//vertix of boarder edges of level 2 and 1
Tt_n1[x2b][y1a][z2a] = ( Tt_n1[x2b][y1a-1][z2a] + Tt_n1[x2b][y1a+1][z2a]
            + Tt_n1[x2b][y1a][z2a-1] + Tt_n1[x2b][y1a][z2a+1]  )/4;
Tt_n1[x2b][y1a][z2b] = ( Tt_n1[x2b][y1a-1][z2b] + Tt_n1[x2b][y1a+1][z2b]
            + Tt_n1[x2b][y1a][z2b-1] + Tt_n1[x2b][y1a][z2b+1]  )/4;
Tt_n1[x2b][y1b][z2a] = ( Tt_n1[x2b][y1b-1][z2a] + Tt_n1[x2b][y1b+1][z2a]
            + Tt_n1[x2b][y1b][z2a-1] + Tt_n1[x2b][y1b][z2a+1]  )/4;
 Tt_n1[x2b][y1b][z2b] = ( Tt_n1[x2b][y1b-1][z2b] + Tt_n1[x2b][y1b+1][z2b]
            + Tt_n1[x2b][y1b][z2b-1] + Tt_n1[x2b][y1b][z2b+1]  )/4;
 //save blood boarder temperature y1a
 Tbd2[index][1][y1a-y2a] = Tt_n1[x2b][y1a][z2a];
 Tbd2[index][3][y1a-y2a] = Tt_n1[x2b][y1a][z2b];
  //save blood boarder temperature y1b
 Tbd2[index][1][y1b-y2a] = Tt_n1[x2b][y1b][z2a];
 Tbd2[index][3][y1b-y2a] = Tt_n1[x2b][y1b][z2b];

 //level 3
 for (j=0;j<2;j++)
 {
   y7=y3a+j*(ly3+ly2);
   y2=y2a+j*(ly3+ly2);
   y3=y2a+j*ly2;
   y4=y3a+j*(2*ly3+ly2);
   for(z=z3a+1;z<z3b;z++)//y-z plane
     for(y=y7+1;y<y2;y++)
     {
       Tt_n1[x3a][y][z] = ( Tt_n1[x3a-1][y][z] + Tb3_n1[index][z-z3a+j*(lz3+1)]
            * deltaX*Bi ) / (1+deltaX*Bi);
       Tt_n1[x3b][y][z] = ( Tt_n1[x3b+1][y][z] + Tb3_n1[index][z-z3a+j*(lz3+1)]
            * deltaX*Bi ) / (1+deltaX*Bi);
     }
   for(z=z3a;z<=z3b;z++)//x-z plane
     for(x=x3a+1;x<x3b;x++)
       Tt_n1[x][y4][z] = ( Tt_n1[x][y4+2*j-1][z] + Tb3_n1[index][z-z3a+j*(lz3+1)]
```

```
                * deltaY*Bi ) / (1+deltaY*Bi);
    for(z=z3a;z<z2a;z++)//x-z plane
      for(x=x3a+1;x<x3b;x++)
        Tt_n1[x][y3][z] = ( Tt_n1[x][y3+1-2*j][z] + Tb3_n1[index][z-z3a+j*(lz3+1)]
              * deltaY*Bi ) / (1+deltaY*Bi);
    for(z=z2b+1;z<=z3b;z++)//x-z plane
      for(x=x3a+1;x<x3b;x++)
        Tt_n1[x][y3][z] = ( Tt_n1[x][y3+1-2*j][z] + Tb3_n1[index][z-z3a+j*(lz3+1)]
              * deltaY*Bi ) / (1+deltaY*Bi);
    //vertical edges
    for(z=z3a+1;z<z3b;z++)
    {
        Tt_n1[x3a][y4][z] = ( Tt_n1[x3a+1][y4][z] + Tt_n1[x3a][y4+1-2*j][z] )/2;
        Tt_n1[x3b][y4][z] = ( Tt_n1[x3b-1][y4][z] + Tt_n1[x3b][y4+1-2*j][z] )/2;
        if(j==0)
        {  //save blood boarder temperature z3a+1 to z3b-1
           Tbd3[index][0][z-z3a+j*(lz3+1)] = Tt_n1[x3a][y4][z];
           Tbd3[index][1][z-z3a+j*(lz3+1)] = Tt_n1[x3b][y4][z];
        }
        else
        { //save blood boarder temperature z3a+1 to z3b-1
           Tbd3[index][2][z-z3a+j*(lz3+1)] = Tt_n1[x3a][y4][z];
          Tbd3[index][3][z-z3a+j*(lz3+1)] = Tt_n1[x3b][y4][z];
        }
}
for(z=z3a+1;z<z2a;z++)
{
  Tt_n1[x3a][y3][z] = ( Tt_n1[x3a+1][y3][z] + Tt_n1[x3a][y3+2*j-1][z] )/2;
  Tt_n1[x3b][y3][z] = ( Tt_n1[x3b-1][y3][z] + Tt_n1[x3b][y3+2*j-1][z] )/2;
   if(j==0)
   { //save blood boarder temperature z3a+1 to z2a-1
       Tbd3[index][2][z-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z];
       Tbd3[index][3][z-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z];
   }
   else
   { //save blood boarder temperature z3a+1 to z2a-1
       Tbd3[index][0][z-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z];
       Tbd3[index][1][z-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z];
   }
}
for(z=z2b+1;z<z3b;z++)
{
   Tt_n1[x3a][y3][z] = ( Tt_n1[x3a+1][y3][z] + Tt_n1[x3a][y3+2*j-1][z] )/2;
   Tt_n1[x3b][y3][z] = ( Tt_n1[x3b-1][y3][z] + Tt_n1[x3b][y3+2*j-1][z] )/2;
   if(j==0)
   {    //save blood boarder temperature z3a+1 to z2a-1
        Tbd3[index][2][z-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z];
        Tbd3[index][3][z-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z];
   }
   else
   {    //save blood boarder temperature z3a+1 to z2a-1
        Tbd3[index][0][z-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z];
       Tbd3[index][1][z-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z];
   }
}
//vertex
```

```
Tt_n1[x3a][y7][z3a] = ( Tt_n1[x3a+1][y7][z3a] + Tt_n1[x3a][y7][z3a+1] )/2;
Tt_n1[x3a][y7][z3b] = ( Tt_n1[x3a+1][y7][z3b] + Tt_n1[x3a][y7][z3b-1] )/2;
Tt_n1[x3a][y2][z3a] = ( Tt_n1[x3a+1][y2][z3a] + Tt_n1[x3a][y2][z3a+1] )/2;
Tt_n1[x3a][y2][z3b] = ( Tt_n1[x3a+1][y2][z3b] + Tt_n1[x3a][y2][z3b-1] )/2;
Tt_n1[x3b][y7][z3a] = ( Tt_n1[x3b-1][y7][z3a] + Tt_n1[x3b][y7][z3a+1] )/2;
Tt_n1[x3b][y7][z3b] = ( Tt_n1[x3b-1][y7][z3b] + Tt_n1[x3b][y7][z3b-1] )/2;
Tt_n1[x3b][y2][z3a] = ( Tt_n1[x3b-1][y2][z3a] + Tt_n1[x3b][y2][z3a+1] )/2;
Tt_n1[x3b][y2][z3b] = ( Tt_n1[x3b-1][y2][z3b] + Tt_n1[x3b][y2][z3b-1] )/2;
 //save blood boarder temperature z3a
Tbd3[index][0][0+j*(lz3+1)] = Tt_n1[x3a][y7][z3a];
Tbd3[index][1][0+j*(lz3+1)] = Tt_n1[x3b][y7][z3a];
Tbd3[index][2][0+j*(lz3+1)] = Tt_n1[x3a][y2][z3a];
Tbd3[index][3][0+j*(lz3+1)] = Tt_n1[x3b][y2][z3a];
 //save blood boarder temperature z3b
 Tbd3[index][0][lz3+j*(lz3+1)] = Tt_n1[x3a][y7][z3b];
 Tbd3[index][1][lz3+j*(lz3+1)] = Tt_n1[x3b][y7][z3b];
  Tbd3[index][2][lz3+j*(lz3+1)] = Tt_n1[x3a][y2][z3b];
  Tbd3[index][3][lz3+j*(lz3+1)] = Tt_n1[x3b][y2][z3b];
  //vertical boarder edges of level 2 and 3
  for(z=z2a+1;z<z2b;z++)
  {
     Tt_n1[x3a][y3][z] = ( Tt_n1[x3a-1][y3][z] + Tt_n1[x3a][y3+2*j-1][z] )/2;
     Tt_n1[x3b][y3][z] = ( Tt_n1[x3b+1][y3][z] + Tt_n1[x3b][y3+2*j-1][z] )/2;
      if(j==0)
      {    //save blood boarder temperature z2a+1 to z2b-1
           Tbd3[index][2][z-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z];
           Tbd3[index][3][z-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z];
      }
      else
      {
           Tbd3[index][0][z-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z];
           Tbd3[index][1][z-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z];
      }
  }
  //horizontal boarder edges of level 2 and 3
 for(x=x3a+1;x<x3b;x++)
 {
        Tt_n1[x][y3][z2a] = ( Tt_n1[x][y3+1-2*j][z2a] + Tt_n1[x3a][y3][z2a-1] )/2;
        Tt_n1[x][y3][z2b] = ( Tt_n1[x][y3+1-2*j][z2b] + Tt_n1[x3b][y3][z2b+1] )/2;
 }
 //vertix of boarder edges of level 2 and 3
Tt_n1[x3a][y3][z2a] = ( Tt_n1[x3a-1][y3][z2a] + Tt_n1[x3a+1][y3][z2a]
            + Tt_n1[x3a][y3][z2a-1] + Tt_n1[x3a][y3][z2a+1]  )/4;
Tt_n1[x3a][y3][z2b] = ( Tt_n1[x3a-1][y3][z2b] + Tt_n1[x3a+1][y3][z2b]
            + Tt_n1[x3a][y3][z2b-1] + Tt_n1[x3a][y3][z2b+1]  )/4;
Tt_n1[x3b][y3][z2a] = ( Tt_n1[x3b-1][y3][z2a] + Tt_n1[x3b+1][y3][z2a]
            + Tt_n1[x3b][y3][z2a-1] + Tt_n1[x3b][y3][z2a+1]  )/4;
Tt_n1[x3b][y3][z2b] = ( Tt_n1[x3b-1][y3][z2b] + Tt_n1[x3b+1][y3][z2b]
            + Tt_n1[x3b][y3][z2b-1] + Tt_n1[x3b][y3][z2b+1]  )/4;
 if(j==0)
{     //save blood boarder temperature z2a
       Tbd3[index][2][z2a-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z2a];
       Tbd3[index][3][z2a-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z2a];
       //save blood boarder temperature z2b
       Tbd3[index][2][z2b-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z2b];
       Tbd3[index][3][z2b-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z2b];
```

```
    }
    else
    {     //save blood boarder temperature z2a
          Tbd3[index][0][z2a-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z2a];
          Tbd3[index][1][z2a-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z2a];
          //save blood boarder temperature z2b
          Tbd3[index][0][z2b-z3a+j*(lz3+1)] = Tt_n1[x3a][y3][z2b];
          Tbd3[index][1][z2b-z3a+j*(lz3+1)] = Tt_n1[x3b][y3][z2b];
    }
}

//fourth level
k=0;
for(r=0;r<2;r++)
  for(j=0;j<2;j++)
  {
    z1=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz4/2;
    z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+lz4/2;
         z3=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz5/2;
    z4=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+lz5/2;
         z5=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4;
         z6=cenZ+(2*r-1)*lz3/2;
         y7=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2-ly4/2;
    y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+ly4/2;
    for(y=y7+1;y<y2;y++)//x-y plane
      for(x=x4a+1;x<x4b;x++)
         Tt_n1[x][y][z5] = ( Tt_n1[x][y][z5+2*r-1]
         + Tb4_n1[index][x-x4a+k*(lx4+1)] * deltaZ*Bi ) / (1+deltaZ*Bi);
    for(y=y7+1;y<y2;y++)//x-y plane
      for(x=x4a+1;x<x3a;x++)
         Tt_n1[x][y][z6] = ( Tt_n1[x][y][z6+1-2*r]
         + Tb4_n1[index][x-x4a+k*(lx4+1)] * deltaZ*Bi ) / (1+deltaZ*Bi);
    for(y=y7+1;y<y2;y++)//x-y plane
      for(x=x3b+1;x<x4b;x++)
         Tt_n1[x][y][z6] = ( Tt_n1[x][y][z6+1-2*r]
         + Tb4_n1[index][x-x4a+k*(lx4+1)] * deltaZ*Bi ) / (1+deltaZ*Bi);
    for(z=z1+1;z<z2;z++)
      for(x=x4a+1;x<x4b;x++)//x-z plane
      {
        Tt_n1[x][y7][z] = ( Tt_n1[x][y7-1][z]
        + Tb4_n1[index][x-x4a+k*(lx4+1)] * deltaY*Bi ) / (1+deltaY*Bi);
        Tt_n1[x][y2][z] = ( Tt_n1[x][y2+1][z]
        + Tb4_n1[index][x-x4a+k*(lx4+1)] * deltaY*Bi ) / (1+deltaY*Bi);
      }
    for(z=z1+1;z<z3;z++)//y-z plane
      for(y=y7+1;y<y2;y++)
      {
        Tt_n1[x4a][y][z] = ( Tt_n1[x4a-1][y][z]
        + Tb4_n1[index][1+k*(lx4+1)] * deltaX*Bi ) / (1+deltaX*Bi);
        Tt_n1[x4b][y][z] = ( Tt_n1[x4b+1][y][z]
        + Tb4_n1[index][lx4-1+k*(lx4+1)] * deltaX*Bi ) / (1+deltaX*Bi);
         //vertical edges
         Tt_n1[x4a][y7][z] = ( Tt_n1[x4a+1][y7][z] + Tt_n1[x4a][y7+1][z] )/2;
         Tt_n1[x4a][y2][z] = ( Tt_n1[x4a+1][y2][z] + Tt_n1[x4a][y2-1][z] )/2;
         Tt_n1[x4b][y7][z] = ( Tt_n1[x4b-1][y7][z] + Tt_n1[x4b][y7+1][z] )/2;
         Tt_n1[x4b][y2][z] = ( Tt_n1[x4b-1][y2][z] + Tt_n1[x4b][y2-1][z] )/2;
```

```
    }
 for(z=z4+1;z<z2;z++)//y-z plane
    for(y=y7+1;y<y2;y++)
    {
       Tt_n1[x4a][y][z] = ( Tt_n1[x4a-1][y][z]
       + Tb4_n1[index][1+k*(lx4+1)] * deltaX*Bi ) / (1+deltaX*Bi);
       Tt_n1[x4b][y][z] = ( Tt_n1[x4b+1][y][z]
       + Tb4_n1[index][lx4-1+k*(lx4+1)] * deltaX*Bi ) / (1+deltaX*Bi);
       //vertical edges
       Tt_n1[x4a][y7][z] = ( Tt_n1[x4a+1][y7][z] + Tt_n1[x4a][y7+1][z] )/2;
       Tt_n1[x4a][y2][z] = ( Tt_n1[x4a+1][y2][z] + Tt_n1[x4a][y2-1][z] )/2;
       Tt_n1[x4b][y7][z] = ( Tt_n1[x4b-1][y7][z] + Tt_n1[x4b][y7+1][z] )/2;
       Tt_n1[x4b][y2][z] = ( Tt_n1[x4b-1][y2][z] + Tt_n1[x4b][y2-1][z] )/2;
}
//horizontal y direction edges
for(y=y7+1;y<y2;y++)
{
      Tt_n1[x4a][y][z1] = ( Tt_n1[x4a+1][y][z1] + Tt_n1[x4a][y][z1+1] )/2;
       Tt_n1[x4a][y][z2] = ( Tt_n1[x4a+1][y][z2] + Tt_n1[x4a][y][z2-1] )/2;
       Tt_n1[x4b][y][z1] = ( Tt_n1[x4b-1][y][z1] + Tt_n1[x4a][y][z1+1] )/2;
       Tt_n1[x4b][y][z2] = ( Tt_n1[x4b-1][y][z2] + Tt_n1[x4a][y][z2-1] )/2;
}
//horizontal x direction edges
for(x=x4a+1;x<x4b;x++)
{
      Tt_n1[x][y7][z5] = ( Tt_n1[x][y7+1][z5] + Tt_n1[x][y7][z5+1-2*r] )/2;
       Tt_n1[x][y2][z5] = ( Tt_n1[x][y2-1][z5] + Tt_n1[x][y2][z5+1-2*r] )/2;
       if(r==0)
       {   //save blood boarder temperature x4a+1 to x4b-1
          Tbd4[index][0][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z5];
          Tbd4[index][1][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z5];
       }
       else
       {   //save blood boarder temperature x4a+1 to x4b-1
           Tbd4[index][2][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z5];
           Tbd4[index][3][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z5];
       }
}
for(x=x4a+1;x<x3a;x++)
{
     Tt_n1[x][y7][z6] = ( Tt_n1[x][y7+1][z6] + Tt_n1[x][y7][z6+2*r-1] )/2;
     Tt_n1[x][y2][z6] = ( Tt_n1[x][y2-1][z6] + Tt_n1[x][y2][z6+2*r-1] )/2;
       if(r==0)
       {   //save blood boarder temperature x4a+1 to x3a-1
          Tbd4[index][2][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z6];
          Tbd4[index][3][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z6];
       }
       else
       {
          //save blood boarder temperature x4a+1 to x3a-1
          Tbd4[index][0][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z6];
          Tbd4[index][1][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z6];
       }
}
 for(x=x3b+1;x<x4b;x++)
 {
```

```
        Tt_n1[x][y7][z6] = ( Tt_n1[x][y7+1][z6] + Tt_n1[x][y7][z6+2*r-1] )/2;
        Tt_n1[x][y2][z6] = ( Tt_n1[x][y2-1][z6] + Tt_n1[x][y2][z6+2*r-1] )/2;
        if(r==0)
        {    //save blood boarder temperature x3b+1 to x4b-1
             Tbd4[index][2][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z6];
             Tbd4[index][3][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z6];
        }
        else
        {    //save blood boarder temperature x3b+1 to x4b-1
             Tbd4[index][0][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z6];
             Tbd4[index][1][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z6];
        }
 }
 //vertix
Tt_n1[x4a][y7][z1] = ( Tt_n1[x4a+1][y7][z1] + Tt_n1[x4a][y7+1][z1] +
                Tt_n1[x4a][y7][z1+1] )/3;
Tt_n1[x4a][y7][z2] = ( Tt_n1[x4a+1][y7][z2] + Tt_n1[x4a][y7+1][z2] +
                Tt_n1[x4a][y7][z2-1] )/3;
Tt_n1[x4a][y2][z1] = ( Tt_n1[x4a+1][y2][z1] + Tt_n1[x4a][y2-1][z1] +
                Tt_n1[x4a][y2][z1+1] )/3;
Tt_n1[x4a][y2][z2] = ( Tt_n1[x4a+1][y2][z2] + Tt_n1[x4a][y2-1][z2] +
                Tt_n1[x4a][y2][z2-1] )/3;
     Tt_n1[x4b][y7][z1] = ( Tt_n1[x4b-1][y7][z1] + Tt_n1[x4b][y7+1][z1] +
                Tt_n1[x4b][y7][z1+1] )/3;
Tt_n1[x4b][y7][z2] = ( Tt_n1[x4b-1][y7][z2] + Tt_n1[x4b][y7+1][z2] +
                Tt_n1[x4b][y7][z2-1] )/3;
Tt_n1[x4b][y2][z1] = ( Tt_n1[x4b-1][y2][z1] + Tt_n1[x4b][y2-1][z1] +
                Tt_n1[x4b][y2][z1+1] )/3;
Tt_n1[x4b][y2][z2] = ( Tt_n1[x4b-1][y2][z2] + Tt_n1[x4b][y2-1][z2] +
                Tt_n1[x4b][y2][z2-1] )/3;
//save blood boarder temperature x4a
Tbd4[index][0][0+k*(lx4+1)] = Tt_n1[x4a][y7][z1];
Tbd4[index][1][0+k*(lx4+1)] = Tt_n1[x4a][y2][z1];
Tbd4[index][2][0+k*(lx4+1)] = Tt_n1[x4a][y7][z2];
Tbd4[index][3][0+k*(lx4+1)] = Tt_n1[x4a][y2][z2];
Tbd4[index][0][lx4+k*(lx4+1)] = Tt_n1[x4a][y7][z1];
 //save blood boarder temperature x4b
Tbd4[index][1][lx4+k*(lx4+1)] = Tt_n1[x4a][y2][z1];
Tbd4[index][2][lx4+k*(lx4+1)] = Tt_n1[x4a][y7][z2];
Tbd4[index][3][lx4+k*(lx4+1)] = Tt_n1[x4a][y2][z2];
 //horizontal x direction boarder edges of level 4 and 3
 for(x=x3a+1;x<x3b;x++)
{
     Tt_n1[x][y7][z6] = ( Tt_n1[x][y7-1][z6] + Tt_n1[x][y7][z6+2*r-1] )/2;
     Tt_n1[x][y2][z6] = ( Tt_n1[x][y2+1][z6] + Tt_n1[x][y2][z6+2*r-1] )/2;
     if(r==0)
     {    //save blood boarder temperature x3a+1 to x3b-1
          Tbd4[index][2][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z6];
          Tbd4[index][3][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z6];
      }
      else
      {  //save blood boarder temperature x3a+1 to x3b-1
         Tbd4[index][0][x-x4a+k*(lx4+1)] = Tt_n1[x][y7][z6];
         Tbd4[index][1][x-x4a+k*(lx4+1)] = Tt_n1[x][y2][z6];
      }
     }
```

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

//fifth level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
  for(j=0;j<2;j++)//repeat variable in y coordinate
    for(p=0;p<2;p++)//repeat variable in x coordinate
    {
        x1=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx5/2;
        x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2+lx5/2;
        x3=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5;
        x4=cenX+(2*p-1)*lx4/2;
        y7=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2-ly5/2;
        y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+ly5/2;
        z1=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz5/2;
        z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+lz5/2;
        y3=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2-ly4/2;
        y4=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+ly4/2;

        for(x=x1+1;x<x2;x++)//x-y plane
          for(y=y7+1;y<y2;y++)
          {
              Tt_n1[x][y][z1] = ( Tt_n1[x][y][z1-1] + Tb5_n1[index][y-y7
              +k*(ly5+1)] * deltaZ*Bi ) / (1+deltaZ*Bi);
```

```
      Tt_n1[x][y][z2] = ( Tt_n1[x][y][z2+1] + Tb5_n1[index][y-y7
      +k*(ly5+1)] * deltaZ*Bi ) / (1+deltaZ*Bi);
   }

for(z=z1+1;z<z2;z++)//y-z plaine
{
  for(y=y7;y<=y2;y++)
      Tt_n1[x3][y][z] = ( Tt_n1[x3+2*p-1][y][z] + Tb5_n1[index][y-y7
      +k*(ly5+1)] * deltaX*Bi ) / (1+deltaX*Bi);
  for(y=y7;y<y3;y++)
      Tt_n1[x4][y][z] = ( Tt_n1[x4+1-2*p][y][z] + Tb5_n1[index][y-y7
      +k*(ly5+1)] * deltaX*Bi ) / (1+deltaX*Bi);
  for(y=y4+1;y<=y2;y++)
      Tt_n1[x4][y][z] = ( Tt_n1[x4+1-2*p][y][z] + Tb5_n1[index][y-y7
      +k*(ly5+1)] * deltaX*Bi ) / (1+deltaX*Bi);
   //vertical boarder edges of level 4 and 5
   Tt_n1[x4][y3][z] = ( Tt_n1[x4+1-2*p][y3][z] + Tt_n1[x4][y3-1][z] )/2;
   Tt_n1[x4][y4][z] = ( Tt_n1[x4+1-2*p][y4][z] + Tt_n1[x4][y4+1][z] )/2;
}
//horizontal y direction edges
for(y=y7+1;y<y2;y++)
{
   Tt_n1[x3][y][z1] = ( Tt_n1[x3+1-2*p][y][z1] + Tt_n1[x3][y][z1+1] )/2;
   Tt_n1[x3][y][z2] = ( Tt_n1[x3+1-2*p][y][z2] + Tt_n1[x3][y][z2-1] )/2;
    if(p==0)
    {
      Tbd5[index][0][y-y7+k*(ly5+1)] = Tt_n1[x3][y][z1];
      Tbd5[index][2][y-y7+k*(ly5+1)] = Tt_n1[x3][y][z2];
    }
    else
    {
      Tbd5[index][1][y-y7+k*(ly5+1)] = Tt_n1[x3][y][z1];
      Tbd5[index][3][y-y7+k*(ly5+1)] = Tt_n1[x3][y][z2];
    }
}

for(y=y7+1;y<y3;y++)
{
     Tt_n1[x4][y][z1] = ( Tt_n1[x4+2*p-1][y][z1] + Tt_n1[x4][y][z1+1] )/2;
     Tt_n1[x4][y][z2] = ( Tt_n1[x4+2*p-1][y][z2] + Tt_n1[x4][y][z2-1] )/2;
      if(p==0)
      {
          Tbd5[index][1][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z1];
          Tbd5[index][3][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z2];
      }
      else
      {
          Tbd5[index][0][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z1];
          Tbd5[index][2][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z2];
      }
 }
 for(y=y4+1;y<y2;y++)
 {
      Tt_n1[x4][y][z1] = ( Tt_n1[x4+2*p-1][y][z1] + Tt_n1[x4][y][z1+1] )/2;
      Tt_n1[x4][y][z2] = ( Tt_n1[x4+2*p-1][y][z2] + Tt_n1[x4][y][z2-1] )/2;
       if(p==0)
```

```
        {
            Tbd5[index][1][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z1];
            Tbd5[index][3][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z2];
        }
        else
        {
            Tbd5[index][0][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z1];
            Tbd5[index][2][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z2];
        }
 }
 //vertix
Tt_n1[x1][y7][z1] = ( Tt_n1[x1][y7+1][z1] + Tt_n1[x1][y7][z1+1] )/2;
Tt_n1[x1][y7][z2] = ( Tt_n1[x1][y7+1][z2] + Tt_n1[x1][y7][z2-1] )/2;
Tt_n1[x1][y2][z1] = ( Tt_n1[x1][y2-1][z1] + Tt_n1[x1][y2][z1+1] )/2;
Tt_n1[x1][y2][z2] = ( Tt_n1[x1][y2-1][z2] + Tt_n1[x1][y2][z2-1] )/2;
Tt_n1[x2][y7][z1] = ( Tt_n1[x2][y7+1][z1] + Tt_n1[x2][y7][z1+1] )/2;
Tt_n1[x2][y7][z2] = ( Tt_n1[x2][y7+1][z2] + Tt_n1[x2][y7][z2-1] )/2;
Tt_n1[x2][y2][z1] = ( Tt_n1[x2][y2-1][z1] + Tt_n1[x2][y2][z1+1] )/2;
Tt_n1[x2][y2][z2] = ( Tt_n1[x2][y2-1][z2] + Tt_n1[x2][y2][z2-1] )/2;
Tbd5[index][0][0+k*(ly5+1)] = Tt_n1[x1][y7][z1];
Tbd5[index][1][0+k*(ly5+1)] = Tt_n1[x2][y7][z1];
Tbd5[index][2][0+k*(ly5+1)] = Tt_n1[x1][y7][z2];
Tbd5[index][3][0+k*(ly5+1)] = Tt_n1[x2][y7][z2];
Tbd5[index][0][ly5+k*(ly5+1)] = Tt_n1[x1][y2][z1];
Tbd5[index][1][ly5+k*(ly5+1)] = Tt_n1[x2][y2][z1];
Tbd5[index][2][ly5+k*(ly5+1)] = Tt_n1[x1][y2][z2];
Tbd5[index][3][ly5+k*(ly5+1)] = Tt_n1[x2][y2][z2];



 //horizontal y direction boarder edges of level 4 and 5
 for(y=y3+1;y<y4;y++)
 {
    Tt_n1[x4][y][z1] = ( Tt_n1[x4+2*p-1][y][z1] + Tt_n1[x4][y][z1-1] )/2;
    Tt_n1[x4][y][z2] = ( Tt_n1[x4+2*p-1][y][z2] + Tt_n1[x4][y][z2+1] )/2;
     if(p==0)
     {
        Tbd5[index][1][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z1];
        Tbd5[index][3][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z2];
     }
     else
     {
       Tbd5[index][0][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z1];
       Tbd5[index][2][y-y7+k*(ly5+1)] = Tt_n1[x4][y][z2];
     }
 }
 //vertix of boarder edges of level 5 and 4
Tt_n1[x4][y3][z1] = ( Tt_n1[x4][y3+1][z1] + Tt_n1[x4][y3-1][z1]
                + Tt_n1[x4][y3-1][z1+1] + Tt_n1[x4][y3][z1-1]  )/4;
Tt_n1[x4][y3][z2] = ( Tt_n1[x4][y3+1][z2] + Tt_n1[x4][y3-1][z2]
                + Tt_n1[x4][y3-1][z2+1] + Tt_n1[x4][y3][z2-1]  )/4;
Tt_n1[x4][y4][z1] = ( Tt_n1[x4][y4+1][z1] + Tt_n1[x4][y4-1][z1]
                + Tt_n1[x4][y4-1][z1+1] + Tt_n1[x4][y4][z1-1]  )/4;
Tt_n1[x4][y4][z2] = ( Tt_n1[x4][y4+1][z2] + Tt_n1[x4][y4-1][z2]
                + Tt_n1[x4][y4-1][z2+1] + Tt_n1[x4][y4][z2-1]  )/4;
if(p==0)
```

```
        {
            Tbd5[index][1][y3-y7+k*(ly5+1)] = Tt_n1[x4][y3][z1];
            Tbd5[index][3][y3-y7+k*(ly5+1)] = Tt_n1[x4][y3][z2];
            Tbd5[index][1][y4-y7+k*(ly5+1)] = Tt_n1[x4][y4][z1];
            Tbd5[index][3][y4-y7+k*(ly5+1)] = Tt_n1[x4][y4][z2];
        }
        else
        {
            Tbd5[index][0][y3-y7+k*(ly5+1)] = Tt_n1[x4][y3][z1];
            Tbd5[index][2][y3-y7+k*(ly5+1)] = Tt_n1[x4][y3][z2];
            Tbd5[index][0][y4-y7+k*(ly5+1)] = Tt_n1[x4][y4][z1];
            Tbd5[index][2][y4-y7+k*(ly5+1)] = Tt_n1[x4][y4][z2];
        }
        k++;
    }

//sixth level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
  for(j=0;j<2;j++)//repeat variable in y coordinate
    for(p=0;p<2;p++)//repeat variable in x coordinate{
      for(q=0;q<2;q++) //repeat variable in inner y coordinate
      {
        x1=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx6/2;
        x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2+lx6/2;
        y7=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2-ly6/2;
        y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2+ly6/2;
        z1=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz6/2;
        z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+lz6/2;
        z3=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2-lz5/2;
        z4=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+lz5/2;
        y3=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*q-1)*ly5/2;
        y4=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6;
        for(z=z1+1;z<z2;z++)
          for(y=y7+1;y<y2;y++)
          {
            Tt_n1[x1][y][z] = ( Tt_n1[x1-1][y][z] + Tb6_n1[index][z-z1+k*(lz6+1)]
                              * deltaX*Bi ) / (1+deltaX*Bi);
            Tt_n1[x2][y][z] = ( Tt_n1[x2+1][y][z] + Tb6_n1[index][z-z1+k*(lz6+1)]
                              * deltaX*Bi ) / (1+deltaX*Bi);
          }
        for(z=z1;z<=z2;z++)
          for(x=x1+1;x<x2;x++)//x-z plane
            Tt_n1[x][y4][z] = ( Tt_n1[x][y4+2*q-1][z] + Tb6_n1[index][z-z1+k*(lz6+1)]
                              * deltaY*Bi ) / (1+deltaY*Bi);
        for(z=z1;z<z3;z++)
        {
          for(x=x1+1;x<x2;x++)//x-z plane
            Tt_n1[x][y3][z] = ( Tt_n1[x][y3+1-2*q][z] + Tb6_n1[index][z-z1
                              +k*(lz6+1)] * deltaY*Bi ) / (1+deltaY*Bi);
          //vertical edges
          Tt_n1[x1][y3][z] = ( Tt_n1[x1+1][y3][z] + Tt_n1[x1][y3+2*q-1][z] )/2;
          Tt_n1[x2][y3][z] = ( Tt_n1[x2-1][y3][z] + Tt_n1[x2][y3+2*q-1][z] )/2;
          if(q==0)
          {
            Tbd6[index][2][z-z1+k*(lz6+1)] = Tt_n1[x1][y3][z];
```

```
            Tbd6[index][3][z-z1+k*(lz6+1)] = Tt_n1[x2][y3][z];
          }
          else
          {
            Tbd6[index][0][z-z1+k*(lz6+1)] = Tt_n1[x1][y3][z];
            Tbd6[index][1][z-z1+k*(lz6+1)] = Tt_n1[x2][y3][z];
          }
        }
        for(z=z4+1;z<z2;z++)
        {
          for(x=x1+1;x<x2;x++)//x-z plane
            Tt_n1[x][y3][z] = ( Tt_n1[x][y3+1-2*q][z] + Tb6_n1[index]
                    [z-z1+k*(lz6+1)] * deltaY*Bi ) / (1+deltaY*Bi);
          //vertical edges
          Tt_n1[x1][y3][z] = ( Tt_n1[x1+1][y3][z] + Tt_n1[x1][y3+2*q-1][z] )/2;
          Tt_n1[x2][y3][z] = ( Tt_n1[x2-1][y3][z] + Tt_n1[x2][y3+2*q-1][z] )/2;


          if(q==0)
          {
            Tbd6[index][2][z-z1+k*(lz6+1)] = Tt_n1[x1][y3][z];
            Tbd6[index][3][z-z1+k*(lz6+1)] = Tt_n1[x2][y3][z];
          }
          else
          {
            Tbd6[index][0][z-z1+k*(lz6+1)] = Tt_n1[x1][y3][z];
            Tbd6[index][1][z-z1+k*(lz6+1)] = Tt_n1[x2][y3][z];
          }
        }
        for(z=z1+1;z<z2;z++)
        {
          //vertical edges
          Tt_n1[x1][y4][z] = ( Tt_n1[x1+1][y4][z] + Tt_n1[x1][y4+1-2*q][z] )/2;
          Tt_n1[x2][y4][z] = ( Tt_n1[x2-1][y4][z] + Tt_n1[x2][y4+1-2*q][z] )/2;
          if(q==0)
          {
            Tbd6[index][0][z-z1+k*(lz6+1)] = Tt_n1[x1][y4][z];
            Tbd6[index][1][z-z1+k*(lz6+1)] = Tt_n1[x2][y4][z];
          }
          else
          {
            Tbd6[index][2][z-z1+k*(lz6+1)] = Tt_n1[x1][y4][z];
            Tbd6[index][3][z-z1+k*(lz6+1)] = Tt_n1[x2][y4][z];
          }
        }
        //vertix
        Tt_n1[x1][y7][z1] = ( Tt_n1[x1+1][y7][z1] + Tt_n1[x1][y7][z1+1] )/2;
        Tt_n1[x1][y7][z2] = ( Tt_n1[x1+1][y7][z2] + Tt_n1[x1][y7][z2-1] )/2;
        Tt_n1[x1][y2][z1] = ( Tt_n1[x1+1][y2][z1] + Tt_n1[x1][y2][z1+1] )/2;
        Tt_n1[x1][y2][z2] = ( Tt_n1[x1+1][y2][z2] + Tt_n1[x1][y2][z2-1] )/2;
        Tt_n1[x2][y7][z1] = ( Tt_n1[x2+1][y7][z1] + Tt_n1[x2][y7][z1+1] )/2;
        Tt_n1[x2][y7][z2] = ( Tt_n1[x2+1][y7][z2] + Tt_n1[x2][y7][z2-1] )/2;
        Tt_n1[x2][y2][z1] = ( Tt_n1[x2+1][y2][z1] + Tt_n1[x2][y2][z1+1] )/2;
        Tt_n1[x2][y2][z2] = ( Tt_n1[x2+1][y2][z2] + Tt_n1[x2][y2][z2-1] )/2;
        Tbd6[index][0][0+k*(lz6+1)] = Tt_n1[x1][y7][z1];
        Tbd6[index][1][0+k*(lz6+1)] = Tt_n1[x2][y7][z1];
```

```
                Tbd6[index][2][0+k*(lz6+1)] = Tt_n1[x1][y2][z1];
                Tbd6[index][3][0+k*(lz6+1)] = Tt_n1[x2][y2][z1];
                Tbd6[index][0][lz6+k*(lz6+1)] = Tt_n1[x1][y7][z2];
                Tbd6[index][1][lz6+k*(lz6+1)] = Tt_n1[x2][y7][z2];
                Tbd6[index][2][lz6+k*(lz6+1)] = Tt_n1[x1][y2][z2];
                Tbd6[index][3][lz6+k*(lz6+1)] = Tt_n1[x2][y2][z2];
                 //vertical boarder edges of level 6 and 5
                 for(z=z3+1;z<z4;z++)
                 {
                      Tt_n1[x1][y3][z] = ( Tt_n1[x1-1][y3][z] + Tt_n1[x1][y3+2*q-1][z] )/2;
                      Tt_n1[x2][y3][z] = ( Tt_n1[x2+1][y3][z] + Tt_n1[x2][y3+2*q-1][z] )/2;

                      if(q==0)
                      {
                         Tbd6[index][2][z-z1+k*(lz6+1)] = Tt_n1[x1][y3][z];
                         Tbd6[index][3][z-z1+k*(lz6+1)] = Tt_n1[x2][y3][z];
                      }
                      else
                      {
                         Tbd6[index][0][z-z1+k*(lz6+1)] = Tt_n1[x1][y3][z];
                         Tbd6[index][1][z-z1+k*(lz6+1)] = Tt_n1[x2][y3][z];
                      }
                 }
                 //vertix of boarder edges of level 5 and 6
                Tt_n1[x1][y3][z3] = ( Tt_n1[x1+1][y3][z3] + Tt_n1[x1-1][y3][z3]
                                + Tt_n1[x1][y3][z3+1] + Tt_n1[x1][y3][z3-1] )/4;
                Tt_n1[x1][y3][z4] = ( Tt_n1[x1+1][y3][z4] + Tt_n1[x1-1][y3][z4]
                                + Tt_n1[x1][y3][z4+1] + Tt_n1[x1][y3][z4-1] )/4;
                Tt_n1[x2][y3][z3] = ( Tt_n1[x2+1][y3][z3] + Tt_n1[x2-1][y3][z3]
                                + Tt_n1[x2][y3][z3+1] + Tt_n1[x2][y3][z3-1] )/4;
                Tt_n1[x2][y3][z4] = ( Tt_n1[x2+1][y3][z4] + Tt_n1[x2-1][y3][z4]
                                + Tt_n1[x2][y3][z4+1] + Tt_n1[x2][y3][z4-1] )/4;
                 if(q==0)
                 {
                     Tbd6[index][2][z3-z1+k*(lz6+1)] = Tt_n1[x1][y3][z3];
                     Tbd6[index][3][z3-z1+k*(lz6+1)] = Tt_n1[x2][y3][z3];
                     Tbd6[index][2][z4-z1+k*(lz6+1)] = Tt_n1[x1][y3][z4];
                     Tbd6[index][3][z4-z1+k*(lz6+1)] = Tt_n1[x2][y3][z4];
                 }
                 else
                {
                      Tbd6[index][0][z3-z1+k*(lz6+1)] = Tt_n1[x1][y3][z3];
                      Tbd6[index][1][z3-z1+k*(lz6+1)] = Tt_n1[x2][y3][z3];
                      Tbd6[index][0][z4-z1+k*(lz6+1)] = Tt_n1[x1][y3][z4];
                      Tbd6[index][1][z4-z1+k*(lz6+1)] = Tt_n1[x2][y3][z4];
                 }

                 k++;
            }

//seventh level
k=0;
for(r=0;r<2;r++)//repeat variable in z coordinate
   for(j=0;j<2;j++)//repeat variable in y coordinate
      for(p=0;p<2;p++)//repeat variable in x coordinate
         for(q=0;q<2;q++) //repeat variable in inner y coordinate
```

```
for(h=0;h<2;h++)//repeat variable in inner z coordinate
{
      x1=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx7/2;
       x2=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2+lx7/2;
       x3=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2-lx6/2;
       x4=cenX+(2*p-1)*lx4/2+(2*p-1)*lx5/2+lx6/2;
    y7=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2-ly7/2;
    y2=cenY+(2*j-1)*ly2/2+(2*j-1)*ly3/2+(2*q-1)*ly5/2+(2*q-1)*ly6/2+ly7/2;
    z1=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+(2*h-1)*lz6/2+(2*h-1)*lz7/2-lz7/2;
    z2=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+(2*h-1)*lz6/2+(2*h-1)*lz7/2+lz7/2;
    z3=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+(2*h-1)*lz6/2;
    z4=cenZ+(2*r-1)*lz3/2+(2*r-1)*lz4/2+(2*h-1)*lz6/2+(2*h-1)*lz7;
    for(y=y7+1;y<y2;y++)//x-y plane
       for(x=x1+1;x<x2;x++)
           Tt_n1[x][y][z4] = ( Tt_n1[x][y][z4+2*h-1] + Tb7_n1[index][x-x1+k*(lx7+1)]
                       * deltaZ*Bi ) / (1+deltaZ*Bi);
    for(y=y7+1;y<y2;y++)//x-y plane
       for(x=x1+1;x<x3;x++)
           Tt_n1[x][y][z3] = ( Tt_n1[x][y][z3+1-2*h] + Tb7_n1[index][x-x1+k*(lx7+1)]
                       * deltaZ*Bi ) / (1+deltaZ*Bi);
    for(y=y7+1;y<y2;y++)//x-y plane
       for(x=x4+1;x<x2;x++)
           Tt_n1[x][y][z3] = ( Tt_n1[x][y][z3+1-2*h] + Tb7_n1[index][x-x1+k*(lx7+1)]
                       * deltaZ*Bi ) / (1+deltaZ*Bi);
    for(z=z1+1;z<z2;z++)
    {
       for(x=x1+1;x<x2;x++)//x-z plane
       {
           Tt_n1[x][y7][z] = ( Tt_n1[x][y7-1][z] + Tb7_n1[index][x-x1+k*(lx7+1)]
                       * deltaY*Bi ) / (1+deltaY*Bi);
           Tt_n1[x][y2][z] = ( Tt_n1[x][y2+1][z] + Tb7_n1[index][x-x1+k*(lx7+1)]
                       * deltaY*Bi ) / (1+deltaY*Bi);
       }
       //vertical edges
       Tt_n1[x1][y7][z] = ( Tt_n1[x1+1][y7][z] + Tt_n1[x1][y7+1][z] )/2;
       Tt_n1[x1][y2][z] = ( Tt_n1[x1+1][y2][z] + Tt_n1[x1][y2-1][z] )/2;
       Tt_n1[x2][y7][z] = ( Tt_n1[x2-1][y7][z] + Tt_n1[x2][y7+1][z] )/2;
       Tt_n1[x2][y2][z] = ( Tt_n1[x2-1][y2][z] + Tt_n1[x2][y2-1][z] )/2;
    }
    //horizontal x direction edges
    for(x=x1+1;x<x2;x++)
    {
       Tt_n1[x][y7][z4] = ( Tt_n1[x][y7+1][z4] + Tt_n1[x][y7][z4+1-2*h] )/2;
       Tt_n1[x][y2][z4] = ( Tt_n1[x][y2-1][z4] + Tt_n1[x][y2][z4+1-2*h] )/2;
       if(h==0)
       {
           Tbd7[index][0][x-x1+k*(lx7+1)] = Tt_n1[x][y7][z4];
           Tbd7[index][1][x-x1+k*(lx7+1)] = Tt_n1[x][y2][z4];
       }
       else
       {
           Tbd7[index][2][x-x1+k*(lx7+1)] = Tt_n1[x][y7][z4];
           Tbd7[index][3][x-x1+k*(lx7+1)] = Tt_n1[x][y2][z4];
       }
    }
    //horizontal x direction edges
```

```
for(x=x1+1;x<x3;x++)
{
    Tt_n1[x][y7][z3] = ( Tt_n1[x][y7+1][z3] + Tt_n1[x][y7][z3+2*h-1] )/2;
    Tt_n1[x][y2][z3] = ( Tt_n1[x][y2-1][z3] + Tt_n1[x][y2][z3+2*h-1] )/2;
    if(h==0)
    {
        Tbd7[index][2][x-x1+k*(lx7+1)] = Tt_n1[x][y7][z3];
        Tbd7[index][3][x-x1+k*(lx7+1)] = Tt_n1[x][y2][z3];
    }
    else
    {
        Tbd7[index][0][x-x1+k*(lx7+1)] = Tt_n1[x][y7][z3];
        Tbd7[index][1][x-x1+k*(lx7+1)] = Tt_n1[x][y2][z3];
    }
}
//horizontal x direction edges
for(x=x4+1;x<x2;x++)
{
    Tt_n1[x][y7][z3] = ( Tt_n1[x][y7+1][z3] + Tt_n1[x][y7][z3+2*h-1] )/2;
    Tt_n1[x][y2][z3] = ( Tt_n1[x][y2-1][z3] + Tt_n1[x][y2][z3+2*h-1] )/2;
    if(h==0)
    {
        Tbd7[index][2][x-x1+k*(lx7+1)] = Tt_n1[x][y7][z3];
        Tbd7[index][3][x-x1+k*(lx7+1)] = Tt_n1[x][y2][z3];
    }
    else
    {
        Tbd7[index][0][x-x1+k*(lx7+1)] = Tt_n1[x][y7][z3];
        Tbd7[index][1][x-x1+k*(lx7+1)] = Tt_n1[x][y2][z3];
    }
}
//horizontal y direction edges
for(y=y7+1;y<y2;y++)
{
    Tt_n1[x1][y][z1] = ( Tt_n1[x1+1][y][z1] + Tt_n1[x1][y][z1+1] )/2;
    Tt_n1[x1][y][z2] = ( Tt_n1[x1+1][y][z2] + Tt_n1[x1][y][z2-1] )/2;
    Tt_n1[x2][y][z1] = ( Tt_n1[x2-1][y][z1] + Tt_n1[x2][y][z1+1] )/2;
    Tt_n1[x2][y][z2] = ( Tt_n1[x2-1][y][z2] + Tt_n1[x2][y][z2-1] )/2;
}
//vertix
Tt_n1[x1][y7][z1] = ( Tt_n1[x1+1][y7][z1] + Tt_n1[x1][y7+1][z1] + Tt_n1[x1][y7][z1+1] )/3;
Tt_n1[x1][y7][z2] = ( Tt_n1[x1+1][y7][z2] + Tt_n1[x1][y7+1][z2] + Tt_n1[x1][y7][z2-1] )/3;
Tt_n1[x1][y2][z1] = ( Tt_n1[x1+1][y2][z1] + Tt_n1[x1][y2-1][z1] + Tt_n1[x1][y2][z1+1] )/3;
Tt_n1[x1][y2][z2] = ( Tt_n1[x1+1][y2][z2] + Tt_n1[x1][y2-1][z2] + Tt_n1[x1][y2][z2-1] )/3;
Tt_n1[x2][y7][z1] = ( Tt_n1[x2-1][y7][z1] + Tt_n1[x2][y7+1][z1] + Tt_n1[x2][y7][z1+1] )/3;
Tt_n1[x2][y7][z2] = ( Tt_n1[x2-1][y7][z2] + Tt_n1[x2][y7+1][z2] + Tt_n1[x2][y7][z2-1] )/3;
Tt_n1[x2][y2][z1] = ( Tt_n1[x2-1][y2][z1] + Tt_n1[x2][y2-1][z1] + Tt_n1[x2][y2][z1+1] )/3;
Tt_n1[x2][y2][z2] = ( Tt_n1[x2-1][y2][z2] + Tt_n1[x2][y2-1][z2] + Tt_n1[x2][y2][z2-1] )/3;
Tbd7[index][0][0+k*(lx7+1)] = Tt_n1[x1][y7][z1];
Tbd7[index][1][0+k*(lx7+1)] = Tt_n1[x1][y2][z1];
Tbd7[index][2][0+k*(lx7+1)] = Tt_n1[x1][y7][z2];
Tbd7[index][3][0+k*(lx7+1)] = Tt_n1[x1][y2][z2];
Tbd7[index][0][lx7+k*(lx7+1)] = Tt_n1[x2][y7][z1];
Tbd7[index][1][lx7+k*(lx7+1)] = Tt_n1[x2][y2][z1];
Tbd7[index][2][lx7+k*(lx7+1)] = Tt_n1[x2][y7][z2];
Tbd7[index][3][lx7+k*(lx7+1)] = Tt_n1[x2][y2][z2];
```

```
                //horizontal y direction boarder edges of level 6 and 7
                for(y=y7+1;y<y2;y++)
                {
                    Tt_n1[x3][y][z3] = ( Tt_n1[x3-1][y][z3] + Tt_n1[x3][y][z3+1-2*h] )/2;
                    Tt_n1[x4][y][z3] = ( Tt_n1[x4+1][y][z3] + Tt_n1[x4][y][z2+1-2*h] )/2;
                }
                //horizontal x direction boarder edges of level 6 and 7
                for(x=x3+1;x<x4;x++)
                {
                    Tt_n1[x][y7][z3] = ( Tt_n1[x][y7-1][z3] + Tt_n1[x][y7][z3+2*h-1] )/2;
                    Tt_n1[x][y2][z3] = ( Tt_n1[x][y2+1][z3] + Tt_n1[x][y2][z3+2*h-1] )/2;
                    if(h==0)
                    {
                        Tbd7[index][2][x-x1+k*(lx7+1)] = Tt_n1[x][y7][z3];
                        Tbd7[index][3][x-x1+k*(lx7+1)] = Tt_n1[x][y2][z3];
                    }
                    else
                    {
                        Tbd7[index][0][x-x1+k*(lx7+1)] = Tt_n1[x][y7][z3];
                        Tbd7[index][1][x-x1+k*(lx7+1)] = Tt_n1[x][y2][z3];
                    }
                }
                //vertix of boarder edges of level 7 and 6
                Tt_n1[x3][y7][z3] = ( Tt_n1[x3+1][y7][z3] + Tt_n1[x3-1][y7][z3]
                                    + Tt_n1[x3][y7+1][z3] + Tt_n1[x3][y7-1][z3] )/4;
                Tt_n1[x3][y2][z3] = ( Tt_n1[x3+1][y2][z3] + Tt_n1[x3-1][y2][z3]
                                    + Tt_n1[x3][y2+1][z3] + Tt_n1[x3][y2-1][z3] )/4;
                Tt_n1[x4][y7][z3] = ( Tt_n1[x4+1][y7][z3] + Tt_n1[x4-1][y7][z3]
                                    + Tt_n1[x4][y7+1][z3] + Tt_n1[x4][y7-1][z3] )/4;
                Tt_n1[x4][y2][z3] = ( Tt_n1[x4+1][y2][z3] + Tt_n1[x4-1][y2][z3]
                                    + Tt_n1[x4][y2+1][z3] + Tt_n1[x4][y2-1][z3] )/4;
                if(h==0)
                {
                    Tbd7[index][2][x3-x1+k*(lx7+1)] = Tt_n1[x3][y7][z3];
                    Tbd7[index][3][x3-x1+k*(lx7+1)] = Tt_n1[x3][y2][z3];
                    Tbd7[index][2][x4-x1+k*(lx7+1)] = Tt_n1[x4][y7][z3];
                    Tbd7[index][3][x4-x1+k*(lx7+1)] = Tt_n1[x4][y2][z3];
                }
                else
                {
                    Tbd7[index][0][x3-x1+k*(lx7+1)] = Tt_n1[x3][y7][z3];
                    Tbd7[index][1][x3-x1+k*(lx7+1)] = Tt_n1[x3][y2][z3];
                    Tbd7[index][0][x4-x1+k*(lx7+1)] = Tt_n1[x4][y7][z3];
                    Tbd7[index][1][x4-x1+k*(lx7+1)] = Tt_n1[x4][y2][z3];
                }
                k++;
            }
            return;
}


void writeSquareXY(int x0, int y0, int x1, int y1, int z, int t, int I)
{
        FILE *file;
        char fname[256], line[8196];
        //strcpy(fname, outPath);
```

```
        sprintf(fname, "Z%d_%d_%d_%d_%d_t%d_I%d.txt", z, x0, y0, x1, y1, t, I);
        file = fopen(fname, "w");
        for(int j=y0;j<=y1;j++)
        {
                strcpy(line, "");
                for(int i=x0;i<=x1;i++)
                {
                        sprintf(tmp, "%10.6f ", Tt_n1[i][j][z]);
                        strcat(line, tmp);
                }
                strcat(line, "\n");
                fwrite(line,strlen(line), 1, file);
        }
        fclose(file);
}

void writeSquareXZ(int x0, int z0, int x1, int z1, int y, int t, int I)
{
        FILE *file;
        char fname[256], line[8196];
        //printf("running writexz.....\n");
        //strcpy(fname, outPath);
        sprintf(fname, "Y%d_%d_%d_%d_%d_t%d_I%d.txt", y, x0, z0, x1, z1, t, I);
        file = fopen(fname, "w");
        for(int k=z0;k<=z1;k++)
        {
                strcpy(line, "");
                for(int i=x0;i<=x1;i++)
                {
                        sprintf(tmp, "%10.6f ", Tt_n1[i][y][k]);
                        strcat(line, tmp);
                }
                strcat(line, "\n");
                fwrite(line,strlen(line), 1, file);
        }
        fclose(file);
}

void writeSquareXZ_Damage(int x0, int z0, int x1, int z1, int y, int t, int I)
{
        FILE *file;
        char fname[256], line[8196];
        //printf("running writexz.....\n");
        //strcpy(fname, outPath);
        sprintf(fname, "Damage_Y%d_%d_%d_%d_%d_t%d_I%d.txt", y, x0, z0, x1, z1, t, I);
        file = fopen(fname, "w");
        for(int k=z0;k<=z1;k++)
        {
                strcpy(line, "");
                for(int i=x0;i<=x1;i++)
                {
                        sprintf(tmp, "%10.4e ", Damage_n1[i][y][k]);
                        strcat(line, tmp);
                }
                strcat(line, "\n");
                fwrite(line,strlen(line), 1, file);
```

```
        }
        fclose(file);
}

void writeSquareYZ(int y0, int z0, int y1, int z1, int x, int t, int I)
{
        FILE *file;
        char fname[256], line[8196];
        //strcpy(fname, outPath);
        sprintf(fname, "X%d_%d_%d_%d_%d_t%d_I%d.txt", x, y0, z0, y1, z1, t, I);
        file = fopen(fname, "w");
        for(int k=z0;k<=z1;k++)
        {
                strcpy(line, "");
                for(int j=y0;j<=y1;j++)
                {
                        sprintf(tmp, "%10.6f ", Tt_n1[x][j][k]);
                        strcat(line, tmp);
                }
                strcat(line, "\n");
                fwrite(line,strlen(line), 1, file);
        }
        fclose(file);
}

void writeSquareYZ_Damage(int y0, int z0, int y1, int z1, int x, int t, int I)
{
        FILE *file;
        char fname[256], line[8196];
        //strcpy(fname, outPath);
        sprintf(fname, "Damage_X%d_%d_%d_%d_%d_t%d_I%d.txt", x, y0, z0, y1, z1, t, I);
        file = fopen(fname, "w");
        for(int k=z0;k<=z1;k++)
        {
                strcpy(line, "");
                for(int j=y0;j<=y1;j++)
                {
                        sprintf(tmp, "%10.4e ", Damage_n1[x][j][k]);
                        strcat(line, tmp);
                }
                strcat(line, "\n");
                fwrite(line,strlen(line), 1, file);
        }
        fclose(file);
}

void writeZCenter(int t)
{
        FILE *file;
        char fname[256], line[8196];
        //strcpy(fname, outPath);
        sprintf(fname, "Z_Center_t%d.txt", t);
        file = fopen(fname, "w");
        for(int j=0;j<=NZ3;j++)
        {
                sprintf(tmp, "%10.6f ", Tt_n1[centerX][centerY][j]);
```

```
                strcat(line, tmp);
                strcat(line, "\n");
                fwrite(line,strlen(line), 1, file);
        }
        fclose(file);
}

void writeAll(int t, int I)
{
        FILE *file;
        char fname[256];
        //strcpy(fname, outPath);
        sprintf(fname, "t%d_I%d.dat", t, I);
        file = fopen(fname, "wb");
        for(int i=0;i<NX+1;i++)
                fwrite(Tt_n1[i],sizeof(double)*(NY+1)*(NZ3+1), 1, file);
        fclose(file);
        return;
}

void writeLinearSys(int x, int y, int t, int I)
{
        FILE *file;
        char fname[256], line[256];
        //strcpy(fname, outPath);
        sprintf(fname, "X%d_Y%d_t%d_I%d.txt", x, y, t, I);
        file = fopen(fname, "w");
        for(int k=0;k<=NZ3;k++)
        {
                sprintf(line, "%d\t%10.6f\t%10.6f\t%10.6f\t%10.6f\n", k,
                        b[x][y][k],a[x][y][k],c[x][y][k],d[x][y][k]);
                fwrite(line,strlen(line), 1, file);
        }
        fclose(file);
}

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

void writebloode(int i, double x)
{
        FILE *file;
        char fname[256], line[256];
        //strcpy(fname, outPath);
        sprintf(fname, "bloodTep.txt");
        file = fopen(fname, "a");
```

```
            sprintf(line, "%d\t%10.6f\n", i, x);
            fwrite(line,strlen(line), 1, file);
           fclose(file);
}
```