AD_____

REPORT NO T94-10

NUMERICAL MODEL OF THE THERMAL BEHAVIOR OF AN EXTREMITY IN A COLD ENVIRONMENT INCLUDING COUNTER-CURRENT HEAT EXCHANGE BETWEEN THE BLOOD VESSELS

U S ARMY RESEARCH INSTITUTE
OF
ENVIRONMENTAL MEDICINE
Natick, Massachusetts

MARCH 1994

UNITED STATES ARMY
MEDICAL RESEARCH & DEVELOPMENT COMMAND

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1994	Technical Report

4. TITLE AND SUBTITLE
Numerical Model of the Thermal Behavior of an Extremity in a Cold Environment Including Counter-Current Heat Exchange Between the Blood Vessels

5. FUNDING NUMBERS

6. AUTHOR(S)
Avraham Shitzer, Leander A. Stroschein, Paul Vital, Richard R. Gonzalez, Kent B. Pandolf

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Institute of Environmental Medicine
Natick, MA, 01760-5007

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Approved for public release; distribution unlimited

12a. DISTRIBUTION AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A numerical model of the thermal behavior of an extremity, e.g., finger, is presented. The model includes the effects of: (a) heat conduction (b) metabolic heat generation, (c) heat transport by blood perfusion, (d) heat exchange between the tissue and the large blood vessels, and, (e) arterio-venous heat exchange. Heat exchange with the environment through a layer of thermal insulation, depicted by thermal handwear is also considered. The tissue is subdivided into four concentric layers. The layers described, from the center outward, as core, muscle, fat and skin. Differential heat balance equations are formulated for the tissue and the major artery and the major vein. These coupled equations are solved numerically by the alternating direction method employing a Thomas algorithm. The numerical scheme was tested extensively for stability and convergence. Results of the convergence tests are presented and discussed and the dependence on the number of grid points is demonstrated. Plots of tissue and blood temperatures along selected nodes of the model are shown for different combinations of parameters. The effect of counter-current heat exchange between the artery and the vein on the thermal balance of the extremity is presented. This shows clearly the conservation of energy achieved due to this mechanism. The report is concluded by considering the effects of cold induced vasodilatation on tissue temperature cycling.

14. SUBJECT TERMS	15. NUMBER OF PAGES
numerical model, physiological model, extremity model, counter-current, heat exchange, blood profusion, heat conduction, cold induced vasodilation, heat balance, alternating direction method, Thomas algorithm, thermal insulation, cold	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NUMERICAL MODEL OF THE THERMAL BEHAVIOR OF AN EXTREMITY IN A COLD ENVIRONMENT INCLUDING COUNTER-CURRENT HEAT EXCHANGE BETWEEN THE BLOOD VESSELS

by

Avraham Shitzer, Leander A. Stroschein, Paul Vital,
Richard R. Gonzalez and Kent B. Pandolf

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

The authors wish to acknowledge the expert help provided by Ms. Laurie A. Blanchard, Sgt. Julio Gonzalez and Ms. Deborah A. Toyota. This work was carried out while the senior author held a Senior NRC Research Associateship at USARIEM.

INTRODUCTION

Heat exchange between the human and the environment has always been a topic of great interest as this is one of the essential manifestations of homeothermy. This interest has intensified over the past few decades as the understanding of the mechanisms involved are spurred on by human's venturing into extreme environments, e.g., outer space. Even less extreme environments may pose life threatening challenges to humans and the literature is laden with examples of both heat, [1] and cold, [2] related casualties.

Physiological studies, in which human subjects are exposed to extreme environmental conditions are essential for collecting data on the actual thermal behavior of men and women. These studies are employed to generate detailed and reliable databases and to test thermoregulation theories. Much information may also be gained by formulating models which simulate qualitatively the thermal behavior of the human body. The chief advantage of these models is their ability to predict and point out trends and limitations while avoiding the dangers and cost involved in actually performing, time consuming and sometimes hazardous experiments. Their inherent disadvantage resides in the necessity to verify their predictions. A variety of models simulating human thermal behavior have been developed. These range from models of single organs, e.g., [3-5] to models of the entire body, e.g., [6-9].

In this report a detailed model of an extremity exposed to cold weather is developed. The reasons for choosing an extremity are two-fold: (a) a model of an extremity may serve as a "building block" for other elements, and, (b) the extremities are usually the most vulnerable body elements particularly in cases involving operations in cold weather.

The extremity is depicted as a right-angle cylinder in which heat flows in both the radial and axial directions. Around the entire external surface of the cylinder different layers of insulation may be applied through which heat is exchanged with the environment. Heat is also exchanged internally by conduction and with the blood flowing both in the major blood vessels and in the vessels of the capillary bed. Counter-current heat exchange between the major blood vessels is also taken into account.

The model is presented as a consistent set of energy balance equations and is solved by a finite-difference, alternating directions numerical scheme employing the Thomas algorithm. This scheme has been tested extensively for stability, convergence and accuracy. It was also run for a number of cases to demonstrate its fundamental capabilities.

ANALYSIS

Energy balance in a right - angle circular cylinder depicted in Fig. 1 is expressed by:

$$\rho c \frac{\partial T^*}{\partial t^*} = k\left[\frac{1}{r^*}\frac{\partial}{\partial r^*}\left(r^* \frac{\partial T^*}{\partial r^*}\right) + \frac{\partial^2 T^*}{\partial z^{*2}}\right] + q_m + w_b c_b (T_a^* - T^*) + u_a (T_a^* - T^*) + u_v (T_v^* - T^*) \quad (1)$$

where the term on the left hand-side represents the rate of change of stored energy and the terms on the right hand-side express radial and axial heat conduction, metabolic heat generation rate, heat exchange with the capillary bed and heat exchange with a large artery and a large vein, respectively. All properties and variables are defined in the Glossary and asterisks indicate dimensional variables.

The following boundary and initial conditions are specified for the problem:

At the center of the cylinder an adiabatic condition is formulated to satisfy symmetry requirements:

$$\frac{\partial T^*}{\partial r^*} = 0 \qquad @\ r^* = 0 \quad (2)$$

On the outer circumferential surface of the cylinder heat is exchanged with the environment by convection:

$$\frac{\partial T^*}{\partial r^*} = \frac{h}{k}(T_0^* - T^*) \qquad @\ r^* = R \quad (3)$$

At the base of the cylinder a variable temperature is assumed:

$$T^* = T_1^*(t^*) \qquad @\ z^* = 0 \quad (4)$$

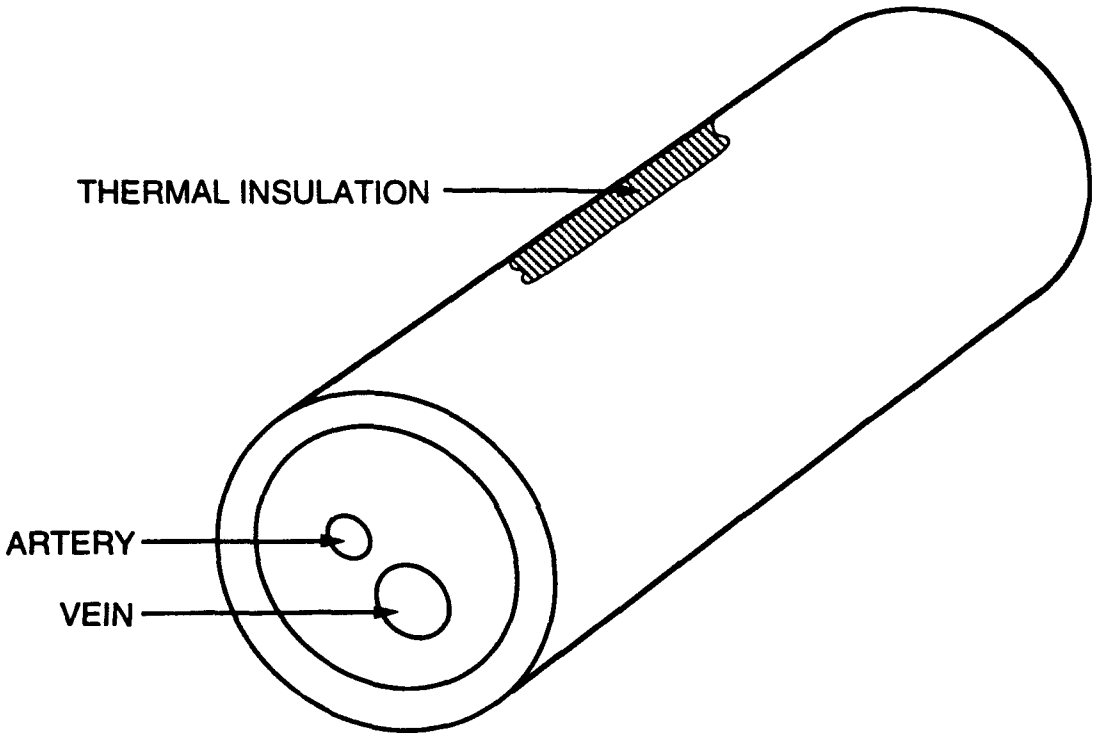


Figure 1: Schematic diagram of the cylindrical model of an extremity

At the tip of the cylinder convective heat exchange with the environment occurs:

$$\frac{\partial T^*}{\partial z^*} = \frac{h_1}{k}(T_0^* - T^*) \qquad @\ z^* = L \tag{5}$$

Initially the temperature distribution in the cylinder is expressed by an arbitrary function:

$$T^* = T_i^*(r^*, z^*) \qquad @\ t^* = 0 \tag{6}$$

Equation (1) contains terms expressing the two modes by which heat is exchanged between the tissue and the circulatory system. In these expressions T_a^* and T_v^* represent arterial and venous temperature distributions, respectively. It is assumed that the cylinder is traversed by one each of these large blood vessels in the axial direction (Fig. 1). In addition to exchanging heat with the surrounding tissue, these two vessels are also coupled by counter-current heat exchange. Two separate, but coupled heat balance equations are now written for the large artery and the large vein, respectively:

$$M_a c_b \frac{d\overline{T}_a^*}{dt^*} = \dot{m}_{a,\,in} c_b T_{a,\,in}^* - \dot{m}_{a,\,out} c_b T_{a,\,out}^* + \int u_a (\overline{T}^* - \overline{T}_a^*)\, dv + h_{av}(\overline{T}_v^* - \overline{T}_a^*) - \int w_b c_b \overline{T}_a^*\, dv \tag{7}$$

and,

$$M_v c_b \frac{d\overline{T}_v^*}{dt^*} = \dot{m}_{v,\,in} c_b T_{v,\,in}^* - \dot{m}_{v,\,out} c_b T_{v,\,out}^* + \int u_v (\overline{T}^* - \overline{T}_v^*)\, dv + h_{av}(\overline{T}_a^* - \overline{T}_v^*) + \int w_b c_b \overline{T}^*\, dv \tag{8}$$

The terms on the left hand-side of Equations (7) and (8) represent the rate of change of the average amount of energy stored in the blood contained at any instance in these two vessels. The terms on the right hand-side of these equations represent, respectively, the enthalpy flux into and out of the control volume, the contribution to the heat balance due to the heat exchange with the average temperature of the surrounding tissue, $\overline{T}$, and the contribution due to counter-current heat exchange between the two large blood vessels. The remaining terms in Equations (7) and (8) indicate, respectively, the drainage into and the collection from the tissue by capillary perfusion as the two vessels traverse the cylinder. The heat transfer coefficients between the blood vessels and the surrounding tissue (u_a and u_v), and between the two blood vessels (h_{av}) are derived in Appendix B.

Two additional mass conservation equations are required for both large blood vessels:

$$\dot{m}_{a,\,in} = \dot{m}_{a,\,out} + \int w_b \, dv \tag{9}$$

and,

$$\dot{m}_{v,\,in} = \dot{m}_{v,\,out} - \int w_b \, dv \tag{10}$$

Prior to applying a numerical solution to the coupled Equations (1), (7) and (8), subject to boundary and initial conditions (2)-(6), these equations are rewritten in dimensionless forms:

$$\frac{\partial T}{\partial \tau} = \frac{\alpha}{\alpha_b} \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial T}{\partial r} \right) + \frac{1}{a^2} \frac{\partial^2 T}{\partial z^2} + \Psi \cdot \left[Q + (W + U_a)(T_a - T) + U_v (T_v - T) \right] \tag{11}$$

$$\frac{d\overline{T}_a}{d\tau} = \frac{R^2}{M_a C_b \alpha_b} \left[\dot{m}_{a,in} C_b T_{a,in} - \dot{m}_{a,out} C_b T_{a,out} + \int u_a (\overline{T} - \overline{T}_a) \, dv + h_{av} (\overline{T}_v - \overline{T}_a) - \int w_b c_b \overline{T}_a dv \right] \tag{12}$$

$$\frac{d\overline{T}_v}{d\tau} = \frac{R^2}{M_v C_b \alpha_b} \left[\dot{m}_{v,\,in} C_b T_{v,\,in} - \dot{m}_{v,\,out} C_b T_{v,\,out} + \int u_v (\overline{T} - \overline{T}_v) \, dv + h_{av} (\overline{T}_a - \overline{T}_v) + \int w_b c_b \overline{T} \, dv \right] \tag{13}$$

$$\frac{\partial T}{\partial r} = 0 \qquad @ \; r = 0 \tag{14}$$

$$\frac{\partial T}{\partial r} = Bi \, (T_0 - T) \qquad @ \; r = 1 \tag{15}$$

$$T = T_1(t) \qquad @\ z = 0 \tag{16}$$

$$\frac{\partial T}{\partial z} = Bi_1 (T_0 - T) \qquad @\ z = 1 \tag{17}$$

$$T = T_i(r, z) \qquad @\ t = 0 \tag{18}$$

where,

$$r = \frac{r^*}{R} \tag{19}$$

$$z = \frac{z^*}{L} \tag{20}$$

$$\tau = \frac{\alpha_b t^*}{R^2} \tag{21}$$

$$T = \frac{T^*}{Temp} \tag{22}$$

$$a^2 = \left(\frac{L}{R}\right)^2 \frac{\alpha_b}{\alpha} \tag{23}$$

$$\Psi = \frac{\rho_b C_b}{\rho C} \tag{24}$$

$$q = \frac{q_s \cdot R^2}{Temp \cdot k_b} \tag{25}$$

$$W = \frac{w_b \cdot C_b \cdot R^2}{k_b} \tag{26}$$

$$U_{a\,or\,v} = \frac{u_{a\,or\,v} \cdot R^2}{k_b} \tag{27}$$

$$Bi = \frac{h \cdot R}{k} \tag{28}$$

and,

$$Bi_1 = \frac{h_1 \cdot L}{k} \tag{29}$$

A finite-difference solution is formulated for the above set of dimensionless equations. The cylinder is divided into four radial compartments depicting the core, muscle, fat and skin, respectively, as shown in Fig. 2. Each of these compartments, the boundaries of which are determined by anatomical and anthropometric considerations, may be further subdivided radially according to the required details of the temperature variations in the cylinder. Axial divisions are uniformly spaced. A cross section of a typical control volume is shown in Fig. 3.

As a first step in the numerical solution , Equation (11) is multiplied by a hollow cylindrical volume element of thickness dr and length dz and integrated:

$$\int_{r-\frac{\Delta r}{2}}^{r+\frac{\Delta r}{2}} \int_{z-\frac{\Delta z}{2}}^{z+\frac{\Delta z}{2}} \frac{\partial T}{\partial \tau} r\, dr\, dz = \int_{r-\frac{\Delta r}{2}}^{r+\frac{\Delta r}{2}} \int_{z-\frac{\Delta z}{2}}^{z+\frac{\Delta z}{2}} \left\{ \frac{\alpha}{\alpha_b} \frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial T}{\partial r} \right) + \frac{1}{a^2} \frac{\partial^2 T}{\partial z^2} + \right. \tag{30}$$

$$\left. \Psi \cdot [\, q + (W + U_a)(T_a - T) + U_v (T_v - T)\,] \right\} r\, dr\, dz$$

The temporal derivative on the left hand-side of Equation (30) is calculated by a forward difference. In evaluating this derivative, a half time step is assumed to facilitate a solution of this two-dimensional problem by the method of alternating directions [10]. In this method the solution of the resulting set of difference equations is performed in two half time steps: first the temperatures in one direction, e.g., radial, are calculated for the first half time step, based on the values of the temperatures at the current time in the other direction. Next, the temperatures at the other direction, e.g., axial, are evaluated for the next half time step based on the values obtained for the first spatial direction in the first half time step. This completes a full time step, and these values are used to initiate the next full time-step iteration.

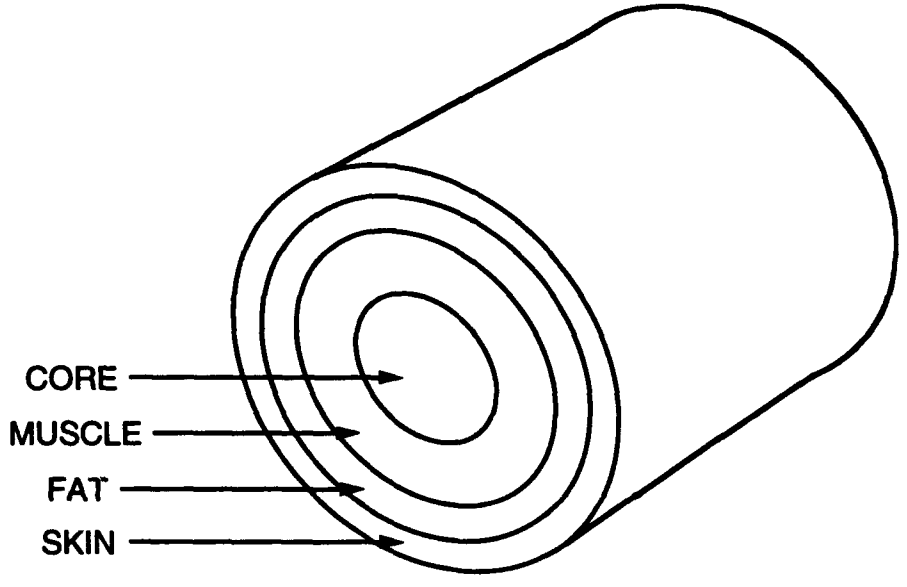


Figure 2: Cross section through the cylindrical model showing the four radial tissue compartments

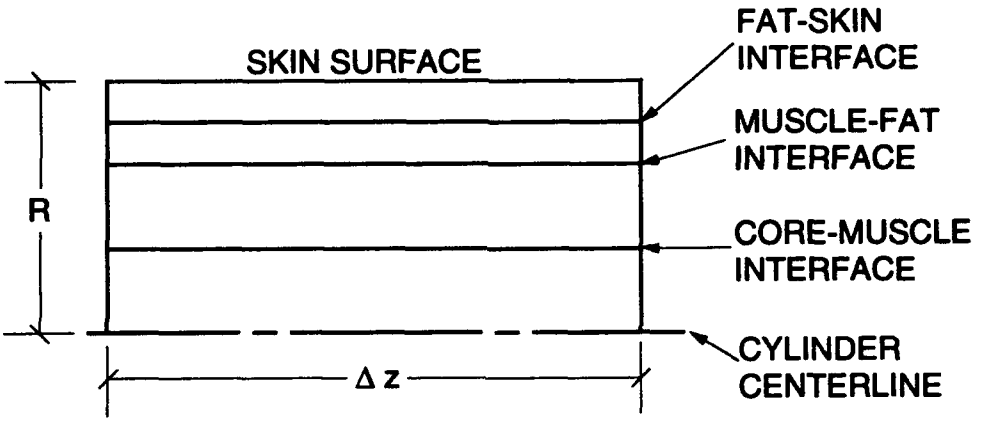


Figure 3: Cross section of a typical control volume of the cylindrical model

Accordingly, the integration of Equation (30) yields:

$$\frac{T_{i,j}^{n+\frac{1}{2}} - T_{i,j}^{n}}{\frac{\Delta\tau}{2}} r_i \cdot \Delta r \cdot \Delta z =$$

$$\frac{\alpha}{\alpha_b} \left\{ (r_i + \frac{\Delta r}{2}) \frac{T_{i+1,j}^{n+\frac{1}{2}} - T_{i,j}^{n+\frac{1}{2}}}{\Delta r} - (r_i - \frac{\Delta r}{2}) \frac{T_{i,j}^{n+\frac{1}{2}} - T_{i-1,j}^{n+\frac{1}{2}}}{\Delta r} \right\} \Delta z +$$

$$\frac{1}{a^2} r_i \Delta r \frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{\Delta z} + q \cdot \Psi \cdot r_i \cdot \Delta r \cdot \Delta z + \quad (31)$$

$$(W + U_a) \cdot \Psi \left\{ T_a^{n+\frac{1}{2}} - \frac{T_{i,j}^{n+\frac{1}{2}} + T_{i,j}^{n}}{2} \right\} r_i \cdot \Delta r \cdot \Delta z +$$

$$U_v \cdot \Psi \left\{ T_v^{n+\frac{1}{2}} - \frac{T_{i,j}^{n+\frac{1}{2}} + T_{i,j}^{n}}{2} \right\} r_i \cdot \Delta r \cdot \Delta z$$

where superscript n indicates full time steps and i and j indicate radial and axial divisions, respectively.

Canceling identical factors, rearranging and redefining the temporal and spatial divisions by:

$$h_r = \Delta r \quad (32)$$

$$h_z = \Delta z \quad (33)$$

$$\delta = \frac{\Delta\tau}{2} \quad (34)$$

yields:

$$\frac{T_{i,j}^{n+\frac{1}{2}} - T_{i,j}^{n}}{\delta} = \frac{\alpha}{\alpha_b}\left\{\frac{1}{h_r^2}\left[T_{i+1,j}^{n+\frac{1}{2}} - 2T_{i,j}^{n+\frac{1}{2}} + T_{i-1,j}^{n+\frac{1}{2}}\right] + \frac{1}{2r_i h_r}\left[T_{i+1,j}^{n+\frac{1}{2}} - T_{i-1,j}^{n+\frac{1}{2}}\right]\right\} +$$

$$\frac{1}{a^2 h_z^2}\left[T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}\right] + q\cdot\Psi +$$

$$(W + U_a)\cdot\Psi\left[T_a^{n+\frac{1}{2}} - \frac{T_{i,j}^{n+\frac{1}{2}} + T_{i,j}^{n}}{2}\right] + \tag{35}$$

$$U_v\cdot\Psi\left[T_v^{n+\frac{1}{2}} - \frac{T_{i,j}^{n+\frac{1}{2}} + T_{i,j}^{n}}{2}\right]$$

Equation (35) is the general finite-difference, or discretized, equation for the tissue temperatures. This equation may more conveniently be rewritten in the following form:

$$T_{i-1}^{n+\frac{1}{2}}\left\{\frac{\alpha}{\alpha_b}\delta\left[-\frac{1}{h_r^2} + \frac{1}{2r_i h_r}\right]\right\} + T^{n+\frac{1}{2}}\left\{1 + \delta\left[\frac{\alpha}{\alpha_b}\frac{2}{h_r^2} + \frac{\Psi(W + U_a + U_v)}{2}\right]\right\} +$$

$$T_{i+1}^{n+\frac{1}{2}}\left\{\frac{\alpha}{\alpha_b}\delta\left[-\frac{1}{h_r^2} - \frac{1}{2r_i h_r}\right]\right\} = \tag{36}$$

$$T_{j-1}^{n}\left\{\frac{\delta}{a^2 h_z^2}\right\} + T^{n}\left\{1 - \delta\left[\frac{2}{a^2 h_z^2} + \frac{\Psi(W + U_a + U_v)}{2}\right]\right\} + T_{j+1}^{n}\left\{\frac{\delta}{a^2 h_z^2}\right\} +$$

$$\delta\cdot\Psi\left\{q + (W + U_a)T_a^{n+\frac{1}{2}} + U_v T_v^{n+\frac{1}{2}}\right\}$$

For simplicity a certain notation convention is adopted in Equation (36) regarding the spatial indices. Accordingly, whenever a nominal spatial index i or j occurs, it is dropped out from the equation. This leaves only stepped indices to be specifically indicated, e.g., $T_{i,j+1}$ is represented by T_{j+1}, etc.

Equation (36) is evaluated for all nodal points in the cylindrical domain. The process involves substitution of the boundary conditions {equations (14)-(17)}, accounting for the dissimilar nodal spacings at the boundaries between the various tissue compartments in the

radial direction, substitution of zero values for perfusion near the external boundaries, etc. Details of the derivation are presented in Appendix A.

This process yields a set of algebraic equations for all the tissue nodal points. Each equation in this set usually includes three terms for the radial direction and three additional terms for the axial direction. An additional term not containing unknown tissue temperatures, is also included in each equation. At the boundaries in both directions only two terms are present, yielding tri-diagonal matrices for the algebraic set of equations. This property of the set of equations renders it solvable by the Thomas algorithm [11]. To apply this algorithm Equation (36) is now written in matrix notation:

$$\left[I - \delta \cdot A_r \right] \cdot \left\{ T_i^{n+\frac{1}{2}} \right\} = \left[I + \delta \cdot A_z \right] \cdot \left\{ T_j^n \right\} + \delta \cdot \left\{ S^{n+\frac{1}{2}} \right\} \qquad (37)$$

where [I] is a unit matrix, A_r and A_z are the elements of the tri-diagonal matrices in the radial and axial direction, respectively, and $\{S^{n+1/2}\}$ is a one dimensional vector containing all remaining terms in Equation (35) which do not multiply the tissue temperatures. Derivation of these quantities for all nodal points is presented in Appendix A and a summary of all coefficients is given in Tables 1-3.

It is noted that Equation (36) indicates the calculation of the first half time-step only. To complete a full time-step, an equation similar to Equation (36) is required:

$$\left[I - \delta \cdot A_z \right] \cdot \left\{ T_j^{n+1} \right\} = \left[I + \delta \cdot A_r \right] \cdot \left\{ T_i^{n+\frac{1}{2}} \right\} + \delta \cdot \left\{ S^{n+\frac{1}{2}} \right\} \qquad (38)$$

The particular formulation employed in the present analysis assumes that the S-vector in Equations (37) and (38) is evaluated only once per full time-step, i.e., at the one half time-step. It is then maintained constant for the two calculation passes in both radial and axial directions. The S-vector contains the temperatures of the large blood vessels at the axial nodal points which provide the thermal coupling between the tissue and the circulatory system. To calculate these temperatures, a forward-difference approximation for the time derivatives in Equations (12) and (13) is employed. Two additional assumptions are made, regarding the average arterial and venous blood temperatures and average tissue temperature appearing in these equations. First the average temperature of the blood in any one of the two vessels is taken as an arithmetic mean of the two enclosing axial nodal points:

$$\overline{T}_b = \frac{1}{2} \left(T_{b,\,in} + T_{b,\,out} \right) \qquad (39)$$

NODE	$i-1$	i	$i+1$
ALL i FIRST j			
CENTER i REGULAR j	0	$-\frac{\alpha}{\alpha_b}\cdot\frac{2}{h_r^2}-\Psi\frac{W+U_a+U_v}{2}$	$\frac{\alpha}{\alpha_b}\cdot\frac{2}{h_r^2}$
REGULAR & INTERFACE i	$\frac{\alpha}{\alpha_b}\cdot\frac{1}{(1+\gamma)\cdot h_r^{(-)}}\left\{\frac{2\cdot\gamma}{h_r^{(-)}}-\frac{1}{r_l}\right\}$	$-\frac{\alpha}{\alpha_b}\cdot\frac{1}{h_r^{(-)}}\left\{\frac{2\cdot\gamma}{h_r^{(-)}}-\frac{1-\gamma}{r_l}\right\}$ $-\Psi\frac{W+U_a+U_v}{2}$	0
EXTERNAL i REGULAR j	$\frac{\alpha}{\alpha_b}\cdot\frac{2}{h_r^2}-\Psi\frac{W+U_a+U_v}{8}$	$-\frac{\alpha}{\alpha_b}\cdot\left\{\frac{2}{h_r^2}+\left[1+\frac{2}{h_r}\right]\cdot Bl\right\}$ $-\Psi\frac{3\cdot(W+U_a+U_v)}{8}$	$\frac{\alpha}{\alpha_b}\cdot\frac{\gamma^2}{(1+\gamma)\cdot h_r^{(-)}}\left\{\frac{2}{h_r^{(-)}}+\frac{1}{r_l}\right\}$
REGULAR i END j	$\frac{\alpha}{\alpha_b}\cdot\frac{1}{h_r}\cdot\left\{\frac{1}{h_r}-\frac{1}{2\cdot r_l}\right\}$	$-\frac{\alpha}{\alpha_b}\cdot\frac{2}{h_r^2}-\frac{\Psi\cdot W}{2}$	$\frac{\alpha}{\alpha_b}\cdot\frac{1}{h_r}\cdot\left\{\frac{1}{h_r}+\frac{1}{2\cdot r_l}\right\}$
CENTER i END j	0	$-\frac{\alpha}{\alpha_b}\cdot\frac{2}{h_r^2}-\frac{\Psi\cdot W}{2}$	$\frac{\alpha}{\alpha_b}\cdot\frac{2}{h_r^2}$
INTERFACE i END j	$\frac{\alpha}{\alpha_b}\cdot\frac{1}{(1+\gamma)\cdot h_r^{(-)}}\left\{\frac{2\cdot\gamma}{h_r^{(-)}}-\frac{1}{r_l}\right\}$	$-\frac{\alpha}{\alpha_b}\cdot\frac{1}{h_r^{(-)}}\left\{\frac{2\cdot\gamma}{h_r^{(-)}}-\frac{1-\gamma}{r_l}\right\}$ $-\frac{\Psi\cdot W}{2}$	$\frac{\alpha}{\alpha_b}\cdot\frac{\gamma^2}{(1+\gamma)\cdot h_r^{(-)}}\left\{\frac{2}{h_r^{(-)}}+\frac{1}{r_l}\right\}$
EXTERNAL i END j	$\frac{\alpha}{\alpha_b}\cdot\frac{2}{h_r^2}-\frac{\Psi\cdot W}{8}$	$-\frac{\alpha}{\alpha_b}\left\{\frac{2}{h_r^2}+\left[1+\frac{2}{h_r}\right]\cdot Bl\right\}$ $-\frac{3\cdot\Psi\cdot W}{8}$	0

Table 1: Coefficients in the radial-direction $[A_r]$ matrix.

NODE	$j-1$	j	$j+1$
ALL i FIRST j	TEMPERATURES ARE SPECIFIED AT THIS NODE; CALCULATION BEGINS AT $j = 2$.		
CENTER i REGULAR j	$\frac{1}{a^2 \cdot h_z^2}$	$-\frac{2}{a^2 \cdot h_z^2} - \Psi \frac{W + U_a + U_v}{2}$	$\frac{1}{a^2 \cdot h_z^2}$
REGULAR & INTERFACE i REGULAR j			
EXTERNAL i REGULAR j	$\frac{1}{a^2 \cdot h_z^2}$	$-\frac{2}{a^2 \cdot h_z^2} - \Psi \cdot \frac{3(W + U_a + U_v)}{8}$	$\frac{1}{a^2 \cdot h_z^2}$
REGULAR i END j	$\frac{2}{a^2 \cdot h_z^2}$	$-\frac{2}{a^2 \cdot h_z^2}[1 + h_z \cdot Bl_1]$	0
CENTER i END j		$-\frac{\Psi \cdot W}{2}$	
INTERFACE i END j			
EXTERNAL i END j	$\frac{2}{a^2 \cdot h_z^2}$	$-\frac{2}{a^2 \cdot h_z^2}[1 + h_z \cdot Bl_1]$ $-\frac{3 \cdot \Psi \cdot W}{8}$	0

Table 2: Coefficients in the axial-direction $[A_z]$ matrix.

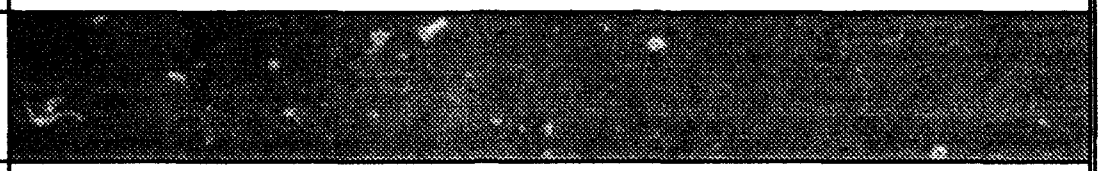
NODE	
ALL i FIRST j	
CENTER i REGULAR j REGULAR & INTERFACE i REGULAR j	$\Psi \cdot \left\{ q + [W + U_a] \cdot T_a^{n+\frac{1}{2}} + U_v \cdot T_v^{n+\frac{1}{2}} \right\}$
EXTERNAL i REGULAR j	$\Psi \cdot \left\{ q + (W + U_a) \cdot [T_a^{n+\frac{1}{2}} - \frac{T_{i-1}}{8}] + U_v \cdot [T_v^{n+\frac{1}{2}} - \frac{T_{i-1}}{8}] \right\} + \frac{\alpha}{\alpha_b} [1 + \frac{2}{h_r}] \cdot Bi \cdot T_0$
REGULAR i END j	$\Psi \cdot \left\{ q + W \cdot T_a^{n+\frac{1}{2}} \right\} + \frac{2 \cdot Bi_1 \cdot T_0}{a^2 \cdot h_z}$
CENTER i END j	
INTERFACE i END j	
EXTERNAL i END j	$\Psi \cdot \left\{ q + W \cdot [T_a^{n+\frac{1}{2}} - \frac{T_{i-1}}{8}] \right\} + \frac{\alpha}{\alpha_b} [1 + \frac{2}{h_r}] \cdot Bi \cdot T_0 + \frac{2 \cdot Bi_1 \cdot T_0}{a^2 \cdot h_z}$

Table 3: Coefficients of the {S} vector.

Similarly, the average tissue temperature for exchanging heat with any of the large blood vessels, is taken as the arithmetic mean of the tissue temperatures at the two enclosing axial nodal points:

$$\overline{T} = \frac{1}{2}\left[T(i,j) + T(i,j-1) \right] \tag{40}$$

With these assumptions the integrals indicated in Equations (12) and (13) are replaced by numerical summations to yield:

$$\overline{T}_a^{n+\frac{1}{2}}(j) = \Delta\tau\left\{ \left[\frac{1}{\Delta\tau} - 2\cdot B_a(j-1) + A_a\cdot SUM1 - Y_a - H_{av} \right] \overline{T}_a^n(j) + \right. \tag{41}$$

$$\left. \left[2\cdot B_a(j-1) - A_a\cdot SUM1 \right] T_a^n(j-1) + \frac{1}{2} Y_a\cdot SUM2 + H_{av}\cdot \overline{T}_v^n(j) \right\}$$

and,

$$\overline{T}_v^{n+\frac{1}{2}}(j) = \Delta\tau\left\{ \left[\frac{1}{\Delta\tau} + 2\cdot B_v(j-1) - 2\cdot A_v\cdot SUM1 - Y_v - H_{va} \right] \overline{T}_v^n(j) + \right.$$

$$\left[-2\cdot B_v(j-1) + A_v\cdot SUM1 \right] T_v^n(j-1) + \frac{1}{2} Y_v\cdot SUM2 + H_{va}\cdot \overline{T}_a^n(j) + \tag{42}$$

$$\left. \frac{1}{2} A_v\cdot SUM3 \right\}$$

where:

$$A_{a\,ax\,v} = \left(\frac{R}{r_{a\,ax\,v}} \right)^2 \tag{43}$$

$$B_{a\,ox\,v} = A_{a\,ox\,v} \frac{\dot{m}_{a\,ox\,v}}{\rho_b \cdot \alpha_b \cdot \pi \cdot \Delta L} \tag{44}$$

$$Y_{a\,ox\,v} = A_{a\,ox\,v} \frac{u_{a\,ox\,v} \cdot R^2}{k_b} = A_{a\,ox\,v}\, U_{a\,ox\,v} \tag{45}$$

$$H_{av\,ox\,va} = A_{a\,ox\,v} \frac{h_{av}}{k_b \cdot \pi \cdot \Delta L} \tag{46}$$

$$SUM1 = \sum_{i=0}^{N-1} W(i, j-1) \cdot (r_{i+1}^2 - r_i^2) \tag{47}$$

$$SUM2 = \sum_{i=0}^{N-1} \{ T(i, j-1) + T(i, j) \} \cdot (r_{i+1}^2 - r_i^2) \tag{48}$$

and,

$$SUM3 = \sum_{i=0}^{N-1} W(i, j-1) \cdot \{ T(i, j-1) + T(i, j) \} \cdot (r_{i+1}^2 - r_i^2) \tag{49}$$

According to the present formulation, the symmetry condition at the centerline of the cylinder, Equation (2), is satisfied by excluding the first one-half division in the radial direction, Fig. A.2. However, in performing the summations indicated in Equations (47)-(49), the contribution of this region is included in order that mass conservation requirements be satisfied.

PHYSICAL AND PHYSIOLOGICAL PARAMETERS

Three interrelated groups of parameters are required for calculating the temperature field in the model. These are: (a) geometrical parameters, depicting the anatomical details

of the modeled organ, (b) thermophysical parameters, representing, primarily, the transport properties of the tissue, and, (c) physiological parameters simulating variables such as blood flow or metabolic heat generation rate.

Accurate and detailed information on these parameters is not available. Moreover, individual variabilities among humans make it almost impossible to formulate a universal set of parameters for the model. Nevertheless, a reasonably accurate set of parameters may be identified for the purpose of studying the behavior of the model.

Table 4 lists the properties used in the model. Data are given for the four compartments, or organs which make up the model, i.e., core, muscle, fat and skin. Additional data are given for blood. Most of the entries in Table 4 were compiled from References [8] and [12]. Blood perfusion and metabolic heat generation rates were estimated as follows. According to Burton [13], the average blood flow in the finger of a subject "who is comfortable as regards the temperature of the surroundings" is in the range of 15-40 cc/min/100 cc tissue. We assumed the lower limit of this range to be representative of the basal blood flow rate in the unperturbed finger. Converted into SI units, and assuming Table 4 value for blood density, this basal value is given by 2.65 kg/m^3 sec. This basal rate was used for calculating the organ-specific values by assuming the geometrical values of Table 4 and accounting for the absence of blood flow in the fat layer.

Also listed in Table 4 are the values for "nutritional" blood flow rates in the various organs of the finger. These values represent the flow rates in a fully constricted finger exposed to a cold environment. Values in the literature for this condition are in the range of 0.3-1.0 cc/min/100 cc tissue [13-15]. For most of this study we assumed a nutritional blood flow value of 0.5 cc/min/100 cc tissue or, 0.0883 kg/m^3 sec. Values for the different organs of the finger are listed in Table 4.

Yet another set of values relates to the basal metabolic heat generation rate in the various organs. These were estimated by assuming that the nutritional blood flow rate is maintained for the purpose of supporting the metabolic activities of the tissues under all conditions. Thus, average oxygen extraction rates may be assumed for estimating the basal metabolic heat generation rates. According to Cooney [16], typical oxygen concentration levels in the blood are 0.195 and 0.145 liter O_2/liter blood for tissue inlet and outlet, respectively. The average caloric equivalent of 1 liter of oxygen may be taken at 20.9 kJ (5 kcal) to yield the basal metabolic heat generation values listed in Table 4.

In listing these values, one adjustment was made in regard to the metabolic rate of the fat layer. Although basal blood flow rate to this organ was assumed to be practically zero, some metabolic activity could still be assumed for this organ. Accordingly, a very low level of 5 W/m^3 was arbitrarily assigned to the fat layer.

	RATIO OF ORGAN TO FINGER RADIUS, R_i/R	THERMAL CONDUC-TIVITY, W/m°C	SPECIFIC HEAT, kJ/kg°C	DENSITY, kg/m^3	BASAL METABOLIC RATE, W/m^3	BASAL BLOOD FLOW RATE, kg/m^{3}sec	NUTRI-TIONAL BLOOD FLOW RATE, kg/m^{3}sec
CORE	0.7057	1.064	2.102	1401	170.5	5.195	0.173
MUSCLE	0.7954	0.418	3.136	1057	631.9	19.225	0.641
FAT	0.8099	0.204	2.520	900	5.0	0	0
SKIN	1.0000	0.293	3.780	1057	247.4	7.526	0.251
BLOOD	------	0.450	3.899	1060	------	------	------

Table 4: Property values used in the numerical computations.

Heat transfer coefficients used to represent the conditions at the surface of the finger are listed in Table 5. Four combinations are considered: bare and gloved finger in either still air (free convection) or windy air (15 km/hr). The values were calculated by standard engineering equations [17] for a 0.08 m long, 0.015 m diameter cylinder. A distinction was made between the cylindrical surface of the finger along its axis versus the spherical-like tip. The glove was represented by a 3-layer ensemble depicting a 2.86 mm (0.09 in) wool layer, 1 mm of still air gap and a 1.27 mm (0.05 in) leather shell. Also shown in this table are the equivalent clo values of the various entries which conform well to the range of values measured on a variety of gloves [18,19].

TESTING OF THE NUMERICAL CODE

A rigorous series of benchmark tests was devised and carried out to verify the stability and convergence of the numerical code written for the model. Programming was done in Turbo Pascal Version 6.0 for IBM-compatible personal computers. Appendix C contains a complete listing of the source code and the operating instructions for the program.

In the first group of tests, all physiological parameters, i.e., q_m, w_b, u_a, u_v and h_{av} were set to zero. This rendered the problem a simple, two dimensional heat transfer problem. In tests #1-3, the heat transfer coefficients on the surfaces of the cylinder, h_1 and h_c were also set to zero thereby creating an adiabatic cylinder except for the base (z=0). In test #1 initial and boundary temperatures at z=0 were set to 30° C and the program was run for 200,000 time steps, 0.1 second each. Throughout the test no temperature changes were observed anywhere in the mesh, as is to be expected. In tests #2 and 3, a change was made in the boundary condition at z=0 after the initial 100 time steps. In test #2, run for 100,00 time steps, 1 second each, the change was from 20° C to 30° C. The inverse change was made in test #3 which was run for a total of 300,000 time steps. In both cases the temperatures anywhere in the mesh approached the boundary temperatures and remained stable.

In test #4 an active heating source was introduced into the cylinder, i.e., $q_m>0$, while still maintaining the other parameters inactive as above. Values used for the heating source were those listed in Table 4 for the basal metabolic heat generation rate. The program was run for a total of 2,000,000 time steps, 0.1 second each. Mesh temperatures have stabilized after 600,000 time steps.

The results of the cylinder model, with an internal heating source, were compared to those calculated by a one-dimensional analytical solution [20]. The comparison between the surface (external) temperatures as calculated by this model and those of the analytical model is shown in Fig. 4. It is noted that precise comparison of these two cases

	$h_c\left[\frac{W}{m^2 \cdot C}\right]$	$h_t\left[\frac{W}{m^2 \cdot C}\right]$
BARE FINGER, STILL AIR	7.02 (0.913 clo)	9.46 (0.672 clo)
BARE FINGER, 15 km/hr WIND	67.2 (0.095 clo)	90.3 (0.071 clo)
GLOVE, STILL AIR	5.17 (1.240 clo)	8.09 (0.792 clo)
GLOVE, 15 km/hr WIND	10.02 (0.640 clo)	13.09 (0.490 clo)

Table 5: Heat transfer coefficients for combinations of wind conditions for a bare and gloved finger.

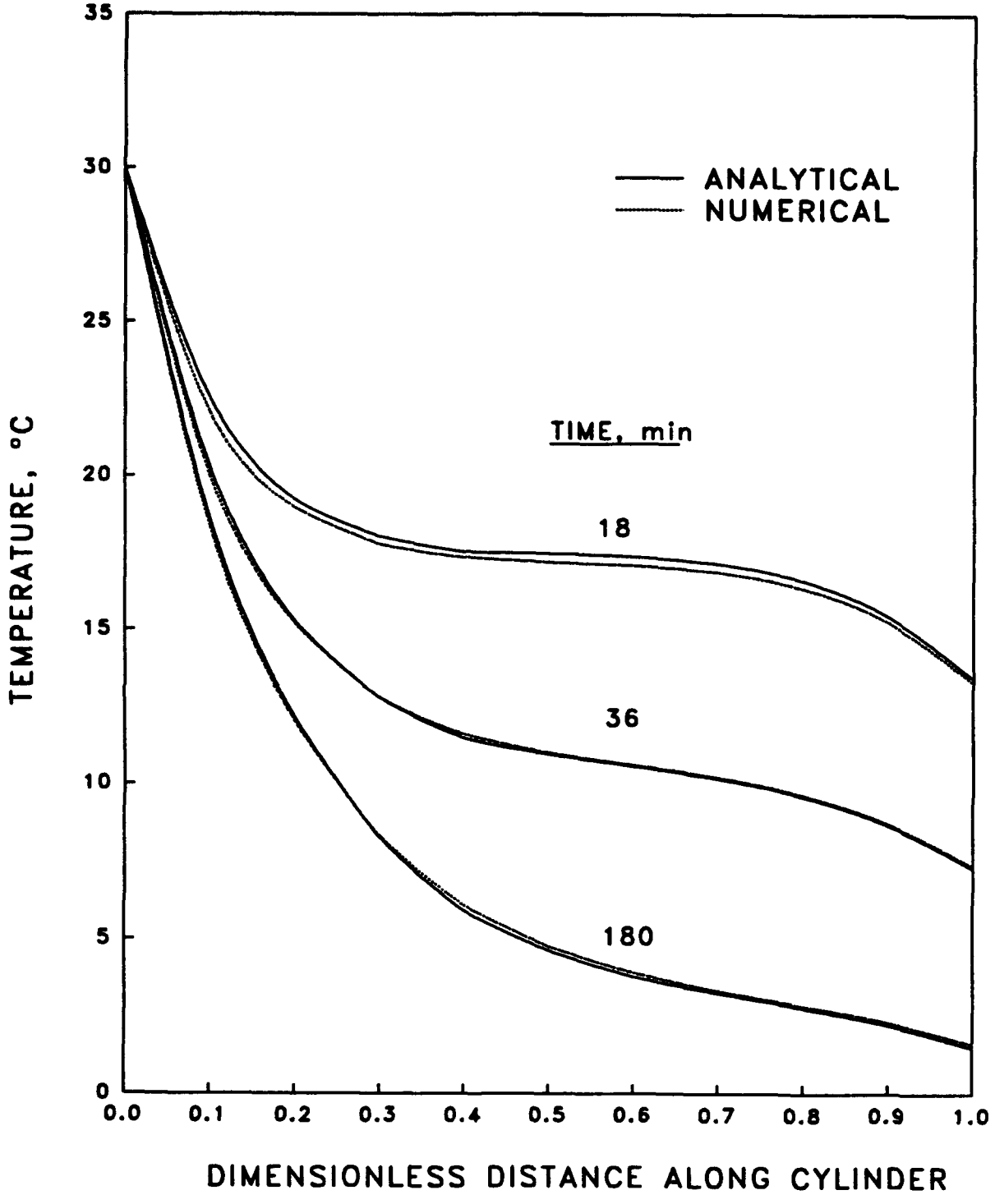


Figure 4: Comparison of temperature distributions along the finger model calculated by an analytical model [20] and the present model.

is not possible since no radial temperature variations are included in the analytical model. Nevertheless, the two sets of plotted results seem to agree well, with the numerical results slightly under predicting the analytical ones at the shorter times into the run.

In test #5, q_m was reset to zero and h_1 and h_c, the heat transfer coefficients with the environment, were set to a very high value of 700 W/m^2 °C. Initial mesh temperatures were set at 30° C and the environmental temperature was maintained at 26° C. After 100 time steps, the boundary temperature at z=0 was reset to 20° C and the program was run for 100,000 time steps, 1 second each. Due to the high value of the heat transfer coefficients used, a rather flat temperature profile of 26° C was established and maintained throughout the mesh except for a short drop to 20° C visualized near the base of the cylinder, as is to be expected.

In test #6, blood perfusion was activated at the basal values listed in Table 4. Other parameters were maintained inactive. Initial mesh and arterial temperatures were set at 20° C. After the initial 1000 time steps, 0.1 second each, both mesh and arterial temperatures were reset to 30° C at z=0. The test was run for a total of 200,000 time steps and mesh temperatures converged on 30° C and remained stable for the duration of the test.

A similar test was run with the addition of heat exchange between the major blood vessels and the mesh points. Results of this test #7 were essentially similar to those of test #6.

In test #8 counter-current heat exchange between the major blood vessels themselves was also activated. Running conditions were identical to those of test #6 except that a total of 1,000,000 time steps were utilized. Mesh temperatures have stabilized at 30° C after 100,000 time steps.

As was to be expected, the execution of the code was sensitive to the size of the time step and the number of spatial divisions used in the numerical code. These two topics are discussed separately. It is firstly noted that the method of alternating directions applied to the solution of the mesh temperatures yields an unconditionally stable scheme of solution [10]. Thus, the source for this sensitivity must reside in Equations (7) and (8) representing the heat balance in the major blood vessels. These two equations are essentially first order ordinary differential equations. Thus, the terms multiplying the independent variables on the right hand-sides may be used to estimate the maximal time step that will ensure stability of the Euler's scheme used to solve them.

In the present analysis this maximal time step is determined by calculating the numerical values of the coefficients of the independent variables in Equations (41) and (42). The larger of the two values, TOTAL, is then substituted into the following equation:

$$\Delta\tau = \frac{0.4}{TOTAL} \qquad (50)$$

to yield the time step which ensures numerical stability. Values obtained by Equation (50) are conservative since a factor of 2 may be used instead of 0.4 [21]. Experience with running of the code proved that this requirement on the time step could, indeed, be relaxed somewhat without adversely affecting the stability of the code.

The sensitivity of the code to the number of divisions used in the numerical network became apparent when an overall steady state heat balance was calculated for the finger model. In all cases studied the number of divisions in the radial direction was kept constant at 12 (Table 6). A relatively simple combination of parameters was used in the computations in which the finger was assumed to be insulated from the environment, no metabolic heat was generated and no counter-current heat exchange between the major blood vessels was allowed. In addition, the thermal conductivities of all tissue compartments were made uniform at the value of the muscle. Under these conditions, the only heat supply to the tissue was due to blood perfusion and the only heat removal mechanism was by conduction at the base of the finger, i.e., at z=0.

Figure 5 shows the ratios calculated for the heat transported by the major blood vessels to the heat conducted away as a function of the number of axial divisions. It is evident that a heat balance is not satisfied for the smaller number of divisions in the axial direction. Only when 20 divisions are used, the heat source essentially equals the heat sink to satisfy a heat balance.

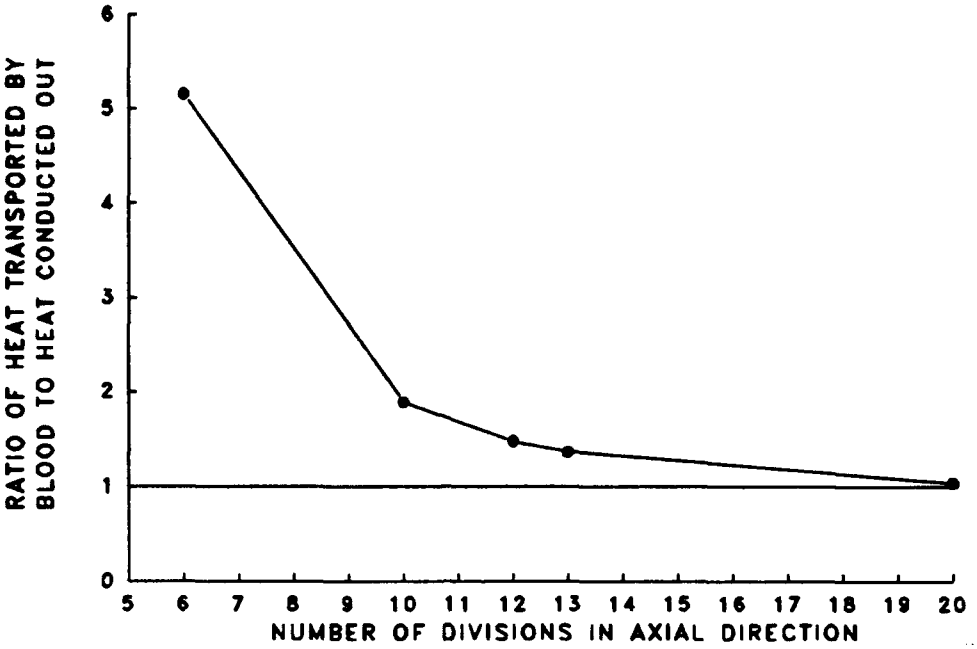
A similar, but more involved, set of benchmark tests was also run. In this set both metabolic heat production and heat exchange with the environment at the finger tip were included in addition to blood perfusion and heat conduction at the finger base. A steady-state energy balance offset of about 32% was initially obtained for 25 axial and 9 radial divisions. This offset gradually dropped to less than 1% when 75 axial divisions were used. This value was deemed quite satisfactory for a steady-state energy balance and served as an additional verification of the numerical code.

Temperature distributions along the external surface of the model are plotted in Fig. 6 also as functions of the number of axial divisions. It is clearly seen that the final temperature obtained is a function of the number of divisions in the axial direction. For the particular case studied here it seems that 20 divisions yield quite an accurate result. This, however, required a much longer running time than for the fewer divisions. Thus, the desired accuracy of the results may have to be determined by considerations such as the total CPU time required for running the program for a given computer.

Based on the series of benchmark tests as outlined here, it may be concluded that the code written for the model is stable and converges to reasonable results for the entire range of parameters considered here.

LENGTH OF CYLINDER, cm				8.0
DIAMETER OF CYLINDER, cm				1.5
DIAMETER OF ARTERY, cm				0.2
DIAMETER OF VEIN, cm				0.3
DISTANCE BETWEEN ARTERY AND VEIN, cm				Eq. (B9)
HEAT TRANSFER COEFFICIENT BETWEEN ARTERY OR VEIN AND THE TISSUE (U_a or U_v)				Eq. (B6)
COUNTERCURRENT HEAT EXCHANGE COEFFICIENT BETWEEN ARTERY AND VEIN (h_a or h_v)				Eq. (B8)
NUMBER OF DIVISIONS IN THE RADIAL DIRECTION				
CORE	MUSCLE	FAT	SKIN	TOTAL
3	3	2	4	12

Table 6: Parameters used in the numerical computations.



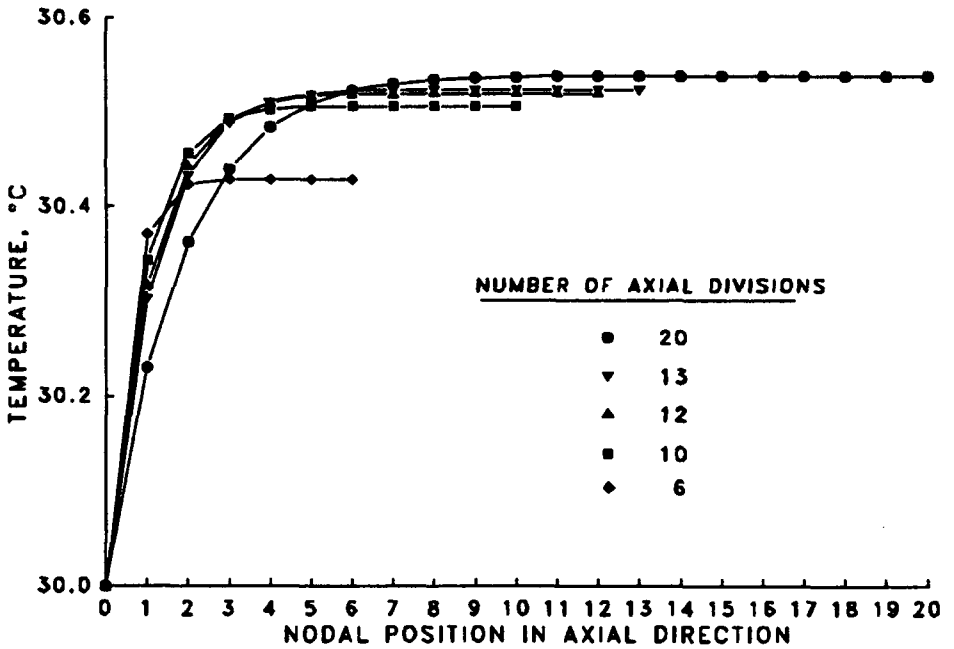


Figure 5 (top): Ratios of steady state heat flow in vs. heat flow out as affected by the number of divisions in the axial direction.

Figure 6 (bottom): Steady state temperature distributions of the external surface of the finger model as affected by the number of divisions in the axial direction.

RESULTS AND DISCUSSION

A number of cases are considered to demonstrate the range of capabilities of this model. Additional parameters used in these demonstrations are listed in Table 6.

Figures 7-10 show the steady state temperature distributions in the finger model for combinations of insulation (bare vs. gloved finger), wind velocities and finger blood flow. In all these cases the environmental temperature was maintained at 0° C, and finger base temperature as well as incoming arterial blood temperature were kept constant at 30° C.

All four figures demonstrate the major role played by blood flow in the thermal economy of the finger. It is clearly noted that rather comfortable temperatures are maintained in the finger for as long as blood flow remains high (at basal level in this case). The exception is the case of the bare finger in a windy environment of 15 km/hr in which the enhanced heat loss offsets much of the beneficial effect of high blood flow to the finger.

In all cases studied, temperature of the distal segments of the finger dropped to very low levels and almost equilibrated with the environment. This is the case even when a two-layer glove is donned on the hand as is also suggested in another study [22]. The only difference among the cases studied here is in the time course of change in these temperatures. This difference is shown in Fig. 11 in which finger tip skin temperatures are plotted vs. time for all 4 cases. Low blood flow, at the nutritional level, which is to be expected for this low environmental temperature, was assumed for all cases.

As seen in Fig. 11, the bare finger in windy air will be the quickest to drop in temperature. It would practically equilibrate with the environment after about 10 minutes. The gloved finger, under the same windy environment, would be much better protected and would require about 60 minutes before it equilibrates with the environment. Interestingly, a bare finger in still air seems to be better protected than a gloved finger in windy air.

These results may also be presented in terms of endurance times, defined as the time for any temperature on the finger to reach 5° C. Accordingly, the endurance times for the bare and gloved fingers in windy air would be about 2.5 and 22 min, respectively. These times would be longer, at 32 and 43 minutes, respectively, when the bare and gloved fingers are exposed to still air.

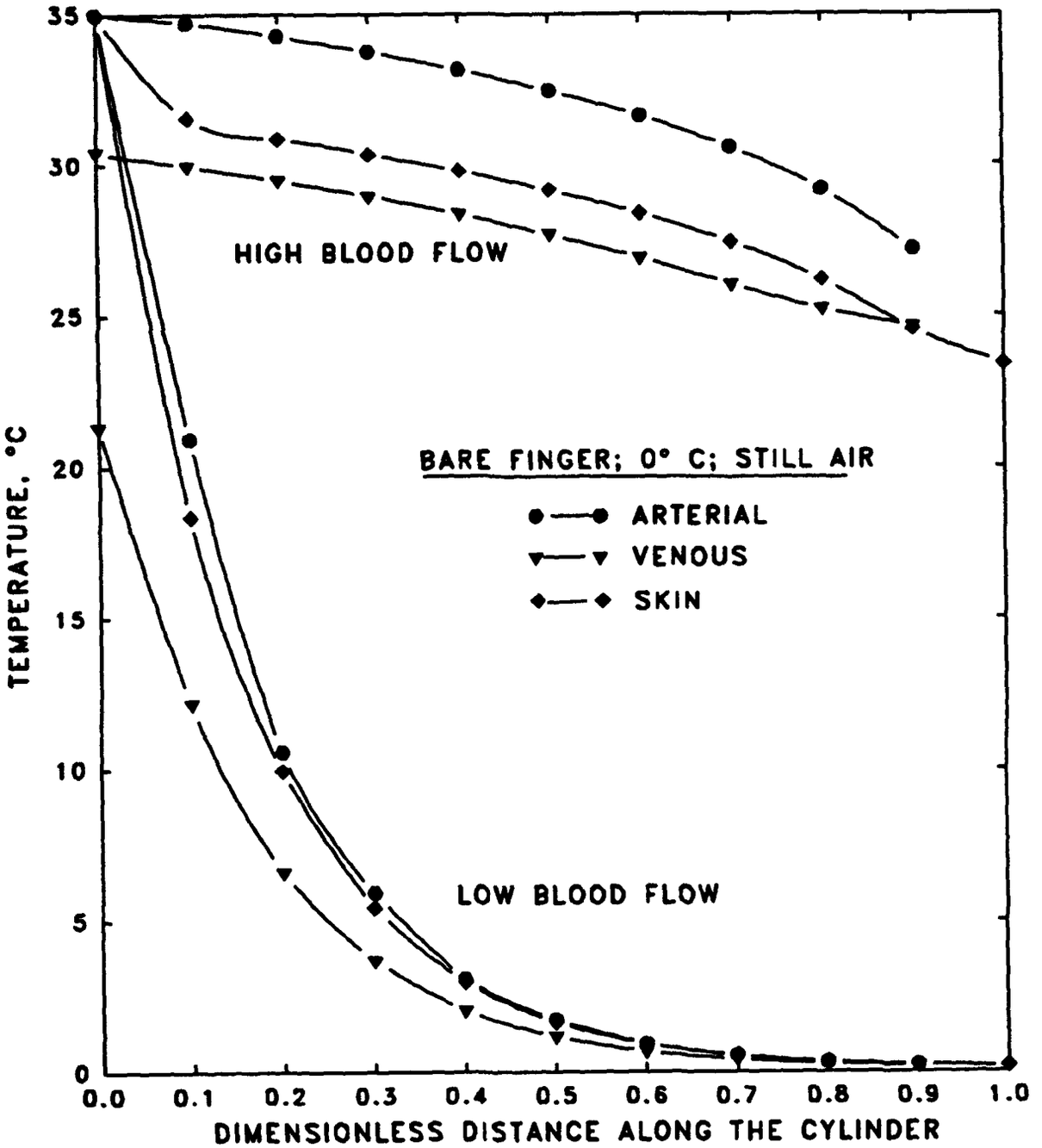


Figure 7: Steady state temperature distributions of a bare finger in still air for basal and nutritional blood flows.

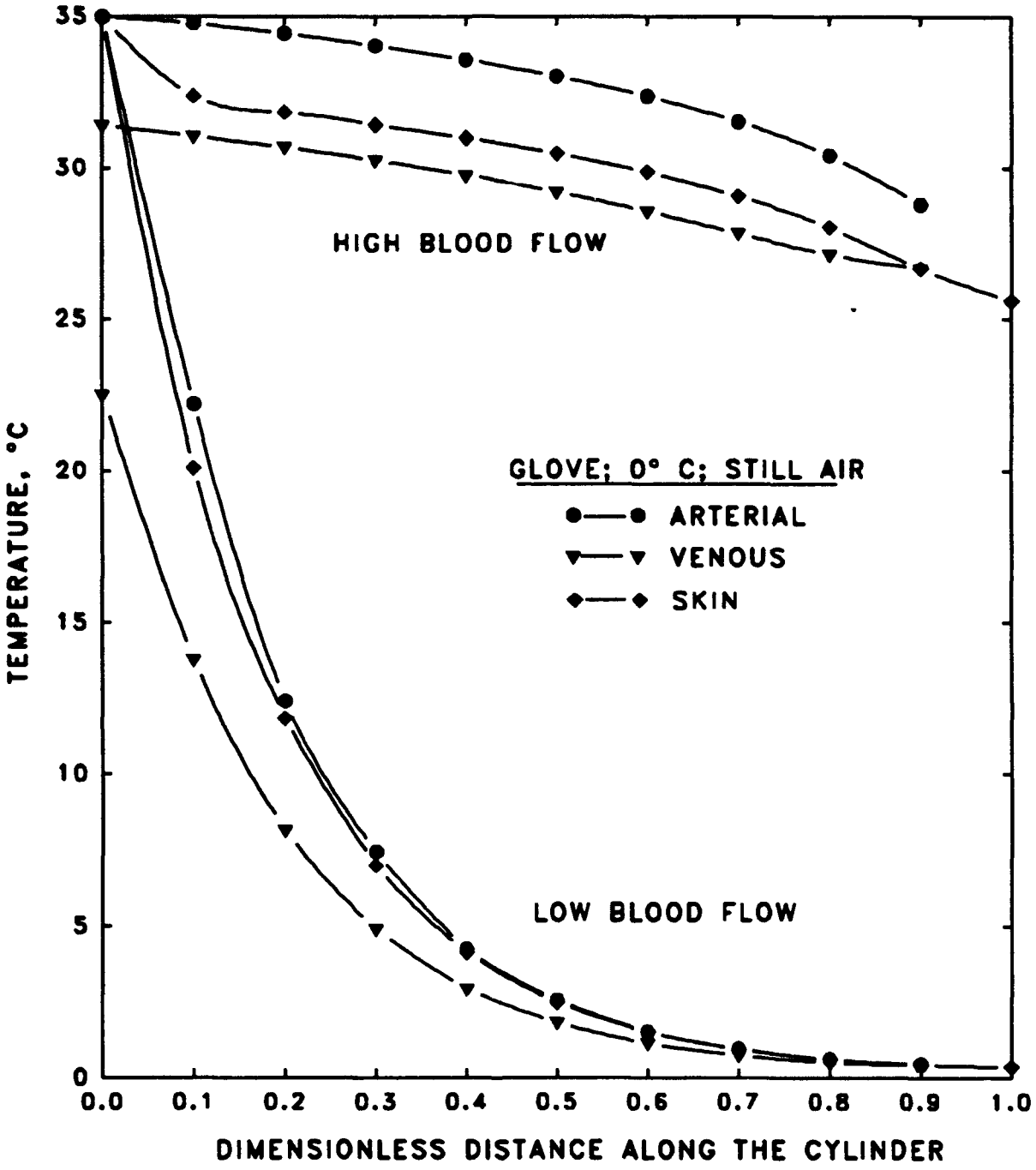


Figure 8: Steady state temperature distributions of a gloved finger in still air for basal and nutritional blood flows.

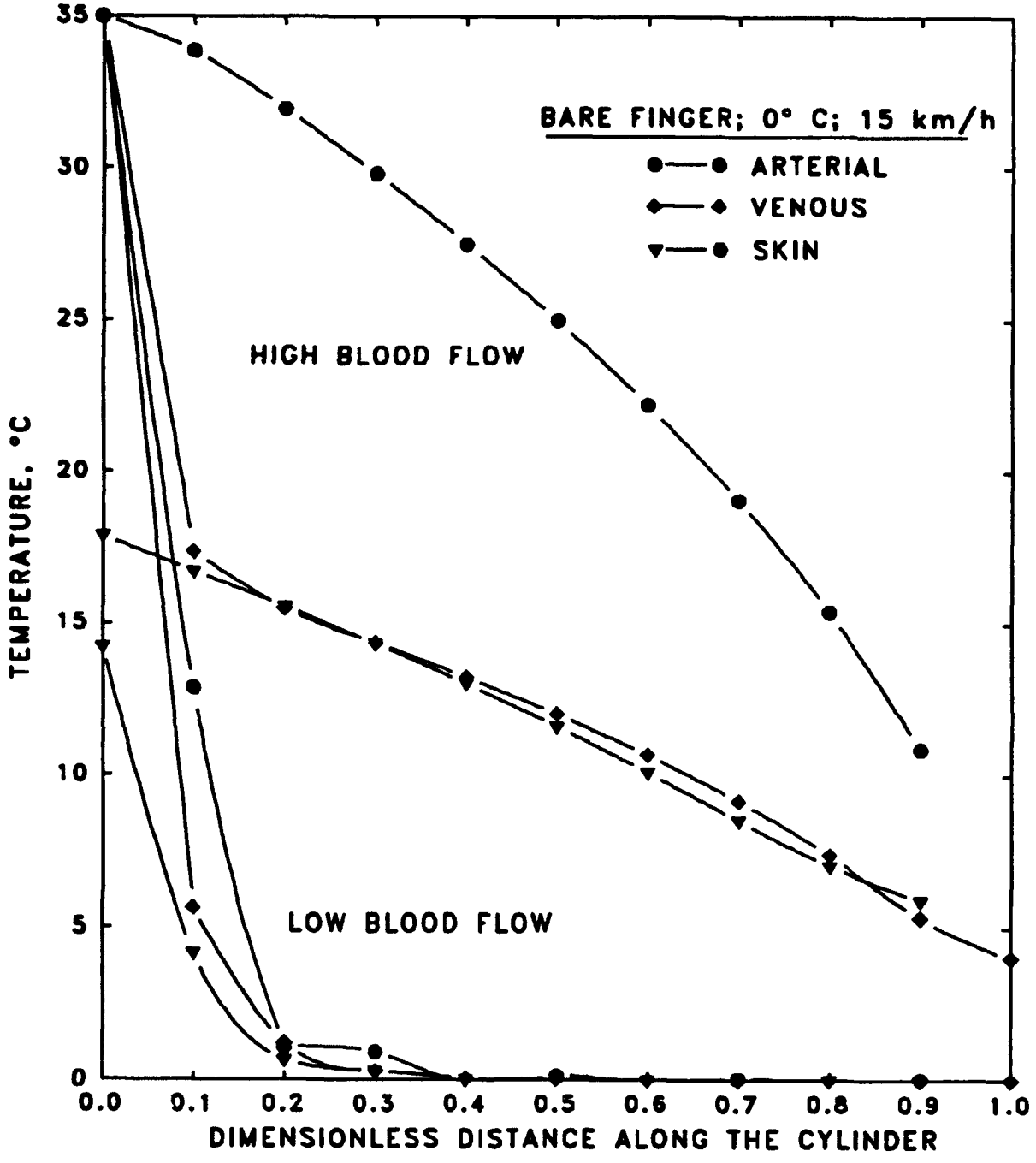


Figure 9: Steady state temperature distributions of a bare finger in windy air for basal and nutritional blood flows.

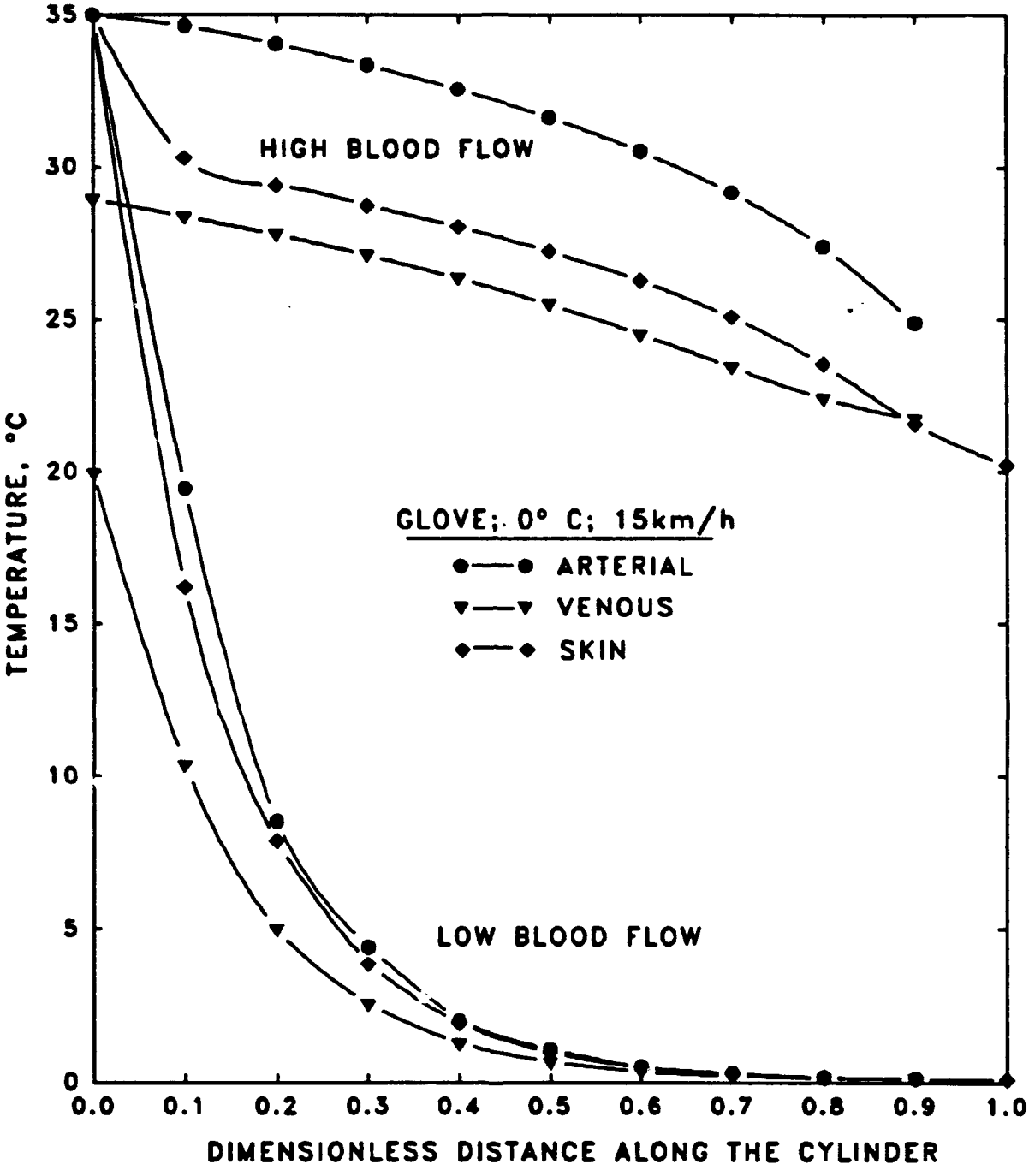


Figure 10: Steady state temperature distributions of a gloved finger in windy air for basal and nutritional blood flows.

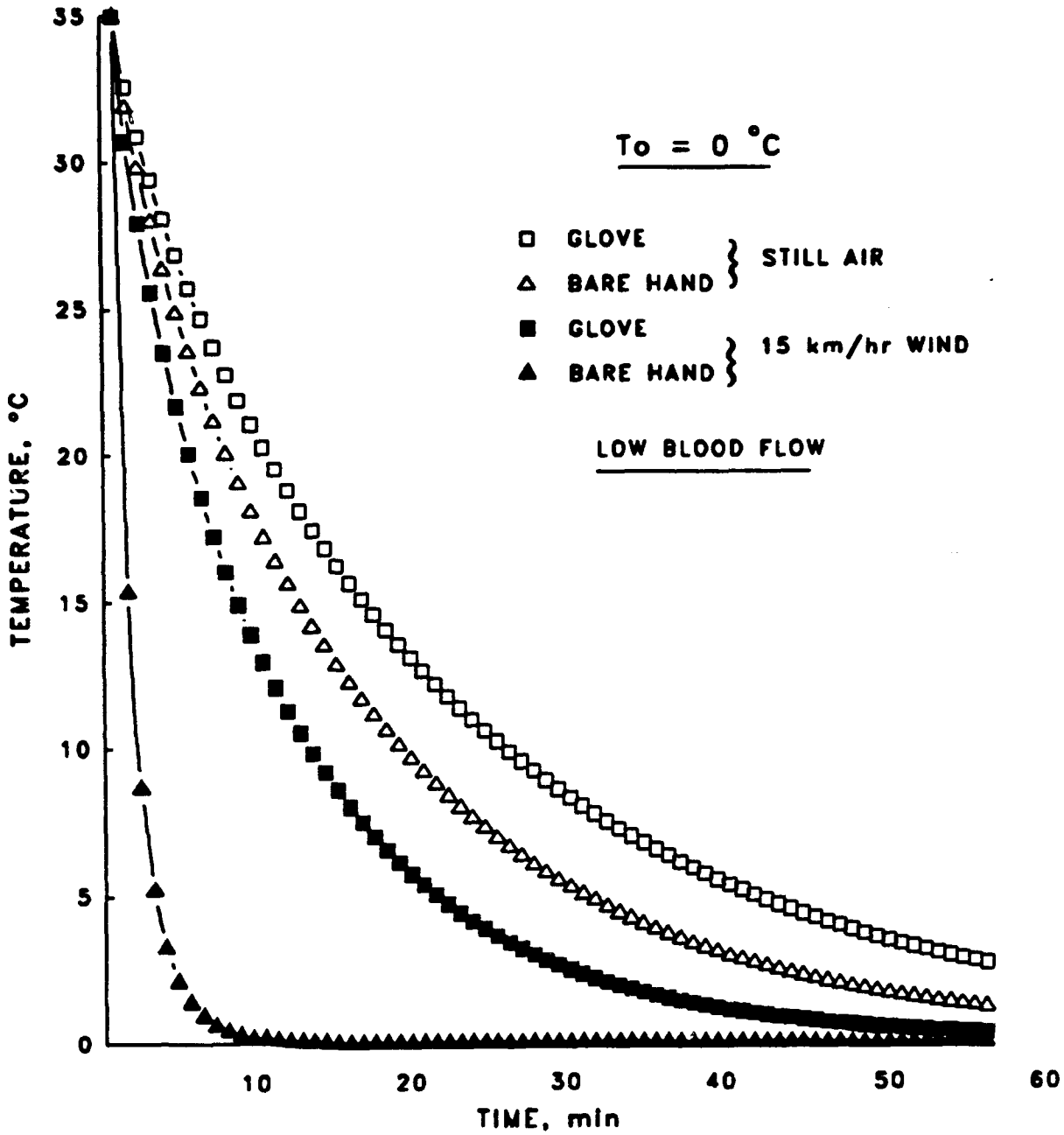


Figure 11: Temperature variations on the dorsal tip of the bare and gloved finger model with nutritional blood flow for still and windy air.

The effect of counter-current heat exchange between the major blood vessels is demonstrated in Fig. 12. Numerical values of the parameters used for this figure are listed in Table 6. The two groups of curves in Fig. 12 show the steady state arterial and venous temperature distributions along the finger with and without counter-current heat exchange between these vessels. As soon as this mechanism is activated, the arterial temperature seems to drop considerably due to the exchange of heat with the cooler vein.

The main purpose of counter-current heat exchange is to conserve body heat in a cold environment. This is effected by firstly depriving the extremity of the rich supply of blood, as is assumed here by dropping blood flow from basal to nutritional level. An additional effect is achieved by lowering the temperature of the extremity through the thermal coupling which exists between the major blood vessels and the tissues. The arterial temperature along the extremity is made to loose heat by counter-current heat exchange to the cooler vein. This, in turn, causes tissue temperatures to decrease as the artery constitutes the main heat source for the extremity. In the case shown in Fig. 12, about 0.093 W is lost to the environment from the finger which decreases to 0.08 W for a reduction of about 14% in finger heat loss when counter-current heat exchange is activated.

Another case of cold induced vasodilatation (CIVD) in the finger is shown in Fig 13. CIVD is known to occur in a percentage of the population and is manifested by rather periodic increases in finger skin surface temperatures, e.g., [23,24]. Although the precise mechanism for this phenomenon is not thoroughly understood, there is ample evidence to indicate that intermittent increases in the otherwise constricted blood flow to the finger cause these temperature changes.

In calculating the data for Fig. 13, it was assumed that the periodic blood flow changes may best be approximated by triangular-shaped surges. These surges were assumed to occur only in and adjacent to the tip of the finger while blood supply to the other segments of the finger remained unchanged. The initial temperature of the entire bare finger, exposed to still air at 0° C, was set at 30° C. At the beginning of the exposure, blood flow in the finger was assumed to drop to the nutritional level (Table 4). This situation was assumed to persist for 20 min. Next, a 3 minute linear increase to 10 times the initial value in blood flow to the tip of the finger was allowed followed by a symmetrical decrease back to the nutritional level. Nutritional blood flow was next maintained for 15 minutes and was followed again by an identical triangular-shaped change in blood flow to the tip of the finger. During the final 15 minutes of the run, blood flow was reset to the nutritional level.

Skin temperature variations are shown in Fig. 13 for three locations along the finger. The curves are plotted for the base, the middle point along the finger and for the tip of the finger. The solid line represents finger tip temperature variations for constant nutritional blood flow without the periodic bouts of CIVD. It thus provides a worst case scenario for comparison purposes. It is clearly seen that increases in blood flow causeincreases in finger temperatures, as is to be expected. These changes are more pronounced at the tip of the finger than at the more proximal locations primarily because blood flow changes due to CIVD are assumed to take place in this area only.

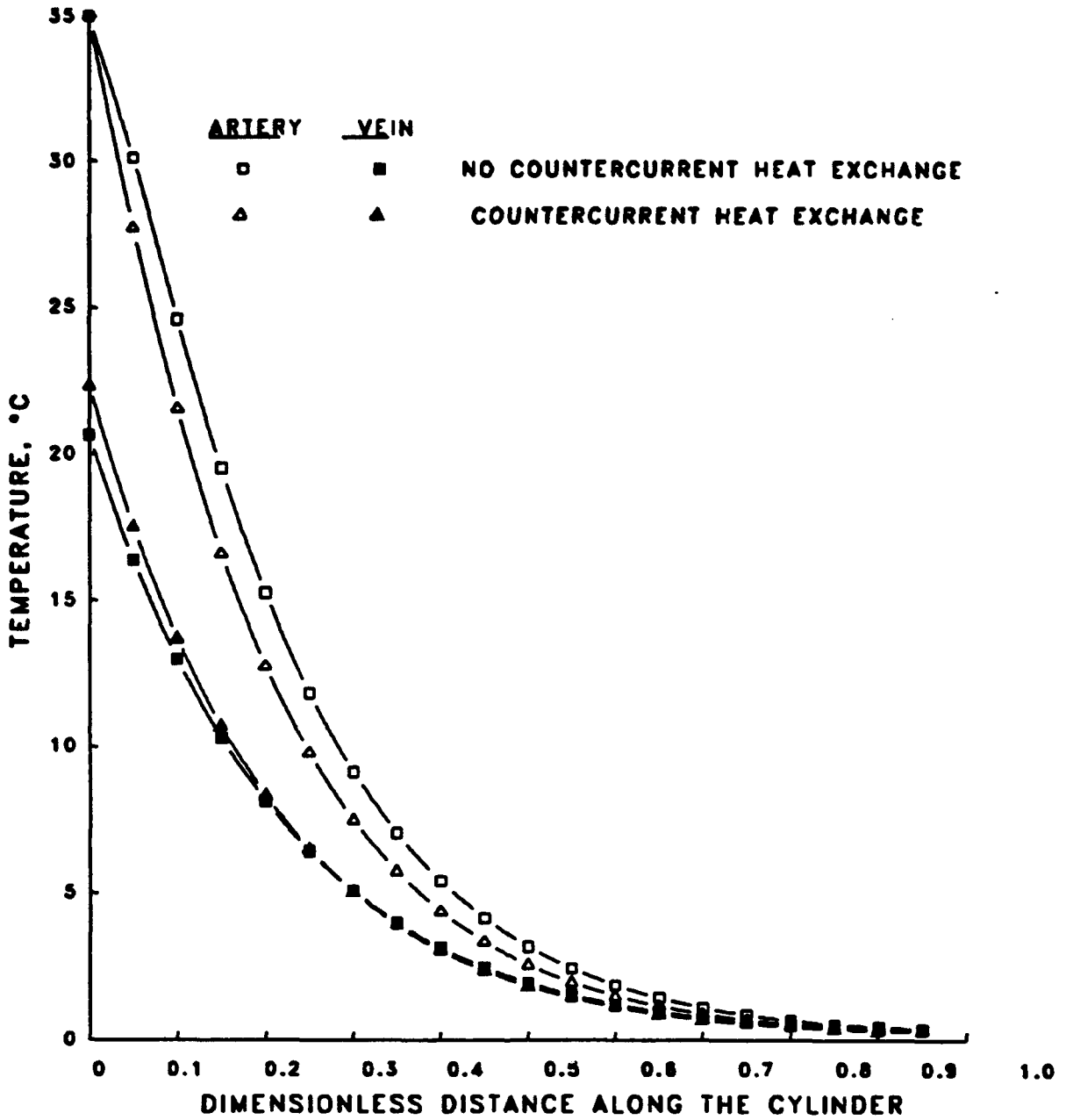


Figure 12: Effect of counter-current heat exchange on arterial and venous temperature distributions.

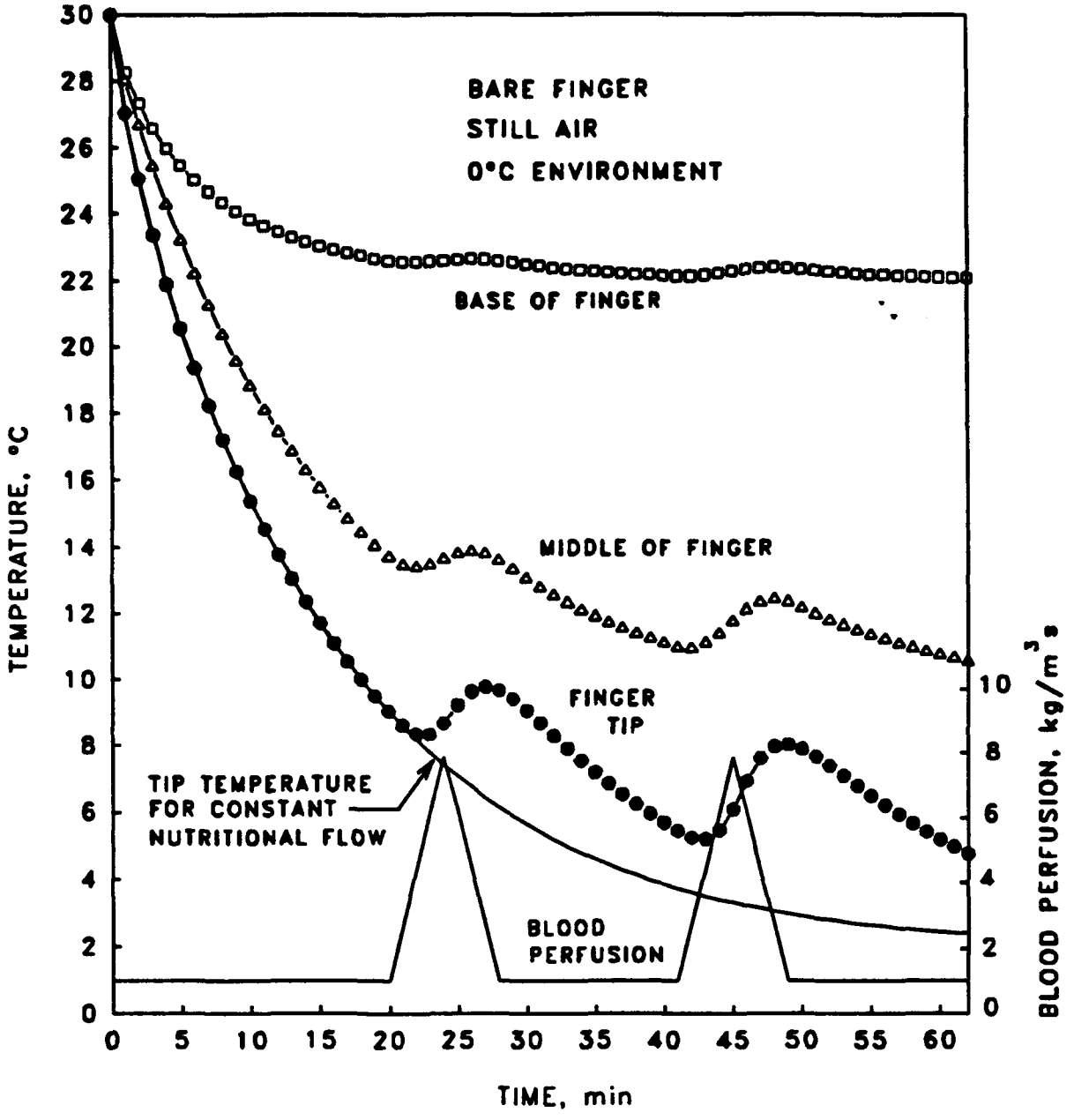


Figure 13: Dorsal temperature variations at the tip and middle of the finger model for cold induced vasodilatation.

Another interesting result relates to the course of change in finger tip temperature for the case shown here. It seems that CIVD, which may be characterized as a heating source, causes the tissue temperatures to increase noticeably for as long as it is active. Once this mechanism gets shut off, tissue temperatures resume the exponential-like decay to levels close to those attained without CIVD. This decay is enhanced by the larger temperature difference between the tissue and the environment that has been established as a result of CIVD. As a matter of fact the temperature difference at the finger tip after one hour between the case involving CIVD and the one without it would be a mere 0.8° C, for the case presented here.

It is recognized that the details shown in Figs. 7-13 are dependent on the parameters and assumptions used in the computations. However, certain trends are clearly indicated which will likely vary in detail and magnitude as the values of these parameters are altered.

REFERENCES

1. Knockel, J.P., Environmental heat illness. An eclectic review, Arch. Intern. Med., 133:841-864, 1974.

2. McCarroll, J.E., Denniston, J.C., Piere, D.R. and Farese, L.J., Behavioral evaluation of a winter warfare training exercise, US Army Research Institute of Environmental Medicine, Report No. T 1/78, 1977.

3. Pennes, H.H., Analysis of tissue and arterial blood temperature in the resting forearm, J. Appl. Physiol., 1:93-122, 1948.

4. Mitchell, J.W., Galvez, T.L., Hengle, J., Myers, G.E. and Siebecker, K.L., Thermal response of human legs during cooling, J. Appl. Physiol., 29:859-865, 1970.

5. Eberhart, R.C., Heat transfer for models in specific organs, in: Heat Transfer in Medicine and Biology: Analysis and Applications, A. Shitzer and R.C. Eberhart (eds.), Plenum Press, N.Y., 1:273-321, 1985.

6. Wissler, E.H., An analysis of factors affecting temperature levels in the nude human, in: Temperature: its Measurement and Control in Science and Industry, J.D. Hardy (ed.), Reinhold, N.Y., Part 3, 603-612, 1963.

7. Stolwijk, J.A.J. and Hardy, J.D., Temperature regulation in man - a theoretical study, Pflugers Arch., 291:129-162, 1966.

8. Arkin, H. and Shitzer, A., Model of thermoregulation in the human body, Reports No. EEC-148 (Part I - the heat transfer model), EEC-149 (Part II - the control model) and EEC-150 (Part III - model behavior and comparison to experimental results of exercising, heat stressed subjects), Energy Engineering Center, Faculty of Mechanical Engineering, Technion, Israel Institute of Technology, Haifa, Israel, 1984.

9. Tikuisis, P., Gonzalez, R.R. and Pandolf, K.B., Thermoregulatory model for immersion of humans in cold water, J. Appl. Physiol., 64(2):719-727, 1988.

10. Ames, W.F., Numerical Methods for Partial Differential Equations, Thomas Nelson and Sons, New York, 1977.

11. Young, D.M., Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.

12. Stolwijk, J.A.J. and Hardy, J.D., Control of body temperature, in: Handbook of Physiology, Ch. 4, 59-60, 1977.

13. Burton, A.C., The range and variability of the blood flow in the human fingers and the vasomotor regulation of body temperature, Am. J. Physiol., 127:437-453, 1939.

14. Freeman, N.E., The effect of temperature on the rate of blood flow in the normal and in the sympathectomized hand, Am. J. Physiol., 113:384-398, 1934.

15. Lotens, W.A., A simple model for foot temperature simulation, TNO-Report 1ZF, 1989.

16. Cooney, D. O., Biomedical Engineering Principles - An Introduction to Fluid, Heat and Mass Transport Phenomena, Marcel Dekker, Inc., New York, 1976.

17. Ozisik, M.N., Heat Transfer - A Basic Approach, McGraw Hill Book Co., N.Y., 1985.

18. United States Army Research Institute of Environmental Medicine, Table of best available data on gloves, unpublished.

19. Santee, W.R., personal communication, 1992.

20. Shitzer, A., Stroschein, L.A., Santee, W.R., Gonzalez, R.R. and Pandolf, K.B., Quantification of conservative endurance times in thermally insulated cold stressed digits, J. Appl. Physiol., 71(6):2528-2535, 1991.

21. Myers, G. E., Analytical Methods in Conduction Heat Transfer, McGraw Hill, New York, 1971.

22. Goldman, R.F., The Arctic soldier: Possible research solutions for his protection, In: Science in Alaska, Proc. of the 15th Alaskan Science Conference, College, Alaska, G. Dahlgren ed., Alaska Division, Assn. for the Advancement of Science, pp. 401-419, 1964.

23. Lewis, T., Observations upon the reactions of the vessels of the human skin to cold, Heart, 15:177-208, 1930.

24. Livingstone, S.D., Changes in cold-induced vasodilatation during Arctic exercises, J. Appl.Physiol., 40(3):455-457, 1976.

25. Elkowitz, A.B., Shitzer, A. and Eberhart, R.C., Transient temperatures in tissues with non-uniform blood flow distributions, ASME Transactions, Journal of Biomechanical Engineering, 104(3), 202-208, 1982.

26. Mitchell, J.W. and Myers, G.E., An analytical model of the countercurrent heat exchange phenomena, Biophysical J., 8:897-911, 1968.

27. Keller, K.H. and Seiler, L., An analysis of peripheral heat transfer in man, J. Appl. Physiol., 30(5):227-234, 1971.

28. Holman, J.P., Heat Transfer, McGraw Hill Book Co., New York, 2nd ed., 1968.

GLOSSARY

VARIABLES

a - dimensionless ratio of cylinder length to radius multiplied by the ratio of blood to tissue thermal diffusivities, Eq. (23).

$A_{(a \text{ or } v)}$ - dimensionless ratio of cylinder to blood vessel radii, Eq. (43).

$[A_r]$ - coefficient matrix in radial direction, Eqs. (37) and (38).

$[A_z]$ - coefficient matrix in axial direction, Eqs. (37) and (38).

$B_{(a \text{ or } v)}$ - dimensionless ratio of blood flow rate to blood mass contained in a vessel element per unit of normalized time, Eq. (44).

Bi - Biot modulus indicating the dimensionless ratio between heat convected by the environment to heat conducted in the cylinder, Eqs. (28) and (29).

c - specific heat, $J \bullet kg^{-1} \bullet °C^{-1}$.

h - heat transfer coefficient between the cylinder and the environment at the circumferential surface, $W \bullet m^{-2} \bullet °C^{-1}$

h_1 - heat transfer coefficient between the cylinder and the environment at the cylinder tip, $W \bullet m^{-2} \bullet °C^{-1}$

$h_{(r \text{ or } z)}$ - dimensionless distance between two adjacent nodes in the radial or axial direction, respectively.

h_{av} - heat transfer coefficient between concomitant artery - vein pairs, $W \bullet °C^{-1}$.

$H_{(av \text{ or } va)}$ - dimensionless heat transfer coefficient between concomitant artery - vein pairs, Eq. (46).

k - thermal conductivity, $W \bullet m^{-1} \bullet °C^{-1}$

L - cylinder length, m.

$m_{(a \text{ or } v)}$ - mass flow rate of arterial or venous blood, $kg \bullet s^{-1}$.

M - total number of nodal points in axial direction.

$M_{(a \text{ or } v)}$ - mass of arterial or venous blood contained in a vessel element, kg.

N - total number of nodal points in radial direction.

q_m - volumetric metabolic heat generation rate, $W \bullet m^{-3}$.

q - dimensionless volumetric heat generation rate, Eq. (25).

r^* - radial coordinate, m.

r - dimensionless radial coordinate, Eq. (19).

R - cylinder radius, m.

$\{S\}$ - vector of terms in matrix equations, Eqs. (37) and (38).

$SUM1$, $SUM2$, $SUM3$ - numerical summations, Eqs. (47) - (49), respectively.

t^* - time, s.

T^* - temperature, °C.

T_1^* - temperature at base node, °C.

T_i^* - initial temperature distribution in the cylinder, °C.

Temp - reference temperature, °C.

T - dimensionless temperature, Eq. (22).

$u_{(a \text{ or } v)}$ - heat transfer coefficient between an artery or a vein, respectively, and the surrounding tissue, $W \bullet m^{-3} \bullet °C^{-1}$.

$U_{(a \text{ or } v)}$ - modified dimensionless heat transfer coefficient between an artery or a vein, respectively, and the surrounding tissue, Eq. (45).

w_b - volumetric blood perfusion rate, $kg \bullet m^{-3} \bullet s^{-1}$.

W - dimensionless volumetric blood perfusion rate, Eq. (26).

$Y_{(a \text{ or } v)}$ - dimensionless heat transfer coefficient between an artery or a vein and the surrounding tissue, Eq. (27).

z^* - axial coordinate, m.

z - dimensionless axial coordinate, Eq. (20).

GREEK LETTERS

α - thermal diffusivity, $m^2 \bullet s^{-1}$.

γ - ratio of radial nodal divisions immediately preceding to immediately following a tissue compartment interface, Eq. (A.10).

δ - dimensionless half a time step, Eq. (34).

ρ - density, $kg \bullet m^{-3}$.

τ - dimensionless time, Eq. (21).

Ψ - dimensionless ratio of blood to tissue thermal inertias, Eq. (24).

SUPERSCRIPTS

* - dimensional quantity.

- - average value.

SUBSCRIPTS

a - arterial.

b - blood.

i - integer (radial direction).

i - initial.

j - integer (axial direction).

n - integer (time).

r - radial.

t - tissue.

v - venous.

z - axial.

0 - environmental.

APPENDIX A - Derivation of the equations for the various matrix nodal points

In this section the elements in the matrix equations, (37) and (38), are derived. Derivation is performed for the different nodal points included in the domain for which the solution is sought, i.e., center nodes, interface nodes, etc. shown in Fig. A.1. The derivation begins with rewriting the general discretized partial differential equation, Equation (35):

$$\frac{T_{i,j}^{n+\frac{1}{2}} - T_{i,j}^{n}}{\delta} = \frac{\alpha}{\alpha_b}\frac{1}{h_r}\left\{T_{i+1,j}^{n+\frac{1}{2}}\left[\frac{1}{h_r} + \frac{1}{2r_i}\right] - T_{i,j}^{n+\frac{1}{2}}\frac{2}{h_r} + T_{i-1,j}^{n+\frac{1}{2}}\left[\frac{1}{h_r} - \frac{1}{2r_i}\right]\right\} +$$

$$\frac{1}{a^2 h_z^2}\left[T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}\right] +$$

$$q\cdot\Psi + (W + U_a)\cdot\Psi\left[T_a^{n+\frac{1}{2}} - T_{av}\right] + U_v\cdot\Psi\left[T_v^{n+\frac{1}{2}} - T_{av}\right] \qquad \text{(A.1)}$$

where T_{av} is a time-average tissue temperature given generally by:

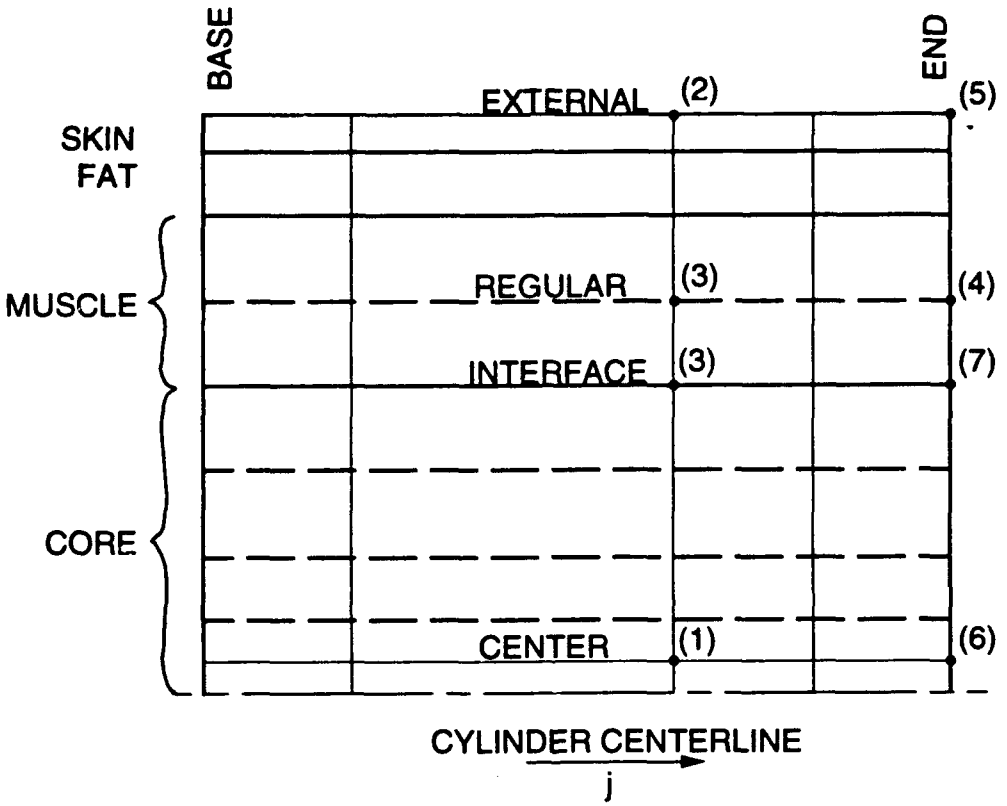
$$T_{av} = \frac{T^{n+\frac{1}{2}} + T}{2} \qquad \text{(A.2)}$$

The specific conditions at the various nodal points are now substituted into Equation (A.1).

Center i-node; regular j-node

In order to satisfy the adiabatic condition specified for this boundary, Equation (14), and to retain a truncation error of the order of $O(h_r^2)$, a central-difference numerical approximation is employed:

$$\frac{\partial T}{\partial r} \approx \frac{T_{i+1,j} - T_{i-1,j}}{2h_r} = 0 \qquad \text{(A.3)}$$



(1) center i; regular j
(2) external i; regular j
(3) regular and interface i; regular j
(4) regular i; end j
(5) external i; end j
(6) center i; end j
(7) interface i; end j

Figure A1: Identification of nodal points in the numerical grid

with geometrical details shown in Fig. A.2. It follows from Equation (A.3):

$$T_{i+1} = T_{i-1} \quad @ \quad r_i = r_0 = 0 \tag{A.4}$$

which yields upon substitution into Equation (A.1):

$$T_{i-1}^{n+\frac{1}{2}} \left\{ 0 \right\} + T^{n+\frac{1}{2}} \left\{ 1 + \frac{\alpha}{\alpha_b} \frac{2\delta}{h_r^2} + \frac{\delta \cdot \Psi (W + U_a + U_v)}{2} \right\} + T_{i+1}^{n+\frac{1}{2}} \left\{ - \frac{\alpha}{\alpha_b} \frac{2\delta}{h_r^2} \right\}$$

$$= T_{j-1} \left\{ \frac{\delta}{a^2 h_z^2} \right\} + T \left\{ 1 - \frac{2\delta}{a^2 h_z^2} - \frac{\delta \cdot \Psi (W + U_a + U_v)}{2} \right\} + T_{j+1} \left\{ \frac{\delta}{a^2 h_z^2} \right\} \tag{A.5}$$

$$+ \delta \cdot \Psi \left\{ q + (W + U_a) T_a^{n+\frac{1}{2}} + U_v T_v^{n+\frac{1}{2}} \right\}$$

External i-node; regular j-node

The boundary condition to be satisfied at these nodes is Equation (15). The numerical approximation to this equation, depicted in Fig. A.3, which retains the desired truncation error, is:

$$\frac{\partial T}{\partial r} \approx \frac{T_{i+1} - T_{i-1}}{2 h_r} = Bi \, (T_0 - T_i) \tag{A.6}$$

which yields:

$$T_{i+1} = T_{i-1} + 2 \cdot h_r \cdot Bi \, (T_0 - T_i) \tag{A.7}$$

Next, a correction is applied to the equivalent tissue temperature at the external node to account for the difference in the heat exchange with capillary perfusion [25]:

$$T_{eq} = \frac{T_{i-1}}{4} + \frac{3}{4} T_i \tag{A.8}$$

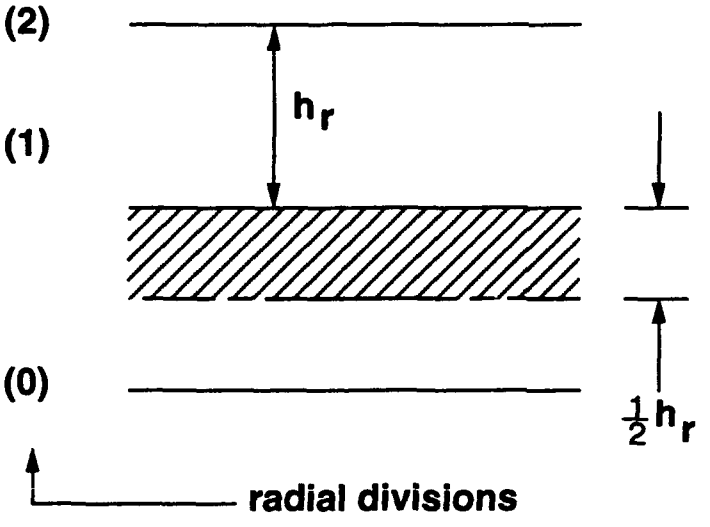


Figure A2: Schematic diagram of the center node

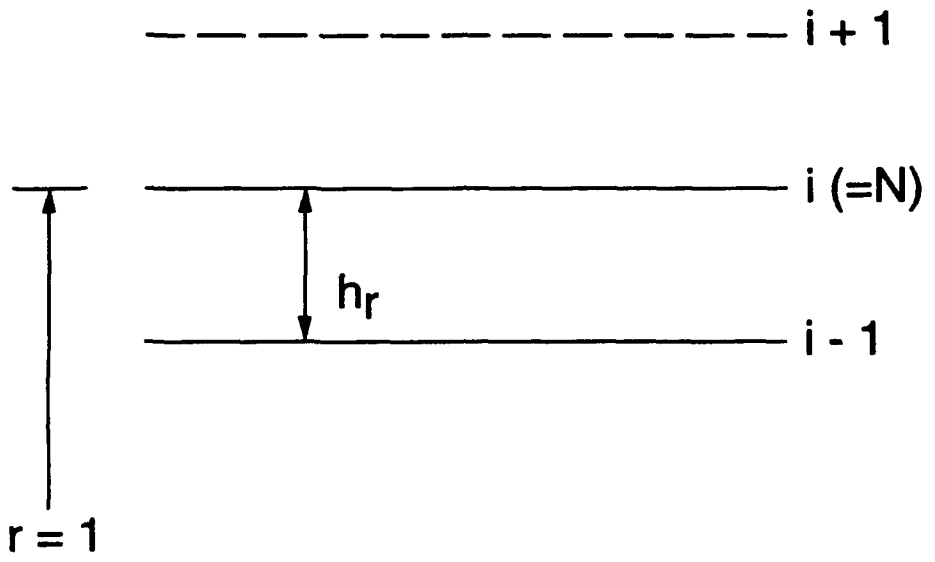


Figure A3: Schematic diagram of the external-regular node

Substitution of Equations (A.7) and (A.8) into Equation (A.1) obtains:

$$T_{i-1}^{n+\frac{1}{2}}\left\{-\frac{2\delta}{h_r^2}\frac{\alpha}{\alpha_b}+\frac{\delta\cdot\Psi\cdot(W+U_a+U_v)}{8}\right\}+$$

$$T^{n+\frac{1}{2}}\left\{1+\frac{\alpha}{\alpha_b}\frac{2\delta}{h_r^2}+\frac{\alpha}{\alpha_b}\delta\left(1+\frac{2}{h_r}\right)Bi+\frac{3\cdot\delta\cdot\Psi\cdot(W+U_a+U_v)}{8}\right\}+T_{i+1}^{n+\frac{1}{2}}\left\{0\right\}=$$

$$T_{j-1}\left\{\frac{\delta}{a^2h_z^2}\right\}+T\left\{1-\frac{2\delta}{a^2h_z^2}-\frac{3\cdot\delta\cdot\Psi\cdot(W+U_a+U_v)}{8}\right\}+T_{j+1}\left\{\frac{\delta}{a^2h_z^2}\right\}+$$

$$\delta\cdot\Psi\left\{q+(W+U_a)\cdot\left[T_a^{n+\frac{1}{2}}-\frac{1}{8}T_{i-1}\right]+U_v\left[T_v^{n+\frac{1}{2}}-\frac{1}{8}T_{i-1}\right]\right\}+$$

$$\frac{\alpha}{\alpha_b}\delta\left(1+\frac{2}{h_r}\right)T_0\cdot Bi \qquad \textbf{(A.9)}$$

Interface i-node; regular j-node

(including a regular i-node; regular j-node by setting γ=1)

Interface nodes define the boundaries between the various compartments, or organs, of the limb model, e.g., core-muscle interface, etc. Numerically the subdivisions in each of the compartments may be different depending on the extent of any specific compartment and the desired numerical details. Additionally, tissue properties generally vary among compartments which further justifies the definition of these interfaces. Figure A.4 depicts the interface region between two adjacent tissue compartments in the radial direction. The ratio of the sub-division immediately preceding the interface to that following it, is defined by:

$$\gamma = \frac{h_r^-}{h_r^+} \qquad \textbf{(A.10)}$$

for simplicity, the superscripts in Equation (A.10) are omitted with the convention:

$$h_r \equiv h_r^- = r_i - r_{i-1} \qquad \textbf{(A.11)}$$

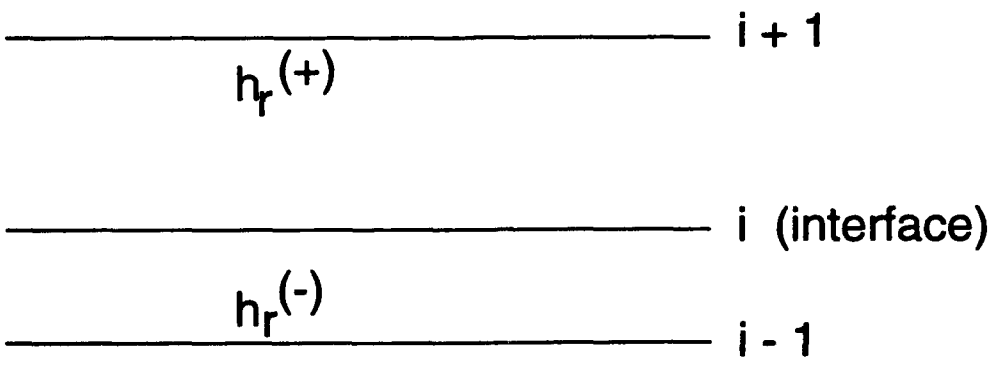


Figure A4: Schematic diagram of the interface node

The following numerical approximations are used in order to retain a truncation error of order $O(h_r^2)$ [10]:

$$\frac{\partial^2 T}{\partial r^2} = \frac{1}{h_r^2}\left\{ \frac{2\gamma}{1+\gamma} T_{i-1} - 2\gamma T + \frac{2\gamma^2}{1+\gamma} T_{i+1} \right\} + O(h_r^2) \qquad \text{(A.12)}$$

$$\frac{\partial T}{\partial r} = \frac{1}{2h_r}\left\{ \frac{-2}{1+\gamma} T_{i-1} + 2(1-\gamma) T + \frac{2\gamma^2}{1+\gamma} T_{i+1} \right\} + O(h_r^2) \qquad \text{(A.13)}$$

After Equations (A.10) - (A.13) are substituted into Equation (A.1), the following expression is obtained:

$$T_{i-1}^{n+\frac{1}{2}}\left\{ -\frac{\alpha}{\alpha_b}\frac{\delta}{h_r(1+\gamma)}\left[\frac{2\gamma}{h_r} - \frac{1}{r_i}\right]\right\} +$$

$$T^{n+\frac{1}{2}}\left\{ 1 + \frac{\alpha}{\alpha_b}\frac{\delta}{h_r}\left[\frac{2\gamma}{h_r} - \frac{1-\gamma}{r_i}\right] + \frac{\delta\cdot\Psi(W+U_a+U_v)}{2}\right\} +$$

$$T_{i+1}^{n+\frac{1}{2}}\left\{ -\frac{\alpha}{\alpha_b}\frac{\delta\cdot\gamma^2}{h_r(1+\gamma)}\left[\frac{2}{h_r} + \frac{1}{r_i}\right]\right\} = \qquad \text{(A.14)}$$

$$T_{j-1}\left\{\frac{\delta}{a^2 h_z^2}\right\} + T\left\{1 - \frac{2\delta}{a^2 h_z^2} - \frac{\delta\cdot\Psi(W+U_a+U_v)}{2}\right\} +$$

$$T_{j+1}\left\{\frac{\delta}{a^2 h_z^2}\right\} + \delta\cdot\Psi\left\{ Q + (W+U_a) T_a^{n+\frac{1}{2}} + U_v T_v^{n+\frac{1}{2}} \right\}$$

End j-node; regular i-node

In the present analysis it is assumed that the major blood vessels terminate (artery) and originate (vein) in the j-node immediately preceding the end j-node, as depicted in Fig. A.5. Thus, heat exchange with the circulatory system is performed at this node with the capillary bed only. The boundary condition at this node is Equation (17) which is approximated by:

$$\frac{\partial T}{\partial z} = \frac{T_{j+1} - T_{j-1}}{2h_z} = Bi_1(T_0 - T_j) \qquad \text{(A.15)}$$

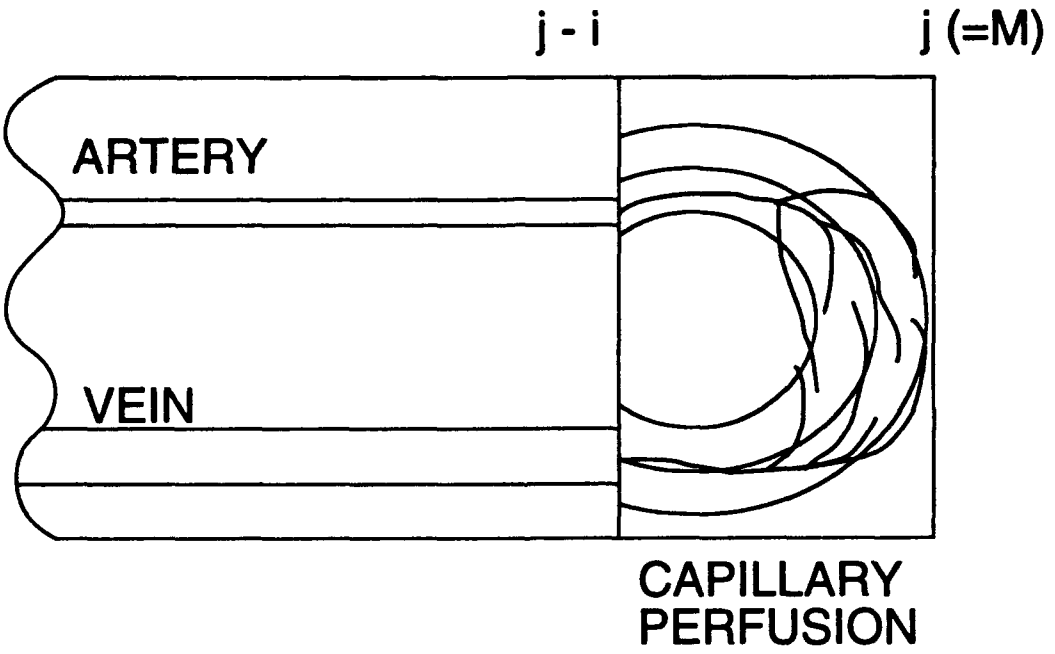


Figure A5: Schematic diagram of the end j - regular i node

which is rewritten as:

$$T_{j+1} = T_{j-1} + 2\,h_z\,Bi_1\,(T_0 - T_j) \qquad \text{(A.16)}$$

yielding, following substitution into Equation (A.1):

$$T_{i-1}^{n+\frac{1}{2}}\left\{-\frac{\alpha}{\alpha_b}\frac{\delta}{h_r}\left[\frac{1}{h_r}-\frac{1}{2\,r_i}\right]\right\} + T^{n+\frac{1}{2}}\left\{1+\frac{\alpha}{\alpha_b}\frac{2\,\delta}{h_r^2}+\frac{\delta\cdot\nabla\cdot W}{2}\right\} +$$

$$T_{i+1}^{n+\frac{1}{2}}\left\{-\frac{\alpha}{\alpha_b}\frac{\delta}{h_r}\left[\frac{1}{h_r}+\frac{1}{2\,r_i}\right]\right\} = \qquad \text{(A.17)}$$

$$T_{j-1}\left\{\frac{2\,\delta}{a^2\,h_z^2}\right\} + T\left\{1-\frac{2\,\delta}{a^2\,h_z^2}\left[1+h_z\,Bi_1\right]-\frac{\delta\cdot\nabla\cdot W}{2}\right\} +$$

$$T_{j+1}\left\{0\right\} + \delta\cdot\nabla\left\{q + W\,T_a^{n+\frac{1}{2}}\right\} + \delta\,\frac{2\,Bi_1\,T_0}{a^2\,h_z}$$

End j-node; external i-node

At this node, which is depicted in Fig. A.6, Equations (15), (17) {or their approximations Equations (A.7) and (A.16)} and the modification given by Equation (A.8) are to be satisfied. Additionally the major blood vessels origination-termination assumption is also applied to yield:

$$T_{i-1}^{n+\frac{1}{2}}\left\{-\frac{\alpha}{\alpha_b}\frac{2\delta}{h_r^2}+\frac{\delta\cdot\Psi\cdot W}{8}\right\}+$$

$$T^{n+\frac{1}{2}}\left\{1+\frac{\alpha}{\alpha_b}\frac{2\delta}{h_r^2}+\frac{\alpha}{\alpha_b}\delta\left[1+\frac{2}{h_r}\right]Bi+\frac{3\cdot\delta\cdot\Psi\cdot W}{8}\right\}+T_{i+1}^{n+\frac{1}{2}}\left\{0\right\}=$$

$$T_{j-1}\left\{\frac{2\delta}{a^2h_z^2}\right\}+T\left\{1-\frac{2\delta}{a^2h_z^2}\left[1+h_zBi_1-\frac{3\cdot\delta\cdot\Psi\cdot W}{8}\right]\right\}+T_{j+1}\left\{0\right\}+ \tag{A.18}$$

$$\delta\cdot\Psi\cdot\left\{q+W\cdot\left[T_a^{n+\frac{1}{2}}-\frac{T_{i-1}}{8}\right]\right\}+\delta\frac{2Bi_1T_0}{a^2h_z}+\frac{\alpha}{\alpha_b}\delta Bi\,T_0\left[1+\frac{2}{h_r}\right]$$

End j-node; center i-node

At this node both Equations (A.4) and (A.16) apply to yield:

$$T_{i-1}^{n+\frac{1}{2}}\left\{0\right\}+T^{n+\frac{1}{2}}\left\{1+\frac{\alpha}{\alpha_b}\frac{2\delta}{h_r^2}+\frac{\delta\cdot\Psi\cdot W}{2}\right\}+T_{i+1}^{n+\frac{1}{2}}\left\{-\frac{\alpha}{\alpha_b}\frac{2\delta}{h_r^2}\right\}=$$

$$T_{j-1}\left\{\frac{2\delta}{a^2h_z^2}\right\}+T\left\{1-\frac{2\delta\,(1+h_zBi_1)}{a^2h_z^2}-\frac{\delta\cdot\Psi\cdot W}{2}\right\}+T_{j+1}\left\{0\right\}+ \tag{A.19}$$

$$\delta\cdot\Psi\left\{q+WT_a^{n+\frac{1}{2}}\right\}+\delta\frac{2Bi_1T_0}{a^2h_z}$$

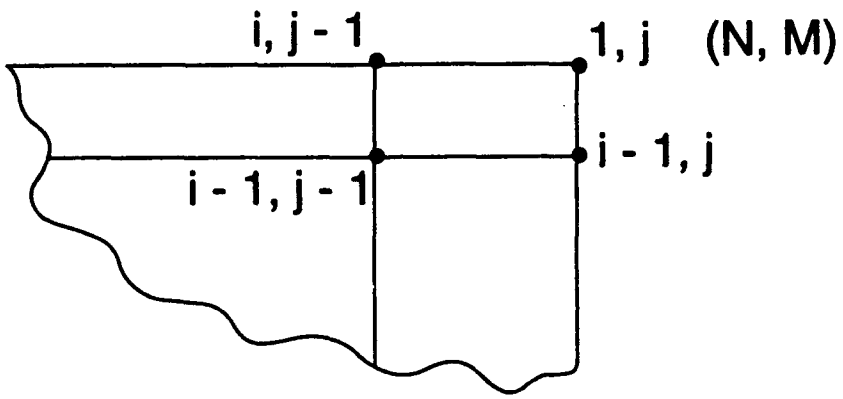


Figure A6: Schematic diagram of the end j - external i node

End j-node; interface i-node

At this node Equations (A.12), (A.13) and (A.16) are substituted to yield:

$$T_{i-1}^{n+\frac{1}{2}}\left\{-\frac{\alpha}{\alpha_b}\frac{\delta}{h_z(1+\gamma)}\left[\frac{2\gamma}{h_z}-\frac{1}{r_i}\right]\right\}+$$

$$T^{n+\frac{1}{2}}\left\{1+\frac{\alpha}{\alpha_b}\frac{\delta}{h_z}\left[\frac{2\gamma}{h_z}-\frac{1-\gamma}{r_i}\right]+\frac{\delta\cdot\Psi\cdot W}{2}\right\}+$$

$$T_{i+1}^{n+\frac{1}{2}}\left\{-\frac{\alpha}{\alpha_b}\frac{\delta\gamma^2}{h_z(1+\gamma)}\left[\frac{2}{h_z}+\frac{1}{r_i}\right]\right\} = \qquad \textbf{(A.20)}$$

$$T_{j-1}\left\{\frac{2\delta}{a^2h_z^2}\right\}+T\left\{1-\frac{2\delta}{a^2h_z^2}\left[1+h_z Bi_1\right]-\frac{\delta\cdot\Psi\cdot W}{2}\right\}+T_{j+1}\left\{0\right\}+$$

$$\delta\cdot\Psi\left\{q+W\cdot T_a^{n+\frac{1}{2}}\right\}+\delta\frac{2\cdot Bi_1\cdot T_0}{a^2h_z}$$

APPENDIX B - Heat transfer coefficients for the major blood vessels

Under the assumptions of the present study the two major blood vessels, an artery and a vein, traverse each body element in the axial direction, Fig. 1. These vessels exchange heat with the surrounding tissue. Additionally under certain circumstances there may also be direct heat exchange between the two vessels. Mitchell and Myers [26] assumed that each one of these major vessels resides alone in the tissue. An improved assumption is due to Keller and Seiler [27] and Arkin and Shitzer [8] according to which an "influence volume" is defined by enclosing each vessel by a larger concentric cylinder, Fig. B.1. The conduction shape factor for this situation is given by [28]

$$SF_1 = \frac{2 \cdot \pi \cdot \Delta L}{\ln\left(\frac{R_c}{r_{a\,or\,v}}\right)} \qquad \text{(B.1)}$$

where R_c is the radius of the enclosing cylinder. As a first approximation this radius may be set as the half distance between the centers of the major blood vessels, D. Thus,

$$R_c = \frac{1}{2} D \qquad \text{(B.2)}$$

The amount of heat exchanged between any of these vessels and the surrounding tissue is now given by

$$q = \frac{2 \cdot \pi \cdot \bar{k} \cdot \Delta L}{\ln\left(\frac{R_c}{r_{a\,or\,v}}\right)} \Delta T^*_{overall} \qquad \text{(B.3)}$$

where k is a volume weighted average value for tissue thermal conductivity given by

$$\bar{k} = \frac{1}{V} \int k \, dV \qquad \text{(B.4)}$$

and

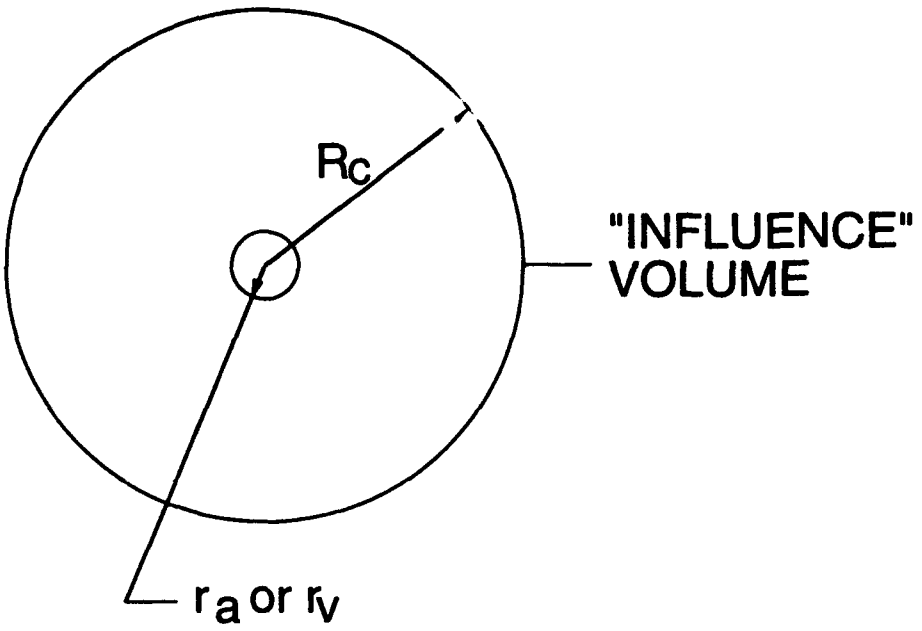


Figure B1: Schematic representation of an "influence" volume enclosing a major blood vessel

$$\Delta T^{*}_{overall} = \overline{T}^{*} - T^{*}_{vessel} \qquad \text{(B.5)}$$

The integral in Equation (B.4) is performed numerically by the trapezoidal rule. The overall temperature difference in Equation (B.5) includes an average value, $\overline{T}$, representing the tissue surrounding the blood vessel. This average tissue temperature is calculated by performing the integrations indicated in Equations (7) and (8). For these integrals a volummetric average heat transfer coefficient, $u_{a\ or\ v}$ is required. This is obtained by "distributing" the overall heat transfer coefficient over the volume of the entire body element to obtain

$$u^{*}_{a\,or\,v} = \frac{2 \cdot \pi \cdot \overline{k} \cdot \Delta L}{ln\left(\frac{R_c}{r_{a\,or\,v}}\right)} \frac{1}{\pi \cdot R^2 \cdot \Delta L} = \frac{2 \cdot \overline{k}}{R^2 \cdot ln\left(\frac{R_c}{r_{a\,or\,v}}\right)} \qquad \text{(B.6)}$$

The direct heat exchange between the artery and the vein may be approximated by assuming a conduction shape factor for two parallel cylinders exchanging heat, the centers of which are spaced by D [28], Fig. B2:

$$SF_2 = \frac{2 \cdot \pi \cdot \Delta L}{\cosh^{-1}\left(\frac{D^2 - r_a^2 - r_v^2}{2 \cdot r_a \cdot r_v}\right)} \qquad \text{(B.7)}$$

The overall heat transfer coefficient between these cylinders is given by

$$h_{av} = \frac{2 \cdot \pi \cdot \overline{k} \cdot \Delta L}{\cosh^{-1}\left(\frac{D^2 - r_a^2 - r_v^2}{2 \cdot r_a \cdot r_v}\right)} \qquad \text{(B.8)}$$

The distance between the centers of the two major blood vessels is usually known, or can be inferred, for any body element. For the purposes of the present computations we assume this distance to be given by four times the geometric average of the radii of these vessels [8]

$$D = 4 \cdot (r_a \cdot r_v)^{\frac{1}{2}} \qquad \text{(B.9)}$$

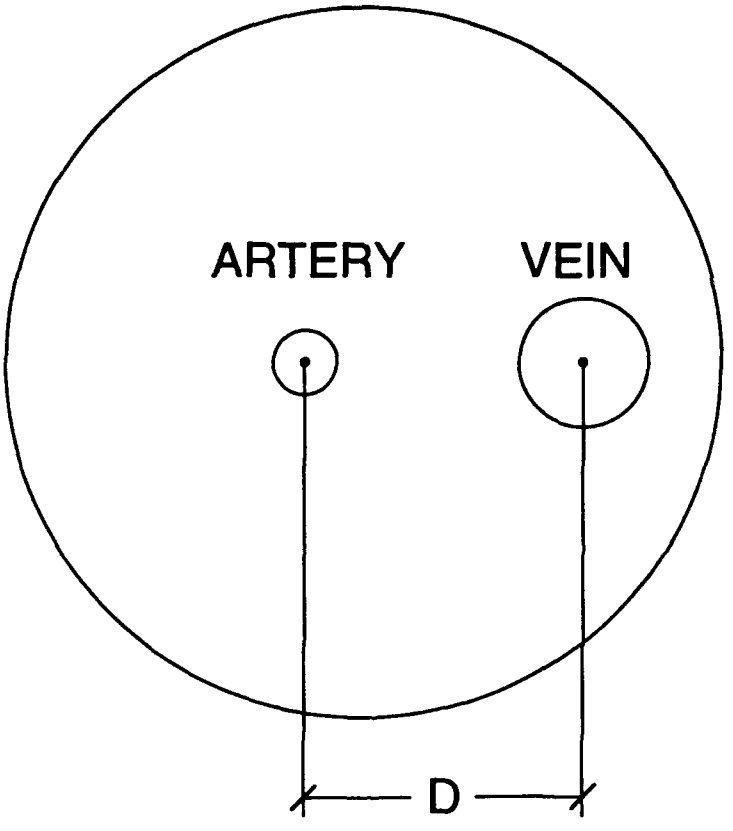


Figure B2: Schematic diagram of an artery-vein pair

which yields upon substitution into Equation (B.8)

$$h_{av} = \frac{2 \cdot \pi \cdot \bar{k} \cdot \Delta L}{\cosh^{-1}\left(7 - \frac{r_a^2 + r_v^2}{2 \cdot r_a \cdot r_v}\right)} \qquad \text{(B.8)}$$

APPENDIX C - Program source code listing and operating instructions

Program Environment

1. Hardware requirements : 386 or 486 PC with a 80X87 math coprocessor.

2. **TURBO PASCAL 6.0** installed and operating.

3. **BGI directory** loaded in the TP directory enabling the graphics interface to operate.

4. Option Switches -

   A. **COMPILE** menu bar - **DESTINATION** set to **disk.**
   B. **OPTION** menu bar - **COMPILER** set to the **8087** processing mode.
   MEMORY SIZE - STACK SIZE set to **65520.**

The program consists of four files:

1. **FINGER93.PAS** is the main body of the program.

2. **VAR_SP.PAS** is a Turbo Pascal UNIT containing all global declarations.

3. **TISUE_SP.PAS** is a Turbo Pascal UNIT containing the procedures necessary to generate the coefficients for the tissue equations.

4. **PHYS_FIN.PAS** is a Turbo Pascal UNIT containing all procedures dealing with anatomical data.

The system of Pascal programs utilizing a main program which accesses three UNITS was necessary since files in TURBO PASCAL 6.0 cannot exceed 65K bytes.

N.B. - all units must be re-compiled to disk each time changes are made to the code under consideration. Turbo Pascal requires that a *.TPU file exist on disk for each UNIT in the program. Consequently, the units must be recompiled each time the user makes changes to these files. Compiling these files with the destination switch set to memory will not create the necessary *.TPU files for the program to execute successfully. THE UNITS MUST BE COMPILED

Description of program Units

UNIT FINGER93.PAS { main program }

1. **Procedure Initialize;**
   Generates formula calculations for all values required by the program. Data for this procedure is acquired from the CONST section of UNIT VAR_SP.

2. **Procedure Data_Dump;**
   Generates file output of initial and calculated data that wail be used by the program.

UNIT VAR_SP.PAS

This unit contains all **CONST**ant declarations, data **TYPE**s and **VAR**iable declarations used by the program. For testing purposes, the initial values assigned in the **CONST**ant section may be altered. This area has been identified by the internal documentation. No changes should be made in the **TYPE** sections or the **VAR** sections since these will have a substantial impact on program execution. **VAR_SP** must be saved and recompiled to disk each time the test values are changed. This unit contains six sub-modules which function as utilities of calculation procedures for the generation of coefficients required by the tissue equations.

1. **FUNCTION GAMMA**
   Calculates the gamma values as required.

2. **PROCEDURE PAUSE_CRT**
   A utility that allows the user to halt the programs execution as required.

3. **PROCEDURE DERIVE_CENTER_TEMP**
   Derives the temperature at I = 0.

4. **PROCEDURE DUMP_HALF_ARRAYS**
   Outputs the all values of the temperature arrays at the half time step.

5. **PROCEDURE DUMP_T_ARRAYS**
   Outputs the all values of the temperature arrays.

6. **PROCEDURE CALC_TA_TV_HALF**
   Calculates the values for arterial and venous temperatures. Variables TA_BAR, TV_BAR, TA_NODE, and TV_NODE are calculated here in preparation for the calculation of the S_VECTOR values.

UNIT TISUE_SP.PAS

1. **Procedure Axial_Dir;**
   Imports the I and J coordinates, identifies the type of node using enumerated data types and calculates three coefficients for the given I,J location. These are stored in the ORIG_A_Z_COEF array which is used for tissue traversal in the Radial Direction to generate the constant term prior to accessing the code for Thomas' Algorithm.

2. **Procedure Rad_Dir;**
   Generates coefficients utilizing the same approach described above. The coefficients are stored in array ORIG_A_R_COEF and are used in the axial traversal of the tissue matrix to generate the constant term prior to accessing the code for Thomas' Algorithm.

3. **Procedure Establish_coef_arrays;**
   Driver block containing nested loops which call procedures 1 and 2 listed above, thereby loading the coefficient arrays.

4. **Procedure Traverse;**
   Contains the following nested procedures:

   a. **Procedure Thomas;**
      Contains code for Thomas Algorithm.

   b. **Procedure First_Traverse_Radial;**
      Moves appropriate coefficients from arrays generated earlier into variables needed by Procedure Thomas. This procedure then calls the Thomas algorithm procedure and places the new temperatures into the tissue temperature matrix. A single dimension array is used as an intermediary data structure in order to hold the I,J coordinates and corresponding temperatures.

   c. **Procedure Second_Traverse_Axial;**
      Organized in the same manner as the First_Traverse procedure. This procedure traverses the temperature matrix in the Axial direction.

   d. **Main Block of Procedure Traverse;**
      Driver block for establishing the temperature matrix and nested looping structure for the number of time steps and print intervals.

UNIT PHYS_SP.PAS

This unit contains the following five procedures:

1. **PROCEDURE Anatomical_Input;**

2. **PROCEDURE Num_Divisions;**
   These procedures load variables with the necessary anatomical data.

3. **PROCEDURE Width_Calculation;**
   Determines the width of each organ, H- values and gamma values.

4. **PROCEDURE Est_Interfaces;**
   Establishes the location of the organ interfaces and stores them in a data structure of type SET.

5. **PROCEDURE Create_Arrays;**
   Establishes a set of values for each organ and a series of arrays containing R_i and Hr- values indexed to each I location.

OUTPUT FILES

The following two external output files are automatically created when the program is executed.

1. **D_VALUES.DAT**
   File contains a listing of all significant values used during the current program run. Some of these are echo prints of values entered in the CONST section of VAR_SP.PAS, while others are the result of formula calculations.

2. **R_TEMPS.DAT**
   This file contains a printout of the time step, arterial, venous, and tissue temperatures generated of at each time step, at user determined intervals during program execution.

OVERVIEW OF PROGRAM OPERATION

1. Prior to running the program, be certain that the following programming conditions have been established :

   A. Turbo Pascal has been properly installed with all the BGI files in the BGI directory within the TP directory.

   B. COMPILE menu bar DESTINATION has been set to DISK.

   C. OPTIONS menu bar has been u[illegible] set the compiler to the 8087 numeric processor.

2. Changing program parameters:

   A. *Accessible Parameters* are found in the VAR_SP.PAS program unit. These have been separated from the rest of the code by the following sets of comments:

```
(*************************************************************************)
(******************** BEGIN CONTROL POINT SECTION ************************)
(*************************************************************************)
```

```
(*************************************************************************)
(******************** END CONTROL POINT SECTION **************************)
(*********** NOTHING SHOULD BE ALTERED BELOW THIS POINT **********)
(*************************************************************************)
```

B. The parameters within the CONTROL POINT SECTION include:

GC_DELTA_SEC { delta value in seconds }

GC_LENGTH and **GC_DIAMETER** { in centimeters }

Radial Divisions

Axial Divisions

TA, TV, TISSUE_TEMP, ENVIRONMENTAL_TEMP, and **NORMALIZING_TEMP** { in Degrees C.}

BASAL METABOLIC RATES

BASAL BLOOD FLOW RATES

VARIABLES IN OTHER SECTIONS OF THE CODE MUST NOT BE ALTERED!

---

3. When changes are made to any of the above listed parameters, the program must be re-compiled to disk before execution in order to generate the appropriate *.TPU files.

N.B. - Turbo Pascal imposes a 65K constraint with respect to variable memory utilization. A compiler error will occur if the radial and axial division entries generate an excessively large tissue matrix.

{ ERROR 96 - TOO MANY VARIABLES }

To avoid this problem it is recommended that the following limitations be observed :

TOTAL AXIAL DIVISIONS	<= 12
TOTAL RADIAL DIVISIONS	<= 10

4. **DELTA VALUE** - The access point for delta is through variable GC_DELTA_SEC which is usually represented as a fraction of a second. The program uses GC_DELTA_SEC to calculate a USER DETERMINED normalized delta value for program execution. The code also calculates a PROGRAM DETERMINED normalized delta value which is the MAXIMUM value for delta based on the existing program parameters. The USER DEFINED value must be less than or equal to the PROGRAM DETERMINED maximum delta value for the program to function effectively. However, if time constraints are important, the USER DEFINED delta value may slightly exceed the program determined maximum delta. The numerical integrity of the program results are directly influenced by the size of the USER DEFINED delta value. There will be an obvious breakdown in the numeric output if the USER DEFINED delta value is too large.

5. **APPROXIMATE RUNNING TIME :**

Hardware:	486 PC running at 33 MHz
Axial Divisions:	10
Radial Divisions:	13
Running Time:	20,000 time steps takes approximately 18 minutes.

Running the Program

1. Be sure that the TURBO PASCAL 6.0 environment has been properly configured.

   a. **Compiler** and **Option** switches have been properly set. (see Program Environment.)

   b. **FINGER93.PAS, VAR_SP.PAS, PHYS_SP.PAS, TISUE.PAS** files currently reside in the TP directory.

   c. **FINGER93.PAS, VAR_SP.PAS, PHYS_SP.PAS, TISUE.PAS** files have been opened on the TURBO PASCAL DESKTOP.

2. If any of the CONTROL POINT parameters are to be changed, use the mouse to double click on the VAR_SP.PAS to bring the program text into the editor.

   a. Change values in the CONTROL POINT SECTION.

   b. Compile VAR_SP.PAS to disk.

   c. If no compilation errors exist, double click on VAR_SP.PAS to exit and return to the desktop.

3. Single click on FINGER93.PAS.

4. Click RUN on the menu bar and click RUN on the pull down menu OR press <CTRL> F9 to execute the program.

5. Press <RETURN> at the title screen.

6. Check the USER DETERMINED DELTA value against the PROGRAM DETERMINED MAXIMUM DELTA value.

   a. The value for GC_DELTA_SEC which was entered in the CONTROL POINT section of UNIT VAR_SP.PAS is used to calculate the delta value which will be accessed by the program. The program also determines a maximum acceptable delta value under the existing parameters. It is essential that the USER DETERMINED DELTA value does not exceed the PROGRAM DETERMINED MAXIMUM DELTA value.

   b. At the time step prompt enter a number 1.

   c. At the interval prompt enter a number 1.

d. USER DETERMINED DELTA and the PROGRAM DETERMINED MAXIMUM DELTA values will appear on the screen. Be sure that the USER DETERMINED DELTA is less than or equal to the PROGRAM DETERMINED MAXIMUM DELTA value.

e. After checking the relationship between the USER DETERMINED DELTA and the PROGRAM DETERMINED DELTA follow the prompts at the bottom of the screen pressing <RETURN> until program execution terminates.

f. If the USER DETERMINED DELTA is greater than the PROGRAM DETERMINED DELTA, repeat steps 2 and 3 using a smaller GC_DELTA_SEC value until the correct relationship is obtain between the USER DETERMINED DELTA and the PROGRAM DETERMINED MAXIMUM DELTA values. **DO NOT proceed to STEP 7 unless a valid DELTA value has been obtained.**

6. Click on program FINGER93.PAS and then press <CTRL> F9 to begin executing the program for the desired time interval.

   a. At the time step prompt, enter the desired number of total times steps for the program run. TOTAL TIME STEPS = total clock minutes * 60 / GC_DELTA_SEC

   b. At the interval prompt, enter the time step interval at which to display data on the screen and send data to output file R_TEMPS.DAT

      INTERVAL = interval minutes * 60 / GC_DELTA_SEC

   c. **GC_DELTA_SEC** = 0.08
      total clock minutes = 62 minutes
      print interval = 1 minute
      TOTAL TIME STEPS = 46500
      INTERVAL = 750

7. When the program begins execution, a time step counter appears on the screen. Temperature output to both screen and file R_TEMPS.DAT will occur at the time step INTERVAL established in step 6. Follow the prompts at the bottom of the screen to continue execution or to view the graph of the important temperatures generated at the established INTERVAL.

8. Upon program completion, two files will have been created in the TP directory:

   a. File D_VALUES.DAT contains the CONTROL POINT values used in program run as well as formula calculations.

   b. File R_TEMPS.DAT contains the information that was printed to the screen at the established INTERVALS during the program run.

   c. To view these files on the screen use the MS-DOS TYPE command.

PROGRAM SOURCE CODE LISTINGS

1. FINGER93.PAS

```
{.....................................................................

  program description : A numerical solution for the thermal
                        exchange of extremities in a cold environment
                        based on the model developed by Dr. Avraham Shitzer.

  written for :  USARIEM, Natick


  written by :  Paul Vital

  language   :  PASCAL 6.0

  WORKING COPY  --- FINGER v3.0 - AUGUST 27, 1993
 .....................................................................}

PROGRAM FINGER_93 (INPUT,OUTPUT);

USES
  var_sp, phys_sp, tisue_sp, DOS, CRT, PRINTER;


PROCEDURE DESTINATION;

      {.......... determines output channel printer/crt..................}

  BEGIN
    CLRSCR;
    ASSIGNCRT (F);
    REWRITE (F);
    ASSIGN(F2,'D_VALUES.DAT');
    REWRITE(F2);
    ASSIGN (OUT_DATA, 'R_TEMPS.DAT');
    REWRITE (OUT_DATA);
   END;


PROCEDURE DATA_DUMP;

      {...prints axial and radial coefficients as well as results of
                        all formula initializations.....................}

  VAR
    INDEX,I,J : INTEGER;
    H_TOTAL : REAL;

  BEGIN   (* data_dump *)
    CLRSCR;
    WRITE ('DUMP DATA [Y, N] ? ');
    IF UPCASE (READKEY) = 'Y' THEN
      BEGIN
        WRITELN (F2, 'DATA DUMP' : 47);
        WRITELN (F2);
        WRITE (F2, 'PHYSICAL LENGTH   = ');
        WRITELN (F2,LEN : 10:7, 'meters' : 10);
        WRITELN (F2, 'DIAMETER = ':20, DIAM : 10:7,'meters' : 10);
        WRITELN (F2);
        WRITELN (F2, 'Radii of artery and vein   { units = meters }');
        WRITELN (F2, 'R_A = ':10,R_A:7:4);
        WRITELN (F2, 'R_V = ':10,R_V:7:4);
        WRITELN (F2);
        WRITELN(F2);
        WRITELN (F2,'INITIAL VALUES ---------------------------------');
```

```
WRITELN (F2);
WRITELN (F2, 'TA = ':25, TA :3);
WRITELN (F2, 'TV = ':25, TV :3);
WRITELN (F2, 'Tissue temperature = ':25, mesh_TEMP:3);
WRITELN (F2);
WRITELN (F2, 'Environmental temperature = ', t_0 * normalizing_temp:4:2);
WRITELN (F2, 'W values   { units = kg. blood / cu.m tissue / sec }');
WRITELN (F2, 'CORE':28, Core_blood_meters :10:3);
WRITELN (F2, 'MUSCLE':28, mus_blood_meters :10:3);
WRITELN (F2, 'FAT' :28, fat_blood_meters:10:3);
WRITELN (F2, 'SKIN' :28, skin_blood_meters :10:3);
WRITELN (F2);
WRITELN (F2, 'organ radius to finger ratio    { R_i / R }');
WRITELN (F2, 'CORE RATIO':28, core_ratio :10:4);
WRITELN (F2, 'MUSCLE RATIO':28, muscle_ratio :10:4);
WRITELN (F2, 'FAT RATIO':28, fat_ratio:10:4);
WRITELN (F2, 'SKIN RATIO':28, skin_ratio:10:4);
WRITELN (F2);
WRITELN (F2,'thermal conductivity    { units = Watt / m / deg.C }');
WRITELN (F2, 'CORE THERM CON':28, core_therm_con :10:3);
WRITELN (F2, 'MUSCLE THER M CON':28, mus_therm_con :10:3);
WRITELN (F2, 'FAT THERM CON':28, fat_therm_con :10:3);
WRITELN (F2, 'SKIN THERM CON':28, skin_therm_con :10:3);
WRITELN (F2, 'BLOOD THERM CON':28, blood_therm_con :10:3);
WRITELN (F2);
WRITELN (F2, 'heat capacity  { units = J / kg / deg. C }');
WRITELN (F2,  'CORE HEAT CAP':28, core_heat_cap : 10:2);
WRITELN (F2, 'MUSCLE HEAT CAP':28,mus_heat_cap :10:2);
WRITELN (F2, 'FAT HEAT CAP':28, fat_heat_cap:10:2);
WRITELN (F2, 'SKIN HEAT CAP':28, skin_heat_cap :10:2);
WRITELN (F2, 'BLOOD HEAT CAP':28, blood_heat_cap :10:2);
WRITELN (F2);
WRITELN (F2,'density   { units = kg / cu. m }');
WRITELN (F2, 'CORE DENSITY':28, core_density :10:2);
WRITELN (F2, 'MUSCLE DENSITY':28, mus_density :10:2);
WRITELN (F2, 'FAT DENSITY':28, fat_density :10:2);
WRITELN (F2, 'SKIN DENSITY':28, skin_density :10:2);
WRITELN (F2, 'BLOOD DENISTY':28, blood_density :10:2);
WRITELN (F2);
WRITELN (F2, 'M = ',M);
WRITELN (F2, 'N = ',N);
WRITELN (F2);
WRITELN (F2, 'CORE_MUSCLE INTERFACE EXISTS AT I VALUE OF ', I_CORE_MUS);
WRITELN (F2);
WRITELN (F2, 'MUSCLE-FAT INTERFACE EXISTS AT I VALUE OF ',I_MUS_FAT);
WRITELN (F2);
WRITELN (F2, 'FAT-SKIN INTERFACE EXISTS AT I VALUE OF ',I_FAT_SKIN);
WRITELN (F2);
WRITELN (F2, 'SKIN SURFACE EXISTS AT I VALUE OF ', SKIN_SURFACE);
WRITELN (F2);

(* DUMP FORMULA CALCULATIONS *)

WRITELN (F2, 'FORMULA CALCULATIONS':50);
WRITELN (F2);
WRITELN (F2, 'DELTA_sec = ',DELTA_sec:13:10, ' secs');
WRITELN (F2, 'DELTA = ',DELTA:13:10, ' normalized');
WRITELN (F2,'PROGRAM CALCULATED MAXIMUM DELTA VALUE = ', MAX_DELTA : 20:15);
WRITELN (F2,'USER DETERMINED delta = ',DELTA:20:15);
WRITELN (F2);
WRITELN (F2,'A_a  = ', a_a:20:10);
WRITELN (F2,'A_v  = ', a_v:20:10);
WRITELN (F2);
WRITE (F2, '****** subscript locations are one greater than');
WRITELN (F2,' tissue location *****');
WRITELN (F2);
FOR I := 1 TO N DO
 BEGIN
    WRITE (F2, 'Q_ARRAY_PHYS[',I,'] = ',Q_ARRAY_PHYS[I]:15:10);
    WRITELN (F2, 'q_array_nrml [':20,i,' ] = ',q_array_nrml[i]:15:10);
 END;
WRITELN (F2);
```

```
        WRITELN (F2, 'U_a_phys = ',U_a_phys :12:5,'U_a = ':10,U_a :12:5);
        WRITELN (F2,  'U_v_phys = ',U_v_phys :12:5,'U_v = ':10,U_v :12:5);
        WRITELN (F2);
        WRITELN (F2, 'K_AVG = ', K_AVG : 20:15);
        WRITELN (F2);
        WRITELN (F2, 'H_A_V_PHYS = ', H_A_V_PHYS : 20:15);
        WRITELN (F2, 'H_a_v = ', h_a_v : 12:5);
        WRITELN (F2, 'H_v_a = ', H_v_a : 12:5);
        WRITELN (F2);
        FOR I := 1 TO N DO
          BEGIN
            WRITE (F2,'w_b_nrml [',I,'] = ',w_b_nrml [ I ]:10:6);
            WRITELN (F2,'w_array [':20, I, '] = ',w_array [ I ]:10:6);
          END;
        WRITELN(F2);
        WRITELN (F2,'capillary loss data :');
        WRITELN (F2,'     capillary sum = ',capil_sum:20:10);
        WRITELN (F2,'     capillary loss = ', capillary_loss:20:10);
        WRITELN (F2,'     starting M_A = ', start_m_a:20:10);
        WRITELN (F2);
        FOR J := 1 TO M DO
          BEGIN
            WRITE (F2, 'm_a [',J,'] = ',m_a[J]:20:14);
            WRITELN (F2, 'm_v [':20,J,' ] = ',m_v[J]:20:14);
          END;
        WRITELN (F2);

        WRITELN (F2,'w_b normalizing_val =  ',W_b_normalizing_val : 20:14);
        WRITELN (F2);
        WRITELN (F2, 'H_1 = ', H_1: 10:4);
        WRITELN (F2, 'H_c = ', H_c: 10:4);
        WRITELN (F2);
        WRITELN (F2, 'B_i_C = ', B_i:20:14);
        WRITELN (F2);
        FOR INDEX := 1 TO N DO
           WRITELN (F2, 'B_i_1 [',index,'] = ',B_I_1[INDEX]:10:8);
        WRITELN (F2);
    END;
 END; (* data_dump *)

(*==================== FORMULAS & TRAVERSAL PROCEDURES ==================*)



FUNCTION PLACE ( I: INTEGER) : ORGAN;

      {.........translates I value to enumerated datatype
                    used to access organ_values array.................}

  BEGIN
     IF I IN CORE_SET THEN
       PLACE := CCORE
     ELSE
       IF I IN MUS_SET THEN
         PLACE := MMUSCLE
       ELSE
         IF I IN FAT_SET THEN
           PLACE := FFAT
         ELSE
           IF I IN SKIN_SET THEN
             PLACE := SSKIN;
   END;   (* place *)

PROCEDURE TEMP_INITIALIZE;

  VAR
    I,J : INTEGER;
    SUM : EXTENDED;

  BEGIN
```

```
    (* initialize temperature array *)

    FOR I := 0 TO N DO
      FOR J := 1 TO M DO
          TEMP_GRID [I,J] := MESH_TEMP / NORMALIZING_TEMP;


    (* initialize and normalize venus and arterial temp arrays *)

      (*  assign t_a_node and calculate_v_node *)
    FOR J := 1 TO M-1 DO
    t_a_node [J] := TA / NORMALIZING_TEMP;

    SUM := 0.0;
    FOR J := M-1 TO M DO
      FOR I := 1 TO N DO
         SUM := SUM + TEMP_GRID[I,J];

    T_V_NODE[M-1] := SUM / (N * 2);

    (* m-2 is original start point *)
      for j := m-1 downto 1 do
         t_v_node [j] := TV / NORMALIZING_TEMP;

    (* initialize t_v_bar, t_a_bar arrays *)

    FOR J := 2 TO M-1 DO
         T_A_BAR [J] := 0.5 * (T_A_NODE [J-1] + T_A_NODE [J]);
    FOR J := 2 TO M-1 DO
          T_V_BAR [J] := 0.5 * (T_V_NODE [J-1] + T_V_NODE [J]);


    (* assign dummy values to unused array elements *)
    t_a_bar[m] := -9999.9;
(*    t_v_bar[m] := -9999.9; *)
    t_a_node[m] := -9999.9;
(*  t_v_node[m] := -9999.9;    *)


    T_A := T_A_BAR;
    T_V := T_V_BAR;

    old_t_a_node := t_a_node;
    old_t_v_node := t_v_node;

END; (* TEMP_INITIALIZE *)

PROCEDURE INITIALIZE;

    {.....initializes all necessary formula calculations
                          and temperature values.....}

  VAR
    I, J : INTEGER;
    I_LOC : ORGAN;
    TOTAL, total_a, total_v, SUM, UaSQRT, UvSQRT, EXP1, GAMM : extended;
    index : organ;

   function hyperbol_cos (x : extended):extended;

      { inverse hyperbolic cosine function, used to calculate H_a_v }

      var
        val, e1,e2,e3 : extended;

      function power (base : extended; exp : integer):extended;
        var
         index : integer;
         temp : extended;

        begin
          temp := 1;
```

```
        for index := 1 to exp do
          temp := temp * base;
        power := temp;
      end; (* power *)

    begin  (* hyperbol_cos *)
      e1 := 1/ (2 * SQR (x));
      e2 := 1 / (4 * POWER(x,4));
      e3 := 1 / (6 * POWER (x,6));
      val := LN(2 * x) - (1/2) * e1 - (3/8) * e2 - (15/48) * e3;
      hyperbol_cos := val;
    end; (* hyperbol_cos *)

BEGIN  (* initialize *)

    (* INITIALIZE ORGAN VALUE ARRAYS USING VALUES FROM CONST BLOCK

        thermal conductivity  *)

    ORG_VALS [CCORE].THERM_CON := core_therm_con;
    ORG_VALS [MMUSCLE].THERM_CON := mus_therm_con;
    ORG_VALS [FFAT].THERM_CON :=  fat_therm_con;
    ORG_VALS [SSKIN].THERM_CON := skin_therm_con;
    ORG_VALS [BBLOOD].THERM_CON := blood_therm_con;

    (*  heat capacity *)

    ORG_VALS [CCORE].HEAT_CAP := core_heat_cap;
    ORG_VALS [MMUSCLE].HEAT_CAP := mus_heat_cap;
    ORG_VALS [FFAT].HEAT_CAP :=  fat_heat_cap;
    ORG_VALS [SSKIN].HEAT_CAP := skin_heat_cap;
    ORG_VALS [BBLOOD].HEAT_CAP := blood_heat_cap;

    (*  density  *)

    ORG_VALS [CCORE].DENSITY := core_density;
    ORG_VALS [MMUSCLE].DENSITY := mus_density;
    ORG_VALS [FFAT].DENSITY := fat_density;
    ORG_VALS [SSKIN].DENSITY := skin_density;
    ORG_VALS [BBLOOD].DENSITY := blood_density;

    (*  basal metabolic rate  *)

    ORG_VALS [CCORE].METAB := core_metab;
    ORG_VALS [MMUSCLE].METAB := mus_metab;
    ORG_VALS [FFAT].METAB := fat_metab;
    ORG_VALS [SSKIN].METAB := skin_metab;
    ORG_VALS [BBLOOD].METAB := -9999;        (* -9999 is a filler value *)


    (* establish basal blood flow rate in cubic meters *)

    ORG_VALS [CCORE].METERS_BAS_BLOOD := core_blood_meters;
    ORG_VALS [MMUSCLE].METERS_BAS_BLOOD := mus_blood_meters;
    ORG_VALS [FFAT].METERS_BAS_BLOOD := fat_blood_meters;
    ORG_VALS [SSKIN].METERS_BAS_BLOOD := skin_blood_meters;
    ORG_VALS [BBLOOD].METERS_BAS_BLOOD := -9999;


    (* establish A_a and A_v Values *)

    A_a := SQR (PHYS_RAD / R_a);
    A_v := SQR (PHYS_RAD / R_v);

     (* establish normalizing value for w_b *)

     W_b_normalizing_val := (SQR(PHYS_RAD) * ORG_VALS [BBLOOD].HEAT_CAP) /
                                ORG_VALS [BBLOOD].THERM_CON;

     (* establish blood alpha *)
```

```
      BL_ALPHA := ORG_VALS [BBLOOD].THERM_CON /
                  (ORG_VALS [BBLOOD].DENSITY * ORG_VALS [BBLOOD].HEAT_CAP);

      (* establish w_b_nrml_array - basal blood flow rate in cubic meters
                  q_array_nrml, b_i_1 array, THERM_CON_ARRAY, HEAT_CAP_ARRAY,
                  DENSITY_ARRAY *)

(* N.B. ------------------------------------------------------------
        subscripts for tissue data access represent the interface
        immediately above the tissue
        ------------------------------------------------------------*)

      FOR I := 1 TO N DO
         BEGIN
           I_LOC := PLACE(I);
           W_ARRAY [I] := ORG_VALS [I_LOC].METERS_BAS_BLOOD;
           Q_ARRAY_PHYS [I] := ORG_VALS [I_LOC].METAB;
           THERM_CON_ARRAY [I] := ORG_VALS[I_LOC].THERM_CON;
           HEAT_CAP_ARRAY [I] :=  ORG_VALS[I_LOC].HEAT_CAP;
           DENSITY_ARRAY [I] :=  ORG_VALS [I_LOC].DENSITY;
           W_b_nrml [I] := W_ARRAY[I] * W_B_NORMALIZING_VAL;
         END;

     W_ARRAY [N] := 0.0;
     W_b_nrml[N] := 0.0;
     W_b_nrml [0] := W_b_nrml [1];
     W_ARRAY [0] := W_ARRAY [1];
     THERM_CON_ARRAY [0] := THERM_CON_ARRAY [1];

         (* checking for w_array values of zero so as to set
              u_a, u_v, h_a_v, h_v_a to zero. ALL w_array values
               must be zero for u_a, u_v, h_a_v, h_v_a to be changed *)

         W_zero_flag := true;

         FOR I := 1 TO N DO
           IF W_ARRAY [I] > 0 THEN
                W_zero_flag := false;

        K_AVG := 0.0;

       FOR I := 0 TO N-1 DO
          K_AVG := K_AVG + THERM_CON_ARRAY [I] *
               (SQR(RADIAL_DIST_ARRAY [I+1]) - SQR(RADIAL_DIST_ARRAY [I]));


(* N.B. ------------------------------------------------------------
        re-assign w_array to W_TISSUE
        REASON : necessity to add n-1 tissue layers before interface
        adjustments have been made. In preparation for calculation
        of capillary data.  Values are found at exact tissues locations,
        not one value greater than actual tissue location.
        ------------------------------------------------------------*)

    for i := 1 to N-1 do
        W_TISSUE [I] := w_array [i+1];
    w_tissue[0] := w_array [1];

      (* apply equalizing formula on interface locations
            p = (GAMMA [i] * p- + p) / (1 + GAMMA [i])
       also determine B_i_1 and ALPHA_PROP arrays     *)

       FOR I := 1 TO N DO
         IF I IN I_INTERFACES THEN
              BEGIN
                GAMM := GAMMA (I,H_r_ARRAY);
                THERM_CON_ARRAY [I] := (GAMM * THERM_CON_ARRAY [I-1] +
                                        THERM_CON_ARRAY [I+1]) / (1 + GAMM);
                HEAT_CAP_ARRAY [I] :=   (GAMM * HEAT_CAP_ARRAY [I-1] +
                                        HEAT_CAP_ARRAY [I+1]) / (1 + GAMM);
                DENSITY_ARRAY [I] :=   (GAMM * DENSITY_ARRAY [I-1] +
```

```
                                    DENSITY_ARRAY [I+1]) / (1 + GAMM);
                  W_ARRAY [I] := (GAMM * W_ARRAY[I-1] + W_ARRAY[I+1]) / (1+GAMM);
                  W_b_nrml [I] := W_ARRAY[I] * W_B_NORMALIZING_VAL;
                Q_ARRAY_PHYS [I] := (GAMM * Q_ARRAY_PHYS [I-1] +
                                    Q_ARRAY_PHYS [I+1]) / (1 + GAMM);

            END;


       FOR I := 1 TO N DO
          BEGIN
           B_i_1 [I] := (H_1 * LEN) / THERM_CON_ARRAY [I];

           ALPHA_PROP [I] := THERM_CON_ARRAY [I] / (DENSITY_ARRAY [I] *
                                    HEAT_CAP_ARRAY [I]);
           Q_ARRAY_nrml [I] := Q_ARRAY_PHYS [I] * SQR(PHYS_RAD) /
                                (NORMALIZING_TEMP * BLOOD_THERM_CON);
(*  b_i_1[i] := 0.0;   *)

          END;


       (* establish U_a and U_v  *)

        IF W_zero_flag THEN
           BEGIN
             U_A := 0.0;
             U_V := 0.0;
             H_A_V := 0.0;
             H_V_A := 0.0;
           END
         ELSE
          BEGIN
           u_a_phys := 2 * k_avg / sqr (phys_rad) / ln (2 * sqrt (r_v/r_a));
           u_v_phys := 2 * k_avg / sqr (phys_rad) / ln (2 * sqrt (r_a/r_v));

           U_a := (U_a_phys * SQR (PHYS_RAD) / ORG_VALS[BBLOOD].THERM_CON);
           U_v := (U_v_phys * SQR (PHYS_RAD) / ORG_VALS[BBLOOD].THERM_CON);

           U_A_DIV := u_a / (n*m);
           U_V_DIV := U_V / (n*m);

           (* establish H_a_v *)

            H_a_v_PHYS := pi* phys_h_z * 2 * K_AVG /
                        HYPERBOL_COS (7 - (SQR(R_v - R_a) / (2 * R_a * R_v)));
            H_A_V := H_A_V_PHYS / (PI * ORG_VALS[BBLOOD].THERM_CON *
                        PHYS_H_Z)* A_a;
            H_V_A := H_A_V_PHYS / (PI * ORG_VALS[BBLOOD].THERM_CON *
                        PHYS_H_Z)* A_v;
          END;

{ reset u_a, u_v, h_a_v, h_v_a,}

(*
h_a_v := 0.0;
h_v_a := 0.0;
*)


(* U_V DIVIDED BY 5 TO ACCOUNT FOR COUNTERCURRENT HEAT EXCHANGE BETWEEN
      THE BLOOD VESSELS *)

       IF (H_V_A <> 0) AND (H_A_V <> 0) THEN
         BEGIN
            U_V := U_V/5.0;
            U_V_DIV := U_V_DIV / 5.0;
         end;


       (* establish m_a and m_v arrays *)
```

```
        (* establish DIMENSIONAL capillary loss *)

        CAPIL_SUM := 0.0;

        FOR I := 0 TO n-1 DO
            CAPIL_SUM := CAPIL_SUM + W_TISSUE [I] *
                     (SQR(PHYS_R_i[I+1]) - SQR(PHYS_R_i[I]));

        CAPILLARY_LOSS :=  PI * PHYS_H_z * CAPIL_SUM;
        START_M_a :=CAPILLARY_LOSS  *  AXIAL_DIVS;
        M_a[1] := START_M_a ;

        FOR J := 2 TO M DO
            M_a[J] := M_a[J-1] - CAPILLARY_LOSS;

        (* controlling the negative zero *)

        IF M_a [M] < CAPILLARY_LOSS THEN
           M_a [M] := 0.0;

        M_v [M] := 0;
        FOR J := M-1 DOWNTO 1 DO
            M_v [J] := M_v [J+1] + CAPILLARY_LOSS;

        (* establish B_a and B_v arrays  *)

        FOR J := 1 TO M DO
           BEGIN
             B_a [J] := (M_a [J] * SQR (PHYS_RAD)) /
                        (ORG_VALS [BBLOOD].DENSITY * PI * SQR (R_a) *
                           PHYS_H_z * BL_ALPHA);
             B_v [J] :=  (M_v [J] * SQR (PHYS_RAD)) /
                        (ORG_VALS [BBLOOD].DENSITY * PI * SQR (R_v) *
                           PHYS_H_z * BL_ALPHA);
             END;

        (* establish  B_i
           N.B.  -   B_I_C in data dump routine is B_i in the program code *)

        B_i := (h_c * PHYS_RAD) / ORG_VALS [SSKIN].THERM_CON;

(*b_i := 0.0;*)


      (* calculate dimensionless DELTA  *)

          SUM := 0.0;
          FOR I := 0 TO N-1 DO
             SUM := SUM + W_tissue [I] * (SQR(RADIAL_DIST_ARRAY[I+1]) -
                        SQR(RADIAL_DIST_ARRAY[I]));

          total_a := -2 * b_a [1] + A_a * SUM - (A_A * U_a + H_a_v);

          total_v := -2 * b_v [1] - 2 *  A_v * SUM  -  (A_V * U_v + H_v_a);
          total := abs(total_v);

          if abs (total_a)  > abs (total_v) then
             total := abs(total_a);

         DELTA := 0.4 / TOTAL ;


         {deltaset}
         {max delta not to exceed 0.004}

         MAX_DELTA := DELTA;
         DELTA_SEC := GC_DELTA;
         WRITELN;
         WRITELN;
         WRITELN ('ORIGINAL DELTA IN SECONDS',DELTA_SEC:10:6);
         WRITELN;
         WRITELN ('PROGRAM CALCULATED MAX DELTA VALUE = ', MAX_DELTA : 20:15);
```

```
        WRITELN;
        DELTA := DELTA_SEC * BL_ALPHA / SQR(PHYS_RAD) ;
        writeln ('USER CALCULATED delta = ',DELTA:20:15);
        WRITELN;
        PAUSE_CRT;


        (* loading numeric constants into variables of type EXTENDED *)

         one := c_one;
         two := c_two;
         three := c_three;
         eight := c_eight;

         (* normalize w_tissue for use in t_a_t_v calculations
             which require an absence of gamma values used in interface
               locations *)

         FOR I := 0 TO N-1 DO
             W_TISSUE[I] := W_TISSUE [I] * W_B_NORMALIZING_VAL;

    END; (*initialize *)



BEGIN  (* main program *)
  clrscr;

  WRITE ('HOW MANY TIMESTEPS ? ');
  READLN (LCV_R);
  WRITELN;
  WRITELN;
  WRITE ('intervals at which to print temperature data ?  ');
  READLN (INTERVAL);
  WRITELN;

  ANATOMICAL_INPUT (DIAM,LEN);

  NUM_DIVISIONS (CORE_DIVS, MUSCLE_DIVS, FAT_DIVS, SKIN_DIVS, N, M);

  WIDTH_CALCULATIONS (H_r_CORE,H_r_MUS,H_r_FAT,H_r_SKIN, CORE_MUS_GAMMA,
                      MUS_FAT_GAMMA, FAT_SKIN_GAMMA, DIAM, LEN,
                      CORE_DIVS, MUSCLE_DIVS,FAT_DIVS, SKIN_DIVS);

  EST_INTERFACES (I_INTERFACES, I_CORE_MUS, I_MUS_FAT, I_FAT_SKIN,
                  SKIN_SURFACE, CORE_DIVS, MUSCLE_DIVS, FAT_DIVS, SKIN_DIVS);

  CREATE_ARRAYS (H_r_ARRAY, RADIAL_DIST_ARRAY,I_CORE_MUS, I_MUS_FAT,
                 I_FAT_SKIN, H_r_CORE, H_r_MUS, H_r_FAT, H_r_SKIN);

  DESTINATION;

  INITIALIZE;

  TEMP_INITIALIZE;

  DATA_DUMP;

  TISSUE;

  CLOSE (F);
  CLOSE (F2);
  CLOSE (OUT_DATA);
  CLRSCR;
  WRITE ('PROGRAM COMPLETED -------');
  PAUSE_CRT;
END. (* main program *)
```

2. VAR_SP.PAS

```
{GLOBAL DECLARATION UNIT  -  INCLUDES CONST, TYPE AND VAR DECLARATIONS }

UNIT var_sp;

INTERFACE
  Uses CRT;

CONST
{////////////////// begin control point section \\\\\\\\\\\\\\\\\\\\\\\\}

   (* GC_DELTA = 0.0 will allow program to determine delta value.
                    GC_DELTA set to any other value will over-ride program
                    calculation of delta value.
         N.B.  -  GC_DELTA MUST BE GIVEN A VALUE IF ALL W VALUES HAVE BEEN
                    SET TO ZERO.      *)

       (* DELTA IS A FULL TIME STEP  -  it will be adjusted later in
            the tissue equations to be a half time step. In the
            blood equations DELTA will be a half time step for the first
            iteration of the loop and a full time step on all subsequent
            iterations.


          N.B.  -- GC_DELTA REPRESENTS DELTA IN SECONDS *)

   GC_DELTA = 1.0;     (*  PROGRAM MINIMUM = 0.0000015; *)

   (* length and diameter must be in CENTIMETERS *)

   GC_LEN = 8.0;
   GC_DIA = 1.5;


   (* control point for number of radial divisions *)

   GC_CORE_DIVS = 3;
   GC_MUS_DIVS =  3;
   GC_FAT_DIVS =  2;
   GC_SKIN_DIVS = 4;


   (* control point for the number of axial divisions *)

      GC_AXIAL_DIVS = 10;

   (* control point for temperature values in degrees C *)

   TA = 35;  (* initial assignment for j=1 to j_max-1 values *)
             (* ta at j=1 held constant during program run at initial value*)

   TV = 35;    (* used for initial assignment at j_max-2 to j = 1 values *)
                    (* seed value at j_max-1 is calculated by program *)

   MESH_TEMP = 35;(* used to set initial tissue temperature throughout mesh*)

   NORMALIZING_TEMP = 37;

   h_1 = 6.09; (* units = Watt / meter squared / deg. C *)
   h_c = 5.17;  (*7.02;*)
```

```
T_0 =  0.0/normalizing_temp ; (* environmental temperature , deg. C *)

(* basal metabolic rate  - CONTROL POINT FOR Q   units = Watt / cu.m *)

core_metab = 170.5;
mus_metab  = 631.9;
fat_metab  = 5.0;
skin_metab = 247.4;

(* basal blood flow rate  CONTROL POINT for W units = kg / sec / cu.m
     N.B. - if ALL of these values are set to zero then GC_DELTA MUST BE
            ASSIGNED A VALUE at the top of this section.  *)

core_blood_meters = 0.173;
mus_blood_meters  = 0.641;
fat_blood_meters  = 0.0;
(* to account for a zero blood flow at the i=n value *)
skin_blood_meters =  0.251 * gc_skin_divs/(gc_skin_divs-1);

(* organ radius to finger ratio (R_i / R)  *)

core_ratio = 0.7057;
muscle_ratio = 0.7954;
fat_ratio    = 0.8099;
skin_ratio   = 1.0;

(*  thermal conductivity  units = Watt / m / deg.C *)

core_therm_con  = 1.064;
mus_therm_con   = 0.418;
fat_therm_con   = 0.204;
skin_therm_con  = 0.293;
blood_therm_con = 0.45;

(* heat capacity  units = J / kg / deg. C *)

core_heat_cap  = 2102.0;
mus_heat_cap   = 3136.0;
fat_heat_cap   = 2520.0;
skin_heat_cap  = 3780.0;
blood_heat_cap = 3899.0;

(* density  units = kg / cu. m *)

core_density  = 1401.0;
mus_density   = 1057.0;
fat_density   = 900.0;
skin_density  = 1057.0;
blood_density = 1060.0;

(* Radii of artery and vein - units = meters *)

R_A = 1000E-6;
R_V = 1500E-6;

{////////////////////// end control point section \\\\\\\\\\\\\\\\\\\\\\\\}
{//////////////  NOTHING should be altered below this point \\\\\\\\\\\\\\\}

J_MAX = GC_AXIAL_DIVS + 1;
I_MAX = GC_CORE_DIVS + GC_MUS_DIVS + GC_FAT_DIVS + GC_SKIN_DIVS + 1;
MAX_PTS = i_max * j_max - i_max;

(* values to be converted to extended precisions variables later *)
```

```
    c_two = 2.0;
    c_eight = 8.0;
    c_three = 3.0;
    c_one = 1.0;

TYPE
    PTR_TYPE = ^TOM_NODE;
    TOM_NODE = RECORD
       INFO : EXTENDED;
       NEXT, BACK : PTR_TYPE;
    END;

    GRID = ARRAY [0..I_MAX, 1..J_MAX] OF extended;
    BOUNDS = SET OF 1..I_MAX;
    extended_ARRAY = ARRAY [1..MAX_PTS] OF extended;
    J_ARRAY = ARRAY [1..J_MAX] OF extended;
    I_ARRAY = ARRAY [0..I_MAX] OF extended;
    A_TYPE = (A_FIRST, A_REG, A_END, A_EXTERNAL, A_EXTERNAL_END);
    R_TYPE = (R_CENTR_END, R_CENTER_REG, R_EXTERNAL_END,
              R_EXTERNAL_REG, R_INTERFACE_REG, R_INTERFACE_END, R_REG_END);
    S_TYPE = (S_REG_CENTER_INT, S_EXTERNAL, S_END, S_END_EXTERNAL);
    CO_ORD = RECORD
         I_LOC, J_LOC : INTEGER;
         tom_temp : extended;
      END;
    L_TYPE = ARRAY [1..MAX_PTS] OF CO_ORD;
    COEF_FIELDS = RECORD
       MINUS, STD, PLUS : extended;
     END;
    COEF_ARRAY = ARRAY [1..I_MAX,1..J_MAX] OF COEF_FIELDS;
    VALUES = RECORD
       THERM_CON, HEAT_CAP, DENSITY,
       METAB, METERS_BAS_BLOOD :extended;
     END;
    ORGAN = (CCORE, MMUSCLE, FFAT, SSKIN, BBLOOD);
    VAL_ARRAY = ARRAY [ORGAN] OF VALUES;
    STR_TYPE = STRING [20];

VAR
  U_PTR, U_TRV, P, P1 : PTR_TYPE;
    av_step, half_av_step,t_step, LCV_R, INTERVAL   :real;
    N , M, AXIAL_DIVS, ROW_TIME, I, J,
    CORE_DIVS, MUSCLE_DIVS, FAT_DIVS,
    SKIN_DIVS, SKIN_SURFACE, D_K, test,lcv,
    count, I_CORE_MUS, I_MUS_FAT, I_FAT_SKIN, COEF_SUB    : INTEGER;

    DIAM, LEN, H_z, PHYS_H_z, RAD, PHYS_RAD,
    PHYS_LEN, U_a, U_v, H_a_v, H_V_A, H_A_V_PHYS,
    DELTA, MAX_DELTA, DELTA_sec, A_a, A_v,
    BL_ALPHA, W_b_normalizing_val,
    B_i, CAPILLARY_LOSS, U_a_phys, U_v_phys,
    START_M_A, CAPIL_SUM,
    H_r_CORE, H_r_MUS, H_r_FAT, H_r_SKIN,
    CORE_MUS_GAMMA, MUS_FAT_GAMMA, FAT_SKIN_GAMMA,
    CORE_WIDTH, MUS_WIDTH, FAT_WIDTH, SKIN_WIDTH,
    CORE_MUS_BOUND, MUS_FAT_BOUND,
    FAT_SKIN_BOUND, SKIN_SURFACE_BOUND, K_AVG,
    one, two, three, eight, zero,u_a_div, u_v_div        : extended;

    I_INTERFACES, CORE_SET, MUS_SET, FAT_SET, SKIN_SET : BOUNDS;

    RADIAL_DIST_ARRAY, H_r_ARRAY, W_ARRAY, W_TISSUE,
    PHYS_R_i, PHYS_H_r_ARRAY, W_b_nrml, Q_ARRAY_nrml,
    THERM_CON_ARRAY, HEAT_CAP_ARRAY, Q_ARRAY_PHYS,
    DENSITY_ARRAY, ALPHA_PROP, B_i_1                    : I_ARRAY;
```

```
   A_NODE                                                    : A_TYPE;
   R_NODE                                                    : R_TYPE;
   S_NODE                                                    : S_TYPE;

   B_a, B_v, M_a, M_v,
   T_V, T_A, T_A_bar, T_V_bar,
   t_a_node, t_v_node, old_t_a_node, old_t_v_node            : J_ARRAY;

   TEMP_GRID, HALF_GRID                                      : GRID;

   LOCATE                                                    : L_TYPE;

   f,f2,OUT_DATA                                        : TEXT;

   ORG_VALS                                                  : VAL_ARRAY;

   RESP2,GRAPH_RESP                                     : CHAR;

   tom_flag,w_zero_flag                                      : boolean;

FUNCTION GAMMA (I_VAL: INTEGER; VAR H_r_ARRAY : I_ARRAY) : extended;
PROCEDURE PAUSE_CRT;
PROCEDURE DERIVE_CENTER_TEMP (VAR DEGREES :GRID);
PROCEDURE DUMP_HALF_ARRAYS;
PROCEDURE DUMP_T_ARRAY (STEP_VAL : REAL; DEGREES : GRID);
PROCEDURE CALC_TA_TV_half (STEP : EXTENDED);

IMPLEMENTATION


(*============================== UTILITIES ================================*)

FUNCTION GAMMA (I_VAL: INTEGER; VAR H_r_ARRAY : I_ARRAY) : extended;

      {.......returns the GAMMA value for any I location in the mesh......}

  BEGIN
    IF I_VAL >= N THEN
       GAMMA := 0.0
    ELSE
      IF I_VAL = 1 THEN
         GAMMA := 1.0
      ELSE
         GAMMA := H_r_ARRAY[I_VAL] / H_r_ARRAY[I_VAL + 1];
  END;


PROCEDURE PAUSE_CRT;

      {..............halts screen output scrolling.....................}
  VAR
    AGAIN : CHAR;

  BEGIN
     WRITE ('PRESS ANY KEY TO CONTINUE : ');
     AGAIN := READKEY;
     WRITELN;
     CLRSCR;
  END;


PROCEDURE DERIVE_CENTER_TEMP (VAR DEGREES :GRID);
```

```
    { establishes center temperatures at t [0,j]
   ***** N.B. this procedure REQUIRES A MINIMUM OF 2 RADIAL DIVISIONS }

  VAR
    J, I : INTEGER;

  BEGIN
     FOR J := 1 TO M DO
       DEGREES [0,J] := DEGREES [2,J] + 1.5 *
                          (DEGREES [1,J] - DEGREES [2,J]);
  END;


PROCEDURE DUMP_HALF_ARRAYS;

     {......prints t_a,t_v arrays.......}

   VAR
      J_val : INTEGER;

   BEGIN
      CLRSCR;
      WRITELN (F,'TIMESTEP = ', HALF_AV_STEP : 5:2, '
**************************');
      WRITE (F,'T_A');
      FOR J_VAL  := 1 TO M-1 DO
          WRITE (F,J_VAL:3,')',T_A_NODE[J_VAL] * NORMALIZING_TEMP:7:3);
      WRITELN(F);
      WRITELN(F);
      WRITE (F,'T_V');
      for J_VAL := 1 to m-1 do
         write (f,j_val:3,')',t_v_node[j_val] * NORMALIZING_TEMP:7:3);
      writeln(f);
  END;


PROCEDURE DUMP_T_ARRAY (STEP_VAL : REAL; DEGREES : GRID);

      {...prints two dimensional temperature matrix or half_step matrix..}

   VAR
     I,J : INTEGER;

   BEGIN
     WRITELN (F);
     WRITELN (F,'TEMPERATURE ARRAY AT TIMESTEP ', STEP_VAL:7:2, '(end/ext)':42);
     FOR I := N DOWNTO 1 DO
       BEGIN
(*         WRITE (F, 'I=',I);  *)
         FOR J := 1 TO M DO
           WRITE (F, DEGREES [I,J] * normalizing_temp:7:3);
         WRITELN (F);
       END;
     WRITELN (F);
     pause_crt;
   END;


PROCEDURE CALC_TA_TV_half (STEP : EXTENDED);

      {....calculates ta_bar[j] and tv_bar[j] in preparation for
                          s_vector calculation  .....................}
```

```
VAR
  SUM1, SUM2, SUM3,W_VAL, HH_aa_vv,
  T_v_plus, B_v_plus, a, b, TEMP, DELTA_VAR          : extended;
  I,J_Val          : INTEGER;
  first,second,third,answer : extended;

BEGIN
  IF STEP = 0.0 THEN
     DELTA_VAR := DELTA / 2
  ELSE
     DELTA_VAR := DELTA;

  FOR J_VAL := 2 TO M-1 DO
    BEGIN

  (* CALCULATE T_A_BAR *)

      SUM1 := 0.0;
      SUM2 := 0.0;
      SUM3 := 0.0;

      FOR I := 0 TO N-1 DO
        BEGIN
            W_VAL := W_TISSUE [I];

            SUM1 := SUM1 + W_val * (SQR (RADIAL_DIST_ARRAY[I+1]) -
                                    SQR (RADIAL_DIST_ARRAY[I]));
            SUM2 := SUM2 + (TEMP_GRID[I,J_VAL-1]  + TEMP_GRID [I,J_VAL]) *
                           (SQR (RADIAL_DIST_ARRAY[I+1]) -
                                    SQR (RADIAL_DIST_ARRAY[I]));

            SUM3 := SUM3 + W_VAL * (TEMP_GRID [I, J_VAL-1] +
                     TEMP_GRID [I,J_VAL]) * (SQR (RADIAL_DIST_ARRAY [I+1])
                         - SQR (RADIAL_DIST_ARRAY [I]));
         END;


        (* FINAL FORMULAS *)

        (* T_A is the t_a_BAR value at the previous full time step
            the same is true with regard to T_V and t_v_BAR  *)

         T_a_bar [J_VAL] := t_a [j_val] + DELTA_VAR * (( -two * B_A [J_VAL-1]
+
                     A_A * SUM1 - A_A * U_A - H_A_V) * T_A [J_VAL] +
                    (two * B_A [J_VAL-1] - A_A * SUM1) * T_A_node [J_VAL-1]
                    + 0.5 * A_A * U_A * SUM2 + H_A_V * T_V [J_VAL]);

      END; (* J-LOOP *)


(* CALCULATE T_V_BAR *)

     FOR J_VAL := 2 TO M-1 DO
      BEGIN

        SUM1 := 0.0;
        SUM2 := 0.0;
        SUM3 := 0.0;

        FOR I := 0 TO N-1 DO
          BEGIN
            W_VAL := W_TISSUE [I];

            SUM1 := SUM1 + W_val * (SQR (RADIAL_DIST_ARRAY[I+1]) -
```

```
                                SQR (RADIAL_DIST_ARRAY[I]));
            SUM2 := SUM2 + (TEMP_GRID[I,J_VAL+1]  + TEMP_GRID [I,J_VAL]) *
                           (SQR (RADIAL_DIST_ARRAY[I+1]) -
                                SQR (RADIAL_DIST_ARRAY[I]));

            SUM3 := SUM3 + W_VAL * (TEMP_GRID [I, J_VAL+1] +
                    TEMP_GRID [I,J_VAL]) *
                      (SQR (RADIAL_DIST_ARRAY [I+1])
                        - SQR (RADIAL_DIST_ARRAY [I]));
        END;  (* I-LOOP *)


        T_v_bar[J_VAL] := t_v [j_val] + DELTA_VAR * ((two * B_V [J_VAL-1]
                - two * A_V * SUM1 - A_V * U_V - H_V_A) * T_V [J_VAL] +
                 (-two * B_V [J_VAL-1] + A_V * SUM1) * T_V_NODE [J_VAL-1] +
                  0.5 * A_V * U_V * SUM2 + H_V_A * T_A [J_VAL] + 0.5 * A_V *
SUM3);


    END; (*J_VAL*)


    (*  calculate t_a_node and t_v_node *)
    for j_val := 2 to m-1 do
        t_a_node [j_val] := 2 * t_a_bar [j_val] - t_a_node [j_val-1];


  SUM1 := 0.0;
  FOR J_VAL := M-1 TO M DO
  FOR I := 1 TO N DO
       SUM1 := SUM1 + HALF_GRID [I,J_VAL];
  T_V_NODE[M-1] := SUM1 /(2* N) ;


  for j_val := m-1 downto 2 do
     t_v_node [j_val-1] := 2 * t_v_bar[j_val] - t_v_node [j_val];

END; (* calc_ta_tv_half *)
END.
```

3. TISUE_SP.PAS

```
Unit tisue_sp;

interface
  PROCEDURE TISSUE;

implementation
   uses var_sp, phys_sp, crt, dos, printer;


PROCEDURE TISSUE;
  VAR
   ORIG_A_r_COEF, ORIG_A_z_COEF                          : COEF_ARRAY;


PROCEDURE AXIAL_DIR  ( I_VAL, J_VAL : INTEGER;
                       VAR J_MINUS_FORM, J_FORM, J_PLUS_FORM : extended);

      {.....imports the I and J coordinates, identifies the type
            of node using enumerated data types and calculates three
            coefficients at the given I,J location to fill the
            ORIG_A_z_COEF array used for the traversal in the
            RADIAL direction.....}

VAR
  HEAT_DEN_FRACT, A_SQRD, FRACT, W_U_A,temp_w_u_a : extended;

BEGIN
  (* establish node type *)

   IF J_VAL = 1 THEN
      A_NODE := A_FIRST;              {pt. 0}
   IF J_VAL = M THEN
     IF I_VAL <> N THEN
       A_NODE := A_END                {pt. 4,6,7}
     ELSE
       A_NODE := A_EXTERNAL_END;      {pt. 5}
   IF (J_VAL <> M) AND (J_VAL <> 1) THEN
      IF I_VAL <> N THEN
         A_NODE := A_REG              {pt. 1,3}
      ELSE
         A_NODE := A_EXTERNAL;        {pt. 2}


   (* preliminary calculations *)


   A_SQRD := (BL_ALPHA / ALPHA_PROP [I_VAL]) * SQR(LEN/PHYS_RAD) ;
   HEAT_DEN_FRACT := (ORG_VALS[BBLOOD].DENSITY * ORG_VALS[BBLOOD].HEAT_CAP) /
                     (DENSITY_ARRAY[I_VAL]  * HEAT_CAP_ARRAY[I_VAL]);
   FRACT := one/(A_SQRD * SQR(H_z)) ;


   W_U_A := W_b_nrml [I_VAL] + U_A_div + U_v_div;

   (* setting w, u_a, and u_v at external skin level to 0.0 *)
   temp_w_u_a := 0.0;


   (* FINAL CALCULATIONS *)

   CASE A_NODE OF
     A_FIRST : ;

     A_REG : BEGIN
               J_MINUS_FORM  := FRACT;
               J_FORM := -one*(two * FRACT) - (HEAT_DEN_FRACT * (W_U_A / two));
               J_PLUS_FORM := FRACT;
             END;
```

```
      A_EXTERNAL : BEGIN
                     J_MINUS_FORM := FRACT;
                     J_FORM := -two * FRACT -  (HEAT_DEN_FRACT *
                        (three * temp_W_U_A) / eight);
                     J_PLUS_FORM := FRACT;

                   END;
      A_END : BEGIN
                J_MINUS_FORM := two * FRACT;
                J_FORM := -two * FRACT * (one + H_z * B_i_1 [I_VAL])
                          - ((HEAT_DEN_FRACT * W_b_nrml [I_VAL]) / two );
                J_PLUS_FORM := 0.0;
              END;

      A_EXTERNAL_END : BEGIN
                       J_MINUS_FORM := two * FRACT;
                       J_FORM := -two * FRACT * (one + H_Z * B_i_1 [I_VAL]) -
                          (three * HEAT_DEN_FRACT * W_b_nrml [I_VAL] / eight);
                       J_PLUS_FORM := 0.0;
                       END;
    END; (*CASE*)
  END;  (* axial_dir *)

PROCEDURE RAD_DIR (I_VAL, J_VAL: INTEGER;
                   VAR  I_MINUS_FORM, I_FORM, I_PLUS_FORM : extended);

      {.....imports the I and J coordinates, identifies the
            type of node using enumerated data types and calculates
            three coefficients at the given I,J location to fill the
            ORIG_A_r_COEF array used for the traversal in the
            SECOND TRAVERSAL.....................................}

    VAR
      ALPHA_FORM, HEAT_DEN_FRACT,
      FRACT, FORM, GAMM, H_R_I, R_I,temp_form : extended;

    BEGIN

      (* establish node type *)

      IF (J_VAL = M) THEN
        BEGIN
          R_NODE := R_REG_END;                    {pt. 4}
          IF (I_VAL IN I_INTERFACES) THEN
             R_NODE := R_INTERFACE_END;           {pt. 7}
          IF (I_VAL = 1) THEN
             R_NODE := R_CENTR_END;               {pt. 6}
          IF I_VAL = N THEN
             R_NODE := R_EXTERNAL_END;            {pt. 5}
        END
       ELSE
        BEGIN
          IF I_VAL = 1 THEN
            R_NODE := R_CENTER_REG;               {pt. 1}
          IF I_VAL = N THEN
            R_NODE := R_EXTERNAL_REG;             {pt. 2}
           IF (I_VAL <> 1) AND (I_VAL <> N) THEN
            R_NODE := R_INTERFACE_REG;           {pt. 3}
        END;

       (* preliminary calculations *)

       ALPHA_FORM := ALPHA_PROP [I_VAL] / BL_ALPHA ;
       HEAT_DEN_FRACT := (ORG_VALS[BBLOOD].DENSITY *
                          ORG_VALS[BBLOOD].HEAT_CAP) /
                         (DENSITY_ARRAY[I_VAL]  * HEAT_CAP_ARRAY[I_VAL]);

       (* H_R ARRAY AT THE 1 LOCATION MUST USE THE VALUE AT H_R_ARRAY [2] *)

       IF I_VAL = 1 THEN
         BEGIN
           FRACT := two / SQR(H_R_ARRAY [2]);
```

```
    GAMM := GAMMA (2, H_R_ARRAY);
    H_R_I := H_R_ARRAY [2];
  END
ELSE
 BEGIN
  FRACT := two / SQR(H_R_ARRAY[I_VAL]);
  GAMM := GAMMA (I_VAL, H_R_ARRAY);
  H_R_I := H_R_ARRAY [I_VAL];
END;


 FORM := W_b_nrml[I_VAL] + U_a_div + U_v_div;

(* setting w, u_a, and u_v at external skin level to 0.0 *)
temp_form := 0.0;


R_I := RADIAL_DIST_ARRAY [I_VAL];

(* final calculations *)
CASE R_NODE OF
  R_CENTER_REG : BEGIN
                 I_MINUS_FORM := 0.0;
                 I_FORM := -one * ALPHA_FORM * FRACT - (HEAT_DEN_FRACT
                           * FORM / two);
                 I_PLUS_FORM := ALPHA_FORM * FRACT;
                 END;
  R_INTERFACE_REG : BEGIN
                    I_MINUS_FORM := ALPHA_FORM * (one / ((1 + GAMM) * H_R_I))
                                    *  (two * GAMM / H_R_I - 1 / R_I);
                    I_FORM := -one * ALPHA_FORM *  (one / H_R_I) *
                           ((two * GAMM / H_R_I)- (one- GAMM) / R_I)
                            - (HEAT_DEN_FRACT * (FORM / two));
                    I_PLUS_FORM := ALPHA_FORM * (SQR(GAMM) /
                       (H_R_I * (one + GAMM))) * (two / H_R_I + one / R_I);
                    END;
   R_EXTERNAL_REG : BEGIN

                    I_MINUS_FORM := ALPHA_FORM * FRACT - (HEAT_DEN_FRACT
                                    * temp_FORM / eight);
                    I_FORM := -one * ALPHA_FORM * (FRACT + (one + (two / H_R_I))
                          * B_I) - (HEAT_DEN_FRACT * (three *  temp_FORM/eight));
                    I_PLUS_FORM := 0.0;
                    END;
   R_REG_END : BEGIN
               I_MINUS_FORM := ALPHA_FORM * (one / H_R_I) *
                               (one / H_R_I - one / (two * R_I));
               I_FORM := -one * ALPHA_FORM * FRACT - (HEAT_DEN_FRACT *
                         W_b_nrml [I_VAL] / two);
               I_PLUS_FORM := ALPHA_FORM * one / H_R_I * (one / H_R_I +
                              one / (two * R_I));
               END;
   R_CENTR_END : BEGIN
                 I_MINUS_FORM := 0.0;
                 I_FORM := -one * ALPHA_FORM * FRACT - (HEAT_DEN_FRACT *
                           W_b_nrml [I_VAL] / two);
                 I_PLUS_FORM := ALPHA_FORM * FRACT;
                 END;
  R_INTERFACE_END : BEGIN
                    I_MINUS_FORM := ALPHA_FORM * (one / ((one + GAMM) * H_R_I)) *
                                    (two * GAMM / H_R_I - one / R_I);
                    I_FORM := -one * ALPHA_FORM *  (one / H_R_I) *
                              ((two * GAMM / H_R_I)- (one- GAMM) / R_I)
                               - (HEAT_DEN_FRACT * W_b_nrml [I_VAL] / two);

                    I_PLUS_FORM := ALPHA_FORM * (SQR(GAMM) /
                           (H_R_I * (one + GAMM))) * (two / H_R_I + one / R_I);
                    END;
   R_EXTERNAL_END : BEGIN
                    I_MINUS_FORM := ALPHA_FORM * (two / SQR(H_R_I)) -
                                    (HEAT_DEN_FRACT *  W_b_nrml [I_VAL] / eight);
                    I_FORM := -one * ALPHA_FORM  * (FRACT + (one + two / H_R_I)
```

```
                                    * B_i) - ( three * HEAT_DEN_FRACT *
                                    W_b_nrml [I_VAL] / eight);
                            I_PLUS_FORM := 0.0;
                            END;
        END;
    END; (* rad_dir *)


PROCEDURE S_VECTOR (I_VAL, J_VAL : INTEGER; VAR S_VECT : GRID);

      {.......calculates s_vector coefficient in preparation for
                                    mesh traversal .....................}

    VAR
      A, W_FORMULA, T_FORMULA,
      HEAT_DEN_FRACT, ALPHA_FORM, A_SQRD,
      H_R,  NODE_VAL : extended;
      temp_u_v :extended;

    BEGIN

      (* estblish node type *)

      IF (J_VAL = M) AND (I_VAL = N) THEN
         S_NODE := S_END_EXTERNAL
       ELSE
         IF (J_VAL = M) THEN
            S_NODE := S_END
          ELSE
            IF (I_VAL = N) THEN
               S_NODE := S_EXTERNAL
             ELSE
               S_NODE := S_REG_CENTER_INT;


      (* S_VECTOR CALCULATIONS - PRELIMINARY FORMULAS *)

      A_SQRD := (BL_ALPHA / ALPHA_PROP [I_VAL]) * SQR(LEN/PHYS_RAD);
      ALPHA_FORM := ALPHA_PROP [I_VAL] / BL_ALPHA;
      HEAT_DEN_FRACT := (ORG_VALS[BBLOOD].DENSITY *
                                    ORG_VALS[BBLOOD].HEAT_CAP) /
                           (DENSITY_ARRAY[I_VAL]  * HEAT_CAP_ARRAY[I_VAL]);

     H_r := H_R_ARRAY [I_VAL];

 if s_node = s_external then
   begin
    temp_u_v := 0.0;
    w_formula := 0.0;
   end
  else
     W_FORMULA := W_b_nrml [I_VAL] + U_A_div;

     IF J_VAL = M THEN
        NODE_VAL := T_A_NODE[J_VAL -1]
     ELSE
        NODE_VAL := T_A_NODE [J_VAL];

     T_FORMULA := NODE_VAL - (TEMP_GRID [I_VAL-1, J_VAL] / eight);

     (* S_VECTOR CALCULATIONS - FINAL FORMULAS *)
     CASE S_NODE OF
        S_REG_CENTER_INT    :
                             S_VECT [I_VAL, J_VAL]:=
                                 HEAT_DEN_FRACT * (Q_ARRAY_nrml [I_VAL] +
                       W_FORMULA * T_A_NODE [J_VAL] + U_V_div * T_V_NODE [J_VAL]);
        S_EXTERNAL      :
                          S_VECT [I_VAL,J_VAL] :=
                              HEAT_DEN_FRACT * (Q_ARRAY_nrml [I_VAL] +
                       W_FORMULA * T_FORMULA +temp_U_V * (T_V_NODE [J_VAL] -
                         TEMP_GRID [I_VAL-1, J_VAL] / eight)) + ALPHA_FORM *
                       (one + two/ H_R) * B_I  * T_0;
```

```
      S_END          :
                      S_VECT [I_VAL,J_VAL] :=
                        HEAT_DEN_FRACT * (Q_ARRAY_nrml[I_VAL] +
                          W_b_nrml [I_VAL] * T_A_NODE [j_val-1]) +
                          (( two * B_i_1 [I_VAL] * T_0) / (A_SQRD * H_Z));
      S_END_EXTERNAL :
                      S_VECT [I_VAL,J_VAL] :=
                        HEAT_DEN_FRACT * (Q_ARRAY_nrml [I_VAL] +
                          W_b_nrml [I_VAL] * T_FORMULA ) +
                            ALPHA_FORM * (one + two/H_R) * B_I * T_0 +
                             (two * B_i_1[I_VAL] * T_0) / (A_SQRD * H_Z);
    END (* CASE *);
END; (* s_vector *)


PROCEDURE ESTABLISH_COEF_ARRAYS;

    {......creates arrays of axial and radial coefficients using
                          arrays of records .....................}

  VAR
    I, J : INTEGER;

  BEGIN

    (* establish arrays of coefficients in the radial direction
          n.b.  --  each I value references a record containing
                      three coefficients found in arrays i_mat, i_mat2,
                      ORIG_A_z.   *)

    FOR i := n downto 1 do
      FOR j := m downto 2 do
        RAD_DIR (I,J, ORIG_A_r_COEF [I,J].MINUS,
            ORIG_A_r_COEF [I,J].STD, ORIG_A_r_COEF [I,J].PLUS);



    FOR I := n downTO 1 DO
       FOR J := m downto 2 do
         AXIAL_DIR (I,j, ORIG_A_z_COEF [I,J].MINUS,
           ORIG_A_z_COEF[I,J].STD, ORIG_A_z_COEF[I,J].PLUS);

  END; (* establish_coef_arrays *)



PROCEDURE TRAVERSE;

    {......traverses the mesh in the radial and axial directions
                       using THOMAS' ALGORITHM.....................}

 VAR
  INDEX, X,start, prev_total, total  : INTEGER;
  S_PTR, S_TRV,
  C_PTR, C_TRV,
  A_PTR, A_TRV,
  B_PTR, B_TRV,
  CI_PTR, CI_TRV,
  AI_PTR, AI_TRV,
  BI_PTR, BI_TRV,
  CJ_PTR, CJ_TRV,
  AJ_PTR, AJ_TRV,
  BJ_PTR, BJ_TRV,
  D_PTR, D_TRV, NEWNODE     :PTR_TYPE;

  S_V_COEF : GRID;
  DUMP, DUMP3 : CHAR;


 PROCEDURE UPDATE_TISSUE_1 (VAR TISSUE : GRID);
```

```
      VAR
        I_COUNT : INTEGER;

      BEGIN
           FOR I_COUNT := 1 TO N DO
                TISSUE [I_COUNT,1] := 30 / NORMALIZING_TEMP;

      END;

PROCEDURE THOMAS_ALG(SYS_SIZE :INTEGER);


   VAR
     I,ii,jj,z : INTEGER;

     P_PTR, P_TRV,
     Q1_PTR, Q1_TRV,
     BK_Q1, BK_Q1_TRV,
     BK_P, BK_P_TRV, NEWNODE : PTR_TYPE;

  PROCEDURE CAL_CPRIME_DPRIME;
   VAR
    I : INTEGER;
    FORM1 : EXTENDED;

   BEGIN
      NEW(NEWNODE);
      NEWNODE^.INFO := B_PTR^.INFO / A_PTR^.INFO;
      NEWNODE^.NEXT := NIL;
      NEWNODE^.BACK := NIL;
      P_PTR := NEWNODE;
      P_TRV := P_PTR;
      BK_P := NIL;
      NEW(NEWNODE);
      NEWNODE^.INFO := D_PTR^.INFO / A_PTR^.INFO;
      NEWNODE^.NEXT := NIL;
      NEWNODE^.BACK := NIL;
      Q1_PTR := NEWNODE;
      Q1_TRV := Q1_PTR;
      BK_Q1 := NIL;
      A_TRV := A_PTR^.NEXT;
      C_TRV := C_PTR^.NEXT;
      B_TRV := B_PTR^.NEXT;
      D_TRV := D_PTR^.NEXT;
      FOR I := 2 TO SYS_SIZE DO
        BEGIN
           FORM1 := A_TRV^.INFO - C_TRV^.INFO * P_TRV^.INFO;
           NEW (NEWNODE);
           NEWNODE^.INFO := B_TRV^.INFO / FORM1;
           NEWNODE^.NEXT := NIL;
           NEWNODE^.BACK := P_TRV;
           P_TRV^.NEXT := NEWNODE;
           P_TRV := P_TRV^.NEXT;
           NEW(NEWNODE);
           NEWNODE^.NEXT := NIL;
           NEWNODE^.INFO := (D_TRV^.INFO - C_TRV^.INFO * Q1_TRV^.INFO) / FORM1;
           NEWNODE^.BACK := Q1_TRV;
           Q1_TRV^.NEXT := NEWNODE;
           Q1_TRV := Q1_TRV^.NEXT;
           A_TRV := A_TRV^.NEXT;
           C_TRV := C_TRV^.NEXT;
           B_TRV:= B_TRV^.NEXT;
           D_TRV:= D_TRV^.NEXT;
        END;
      BK_P := P_TRV;
      BK_Q1 := Q1_TRV;

  END;

  PROCEDURE CALC_VALS;
     VAR
       I : INTEGER;
```

```
      BEGIN
        NEW(NEWNODE);
        NEWNODE^.INFO := BK_Q1^.INFO;
        NEWNODE^.NEXT := NIL;
        U_PTR := NEWNODE;
        BK_Q1_TRV := BK_Q1^.BACK;
        U_TRV := U_PTR;
        BK_P_TRV := BK_P^.BACK;

    (* U IS BEING FILLED VIA FIRST NODE INSERTION *)

    FOR I := SYS_SIZE-1 DOWNTO 1 DO
      BEGIN
        NEW(NEWNODE);
        NEWNODE^.INFO := BK_Q1_TRV^.INFO - BK_P_TRV^.INFO * U_TRV^.INFO;
        NEWNODE^.NEXT := U_PTR;
        U_PTR := NEWNODE;
        BK_Q1_TRV := BK_Q1_TRV^.BACK;
        BK_P_TRV := BK_P_TRV^.BACK;
        u_trv := u_ptr;
      END;
  END;

BEGIN  (* THOMAS ALG *)

    MARK (p);
    NEW (U_PTR);
    U_PTR^.INFO := 0.0;
    U_PTR^.NEXT := NIL;
    NEW (Q1_PTR);
    Q1_PTR^.INFO := 0.0;
    Q1_PTR^.NEXT := NIL;
    NEW (P_PTR);
    P_PTR^.INFO := 0.0;
    P_PTR^.NEXT := NIL;
    U_TRV := U_PTR;
    Q1_TRV := Q1_PTR;
    P_TRV := P_PTR;
    FOR I :=2 TO SYS_SIZE DO
      BEGIN
        NEW(NEWNODE);
        NEWNODE^.INFO := 0.0;
        NEWNODE^.NEXT := NIL;
        U_TRV^.NEXT := NEWNODE;
        U_TRV := U_TRV^.NEXT;
        NEW(NEWNODE);
        NEWNODE^.INFO := 0.0;
        NEWNODE^.NEXT := NIL;
        Q1_TRV^.NEXT := NEWNODE;
        Q1_TRV := Q1_TRV^.NEXT;
        NEW(NEWNODE);
        NEWNODE^.INFO := 0.0;
        NEWNODE^.NEXT := NIL;
        P_TRV^.NEXT := NEWNODE;
        P_TRV := P_TRV^.NEXT;
      END;

    CAL_CPRIME_DPRIME;
    CALC_VALS;

END;


    PROCEDURE SECOND_TRAVERSE_AXIAL;

        {...traverses the mesh in the axial direction moving necessary
              coefficients from record storage structure into extended
                 precision arrays for access by Thomas' Algorithm.....}

      VAR
        I,J,j_find : INTEGER;
        I_MIN_TEMP, I_MAX_TEMP,HALF_DELTA : extended;
```

```
  INDEX : INTEGER;

BEGIN

   MARK (P1);
   S_PTR := NIL;
   C_PTR := NIL;
   A_PTR := NIL;
   B_PTR := NIL;
   CI_PTR := NIL;
   AI_PTR := NIL;
   BI_PTR := NIL;
   D_PTR := NIL;

   half_delta := delta/2;
   TOTAL := 0;
   PREV_TOTAL := 1;
   FOR I := 1 to n do
    BEGIN
      X := 0;
      FOR j := 2 to m do
       BEGIN
         X := X + 1;
         (* load coefficient arrays *)

         S_VECTOR (I, J, S_V_COEF);

         NEW (NEWNODE);

         NEWNODE^.INFO := S_V_COEF [I,J] * HALF_DELTA;      (* S[X]*)
         NEWNODE^.NEXT := NIL;
         IF S_PTR = NIL THEN
           BEGIN
             S_PTR := NEWNODE;
             S_TRV := S_PTR;
           END
         ELSE
           BEGIN
             S_TRV^.NEXT := NEWNODE;
             S_TRV := S_TRV^.NEXT;
           END;

        NEW (NEWNODE);
        NEWNODE^.INFO := -1 * ORIG_A_z_COEF [I,J].MINUS * HALF_DELTA;  (*C[X]*)
        NEWNODE^.NEXT := NIL;
        IF C_PTR = NIL THEN
           BEGIN
             C_PTR := NEWNODE;
             C_TRV := C_PTR;
           END
         ELSE
           BEGIN
             C_TRV^.NEXT := NEWNODE;
             C_TRV := C_TRV^.NEXT;
           END;

           NEW (NEWNODE);
           NEWNODE^.INFO := -1 * ORIG_A_z_COEF [I,J].STD * HALF_DELTA + 1;  (*A[X]*)
           NEWNODE^.NEXT := NIL;
           IF A_PTR = NIL THEN
           BEGIN
             A_PTR := NEWNODE;
             A_TRV := A_PTR;
           END
         ELSE
           BEGIN
             A_TRV^.NEXT := NEWNODE;
             A_TRV := A_TRV^.NEXT;
           END;

           NEW (NEWNODE);
           NEWNODE^.INFO := -1 * ORIG_A_z_COEF [I,J].PLUS * HALF_DELTA;  (* B[X]*)
```

```
  NEWNODE^.NEXT := NIL;
  IF B_PTR = NIL THEN
  BEGIN
    B_PTR := NEWNODE;
    B_TRV := B_PTR;
  END
ELSE
  BEGIN
    B_TRV^.NEXT := NEWNODE;
    B_TRV := B_TRV^.NEXT;
  END;

  NEW (NEWNODE);
  NEWNODE^.INFO := ORIG_A_r_COEF [I,J].MINUS * HALF_DELTA;   (* CI[X]*)
  NEWNODE^.NEXT := NIL;
  IF CI_PTR = NIL THEN
  BEGIN
    CI_PTR := NEWNODE;
    CI_TRV := CI_PTR;
  END
ELSE
  BEGIN
    CI_TRV^.NEXT := NEWNODE;
    CI_TRV := CI_TRV^.NEXT;
  END;

  NEW (NEWNODE);
  NEWNODE^.INFO := ORIG_A_r_COEF [I,J].STD * HALF_DELTA + 1;   (* AI[X]*)
  NEWNODE^.NEXT := NIL;
  IF AI_PTR = NIL THEN
  BEGIN
    AI_PTR := NEWNODE;
    AI_TRV := AI_PTR;
  END
ELSE
  BEGIN
    AI_TRV^.NEXT := NEWNODE;
    AI_TRV := AI_TRV^.NEXT;
  END;

  NEW (NEWNODE);
  NEWNODE^.INFO := ORIG_A_r_COEF [I,J].PLUS * HALF_DELTA;   (* BI[X] *)
  NEWNODE^.NEXT := NIL;
  IF BI_PTR = NIL THEN
  BEGIN
    BI_PTR := NEWNODE;
    BI_TRV := BI_PTR;
  END
ELSE
  BEGIN
    BI_TRV^.NEXT := NEWNODE;
    BI_TRV := BI_TRV^.NEXT;
  END;

(* calculate constant term *)

NEW (NEWNODE);
NEWNODE^.NEXT := NIL;
IF D_PTR = NIL THEN
  BEGIN
    D_PTR := NEWNODE;
    D_TRV := D_PTR;
  END
ELSE
  BEGIN
    D_TRV^.NEXT := NEWNODE;
    D_TRV := D_TRV^.NEXT;
  END;

IF I = 1 THEN
    D_TRV^.INFO := AI_TRV^.INFO * HALF_GRID [I,J] +
             BI_TRV^.INFO * half_grid [I+1,J] + S_TRV^.INFO
```

```
                ELSE
               IF I = N THEN
                   D_TRV^.INFO := CI_TRV^.INFO * HALF_GRID [I-1,J] +
                       AI_TRV^.INFO * HALF_GRID [I,J] + S_TRV^.INFO
                 ELSE
                     D_TRV^.INFO := CI_TRV^.INFO * half_grid [i-1,j] +
                       AI_TRV^.INFO * HALF_GRID [I,J] + bi_TRV^.INFO
                         * half_grid [i+1,j] + S_TRV^.INFO;

               (* adjust d[x] for j = 2 position *)

               IF J = 2 THEN
                   D_TRV^.INFO := D_TRV^.INFO - C_TRV^.INFO * HALF_GRID [I,1];

              (* save I,J locations to facilitate placing of new
                     temperatures in proper locations *)
                TOTAL := TOTAL + 1;
                LOCATE [TOTAL].I_LOC := I;
                LOCATE [TOTAL].J_LOC := J;

          END;    (* J_LOOP *)


       THOMAS_ALG (X);

      (* move temperatures generated by THOMAS_ALG to
                linear array locations *)

        U_TRV := U_PTR;
        FOR INDEX := PREV_TOTAL TO TOTAL DO
          BEGIN
            LOCATE [INDEX].TOM_TEMP := U_TRV^.INFO;
            U_TRV := U_TRV^.NEXT;
          END;

        RELEASE (P);
        RELEASE (P1);
        PREV_TOTAL := PREV_TOTAL + X;
       END; (* I_LOOP *)

      (* move temperatures generated by THOMAS_ALG to
                proper mesh locations *)
        FOR INDEX := 1 TO TOTAL DO
            HALF_GRID [LOCATE[INDEX].I_LOC,LOCATE[INDEX].J_LOC] := locate [INDEX].tom_temp;

        DERIVE_CENTER_TEMP (HALF_GRID);

    END;

    PROCEDURE FIRST_TRAVERSE_RADIAL;

         {...traverses the mesh in the radial direction moving necessary
               coefficients from record storage structure into extended
               precision arrays for access by Thomas' Algorithm .....}

       VAR
         I,J,J_FIND : INTEGER;
         J_MAX_TEMP,HALF_DELTA : extended;
         ti,tj : integer;
         INDEX : INTEGER;

        BEGIN

          S_PTR := NIL;
          C_PTR := NIL;
          A_PTR := NIL;
          B_PTR := NIL;
          CJ_PTR := NIL;
          AJ_PTR := NIL;
          BJ_PTR := NIL;
          D_PTR := NIL;
```

```
MARK(P1);
HALF_DELTA := DELTA / 2;
PREV_TOTAL := 1;
TOTAL := 0;
FOR J := 2 TO m DO
  BEGIN

      X := 0;
      FOR I := 1 to n DO
      BEGIN
        X := X + 1;

        (* load coefficient arrays *)

        S_VECTOR (I, J, S_V_COEF);
        NEW (NEWNODE);
        NEWNODE^.INFO := S_V_COEF[I,J] * HALF_DELTA;  (* S[X] *)
        NEWNODE^.NEXT := NIL;
      IF S_PTR = NIL THEN
        BEGIN
          S_PTR := NEWNODE;
          S_TRV := S_PTR;
        END
      ELSE
        BEGIN
          S_TRV^.NEXT := NEWNODE;
          S_TRV := S_TRV^.NEXT;
        END;


        NEW (NEWNODE);
       NEWNODE^.INFO := -1 * ORIG_A_r_COEF [I,J].MINUS * HALF_DELTA;  (* C[X]*)
       NEWNODE^.NEXT := NIL;
       IF C_PTR = NIL THEN
        BEGIN
          C_PTR := NEWNODE;
          C_TRV := C_PTR;
        END
      ELSE
        BEGIN
          C_TRV^.NEXT := NEWNODE;
          C_TRV := C_TRV^.NEXT;
        END;

        NEW (NEWNODE);
        NEWNODE^.INFO := -1 * ORIG_A_r_COEF [I,J].STD * HALF_DELTA + 1;  (* A[X] *)
       NEWNODE^.NEXT := NIL;
        IF A_PTR = NIL THEN
        BEGIN
          A_PTR := NEWNODE;
          A_TRV := A_PTR;
        END
      ELSE
        BEGIN
          A_TRV^.NEXT := NEWNODE;
          A_TRV := A_TRV^.NEXT;
        END;


        NEW (NEWNODE);
        NEWNODE^.INFO := -1 * ORIG_A_r_COEF [I,J].PLUS * HALF_DELTA;  (* B[X]*)
      NEWNODE^.NEXT := NIL;
        IF B_PTR = NIL THEN
        BEGIN
          B_PTR := NEWNODE;
          B_TRV := B_PTR;
        END
      ELSE
        BEGIN
          B_TRV^.NEXT := NEWNODE;
          B_TRV := B_TRV^.NEXT;
        END;
```

```
  NEW (NEWNODE);
  NEWNODE^.INFO := ORIG_A_z_COEF [I,J].MINUS * HALF_DELTA;  (* C[J]*)
  NEWNODE^.NEXT := NIL;
  IF CJ_PTR = NIL THEN
  BEGIN
    CJ_PTR := NEWNODE;
    CJ_TRV := CJ_PTR;
  END
ELSE
  BEGIN
    CJ_TRV^.NEXT := NEWNODE;
    CJ_TRV := CJ_TRV^.NEXT;
  END;


  NEW (NEWNODE);
  NEWNODE^.INFO := ORIG_A_z_COEF [I,J].STD * HALF_DELTA + 1;  (* AJ[X]*)
  NEWNODE^.NEXT := NIL;
  IF AJ_PTR = NIL THEN
  BEGIN
    AJ_PTR := NEWNODE;
    AJ_TRV := AJ_PTR;
  END
ELSE
  BEGIN
    AJ_TRV^.NEXT := NEWNODE;
    AJ_TRV := AJ_TRV^.NEXT;
  END;


  NEW (NEWNODE);
  NEWNODE^.INFO := ORIG_A_z_COEF [I,J].PLUS * HALF_DELTA;  (* BJ[X]*)
  NEWNODE^.NEXT := NIL;
  IF BJ_PTR = NIL THEN
  BEGIN
    BJ_PTR := NEWNODE;
    BJ_TRV := BJ_PTR;
  END
ELSE
  BEGIN
    BJ_TRV^.NEXT := NEWNODE;
    BJ_TRV := BJ_TRV^.NEXT;
  END;


  (*calculate constant term *)

NEW (NEWNODE);
newnode^.info := 0.0;
NEWNODE^.NEXT := NIL;
IF D_PTR = NIL THEN
  BEGIN
    D_PTR := NEWNODE;
    D_TRV := D_PTR;
  END
ELSE
  BEGIN
    D_TRV^.NEXT := NEWNODE;
    D_TRV := D_TRV^.NEXT;
  END;


  IF J = M THEN
    D_TRV^.INFO := CJ_TRV^.INFO * HALF_GRID[I,J-1] +
           AJ_TRV^.INFO * HALF_GRID [I,J] +
           s_TRV^.INFO

  ELSE

   D_TRV^.INFO := CJ_TRV^.INFO * HALF_GRID[I,J-1] +
```

```
                        AJ_TRV^.INFO * HALF_GRID [I,J] +
                        BJ_TRV^.INFO * half_grid [i,j+1] + s_TRV^.INFO;


              (* save I,J locations to facilitate placing of new
                    temperatures in proper locations *)
                TOTAL := TOTAL + 1;
                LOCATE [TOTAL].I_LOC := I;
                LOCATE [TOTAL].J_LOC := J;


          END;

       THOMAS_ALG (X);

       (* move temperatures generated by THOMAS_ALG to
                linear array locations *)

        U_TRV :=  U_PTR;
        FOR INDEX := PREV_TOTAL TO TOTAL DO
          BEGIN
            LOCATE [INDEX].TOM_TEMP := U_TRV^.INFO;
            U_TRV := U_TRV^.NEXT;
          END;

        RELEASE(P);
        RELEASE(P1);
        PREV_TOTAL := PREV_TOTAL + X;
       END;

       (* move temperatures generated by THOMAS_ALG to
                   proper mesh locations *)
        FOR INDEX := 1 TO TOTAL DO
            HALF_GRID [LOCATE[INDEX].I_LOC,LOCATE[INDEX].J_LOC] := locate [INDEX].tom_temp;

       DERIVE_CENTER_TEMP (HALF_GRID);


    END;


   PROCEDURE FILE_OUT (STEP:REAL; VAR TISSUE:GRID;
                       T_A_TEMP, T_V_TEMP : J_ARRAY);
      VAR
         INDEX : INTEGER;
         ROW, COL : integer;
         AVG : J_ARRAY;
        BEGIN

          (* SENDS OUTPUT AT IDENTIFIED TIME INTERVAL TO FILE TEMP_OUT.DAT
               DATA IS PRINTED AS FORMATTED REALS IN A 7:3 FIELD USING
                  A COMMA AS THE DELIMITER *)
(*

          IF T_STEP = 0.0 THEN
            BEGIN
             WRITELN (OUT_DATA, 'ORIGINAL PROGRAM');
             WRITELN (OUT_DATA,'ALL VALUES ARE AT THE EXTERNAL LAYER');
             WRITE (OUT_DATA,'STEP':10,'J2':6, 'J6':12);
             WRITELN (OUT_DATA,'END11':13, 'WMUS-6':10, 'WMUS-6':8,'TA6':5,'TV6':8);
            END;
          WRITE (OUT_DATA, T_STEP:10:2,' ,');
          WRITE (OUT_DATA, HALF_GRID[N,2]* NORMALIZING_TEMP:7:3, ' ,');
          WRITE (OUT_DATA, HALF_GRID[N,6]*NORMALIZING_TEMP:7:3,' ,');
          WRITE (OUT_DATA, HALF_GRID[N,M]*NORMALIZING_TEMP:7:3,' ,');
          WRITE (OUT_DATA, W_ARRAY[6]:7:3,' ,', W_ARRAY[6]:7:3,' ,');
          WRITELN (OUT_DATA, T_A_NODE[6]* NORMALIZING_TEMP:7:3,' ,'
,T_V_NODE[6]*NORMALIZING_TEMP:7:3);
```

```
*)
        (*  necessary M and N values for use by GRAPH program *)
        IF T_STEP = 0.0 THEN
           BEGIN
             WRITELN (OUT_DATA, M, ' =M');
             WRITELN (OUT_DATA,N, ' =N');
           END;

         WRITELN (OUT_DATA, STEP:5:2);
         WRITELN (OUT_DATA,'T_A');
         FOR INDEX := 1 TO M-1 DO
           BEGIN
             WRITE (OUT_DATA, T_A_TEMP [INDEX]* NORMALIZING_TEMP:7:3);
             IF INDEX <> M-1 THEN
                WRITE (OUT_DATA, ' ,');
           END;
         WRITELN (OUT_DATA);

         WRITELN (OUT_DATA, 'T_V');
         FOR INDEX := 1 TO M-1 DO
            BEGIN
              WRITE (OUT_DATA, T_V_TEMP [INDEX]* NORMALIZING_TEMP:7:3);
              IF INDEX <> M-1 THEN
                 WRITE (OUT_DATA,' ,');
            END;
         WRITELN (OUT_DATA);

         WRITELN (OUT_DATA, 'TISSUE TEMPERATURES');
         FOR ROW := N DOWNTO 1 DO
           BEGIN
             FOR COL := 1 TO M DO
               BEGIN
                 WRITE (OUT_DATA, TISSUE[ROW,COL]* NORMALIZING_TEMP : 7:3);
                 IF COL <> M THEN
                    WRITE (OUT_DATA, ' ,');
               END;
          WRITELN (OUT_DATA);
         END;

         WRITELN (OUT_DATA, 'EXTRAPOLATED CENTER LINE TEMPERATURES');
         FOR COL := 1 TO M DO
            WRITE (OUT_DATA, TISSUE[0,COL]* NORMALIZING_TEMP:7:3);
         WRITELN (OUT_DATA);
         WRITELN (OUT_DATA);


         END;


BEGIN (*TRAVERSE *)

 (* initialize step counters *)

 T_STEP := 0.0;
 AV_STEP := 0.0;
 Half_AV_STEP := 0.0;

 (* prompt for memory dump options *)

 CLRSCR;
    WRITELN (F, 'INITIAL VALUES ');
    WRITELN (F);
    DUMP_HALF_ARRAYS;
    DUMP_T_ARRAY (T_STEP, TEMP_GRID);

 (* INITIALIZE GRID *)

 HALF_GRID := TEMP_GRID;
 HALF_AV_STEP := HALF_AV_STEP + 0.5;
 FILE_OUT (T_STEP, HALF_GRID, T_A_NODE, T_V_NODE);
```

```
      CALC_TA_TV_HALF (T_STEP);

        REPEAT
            RELEASE (HEAPORG);
            T_STEP := T_STEP + 0.5;

            CALC_TA_TV_HALF (T_STEP);

            FIRST_TRAVERSE_RADIAL;

            RELEASE (HEAPORG);

            IF T_STEP = 0.5 THEN
              BEGIN
                DUMP_HALF_ARRAYS;
                DUMP_T_ARRAY (T_STEP, HALF_GRID);
              END;

            T_STEP := T_STEP + 0.5;

            SECOND_TRAVERSE_AXIAL;
            RELEASE (HEAPORG);


            TEMP_GRID := HALF_GRID;
            HALF_AV_STEP := HALF_AV_STEP + 1;
            AV_STEP := AV_STEP + 1;
            T_a := T_a_bar;
            T_v := T_v_bar;
            old_t_v_node := t_v_node;
            old_t_a_node := t_a_node;


          (* print time step  counter *)
           CLRSCR;
           GOTOXY (40,12);
           WRITELN (trunc(t_step));

          IF (T_STEP/INTERVAL = TRUNC(T_STEP / interval) * 1.0) THEN
             BEGIN
               CLRSCR;
               DUMP_HALF_ARRAYS;
               DUMP_T_ARRAY (T_STEP, HALF_GRID);
               FILE_OUT (T_STEP, HALF_GRID,T_A_NODE, T_V_NODE);
              end;

(* DUMP DATA FOR GRAPHICS TEST PROGRAM *)
(*       IF T_STEP = 1000 THEN
         file_out (t_step, half_grid, t_a_node, t_v_node);  *)

(*RETURN TISSUE TEMP AT J=1 TO 30
   IF T_STEP = 100 THEN
       UPDATE_TISSUE_1 (HALF_GRID);
*)

  UNTIL LCV_R = T_STEP;

 END;    (* TRAVERSE *)


BEGIN (* TISSUE *)

  ESTABLISH_COEF_ARRAYS;

  TRAVERSE;

END; (*TISSUE*)

end.
```

4. PHYS_SP.PAS

```
{ ANATOMICAL INPUT MODULE }

UNIT PHYS_sp;


 INTERFACE
  USES var_sp;
  PROCEDURE ANATOMICAL_INPUT (VAR DIA, LEN : extended);
  PROCEDURE NUM_DIVISIONS (VAR CORE_SUBDIV, MUSCLE_SUBDIV,
                           FAT_SUBDIV, SKIN_SUBDIV, N, M : INTEGER);
  PROCEDURE WIDTH_CALCULATIONS (VAR H_CORE, H_MUS, H_FAT, H_SKIN,
                                    CORE_MUS_GAMMA, MUS_FAT_GAMMA,
                                     FAT_SKIN_GAMMA : extended;
                                DIA, LEN : REAL;
                                CORE_DIV, MUS_DIV, FAT_DIV,
                                SKIN_DIV : INTEGER);
  PROCEDURE EST_INTERFACES (VAR BOUND_SET : BOUNDS;
                            VAR CORE_MUS_BNDRY, MUS_FAT_BNDRY,
                                FAT_SKIN_BNDRY, SKIN_SURFACE: INTEGER;
                            CORE_SUBDIV, MUS_SUBDIV, FAT_SUBDIV,
                                SKIN_SUBDIV : INTEGER);
  PROCEDURE CREATE_ARRAYS (VAR H_ARRAY,DISTANCE_ARRAY : I_ARRAY;
                           CORE_MUS_BOUND,MUS_FAT_BOUND,FAT_SKIN_BOUND:INTEGER;
                           H_r_CORE, H_r_MUS, H_r_FAT, H_r_SKIN : REAL);
IMPLEMENTATION
   USES
     CRT, DOS, PRINTER ;
(*============================== ANATOMICAL PROCEDURES ======================*)


PROCEDURE ANATOMICAL_INPUT (VAR DIA, LEN : extended);

      {.......provides for keyboard input of length and diameter in cm.
                      exports length and diameter in meters .............}

  VAR
    RESPONSE : CHAR;

  BEGIN
   dia := GC_DIA;
   len := GC_LEN;
{
  CLRSCR;
   WRITE ('ENTER THE DIAMETER (cm) : ');
   READLN (DIA);
   WRITELN;
   WRITE ('ENTER THE LENGTH (cm) : ');
   READLN (LEN);
   REPEAT
     CLRSCR;
     WRITELN ('EXIT / VERIFY SCREEN');
     WRITELN;
     WRITELN ('1) DIAMETER = ':20, DIA : 15:7 , ' cm');
     WRITELN ('2) LENGTH   = ':20, LEN : 15:7 , ' cm');
     WRITELN;
     WRITELN ('ENTER THE NUMBER OF THE ITEM TO BE CHANGED');
     WRITE ('-------- ENTER   V   TO VERIFY AND CONTINUE --------> ');
     READLN (RESPONSE);
     IF (RESPONSE = '1') OR (RESPONSE = '2') THEN
       BEGIN
        WRITELN;
        CASE RESPONSE OF
            '1' : BEGIN
                    WRITE ('ENTER THE NEW DIAMETER (cm) : ');
                    READLN (DIA);
                    WRITELN;
                  END;
            '2' : BEGIN
```

```
                    WRITE ('ENTER THE NEW LENGTH (cm) : ');
                    READLN (LEN);
                    WRITELN;
                  END;
          END; (* CASE *)
        END;
    UNTIL ((RESPONSE = 'V') OR (RESPONSE = 'v'));
}

 (* CONVERSION TO METERS *)

    LEN := LEN / 100;
    DIAM := DIAM / 100;
    RAD := DIA/2;
    PHYS_RAD := RAD;


END;  (* ANATOMICAL_INPUT *)


PROCEDURE NUM_DIVISIONS (VAR CORE_SUBDIV, MUSCLE_SUBDIV,
                         FAT_SUBDIV, SKIN_SUBDIV, N, M : INTEGER);

      {.....provides for user entry for the number
                    of divisions in each organ.....}

  VAR
    RESPONSE : CHAR;

  BEGIN
    axial_divs := GC_AXIAL_DIVS;
    m := axial_divs + 1;
    core_subdiv := GC_CORE_DIVS;
    muscle_subdiv := GC_MUS_DIVS;
    fat_subdiv := GC_FAT_DIVS;
    skin_subdiv := GC_SKIN_DIVS;
    n := core_subdiv + muscle_subdiv + fat_subdiv + skin_subdiv + 1;
{
    N := 9999;
    REPEAT
        CLRSCR;
        WRITELN;
        WRITELN ('MAXIMUM AXIAL DIVISIONS = ', J_MAX-1);
        WRITELN;
        WRITE ('ENTER THE NUMBER OF AXIAL DIVISIONS DESIRED : ');
        READLN (AXIAL_DIVS);
        WRITELN;
        IF (AXIAL_DIVS > J_MAX-1) OR (AXIAL_DIVS <=0) THEN
          BEGIN
            WRITELN ('AXIAL DIVISIONS EXCEED PROGRAM PARAMETERS');
            WRITELN ('AXIAL INPUT SEQUENCE WILL BEGIN AGAIN');
            PAUSE_CRT;
          END;
       IF (AXIAL_DIVS >=1) AND (AXIAL_DIVS <= J_MAX-1) THEN
          BEGIN
            CLRSCR;
            WRITELN;
            WRITELN ('VERIFY / EDIT SCREEN');
            WRITELN;
            WRITELN ('          AXIAL DIVISIONS = ', AXIAL_DIVS);
            M := AXIAL_DIVS + 1;
            WRITELN;
            WRITELN ('          M = ',M);
            WRITELN;
            WRITELN ('ENTER  V  TO VERIFY AND CONTINUE OR');
            WRITE ('        PRESS E TO CHANGE AXIAL DIVISIONS DATA ');
            READLN (RESPONSE);
            WRITELN;
            CLRSCR;
          END;
       UNTIL ((RESPONSE = 'V') OR (RESPONSE = 'v'));
    RESPONSE := ' ';
```

```
    WHILE N > I_MAX DO
      BEGIN
        WRITELN ('ENTER THE NUMBER OF SUBDIVISIONS FOR THE FOLLOWING LAYERS -');
        WRITELN ('( MAXIMUM TOTAL SUBDIVISIONS = ':50, I_MAX-1, ' )');
        WRITELN;
        WRITE ('CORE : ': 25);
        READLN (CORE_SUBDIV);
        WRITELN;
        WRITE ('MUSCLE : ' : 25);
        READLN (MUSCLE_SUBDIV);
        WRITELN;
        WRITE ('FAT : ' :25);
        READLN (FAT_SUBDIV);
        WRITELN;
        WRITE ('SKIN : ':25);
        READLN (SKIN_SUBDIV);
        WRITELN;
        REPEAT
          CLRSCR;
          WRITELN;
          WRITELN ('VERIFY / EDIT SCREEN');
          WRITELN;
          WRITELN ('SUBDIVISION VALUES ENTERED ');
          WRITELN;
          WRITELN ('1)  CORE     : ':40, CORE_SUBDIV);
          WRITELN ('2)  MUSCLE   : ':40, MUSCLE_SUBDIV);
          WRITELN ('3)  FAT      : ':40, FAT_SUBDIV);
          WRITELN ('4)  SKIN     : ':40, SKIN_SUBDIV);
          WRITELN;
          WRITELN ('ENTER THE NUMBER OF THE LAYER TO BE CHANGED');
          WRITE ('------- ENTER  V  TO VERIFY AND CONTINUE ----------> ');
          READLN (RESPONSE);
          WRITELN;
          IF (RESPONSE >= '1') AND (RESPONSE <='4') THEN
            BEGIN
              WRITELN;
              WRITE ('ENTER THE NEW VALUE FOR THE ');
              CASE RESPONSE OF
                '1' : BEGIN
                        WRITE ('CORE : ');
                        READLN (CORE_SUBDIV);
                      END;
                '2' : BEGIN
                        WRITE ('MUSCLE : ');
                        READLN (MUSCLE_SUBDIV);
                      END;
                '3' : BEGIN
                        WRITE ('FAT : ');
                        READLN (FAT_SUBDIV);
                      END;
                '4' : BEGIN
                        WRITE ('SKIN : ');
                        READLN (SKIN_SUBDIV);
                      END;
               END; (* CASE *)
              WRITELN;
           END; (* IF *)
        UNTIL ((RESPONSE = 'V') OR (RESPONSE = 'v'));
        N := CORE_SUBDIV + MUSCLE_SUBDIV + FAT_SUBDIV + SKIN_SUBDIV + 1;
        IF N > I_MAX-1 THEN
          BEGIN
            WRITELN ('DIVISION TOTAL EXCEEDS PROGRAM PARAMETERS');
            WRITELN ('MAXIMUM TOTAL DIVISIONS CANNOT EXCEED ', I_MAX-1);
            WRITELN;
            WRITELN ('DATA ENTRY SEQUENCE WILL START AGAIN');
            PAUSE_CRT;
          END;
      END; (* WHILE *)
}
    (* ESTABLISH H_z AND NORMALIZE *)

    PHYS_H_z := LEN / AXIAL_DIVS;
```

```
    H_Z :=PHYS_H_z / LEN;

  END;    (* num_divisions *)

PROCEDURE WIDTH_CALCULATIONS (VAR H_CORE, H_MUS, H_FAT, H_SKIN, CORE_MUS_GAMMA,
                                  MUS_FAT_GAMMA,
                                  FAT_SKIN_GAMMA : extended;
                              DIA, LEN : REAL; CORE_DIV, MUS_DIV, FAT_DIV,
                              SKIN_DIV : INTEGER);

      {......imports the location of organ boundaries, length, diameter
             calculates width of each organ
             exports organ width, H- values, and gamma values.............}

   BEGIN  (* width_calculations *)


     (* establish sample boundary *)
      core_mus_bound := core_ratio;
      mus_fat_bound := muscle_ratio;
      fat_skin_bound := fat_ratio ;
      skin_surface_bound := skin_ratio ;

     (* DETERMINE LAYER WIDTHS  *)

      CORE_WIDTH := CORE_MUS_BOUND  ;
      MUS_WIDTH := (MUS_FAT_BOUND - CORE_MUS_BOUND ) ;
      FAT_WIDTH := (FAT_SKIN_BOUND - MUS_FAT_BOUND) ;
      SKIN_WIDTH := (SKIN_SURFACE_BOUND - FAT_SKIN_BOUND);



     (* DETERMINE H- VALUES  *)

     H_CORE := (CORE_WIDTH / (CORE_DIV + 0.5));
     H_MUS  := (MUS_WIDTH / MUS_DIV);
     H_FAT  := (FAT_WIDTH / FAT_DIV) ;
     H_SKIN := (SKIN_WIDTH / SKIN_DIV) ;

    (* DETERMINE GAMMA VALUES *)

     CORE_MUS_GAMMA := H_CORE / H_MUS;
     MUS_FAT_GAMMA := H_MUS / H_FAT ;
     FAT_SKIN_GAMMA := H_FAT / H_SKIN;

END; (* width_calculations *)

PROCEDURE EST_INTERFACES (VAR BOUND_SET : BOUNDS;
                          VAR CORE_MUS_BNDRY, MUS_FAT_BNDRY,
                              FAT_SKIN_BNDRY, SKIN_SURFACE: INTEGER;
                          CORE_SUBDIV, MUS_SUBDIV, FAT_SUBDIV,
                              SKIN_SUBDIV : INTEGER);

      {......establishes a SET of I values representing the location of
                              organ interfaces.........................}

  BEGIN
    CORE_MUS_BNDRY := CORE_SUBDIV + 1;
    MUS_FAT_BNDRY := MUS_SUBDIV + CORE_MUS_BNDRY;
    FAT_SKIN_BNDRY := MUS_FAT_BNDRY + FAT_SUBDIV;
    SKIN_SURFACE := FAT_SKIN_BNDRY + SKIN_SUBDIV;
    BOUND_SET := [CORE_MUS_BNDRY, MUS_FAT_BNDRY, FAT_SKIN_BNDRY];
  END;


PROCEDURE CREATE_ARRAYS (VAR H_ARRAY,DISTANCE_ARRAY : I_ARRAY;
                         CORE_MUS_BOUND,MUS_FAT_BOUND,FAT_SKIN_BOUND:INTEGER;
                         H_r_CORE, H_r_MUS, H_r_FAT, H_r_SKIN : REAL);

      {.......creates a set of I values for each organ and establishes
               a series of arrays containing R_i and H_r values
                        indexed to the I locations.....................}
```

```
    VAR
      INDEX : INTEGER;
      H_TOTAL : REAL;

    BEGIN   (* create arrays *)
      CORE_SET := [1..(CORE_MUS_BOUND)];
      MUS_SET := [CORE_MUS_BOUND + 1..MUS_FAT_BOUND];
      FAT_SET := [MUS_FAT_BOUND + 1..FAT_SKIN_BOUND];
      SKIN_SET := [FAT_SKIN_BOUND + 1..N];
      FOR INDEX := 1 TO N DO
        BEGIN
          IF INDEX IN CORE_SET THEN
            H_ARRAY [INDEX] := H_r_CORE;
          IF INDEX IN MUS_SET THEN
            H_ARRAY [INDEX] := H_r_MUS;
          IF INDEX IN FAT_SET THEN
            H_ARRAY[INDEX] := H_r_FAT;
          IF INDEX IN SKIN_SET THEN
            H_ARRAY [INDEX] := H_r_SKIN;
        END;
      H_ARRAY[0] := 0.0;
      DISTANCE_ARRAY [0] := 0.0;
      H_ARRAY[1] := H_ARRAY[2]/2;

      (* establish R_i and physical_r_i arrays *)

      FOR INDEX := 1 TO N DO
         PHYS_H_r_ARRAY [INDEX] := H_ARRAY[INDEX] * PHYS_RAD;

      DISTANCE_ARRAY [1] := H_ARRAY[1];
      PHYS_R_i [1] := PHYS_H_r_ARRAY [1];
phys_r_i [0] := 0;
      FOR INDEX := 2 TO N DO
        BEGIN
          DISTANCE_ARRAY[INDEX] := DISTANCE_ARRAY[INDEX-1] + H_ARRAY[INDEX];
          PHYS_R_i [INDEX] := DISTANCE_ARRAY[INDEX] * PHYS_RAD;
        END;
       H_TOTAL := 0.0;
       FOR INDEX := 1 TO N DO
         H_TOTAL := H_TOTAL + H_ARRAY[INDEX];
  END; (* create_arrays *)

 END.
```

APPENDIX D - DISTRIBUTION LIST

2 Copies to:

Defense Technical Information Center
ATTN: DTIC-DDA
Alexandria, VA 22304-6145

Office of the Assistant Secretary of Defense (Hlth Affairs)
ATTN: Medical Readiness
Washington, DC 20301-1200

Commander
U.S. Army Medical Research and Development Command
ATTN: SGRD-PLC
Fort Detrick
Frederick, MD 21702-5012

Commander
U.S. Army Medical Research and Development Command
ATTN: SGRD-PLE
Fort Detrick
Frederick, MD 21702-5012

Commandant
Army Medical Department Center and School
ATTN: HSMC-FR, Bldg. 2840
Fort Sam Houston, TX 78236

1 Copy to:

Joint Chiefs of Staff
Medical Plans and Operations Division
Deputy Director for Medical Readiness
ATTN: RAD Smyth
Pentagon, Washington, DC 20310

HQDA
Office of the Surgeon General
Preventive Medicine Consultant
ATTN: SGPS-PSP
5109 Leesburg Pike
Falls Church, VA 22041-3258

HQDA

Assistant Secretary of the Army for Research, Development and Acquisition
ATTN: SARD-TM
Pentagon, Washington, DC 20310

HQDA
Office of the Surgeon General
ATTN: DASG-ZA
5109 Leesburg Pike
Falls Church, VA 22041-3258

HQDA
Office of the Surgeon General
ATTN: DASG-DB
5109 Leesburg Pike
Falls Church, VA 22041-3258

HQDA
Office of the Surgeon General
Assistant Surgeon General
ATTN: DASG-RDZ/Executive Assistant
Room 3E368, The Pentagon
Washington, DC 20310-2300

HQDA
Office of the Surgeon General
ATTN: DASG-MS
5109 Leesburg Pike
Falls Church, VA 22041-3258

Uniformed Services University of the Health Sciences
Dean, School of Medicine
4301 Jones Bridge Road
Bethesda, MD 20814-4799

Uniformed Services University of the Health Sciences
ATTN: Department of Military and Emergency Medicine
4301 Jones Bridge Road
Bethesda, MD 20814-4799

Commandant
Army Medical Department Center & School
ATTN: Chief Librarian Stimson Library
Bldg 2840, Room 106
Fort Sam Houston, TX 78234-6100

Commandant
Army Medical Department Center & School
ATTN: Director of Combat Development
Fort Sam Houston, TX 78234-6100

Commander
U.S. Army Aeromedical Research Laboratory
ATTN: SGRD-UAX-SI
Fort Rucker, AL 36362-5292

Commander
U.S. Army Medical Research Institute of Chemical Defense
ATTN: SGRD-UVZ
Aberdeen Proving Ground, MD 21010-5425

Commander
U.S. Army Medical Materiel Development Activity
ATTN: SGRD-UMZ
Fort Detrick
Frederick, MD 21702-5009

Commander
U.S. Army Institute of Surgical Research
ATTN: SGRD-USZ
Fort Sam Houston, TX 78234-5012

Commander
U.S. Army Medical Research Institute of Infectious Diseases
ATTN: SGRD-UIZ-A
Fort Detrick
Frederick, MD 21702-5011

Director
Walter Reed Army Institute of Research
ATTN: SGRD-UWZ-C (Director for Research Management)
Washington, DC 20307-5100

Commander
U.S. Army Natick Research, Development & Engineering Center
ATTN: SATNC-Z
Natick, MA 01760-5000

Commander
U.S. Army Natick Research, Development & Engineering Center
ATTN: SATNC-T
Natick, MA 01760-5002

Commander
U.S. Army Natick Research, Development & Engineering Center
ATTN: SATNC-MIL
Natick, MA 01760-5040

Commander
U.S. Army Research Institute for Behavioral Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Commander
U.S. Army Training and Doctrine Command
Office of the Surgeon
ATTN: ATMD
Fort Monroe, VA 23651-5000

Commander
U.S. Army Environmental Hygiene Agency
Aberdeen Proving Ground, MD 21010-5422

Director, Biological Sciences Division
Office of Naval Research - Code 141
800 N. Quincy Street
Arlington, VA 22217

Commanding Officer
Naval Medical Research & Development Command
NNMC/Bldg 1
Bethesda, MD 20889-5044

Commanding Officer
U.S. Navy Clothing & Textile Research Facility
P.O. Box 59
Natick, MA 01760-0001

Commanding Officer
Navy Environmental Health Center
2510 Walmer Avenue
Norfolk, VA 23513-2617

Commanding Officer
Naval Aerospace Medical Institute (Code 32)
Naval Air Station
Pensacola, FL 32508-5600

Commanding Officer
Naval Medical Research Institute
Bethesda, MD 20889

Commanding Officer
Naval Health Research Center
P.O. Box 85122
San Diego, CA 92138-9174

Commander
Armstrong Medical Research Laboratory
Wright-Patterson Air Force Base, OH 45433

Strughold Aeromedical Library
Document Services Section
2511 Kennedy Circle
Brooks AFB, TX 78235-5122

Commander
US Air Force School of Aerospace Medicine
Brooks Air Force Base, TX 78235-5000

Director
Human Research & Engineering
US Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5001