1994

Unstructured grid analysis of three-dimensional conduction in complex geometries

John Charles Minor
Iowa State University

Unstructured grid analysis of three-dimensional conduction in complex geometries

by

John Charles Minor

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Mechanical Engineering
Major: Mechanical Engineering

Iowa State University
Ames, Iowa
1994

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Symbols	Description
T	Temperature
k	Thermal Conductivity
c_p	Specific Heat
h	Convective Coefficient
L	Length
A	Surface Area
t	Time
J_μ	Bessel function of the first kind of order μ
$\hat{n}$	Unit vector normal to a surface

Greek Symbols

α	Thermal Diffusivity
β	Zero of a characteristic function
ϵ	Emissivity
ρ	Density
σ	Stefan-Boltzmann constant

Subscripts

r	Reference Values
o	Reference Values or Initial Conditions

Superscripts

$\tilde{}$	Dimensional values
$\hat{}$	Unit vector

ACKNOWLEDGMENTS

The author gratefully acknowledges the support provided for this study by the U.S. Department of Commerce through Grant No. ITA 87-02 and support from the Iowa Manufacturing Technology Center administered through the Iowa State University Center for Advanced Technology Development, Office of Technology Commercialization. I would also like to express special thanks to the Institute for Physical Research and Technology, Dr. R.H. Pletcher, Dr. B. Lograsso, my loving wife Chris, and my daughter Savannah.

CHAPTER 1. INTRODUCTION

Cooling of electronic components has become a very important research topic with the emphasis on fast processors for the computer industry. As computer processors become faster and more capable yet physically smaller, the heat dissipation required to maintain the processors below a critical temperature has increased dramatically. Arbitrary arrangements of passive heat sinks and fans are approaching their heat dissipation limits. A method for modeling the complex thermal interactions of electronic components is needed. The research presented in this thesis is the first step toward such a modeling capability.

Several recent works describe the trend of electronic components to become more capable and smaller, requiring the dissipation of extreme amounts of heat. Bar-Cohen [1] presents a nice overview of the direction thermal management schemes have taken in the past and where they seem to be going. Ahmed, Krane and Parsons [2] present a preliminary investigation of enhancement of heat transfer for electronic devices using flat plate heat sinks. The flat plate heat sink enhances radiative heat transfer and is therefore useful where convective heat transfer is limited.

Much of the research in electronic cooling focuses on convection cooling using air as the fluid. This is because using air as the primary means of cooling is the most cost effective thermal management technique. Wirtz and Mathur [3] describe

the variation of the heat transfer coefficient on a thin, flat electronic package. This can give information on local hot spots or areas where enhancement will do the most good. Keyhani, Prasad and Cox [4] investigated natural convection heat transfer in a vertical cavity with discrete heat sources. This information shows local effects which are generated by having multiple heat sources in a single cavity. Moffat and Anderson [5] have shown that care must be taken when applying heat transfer coefficient data to electronic systems. It is very important to define terms in specific ways otherwise large errors can result in heat transfer calculations.

Work in the use of fins and other techniques for thermal enhancement has been progressing. Guglielmimi, Nannei and Tanda [6] studied the effects of shrouding on heat transfer from staggered fin arrays in natural convection. This type of enhancement is used quite often in electronic equipment by applying finned heat sinks to increase heat dissipation. This has been done successfully in personal computers (PCs) and workstations; however, the greater demands of the newest computer chips require more scientific and less arbitrary use of these heat sinks. Incropera [7] presents an overview of the considerations which must be met when using enhanced convective cooling to dissipate heat.

Xie and Lee [8] have presented a thermal management scheme and design criteria for an electronic system. Their scheme uses a specified arrangement of boards and holes to dissipate the heat generated by a system.

Although these works are important there is still a need to model the complex thermal interactions of an electronic system such as a computer. Most of the commercial methods now available use the finite element method (FEM) to solve the conduction problem within a geometry. FEM algorithms tend to use tetrahedral

based unstructured grids because of the ease with which tetrahedra can fill a space. The research presented in this thesis is the development of a finite volume algorithm and numerical solver for three-dimensional conduction problems on tetrahedral unstructured grids. The use of a finite volume formulation for solution of this type of problem has not been discovered in the literature by the author. Therefore it seems appropriate to extend the finite volume formulation to this class of problem. The method is vertex based and uses an accumulation scheme to sum the fluxes out of a control volume. The flux summation is not complete until the entire grid is accumulated. The accumulation scheme devised allows for a reduction in memory usage compared to traditional vertex based finite volume formulations by not requiring information about neighboring tetrahedra to be stored. This allows the numerical solver to work on computers with limited memory resources.

The numerical method is validated using test cases with known analytical solutions and boundary conditions. The testing of the scheme gives experience in using the solver and validates the method by comparison with accepted solutions. Finally the usefulness of the algorithm is demonstrated by running a sample calculation on a complex geometry which is typically found in the electronics industry.

CHAPTER 2. GOVERNING EQUATION FOR CONDUCTION

In this chapter the governing equation for transient three-dimensional conduction is presented and the nondimensional form is developed. In addition, the initial and boundary conditions necessary for the solution of the governing equation are discussed.

The governing equation for conduction can be given in the following form [9],

$$\tilde{\rho}\tilde{c}_p \frac{\partial \tilde{T}}{\partial \tilde{t}} = \nabla \cdot \left(\tilde{k} \nabla \tilde{T}\right) \tag{2.1}$$

where the term on the left hand side of the equals sign is a storage term and the term on the right hand side is the sum of fluxes into an arbitrary control volume.

Nondimensional Form of the Governing Equation

The governing equation can be nondimensionalized using the definitions in Table 2.1. Note that the definitions do not include nondimensional time; this will be determined as part of the nondimensionalization process, so that the form of the nondimensionalized governing equation will appear similar to equation (2.1). Introducing the nondimensional variables into equation (2.1) yields,

$$\rho \rho_o c_p c_{p_o} \frac{\partial T_r T}{\partial \tilde{t}} = \nabla \cdot \left(k k_o \nabla \left(T T_r\right)\right) \tag{2.2}$$

Table 2.1: Definitions for nondimensional terms

$T = \frac{\tilde{T}}{T_r}$	$k = \frac{\tilde{k}}{k_o}$	$c_p = \frac{\tilde{c_p}}{c_{p_o}}$
$\rho = \frac{\tilde{\rho}}{\rho_o}$	$x = \frac{\tilde{x}}{L_r}$	$y = \frac{\tilde{y}}{L_r}$
$z = \frac{\tilde{z}}{L_r}$		

The terms involving x, y, z, and L_r are hidden in the derivative (∇) or gradient terms. Since T_r, k_o, and L_r are constant they can be removed from the derivatives. Once removed, the T_r terms will cancel. This leaves the equation in the form,

$$\rho c_p \frac{\partial T}{\partial \tilde{t}} = \frac{k_o}{\rho_o c_{p_o} L_r^2} \nabla \cdot (k \nabla T) \tag{2.3}$$

The group of terms $k_o/\rho_o c_{p_o}$ is usually referred to as the thermal diffusivity (α_o) of the material. In order to complete the nondimensionalization and to make the final equation appear similar to equation (2.1), the nondimensional time must have the form,

$$t = \frac{\tilde{t}\alpha_o}{L_r^2} \tag{2.4}$$

which when substituted into the equation yields,

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) \tag{2.5}$$

This is the final form of the nondimensionalized governing equation. The numerical method of solution is discussed in Chapter 3.

Initial and Boundary Conditions

The governing equation is useful for calculating the temperature in the interior of a solid material or geometry. However, modeling the temperature within a geometry can only be accomplished if boundary and initial conditions are known. Boundary conditions are set on the boundary of a geometry and are necessary to determine the temperature field within the geometry. The initial condition is necessary so that the temperature field, in the geometry, at a specific time can be calculated. Since the conduction equation is solved for temperature, the initial condition is simply a temperature distribution in the geometry. The most important aspect of this temperature field is that it should approximate, as closely as possible, the actual initial conditions of the solid.

Appropriate boundary conditions are established by specifying the boundary temperature, heat flux, heat transfer rate due to convection, or the heat transfer rate due to radiation. Any of these four conditions or their combinations can be used to approximate the actual conditions experienced by materials undergoing heat transfer.

The simplest boundary condition is that of specified temperature. This means that the temperature is set on the boundary. The boundary temperature may be a function of position and/or time (e.g. $T_b = f(x, y, z, t)$), but in all of the applications of the present study the boundary conditions did not depend on time. Specified temperature boundary conditions can be used to approximate conditions of phase change or the upper limit of convection. For example, if the surface of an object were placed in an ice water bath it may be appropriate to assume that the surface is at a constant temperature. The idea that the upper limit of convection can be approximated by a specified boundary temperature will be proven when convection

boundary conditions are discussed. The nondimensional form of the fixed boundary temperature is the same as the definition used to nondimensionalize the governing equation ($T_b = \tilde{T}_b/T_r$).

The specified flux boundary condition is useful when the rate at which energy is entering or leaving a geometry is known. Many times in electronic devices such as a computer's central processing unit (CPU), the device is rated for a specific heat dissipation rate. This information can be used in modeling the electronic device or package and/or the heat sink. For example, a flux can be applied to the bottom surface of a heat sink and the temperature at that surface can be calculated. This temperature can be used to determine if the device to which the heat sink is attached is operating within the safe temperature range for that device.

A flux boundary condition can be represented by,

$$-\tilde{k}\nabla\tilde{T}|_b \cdot \hat{n} = \tilde{q}_b \tag{2.6}$$

Using the definitions in Table 2.1 the nondimensional value of the flux can be derived as,

$$-k\nabla T|_b \cdot \hat{n} = \frac{\tilde{q}_b L_r}{k_o T_r} = q_b \tag{2.7}$$

This boundary condition also includes the special case where no flux is allowed to cross a surface; as in a heavily insulated wall or in planes of symmetry. If a problem has a plane of symmetry then by using a zero flux (adiabatic) boundary condition the physical size of the geometry can be reduced by removing everything on one side of the plane of symmetry.

Convection is used as the primary cooling mechanism for most electronic components. It is therefore a requirement for any analysis method which is intended

to be used in modeling heat transfer in electronic components to have a model for convection. The basic equation for convection can be found in any heat transfer text and usually has the form,

$$-\tilde{k}\nabla\tilde{T}|_b \cdot \hat{n} = \tilde{h}\left(\tilde{T}_b - \tilde{T}_\infty\right) \tag{2.8}$$

where $\tilde{h}$ is the convective coefficient. The convective coefficient varies depending on the type of convection (e.g. natural or forced), the type of fluid and other factors. Typical values for the convective coefficient range from $2W/m^2K$ for air in natural convection to $100,000W/m^2K$ for the boiling or condensing of water [10].

By applying equation (2.7) to this equation and rearranging, the equation can be nondimensionalized as,

$$q_b = h\left(T_b - T_\infty\right) \tag{2.9}$$

Where the nondimensional form of the convective coefficient is given as,

$$h = \frac{L_\tau \tilde{h}}{k_o} \tag{2.10}$$

This nondimensional convection coefficient is known as the Biot number (Bi). The Biot number is commonly used in heat transfer calculations and is a measure of the importance of convection relative to conduction.

The convective boundary condition can be viewed as a linear combination of the specified temperature and specified flux boundary conditions. By rearranging equation (2.9) the idea of a linear combination of temperature and flux can more easily be seen.

$$T_b = \frac{q_b}{h} + T_\infty \tag{2.11}$$

This equation is a justification of the earlier statement that a specified temperature is the upper limit of convection. As the convective coefficient (h) becomes large the

temperature at the surface of the solid approaches the temperature of the surrounding fluid.

Radiation is almost always present in any heat transfer situation. The importance of radiation depends on the absolute temperature of the radiating surface and the degree to which other forms of heat transfer are present. For example, if the convective coefficient is small or the temperature difference between the surface of the object and the surroundings is large then radiation can be the dominate mechanism for heat transfer. As the convective heat transfer increases and the temperature difference between the object and the surroundings decreases, radiation plays less of a role in the overall heat transfer and may become negligible.

Model equations for radiation can vary from simple to complex depending on which factors need to be taken into account. A simple model was chosen for the first effort, although future work will take more effects into account and improve the model. The form of the equation chosen can be found in most heat transfer texts.

$$\tilde{q}_b = \epsilon\tilde{\sigma}\left(\tilde{T}_b^4 - \tilde{T}_{sur}^4\right) \tag{2.12}$$

This equation is highly nonlinear in temperature and is restricted to small, gray bodies in large enclosures.

In this case it is convenient to represent the equation in the form,

$$\tilde{q}_b = \tilde{h}_r\left(\tilde{T}_b - \tilde{T}_{sur}\right) \tag{2.13}$$

where

$$\tilde{h}_r = \epsilon\tilde{\sigma}\left(\tilde{T}_b + \tilde{T}_{sur}\right)\left(\tilde{T}_b^2 + \tilde{T}_{sur}^2\right) \tag{2.14}$$

By applying equation (2.7) to this equation and rearranging, the equation can be

nondimensionalized as,

$$q_b = h_r \left(T_b - T_{sur}\right) \tag{2.15}$$

where

$$h_r = \frac{L_r \tilde{h}_r}{k_o} \tag{2.16}$$

Note the similarities between these equations and the equations for convection (2.9). The most important difference in the equations for convection and radiation is that the convective coefficient (h) is weakly dependent on the temperature of the surface and surroundings while the radiative coefficient (h_r) is strongly dependent on these temperatures.

CHAPTER 3. METHOD OF SOLUTION FOR THE CONDUCTION EQUATION

The method of solution of the governing equation for conduction is discussed in this chapter. Since closed form (analytical) solution of a three-dimensional time accurate conduction problem is impossible except for special cases, a numerical method has been developed to solve this type of problem and will be discussed later in this chapter. The adaption of the previously discussed boundary conditions to the numerical method is also discussed. As with any numerical scheme, the geometry which can be as simple as a cube or as complex as a printed circuit board, must be broken into smaller simpler shapes or discretized. The software used to discretize the geometries will now be discussed.

Grid Generation

Many ways exist to break up or discretize a geometry, the most important aspect to remember about discretizations is that a discretization must fill the entire volume of the shape (i.e. no holes). For example, filling a space with spheres would not be valid. The most elemental shape which can be used to fill a volume is the tetrahedron. Other shapes can be used such as the cube but cubes are harder to use to fill complex shapes. It is also possible to use more than one shape, however, in this work only

tetrahedra were used as elements in the discretizations.

The discretizations or grids were generated using a computer aided design (CAD) type software package created by the Structural Dynamics Research Corporation (SDRC). The software package, known as I-DEAS, was used to generate the grids and apply the boundary conditions. This was done so that grid generation software would not have to be developed; therefore, more effort could be spent on the numerical solution of the conduction equation.

Finite Volume Formulation of the Governing Equations

In this section the governing equation for conduction is integrated and rearranged so that it may be discretized. The starting point is the nondimensional heat equation in differential form (equation (2.5)), repeated here for convenience.

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T)$$

Following a finite volume formulation, equation (2.5) is integrated over the control volume.

$$\int_{\Omega} \rho c_p \frac{\partial T}{\partial t} d\Omega = \int_{\Omega} \nabla \cdot (k \nabla T)\, d\Omega \tag{3.1}$$

At this point the control volume (Ω) is simply an arbitrary region of material, within the geometry, which can conduct heat.

Applying the divergence theorem to the right hand side of equation (3.1) allows a change from integrating through the control volume to integrating over the surface of the control volume.

$$\int_{\Omega} \rho c_p \frac{\partial T}{\partial t} d\Omega = \int_{S} (k \nabla T) \cdot d\vec{S} \tag{3.2}$$

Using the mean value theorem ρ, c_p and T can be considered constant in the control volume. That is, ρ, c_p and T are now considered averages within the control volume which represent the values of the entire control volume. Removing these terms from the integral and integrating yields,

$$\rho c_p \frac{\partial T}{\partial t} \Omega = \int_S (k \nabla T) \cdot d\vec{S} \tag{3.3}$$

Equation (3.3) is the finite volume statement which is ready to be discretized. The discretization of this equation into a numerical method for solving the three-dimensional time accurate conduction problem is discussed in the next section.

Time Accurate, Implicit Discretization

In this section the finite volume formulation of the governing equation is discretized using a Crank-Nicolson type time representation [11]. Basically, the time integration is handled as in a finite difference representation using the Crank-Nicolson scheme; that is, the fluxes are evaluated as an average of the fluxes at two time levels. Once the discretization is accomplished some of the terms will have unknowns in them due to the implicit character of the method. These terms will then be discussed in detail.

To begin the discretization, the integral over the surface of the control volume, on the right hand side of equation (3.3) can be calculated numerically if the control volume is assumed to have an unknown but finite number of sides (m_k).

$$\int_S (k \nabla T) \cdot d\vec{S} = \sum_{m=1}^{m_k} (k \nabla T) \cdot \vec{S}_m \tag{3.4}$$

In this equation $\vec{S}_m$ is the area weighted normal vector for side m of the control volume. The calculation of this term depends of the control volume used.

Using a Crank-Nicolson type of time representation for the fluxes on the right hand side to achieve second order time accuracy, equation (3.3) can be discretized as,

$$(\rho c_p \Omega)\left(T^{n+1} - T^n\right) = \frac{\Delta t}{2}\left\{\left[\sum_{m=1}^{m_k}(k\nabla T)\cdot\vec{S}_m\right]^{n+1} + \left[\sum_{m=1}^{m_k}(k\nabla T)\cdot\vec{S}_m\right]^{n}\right\} \tag{3.5}$$

At this point the governing equation is discretized and can be solved. However, many details are still hidden in the equation which can affect memory usage and efficiency. For example, the solution method (direct or iterative) has not been specified, the control volume has not been specified nor have details been provided on how some of the terms are calculated. The rest of this section will discuss these details and the final form of the equation will be presented.

Equation (3.5) can be solved using either a direct or iterative solution method for each time step. In this case the iterative method is preferable due to the fact that the problem size could grow to hundreds of thousands of nodes. With a problem size of 100,000 nodes the direct method would require the solution of a sparse matrix equation where the size of the matrix would be 100,000 by 100,000. At this point it is important to note that iterations are continued at each time step until the temperature field for that time step has stopped changing (converged).

Equation (3.5) is not in the most convenient form in terms of determining convergence. A substitution will be made so that the temperature at a node is no longer calculated but that the change in temperature at a node is calculated instead. This is convenient for checking convergence since the change in temperature will approach zero as the solution converges. The change can be accomplished by substituting

$T^{n+1} = \hat{T}^{n+1} + \delta$ into the equation, where the $\hat{T}^{n+1}$ represents the temperature at the previous iteration.

$$(\rho c_p \Omega)\left(\hat{T}^{n+1} + \delta - T^n\right) = \frac{\Delta t}{2}\left\{\left[\sum_{m=1}^{m_k}(k\nabla T_\delta)\cdot\vec{S}_m\right]^{n+1} + \left[\sum_{m=1}^{m_k}(k\nabla T)\cdot\vec{S}_m\right]^{n}\right\} \tag{3.6}$$

In this equation the unknown terms are the change in temperature (δ) terms. Some unknowns are hidden in the calculation for the gradient of temperature (∇T). Therefore the ∇T term is subscripted with a δ in order to signify that parts of the term at the $n+1$ time level are unknown. The ∇T_δ term will be discussed in detail later in this chapter, but the control volume must be specified and the method used to calculate the gradient of temperature must be discussed first.

There are two common ways of defining the control volume in tetrahedral based unstructured grid schemes. The first approach uses the tetrahedral element itself as the control volume (cell-centered scheme). The other defines the control volume as some region around a node or vertex. There are many different types of regions which are used to define the control volumes [12]. The region of interest in this research is the median dual. A median dual is a region encompassed by a polygon composed of lines in 2-D (planes in 3-D) connecting the centroids of the elements to the midpoints of the element faces. See Figure 3.1 for a two dimensional representation of the cell centered and vertex schemes. Where the two dimensional equivalent of a tetrahedron is a triangle. The most important difference between these two control volumes is where the information (material properties, temperature, etc.) is stored. In the cell-centered scheme information is stored at the center of the tetrahedral elements (cells). The median dual or vertex approach stores information at the vertices (nodes) of the

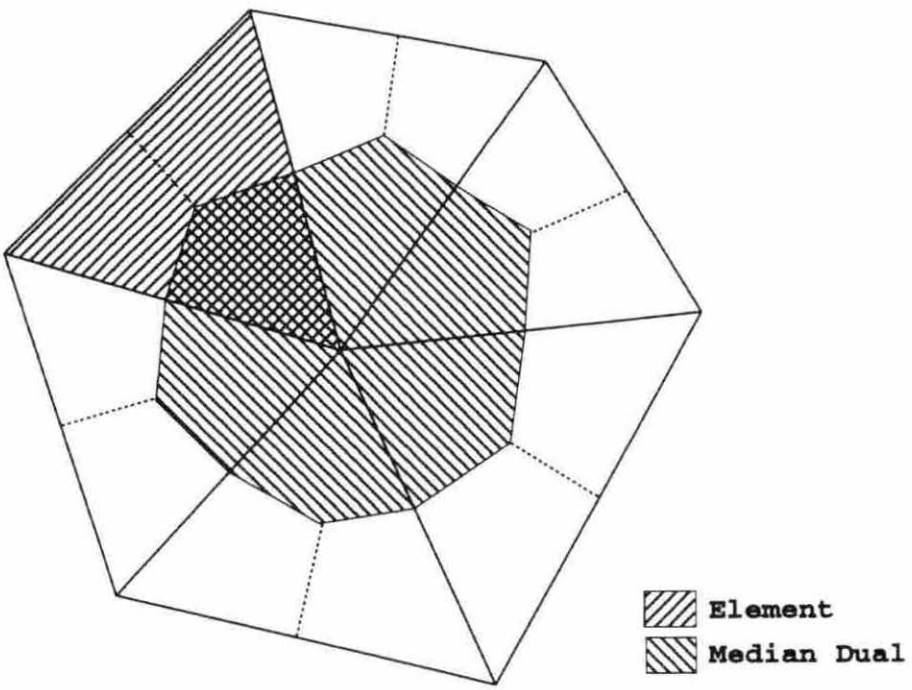

Figure 3.1: Median dual vs. element

tetrahedral elements. Other differences include the number of faces on the control volume. In the cell-centered approach there are always four faces on the cell; there are an unknown but finite number of faces on the median dual. By using the median dual as the control volume, the interpolations, which are needed in the cell-centered scheme to get information at the nodes, are avoided. For this analysis the median dual was used as the control volume.

The gradient of temperature (∇T) used to calculate fluxes crossing the control volume faces can be approximated by a simple surface integral.

$$\nabla T \approx \frac{1}{V} \int_S T d\vec{S} \tag{3.7}$$

In this equation V is the volume enclosed by the surface (S) in which the gradient is to be calculated. The gradient which is calculated is interpreted as an average value

for the volume (V).

In order to convert the integral into a summation, which can be calculated numerically, a decision must be made as to the volume in which the gradient should be known. There are several possibilities but two stand out as the most consistent. The first is to calculate the gradient in the control volume (median dual). Using the median dual would create a summation with an unknown number of parts and since each control volume can have a different number of faces, the number of parts to the summation would change for each control volume. Also interpolations would have to be performed between control volumes in order to calculate the flux on each face of the control volume. These problems can be avoided by calculating the gradient in the tetrahedral element. This creates an overlapping of the control volume and the element in which the gradient is known (see Figure 3.1). This overlapping is shown in three dimensions in Figure 3.2, where the tetrahedron connecting nodes 1, 2, 3 and 4 and containing six control volume faces is shown. This overlapping eliminates the need to interpolate between control volumes to calculate the flux passing through the face of each control volume. This can help reduce the memory requirement of a program using this method because only the node positions and element connectivity need to be stored. There is no need to store information about elements adjacent to any particular element. Using the tetrahedral element in which to calculate the gradient also produces a summation with a known number of parts which is the same for each element.

$$\nabla T = \frac{1}{V_e} \sum_{l=1}^{4} \bar{T}_l \vec{S}_l \tag{3.8}$$

The $\bar{T}_l$ term is the average temperature of the three nodes which make up face l of the tetrahedron (see Table 3.1). The V_e term is the volume of the tetrahedral element

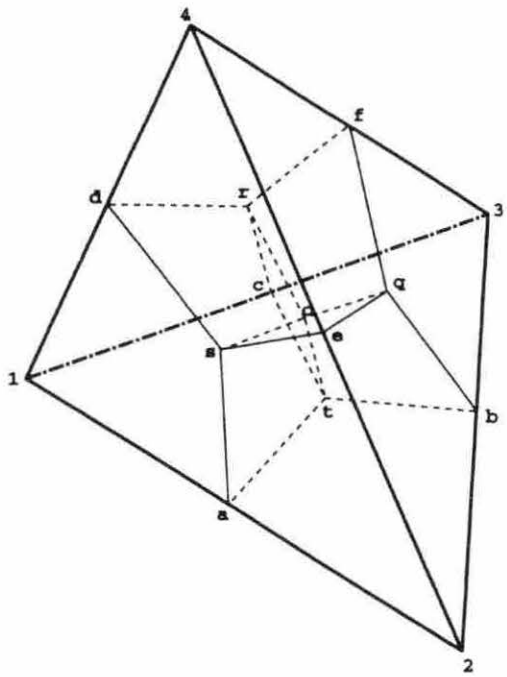

Figure 3.2: Tetrahedral element and control volume faces

used in the ∇T calculation (see APPENDIX A). At this point it is important to note that $\vec{S}_l$ in equation (3.8) is different from $\vec{S}_m$ in equation (3.6). $\vec{S}_l$ is the area weighted normal vector on side l of the tetrahedral element and $\vec{S}_m$ is the area weighted normal vector for side m of the control volume. Note the area $\vec{S}_l$ represents is a triangle and the area $\vec{S}_m$ represents is a quadrilateral (see APPENDIX B). Now that the control

Table 3.1: Face definitions

Face	Nodes
1	1,2,3
2	1,2,4
3	1,3,4
4	2,3,4

volume and gradient calculation have been discussed, how these are incorporated into the flux calculation for an unstructured grid will be discussed in the rest of this section.

Due to the unstructured nature of the grids, it is unreasonable to think that all the elements which share a common node would be numbered consecutively, or have any order at all. This means that in order to update one specific node, all the nodes and elements connected to that node would have to be known. There are two common ways to maintain this connectivity information. The first would be to generate the lists of nodes and elements connected to a specific node every time the information is needed. Generating that type of information again and again for every node in the grid would make the program very inefficient. The other method is to generate the information once for each node and store it. As the problem size grows the memory necessary to store the information can quickly grow beyond the capabilities of all but the largest computers. For these reasons a method of summing the fluxes crossing the control volume boundaries which does not require any information beyond the nodes' physical location and which nodes make up specific tetrahedra is desirable.

The method chosen is one where all of the nodes are updated when all the flux summations have been calculated and stored (accumulated). This is similar to a Jacobi iteration scheme where the data from the previous iteration is used for every node and all nodes are updated at once. This type of scheme is slower to converge than a Gauss-Seidel type approach where the nodes are updated as soon as the flux summation for a specific node is known. However, a Gauss-Seidel scheme would require all the overhead and storage problems discussed previously.

The basic idea behind the accumulation scheme is that each tetrahedron is vis-

ited and the fluxes passing through the faces of the control volumes within that tetrahedron (see Figure 3.2), are added or subtracted to the appropriate nodes in the element, depending on the sign convention used. In this discussion a node is where the information is stored for a control volume, so there is a certain amount of interchangeability. So when each of the tetrahedra which share a common node are visited, the flux summation for the common node is complete and when all the elements are visited the flux accumulation for every node in the geometry is complete.

Now with the basic idea in mind, the mathematics needed to accomplish the scheme can be discussed, remembering that the method is to be implicit. By expanding equation (3.8) and grouping the geometric terms the equation has the form,

$$\begin{aligned}\nabla T = \frac{1}{3V_e}\Big[&T_1\left(\vec{S}_1+\vec{S}_2+\vec{S}_3\right)+T_2\left(\vec{S}_1+\vec{S}_2+\vec{S}_4\right)\\ &+T_3\left(\vec{S}_1+\vec{S}_3+\vec{S}_4\right)+T_4\left(\vec{S}_2+\vec{S}_3+\vec{S}_4\right)\Big]\end{aligned} \tag{3.9}$$

In this equation all the geometric terms are known. Also, note that there are four control volumes within each element, hence the four temperatures in equation (3.9). Looking at equation (3.9) it is clear that for the implicit part of the calculation that some or all of the temperatures will be unknown.

Using Jacobi iteration for an implicit scheme, the temperature within a particular control volume is unknown while the temperatures in the surrounding control volumes are treated as known, in the sense that they are evaluated from information obtained from a previous iteration (lagged). Equation (3.9) can be split into an implicit (unknown) and an explicit (known in the Jacobi sense) vector for each node within an element. The implicit and explicit components of the gradient vector differ from node to node in an element and across the entire grid. For example, at node 1 in an element, which could be any node in the geometry, the term containing T_1

is the implicit vector and all other terms are contained in the explicit vector. This gives the implicit and explicit vectors the form,

$$I\vec{M}P = \frac{1}{3V_e}\left(\vec{S}_1 + \vec{S}_2 + \vec{S}_3\right) \tag{3.10}$$

$$\begin{aligned} E\vec{X}P = \frac{1}{3V_e}\Big[&T_2\left(\vec{S}_1 + \vec{S}_2 + \vec{S}_4\right) \\ &+ T_3\left(\vec{S}_1 + \vec{S}_3 + \vec{S}_4\right) + T_4\left(\vec{S}_2 + \vec{S}_3 + \vec{S}_4\right)\Big] \end{aligned} \tag{3.11}$$

for node 1 of an element. Notice that only the geometric terms are stored in the implicit vector ($I\vec{M}P$), this is due to the fact that the temperature at node 1 is unknown. Once these vectors are determined, they can simply be added to the implicit and explicit vectors of the node in the geometry. This is then repeated for nodes 2, 3 and 4 in the element, the only difference is that $I\vec{M}P$ and $E\vec{X}P$ have a different form, where the temperature at the node for which the vectors are being calculated is unknown. This could be interpreted as each node taking its turn being unknown while the other three nodes in an element are lagged. This is repeated until all the elements have been visited. Once all the elements have been visited, the implicit and explicit vectors have been accumulated at each node in the geometry. The accumulated vectors are part of the ∇T_δ term (see equation (3.6)) which now has the form,

$$\nabla T_\delta = \frac{1}{3V_e}\left[T_{unknown} I\vec{M}P + E\vec{X}P\right] \tag{3.12}$$

In this equation $T_{unknown}$ is the temperature to be determined at the new time level T^{n+1}. This form of the gradient can be substituted into equation (3.6) and since multiplications and dot products are distributable, the term containing ∇T_δ becomes,

$$\left[\sum_{m=1}^{m_k}\left(k\nabla T_\delta\right)\cdot\vec{S}_m\right]^{n+1} = \left[\sum_{m=1}^{m_k} T^{n+1}\left[I\right] + \left[E\right]\right] \tag{3.13}$$

where I and E are scalars due to the dot products of $I\vec{M}P$ and $\vec{S}_m$ and also $E\vec{X}P$ and $\vec{S}_m$. The I and E terms have the form,

$$I = \left[k\left[I\vec{M}P\right]\right] \cdot \vec{S}_m \tag{3.14}$$

$$E = \left[k\left[E\vec{X}P\right]\right] \cdot \vec{S}_m \tag{3.15}$$

The T^{n+1} term in equation (3.13) can be removed from the summation because it is a constant for a particular node, even though it is unknown. So the I and E terms can be accumulated simply by using the equations for I and E at every node and adding the results. This means that the $I\vec{M}P$ and $E\vec{X}P$ vectors do not need to be stored, only the I and E scalars need to be stored. Making all the substitutions into equation (3.6) completes the discretization of the governing equation leaving the form,

$$\begin{aligned}(\rho c_p \Omega)\left(\hat{T}^{n+1} + \delta - T^n\right) = \\ \frac{\Delta t}{2}\left\{\left(\hat{T}^{n+1} + \delta\right)\left[\sum I\right] + \left[\sum E\right] + \left[\sum_{m=1}^{m_k} (k\nabla T)\cdot \vec{S}_m\right]^n\right\}\end{aligned} \tag{3.16}$$

The final step in generating the equation used to solve the three-dimensional conduction problem is to solve for the change in temperature (δ),

$$\begin{aligned}\delta = \left\{\frac{1}{\frac{2}{\Delta t}(\rho c_p \Omega) - [\sum I]}\right\}\left\{\frac{2}{\Delta t}(\rho c_p \Omega)\left(T^n - \hat{T}^{n+1}\right)\right. \\ \left. + \hat{T}^{n+1}\left[\sum I\right] + \left[\sum E\right] + \left[\sum_{m=1}^{m_k} (k\nabla T)\cdot \vec{S}_m\right]^n\right\}\end{aligned} \tag{3.17}$$

As discussed earlier, in order to solve this equation in a time accurate fashion, the solution is iterated to convergence at each time step. The terms which are at the previous time step (n) are only updated at the beginning of each time step. The terms at the current time step ($n+1$) are updated at the beginning of each time step and each iteration.

Steady State, Implicit Discretization

If the steady state solution is needed without any knowledge of the transient solution, the finite volume equation can be simplified. This section will discuss the derivation of the simplified equation for steady state solution.

Reducing the finite volume equation, equation (3.3), to Laplace's equation yields,

$$\int_S (k\nabla T) \cdot d\vec{S} = 0 \tag{3.18}$$

Using the simple implicit method [11] this equation can be discretized as,

$$\sum_{m=1}^{m_k} (k\nabla T) \cdot \vec{S}_m = 0 \tag{3.19}$$

Substituting the right hand side of equation (3.13) for this equation gives an equation of the form,

$$T^{n+1} \left[\sum I\right] + \left[\sum E\right] = 0 \tag{3.20}$$

Substituting $T^{n+1} = \hat{T}^{n+1} + \delta$ and solving for δ gives,

$$\delta = -\left[\hat{T}^{n+1} + \frac{[\sum E]}{[\sum I]}\right] \tag{3.21}$$

where n is an iteration counter as opposed to a time step counter.

Using this form of the equation can give steady state results much faster than using the transient method and stepping in time until steady state is reached. Less work is required per iteration due to the reduced number of terms in the simplified equation. Also, less iterations are needed to reach steady state.

Numerical Boundary Conditions

Using the equations discussed in the previous section, the conduction in the interior of a geometry can be modeled. This is of little use without boundary conditions

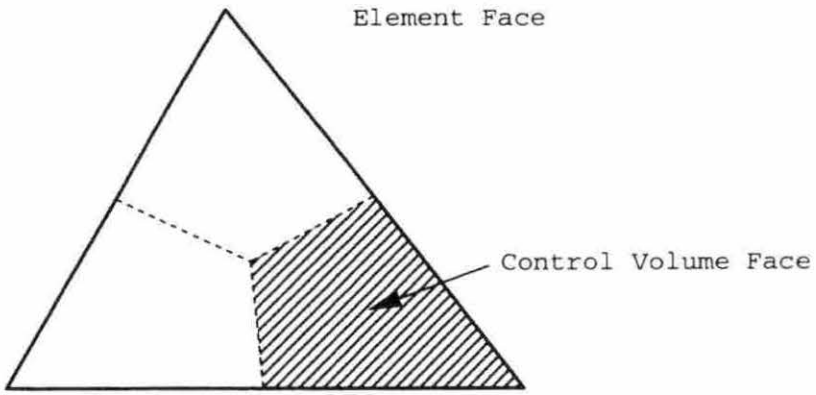

Figure 3.3: Tetrahedral faces vs. control volume faces

to propagate temperature information into the solid. In this section the method for incorporating the boundary conditions into the accumulation is discussed.

The constant temperature boundary condition is the simplest to add to the accumulation due to the fact that there is nothing to add, since the boundary temperatures do not change for this condition. For constant temperature boundary conditions the change in temperature calculated for each specified temperature boundary node, during each iteration, is simply ignored so that the temperature at the node does not change.

The constant flux boundary condition implies the fluxes are known on the boundary. Since the accumulation is really an energy accumulation, the flux must be multiplied by the area of the control volume face through which it is passing. The grid generator applies the boundary conditions on the face of the tetrahedral element, not the control volume face. The relationship between the two is shown in Figure 3.3. It can be shown that the area of each control volume face, on an element face, is one third that of the element face. This means that the energy passing through the element face can be calculated and divided by three, then added to each of the three

nodes (control volumes) on that element's face. The known fluxes are explicit and therefore are only added to the explicit part of the control volumes accumulation.

The convection and radiation boundary conditions are handled in a very similar manner. Recall that the equations for both are very similar, $q = hA(T - T_\infty)$ for convection, and $q = h_r A(T - T_{sur})$ for radiation. The only difference in how these equations are handled is how the h and h_r terms are calculated. For convection, the convective coefficient (h) is given. For radiation the h_r term must be calculated. This term is nonlinear in temperature so it is lagged, meaning that the temperatures needed to calculate h_r are obtained from values computed at the previous iteration. This lagging of the h_r term has the effect of linearizing the radiation equation. Otherwise these equations are handled in exactly the same way. They have an implicit or unknown part as well as an explicit part. The implicit part is the temperature (T_b) at the surface. While the explicit part is the temperature of the fluid (T_∞) or the temperature of the surroundings (T_{sur}). So the part of the equation added to the implicit (I) term has the form hA or $h_r A$, and the part of the equation added to the explicit (E) term has the form hAT_∞ or $h_r AT_{sur}$.

These boundary conditions can be used for both the transient and steady state conduction solvers. These solvers are validated in the next chapter using several test cases. The test cases are chosen so that analytical solutions to the problems can be derived. Comparisons of these analytical solution to the numerical solutions provide the validation for the numerical program.

CHAPTER 4. RESULTS

The programs and methods used to view or plot the results generated by the numerical program are discussed. Selected results from the steady state and transient formulations are discussed and any limitations of the methods are mentioned. Emphasis is given to validating the accuracy of the numerical program by using cases with known analytical solutions and comparing the numerical results to the analytical solutions. The usefulness of the numerical program for modeling three-dimensional conduction in complex geometries is discussed and an example of a solution in a geometry commonly found in electronic cooling is provided.

Post Processing

This section will briefly discuss the software used and developed for retrieving information about, viewing and plotting the numerical results. This is commonly called post processing.

Due to the fact that unstructured grids are not ordered, it is almost impossible to get any but the most rudimentary information from viewing the raw numerical output from the numerical solver. This requires that some form of post processing software be used to extract the necessary information and present that information in a useful form.

I-DEAS can be used to view the numerical results in a qualitative sense. By formatting the output from the numerical program in a form that I-DEAS is able to process, its contour plotting capabilities can be used to present a three-dimensional view of the temperature field in a geometry. Other viewing possibilities include cutting planes which can be used to slice the geometry to give visual information about the temperature field in the interior of a geometry. This type of information can be used to quickly see local hot spots and other potential problems within a design. This can accelerate the iterative design process by allowing changes to be made to the design, calculations to be run and results viewed quickly and easily. For more exact information, such as the temperature at some point in the geometry, two programs have been written to calculate temperature information from the output of the numerical solver.

One of these programs is used to find the temperature at a point or list of points in the geometry. The points do not have to be nodes, but they must be within the domain of the geometry. The first step in finding the temperature at a point in the geometry is to find the tetrahedral element in which the point is contained. This is accomplished by a simple search of the tetrahedra in the grid. Once the element containing the point is found the gradient of temperature is calculated within the tetrahedron using equation (3.8). This equation assumes the temperature gradient within the tetrahedron is constant. This assumption is used in the numerical solver so should give answers which are as accurate as the solution. Knowing the gradient of temperature, the position of the point at which the temperature is to be calculated, and the position and temperature of one of the nodes the temperature at the point can be calculated. The position of the point and its temperature are output to a file,

which can be used to generate plots of temperature profiles within a geometry.

The other program is used to read the output files from a transient numerical solution. The transient solver generates one solution file for each time step in the calculation. The post processing program reads each of the files, finds the temperature at a point, as described in the previous paragraph, and generates a file with a list of times and the temperature at the point for each time. This file can be used to generate transient temperature plots for a point within the geometry.

These pieces of software give easy to understand information about the results from the numerical program and are used in the rest of this chapter to produce plots of the selected results.

Steady State Results

The test cases which were run to validate the steady state numerical scheme are discussed in this section. The use of simple geometries and boundary conditions allowed comparisons to be made between cases as well as with some analytical results.

Adiabatic and constant wall temperature boundary conditions

The first two test cases are related and therefore will be discussed together. The geometry for the first case is a cube with fixed temperature boundary conditions. Figure 4.1 is a view of the grid used for this case. The boundary conditions are set so that one side of the cube is at $1000K$ and the other five sides are at $500K$. These are nondimensionalized using a reference temperature of $1000K$. The second case is half of the cube mentioned above where the $1000K$ face is one of the faces cut. The boundary conditions are similar except that the surface exposed by the cut

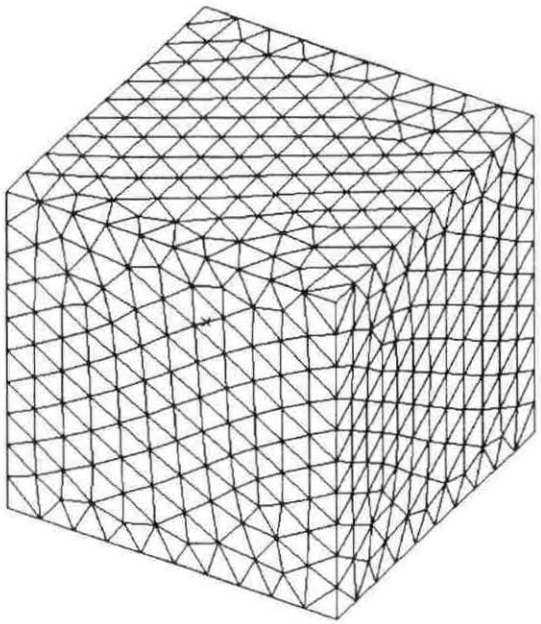

Figure 4.1: Grid generated in cube

Table 4.1: Steady state cube results

Description	Nodes	Elements	Iterations	CPU (m:s)
Constant	1626	7471	196	0:50.41
Adiabatic	907	3898	221	1:02.53

has adiabatic boundary conditions applied.

Table 4.1 shows the CPU time required for each of the problems. This time includes the initialization of the program as well as all the file accesses. "Constant" describes the cube with all constant wall temperatures and "Adiabatic" is the block with adiabatic boundary conditions on one face. Note that the adiabatic case took more iterations to converge to the same level as the constant case. See Figure 4.2 for a convergence plot of the two cases. The slower convergence of the adiabatic case

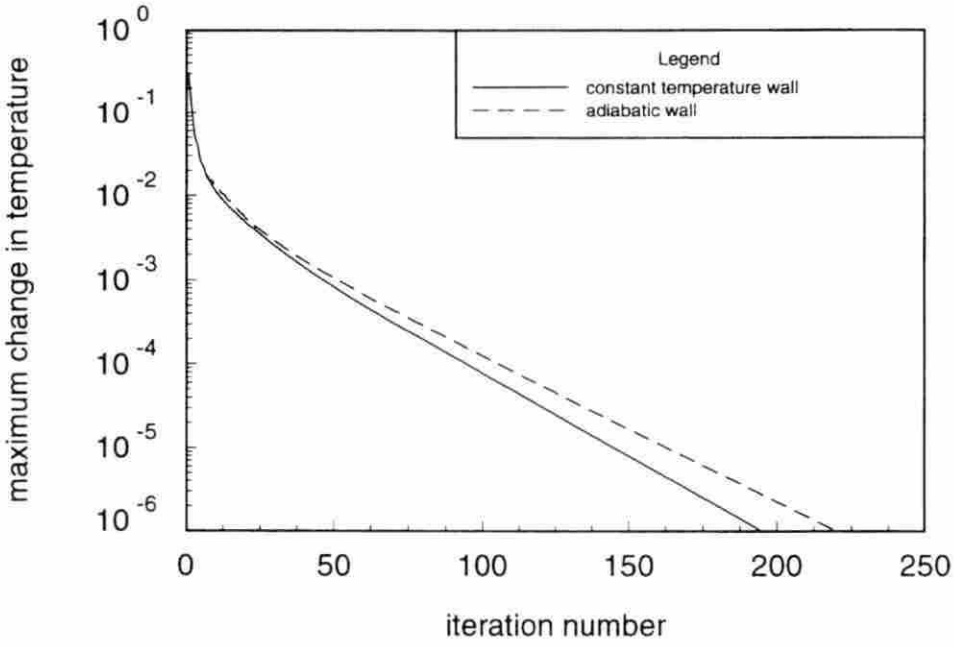

Figure 4.2: Convergence of constant temperature and adiabatic cube

is due to the fact that adiabatic or symmetric boundary conditions tend to converge like the full problem. However, the effort per iteration is less due to the reduced number of tetrahedra which must be accumulated each iteration. Also note that this problem has two planes of symmetry although only one was utilized for the test.

Using the idea that Laplace's equation is simply an average, it makes sense that the center of the constant temperature cube and the center of the adiabatic face have a temperature that is equal to the average of the six face temperatures of the cube. Therefore the temperature at the center of the cube is expected to be $583.333K$. The temperature calculated at the center of the cube by the numerical solver was $583.931K$, and the temperature at the corresponding point on the adiabatic face was $583.224K$. These results show a very close agreement with the expected temperature at the center of the cube.

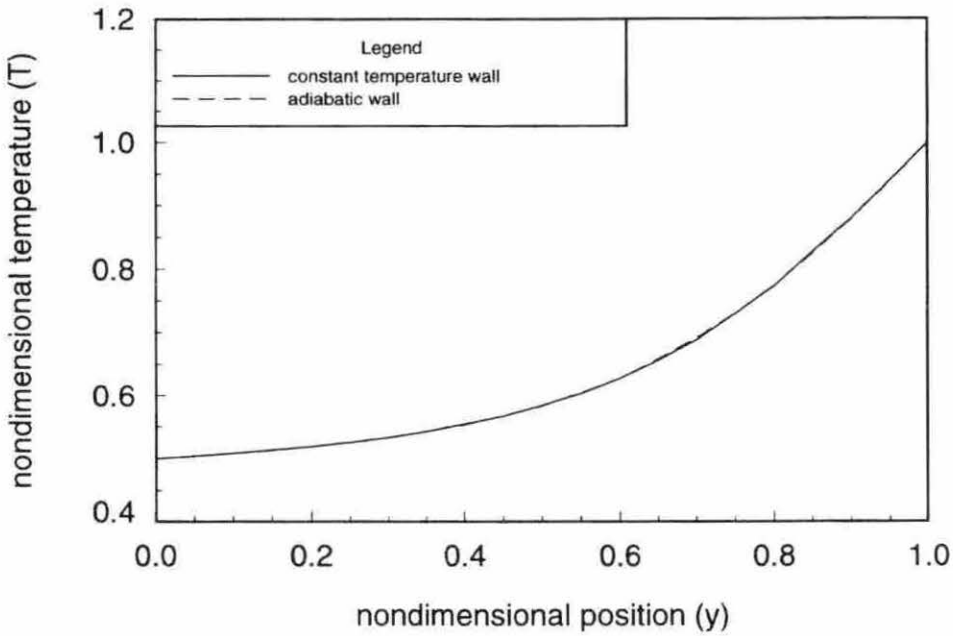

Figure 4.3: Temperature profile through the center of a cube

Figure 4.3 shows the temperature profile from the center of the top to the center of the bottom of the constant temperature cube, that is the center of the face at $1000K$ to the center of the opposing face. This profile is compared to the temperature profile down the center of the adiabatic face from the same two opposing faces because the adiabatic surface of the half cube is a plane of symmetry for the problem. The figure shows very good agreement for the two cases.

Convecting fin with adiabatic tip

The problem of a thin fin undergoing convection heat transfer with negligible heat loss at the tip was solved using the steady state numerical scheme. This problem

has an analytical solution given by the following equation [13],

$$\frac{\theta(x)}{\theta_0} = \frac{T(x) - T_\infty}{T_0 - T_\infty} = \frac{\cosh m(L - x)}{\cosh mL} \tag{4.1}$$

where

$$m^2 = \frac{hP}{Ak} \tag{4.2}$$

and where the fin is assumed to be sufficiently thin so that the temperature effectively only varies along the fin. In these equations the cross-sectional area A, the perimeter P, the heat transfer coefficient h, the thermal conductivity of the material k, and the length of the fin L are known and constant.

The steady state numerical solver was used to solve the temperature profile in a fin with a square cross-sectional profile. The fin used had a side length of $0.01m$ and was $0.1m$ long. The thermal conductivity of the fin was chosen to be $400W/mK$ and the heat transfer coefficient was chosen to be $100W/m^2K$. The temperature at the base of the fin was fixed at $350K$ while the temperature of the surrounding fluid was set to $300K$. The grid used was relatively coarse with only 171 nodes and 483 elements. This created a grid with essentially a line of nodes down the middle of the geometry as the only interior nodes.

The convergence of the steady state numerical program is shown in Figure 4.4, where the convergence criteria was the maximum nondimensional temperature change between iterations. The numerical and analytical solutions are shown in Figure 4.5, where the temperature and distance are given in nondimensional form (T/T_r) and (x/L_r) respectively. Note that the reference temperature and reference length are $350K$ and $0.1m$ respectively.

The final numerical solution was very close to one-dimensional. Even though the grid was coarse, the solver provided a robust and accurate result.

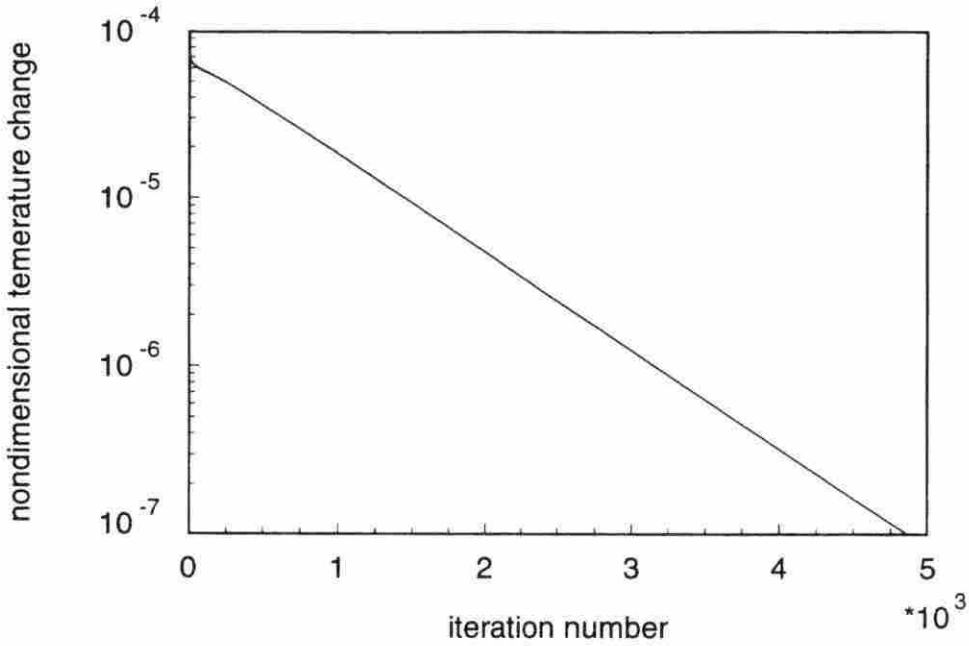

Figure 4.4: Convergence of convecting fin with adiabatic tip

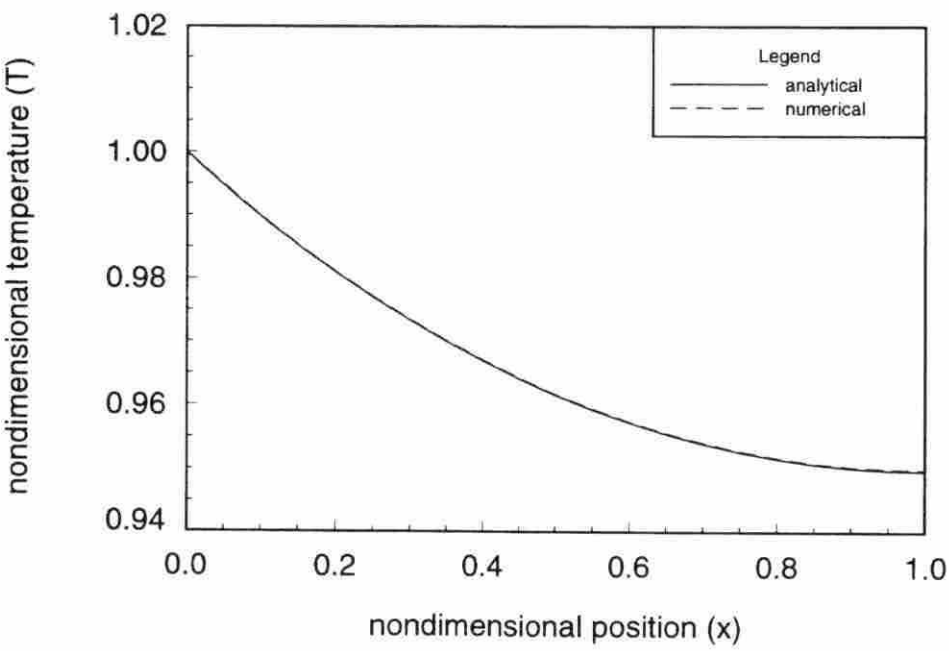

Figure 4.5: Temperature profile along convecting fin with adiabatic tip

Radiating fin with adiabatic tip

In this section the solution of the temperature profile in a thin fin radiating to free space is discussed. If it is assumed that there is no internal generation of energy within the fin, the governing equation can be expressed as [14],

$$\frac{d^2\theta}{d\eta^2} - \left\{ \frac{\left(\frac{b}{l} - \frac{e}{l}\right)}{\delta(\eta)} \right\} \frac{d\theta}{d\eta} - \left\{ \frac{2\epsilon\sigma l T_w^3}{k\delta(\eta)} \right\} \theta = 0 \tag{4.3}$$

where

$$\delta(\eta) = \frac{e}{l} + \left(\frac{b}{l} - \frac{e}{l}\right)(1 - \eta) \tag{4.4}$$

and the fin is assumed sufficiently thin so that the temperature varies only along the fin. In these equations e, b, and l are the height at the end, the height at the base, and the length of the fin, respectively. The η term is simply (x/L_r) and θ is (T/T_r). Due to the non-linearity of equation (4.3) the solution used for comparison was obtained by using a fourth order Runge-Kutta procedure. The values of the constants used in the fin problem were: $L = 0.25m$, $b = e = 0.01m$, $\epsilon = 0.85$, $T_w = 700K$, and $k = 150W/mK$. In order to compare the solution obtained from equation (4.3) to the solution calculated using the unstructured numerical solver the unstructured numerical solver was run on the fin problem with an initial condition of $500K$. The grid generated for the numerical solver contained 510 nodes and 1504 elements. This grid is more refined than the grid used for the convection test case.

Convergence for the fin problem with radiative boundary conditions was slow due to the fact that the boundary conditions had to be linearized with respect to temperature. This tended to slow convergence because part of the temperature influences had to be lagged for each iteration. Convergence to 10^{-9} took approximately 50000 iterations.

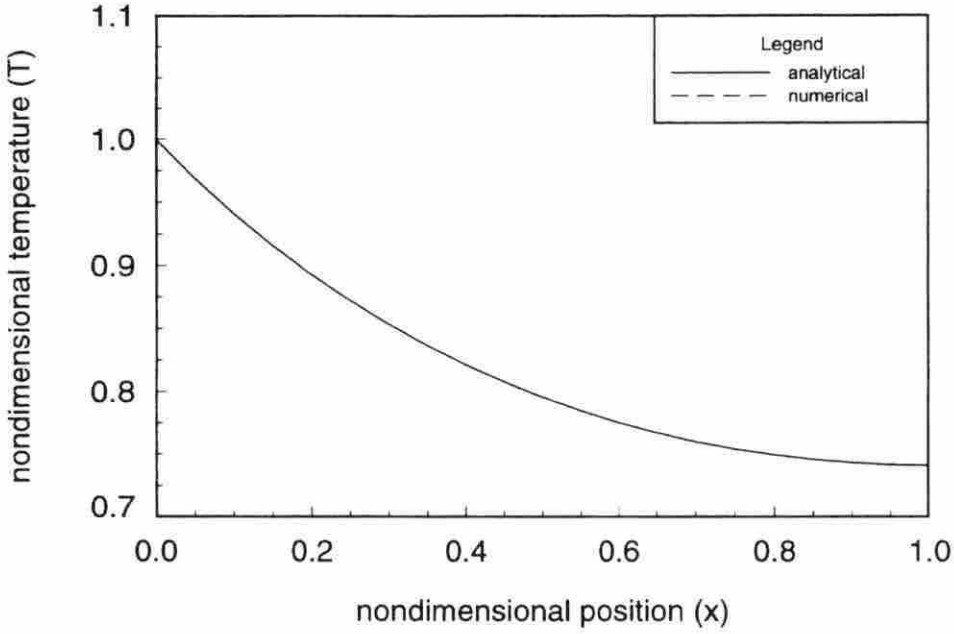

Figure 4.6: Temperature profile along radiating fin with adiabatic tip

Figure 4.6 shows the comparison of the numerical finite volume solution to the Runge-Kutta solution for the radiating fin problem. The finite volume solution was very close to one-dimensional and the solutions show good agreement for this case, therefore it was concluded that the radiative boundary condition treatment was correct.

The next section discusses some selected results for the transient numerical scheme.

Transient Results

The transient cases which were run to test the accuracy of the numerical solver and demonstrate its capabilities are discussed in this section. The selected test cases

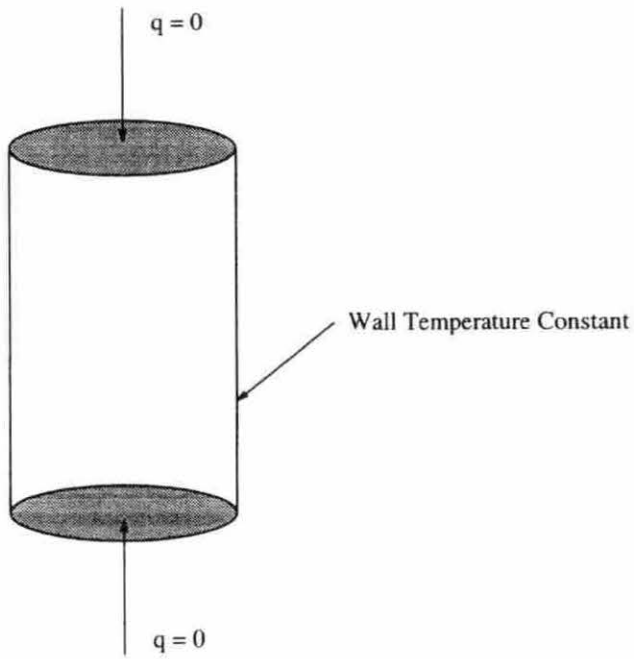

Figure 4.7: Geometry used for infinite cylinder problem

have simple geometries and known analytical solutions. Since the boundary condition formulation for the steady state and transient solvers are the same and the boundary condition formulation has been verified, this section will focus on the time dependent part of the governing equations.

Infinite cylinder with constant wall temperature

The first transient test case for the numerical solver was chosen to be an infinite cylinder with a constant wall temperature and an initial temperature different from the wall temperature. This test case was chosen for its simple geometry and boundary conditions (See Figure 4.7) in order to test the accuracy of the transient solution generated by the numerical program. This problem is a classic one-dimensional problem with a known analytical solution, which can be given in the form of the following

summation [15],

$$\tilde{T}(r,t) = \left(\tilde{T}_o - \tilde{T}_\infty\right) \sum_{n=1}^{\infty} \frac{2\exp(-\alpha\lambda_n^2\tilde{t})J_0(\lambda_n\tilde{r})}{\beta_n J_1(\beta_n)} + \tilde{T}_\infty \tag{4.5}$$

In this equation $\beta_n = \lambda_n \tilde{R}$, which represents the nth zero of the characteristic equation. The characteristic equation is a Bessel function of the first kind of order zero ($J_0(\lambda r)$). The value of a Bessel function of the first kind of order μ can be calculated using the following equation [16].

$$J_\mu(\lambda r) = \left(\frac{\lambda r}{2}\right)^\mu \sum_{m=0}^{\infty} \frac{(-1)^m}{m!(\mu+m)!} \left(\frac{\lambda r}{2}\right)^{2m} \tag{4.6}$$

The values of β_n can either be calculated by finding the roots of the Bessel function numerically or by finding them in a mathematical reference [17]. The data which come from these equations can be nondimensionalized using the same definitions used to nondimensionalize the governing equation.

For this case the properties of the material were set to that of pure copper where $\rho = 8933 kg/m^3$, $c_p = 385 J/kgK$ and $k = 400 W/mK$, from these the thermal diffusivity (α) can be calculated. The boundary condition on the cylinder wall was set to a constant temperature of $500K$ and the initial condition of the interior was set to $100K$. The reference temperature was set to the wall temperature which when nondimensionalized gives 1.0 as the wall temperature and 0.2 for the initial interior temperature. The reference length was set to the radius of the cylinder ($0.05m$). These are all the geometric values and material properties needed to run the numerical solver.

Since generating a grid for an infinitely long cylinder would be difficult at best, the grids were generated using a finite cylinder with adiabatic or zero flux boundary conditions on the ends. This mimics the one-dimensional nature of the infinite

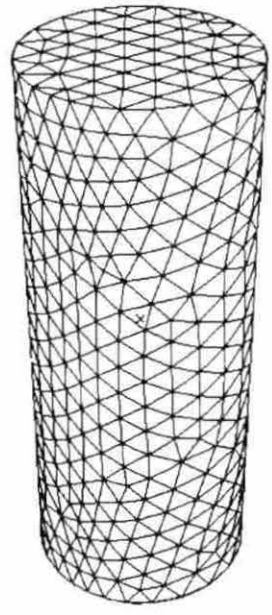

Figure 4.8: Grid for infinite cylider problem

cylinder by not allowing heat to flow along the length of the cylinder. The three different grid sizes used each approximately doubled the number of nodes in the previous grid (See Table 4.2). Figure 4.8 show the grid which was generated using 1495 nodes. All three cases were run to steady state using an SGI Indigo^2. It is easy to see from the amount of CPU time used, that doubling the nodes more than doubles the computational effort needed to calculate a solution.

Table 4.2: Information for constant wall temperature problem

Nodes	Elements	Δt	Time Steps	CPU (m:s)
752	3185	0.01	201	2:27.69
1495	6822	0.01	196	6:53.00
3144	15217	0.01	193	22:01.44

The size of the time step was chosen so that the accuracy of the transient solution would not be affected by the truncation error due to a large time step. If the time step is too large the temperature distribution as a function of time will not be correct; however, the solution will still converge to the correct steady state temperature field. This indicates that there is a limit as to how large the time step can be. This time step limit changes with the geometric and boundary conditions. The more refined the grid and the more complex the boundary conditions, the smaller the time step size must be.

Figure 4.9 shows the comparison of the numerical and analytical solutions for the temperature of the cylinder from the initial condition to steady state. The temperature is given at the center ($r = 0$) of the cylindrical geometry. Figure 4.10 shows less of the time scale than Figure 4.9. It is easy to see that grid refinement produces an improvement in the transient solution. The improvement is significant for the first refinement (752 to 1495) but there is less improvement for the second refinement. Successive refinements will show less improvement until there is no improvement in the solution. Note that the solution on all three grids is the same at steady state. The errors in temperature at times close to zero are due to the fact that the instantaneous application of a constant temperature at the boundary is a harsh condition. Such instantaneous changes in temperature are very difficult to duplicate physically. The errors can be reduced somewhat by reducing the time step. However, reducing the time step will increase the CPU time required for the steady state solution to be reached.

Figures 4.11 and 4.12 show the temperature along a line from the center to the outer edge of the cylinder. The direction of the line from the center to the edge of the

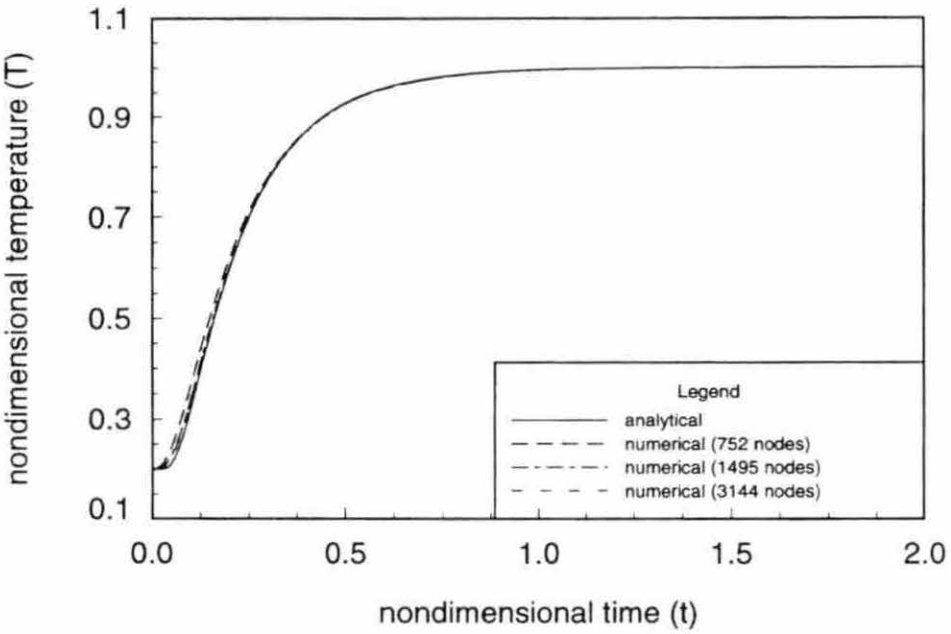

Figure 4.9: Transient temperature at the center of the cylinder

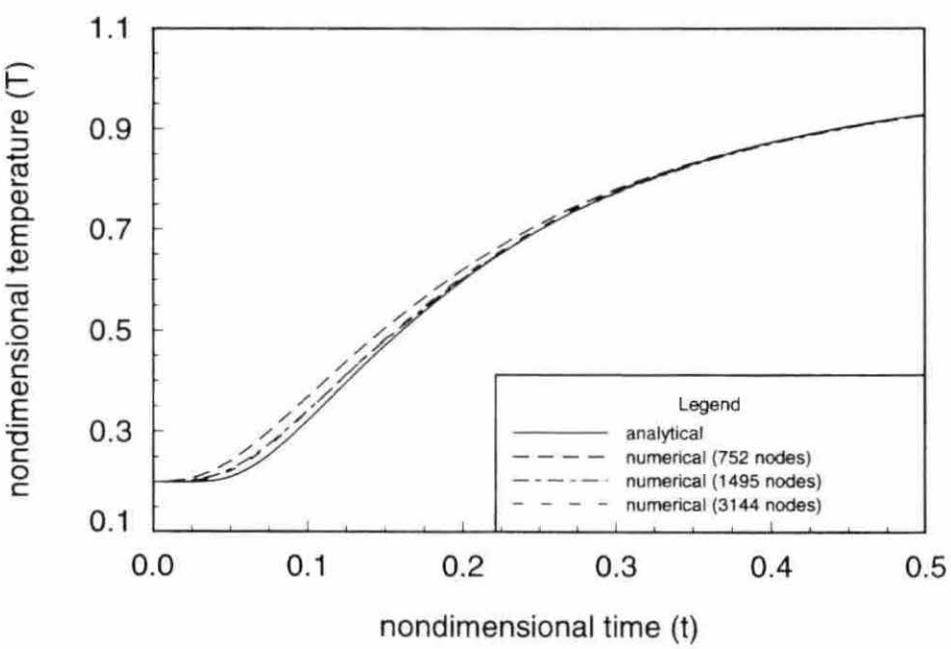

Figure 4.10: Transient temperature at the center of the cylinder

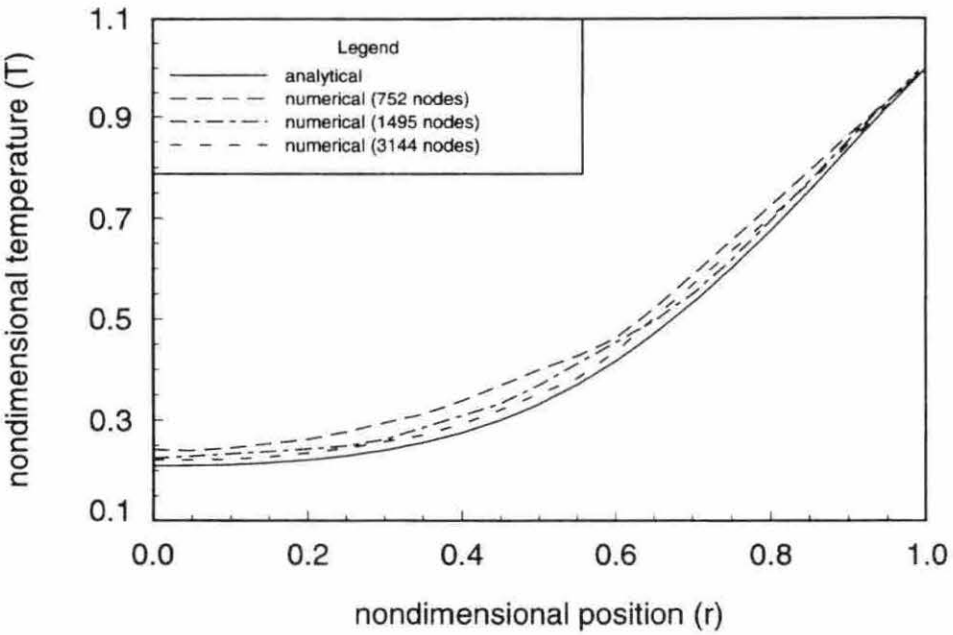

Figure 4.11: Temperature along cylinder radius (t=0.05)

cylinder is immaterial since the problem is symmetrical in the radial direction. The figures show the temperature distribution at $t = 0.05$ (1.075 seconds) and $t = 0.5$ (10.75 seconds).

The results of the constant wall temperature test case show a good agreement with the analytical solution. They show that grid refinement improves both the spatial and temporal accuracy and that the errors tend to decrease as steady state is approached.

Infinite cylinder with convection

An infinite cylinder with convective boundary conditions was used as a test case to demonstrate the transient accuracy of the numerical solver for convective boundary conditions. This problem is very similar to the constant temperature case discussed

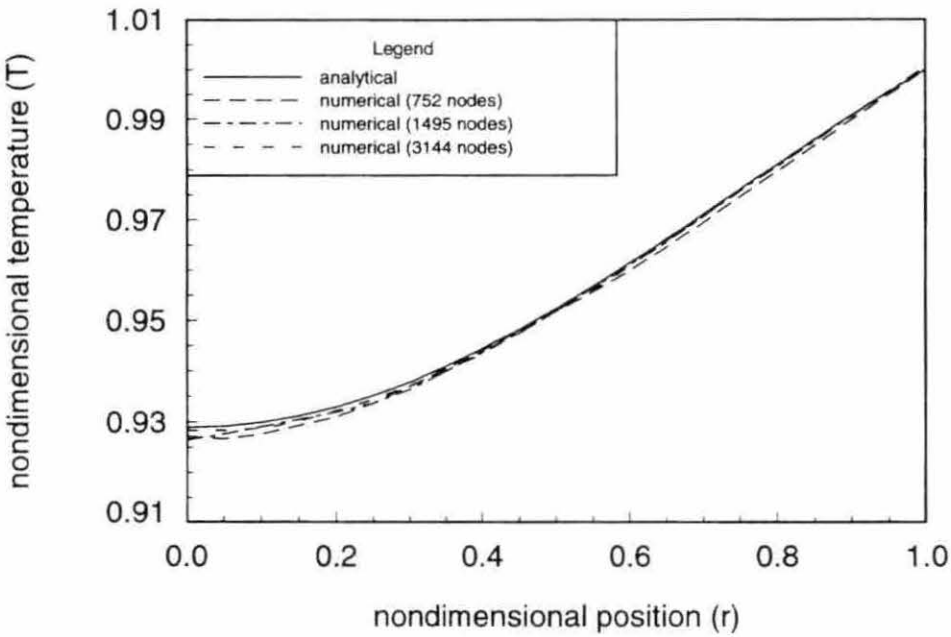

Figure 4.12: Temperature along cylinder radius (t=0.50)

previously. The convective case is also a classic one-dimensional problem with a known analytical solution, which can be given in the form [18],

$$\tilde{T}(r,t) = \left(\tilde{T}_o - \tilde{T}_\infty\right) 2Bi \sum_{n=1}^{\infty} \frac{\exp(-\alpha\lambda_n^2 \tilde{t}) J_0(\lambda_n \tilde{r})}{(\lambda_n^2 R^2 + Bi^2) J_0(\lambda_n R)} + \tilde{T}_\infty \tag{4.7}$$

where Bi is the Biot number (hR/k). In this equation $\lambda_n R$ represents the nth zero of the characteristic equation, which for this problem is [18],

$$(\lambda_n R) J_1(\lambda_n R) - Bi J_0(\lambda_n R) = 0 \tag{4.8}$$

The zeros of this function were calculated numerically using equation (4.6) to calculate the value of the Bessel functions.

The geometry for this problem is the same as the constant wall temperature problem (See Figure 4.7); only the wall boundary conditions were changed. The material properties for this problem were chosen to be: $\rho = 8933 kg/m^3$, $c_p = 385 J/kgK$

Table 4.3: Information for convective wall problem

Nodes	Elements	Δt	Time Steps	CPU (h:m:s)
752	3185	0.001	36781	8:58:03.8
1495	6822	0.001	36580	22:09:44.1

and $k = 400W/mK$. The initial condition was chosen as $T_o = 100K$ and the reference temperature was chosen to be the same as the fluid temperature ($T_r = T_\infty = 500K$). The reference length was set to $R = 0.05m$ and the convective coefficient was chosen as $\tilde{h} = 500W/m^2K$.

An HP 9000/730 workstation was used to run the unstructured numerical program on the two grids. The results from these calculations are given in Table 4.3. Note that the time step was decreased an order of magnitude from the constant wall temperature problem. This was done in order to maintain the accuracy of the solution at each time step. As discussed earlier, the more complex the boundary conditions the smaller the time step must be in order to maintain the accuracy of the transient solution. The decrease in time step gave a corresponding increase in the number of time steps and the amount of CPU time required to reach steady state.

Figure 4.13 shows the comparison of the numerical and analytical solutions for the transient temperature at the center ($r = 0$) of the cylindrical geometry. Figures 4.14 and 4.15 show the temperature profile along the radius of the cylinder. Just as for the constant wall temperature case, the direction of the line from the center to the edge of the cylinder is immaterial since the problem is symmetrical in the radial direction. The figures show the temperature distribution at $t = 1.0$ (21.5 seconds) and $t = 10.0$ (215 seconds).

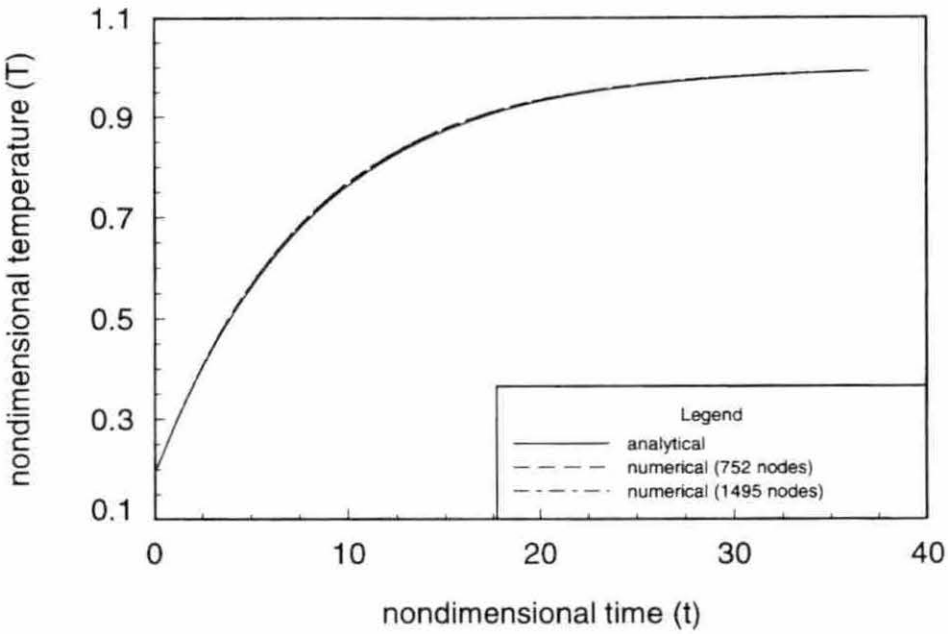

Figure 4.13: Transient temperature at center of cylinder

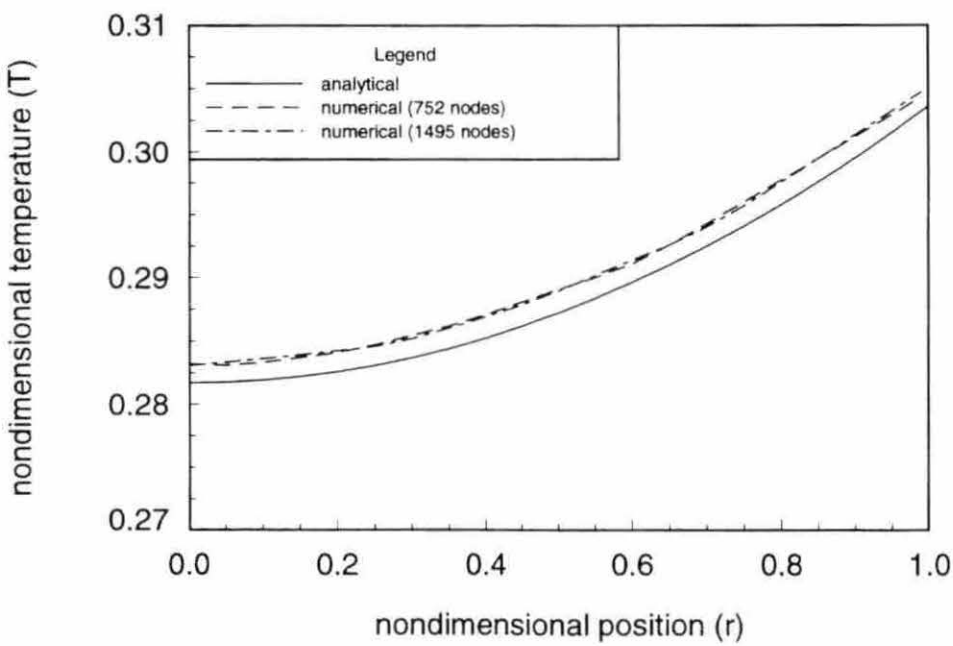

Figure 4.14: Temperature along cylinder radius (t=1.0)

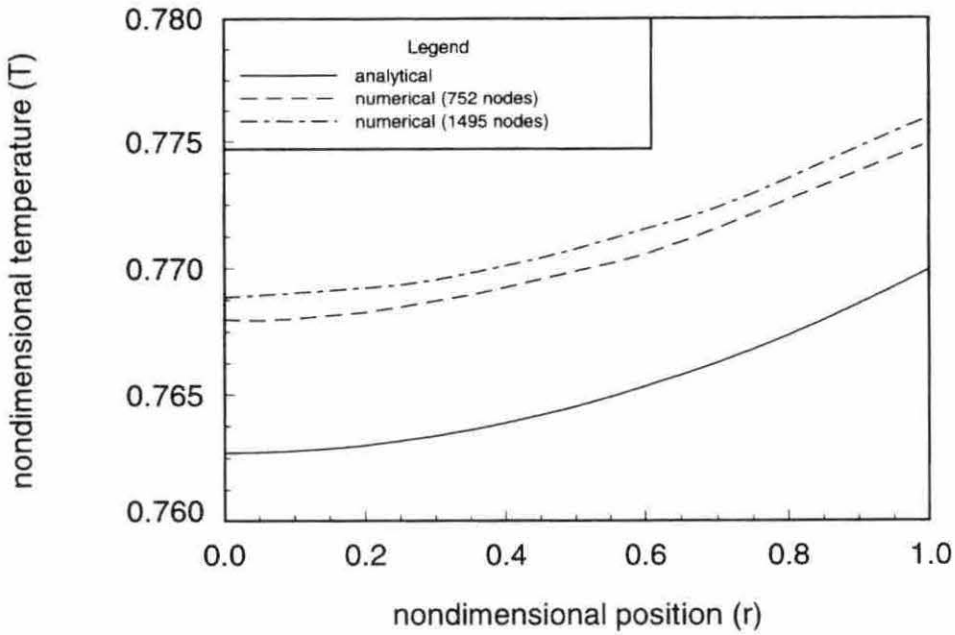

Figure 4.15: Temperature along cylinder radius (t=10.0)

There was very good agreement between the analytical and numerical solutions of this problem. The maximum error in the numerical solution is approximately one percent above the analytical solution.

Complex Geometry Results

In this section the solution of the temperature field within a complex geometry, similar to those often encountered in electronic cooling, is discussed. The problem is steady with convective, adiabatic, and fixed flux boundary conditions. Since this is a three-dimensional problem with a complex geometry and complex boundary conditions, an analytical solution would be difficult if not impossible.

The geometry shown in Figures 4.16 and 4.17 is one fourth of the total geometry

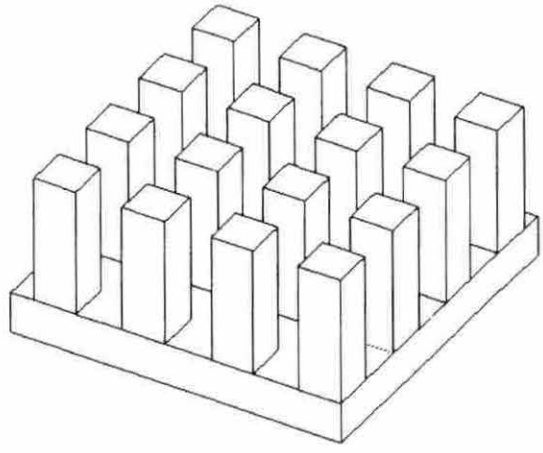

Figure 4.16: Heat sink geometry (perspective view)

for the heat sink. Adiabatic boundary conditions are used along lines of symmetry to reduce the overall size of the problem. The actual problem being solved is an 8×8 array of pins on a base with a pedestal one fourth the size of the base centered underneath the base. In this geometry the fin width and height are $3.3mm$ and $10.0mm$ respectively. The distance between fins is $3.6mm$. The base thickness and length are $3.2mm$ and $51.6mm$ respectively and the pedestal height is $1.6mm$ and the pedestal width is $25.8mm$. The material properties were set as to approximate aluminum where the thermal conductivity (k) was set to $160.0W/mK$. The reference temperature and length were set to $295.0K$ and $0.0258m$ respectively.

The grid generated in the heat sink geometry had 7938 nodes and 32939 tetrahedral elements (see Figure 4.18). The boundary conditions were chosen to approx-

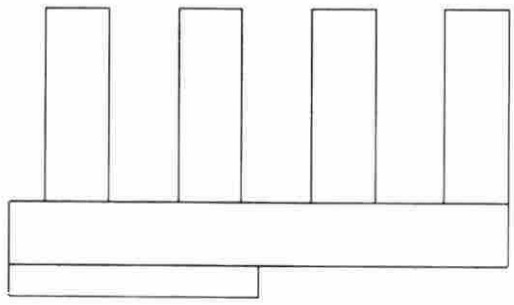

Figure 4.17: Heat sink geometry (side view)

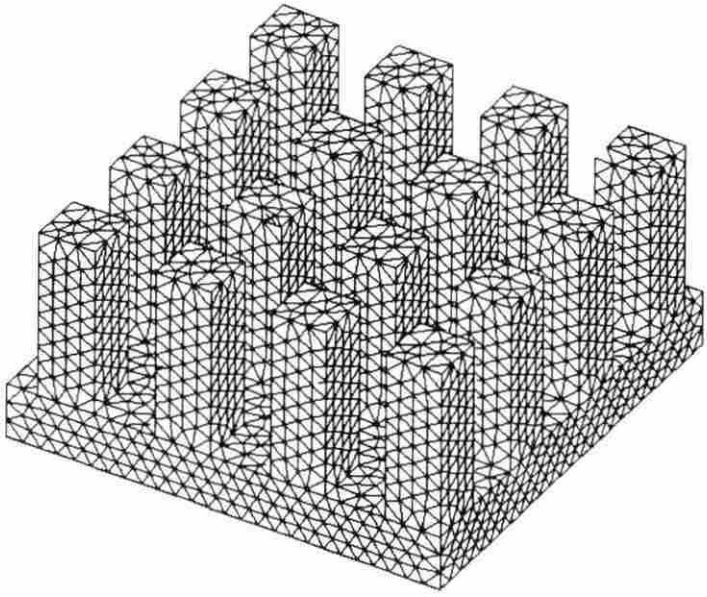

Figure 4.18: Heat sink grid

imate conditions which such a device might be subjected to in service. The bottom of the pedestal had a fixed flux of $15500W/m^2$ applied. This power is equivalent to approximately $10W/in^2$. The sides of the pedestal and the underside of the heat sink base as well as the two sides of the base where symmetry is used to reduce the problem size had adiabatic conditions applied. The rest of the geometry had convective boundary conditions where the heat transfer coefficient and the fluid temperature were set to $100.0W/m^2K$ and $295K$ respectively. An initial temperature of $295K$ was applied throughout the heat sink.

It was found that the solution had to be converged to 10^{-9} in order for the temperature field to stop changing with successive decreases in the convergence tolerance. In other words each successive order of magnitude decrease in convergence tolerance starting with 10^{-6} would change the temperature profile in the geometry significantly enough that another order of magnitude reduction in tolerance was warranted. It was found that with each order of magnitude reduction the differences in the temperature field would be less than the previous reduction. This method was repeated until the difference was reduced to an acceptable level.

The solution is shown in Figures 4.19 and 4.20 for the temperature field on the surface. The total temperature difference between the hottest and coolest points were approximately $4K$. As shown in Figure 4.19 the hottest point in the heat sink is at the center of the bottom of the pedestal. While the coolest is a the top of the fin furthest from the heat source.

The results shown are typical of the results which can be achieved using the combinations of programs discussed in this thesis. This type of information can help designers produce efficient thermal management devices.

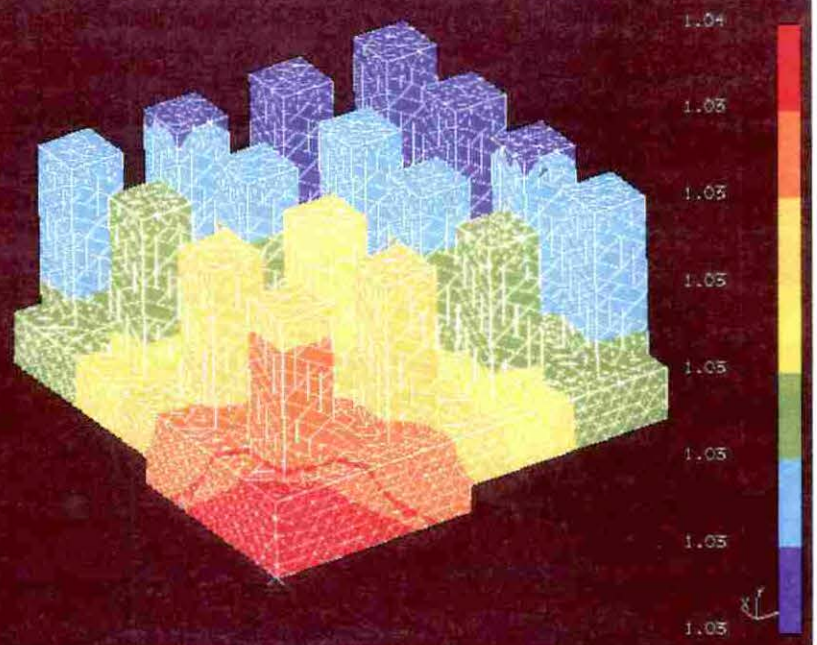

Figure 4.19: Surface temperature (perspective view)

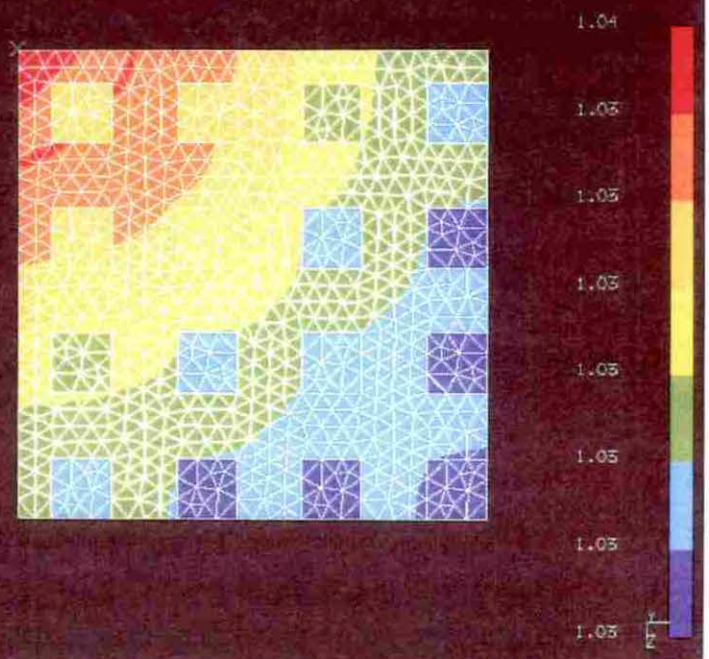

Figure 4.20: Surface temperature (top view)

CHAPTER 5. CONCLUDING REMARKS

An algorithm to solve the conduction equations for arbitrary three-dimensional geometries for both steady and unsteady conduction problems was developed. The algorithm was used to write a computer program which can be used to calculate the temperature field in a material. This software can be used to solve problems encountered in the electronic cooling industry. The use of complex geometries and the need for cost effective design tools are a must in order to stay competitive in the electronics industry. The use of the conduction software can reduce the overall time required to generate a workable design for maintaining electronic components at acceptable temperatures.

The scheme uses unstructured linear tetrahedral grids generated by a preprocessor known as I-DEAS. I-DEAS is a program developed by SDRC and can be used to generate grids and view results. The use of unstructured grids greatly enhances the ease with which a grid can be generated within a geometry. Tetrahedra can be used to fill any shape easily, whereas a structured grid would be difficult to generate inside some of the more complex heat sinks used today. Boundary conditions and initial conditions are applied to the grid for use by the finite volume solver. These boundary conditions range from fixed temperature to radiation and can be used to simulate the actual conditions experience by a material undergoing several forms of

heat transfer. The scheme was validated using known analytical solutions and simple geometries.

The algorithm developed employs a strategy which reduces the huge memory requirements usually required by vertex schemes. This memory requirement is due to the amount of neighboring geometry information required for the typical scheme, whereas the algorithm developed does not require such information. This makes the method very competitive for use on smaller computers which have limited memory resources. The scheme also uses an accumulation technique which is iterative. This is different from finite element schemes which usually use direct solution methods.

The accuracy of the numerical conduction solver was verified for both steady and unsteady problems with a variety of boundary conditions. The scheme is capable of generating very accurate solutions as demonstrated in the previous chapter.

Future work should include improvements in the radiation boundary conditions so that more factors which affect radiation can be introduced. These factors would include view factors and such things as allowing the emissivity to vary with direction and wavelength. A flow solver should be included so that the entire electronic system can be modeled. The use of a flow solver would make more accurate convection coefficients possible thereby enhancing the modeling process. With better modeling capabilities, better designs for enhancing heat transfer in electronic systems would allow cost effective air cooling to continue to be used even for the most advanced and demanding systems.

BIBLIOGRAPHY

[1] Bar-Cohen, Avram "State-of-the-Art and Trends in the Thermal Packaging of Electronic Equipment" *Journal of Electronic Packaging*, 114, (1992), 257-270.

[2] Ahmed, I., Krane, R. J. and Parsons, J. R. "A Preliminary Investigation of the Cooling of Electronic Components with Flat Plate Heat Sinks" *Journal of Electronic Packaging*, 116, (1994), 60-67.

[3] Wirtz, R. A. and Mathur, Ashok "Convective Heat Transfer Distribution on the Surface of an Electronic Package" *Journal of Electronic Packaging*, 116, (1994), 49-54.

[4] Keyhani, M., Prasad, V. and Cox, R. "An Experimental Study of Natural Convection in a Vertical Cavity with Discrete Heat Sources" *Journal of Heat Transfer*, 110, (1988), 616-624.

[5] Moffat, R. J. and Anderson, A. M. "Applying Heat Transfer Coefficient Data to Electronic Cooling" *Journal of Heat Transfer*, 112, (1990), 882-890.

[6] Guglielmini, G., Nannei, E. and Tanda, G. "Effect of Shrouding on Air Natural-Convection Heat Transfer from Staggered Vertical Fins" *Experimental Heat Transfer*, 2, (1989), 105-112.

[7] Incropera, F. P. "Convection Heat Transfer in Electronic Equipment Cooling" *Journal of Heat Transfer*, 110, (1988), 1097-1111.

[8] Xie, H. and Lee, Y. C. "Study of an Air Cooling Scheme for 3-D Packaging" *Journal of Electronic Packaging*, 116, (1994), 30-36.

[9] Mills, Anthony F. *Heat Transfer*. Homewood, Illinois: Irwin, 1992.

[10] Incropera, Frank P., DeWitt, David P. *Fundamentals of Heat and Mass Transfer 3rd*. New York: John Wiley and Sons, 1990.

[11] Anderson, Dale A., Tannehill, John C., Pletcher Richard H. *Computational Fluid Mechanics and Heat Transfer.* New York: Hemisphere Publishing Corporation, 1984.

[12] Barth, Timothy J. *On Unstructured Grids and Solvers* CFD Branch, NASA Ames Research Center, Moffett Field, CA.

[13] Özişik, M. Necati. *Heat Transfer, A Basic Approach.* New York: McGraw-Hill Book Company, 1985.

[14] Adams, J. Alan, Rogers, David F., *Computer-Aided Heat Transfer Analysis* New York, New York: McGraw-Hill Book Company, 1973.

[15] Arpaci, Vedat S. *Conduction Heat Transfer.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1966.

[16] Powers, David L. *Boundary Value Problems 3rd.* San Diego: Harcourt Brace Jovanovich Publishers, 1987.

[17] U. S. Department of Commerce. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables 7th.* National Bureau of Standards, 1968.

[18] Özişik, M. Necati. *Heat Conduction.* New York: John Wiley and Sons, 1980.

[19] Beyer, William H. *Standard Mathematical Tables 28th.* Boca Raton, Florida: CRC Press, Inc., 1987.

APPENDIX A. VOLUME OF ARBITRARY TETRAHEDRON

The volume of a tetrahedron is given by [19]:

$$V_e = \left| \frac{1}{6} \begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} \right| \tag{A.1}$$

Where x_1 is the x-coordinate of the first node and y_2 is the y-coordinate of the second node and so on. This can be shown to equal:

$$V_e = \left| \frac{1}{6} \left[\vec{r}_4 \cdot (\vec{r}_2 \times \vec{r}_3) \right] \right| \tag{A.2}$$

where

$$\begin{aligned} \vec{r}_2 &= (x_2 - x_1)\,\hat{\imath} + (y_2 - y_1)\,\hat{\jmath} + (z_2 - z_1)\,\hat{k} \\ \vec{r}_3 &= (x_3 - x_1)\,\hat{\imath} + (y_3 - y_1)\,\hat{\jmath} + (z_3 - z_1)\,\hat{k} \\ \vec{r}_4 &= (x_4 - x_1)\,\hat{\imath} + (y_4 - y_1)\,\hat{\jmath} + (z_4 - z_1)\,\hat{k} \end{aligned}$$

APPENDIX B. AREA OF A TRIANGLE OR QUADRILATERAL IN 3-D

The area of a triangle in 3-D space can be calculated as [19]:

$$A_t = \frac{1}{2}\left|(\vec{r}_1 \times \vec{r}_2)\right| \tag{B.1}$$

where

$$\begin{aligned}\vec{r}_1 &= (x_2 - x_1)\,\hat{\imath} + (y_2 - y_1)\,\hat{\jmath} + (z_2 - z_1)\,\hat{k} \\ \vec{r}_2 &= (x_3 - x_1)\,\hat{\imath} + (y_3 - y_1)\,\hat{\jmath} + (z_3 - z_1)\,\hat{k}\end{aligned}$$

Note that the corners of the triangle can be ordered either in a clockwise or counter-clockwise direction.

The area of a quadrilateral in 3-D space can be approximated as [19]:

$$A_q = \frac{1}{2}\left|(\vec{r}_1 \times \vec{r}_2)\right| \tag{B.2}$$

where

$$\begin{aligned}\vec{r}_1 &= (x_3 - x_1)\,\hat{\imath} + (y_3 - y_1)\,\hat{\jmath} + (z_3 - z_1)\,\hat{k} \\ \vec{r}_2 &= (x_4 - x_2)\,\hat{\imath} + (y_4 - y_2)\,\hat{\jmath} + (z_4 - z_2)\,\hat{k}\end{aligned}$$

Note that the corners of the quadrilateral must be ordered either in a clockwise or counter-clockwise direction. This is equivalent to saying that the area is equal to one-half the magnitude of the cross product of the vectors created by connecting opposite corners of a quadrilateral. The main assumption is that the points do not vary greatly from coplanar.